Duarte Miguel da Silva Arrobe

Licenciado em Ciências de Engenharia Eletrotécnica

e de Computadores

Modelo de Ocupação e Consumo Energético Habitacional

Dissertação para a obtenção do Grau de Mestre em

Engenharia Eletrotécnica e de Computadores

Orientador: Prof. Doutor Celson Pantoja Lima, UFOPA

Coorientador: Prof. Doutor João Francisco Alves Martins, FCT/UNL

Júri

Presidente: Prof. Doutor João Miguel Murta Pina, FCT/UNL

Arguente: Prof. Doutor Tiago Oliveira Machado de Figueiredo Cardoso, FCT/UNL

Vogais: Prof. Doutor Celson Pantoja Lima, UFOPA

Prof. Doutor João Francisco Alves Martins, FCT/UNL

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

novembro, 2013

Copyright

Dedicatória

À minha família, por me apoiar incondicionalmente,

e aos meus amigos mais próximos, sem os quais isto não seria possível.

Agradecimentos

Em primeiro lugar gostaria de agradecer à Universidade Nova de Lisboa e à Faculdade de Ciências de Tecnologias por me terem acolhido e disponibilizado as condições que me permitiram continuar os meus estudos. Foi esta segunda casa que me permitiu crescer, quer a nível pessoal através das amizades e laços criados, quer a nível intelectual, proporcionado pelos objetivos e metas a ultrapassar. Um obrigado aos professores que contribuíram para a minha educação e que ficarão para sempre na minha memória.

Todo este percurso não teria sido possível sem a ajuda, boa disposição e momentos de desespero passados ao lado dos amigos que fiz ao longo deste 6 anos. Um especial agradecimento a Pedro Leitão, José Vieira, Marisa Amaral, Rui Lopes, Mauro Dias, Telmo Ferraria, Pedro Arsénio, Nuno Vilhena e Bruno Vilhena. Aproveito para agradecer à minha classe de acrobática dos Bombeiros Voluntários/Junta Freguesia de Silves, por me terem sempre apoiado durante o meu percurso académico, não só graças às novas experiências e desafios, mas também através de toda a amizade. Um obrigado a todos, especialmente ao Emanuel Perpétuo, Ana Cardoso, João Simões, João Vasconcelos, Márcia Martins, David Eleutério, André Eleutério, Cláudia Sequeira, André Tinoco e Miguel Tinoco, nunca esquecendo os treinadores António Lourenço, Elsa Tomás e Isabel Frade.

Gostaria também de agradecer a todos os que contribuíram e me aturaram no decorrer desta dissertação, cuja colaboração, apoio e orientação foi essencial para o seu sucesso. Desta forma, gostaria de agradecer ao meu orientador, Professor Doutor Celson Lima, e ao meu coorientador, Professor Doutor João Martins, pela orientação, conhecimentos e disponibilidade. Fica aqui outro agradecimento a Pedro Arsénio, Nuno Vilhena e Bruno Vilhena que colaboraram e auxiliaram durante a validação da fase experimental, cuja participação foi essencial para a conclusão deste projeto. Um agradecimento especial para André Tinoco, não só por ter facultado material que me permitiu desenvolver este projeto, mas por todo o apoio e amizade ao longo destes anos.

A universidade pode ter sido a minha segunda casa, mas foi graças às minhas raízes e educação que cheguei onde estou hoje. Deste modo, um grande agradecimento à minha mãe, ao meu pai, à minha irmã, ao meu cunhado e ao recém-chegado, o meu sobrinho, por todo o amor, educação e suporte. Mesmo nos momentos mais difíceis, sempre estiveram lá para me apoiar no que fosse preciso, mesmo que isso colocasse as suas necessidades em segundo lugar. Agradeço-vos do fundo do coração.

Resumo

O paradigma energético atual tem como base o uso de fontes de energia não renováveis, mais nomeadamente os combustíveis fósseis, para satisfazer a procura energética global. O crescente nível de vida mundial causa um aumento da procura energética que, aliado ao paradigma atual, não permite a utilização dos recursos da Terra de forma sustentável e sem graves consequências para o ambiente.

Uma solução é a supressão e substituição das fontes energéticas poluentes por outras que não causem tanto impacto ecológico, como as fontes de energia solar e eólica. Outros métodos implicam o uso da tecnologia para melhorar a eficiência energética, quer ao nível de produção, transporte e consumo final, de modo a contrariar a crescente procura.

Ao nível do consumidor, cada vez mais se verifica a existência de aparelhos energeticamente mais eficientes. No entanto, a disponibilidade não implica necessariamente a aquisição. A sociedade tem de se consciencializar da realidade energética e efetuar mudanças comportamentais, de modo a reduzir o consumo a nível habitacional. Porém, para se mudarem hábitos, é necessário identificar quais os comportamentos antigos a melhorar.

Desta forma, com o intuito de fornecer aos ocupantes de uma habitação uma ferramenta que os auxilie no processo de redução de consumos energéticos a nível habitacional, através de mudanças comportamentais, a presente dissertação descreve o desenvolvimento de um sistema de monitoração energética habitacional, que recorre a serviços de localização para correlacionar o consumo energético do edifício com o posicionamento dos seus ocupantes. Deste modo, espera-se conseguir determinar perfis energéticos da habitação, das divisões e de cada utilizador, que serão utilizados para corrigir comportamentos energéticos menos corretos.

Com o objetivo de reaproveitar a infraestrutura da habitação, o sistema de localização recorre à tecnologia Wi-Fi, ao método de "impressão digital" e a variantes do algoritmo k-*NNS*, para garantir uma precisão por divisão superior a 85 %. O sistema de monitoração tentará associar em tempo real a informação energética da habitação com o posicionamento dos utilizadores, aplicando premissas simples que se espera serem capazes de categorizar corretamente o consumo.

Termos Chave: Eficiência Energética, Consciencialização Energética, Monitoração de Consumos, Localização de Pessoas a nível Habitacional, Algoritmo k-*NNS*, Correlação de Dados.

Abstract

The current energy paradigm allows to satisfy the global energy demand using mainly non-renewable energy sources, such as fossils fuels. Also, the rising living standards around the world increased this demand, jeopardizing the sustainability of Earth's resources. The environmental impact of energy production and consumption is another concern that must be properly handled.

One possible solution is the suppression and replacement of all fossil fuels by "cleaner" energy sources, like solar or wind power. Others means to counter the rising energy demand resorts to technology in order to improve energy efficiency on production, transportation and consumption chain levels.

On the consumer level, there has been a great improvement and availability of more efficient electrical equipment. Nevertheless, availability does not always implies equipment acquisition. In general, society must become aware of the current energy reality and change its behaviour to reduce energy consumption. However, someone can only change behaviours if he/she is aware of them and is also aware about the fact that they are not appropriate.

Therefore, in order to provide building's tenants a tool that helps them to reduce their energy bill through behavioural changes, this work focuses on the development of a home energy monitoring system that defines energy profiles for the house, its divisions and for each user. The latter results from the correlation between energy consumed and tenants location inside the house, provided by a location service. The information will give feedback about the energy consumption of a given house in a detailed way, helping the users to change their behaviours.

The proposed system must reuse the existing building infrastructure to provide a cheap and comfortable solution. This goal is achieved by a Wi-Fi location system that uses fingerprint method and k-NNS algorithm variants to locate tenants with accuracy above 85 % per division. The energy monitoring system will try to combine (in real time) the data from the house energy consumption with the given users location. In order to support that, only simple rules based on the users movement between divisions and consumption changes are implemented.

Key Words: Energy Efficiency, Energetic Awareness, Energy Monitoring, Home Tenants Location, k-NNS Algorithm, Data Correlation.

Índice de Matérias

Índice de Figuras

Índice de Tabelas

Lista de Siglas e Abreviaturas

A	Ámpere
W	Watt
Wh	Watt-hora
V	Volt
dB	decibel
dBm	decibel miliwatt
WiLOS	*Wi-Fi Locator and Occupancy System*
UEPOS	*User Energetic Profile Obtainer System*
SM	*Smart Meter*
EB	*Energy Box*
AP	*Access Point* / Ponto de Acesso
RSSI	*Received Signal Strength Indicator*
SGBD	Sistema de Gestão de Bases de Dados
BD	Base de Dados
EDP	Energias de Portugal
RS-485	Recommend Standard 485
UML	*Unified Modeling Language*
DER	Diagrama de Entidades e Relacionamentos
OECD	*Organisation for Economic Co-operation and Development*
Wi-Fi	*Wireless Fidelity*
GSM	*Global System For Mobile Communications*
GPS	*Global Positioning System*
RFID	*Radio-Frequency IDentifier*
IR	*InfraRed*

Capítulo 1: Introdução

A título introdutório, o enquadramento do trabalho a desenvolver é essencial para demonstrar ao leitor quais os problemas existentes e as suas possíveis soluções, servindo como fonte de motivação para este projeto. Seguidamente definem-se os objetivos a atingir com o desenvolvimento deste trabalho e, para finalizar, apresenta-se a estrutura sobre a qual se redige este documento.

1.1 Enquadramento e Motivação

Num mundo em constante evolução e crescimento, torna-se necessária a satisfação de todas as necessidades energéticas proporcionadas pela explosão tecnológica e pelo aumento da população mundial. Consequentemente, o elevado crescimento económico e aumento do nível de vida, especialmente no continente asiático (China e Índia), agrava a procura energética, originando mercados mais competitivos [1-3]. No século passado a solução para este problema seria simples. Se a procura energética é superior, então deve-se produzir em maior quantidade de modo a satisfazer todas as necessidades. Atualmente sabe-se que esta solução não é propriamente simples. O planeta Terra e os seus recursos têm de ser utilizados de forma sustentável [4]. Do mesmo modo, é necessário que o ser humano reduza o impacto ambiental negativo associado à produção de energia, mais propriamente através da redução das emissões de dióxido de carbono (CO2) [1,2,5-10].

A produção energética mundial foca-se primariamente em fontes de energia não renováveis, à qual corresponde 80 % de toda a procura energética a nível mundial (sem contar a energia nuclear) [1]. Estas fontes de energia são bastante versáteis e permitem obter grandes quantidades de energia por unidade de peso, o que as torna bastante atrativas para a produção energética [4]. No entanto, o processo de conversão matéria-energia é bastante poluente, libertando para a atmosfera gases de efeito de estufa, entre os quais o CO2 [1,2,4,11-13]. Desde a revolução industrial, tem-se vindo a verificar que o aumento das emissões destes gases reflete-se no aumento da temperatura global e consequentes agravamentos climatéricos. O degelo dos polos, a subida do nível do mar, a desertificação e ocorrência de fenómenos extraordinários com maior frequência são apenas alguns exemplos [9].

É claro que o controlo e manuseamento da energia foram decisivos para a evolução da sociedade [3]. No entanto, como se pode observar, o caminho atual da produção energética já não permite a sustentabilidade, devido à degradação do ecossistema e à extração de recursos irrecuperáveis. Sendo assim, outras soluções têm de ser exploradas para possibilitar a satisfação das necessidades energéticas atuais e futuras, sem comprometer a sustentabilidade [1,2,5-11].

A primeira resposta a este problema é a redução da utilização de combustíveis fósseis, através do recurso a energias renováveis e menos poluentes, como a energia solar, eólica, biomassa, etc. [1,2,5]. A nível empresarial, estas alternativas não se tornam atrativas devido ao menor rendimento energético e ao facto de a sua implementação exigir custos bastante superiores àqueles observados nas energias não renováveis [2,5]. Vários estudos demonstram que, até 2030, a dependência do ser humano pelos combustíveis fósseis continuará a ser elevada devido ao aumento da procura. No entanto, haverá uma maior tendência para o uso do gás natural como fonte primária, graças ao seu potencial e menor nível de poluição. As fontes de energia alternativas representarão uma força significativa na produção energética, especialmente na União Europeia, mas não constituirão a maioria mundial [5]. A figura 1.1 apresenta um gráfico que demonstra a previsão da procura energética para 2030, bem como quais as fontes energéticas que contribuem para a sua satisfação.

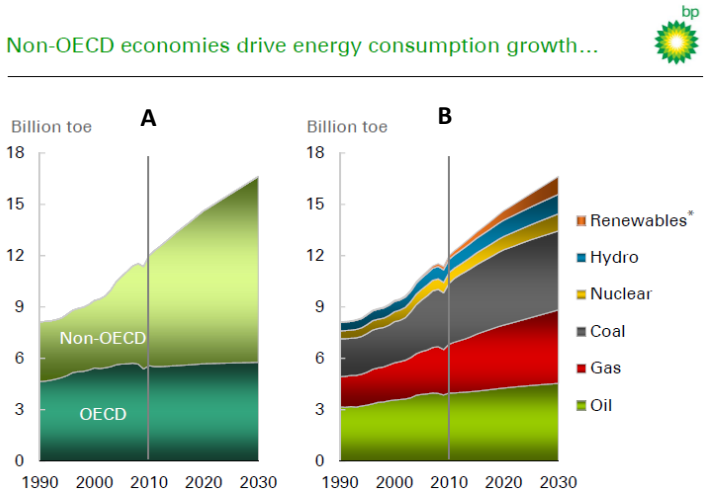


Figura 1.1 - (**A**): Procura energética mundial. (**B**) Contribuição de cada fonte energética para a satisfação da procura [5].

Outros métodos para um desenvolvimento energético sustentável recorrem ao avanço tecnológico. Todos os anos desenvolvem-se novas técnicas e/ou equipamentos que tornam os processos de produção, transporte e consumo de energia mais eficientes, económicos e menos poluentes [1,2,5,10,11]. A maioria dos estudos assume imediatamente que a eficiência energética é algo essencial e que terá de ocorrer para que se possa prever um cenário realista [1,2,5]. A figura 1.2 permite visualizar dois cenários para a evolução das emissões de CO2, um com as medidas atuais e outro com a implementação do cenário 450. Este último implica a imposição e execução de várias medidas para reduzir as emissões de CO2 a longo prazo. Verifica-se que, para além da adoção de energias alternativas, a eficiência terá um papel determinante nas emissões de CO2.

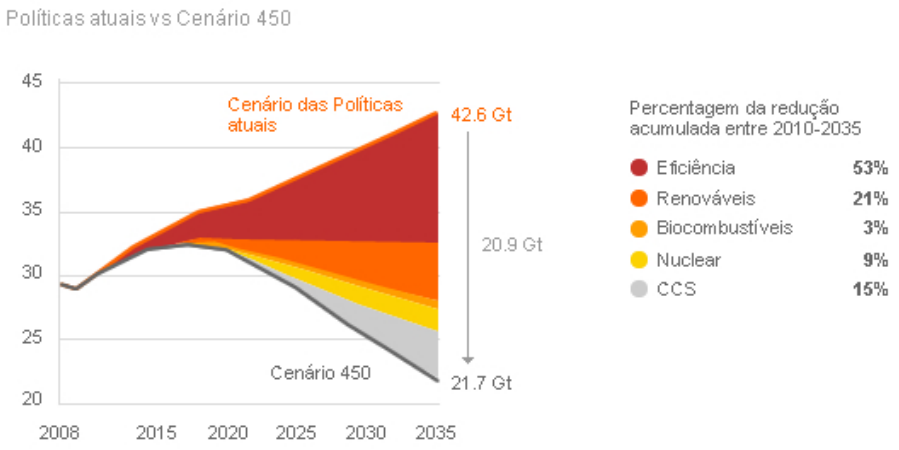


Figura 1.2 - Previsão da redução de emissões de CO2 para 2035 de acordo com 2 cenários [11].

Para além disso, o consumo energético é um ato de todos nós. Desta forma existe a responsabilidade social, onde a consciencialização energética e a redução de consumos podem ser preponderantes para o decréscimo da quantidade de energia utilizada [7,13,15-17]. Os termos eficiência energética e conservação energética começam a ser bastante comuns no dia-a-dia da sociedade. Ambos encontram-se intrinsecamente relacionados, mas possuem significados distintos.

A conservação energética consiste em todo o comportamento que permite a diminuição do consumo energético. Um exemplo simples é o apagar das luzes de uma divisão quando se deixa de usufruir dela. A eficiência energética traduz-se tecnologicamente através do uso de equipamentos que utilizam menos energia para desempenharem as mesmas funções. Como exemplo têm-se as lâmpadas incandescentes e as lâmpadas fluorescentes. Estas últimas permitem obter o mesmo nível de iluminação que as primeiras, só que com menor uso de energia. No entanto, quando se decide substituir uma lâmpada incandescente por uma lâmpada fluorescente, está-se perante um ato de conservação energética [8,9,17,18].

Num mundo comandado pelas telecomunicações e pela informação, cada vez mais as pessoas se apercebem da realidade energética e das suas consequências, e tentam fazer algo para modificar essa tendência. Surge assim a consciencialização energética. Uma pessoa dotada de consciencialização energética sabe de onde vem a energia, qual a quantidade utilizada, a sua aplicação e as consequências da sua aplicação. Esta origina a adoção de novos comportamentos, utilização de equipamentos mais eficientes, e até mesmo à implementação de energias alternativas na sua habitação [16,19].

Apesar da quantidade de informação disponível e da vontade das populações em fazer a diferença, a realidade é que continua a faltar informação. As pessoas não possuem conhecimento acerca dos seus hábitos energéticos, nem onde é que podem atuar para reduzir o consumo nas suas habitações. Para além disso, os mecanismos existentes no mercado que efetuam a aquisição desta

informação automaticamente são poucos e, na sua maioria, implicam algum investimento. Outro método para a obtenção dos níveis de consumo é através de inquéritos estatísticos. A sua análise permite concluir, por exemplo, que o consumo a nível residencial em Portugal diminuiu até 2001 mas este voltou a crescer até atingir os 17,7 % de toda a energia consumida em 2009 [20]. Uma análise mais detalhada permite diferenciar a aplicação desse consumo dentro de uma habitação [15].

No entanto isto não permite afirmar que todas as habitações devem possuir equipamentos de refrigeração mais eficientes. Cada caso é um caso, e numa determinada habitação realmente pode-se verificar essa situação, mas noutra o aumento do consumo pode simplesmente dever-se a uma má gestão do ar condicionado [15]. Estudos estatísticos em larga escala não permitem este tipo de detalhe, daí a necessidade da criação e implementação de mecanismos que permitam determinar em tempo real o consumo de uma habitação. Através dessa informação torna-se possível analisar o consumo diário, semanal, mensal e anual de uma habitação, de modo a detetar zonas de consumo inesperadas. Aqui surgem as oportunidades para reduzir o consumo energético através da mudança de comportamentos ou troca de equipamento.

Vários estudos demonstram que, a nível habitacional, é possível ter uma melhoria de eficiência energética entre os 30 - 35 %, através de alterações da infraestrutura de edifícios (janelas, isolamento, etc.) [12]. A adoção de novos comportamentos relacionados com a conservação energética (desligar luzes, ar condicionado, aparelhos em standby,…) pode originar reduções no consumo habitacional entre os 10 e os 25 % [11,13]. Assim, para além de se reduzir o consumo energético e as suas consequências, diminuem-se também os custos para o consumidor final. Nos dias de hoje toda a diminuição de custos é bem-vinda pois verifica-se o constante aumento das tarifas de eletricidade e gás [21,22].

Este trabalho foca-se na vertente da redução do consumo energético a nível residencial, de modo a fornecer uma ferramenta que possibilite a análise do consumo de uma habitação e dos seus ocupantes. O facto de a sociedade possuir cada vez mais consciência energética, associada à nova informação que possui ao seu dispor, permite que esta contribua para a diminuição da procura energética e das consequências negativas resultantes.

1.2 Objetivos

1.2.1 Objetivo Geral

Esta dissertação tem como objetivo o desenvolvimento de uma ferramenta que possibilite a monitoração do consumo energético de uma habitação e o traçar de perfis energéticos para as divisões e seus ocupantes. Estes perfis energéticos são modelos obtidos através da correlação do consumo energético global da habitação com o posicionamento dos seus ocupantes, fornecido por um serviço de localização. Através de cada perfil energético será possível apresentar estatísticas e informações relativas ao consumo energético verificado ao longo do tempo para cada elemento da habitação. Cabe depois ao utilizador, dotado de consciencialização energética ou não, a utilização da informação disponibilizada para inferir as áreas de atuação que permitem a diminuição do consumo (mudanças de comportamento ou aquisição de equipamentos mais eficientes).

1.2.2 Objetivos Específicos

O desenvolvimento desta ferramenta pressupõe o conhecimento do consumo energético da habitação em tempo real, bem como a localização de cada um dos seus ocupantes. Uma vez que se tratam de temáticas distintas, propõe-se a implementação de dois sistemas que interagem para oferecer ao utilizador todas as funcionalidades desejadas.

O sistema responsável pelo serviço de localização deve ser capaz de localizar todos os utilizadores dentro da habitação com uma precisão ao nível da divisão. Esta informação tem de ser disponibilizada em tempo real a outros sistemas que necessitem do posicionamento dos ocupantes, de modo a garantir as suas funcionalidades. Este serviço de localização é disponibilizado pelo WiLOS, *Wi-Fi Location and Occupancy System*, que recorre à rede Wi-Fi de uma habitação para determinar a localização dos seus utilizadores através dos dispositivos móveis em sua posse.

A obtenção do consumo energético é realizada através de um *Smart Meter* (SM) a instalar no quadro elétrico da habitação. Desta forma é possível obter várias informações referentes à rede elétrica da habitação. O HUEPS, *House and User Energy Profile System*, recolhe os dados disponibilizados pelo SM e efetua a sua correlação com o posicionamento dos inquilinos. Os perfis energéticos e a informação gerada são apresentados ao utilizador de uma forma intuitiva para que este determine com facilidade as zonas de consumo críticas que têm de ser solucionadas.

Desta forma, este projeto tem como objetivos específicos a definição e implementação dos 2 sistemas referidos, bem como a sua respetiva validação em cenários práticos reais. Deste modo pode-se determinar o desempenho do sistema global, WiLOS e HUEPS, e a sua aplicabilidade no quotidiano.

1.2.3 Estrutura do Documento

Este documento, para além do presente capítulo introdutório, encontra-se organizado nos seguintes capítulos:

- **Capítulo 2 – Sistemas de Monitoração e Gestão Energética**

Este capítulo dá a conhecer o trabalho realizado por outros autores na área de monitoração e gestão energética a nível habitacional. Para além da análise destes sistemas e conceitos servir como fonte de inspiração para o projeto em questão, também permite determinar a inovação associada ao sistema a desenvolver (HUEPS e WiLOS).

- **Capítulo 3 – Sistemas de Localização sem Fios**

O terceiro capítulo ilustra grande parte da investigação efetuada ao longo dos anos na área da localização de pessoas ou objetos através de tecnologias sem fios. Aqui são caracterizados os sistemas de localização quanto à sua tipologia, método de deteção, etc., bem como as diversas tecnologias utilizadas para a sua implementação.

- **Capítulo 4 – A Solução Proposta**

A solução proposta para o problema apresentado no capítulo 1 passa pela modelação e implementação de dois sistemas, o WiLOS e o HUEPS. Desta forma, o capítulo 4 divide-se em dois grandes subcapítulos, um dedicado ao WiLOS e outro ao HUEPS. Em cada um deles são apresentadas e explicadas todas as decisões tomadas, bem como os diferentes módulos que constituem cada um dos sistemas.

- **Capítulo 5 – Validação e Resultados Experimentais**

Neste capítulo apresentam-se vários exemplos da aplicação do WiLOS e do HUEPS em cenários reais. Deste modo é possível inferir o desempenho de cada um dos sistemas, analisar os métodos implementados e validar os resultados obtidos.

- **Capítulo 6 – Conclusão e Trabalhos Futuros**

Para finalizar, os resultados obtidos permitem concluir sobre a eficácia dos sistemas desenvolvidos, bem como a possibilidade da sua utilização no quotidiano. A discussão e crítica do trabalho desenvolvido permite propor melhorias e trabalhos futuros. As contribuições para a investigação proporcionadas por este projeto também são destacadas neste capítulo.

Capítulo 2: Sistemas de Monitoração e Gestão Energética

Os sistemas de gestão e monitoração energética habitacionais, de acordo com as suas funcionalidades, podem ser divididos em 3 grupos: sistemas de monitoração energética (SME); sistemas de monitoração e gestão energética (SMGE) e sistemas *Smart Home* (SSH).

Os SME são sistemas que recolhem, processam e apresentam informação habitacional ao utilizador. Os SMGE, para além de incorporarem as funcionalidades dos SME, possibilitam a interação com os equipamentos elétricos, de modo a implementar novos serviços. Por fim, os SSH são o pináculo dos sistemas de gestão e monitoração, utilizando o conceito de *Internet of Things* para oferecer todos os mecanismos supracitados aos ocupantes de uma habitação, em qualquer lugar. A estruturação por camadas da figura 2.1 ilustra a relação existente entre os grupos mencionados, bem como as funcionalidades disponibilizadas.

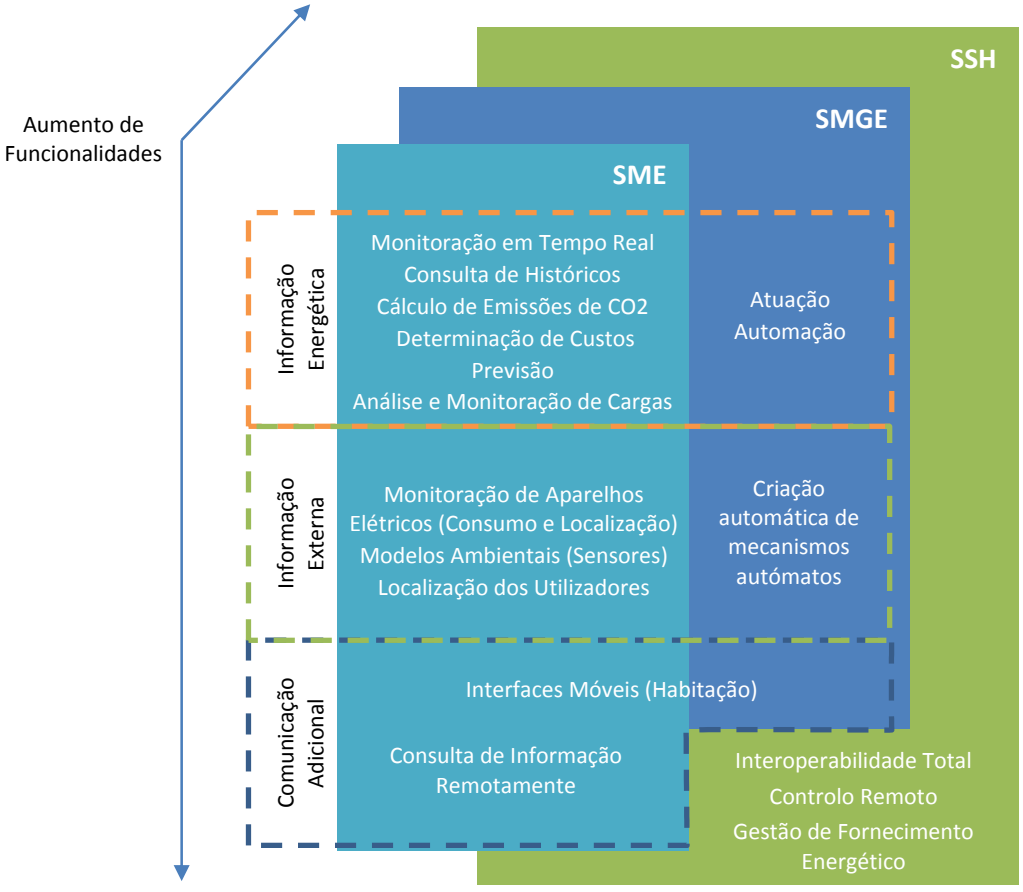


Figura 2.1 - Subdivisão dos sistemas de gestão e monitoração energética habitacionais em 3 grupos: SME; SMGE e SSH.

As pessoas, para alterarem os seus comportamentos energéticos, necessitam de informação referente aos seus hábitos energéticos atuais, ou de algo semelhante que sirva como ponto de referência [23-27]. A implementação de um SME permite a recolha do consumo energético de uma habitação em tempo real, de modo a mostrar aos seus ocupantes o consumo energético atual [23-28]. A sua configuração mais simples consiste na implementação de um dispositivo composto por um módulo de aquisição de dados, uma unidade de processamento e um módulo de interface com o utilizador, nas proximidades do fornecimento energético da habitação (e.g., quadro elétrico). Desta forma garante-se que o SME recolhe os valores de consumo energético referentes a toda a habitação, como ilustrado na figura 2.2.

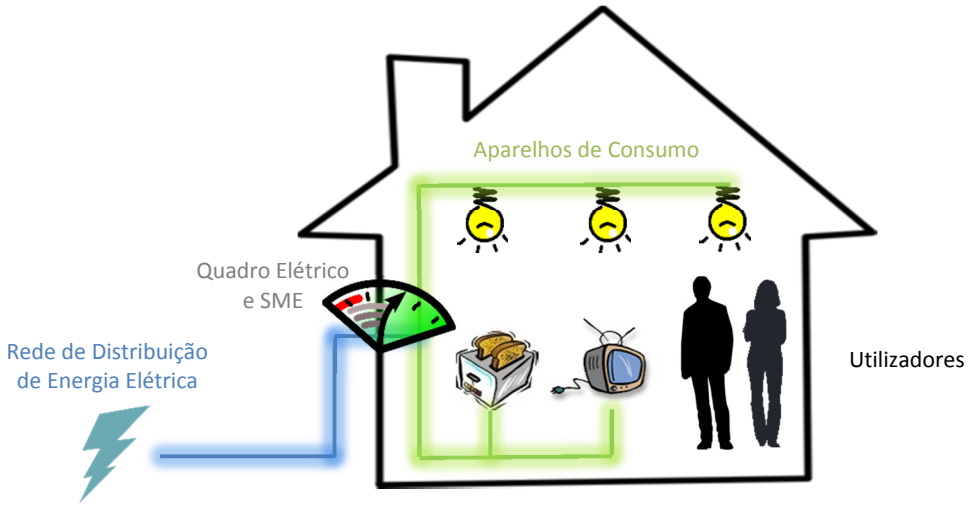


Figura 2.2 - Configuração típica de um SME com o dispositivo instalado nas proximidades do quadro elétrico da habitação.

O armazenamento de toda a informação recolhida permite mais tarde a sua organização e visualização sob a forma de históricos de consumo energético. Estes permitem analisar zonas críticas de consumo, isto é, zonas com consumos elevados não justificados ou com ocorrências de consumo energético em alturas do dia em que a habitação se encontra vazia, fornecendo aos seus ocupantes oportunidades para reduzir o consumo energético [25,26-29]. Por norma, a maioria dos SME também oferecem mecanismos para cálculo de emissões de CO2 e de custos associados à quantidade de energia consumida [25,26]. O primeiro procura apelar à consciencialização ecológica do utilizador, enquanto a segunda fornece uma ferramenta mais atrativa do ponto de vista financeiro. Afinal, a informação relativa aos custos atuais reflete-se na fatura final, cujo valor o utilizador deseja sempre que seja o mais baixo possível [30].

Vários estudos demonstram que a aplicação de um SME com estas características já oferece aos seus utilizadores mecanismos para se obter uma melhoria do consumo energético, através da adoção de novos comportamentos [30]. Existem até casos em que este tipo de sistemas é interpretado como

um jogo, onde o objetivo é tentar alcançar um consumo nulo, determinando quais os aparelhos que produzem consumo indesejado quando não se encontram em funcionamento [25]. No entanto, a informação disponibilizada continua a ser pouco detalhada, uma vez que apenas incide sobre o consumo energético habitacional. Mesmo que se apliquem algoritmos de correlação de consumos passados com consumos correntes, de modo a estimar-se o consumo total no final do mês e respetivos custos [23,31], continua-se a fornecer ao utilizador uma informação muito generalizada.

A obtenção de dados mais detalhados referentes aos consumos ocorridos na habitação através de SME, sem recurso a informação de outra natureza que não energética, pode ser realizada através da análise da curva de consumo energético da habitação. Uma vez que as variações energéticas são causadas pelos aparelhos elétricos existentes na rede, e que cada aparelho possui o seu próprio comportamento energético (figura 2.3), é possível, até certo ponto, identificar cada um dos aparelhos que se encontra em funcionamento. Este processo denomina-se deteção ou desagregação de cargas [32-35]. O facto de não se recorrer ao auxílio de sensores adicionais ao longo da habitação para determinar o aparelho ativo, torna o método não intrusivo e mais apelativo em termos de custos associados à infraestrutura do SME [32,35]. Do ponto de vista tecnológico, a deteção de cargas é um problema mais complexo.

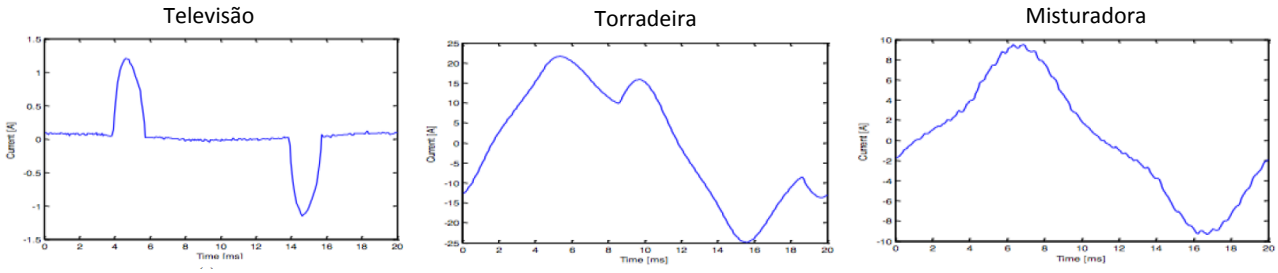


Figura 2.3 - Curvas de comportamento energético de diversos aparelhos elétricos, corrente [A] ao longo do tempo [ms].

Com esta informação, os ocupantes já conseguem detalhar o consumo energético habitacional, categorizando-o por aparelhos elétricos [23-25,36]. Desta forma, as zonas críticas de consumo já possuem um certo número de aparelhos associados, o que facilita a sua análise e diminui a área de atuação. Também é possível utilizar o comportamento energético de um equipamento elétrico para determinar se este continua a apresentar um bom funcionamento ou se é necessário realizar a sua manutenção [34]. Um aparelho avariado ou com fraca manutenção obriga a um maior consumo elétrico e, em alguns casos, durante um período de funcionamento maior do que o habitual. Se o SME não possuir mecanismos para detetar estas anomalias, cabe ao utilizador determinar se algum aparelho apresenta consumos fora do normal.

No entanto, muitos SME não usufruem somente da informação energética recolhida num só ponto da habitação. A aquisição de informação adicional ocorre através de uma rede de sensores, implementada ao longo de habitação, que pode ser associada aos aparelhos elétricos existentes [23-25,31,34]. Estes sensores, para além de recolherem o consumo parcial efetuado por cada aparelho, também podem verificar as condições ambientais do edifício, como a luminosidade, temperatura e humidade, de modo a obter-se um modelo do estado da habitação [27]. A integração e comunicação destes sensores obedece a uma das muitas estruturas e protocolos de comunicação desenvolvidos para o efeito. Destaca-se a comunicação através de *Power Line Communication* (PLC) [37,41] e Wi-Fi (norma IEEE 802.11) [38,41], que tentam reaproveitar a infraestrutura existente, e o ZigBee (norma IEEE 802.15.4) [39,41] e Bluetooth (IEEE 802.15) [39-41] que são utilizados pelo seu baixo custo e consumo energético.

A utilização de sensores nos equipamentos elétricos da habitação oferece mais algumas vantagens que o processo de desagregação de cargas. Para além de ser possível determinar qual é o aparelho associado ao sensor, através da análise da sua curva de consumo recorrendo ao reconhecimento de cargas, a aplicação de métodos de localização permite obter o posicionamento exato de cada aparelho na habitação [29]. Esta informação pode ser associada aos seus períodos de funcionamento, ou até a outros aparelhos manuseados em instantes temporais bastante próximos [29]. Deste modo consegue-se inferir o consumo associado a uma divisão, agrupar aparelhos por divisão ou determinar rotinas e comportamentos energéticos dos utilizadores. Tudo com o intuito de categorizar e detalhar a informação relacionada com o consumo energético habitacional.

Note-se que só se estão a admitir processos de aquisição de informação mais sofisticados. Desta forma, não se abordam as temáticas em que um SME oferece funcionalidades para um ocupante definir a sua rotina no sistema (levantar, ir para o trabalho, cozinhar, etc.), ou os aparelhos que fazem parte da habitação, através da sua localização e consumo. O objetivo é destacar sistemas que efetuam correlação de informação, de modo a oferecer ao utilizador maior conforto e informação detalhada acerca do consumo habitacional.

A incorporação de informação relativa à ocupação de um edifício e localização dos seus inquilinos, permite inferir mais facilmente os períodos de consumo não justificado, como luzes ligadas numa divisão vazia, e alertar os utilizadores para essas ocorrências em tempo real [29]. Estas funcionalidades garantem ao utilizador a capacidade de reduzir o consumo excessivo no instante em que ele ocorre. A associação de toda a informação, posicionamento dos utilizadores com a ocorrência de variações energéticas causadas pelos aparelhos localizados em certas divisões, possibilita a determinação das rotinas dos ocupantes, através dos seus hábitos e preferências energéticas [24,25,29]. Estes podem ser utilizados para várias finalidades, sendo uma delas a melhoria dos modelos de consumo energético

de um utilizador. Estes podem ser utilizados para prever o consumo futuro de um utilizador ou indicar os hábitos energéticos que devem ser melhorados.

Através dos mecanismos e funcionalidades supracitados, os SME conseguem oferecer aos ocupantes de uma habitação toda a informação referente ao seu consumo energético, com um nível de detalhe relativamente pormenorizado. Estes permitem a redução e melhor utilização da energia elétrica de uma habitação, caso o utilizador consulte a informação e aja conforme a necessidade. No entanto, um SME localizado no quadro ou noutro local pré-determinado da habitação, torna-se pouco vantajoso para quem pretenda usufruir dos seus serviços, tendo que se deslocar ao local para verificar algum aviso ou verificar o consumo energético atual.

A introdução de tecnologias de rede e internet nos SME permite a disponibilização da sua informação através de várias interfaces com o utilizador, incluindo computadores, telas digitais e dispositivos móveis [36,40]. O acesso remoto aos dados do consumo energético habitacional, e outros, possibilita ao utilizador visualizar, em qualquer lugar, o estado da sua habitação (figura 2.4) [24,26,28]. A disponibilização desta informação para outos sistemas ou entidades permite a existência de serviços de mais alto nível, como as *Smart Grids*, onde a perceção em tempo real do consumo de cada habitação, e dos micro-produtores existentes na rede de distribuição, garante uma maior flexibilidade, gestão e eficiência da rede. Permite também a aplicação de novos métodos de taxação, mais económicos e justos para o utilizador, através da criação de um mercado energético mais liberalizado [41,42].

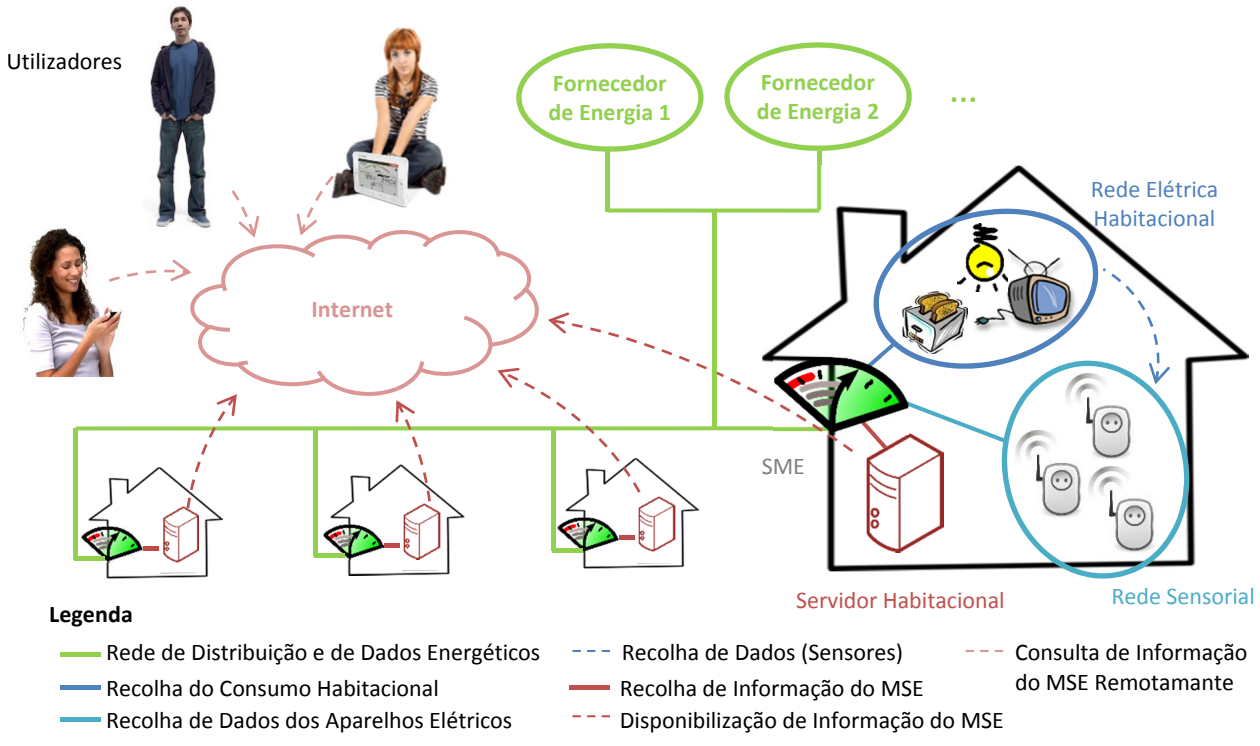


Figura 2.4 - Configuração de um SME, dotado de redes de sensores e acesso remoto, aplicado a uma zona residencial.

Como a necessidade de conforto para o ser humano é essencial, somente a disponibilização e consulta da informação dos SME, de modo a indicar zonas de intervenção e originar mudanças comportamentais, nem sempre é o mais cómodo ou oportuno para o utilizador. Porém, se estes sistemas conseguissem atuar sob a habitação, em aparelhos específicos e a partir de qualquer ponto do edifício, o nível de comodidade aumentaria consideravelmente. Surgem assim os SMGE, que oferecem todas as funcionalidades de um SME e a capacidade de atuar sobre a habitação [28,40]. Se um SMGE utilizar a informação recolhida da habitação para gerir automaticamente todas as operações de excesso de consumo injustificado, o próprio SMGE deixa de ser apenas uma ferramenta auxiliar no processo de redução de consumo e mudanças comportamentais, e passa a ser algo que contribuí, em tempo real, para a poupança energética [37]. Não se recorre ao termo de sistema de domótica, uma vez que o processo de gestão e automação de um edifício não implica necessariamente o conhecimento do consumo energético da habitação, ou o conceito de consciencialização energética [40]. Um sistema de domótica pode ser utilizado somente para oferecer uma maior comodidade ao utilizador final, podendo ou não usufruir dos mecanismos existentes nos SME.

Por fim, a disponibilização de todas as funcionalidades de um SMGE remotamente, através da Internet, aplica o conceito da *Internet of Things* (*IoT*) à habitação e a todos os aparelhos existentes na sua rede elétrica. O objetivo da *IoT* é interligar todos os aparelhos à internet, de modo a que as pessoas possam consultar, gerir e operar qualquer aparelho em qualquer lugar. A disponibilização e associação de tanta informação, através de mecanismos de interoperabilidade, permite a criação de novos serviços, de preferência mais autónomos e inteligentes, que garantem o aumento de conforto, eficiência e produtividade do ser humano [39,43].

Desta forma, a disponibilização de informação, o controlo de equipamentos e a automação de uma habitação, associados às tecnologias utilizadas para realizar a sua interligação, origina um novo conceito, a *Smart Home* ou "casa inteligente". Um sistema *Smart Home* (SSH) também implementa mecanismos que permitem gerir e monitorar a produção energética a nível habitacional, através de energias renováveis, bem como garantir o fornecimento energético às *Smart Grids*. É portanto um conceito tecnológico a pensar no futuro, onde todas as habitações terão mecanismos para garantir a eficiência e produção energética, controlo automático de consumos, gestão e monitoração dos seus recursos em qualquer lugar do mundo. Deste modo, oferece-se às pessoas uma ferramenta que tenta utilizar a energia de uma forma mais consciente e sustentável, proporcionando um maior conforto. [28,39,43].

A figura 2.5 ilustra todos os desafios associados à implementação de cada um destes sistemas, SME, SMGE e SSH, que têm de ser ultrapassados durante o seu desenvolvimento.

Figura 2.5 - Problemas associados ao desenvolvimento de sistemas de gestão e monitoração energética habitacionais.

De todos os sistemas de gestão e monitoração energética abordados, o HUEPS enquadra-se na categoria dos SME que recorrem a informação adicional, nomeadamente à localização de utilizadores. Espera-se conseguir correlacionar o consumo energético verificado na habitação com o posicionamento de cada utilizador, de modo a definir perfis energéticos para as divisões e para cada utilizador. Os trabalhos desenvolvidos em [26,28,29] utilizam serviços de localização neste tipo de sistemas com diferentes propósitos.

Em [26] é apresentado um SME que incorpora tecnologias de computação ubíqua e acesso remoto, para contextualizar a informação energética apresentada ao utilizador. O serviço de localização desenvolvido recorre ao GPS (*Global Positioning System*) e à proximidade de um utilizador com a rede Wi-Fi para determinar se este se encontra, ou não, dentro da sua habitação durante os picos de maior consumo. Os autores em [28] desenvolveram um SSH com serviços básicos de monitoração, controlo e automação, que consegue utilizar a localização do utilizador para desencadear

eventos pré-programados no SSH. A obtenção dos perfis e rotinas dos utilizadores e dos aparelhos energéticos não é realizada de forma automática, obrigando a inserção destes dados no sistema. Os eventos são configurados do mesmo modo, onde o utilizador define condições como “Ligar o ar condicionado quando estiver a 500 m do trabalho”. Desta forma, o sistema deteta o evento e atua automaticamente, de modo a garantir a comodidade do utilizador.

Por fim, o SME implementado em [29] é composto por uma rede de sensores (tomadas "inteligentes") que verificam o consumo energético dos vários aparelhos elétricos da habitação. Este recorre a um sistema de posicionamento composto por sensores de pressão e tecnologia RFID (*Radio-Frequency IDentification*) para localizar tanto utilizadores como aparelhos. O objetivo é associar um utilizador a um aparelho recentemente utilizado de modo a que, caso um utilizador se desloque para outra divisão e o aparelho continue ativo, este seja alertado para a ocorrência de desperdício energético.

De modo a perceber melhor o funcionamento e os tipos de sistemas de localização existentes e que podem ser aplicados no contexto deste trabalho, o capítulo 3 aborda a temática dos sistemas de localização sem fios.

Capítulo 3: Sistemas de Localização sem Fios

3.1 Caracterização dos Sistemas de Localização

Ao longo das duas últimas décadas, os sistemas de estimação de localização têm sido alvo de investigação por diversas entidades. Várias são as tecnologias utilizadas para o desenvolvimento destes sistemas, cujo objetivo é detetar um objeto, ou pessoa, que se encontre no alcance do sistema. Esta deteção pode ser mais ou menos precisa consoante a necessidade e a aplicação final do sistema de localização. No entanto, quais foram as causas para o interesse neste tipo de sistemas?

Um dos principais impulsionadores ocorreu em 1996, quando a *US Federal Communications Commision* propôs o mandato "E911". Este estabeleceu um prazo (até 2001) para que as operadoras norte-americanas desenvolvessem e aplicassem um sistema de localização automática. O objetivo seria garantir a localização de um indivíduo após a realização de uma chamada de emergência através de um telemóvel com uma precisão de 125 m, 67 % das vezes [44,45]. Inicialmente considerou-se que o *Global Positioning System* (GPS) seria a solução ideal, devido ao seu baixo custo, consumo e elevada precisão. No entanto, o seu desempenho em áreas suburbanas e no interior de edifícios não permite que esta tecnologia seja utilizada em todas as situações [44-46]. Desta forma começou a investigação em torno de sistemas de localização alternativos que fossem capazes de substituir o GPS ou de o complementar.

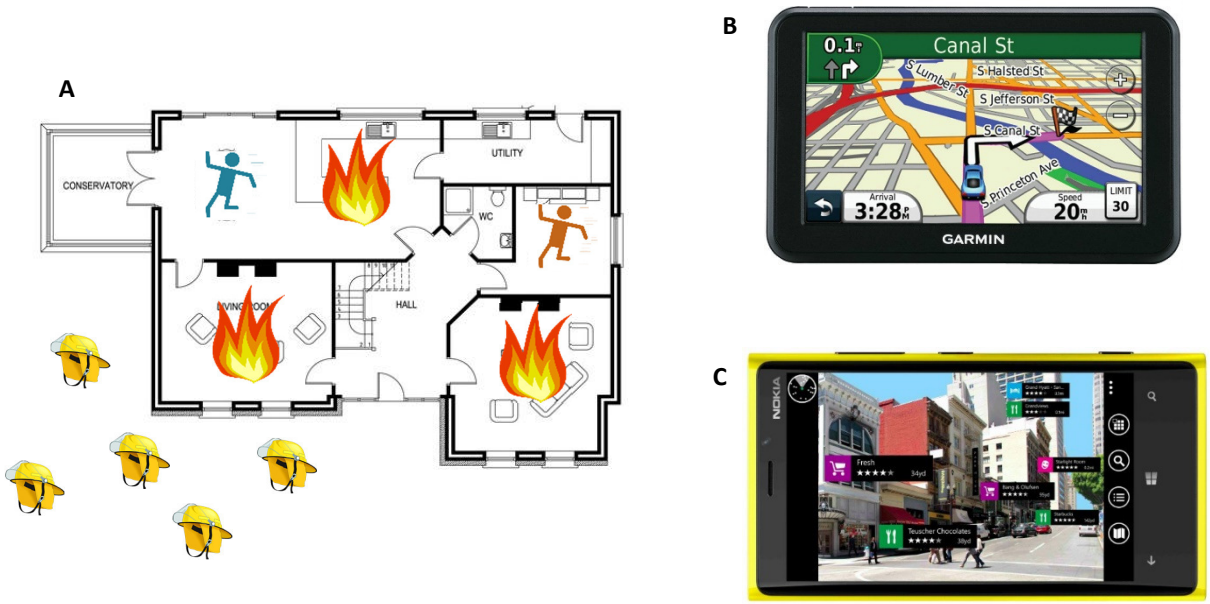


Figura 3.1 - (**A**): Socorro aos ocupantes durante um incêndio. (**B**): Auxílio à navegação em tempo real (GPS). (**C**): Outros níveis de interatividade com a realidade através de localização e de processamento de imagem.

Com o passar do tempo, as operadoras e outras entidades verificaram que os sistemas de localização possuíam um potencial maior do que o considerado inicialmente. Para além dos serviços de emergência, a existência de outros serviços que dependem da localização de um dispositivo móvel são uma mais-valia [44]. Nasce assim a necessidade de fornecer serviços inteligentes que utilizam a localização e a contextualização como foco principal [45]. Estes possuem inúmeros benefícios, permitindo implementar novos mecanismos de segurança, controlo de inventário, gestão de eventos e coordenação de forças de socorro [46-48]. Para além disso, serviços mais específicos como o redireccionamento de chamadas, visitas guiadas a museus, bem como sistemas de navegação e orientação de utilizadores em centros comerciais e livrarias são mecanismos que oferecem um maior comodismo no dia-a-dia [46,49]. Algumas destas vantagens são destacadas na figura 3.1.

Outras vantagens relacionadas com o conforto do utilizador passam pelas próprias aplicações informáticas, oferecendo um novo nível de interatividade. Consoante a localização atual, os menus adaptam-se de modo a apresentar a informação contextualizada, o que reduz o número de opções visualizadas e facilita o processo de navegação através dos menus de uma aplicação [46]. Ao nível das operadoras, a localização de um utilizador permite otimizar e melhorar a rede. O processo de *caching* é um dos exemplos que permite melhorar o serviço oferecido, colocando a informação passível de ser consultada por um utilizador numa localidade próxima deste, o que resulta num aumento da velocidade de acesso [46]. O facto de se saber onde cada utilizador se encontra pode originar novos algoritmos de encaminhamento de dados, permitindo o aumento do desempenho das redes através de uma melhor gestão [46,50].

Durante a fase de projeto de um sistema de localização, vários elementos têm de ser considerados de modo a definir o sistema final. Um sistema de localização pode ser categorizado de acordo com a sua área de intervenção, infraestrutura tecnológica, interações entre atores do sistema e método de localização. Em primeiro lugar, tem de se definir uma zona de intervenção para a implementação do sistema de localização. Esta vai condicionar o leque de soluções tecnológicas à disposição para o desenho do sistema. Desta forma, um sistema de localização pode ser caracterizado como um sistema "interior", "exterior" ou ambos ("interior/exterior").

Zona de Intervenção

Um sistema interior, como o nome indica, é um sistema desenvolvido com o objetivo de atuar no interior de edifícios, ultrapassando as dificuldades apresentadas por estes ambientes, nomeadamente paredes e obstáculos dinâmicos em espaços relativamente pequenos. Um sistema exterior é o conceito oposto, sendo a sua zona de atuação o exterior dos edifícios. Zonas urbanas, áreas rurais de campo aberto, zonas montanhosas, litoral e alto mar são alguns exemplos de ambientes considerados

exteriores [45,49]. Por último, os sistemas "interior/exterior" devem ser capazes de operar em ambos os ambientes, através da superação da dificuldade associada a ambos os casos. Desta forma representa um sistema capaz de atuar em qualquer ambiente.

Infraestrutura

O sistema em questão terá que possuir uma infraestrutura tecnológica que permita inferir o posicionamento dos indivíduos dentro da área de intervenção. Desta forma, um sistema pode optar por utilizar uma infraestrutura já existente ou a criação de uma nova [50]. Os sistemas de localização de infraestrutura "existente" originam uma solução maioritariamente focada no desenvolvimento de *software* com recurso a algoritmia. Estes são normalmente implementados sobre uma rede de comunicação já instalada [50], o que permite usufruir da própria capacidade de comunicação da rede e evita custos adicionais associados à alteração/criação de uma nova infraestrutura. Por outro lado, a criação ou expansão de uma infraestrutura pode ser necessária. Esta abordagem ocorre caso a tecnologia utilizada não se encontre disponível, ou caso a infraestrutura atual não seja capaz de obter a informação necessária para o serviço de localização. A este tipo de sistemas denomina-se sistemas de infraestrutura "dedicada".

Interações entre atores do sistema

Dentro de uma infraestrutura existem vários intervenientes que colaboram para cumprir os objetivos propostos pelo sistema. Por norma podem existir 3 tipos de elementos de maior importância dentro de uma infraestrutura: os dispositivos móveis ou identificadores; os pontos de referência e a estação central.

Os dispositivos móveis possibilitam a identificação do utilizador em toda a zona de intervenção, uma vez que se encontram na sua posse e variam a sua localização em conjunto com o utilizador ou o objeto a localizar. Os pontos de referência podem ser emissores ou recetores de sinais, instalados ao longo da infraestrutura do sistema, que permitem o envio ou receção de informação referente à localização dos dispositivos móveis. De acordo com a tecnologia aplicada, estes podem ser considerados pontos de acesso (e.g., redes Wi-Fi), estações bases (redes GSM, 3G, etc.) ou apenas placas de sensores que colaboram no processo de localização. Normalmente possuem mecanismos de comunicação para permitir o envio de dados através da infraestrutura. A maioria dos sistemas de localização possui uma estação central que fica responsável por gerir todos os atores e a comunicação dentro da infraestrutura. Como tem acesso a toda a informação existente, é comum que seja esta a estimar o posicionamento dos dispositivos. Note-se que a existência de um dispositivo auxiliar para identificar um utilizador, ou efetuar a localização de um determinado objeto, não é obrigatória. Deste

modo, os pontos de referência e a estação base, se existir, recolhem a informação necessária para se efetuar a localização a partir da localização real do próprio objeto ou pessoa a localizar.

Conforme o papel desempenhado por cada um destes elementos e as suas interações dentro da topologia da rede, os sistemas de localização podem ser classificados quanto ao tipo de posicionamento efetuado. Desta forma existem os sistemas de "posicionamento remoto direto", "posicionamento remoto indireto", "auto posicionamento direto" e "auto posicionamento indireto" [50].

No primeiro caso, o dispositivo móvel é a entidade que emite sinais ao longo da zona de implementação (transmissor). Os pontos de referência encontram-se instalados em posições fixas, pré-determinadas e efetuam o papel de recetores, de modo a medir o sinal enviado pelo dispositivo móvel. De seguida, os vários pontos de referência enviam a informação recolhida para uma estação central, que procede à sua análise e determina a localização do dispositivo, como ilustrado na figura 3.2. No posicionamento remoto indireto o dispositivo móvel e os pontos de referência invertem os papéis. Assim, o dispositivo móvel age como um recetor e recolhe os sinais enviados pelos pontos de referência. Estes dados são reencaminhados para uma estação central, através de um canal de comunicação dedicado, que infere a posição do identificador [50].

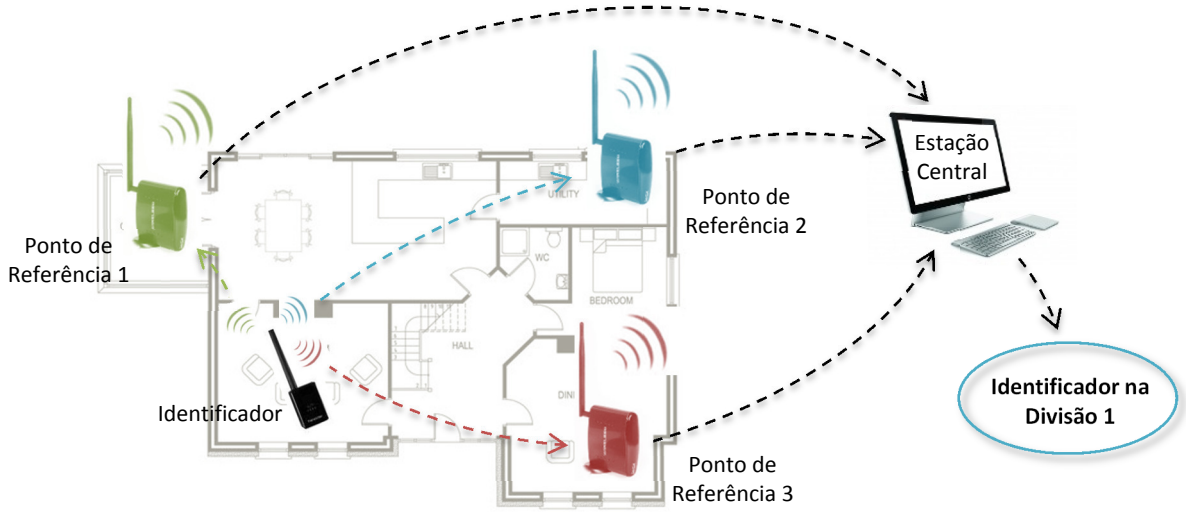


Figura 3.2 - Tipologia de um sistema de localização de posicionamento remoto direto e interações existentes.

Um sistema de auto posicionamento direto implica que seja o próprio dispositivo móvel a determinar a sua própria localização. Para tal, vários pontos de referência funcionam como transmissores colocados em locais conhecidos, e enviam periodicamente diversos sinais. Estes são captados pelo dispositivo móvel que analisa a informação recolhida e deduz a sua posição (figura 3.3). No auto posicionamento indireto a informação que permite determinar a localização do dispositivo

móvel deixa de ser obtida pelo próprio dispositivo. Desta forma, os pontos de referência recolhem os dados necessários para se efetuar o posicionamento e reencaminham a informação para o dispositivo móvel, que acaba por determinar a sua localização. Esta operação pode ser gerida por uma estação central ou não [50].

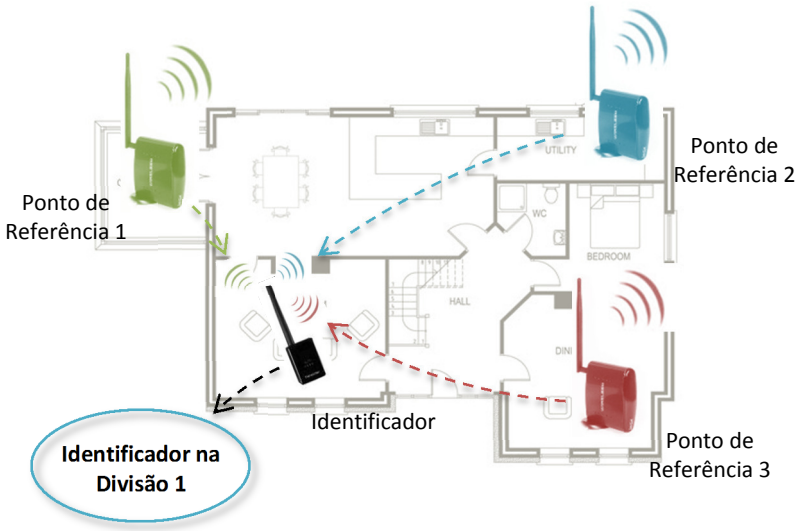


Figura 3.3 - Tipologia de um sistema de localização de auto posicionamento direto e interações existentes.

Método de Localização

A escolha da tipologia da infraestrutura e o papel de cada interveniente do sistema estão intrinsecamente relacionados com o método escolhido para se estimar a localização de um determinado dispositivo. Os métodos de localização utilizados dividem-se em 5 categorias: célula [44,48,51]; trilateração [47,48]; multilateração [47,51]; triangulação [48,50]; e análise da zona de intervenção [44,46-48,50].

Sempre que um dispositivo móvel se encontre dentro do alcance de um determinado ponto de referência, este pode estabelecer uma ligação com o mesmo. A localização atual do dispositivo passa a ser representada pelo ponto de referência, bem como por toda a área definida pelo seu alcance. Como diversos autores designam esta área de "célula", surge o termo "localização por célula". Trata-se de uma técnica simples e fácil de implementar, mas que depende da dimensão da célula e do algoritmo utilizado para determinar o ponto de referência mais próximo. Esta dependência faz com que este método não seja preciso, uma vez que o alcance de um ponto de referência pode variar entre dezenas de metros a alguns quilómetros [44,46,51]. A figura 3.4 ilustra o modo de funcionamento deste método.

Figura 3.4 - Exemplificação do método de localização por célula.

O método de trilateração utiliza o modelo de propagação do sinal (estimação da velocidade) e o seu tempo de propagação (*TOF*, *Time of Flight* ou *TOA*, *Time of Arrival*) para determinar a distância relativa existente entre um emissor e um recetor. A utilização de 3 pontos de referência e as distâncias a eles associadas permite estimar a localização de um dispositivo com "precisão". Esta precisão depende do modelo de propagação escolhido, bem como da existência de mecanismos de sincronismo entre emissores e recetores, de modo a garantir o cálculo correto da distância percorrida pelo sinal [52]. A existência de uma simples discrepância de um microssegundo origina um erro de precisão no sistema substancial [44,47]. Caso se saiba a localização dos pontos de referência dentro da área de implementação, torna-se possível determinar a posição absoluta de um dispositivo.

A multilateração é uma variação da trilateração que deixa de recorrer diretamente aos valores de *TOA*. Este método utiliza a diferença dos tempos de chegada do sinal (*TDOA*, *Time Difference of Arrival*) a diferentes pontos de referência espacialmente dispersos para efetuar a localização. Um sinal emitido demora mais tempo a percorrer uma distância maior logo, pontos de referência em localizações diferentes, recebem o sinal em instantes diferentes. A diferença temporal evidenciada permite definir uma hipérbole que representa as localizações possíveis para um dispositivo a localizar. Vários pontos de referência permitem traçar várias hipérboles cuja interseção garante a localização "exata" do dipositivo. Tem a vantagem de não necessitar de mecanismos de sincronismo por parte do elemento a localizar (emissor), lidando apenas com as diferenças temporais nos pontos de referência [44,47]. Na figura 3.5 representa-se a utilização deste método para localizar um avião comercial, a elevada altitude, através de vários pontos de referência próximos de uma torre de controlo.

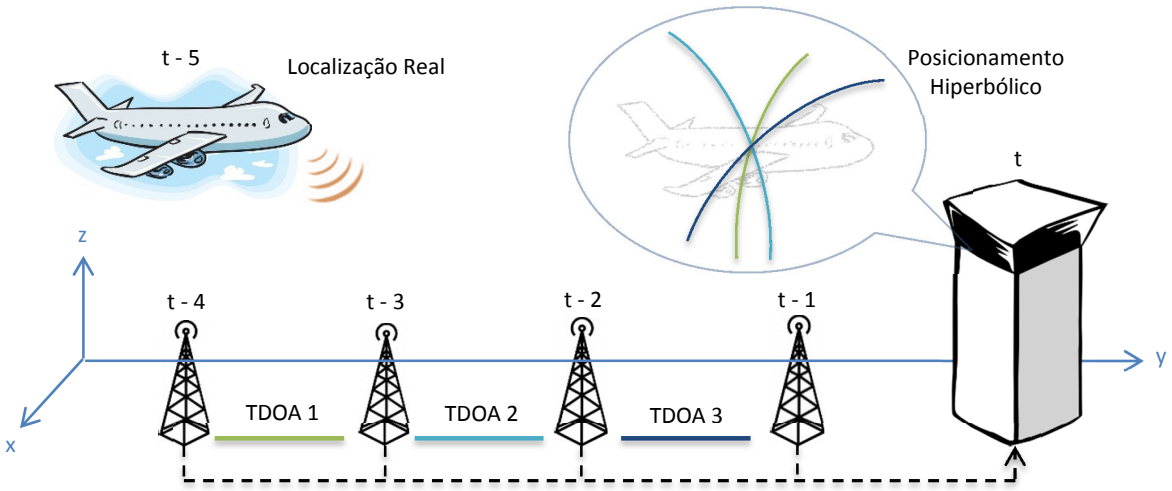


Figura 3.5 – Exemplificação do método de localização por multilateração.

É possível evitar a utilização da relação velocidade-tempo para determinar a distância relativa aos pontos de referência, como na trilateração. Para tal, a triangulação recorre aos ângulos de chegada do sinal (*AOA*, *Angle of Arrival*), obtidos por um ponto de referência dotado de uma antena rotativa ou de uma antena composta por vários recetores direcionais, para determinar a direção do dispositivo móvel. Para uma antena com vários recetores, é normal a utilização da técnica de *TDOA* para se verificar qual dos recetores da antena recebeu em primeiro lugar o sinal, de modo a inferir corretamente a direção de onde o sinal foi originado. A utilização de pelo menos 2 pontos de referência com estas características permite inferir a posição "exata" do objeto em questão. Necessita de equipamento mais sensível para detetar diferenças temporais subtis que, por norma, é mais dispendioso [44,46,51].

Quando se implementa um sistema de localização cuja infraestrutura garanta uma distribuição do sinal ao longo de toda a área de intervenção, pode ser vantajoso efetuar a localização através da análise dessa distribuição. Caso o planeamento da infraestrutura garanta a não uniformidade do sinal, é possível verificar que, em cada local, o sinal possui características únicas que permitem distinguir uma zona das outras. Estas características podem ser vistas como uma "impressão digital" do sinal, que apenas depende da sua localização dentro da área de implementação. Desta forma, caso se possuam todas as impressões digitais existentes, torna-se possível comparar e fazer corresponder uma determinada impressão digital, recolhida em tempo real, a uma já existente. Esta acaba por determinar a posição de um dispositivo dentro da zona de implementação do sistema [45,50,53,54]. Devido à analogia com uma impressão digital, o método de análise da zona de intervenção é também conhecido como método de impressão digital.

A simplicidade e precisão associada aos sistemas de localização por impressão digital desperta o seu interesse para a sua aplicação conjunta com o sistema de monitoração proposto. Desta forma, detalha-se o funcionamento deste tipo de sistemas de localização no subcapítulo 3.2.

3.2 Método de Localização por Impressão Digital

Este método de localização consiste na realização de dois passos. O primeiro efetua o estudo da distribuição do sinal ao longo da zona de implementação e respetiva aquisição de impressões digitais. Este processo é denominado fase de calibração ou fase *offline*. O segundo passo ocorre em tempo real e consiste na comparação das impressões digitais recolhidas durante a fase de calibração com as impressões digitais obtidas através da interação com o dispositivo a localizar. Devido à natureza deste processo, esta fase denomina-se fase *online*, de estimação ou de monitoração.

Como se tratam de sistemas que utilizam redes sem fios (*WLAN*, *Wireless Local Area Network*), que transmitem sinais de potência ao longo da zona de implementação, uma impressão digital possui sempre como atributo base a potência do sinal verificada num determinado local (ou uma métrica com ela relacionada) [47,50,51,53]. Este valor é conhecido como indicador da força do sinal recebido, ou *RSSI* (*Received Signal Strength Indicator*), e oscila tipicamente entre os -10 e os -100 dBm [50,54]. Seguidamente detalha-se a fase de calibração e monitoração, apresentando-se algumas técnicas e mecanismos utilizados por sistemas de localização por impressão digital para determinar o posicionamento dos dispositivos, ou utilizadores, alvos de monitoração.

3.2.1 Fase de Calibração

A fase de calibração, como já referido, consiste no estudo da zona de implementação para adquirir informação relativa à distribuição do sinal. Esta informação pode ser a variação dos valores de *RSSI*, a relação sinal-ruído verificada (*SNR*, *Signal-Noise Ratio*) [8], ou outra métrica relacionada com a força do sinal que possa ser armazenada numa base de dados ou semelhante. Todos os valores recolhidos permitem a criação de um modelo de distribuição do sinal, que permite associar a posição física de um dispositivo com os valores recolhidos dentro do local de implementação, como ilustrado na figura 3.6. Um modelo constituído por valores de *RSSI* costuma se designar mapa de potência do sinal.

O mapa de potência do sinal é um modelo discreto composto por vários pontos de calibração. Existe sempre um compromisso associado ao número de pontos de calibração que compõe um mapa de potência. Dependendo do algoritmo de localização utilizado, bem como das necessidades de localização em tempo real, precisão e complexidade do sistema, o número de amostras recolhidas torna-se determinante nestes sistemas [51]. Por exemplo, um número excessivo de pontos consegue

detalhar com maior precisão a distribuição do sinal, mas aumenta a carga computacional. Por outro lado, a utilização de poucas amostras pode não ser suficiente para fornecer o nível de precisão desejado [43]. Por norma, a força do sinal não oscila entre pontos distanciados por menos de 1 m. A uniformidade da grelha de pontos que define um mapa de potência nem sempre é possível devido à presença de obstáculos na zona de implementação [46,50].

Existem dois métodos para se efetuar a fase de calibração. O primeiro consiste na colaboração dos utilizadores para a obtenção dos valores de *RSSI*. Esta abordagem diz-se "estática" uma vez que cada utilizador se desloca para uma localização pré-definida e, nesse ponto, executa o *software* de aquisição da força do sinal. Ao fim de alguns instantes, a informação relativa a esse local é processada e armazenada. No método "dinâmico", a aquisição é efetuada por um dispositivo de treino capaz de se deslocar, como um carro telecomandado ou semelhante, que recolhe os valores necessários ao longo da zona de implementação. A automatização do processo não é obrigatória mas reforça a noção de comodidade e de dinâmica inerente ao método. A desvantagem deste tipo de calibração é a necessidade de se precisar a localização atual do dispositivo de calibração, de modo a associar a força do sinal ao local onde foi efetuada a amostragem. Uma solução típica passa por dotar o aparelho com capacidade de determinar a distância relativa percorrida face ao ponto de partida [51].

O processo de calibração, apesar de ser um passo moroso de acordo com a dimensão da zona de implementação e com os pontos que se pretendem recolher, só é efetuado uma vez durante o tempo de vida útil do sistema. A sua recalibração só deve ser efetuada caso ocorra alguma alteração de grandes dimensões na infraestrutura do local ou do próprio sistema de localização. Afinal, a natureza das tecnologias associadas a estes sistemas, mais propriamente a utilização de tecnologias de comunicação sem fios passíveis de sofrer fenómenos de reflexão, difração, multicaminho e perdas por propagação, originam uma dependência entre a distribuição do sinal e a infraestrutura do edifício (localização dos pontos de referência, disposição das paredes, janelas e obstáculos) [45,47,51]. Porém, esta dependência permite atenuar a variabilidade do sinal associada a estes fenómenos, uma vez que o mapa de potência do sinal já os tem em conta. No entanto, um certo grau de variabilidade da distribuição do sinal está sempre inerente a este tipo de sistemas. Tem-se como exemplo a influência das próprias condições climatéricas, como a humidade e a temperatura, que origina uma variação no desempenho dos sistemas de localização ao longo do próprio dia, como verificado em [46].

Para além da variabilidade causada pela infraestrutura e os seus obstáculos, outra fonte de erro é introduzida por elementos dinâmicos, como os próprios utilizadores do sistema [46,51]. Note-se que o corpo humano é constituído maioritariamente por água e, caso este se encontre entre o emissor e o recetor, serve como obstáculo para a propagação do sinal. Para além disso, as antenas dos dispositivos móveis possuem ganhos não uniformes de acordo com a orientação do dispositivo. Em [46], vários

testes preliminares permitiram verificar que apenas este fator introduz em cada ponto de calibração uma variação de 5 dB. Deste modo, ambientes muito dinâmicos conseguem originar uma queda de desempenho ainda considerável. Um método recorrente para tentar atenuar a influência do corpo humano e a direção do dispositivo no processo de localização, consiste em recolher amostras auxiliares de *RSSI* em cada ponto de calibração. As medições efetuadas são obtidas de acordo com o sentido dos 4 pontos cardeais (Norte, Sul, Este e Oeste), sendo utilizadas durante o processo de monitoração para compensar o erro resultante do desconhecimento da orientação do dispositivo móvel e da localização do utilizador, que introduzem ambiguidades no sistema [46,55].

Para além dos problemas supracitados, existem outras causas que influenciam negativamente o desempenho dos sistemas de localização por impressão digital, cujas tentativas de resolução ocorrem durante ou antes da fase de calibração. A primeira é um caso particular da tecnologia de comunicação sem fios Wi-Fi, ou norma IEEE 802.11, mas que provavelmente se aplica a todas as outras tecnologias de comunicação sem fios. A norma IEEE 802.11 especifica que o valor de *RSSI* é uma informação opcional que representa uma função da força do sinal relativamente à distância relativa entre os dispositivos emissor e recetor. Como se trata de um indicador relativo e opcional, a sua resolução, sensibilidade e intervalo de deteção fica ao critério de cada fabricante. Isto origina uma discrepância entre os valores recolhidos por diferentes dispositivos, de fabricantes diferentes, no mesmo local [44]. Este problema torna-se evidente durante a fase de calibração, propagando-se para a fase de monitoração.

Normalmente, durante a fase de calibração, a aquisição dos valores da força do sinal são realizados apenas com um dispositivo, denominado “dispositivo de treino”. Após a aquisição do mapa de potência do sinal, pode-se utilizar o dispositivo de treino como dispositivo de monitoração, de modo a representar um utilizador e permitir a estimação da sua localização. Caso se adicionem outros dispositivos ao sistema, de fabricantes diferentes, o desempenho degrada-se devido à discrepância existente entre os valores do mapa de potência e os valores recolhidos em tempo real, como previamente mencionado. Para solucionar este problema, os autores [44] propõem a calibração dos novos dispositivos de monitoração face ao dispositivo de treino, de forma a ajustar os valores obtidos pelos novos dispositivos aos valores para o qual o sistema foi desenhado.

Apesar dos valores de *RSSI* serem algo facultativo, todos os fabricantes disponibilizam-nos devido às funcionalidades e serviços oferecidos.

3.2.2 Fase de Monitoração

A fase de monitoração consiste na obtenção da localização atual do dispositivo, ou do utilizador, durante todo o tempo de funcionamento do sistema. Para tal, comparam-se os valores de *RSSI*

recolhidos em tempo real com as observações obtidas no processo de calibração, de modo a estimar-se a localização física atual do dispositivo móvel [47,48,50,51,53]. No entanto, o processo de monitoração pode não recorrer diretamente aos valores de *RSSI* para determinar qual o melhor ponto de calibração que se adequa aos valores obtidos. Existem diversas técnicas e métodos que utilizam outras representações dos valores de *RSSI*, desenvolvidas ou implementadas por diversos autores ao longo dos anos que recorrem a um dos tipos de interação descritos no subcapítulo 3.1. Porém, as interações do tipo posicionamento remoto indireto e auto posicionamento direto são as mais comuns.

O reconhecimento de padrões e características, através de mecanismos de classificação ou outras técnicas que podem ser aplicadas durante a fase de monitoração, sempre foram alvo de intensivo estudo. No caso dos sistemas de localização por impressão digital têm-se métodos probabilísticos [46,50,51,55], métodos de vizinhança [46,50,51,53,55-57], rede neuronais [58], filtros [59-61], máquinas de suporte a vetores [50] e inferência difusa [58,62]. No âmbito deste trabalho, efetua-se somente a abordagem dos métodos de vizinhança, devido à sua simplicidade e nível de precisão obtido.

Os métodos de vizinhança são algoritmos de procura exaustiva que efetuam a comparação de uma determinado conjunto de características com todos os conjuntos de características existentes. Como o objetivo é determinar qual dos conjuntos existentes mais se assemelha ou se aproxima do conjunto fornecido, estes métodos são também conhecidos como algoritmos de procura, inferência ou interpolação do vizinho mais próximo (*NNS*, *Nearest Neighbour Search*) [50,56]. O algoritmo *NNS* é computacionalmente mais demorado e deve ser apenas utilizado caso o número de características ou de conjuntos não seja demasiado elevado, de modo a não comprometer o desempenho do sistema.

A variação mais simples do algoritmo *NNS* analisa todos os conjuntos existentes e devolve apenas o valor mais semelhante com o conjunto apresentado. Este método é mais simples mas também pode introduzir um erro bastante elevado num sistema de localização por impressão digital. Afinal, os valores de *RSSI* possuem uma certa variabilidade associada e os pontos de calibração recolhidos, utilizados como conjuntos de comparação, também possuem valores relativamente próximos. Para contornar este problema, o algoritmo k-*NNS* devolve o conjunto que mais vezes se repete dentro de uma amostra composta por 'k' conjuntos mais próximos do conjunto fornecido [56,57]. A determinação do número de conjuntos 'k' é normalmente efetuada através de testes preliminares, quer por implementação ou por simulação [50]. Os testes realizados pelos autores [53,57] permitiram verificar que os valores '3' e '4' garantiam um bom desempenho para os casos em estudo.

No entanto, caso não se deseje ou não seja possível determinar qual o conjunto que mais vezes ocorre dentro da amostra de 'k' conjuntos, pode-se efetuar a média dos conjuntos obtidos, de modo

a se estimar a localização de um dispositivo. O cálculo desta média não tem que ser uniforme, podendo-se atribuir um grau de certeza, ou incerteza, a cada conjunto obtido através da definição de pesos. Desta forma, o algoritmo k-*NNS* possui duas variantes: *weighted* k-*NNS*, que define diversos pesos para cada conjunto, e o *unweighted* k-*NNS*, que recorre somente ao cálculo da média sem preferências [46,50]. A abordagem do *weighted* k-*NNS* torna-se interessante caso se saiba a credibilidade associada aos valores *RSSI* recolhidos em tempo real, nomeadamente aos valores obtidos através de pontos de referência mais distantes ao dispositivo a localizar, capazes de introduzir erro no sistema.

O conceito por detrás do algoritmo *NNS* e as suas variantes é bastante simples, consistindo num método de determinação de conjuntos que possuam características mais próximas de outro conjunto. No entanto, como se pode deduzir, a sua aplicabilidade só é possível para conjuntos que possuam características com representações numéricas, onde a verossemelhança entre conjuntos é determinada através do cálculo da distância entre características. Desta forma, um conjunto diz-se mais próximo ou semelhante de outro conjunto caso as distâncias das suas características sejam mais próximas. A proximidade entre conjuntos com 'd' características pode ser determinada por uma das funções de distância (heurísticas) dadas por (3.1), (3.2), (3.3) e (3.3) [63].

Distância Euclidiana a 'd' dimensões $$J_e[k,l] = \sqrt{\sum_{i=1}^{d}(x_{ik} - x_{il})^2} \quad (3.1)$$

Distância de Manhattan $$J_{cb}[k,l] = \sum_{i=1}^{d}|x_{ik} - x_{il}| \quad (3.2)$$

Distâncias de Minkowski $$J_M[k,l] = \left[\sum_{i=1}^{d}(|x_{ik} - x_{il}|)^{\lambda}\right]^{1/\lambda} \quad (3.3)$$

Distância de Mahalanobis $$d(x_i, x_0) = (x_0 - x_i)^T S^{-1}(x_0 - x_i) \quad (3.4)$$

Todas as funções apresentadas possuem como objetivo minimizar a diferença entre o conjunto observação e o conjunto armazenado. Desta forma, quanto mais próximo de '0' for o resultado da aplicação da função de distância, maior a similaridade entre os conjuntos. A distância de Manhattan, ou *city-block*, recorre ao conceito típico da deslocação de um indivíduo ao longo das ruas de uma cidade, sendo obrigado a contornar os edifícios de modo a alcançar o seu destino. A distância Euclidiana ignora os edifícios e determina a distância mínima entre origem e destino, dada pela hipotenusa entre pontos. As distâncias de Minkowski são consideradas a generalização das distâncias Euclidiana (λ = 2), de Manhattan (λ = 1) e de Chebyshev (λ = ∞). O valor 'd' representa as dimensões ou o número de características que se pretendem analisar e os valores de x_{ik} e x_{il} representam a característica a ser comparada, dada pelo conjunto de observação x_k= $\{x_{1k}, x_{2k}, \ldots, x_{dk}\}$ e um dos

conjuntos armazenados $x_l = \{x_{1l}, x_{2l}, \ldots, x_{dl}\}$ [63]. As várias noções de distância que cada uma das funções supracitadas define encontra-se ilustrada na figura 3.5.

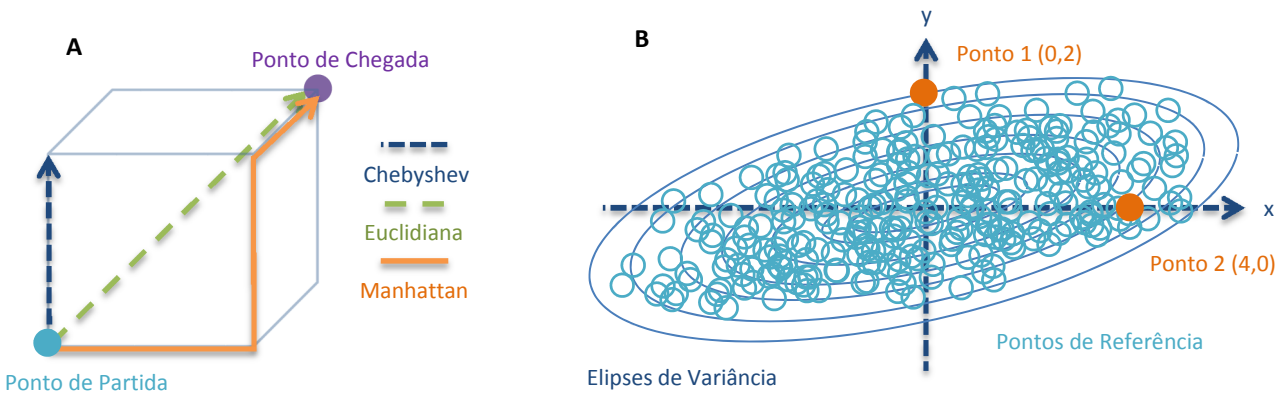


Figura 3.6 - (**A**): Definição de distância Euclidiana, Manhattan e Chebyshev. (**B**): Problemática da variância das características 'x' e 'y' de vários conjuntos que obrigam a introdução do conceito de distância de Mahalanobis.

A distância de Mahalanobis efetua uma análise mais estatística através da definição de uma correlação entre as características dos conjuntos e a variância associada a cada característica, dadas pela matriz de covariância 'S'. Isto deve-se ao facto de algumas características dos conjuntos possuírem diferentes probabilidades de ocorrerem, o que acaba por influenciar a definição de distância. Caso a matriz de covariância S seja uma matriz identidade, então não existe nenhuma correlação entre características e obtém-se o caso da distância Euclidiana. No caso da figura 3.6, a aplicação da distância Euclidiana diria que o ponto 1 está mais próximo do centro de massa (origem do referencial), logo seria mais similar. No entanto, o ponto 1 encontra-se dentro de uma elipse de variância de 90 %, logo a sua probabilidade de ocorrer é menor face ao ponto 2. A distância de Mahalanobis tem em conta este facto.

Não existe propriamente uma vantagem ou desvantagem para a utilização de uma ou outra métrica, uma vez que se tratam de heurísticas para definir a noção de distância e extrair a distância mais curta entre conjuntos, podendo a distância Euclidiana ser mais adequada para uns casos e a distância de Mahalanobis para outros [64].

3.2.3 Tecnologias e Sistemas

Como referido ao longo deste subcapítulo, os sistemas de localização utilizam tecnologias de comunicação sem fios, ou semelhantes, para efetuar a localização de um dispositivo, ou utilizador, dentro da sua zona de intervenção. Desta forma têm-se sistemas que recorrem a infravermelhos (*IR*, *InfraRed*) [48,55,65], *Ultra-High-Frequency/Very-High-Frequency* (*UHF/VHF*) [66], *Ultra-Wide-Band* (*UWB ou* IEEE 802.15.3) [45,48], Ultrassom [55], Processamento de Imagem [67], Identificação por

Radiofrequência [48,55,68], ZigBee [48,69,70], Bluetooth [48,49,71], Wi-Fi (IEEE 802.11) [51,54,56,57,72], entre outras [48]. Algumas tabelas comparativas entre sistemas e tecnologias são apresentadas em [48,53]. No caso particular da localização por impressão digital, é comum a utilização das tecnologias ZigBee, Bluetooth e Wi-Fi devido ao seu baixo custo, alcance razoável e capacidade de comunicação e transmissão de dados disponibilizada pela *WLAN* [48]. Uma vez que se pretende reaproveitar a infraestrutura do edifício, através da *WLAN* composta pelos pontos de acesso ao longo do edifício e os aparelhos existentes na posse dos utilizadores (e.g. *smartphones*), efetua-se o foco da utilização da norma IEEE 802.11 para a implementação de um sistema de localização por impressão digital.

A norma IEEE 802.11, mais conhecida por Wi-Fi (*Wireless Fidelity*), possibilita a comunicação sem fios com uma elevada largura de banda, permitindo comunicações até aos 100 Mb/s [45,73]. O alcance e ritmo de transmissão de dados, bem como a eficiência energética associada aos transmissores e recetores, depende da variação da norma utilizada, sendo atualmente mais utilizadas as variações IEEE 802.11g e IEEE 802.11n, que garantem alcances elevados (≈ 140 - 250 m) em ambientes sem obstáculos [73]. Desta forma, é uma tecnologia que facilmente consegue garantir a cobertura de todo o edifício. A sua criação possui como objetivo a substituição da comunicação por fio existente nos computadores e noutros dispositivos.

A tecnologia Wi-Fi possui um enorme potencial para sistemas de localização, devido à existência da infraestrutura que tem vindo a expandir-se nos últimos anos, o que permite uma enorme redução de custos [48,53]. Como não se trata de uma tecnologia com *hardware* proprietário, imensos dispositivos podem usufruir dela e só dependem da existência da infraestrutura [45]. Atualmente, uma *WLAN* do tipo Wi-Fi pode ser encontrada nas escolas, campus universitários, centros comerciais, empresas, habitações e em alguns locais públicos [48,53]. A sua disponibilização é efetuada por aparelhos denominados pontos de acesso (*AP, Access Point*), que se encontram dispersos ao longo da zona de intervenção de modo a garantir uma boa cobertura e a qualidade do serviço ao todos os seus utilizadores [45]. A disponibilidade desta tecnologia no lado do utilizador, através de computadores portáteis, *smartphones* ou *tablets*, também permite a redução de custos uma vez que não é necessário para o utilizador carregar um novo dispositivo, ou semelhante, para interagir com a rede e se estimar a sua localização [53].

A determinação da localização de um dispositivo móvel através da norma IEEE 802.11 pode ser efetuada a partir de um dos métodos de localização descritos em 3.1 onde, por exemplo, a localização por célula é determinada pela proximidade a um determinado *AP* ou a análise do tempo decorrido entre pacotes transmitidos permite determinar a distância relativa entre emissor e recetor (multilateração) [45]. Como supracitado, o método de localização por impressão digital é possível

devido à distribuição da potência do sinal verificada ao longo da zona de implementação, através da análise dos valores de *RSSI*.

No entanto também existem desvantagens associadas à utilização de *WLAN* do tipo Wi-Fi para a implementação de um serviço de localização. A configuração da própria infraestrutura condiciona a capacidade de estimação efetuada pelo sistema de localização. Um elevado número de *APs*, por norma, garante uma melhor estimação. No entanto implica maiores custos caso o edifício não possua esse equipamento. Porém, verifica-se em [50] que um número excessivo de *APs*, especialmente aqueles quase inaudíveis, acaba por degradar o desempenho do sistema, sendo necessária a introdução de algoritmos para determinar qual o melhor número de *APs* a utilizar durante a fase de monitoração [74]. Para além disso, a infraestrutura pode não garantir a melhor configuração de *APs* que permita ao sistema de localização demonstrar todo o seu potencial. Afinal, a maioria dos edifícios efetua a distribuição dos *APs* para garantir a sua cobertura e a qualidade do serviço, não para efetuar localização [46].

Para além disso, a partilha da banda dos 2.4 GHz da tecnologia Wi-Fi com outros aparelhos existentes nos edifícios também se torna num inconveniente. Desta forma, equipamentos industriais, médicos e científicos que recorrem à tecnologia Bluetooth [50], podem causar interferências num sistema de localização que utiliza Wi-Fi.

Outro problema está relacionado com os fenómenos de multicaminho, refração, difração e perdas por propagação, típicos das tecnologias de comunicação sem fios em zonas de intervenção interiores, que causam variabilidade do sinal, como descrito no subcapítulo 3.2.1. Deste modo, grande parte da investigação incide sobre o método de localização por impressão digital, que consiste na análise do local e armazenamento da informação relativa ao sinal, que já possui em conta os fenómenos descritos e atenua a variabilidade do sistema [45]. Uma vez que o Wi-Fi não foi desenhado com o objetivo de localizar e existe sempre uma variabilidade inerente ao sistema, os sistemas de localização que se baseiam nesta tecnologia possuem, normalmente, uma menor precisão face a outros sistemas.

São muitos os sistemas de localização por impressão digital que se baseiam nesta tecnologia. O primeiro foi o RADAR [46,55,57] que começou por ser um estudo para determinar qual seria a melhor metodologia para determinar a localização de um dispositivo. Entre o método de impressão digital e o método de propagação (*TOA*), chegou-se à conclusão que o primeiro, apesar da calibração inicial, apresenta melhores resultados. O método de propagação implica lidar-se com o desconhecido devido à tentativa de expressar-se matematicamente as perdas por multicaminho ou por propagação, devido à existência de obstáculos. Daí os resultados não terem sido tão favoráveis para esta abordagem. O método de impressão digital recorre ao algoritmo k-*NNS* para estimar a localização do dispositivo alvo

de monitoração, tendo sido determinado que o melhor desempenho do sistema era obtido com um valor de 'k' igual a 3 [53,57].

Outro sistema bastante conhecido é o PlaceLab. Este sistema *open source* possui como objetivo dotar os dispositivos (computadores, telemóveis, etc.) com a capacidade de se localizarem através da análise de sinais de Wi-Fi, GSM e/ou Bluetooth. [48]. Existem ainda imensos sistemas comerciais ou em investigação, como o AeroScout [48], Ekahau [46,48], Aruba Networks [48], Cisco [48], Meru Networks [48], Zebra WhereNet [48] e Trapeze Networks [48], que se baseiam no princípio do RADAR para fornecerem um melhor serviço de localização e outras funcionalidades. Também existem outros sistemas de localização que combinam várias tecnologias para melhorar o desempenho do sistema. No entanto, a maioria dos casos origina custos adicionais e não serve como fonte de inspiração para este trabalho. Porém, existem alguns sistemas que incorporam outros modelos da tipologia do edifício, ou da deslocação da pessoa, de modo a melhorar a precisão de um sistema de localização por impressão digital por Wi-Fi.

Uma planta ou mapa de um edifício oferece informação relativa à existência de obstáculos de maior dimensão e à disposição das paredes, portas e janelas. Estes elementos permitem não só determinar a área transitável pelos ocupantes de um edifício, como também inferir quais as zonas que poderão ser mais problemáticas para a estimação da localização de um dispositivo, como demonstrado na figura 3.7. O uso desta informação deve ser efetuado com precaução, uma vez que se pretende eliminar ambiguidades ocorridas durante a fase de monitoração, e não o contrário.

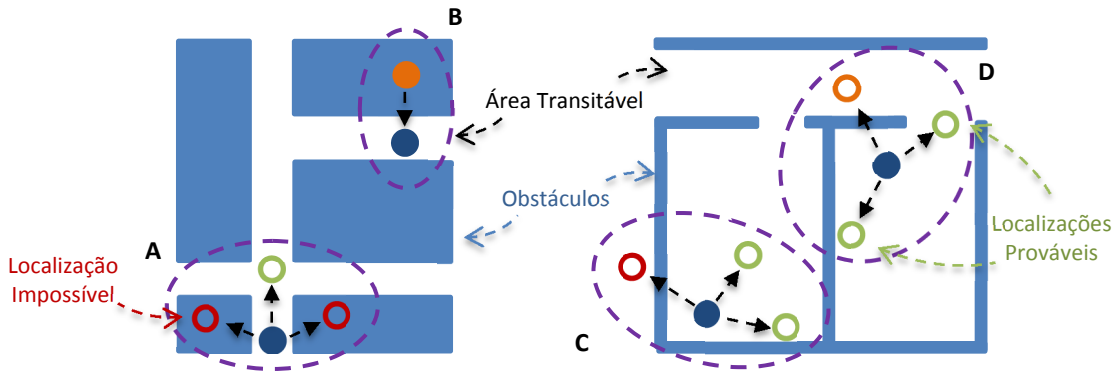


Figura 3.7 - (**A**): Impossibilidade da próxima localização ser dentro de um obstáculo. (**B**): Erro de estimação inicial (laranja) e respetiva correção. (**C**): Impossibilidade de atravessar paredes. (**D**): Baixa probabilidade de percorrer uma distância relativamente elevada num curto espaço de tempo.

Em [59] o sistema de localização recorre a filtros (Partículas e Kalman) para estimar a posição atual de um dispositivo móvel. A utilização do modelo do edifício no filtro de partículas permite conferir aos pontos de localização provável outro grau de confiança. Para tal, verifica-se em tempo real se existe algum obstáculo a separar a posição anterior da posição atual, dada pelos valores de *RSSI* observados e os pontos resultantes do filtro de partículas. Se tal se verificar, determina-se o caminho

mais curto entre o ponto anterior e o ponto atual, de modo a atualizar os pesos das várias partículas que definem a posição atual. A determinação do caminho mais curto é efetuada através do algoritmo de Dijkstra.

No sistema de localização descrito em [60], a modelação das paredes de um edifício passa por associar um campo de potencial à zona de implementação do sistema. O modelo consiste numa grelha composta por diversas células que poderão estar, ou não, ocupadas por um obstáculo. No caso em estudo, os obstáculos são dados por uma secção da parede ou por móveis de dimensões consideráveis. A definição de uma expressão de repulsão para as células da grelha possibilita a diminuição da probabilidade de ocupação das células com obstáculos por parte do utilizador. A determinação das células adjacentes a um utilizador é efetuada com base na posição atual do dispositivo móvel e num modelo de deslocação do ser humano. Este define a velocidade e aceleração máximas que um indivíduo pode exercer, resultando numa janela de análise mais reduzida durante o processo de monitoração.

Em [55] a obtenção do modelo de deslocamento de um utilizador não se baseia somente na aceleração e velocidade teórica, mas também nos dados fornecidos por sensores de vibração dispersos pelo edifício. Deste modo consegue-se pré-determinar a distância máxima que o utilizador poderá percorrer, sendo essa distância ajustada de acordo com a velocidade atual do utilizador, obtida através das vibrações.

Capítulo 4: A Solução Proposta

Para se cumprirem os objetivos propostos no capítulo 1, é necessário implementar um sistema capaz de distinguir e atribuir o consumo elétrico global de uma habitação a cada um dos seus ocupantes. Como já referido, em vez de se abordar o problema como um todo, propõe-se a criação de dois sistemas, o WiLOS (*Wi-Fi Location and Occupancy System*) e o HUEPS (*House and User Energy Profile System*). A troca de informação entre WiLOS e HUEPS, bem como a sua interação com a habitação e os seus ocupantes, são determinantes para garantir o bom funcionamento de todo o sistema. O conceito global do sistema proposto encontra-se representado na figura 4.1.

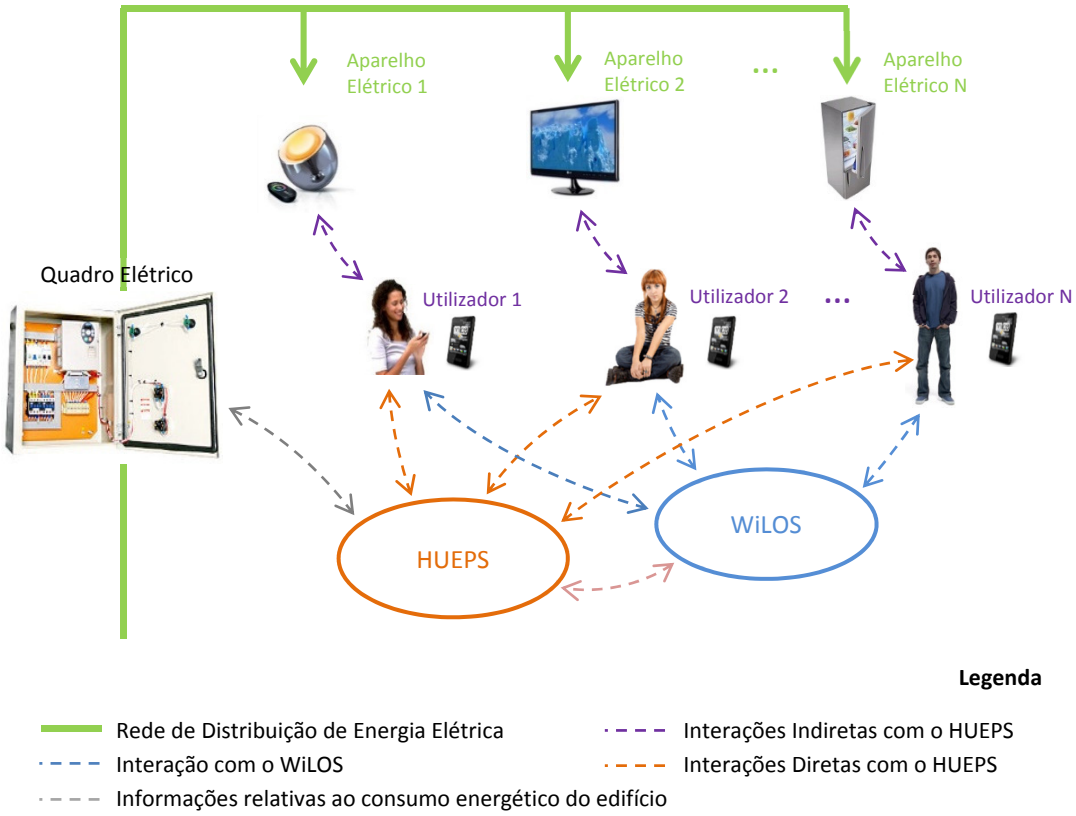


Figura 4.1 - Conceito do sistema global. Definição das interações existentes entre atores e fluxo de informação.

O WiLOS recorre aos dispositivos móveis dos utilizadores para determinar as suas localizações dentro da rede Wi-Fi da habitação. Caso esta infraestrutura seja inexistente, tem de se proceder à sua instalação. A deslocação dos utilizadores ao longo da habitação é detetada pelo WiLOS, ocorrendo uma interação entre o utilizador e o WiLOS sem que este se aperceba. Este tipo de interações são consideradas "interações indiretas" com o sistema.

De modo semelhante, sempre que um inquilino liga ou desliga um aparelho elétrico existe uma flutuação do consumo energético da habitação. Estas variações são detetadas pelo HUEPS, que recorre

à informação posicional dos inquilinos (WiLOS) para efetuar a distribuição do consumo energético. Sendo assim, uma interação entre um utilizador e um aparelho elétrico enquadra-se no conceito de interação indireta. Por outro lado, a manipulação da informação de um sistema de forma propositada, como a consulta de dados ou configuração do sistema, é uma forma de "interação direta".

O conceito do sistema global é importante para definir bem quais os papéis do WiLOS e do HUEPS dentro da habitação. As interações entre os inquilinos e os sistemas, quer sejam diretas ou indiretas, têm de ser realçadas pois são o ponto de partida para se deduzir as funcionalidades que cada sistema, WiLOS ou HUEPS, devem oferecer aos seus utilizadores. A definição destes sistemas começa pelo WiLOS, uma vez que o HUEPS necessita dos mecanismos por ele implementados para garantir o seu funcionamento.

4.1 WiLOS

O WiLOS é um sistema de localização habitacional sem fios. Este tem como objetivo determinar a presença dos seus utilizadores dentro da habitação, indicando com precisão a divisão por eles ocupada. Trata-se, portanto, de um sistema interior. No subcapítulo 3.2.3 verifica-se um aumento da expansão de redes Wi-Fi a nível empresarial e habitacional, bem como uma maior acessibilidade por parte da população a dispositivos móveis capazes de interagir com este tipo de infraestruturas [75]. Propõe-se então que o WiLOS seja implementado nessa infraestrutura já existente, de modo a reduzir custos e reaproveitar recursos. Porém, caso seja necessária a introdução de mais alguns *APs*, o custo não é muito elevado e fornece aos utilizadores maior cobertura para comunicação.

Também está comprovado estatisticamente que, atualmente, as pessoas transportam sempre consigo um dispositivo móvel, quer para conversar, trocar mensagens ou simplesmente para navegar na internet. Existe inclusive um estudo que afirma que 90% das pessoas dos 18 aos 29 anos dormem com um *smartphone* ao lado da cama [76]. Esta informação é crucial para o WiLOS, pois o dispositivo móvel torna-se num instrumento que permite representar o utilizador e, consigo, a sua localização. Assim, o papel do WiLOS consiste em verificar se o dispositivo móvel se encontra no alcance da rede Wi-Fi da habitação e inferir a sua localização. Esta metodologia torna o sistema passivo, isto é, o WiLOS atua e cumpre o seu objetivo de forma transparente ao utilizador, sem necessitar de grande intervenção.

Para realizar este feito, propõe-se que o WiLOS seja um sistema distribuído do tipo *master-slave*, onde um subsistema principal, o "Mestre", gere e coordena vários subsistemas secundários, os "Escravos", e toda a informação que circula no sistema. O Mestre pode ser instalado num computador ou equipamento equivalente, desde que possua capacidade de processamento e mecanismos que permitam a gestão de todo o sistema (rede Wi-Fi e os seus clientes). O Escravo consiste numa aplicação

para dispositivos móveis que responde aos pedidos de informação efetuados pelo Mestre. No entanto, ele próprio pode requisitar alguma informação caso o utilizador desejar e o Mestre permitir. Desta forma surgem o mWiLOS, *master* WiLOS, e o sWiLOS, *slave* WiLOS. Como os nomes sugerem, o primeiro efetua o papel de Mestre e gestor do WiLOS, enquanto o sWiLOS representa o Escravo que é alvo de monitoração do mWiLOS. Estas interações correspondem a um paradigma de localização do tipo auto posicionamento indireto.

Um dos principais desafios do WiLOS é a utilização de tecnologias que usufruem da norma 802.11 para a deteção do dispositivo móvel do utilizador, onde a margem de erro deve ser minimizada. O método de localização escolhido para superar este desafio consiste no método de impressão digital. Este implica a existência de dois modos de funcionamento no WiLOS: um de calibração e um de monitoração. Na fase de calibração efetua-se o mapeamento da potência do sinal da rede Wi-Fi da habitação. O mapa obtido deve retratar com a maior fidelidade possível o sinal de potência existente. Posteriormente, a fase de monitoração requer que o mWiLOS questione periodicamente o sWiLOS, de modo a se obterem os valores de potência evidenciados pelo dispositivo móvel. Esta informação é utilizada pelo mWiLOS durante a aplicação do algoritmo k-*NNS*, onde o ponto do mapa de potência mais semelhante com os valores de *RSSI* recolhidos permite determinar a divisão onde o utilizador se encontra. Para além disso, o WiLOS recorre a modelos da tipologia da habitação para tentar melhorar o desempenho oferecido pelo serviço de localização.

Este subcapítulo aborda o desenvolvimento do WiLOS através da descrição dos seus elementos, o mWiLOS e o sWiLOS. Primeiramente é apresentado o modelo concetual do mWiLOS, seguido do modelo concetual do sWiLOS. Para finalizar, o processo de implementação de ambos os constituintes do WiLOS é abordado com pormenor, bem como todas as opções realizadas.

4.1.1 Modelo Concetual – mWiLOS

O mWiLOS é um subsistema capaz de gerir e monitorar múltiplos dispositivos móveis que se encontrem registados na rede WiLOS através do sWiLOS. É nele que se centra o processamento da informação utilizada para se determinar a localização dos utilizadores. Também permite a interação com o utilizador através da configuração do sistema, inserção de dados referentes aos utilizadores e consulta de informação gerada. Para além disso, o WiLOS não deve ser um sistema isolado do mundo. O seu conhecimento deve ser partilhado e utilizado por outros sistemas que pretendam executar tarefas mais complexas. Deste modo, o mWiLOS possui a capacidade de disponibilizar os seus serviços (e.g., serviço de localização) para vários sistemas externos.

Como funcionalidade extra, o WiLOS possibilita a comunicação entre os utilizadores da habitação através do conceito de mensagens de texto. Assim aproveita-se a infraestrutura da rede para implementar um serviço gratuito para os utilizadores. Este mecanismo confere aos utilizadores um papel mais ativo dentro do sistema, de modo a não serem apenas consumidores de informação ou alvos de monitoração.

O modelo concetual do mWiLOS é composto por modelo funcional, modelo arquitetural e modelo de dados. O modelo funcional descreve todas as funcionalidades que o mWiLOS tem para oferecer aos seus utilizadores (humanos ou não humanos), possuindo um maior nível de abstração. O modelo arquitetural permite observar as diversas camadas que constituem o mWiLOS. Aqui define-se a arquitetura do subsistema, onde cada módulo do mWiLOS possui um papel determinante em cada uma dessas camadas. Por fim, o modelo de dados identifica as entidades que regem o mWiLOS, assim como as relações existentes entre elas.

4.1.1.1 Modelo Funcional

O mWiLOS oferece ao utilizador várias funcionalidades relacionadas com a localização dos ocupantes na habitação, bem como outras necessárias para a configuração do sistema. As diferentes funcionalidades do mWiLOS permitem definir 7 grupos distintos: “Definir Tipologia da Habitação”; “Definir Mapa de Potência do Sinal”; “Gerir Utilizadores”; “Monitorar”; “Trocar Informação com Sistemas Externos”; “Aceder ao Serviço de Mensagens” e “ Consultar Estatísticas”.

A definição da tipologia da habitação permite definir as divisões da habitação onde o WiLOS se encontra instalado. Outras informações relativas à habitação, como morada, código postal ou telefone, são passíveis de serem introduzidas. Toda a informação inserida pode ser posteriormente consultada e/ou alterada. Após a definição da tipologia da habitação, o mWiLOS possui informação suficiente para obter o mapa de calibração a ser utilizado durante o processo de definição do mapa de potência do sinal.

O mapa de potência do sinal, na realidade, consiste em vários mapas obtidos através da recolha dos valores de *RSSI* relativos a cada um dos *APs* que constituem a rede Wi-Fi da habitação. Estes são adquiridos através de um mecanismo que suporta a troca de informação entre o mWiLOS e o sWiLOS, onde um utilizador com privilégios de administrador age como mediador. Caso seja necessário, a recalibração dos mapas pode ser efetuada posteriormente. Qualquer utilizador possui privilégios para consultar os mapas gerados.

Outro elemento essencial no WiLOS é o utilizador ou, neste caso, utilizadores. Um utilizador com privilégio de administrador pode adicionar novos utilizadores ao sistema, definir o seu nível de acesso e inserir alguns dados pessoais (e.g., idade, contacto, etc.). Também pode associar um dispositivo

móvel a um utilizador, de modo a que o WiLOS identifique de forma inequívoca o dono do dispositivo. A consulta do histórico de localizações de um determinado utilizador é outra funcionalidade existente, e um dos principais objetivos deste sistema.

A monitoração do WiLOS efetua automaticamente a troca de mensagens entre o mWiLOS e o sWiLOS, com o intuito de se obterem as localizações dos dispositivos móveis dos utilizadores. Estes pedidos devem ser realizados periodicamente, não só para determinar o posicionamento do utilizador, mas também para verificar se o utilizador saiu da habitação ou se o dispositivo móvel ficou sem bateria. A constante atualização do posicionamento dos utilizadores permite, a qualquer utilizador, visualizar no mWiLOS a localização de todos os ocupantes em tempo real. O serviço de monitoração pode ser interrompido a qualquer instante através do mWiLOS, caso o utilizador autenticado possua privilégio para executar tal operação.

O funcionamento e desempenho do WiLOS está intrinsecamente ligado às configurações do serviço de monitoração. Para além da definição da periodicidade dos pedidos realizados ao sWiLOS, é necessário definir qual o algoritmo de localização a utilizar durante a sua execução. O administrador pode optar pela utilização de 3 algoritmos: o algoritmo k-*NNS*, que recorre apenas aos mapas de potência baseados nas orientações cardeais; o algoritmo k-*NNS* com compensação de orientação; ou o algoritmo k-*NSS* com filtro de distância aplicado. É também possível utilizar uma configuração que combina os 3 algoritmos apresentados. O detalhar destes algoritmos é efetuado no subcapítulo 4.1.1.2, durante a modelação do módulo responsável pela sua execução.

Outra funcionalidade crucial do WiLOS consiste na disponibilização da informação gerada a outros sistemas. Desta forma, é possível implementar serviços baseados em localização que oferecerem novas ferramentas aos utilizadores. No entanto, a disponibilização da informação do WiLOS na rede Wi-Fi da habitação não é um problema trivial. Note-se que o WiLOS lida com muita informação sensível e de carácter confidencial, tal como os dados pessoais de um utilizador ou as suas rotinas dentro da habitação. Este tipo de dados não pode ser acessível a qualquer sistema da rede. Para garantir a privacidade de um utilizador e a segurança da informação disponibilizada, o mWiLOS oferece mecanismos de autenticação e autorização.

A autenticação de um sistema externo no WiLOS é efetuada através de um par de credenciais, nome e palavra-passe, que são comparados com as credenciais dos sistemas registados no mWiLOS. Este mecanismo é utilizado sempre que um sistema externo deseje aceder a um determinado recurso do WiLOS. No entanto, nem todos os sistemas registados devem ter o mesmo nível de acesso à informação disponibilizada. A associação das funcionalidades do WiLOS a um sistema previamente registado permite definir os recursos que o sistema está autorizado a aceder. Toda a informação

trocada entre mWiLOS e sistemas externos fica registada para consulta futura. Deste modo é possível detetar problemas de comunicação existentes entre os sistemas, ou verificar se algum sistema não autenticado e/ou autorizado se encontra a aceder ao WiLOS.

O serviço de conversação e troca de mensagens permite a comunicação rápida entre os utilizadores do WiLOS, sem custos adicionais. Ao mWiLOS compete a gestão e encaminhamento das mensagens efetuadas pelos utilizadores através do sWiLOS. Esta gestão também passa pelo registo das conversações realizadas, de modo a serem consultadas mais tarde. Caso o utilizador deseje, pode requisitar a eliminação de uma determinada conversação desses registos.

O recurso a elementos estatísticos permite analisar mais facilmente toda a informação recolhida e criada pelo WiLOS. A consulta de gráficos que indiquem quanto tempo é que um determinado utilizador passa em determinadas divisões, qual o sistema externo que solicita mais informação, ou apenas qual o utilizador que passa mais tempo em casa, são elementos úteis que permitem satisfazer a curiosidade dos utilizadores. Cabe ao mWiLOS utilizar a informação do sistema para gerar os elementos estatísticos que o utilizador deseja visualizar.

Uma vez que os utilizadores do WiLOS possuem diferentes níveis de acesso à informação disponibilizada, o mWiLOS necessita de mecanismos de autenticação de utilizadores. Podem existir 3 tipos de utilizadores no sistema: “Administrador”; “Utilizador Comum” e “Visitante”. O administrador possui acesso ao WiLOS sem restrições, de modo a configurar o sistema, consultar a informação, registar utilizadores e alterar a informação existente, caso necessário.

Um utilizador comum representa um ocupante da habitação onde o WiLOS atua. Este consegue consultar toda a informação geral do sistema, exceto aquela relacionada com a sua configuração, calibração e teste. Das funcionalidades disponibilizadas no âmbito da gestão de utilizadores, o utilizador comum pode apenas consultar e alterar a informação a ele associada. Relativamente ao serviço de mensagens, o utilizador apenas tem acesso às suas conversações, o que garante alguma privacidade ao WiLOS.

Caso um utilizador desconhecido deseje observar o funcionamento do WiLOS, pode fazê-lo sob a forma de visitante. Um visitante apenas possui acesso à tipologia da habitação, ao mapa de potência do sinal, à lista de utilizadores registados no WiLOS e às suas localizações atuais dentro da habitação. Como se pode verificar, é um consumidor de informação bastante limitado.

As funcionalidades supracitadas, bem como os elementos que interagem com o mWiLOS, encontram-se representados nos diagramas de casos de uso das figuras 4.2, 4.3 e no anexo A.

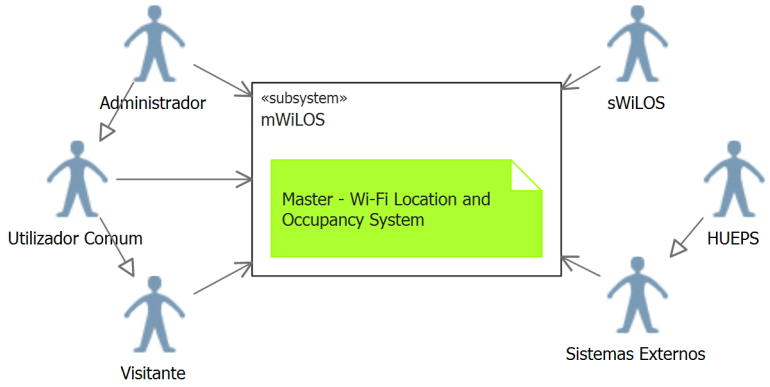


Figura 4.2 - Diagrama de casos de uso do mWiLOS – Atores que interagem com o mWiLOS.

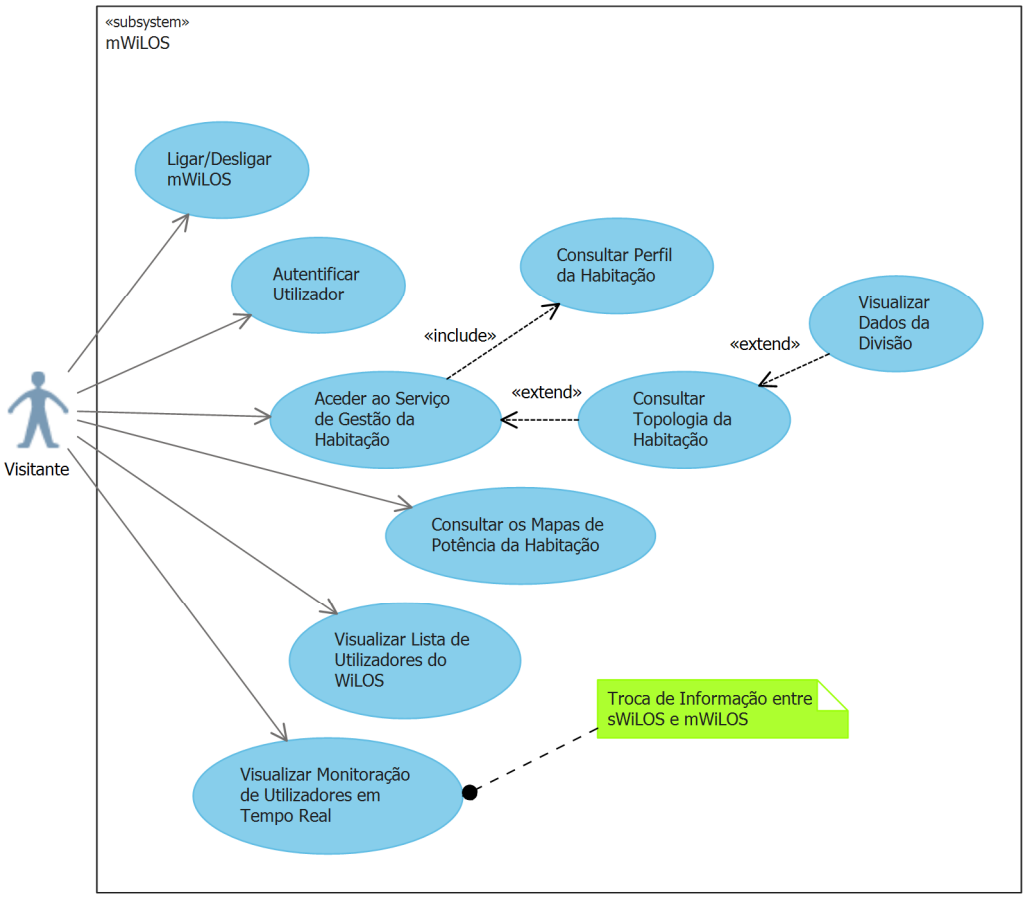


Figura 4.3 - Diagrama de casos de uso do mWiLOS – Funcionalidades disponibilizadas ao Visitante.

4.1.1.2 Modelo Arquitetural

A estrutura do mWiLOS é definida pelas camadas "Interface", "Controlo" e "Entidade". A figura 4.4 permite observar as camadas enumeradas, bem como os diversos módulos que as constituem.

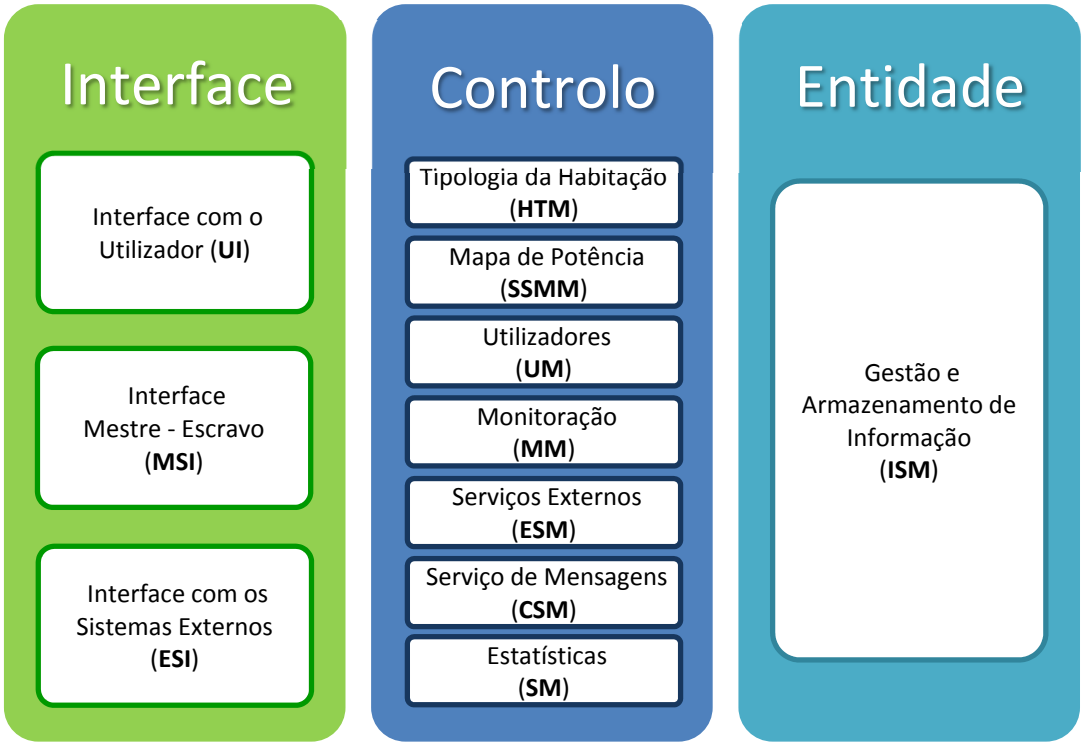


Figura 4.4 - Modelo arquitetural do mWiLOS. Estruturação em 3 camadas: Interface; Controlo e Entidade.

A camada Interface possibilita ao mWiLOS comunicar com o mundo exterior, definindo como este interage com o utilizador e com os outros sistemas. A camada Controlo descreve o modo de funcionamento interno do mWiLOS. Este consiste na satisfação dos pedidos recebidos através da camada Interface ou na utilização dos recursos existentes na camada Entidade para a execução de determinadas operações. A camada Entidade possui o reportório de toda a informação que circula no WiLOS. Cabe a esta gerir e armazenar todos os dados tratados pelo sistema.

Interface

As interfaces disponibilizadas pelo mWiLOS permitem que o utilizador interaja com o sistema de duas formas, direta e indireta. Uma interação direta ocorre sempre que um utilizador deseje visualizar alguma informação do sistema ou requisite a utilização de uma funcionalidade. Este tipo de interação com o utilizador é proporcionado por uma interface gráfica implementada pelo módulo "Interface com o Utilizador" (*User Interface*).

A interação indireta é resultante dos mecanismos de monitoração do WiLOS, que atuam de forma automática e transparente ao utilizador. Isto implica a existência de uma comunicação periódica entre o mWiLOS e o sWiLOS, de modo a se inferir a localização atual do dispositivo móvel. Para além disso, o sWiLOS pode efetuar pedidos de informação ao mWiLOS, de acordo com as necessidades do utilizador. A ligação que suporta a troca de informação entre os dois subsistemas é efetuada pela "Interface Mestre-Escravo" (***M**aster-**S**lave **I**nterface*).

Grande parte da informação adquirida pelo WiLOS tem aplicação noutras áreas, podendo ser utilizada por outros sistemas para gerar novo conhecimento. Para evitar que o WiLOS se torne num sistema fechado, a comunicação com outros sistemas é garantida através da "Interface com os Sistemas Externos" (***E**xternal **S**ystems **I**nterface*).

Controlo

No subcapítulo 4.1.1.1 verificou-se possível a distribuição das diversas funcionalidades do mWiLOS por 7 grupos distintos. Assim, a camada Controlo encontra-se estruturada em 7 módulos, onde cada um deles é responsável por disponibilizar e implementar as funcionalidades específicas de cada grupo.

O módulo "Tipologia da Habitação" (***H**ouse **T**opology **M**odule*) é responsável pela planta da habitação e todas as tarefas a ela associadas. Entre estas encontram-se a definição de divisões e a alteração da informação relacionada com a habitação (e.g., morada e contacto). O HTM recorre à UI para receber os dados introduzidos pelo utilizador, realizando o seu processamento e reencaminhado a informação para a camada Entidade. A MSI também é utilizada para satisfazer alguns pedidos referentes à tipologia da habitação realizados pelo sWiLOS.

A definição das divisões e das portas da habitação é crucial para o WiLOS. É através deste processo que se obtém o modelo da tipologia da habitação, a partir do qual se estabelece o mapa de calibração do WiLOS. Este modelo é também utilizado durante a fase de monitoração dos utilizadores, dependendo do algoritmo de localização selecionado.

As funcionalidades relativas à configuração e calibração do mapa de potência do sinal da habitação ficam a cargo do módulo "Mapa de Potência do Sinal" (***S**ignal **S**trength **M**ap **M**odule*). Este módulo usufrui da UI e da MSI para interagir com o utilizador e com o sWiLOS durante o processo de aquisição dos mapas de potência. A UI também permite a visualização dos mapas resultantes, possibilitando a análise da qualidade do sinal da rede Wi-Fi da habitação.

A monitoração dos utilizadores do WiLOS ocorre automaticamente, sem intervenção do utilizador. As funcionalidades de monitoração são implementadas no mWiLOS através do módulo

"Monitoração" (***M**onitor **M**odule*). O funcionamento do MM implica a escolha e configuração do algoritmo de localização a utilizar pelo WiLOS. Só depois é que a MSI é utilizada para questionar o sWiLOS sobre os valores de *RSSI* verificados no dispositivo móvel. O utilizador pode ainda consultar um histórico de todos os dados trocados entre o mWiLOS e o sWiLOS, de modo a validar os mecanismos de comunicação idealizados.

A localização dos utilizadores é realizada através do algoritmo k-*NNS*, que recorre aos mapas de potência recolhidos durante a fase de calibração do WiLOS. Pelo capítulo 3 sabe-se que o caso mais simples implica a existência de um mapa de potência criado somente a partir dos *APs*. Uma vez que a distribuição do sinal de um *AP* é considerado um mapa de potência, o número de mapas obtidos será proporcional ao número de *APs* existentes na rede. No entanto, vários autores efetuam o processo de calibração tendo em conta as 4 orientações cardeais. O objetivo é mitigar o erro introduzido pelo corpo humano durante a fase de monitoração, pois trata-se de uma variável desconhecida. Desta forma, o número de mapas por *AP* aumenta para 4, um por cada ponto cardeal (Norte, Sul, Este e Oeste). Esta última metodologia apresentada é a base na qual assenta o funcionamento do WiLOS, sendo denominado como "algoritmo k-*NNS*" ao longo deste documento.

Também verificou-se que nunca foi efetuado um estudo que determinasse a influência da orientação vertical e horizontal do dispositivo móvel no desempenho deste tipo de sistemas de localização. Sendo assim, em vez de se obter apenas 1 mapa de potência por orientação cardeal, obtêm-se 2 mapas de acordo com as orientações descritas (horizontal e vertical). Deste modo, um mapa de potência associado a um *AP* passa a ser representado por 8 mapas de potência. O algoritmo que recorre a todos os mapas de potência obtidos é referido como "algoritmo k-*NNS* com compensação de orientação".

Por último, os autores [55,59,60] recorrem a informação adicional, isto é, não relacionada com a potência do sinal, para melhorar a precisão dos seus sistemas. Ambas as técnicas estão relacionadas com a distância máxima que um utilizador consegue percorrer num determinado espaço de tempo. Em [55] esta informação é adicionada através de uma equação que representa o modelo de deslocação humano. Em [59,60] recorre-se à tipologia da zona em estudo para determinar locais menos prováveis para a localização futura de um utilizador. Estes trabalhos são fonte de inspiração para outra metodologia utilizada pelo WiLOS, que recorre ao modelo da tipologia da habitação para determinar mapas de distância para cada ponto de calibração. Estes mapas são utilizados para filtrar a pesquisa realizada pelo algoritmo k-*NNS*, limitando o alcance máximo que o utilizador pode percorrer. Este algoritmo denomina-se "algoritmo k-*NNS* com filtro de distância aplicado".

O UM (***U**sers **M**odule)* efetua a ponte entre as camadas Interface e Entidade, através da implementação de mecanismos que permitem adicionar, alterar e visualizar a informação relativa aos

utilizadores do WiLOS. Durante estas operações, este analisa os dados recebidos, de modo a garantir a integridade da informação. Para além disso, o UM interage com a MSI para supervisionar o processo de emparelhamento do sWiLOS com o mWiLOS. É através deste que se associa um dispositivo móvel a um utilizador, de forma a identificá-lo dentro da rede Wi-Fi da habitação. O UM também é responsável pelos mecanismos de autenticação de um utilizador, condicionando a informação disponibilizada pelo mWiLOS conforme o nível de acesso atribuído.

O módulo "Serviço de Mensagens" (***C**hat **S**ervice **M**odule*) efetua o reencaminhamento das mensagens trocadas entre utilizadores através do sWiLOS e cria cópias das conversações para mais tarde serem consultadas. As restantes funcionalidades relacionadas com este serviço também se encontram definidas neste módulo. No CSM é crucial a existência de mecanismos que garantam a escalabilidade do serviço, de forma a garantir que muitos utilizadores usufruam deste serviço em simultâneo. No entanto, o funcionamento do CSM é menos prioritário que o MM, não devendo prejudicar o mecanismo de localização do WiLOS. Como se pretende que os destinatários recebam as mensagens corretamente e rapidamente, uma boa coordenação entre a MSI, o CSM e a camada Entidade é fundamental para determinar a qualidade das funcionalidades oferecidas pelo CSM.

A disponibilização dos vários serviços do WiLOS para outros sistemas é realizada através da ESI. No entanto, cabe ao módulo "Serviços Externos" (***E**xternal **S**ervices **M**odule*) implementar e definir como é que os mesmos são acedidos. Assim, o ESM recorre à UI para que o administrador defina quais os serviços que se encontram disponíveis e quais os sistemas que podem usufruir deles. É um módulo que implementa mecanismos de segurança, de modo a garantir que a informação disponibilizada não é utilizada indevidamente.

O módulo "Estatísticas" (***S**tatistics **M**odule*) recolhe a informação existente na camada Entidade e organiza-a através da criação de elementos estatísticos para mostrar ao utilizador. O utilizador recorre à UI para indicar quais os elementos que pretende consultar e o SM encarrega-se do processamento.

Entidade

A camada Entidade é composta pelo módulo "Gestão e Armazenamento de Informação" (***I**nformation and **S**torage **M**odule*). O ISM, como o nome indica, possibilita o armazenamento de toda a informação que circula no WiLOS, e possui mecanismos que permitem uma boa gestão e a integridade de todos os dados. A camada Controlo necessita da informação registada no ISM para garantir o funcionamento dos módulos da sua camada, bem como a satisfação dos pedidos realizados na camada Interface. O ISM do mWiLOS obedece ao modelo de dados definido no subcapítulo 4.1.1.3.

4.1.1.3 Modelo de Dados – Visão Lógica

A estruturação e organização da informação são essenciais, não só para facilitar o seu acesso e consulta, como também para correlacionar diversos elementos que permitem a criação de nova informação. Assim, o modelo de dados a aplicar ao ISM é representado pelo diagrama de entidades e relacionamentos (DER) da figura 4.5. O DER pode ser repartido em 6 secções: "*House Information*"; "*User Information*"; "*Positioning*"; "*Chat Service*"; "*External Services*" e "*mWiLOS Configuration*". Esta divisão ocorre devido à natureza dos dados que são manipulados pelo WiLOS.

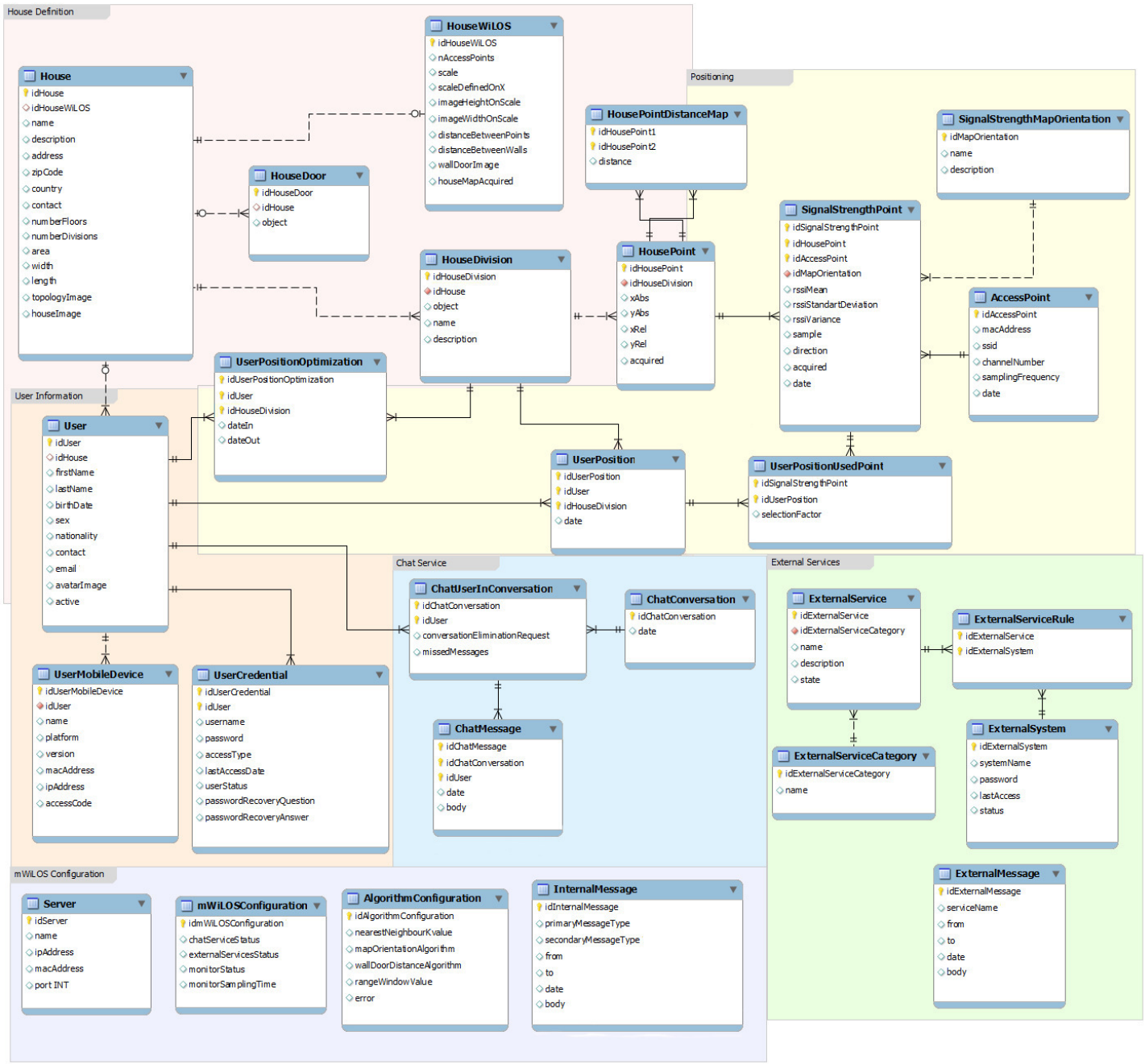


Figura 4.5 - Diagrama de entidades e relacionamentos do mWiLOS – Visão Lógica.

A área "*House Information*" é responsável pela modelação de uma habitação, através das suas características mais generalistas, como a sua morada, contactos, dimensões, e a sua tipologia, recorrendo à definição de divisões e de portas. É também nesta área que se armazenam todos os dados que permitem inferir os restantes modelos para a habitação, como o modelo discreto da habitação, composto por 'N' pontos (entidade "*House Point*"), e o modelo de distância, que efetua as ligações entre os pontos no modelo discreto (entidade "*House Point Distance Map*").

A informação referente aos *APs* da rede Wi-Fi, ao mapa de potência do sinal e à localização dos utilizadores é definida pelas entidades da área "*Positioning*". A manipulação destes dados é crucial para a fase de calibração e de monitoração do WiLOS. A identificação do número de *APs* estabelece o número de mapas de potências que fazem parte do sistema (entidade "*Access Point*"). A aquisição de cada um dos pontos de calibração, obtidos através do modelo discreto da habitação, e a sua organização possibilita a aplicação dos vários algoritmos de estimação da localização por parte do WiLOS (entidade "*Signal Strength Point*"). Por fim, as posições determinadas para cada utilizador são registadas e utilizadas para a criação de elementos estatísticos que representem a taxa de ocupação da habitação (entre outros). A localização dos utilizadores pode também ser disponibilizada para outros sistemas externos (entidade "*User Positioning*").

A área "*User Information*" possui os dados necessários para a caracterização de um utilizador e do seu dispositivo móvel. Para tal, os atributos mais generalistas, como o nome, idade e nacionalidade, são modelados pela entidade "*User*". O perfil do WiLOS, com as suas credenciais e avatar, são definidos pela entidade "*User Credential*". O dispositivo móvel fica a cargo de "*User Mobile Device*", que armazena dados como o endereço MAC, a versão do sistema operativo e o endereço IP.

Todas as mensagens trocadas entre utilizadores WiLOS ficam registadas na área "*Chat Service".* Aqui, as conversações criadas (entidade "*Chat Conversation*") possuem diversos utilizadores associados, de modo a se reencaminhar as mensagens para os destinatários corretos (entidade "*Chat Users in Conversation*"). Cada mensagem reencaminhada pelo mWiLOS fica também interligada à respetiva conversação, para possibilitar a sua consulta posterior (entidade "*Chat Message*").

Os serviços externos, e respetivos sistemas externos com privilégio para aceder à informação disponibilizada pelo WiLOS, são definidos pela área "*External Services*". A entidade "*External System*" permite efetuar a autenticação de um sistema externo, através de dados com o seu nome e palavra-passe. A associação de serviços (entidade "*External Service*") a sistemas externos é utilizada para os mecanismos de autorização, onde cada sistema só pode aceder a um determinado número de serviços pré-determinado. Esta funcionalidade é possível através da entidade "*External Service Rule*". Por fim, todas as mensagens trocadas entre WiLOS e sistemas externos são armazenadas na entidade "*External*

Message", de modo a se analisar o processo de comunicação entre sistemas e verificar a ocorrência do uso indevido da informação do WiLOS (e.g., sistemas não autorizados).

A área "*mWiLOS Configuration*" é responsável pelas configurações de alguns módulos do mWiLOS, de modo a garantir o seu funcionamento. A entidade "*Server*" define os atributos necessários para a comunicação através da MSI, enquanto a entidade "*mWiLOS Configuration*" permite ativar ou desativar o MM, o ESM e o CM, e definir o tempo de amostragem a utilizar durante a fase de monitoração. A seleção do algoritmo para a estimação da localização dos utilizadores, bem como a definição dos seus parâmetros, é efetuada através da entidade "*Algorithm Configuration*". Por fim, todas as mensagens trocadas entre subsistemas, sWiLOS e mWiLOS, ficam registadas em "*Internal Message*".

Uma maior pormenorização das escolhas efetuadas para cada entidade dentro de cada área apresentada no DER da figura 4.5 é disponibilizada no anexo D.

4.1.2 Modelo Concetual – sWiLOS

O sWiLOS é um subsistema que fornece ao mWiLOS a informação necessária para a determinação da localização de um utilizador. Pretende-se que o sWiLOS seja simples e que minimize a utilização dos recursos da plataforma móvel onde se encontra aplicado. É também obrigatório que o utilizador possua constantemente o seu dispositivo móvel enquanto se desloca dentro da habitação. Caso contrário, o sistema perde o ponto de referência que permite identificar um utilizador dentro da rede. Os mecanismos e a quantidade de informação disponibilizados pelo mWiLOS permitem que o primeiro requisito seja facilmente cumprido. Para tal, basta que o sWiLOS seja um consumidor de informação, realizando pedidos ao mWiLOS sempre que o utilizador solicitar (e.g., dados estatísticos.). Posteriormente cabe ao sWiLOS apresentar os dados recebidos ao utilizador.

A informação gerada pelo sWiLOS é bastante limitada. Um caso onde se verifica esta ocorrência é durante o processo de monitoração. Este consiste na recolha dos valores de *RSSI* verificados pelo dispositivo móvel e respetivo envio para o mWiLOS. Neste, os dados recolhidos são analisados de modo a inferir a posição atual do utilizador. Outra situação onde se gera informação é através da troca de mensagens entre utilizadores, uma funcionalidade que confere ao utilizador um papel mais ativo no WiLOS.

No entanto, um dispositivo móvel não possui recursos computacionais ao mesmo nível de um computador, tais como a velocidade de processamento computacional, gráfico ou a quantidade de memória disponível. Para além disso, estes aparelhos já se encontram a realizar outras tarefas que

consomem os seus recursos. Para minimizar a carga computacional, deve existir "bom senso" e contenção nas potencialidades que se deseja impor ao sWiLOS.

Tendo em conta as especificações desejadas para o sWiLOS, procede-se à sua modelação através do modelo funcional, arquitetural e de dados apresentados ao longo deste subcapítulo.

4.1.2.1 Modelo Funcional

Em primeiro lugar tem de se garantir que um dispositivo móvel dotado de sWiLOS seja reconhecido pelo mWiLOS através da rede Wi-Fi da habitação. Uma vez que se pretende apenas identificar os utilizadores registados no mWiLOS, não se deve aceitar qualquer dispositivo móvel que se encontre no alcance da rede. Desta forma, é necessário que o sWiLOS ofereça mecanismos de registo e autenticação para o dispositivo móvel.

Tendo em conta as circunstâncias apresentadas, uma funcionalidade de emparelhamento inicial do sWiLOS com o mWiLOS torna-se obrigatória, através da troca de informação entre os dois subsistemas, mediada pelo utilizador. Esta tarefa transfere automaticamente para o dispositivo móvel a informação necessária para o que o sWiLOS se autentique futuramente na rede WiLOS, sem a intervenção do utilizador (e.g., um código de acesso). Alguma informação adicional também é transferida para garantir as funcionalidades mínimas do sWiLOS (e.g., utilizadores registados no WiLOS e dados da habitação). Sempre que o sWiLOS se autentique com sucesso e se encontre dentro do alcance da rede WiLOS, o utilizador passa a ser alvo de monitoração e pode usufruir das restantes funcionalidades.

Tal como para o mWiLOS, a natureza das funcionalidades permite distribuí-las em 5 grupos, "Definir o Mapa de Potência do Sinal", "Monitorar", "Gerir Utilizadores", "Usufruir do Serviço de Mensagens" e "Aceder a Estatísticas".

Durante a configuração inicial do WiLOS é necessária a aquisição do mapa de potência do sinal da rede habitacional. A obtenção deste mapa pode ser efetuada de diversos modos, no entanto define-se que o processo de calibração do WiLOS ocorre através da interação entre mWiLOS, sWiLOS e administrador. Desta forma, pretende-se que o administrador se desloque ao longo da habitação com o sWiLOS na sua posse. Em cada ponto do mapa de calibração é efetuada a análise dos valores de *RSSI* verificados pelo dispositivo móvel, sendo os resultados obtidos enviados para o mWiLOS, que procede ao seu armazenamento. A figura 4.6 ilustra o processo de calibração idealizado pelo WiLOS.

A precisão do mapa de potência obtido depende da interação do administrador com o sWiLOS e o mWiLOS, bem como da sua capacidade para se colocar sobre os pontos de calibração de forma adequada. Caso um ponto fique mal calibrado, o administrador pode voltar a executar o processo de

aquisição de dados somente para o ponto em questão. Como se pode verificar, a fase de calibração implica que o mWiLOS forneça ao sWiLOS o mapa de calibração da habitação, de modo a ser consultado pelo administrador sempre que necessário. Estas funcionalidades enquadram-se no grupo "Definir o Mapa de Potência do Sinal".

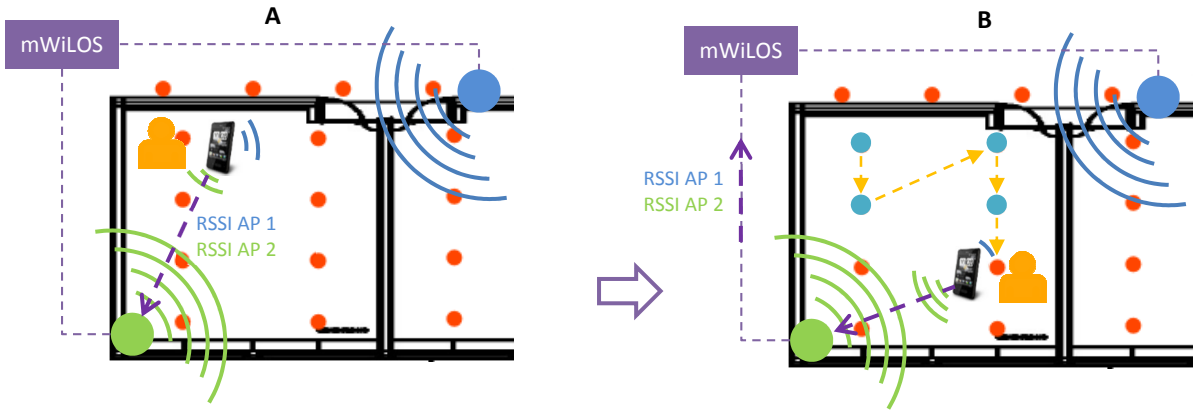


Figura 4.6 - (**A**): Início do processo de calibração no 1º ponto da habitação e envio dos valores de *RSSI* recolhidos para o mWiLOS. (**B**): Execução da calibração do sistema ao longo da divisão.

A monitoração do sWiLOS consiste na resposta a pedidos de informação periódicos, efetuados pelos mWiLOS, relativos aos valores de *RSSI* verificados pelo dispositivo móvel. O utilizador pode consultar o resultado da monitoração em qualquer instante, de modo a visualizar a posição atual de cada utilizador do WiLOS. Este mecanismo implica a execução de um pedido de informação ao mWiLOS.

As funcionalidades relacionadas com a gestão de utilizadores, para além de incluírem os mecanismos de registo e autenticação de um dispositivo móvel, também possibilitam a consulta de informação dos utilizadores registados no WiLOS. Como já mencionado, alguma informação encontra-se inicialmente disponível no sWiLOS, porém, caso se deseje detalhar ou atualizar esse conhecimento, o sWiLOS tem de recorrer ao mWiLOS para obter mais informação.

A troca de mensagens entre utilizadores implica que o sWiLOS disponibilize ferramentas para a criação de uma conversação, tais como a adição de utilizadores à mesma. Também tem de garantir a escrita, envio e respetiva visualização de mensagens através do dispositivo móvel. Ao mWiLOS compete a gestão, armazenamento e reencaminhamento das mensagens para os destinatários corretos. O utilizador pode ainda recorrer ao sWiLOS para consultar as conversações realizadas anteriormente e requisitar a sua eliminação. Mais uma vez, estas funcionalidades implicam a troca de informação entre sWiLOS e mWiLOS.

Toda a informação estatística obtida pelo mWiLOS pode ser consultada através do sWiLOS. Para simplificar e reduzir o consumo de recursos do dispositivo móvel, o processamento computacional e

gráfico destes elementos deve ser realizado pelo mWiLOS. Deste modo, apenas o resultado final é transmitido para o sWiLOS. Algumas funcionalidades disponibilizadas pelo sWiLOS encontram-se representadas nos diagramas de casos de uso das figuras 4.7 e 4.8. As restantes funcionalidades estão disponíveis para consulta no anexo B.

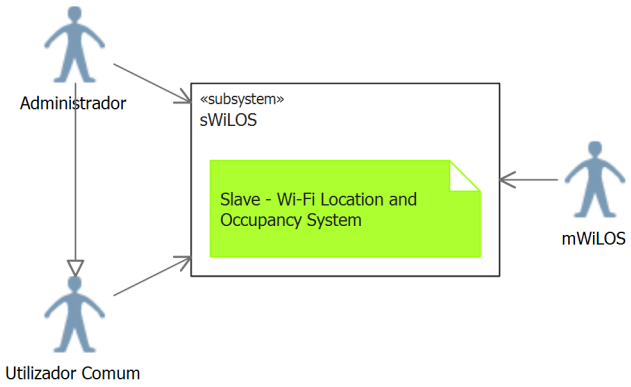


Figura 4.7 - Diagrama de casos de uso do sWiLOS – Atores que interagem com o sWiLOS.

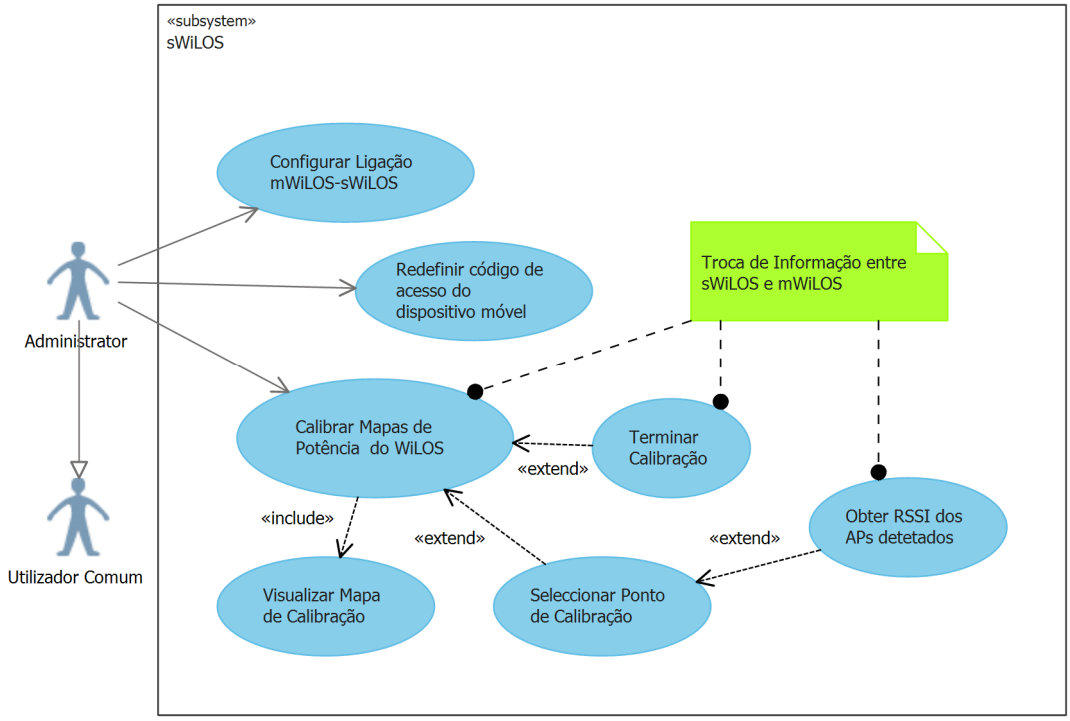


Figura 4.8 - Diagrama de casos de uso do sWiLOS – Funcionalidades oferecidas ao Administrador.

4.1.2.2 Modelo Arquitetural

A arquitetura do sWiLOS também é composta por 3 camadas: Interface; Controlo e Entidade. Os diversos módulos que as constituem encontram-se distribuídos de acordo com a estrutura apresentada na figura 4.9.

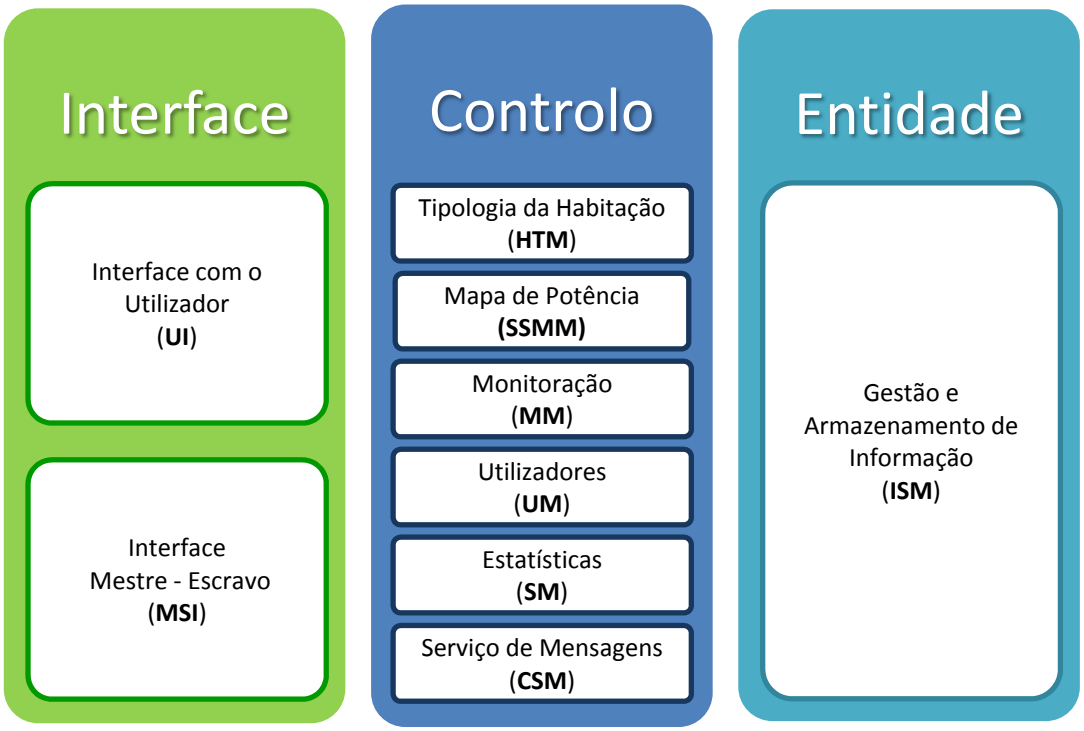


Figura 4.9 - Modelo arquitetural do sWiLOS. Estruturação em 3 camadas: Interface; Controlo e Entidade.

Interface

A camada Interface efetua a ponte entre o sWiLOS e o mundo exterior, disponibilizando as funcionalidades das camadas mais internas aos utilizadores e ao mWiLOS. Para tal, o sWiLOS implementa duas interfaces, a Interface com o Utilizador (***U**ser **I**nterface*) e a Interface Mestre-Escravo (***M**aster-**S**lave **I**nterface*).

A UI consiste numa interface gráfica que possibilita ao utilizador usufruir dos serviços disponibilizados pelos módulos da camada Controlo. Desta forma, mecanismos de visualização de informação, navegação, inserção de dados e escolha de opções, permitem que o utilizador interaja com o WiLOS de uma forma direta.

A comunicação com o mWiLOS é efetuada através da MSI. Esta interface é de grande importância para o sWiLOS, não só para garantir a monitoração do utilizador, como também para obter informação não existente no sWiLOS ou requisitar a atualização dos dados existentes. Todas as interações

realizadas por esta interface são de natureza indireta. Tanto o processo de monitoração, que ocorre automaticamente, como a comunicação efetuada pela MSI, resultante da interação do utilizador com a UI, ocorrem de forma transparente ao utilizador.

Controlo

As funcionalidades oferecidas pela camada Controlo são acedidas através da camada Interface. Assim, a camada Controlo possui como principal função satisfazer os pedidos e as necessidades do utilizador. Também deve conter mecanismos que permitam ao sWiLOS alcançar o seu objetivo, isto é, enviar ao mWiLOS os dados que possibilitem determinar a localização do dispositivo móvel.

Dos módulos que compõem esta camada, os módulos "Tipologia da Habitação" (***H**ouse **T**opology **M**odule*), "Serviço de Mensagens" (***C**hat **S**ervice **M**odule*) e "Estatísticas" (***S**tatistics **M**odule*) são aqueles que implementam exclusivamente mecanismos que permitem satisfazer os pedidos do utilizador.

Cada um destes módulos possui associado um dos grupos de funcionalidades descritos no subcapítulo 4.1.2.1. A associação do nome dos grupos ao respetivo módulo é efetuada de forma natural. Assim, o HTM permite consultar a informação relacionada com a habitação, incluindo a sua tipologia. O CSM é responsável pelo funcionamento do serviço de mensagens, através da gestão de conversações, envio e receção de mensagens. O SM reencaminha o pedido do utilizador referente a um determinado dado estatístico para a MSI (camada Interface) e aguarda uma resposta do mWiLOS. Após a sua receção, o SM efetua o seu processamento e redireciona-a para a UI, de modo a ser visualizada pelo utilizador.

Os restantes módulos, "Utilizadores" (***U**sers **M**odule*), "Mapa de Potência" (***S**ignal **S**trength **M**ap **M**odule*) e "Monitoração" (***M**onitor **M**odule*) possuem, maioritariamente, funcionalidades de configuração ou funcionamento interno do sWiLOS. O UM, para além de permitir a consulta e atualização dos dados dos utilizadores, implementa mecanismos para o emparelhamento do dispositivo móvel e posterior autenticação e ligação automática à rede do WiLOS. O MM gere automaticamente a MSI de forma a efetuar a monitoração dos sWiLOS através da resposta aos pedidos do mWiLOS. O SSMM é utilizado somente por um administrador do WiLOS caso se deseje obter o mapa de potência da habitação, ou melhorar a sua qualidade.

Entidade

Esta camada é exclusivamente composta pelo módulo "Gestão e Armazenamento de Informação" (***I**nformation and **S**torage **M**odule*). De um modo semelhante ao ISM do mWiLOS, este gere e armazena

a informação necessária para o funcionamento do sWiLOS. Estes dados são acedidos pelos módulos da camada Controlo que, a partir de mecanismos automáticos ou da interação com o utilizador ou o mWiLOS, podem ser modificados, visualizados ou utilizados para gerar novo conhecimento. O modelo de dados estabelecido no subcapítulo 4.1.2.3 é utilizado para estruturar e organizar a informação disponibilizada pelo ISM do sWiLOS.

4.1.2.3 Modelo de Dados – Visão Lógica

O esquema de dados que suporta a informação do ISM do sWiLOS respeita o diagrama de entidades e relacionamentos presente na figura 4.10.

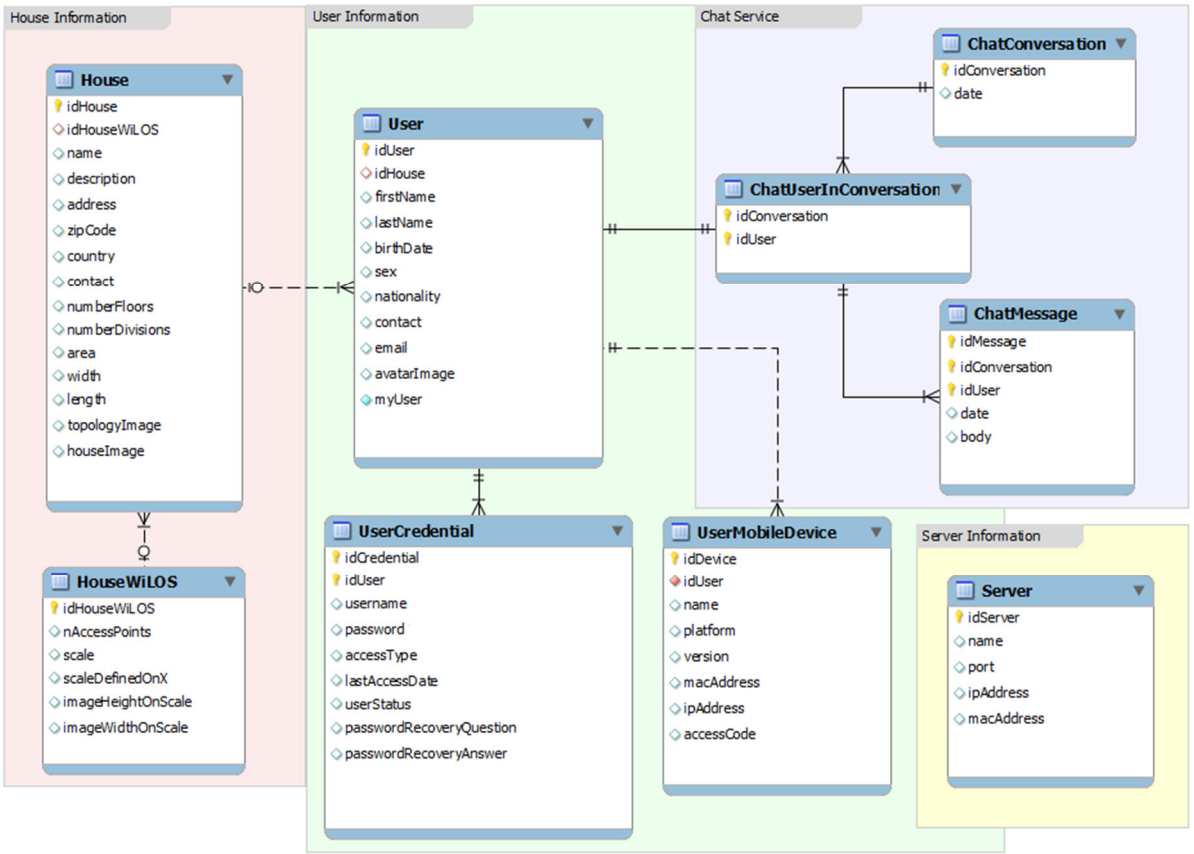


Figura 4.10 - Diagrama de entidades e relacionamentos do sWiLOS – Visão Lógica.

Como se pode observar, o DER aqui apresentado consiste numa simplificação do DER definido no subcapítulo 4.1.1.3, referente ao modelo de dados do mWiLOS. Uma vez que o sWiLOS só recebe informação proveniente do mWiLOS, a reutilização da mesma estrutura de informação é uma opção lógica para manter a integridade da informação original. Desta forma, as entidades apresentadas na DER da figura 4.19 são semelhantes, senão iguais, às entidades definidas no modelo de dados do mWiLOS.

A área "*House Information*" é composta pelas entidades "*House*" e "*House WiLOS*", permitindo ao sWiLOS o armazenamento da informação mais generalista da habitação e a manipulação das imagens fornecidas pelo mWiLOS. A entidade "*House WiLOS*" é uma simplificação da entidade "*House WiLOS*" do mWiLOS, uma vez que o sWiLOS não oferece mecanismos para a obtenção de modelos habitacionais.

O registo de utilizadores pertencentes ao WiLOS é efetuado pela entidade "*User*", da secção "*User Information*". Esta, para além de caracterizar um utilizador através do seu nome, data de nascimento, sexo, etc., define o atributo "*myUser*" que permite identificar qual dos utilizadores é proprietário do dispositivo móvel. As restantes entidades desta área, "*User Credential*" e "*User Mobile Device*", são apenas utilizadas para modelar o perfil de autenticação no WiLOS e os dados do dispositivo móvel associados ao utilizador de um dispositivo. Durante o processo de autenticação na rede, alguns elementos destas entidades são utilizados para validar o dispositivo móvel que está a tentar aceder ao mWiLOS (e.g. "*username"* de "*User Credential*" e "*macAddress"* de "*User Mobile Device*").

A informação referente aos utilizadores registados no WiLOS é de grande importância para o funcionamento de vários módulos do sWiLOS, sendo um deles o CSM. Note-se que só é possível iniciar uma conversação e trocar mensagens caso o(s) destinatário(s) sejam identificados e adicionados à conversação. Isto só é possível se o sWiLOS possuir a informação referente aos utilizadores que se encontram disponíveis para efetuar uma conversação. A associação de utilizadores a uma conversação é efetuada através da entidade "*Chat User In Conversation*". Esta entidade, ao contrário de "*Chat User in Conversation*" do mWiLOS, não possui informação referente às mensagens perdidas ou ao estado de eliminação de uma conversação. Afinal, a gestão do serviço de mensagens é uma funcionalidade exclusiva do mWiLOS.

A conjugação das entidades "*Chat Conversation*", "*Chat User In Conversation*" e "*Chat Message*" permite ao sWiLOS armazenar conversações. No entanto, não se pretende que este histórico seja composto por um vasto reportório. Este é utilizado para registar somente uma cópia das conversações atuais e respetivas mensagens. Após uma conversação expirar, e caso a mesma continue aberta, uma nova conversação é automaticamente gerada pelo mWiLOS e transmitida para o sWiLOS, para se efetuar a sua substituição. A conversação anterior não é eliminada de todo do sistema, apenas do sWiLOS. O mWiLOS contém sempre um registo de todas as conversações realizadas para consulta futura, desde que o utilizador não tenha requerido a sua eliminação.

Para finalizar, a entidade "*Server*" reúne todas as características que permitem a comunicação entre o sWiLOS e o mWiLOS. A definição do endereço IP possibilita ao sWiLOS localizar o mWiLOS na rede Wi-Fi da habitação. Por sua vez, a informação referente à porta da rede é utilizada para se aceder

à MSI do mWiLOS, de modo a garantir a comunicação. O valor do endereço MAC e do nome atribuído ao mWiLOS permite ao sWiLOS autenticar o Mestre do sistema. Afinal, alguém pode ter introduzido na rede um mestre falso com o intuito de recolher informações privadas acerca dos utilizadores. Desta forma, utilizam-se os dados mencionados para se verificar a autenticidade do mWiLOS, do mesmo modo que o mWiLOS efetua o processo de autenticação do sWiLOS.

4.1.3 Implementação – mWiLOS

Este subcapítulo descreve a fase de implementação do modelo concetual que define o mWiLOS. Aqui são referidas todas as tecnologias utilizadas para o seu desenvolvimento, bem como todas as ferramentas fornecidas por outros autores que contribuem para a sua concretização. A pormenorização da arquitetura do mWiLOS demonstra como é que as funcionalidades propostas foram implementadas, e o modelo comportamental é utilizado para detalhar o funcionamento interno dos módulos mais importantes do mWiLOS.

4.1.3.1 Tecnologias Adotadas

A principal plataforma de desenvolvimento utilizada neste projeto foi o Microsoft Visual Studio 2012 (MVS 2012). As potencialidades oferecidas por este são imensas, incluindo ferramentas de modelação UML e permitindo o desenvolvimento de aplicações através de linguagens de programação reconhecidas pela Microsoft. A linguagem C# foi escolhida como linguagem base para implementar a arquitetura do mWiLOS, com exceção do módulo “Interface com os Utilizadores”. Este último foi desenvolvido a partir da linguagem XAML, acrónimo de *eXtensible Application Markup Language*. Ambas as linguagens fazem parte da plataforma “.NET” da Microsoft, sendo o C# uma linguagem orientada a objetos baseada no C++ e o XAML uma linguagem descritiva baseada em XML. Para além da implementação do mWiLOS, o MVS 2012 foi também utilizado para se traçarem os diagramas de casos de uso que se encontram ao longo deste documento. Em termos de compatibilidade, o mWiLOS foi desenvolvido para ser executado em Windows 7 ou 8, desde que a resolução utilizada seja superior a 1024x760.

A gestão e organização de toda a informação do mWiLOS é realizada através de bases de dados (BD). Para tal, o mWiLOS interage com um sistema de gestão de bases de dados (SGBD) do tipo MySQL, fornecido pelo conjunto aplicacional XAMPP. A comunicação entre o mWiLOS e a BD é efetuada através do MySQL Connector/.NET que funciona como um *driver* para a BD, de modo a garantir a troca de informação. Para auxiliar o processo de criação e validação da BD recorre-se à plataforma MySQL Workbench (versão 5.2.34). Este *software* possui uma interface gráfica intuitiva que permite a administração das BDs MySQL existentes, bem como a criação de DERs que podem ser facilmente

convertidas em BDs. Os DERs visualizados ao longo deste trabalho foram projetados a partir desta ferramenta.

A comunicação entre o mWiLOS e o sWiLOS é efetuada através do protocolo TCP/IP. Esta opção deve-se à familiaridade com o protocolo em questão e sua popularidade. Para além disso, é um protocolo orientado à ligação que garante que os dados enviados são todos entregues ao destinatário na ordem em que foram enviados. A sua característica de encaminhamento, isto é, entrega de pacotes ao destinatário independentemente do caminho efetuado, é uma grande vantagem tendo em conta as características dinâmicas da rede WiLOS.

A disponibilização de serviços a sistemas externos ao mWiLOS é efetuada através de *webservices*. Esta solução é ideal uma vez que permite o acesso a qualquer aplicação, independentemente da linguagem ou recursos por ela utilizados. É um mecanismo de interoperabilidade muito eficaz que utiliza tecnologias padronizadas (HTTP, XML, SOAP, etc.) para efetuar a troca de informação entre sistemas via Internet, e, neste caso, através da rede WiLOS. A acessibilidade dos *webservices* tem de ser garantida por um servidor compatível com a linguagem utilizada para os implementar. Como estes são definidos recorrendo a tecnologias da Microsoft, os *webservices* do mWiLOS encontram-se alojados no servidor IIS (*Internet Information Services*) fornecido com o Windows 7 ou 8 (Profissional ou Ultimate). Com a informação adequada, um sistema externo consegue facilmente aceder a estes serviços, desde que se encontre devidamente autorizado e autenticado.

Na figura 4.11 é apresentada a configuração do WiLOS e os seus intervenientes. Nesta também se representam as tecnologias utilizadas para a implementação de todo o sistema (mWiLOS e sWiLOS).

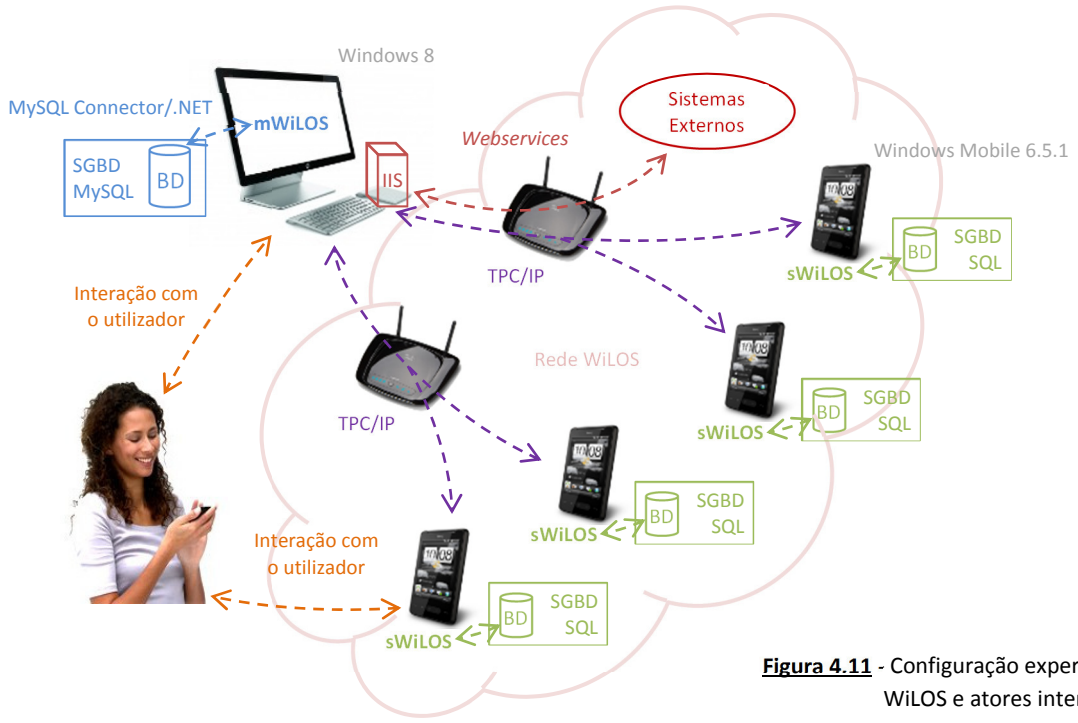


Figura 4.11 - Configuração experimental do WiLOS e atores intervenientes.

4.1.3.2 Pormenorização do Modelo Arquitetural

O mWiLOS é uma aplicação WPF (*Windows Presentation Foundation*) estruturada em dois níveis distintos: o nível de apresentação, dado pela interface gráfica implementada em XAML, e o nível comportamental, representado pelos procedimentos e funções desenvolvidos em C#. A interligação entre os dois níveis é realizada a partir de eventos. Estes podem resultar da interação do utilizador com a interface gráfica ou do processamento interno do mWiLOS, que obriga a sua execução. Desta forma, o módulo UI da camada Interface corresponde ao nível de apresentação da aplicação WPF. A camada Controlo e os restantes módulos da camada Interface são implementados pelo nível comportamental.

A camada Entidade, e o módulo ISM, são implementados pelo *namespace* "_mWiLOS_Database" e pela BD do mWiLOS. Este *namespace* define todas as classes e funcionalidades que permitem aceder e gerir a BD. A BD resulta da concretização da DER do subcapítulo 4.1.1.3. A comunicação entre o mWiLOS e o SGBD MySQL é efetuada através do MySQL Connector/.Net.

Estrutura da Camada Interface

O mWiLOS possui mecanismos que permitem ao utilizador aceder e manipular a informação que circula no WiLOS. Este tipo de interação entre o utilizador e o mWiLOS é efetuada através de uma interface gráfica moderna e agradável, implementada através do módulo UI. A interface gráfica tem como base a estrutura definida pela classe "_Interface", que pode ser consultada no anexo F, assim como os restantes *layouts* gráficos disponibilizados para os restantes módulos do mWiLOS.

Se o WiLOS estiver configurado, a execução do mWiLOS apresenta ao utilizador a página de boas-vindas, definida pela classe "_H_Home." Aqui, as únicas opções disponíveis para o utilizador são a sua autenticação ou a saída da aplicação. A zona de autenticação é implementada pela classe "_H_Logger" e permite redirecionar o utilizador para a classe "_H_Login" ou entrar como "Visitante", de modo a visualizar o funcionamento geral do sistema. Na classe "_H_Login" o utilizador pode introduzir as suas credenciais, autenticando-se no sistema, ou requisitar a recuperação da sua palavra-passe. Esta última opção redireciona o utilizador para a classe "_H_RecoverPassword", onde se recorre ao nome de utilizador e à sua pergunta secreta para gerar uma nova palavra-chave.

Caso o WiLOS nunca tenha sido configurado, a classe "_H_ConfirmAdmin" é utilizada para se certificar que apenas um utilizador com privilégios de "Administrador" acede ao sistema, de modo a concluir o processo de configuração. Todas as classes supracitadas, bem como aquelas utilizadas para definir a estrutura da interface gráfica, encontram-se agrupadas no *namespace* "_Interface",

apresentado na figura 4.12. Os estilos e tipos de letra utilizados para melhorar a aparência da interface gráfica encontram-se definidos pelos dicionários XAML "_styles" e "_comboBoxStyles".

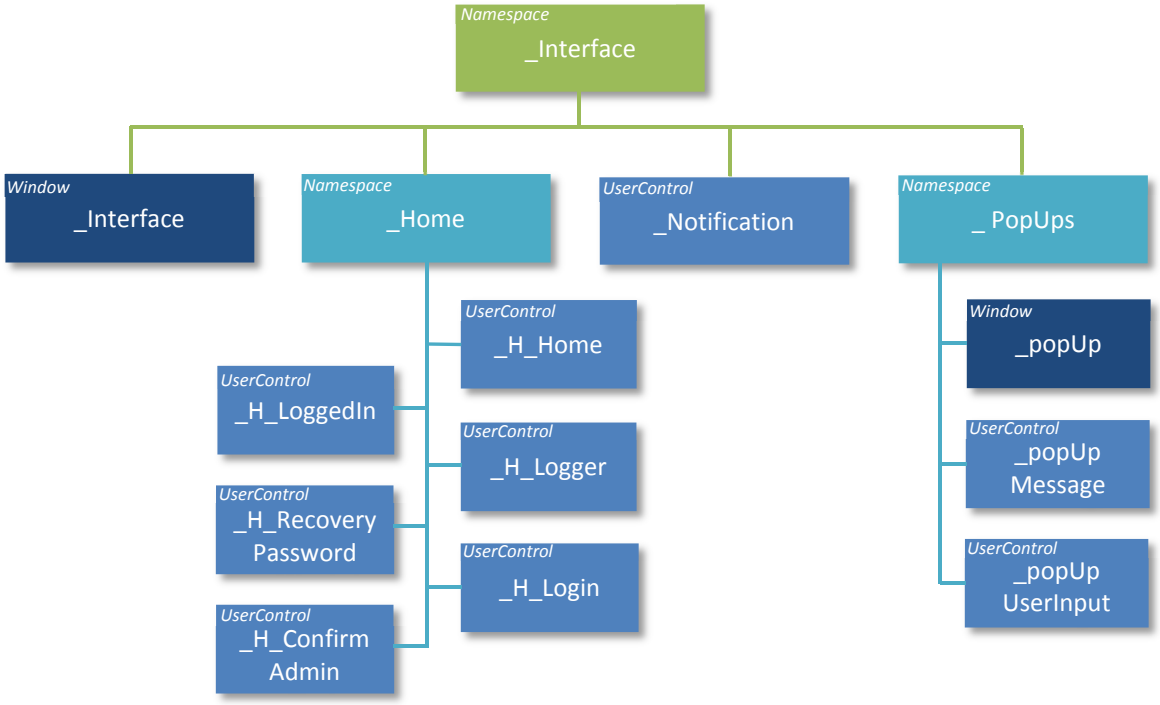


Figura 4.12 - Organização das classes que definem o UI da camada Interface do mWiLOS.

As classes "_H_Login", "_H_RecoverPassword", "_H_ConfirmAdmin" apenas implementam o conteúdo que permite a visualização e introdução de dados por parte do utilizador durante o processo de autenticação. A janela que é utilizada para apresentar o conteúdo destas classes é disponibilizada pela classe "_popUp". Esta janela de menores dimensões destaca-se da interface gráfica principal de modo a chamar a sua atenção perante o utilizador. Para além de apresentar os mecanismos de autenticação, também é utilizada ao longo da aplicação para questionar tomadas de decisão do utilizador, mostrar mensagens relacionadas com o funcionamento do mWiLOS (classe "_popUpMessage") ou pedir a inserção de dados para efetuar uma determinada operação (classe "_popUpUserInput").

Durante a execução do mWiLOS podem ocorrer eventos interessantes de se evidenciar, como o facto de um dispositivo móvel de um utilizador ter acabado de entrar na rede. A visualização deste tipo de informação é uma funcionalidade extra implementada pela classe "_Notification". Deste modo, uma pequena janela aparece no canto inferior direito durante alguns segundos para apresentar uma ou outra informação mais pertinente, sem influenciar a janela de interação principal do mWiLOS. Tanto o mecanismo de notificações, como o ícone da barra de tarefas do mWiLOS, recorrem à biblioteca "WPF-NotifyIcon", disponibilizada pelo autor Philipp Sumi em www.hardcodet.net.

A MSI permite a comunicação entre mWiLOS e sWiLOS através de duas classes: "_server" e "_communicationProtocol". A classe "_server" implementa as funcionalidades típicas de um servidor TCP/IP. Deste modo, permite aguardar a ligação de novos clientes, efetuar pedidos de informação ou configuração e analisar eventuais pedidos realizados pelo sWiLOS. O servidor TCP/IP está implementado de forma a garantir um *socket* e um *semáforo* por cada dispositivo móvel que se ligue ao servidor. A classe "_communicationProtocol" define todos os tipos de mensagens que podem ser trocadas entre mWiLOS e sWiLOS. O modo de funcionamento da MSI é ilustrado na figura 4.13.

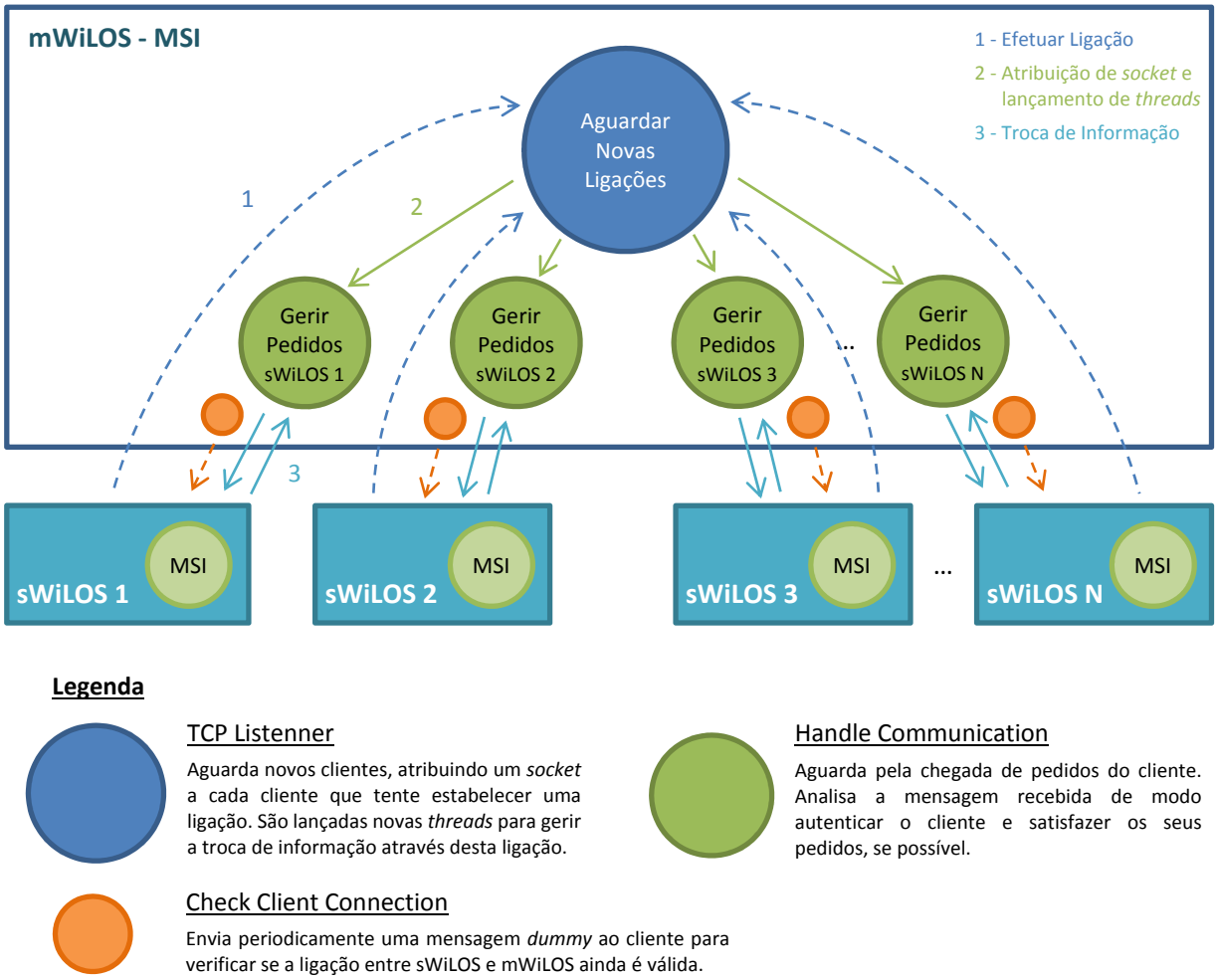


Figura 4.13 - Modo de funcionamento da camada MSI do mWiLOS. Visualização das *threads* implementadas e do processo de atribuição de *sockets* a cada dispositivo móvel sWiLOS.

O detalhar do protocolo de comunicação implementado e da sequência de mensagens a realizar, apesar de crucial para o funcionamento do WiLOS, é apenas efetuado no subcapítulo 4.1.4, referente à implementação do sWiLOS. Esta opção permite maior foco no desenvolvimento do mWiLOS e nos mecanismos utilizados para se efetuar a localização dos utilizadores e a disponibilização dos seus serviços para outros sistemas.

O módulo ESI é responsável por implementar e disponibilizar os serviços externos do WiLOS. A implementação dos serviços externos é realizada num projeto diferente daquele que define a aplicação do mWiLOS propriamente dita. A razão para a separação de projetos encontra-se na disparidade das tecnologias utilizadas por ambos. Os serviços externos do WiLOS são *webservices* que têm de ser publicados num servidor *Web*, de forma a serem acedidos pelos diversos sistemas da rede WiLOS. Por outro lado, a aplicação mWiLOS é um programa executável a partir do sistema operativo.

A separação em 2 projetos diferentes não implica a dissociação da ESI do resto da arquitetura do mWiLOS. Ao tornar o mWiLOS num consumidor dos seus próprios serviços, é possível efetuar a sincronização com os serviços disponibilizados de modo a serem geridos pelo ESM da camada Controlo. O processo de sincronismo recorre à classe "_webServiceInfo" para analisar o ficheiro WSDL (*Web Service Definition Language*), disponibilizado pelo IIS, e extrair a descrição de todos os serviços existentes. Esta informação é utilizada pelo ESM, que procede à respetiva atualização de serviços no ISM (camada Entidade) caso seja necessário.

Para além da implementação dos serviços, o ESI também é responsável pela sua disponibilização e acessibilidade. No entanto, esta funcionalidade é garantida pelo servidor IIS, local onde os serviços são publicados. Desta forma, pode-se considerar que uma parte da ESI é implementada por este servidor.

Uma vez que os dados trocados entre os sistemas podem conter informações sigilosas, como os dados dos utilizadores do WiLOS ou dos próprios sistemas, torna-se necessário definir mecanismos que garantam a segurança da informação em trânsito. Para tal utiliza-se um certificado digital SSL (*Secure Sockets Layer*) que permite implementar uma ligação segura HTTPS (*Hypertext Transfer Protocol Secure*). Esta efetua a encriptação da informação para que terceiros não sejam capazes de determinar o conteúdo das mensagens trocadas.

A obtenção de um certificado digital implica a realização de um pedido de certificado a uma Autoridade Certificadora. A qualidade do certificado vai depender da sua finalidade e do nível de segurança desejado. Por norma, maior qualidade implica um maior custo financeiro no ato de aquisição do certificado. No entanto, como se trata de um projeto académico, o certificado obtido foi gerado pela ferramenta "*Certificate Creation Tool*" do MVS 2012, fornecida pela plataforma para estas situações. A figura 4.14 apresenta a estrutura do mWiLOS nos 2 projetos referidos, "mWiLOS WPF *Application"* e "mWiLOS *Webservices",* que possibilitam a implementação dos módulos ESI e ESM pertencentes à sua arquitetura.

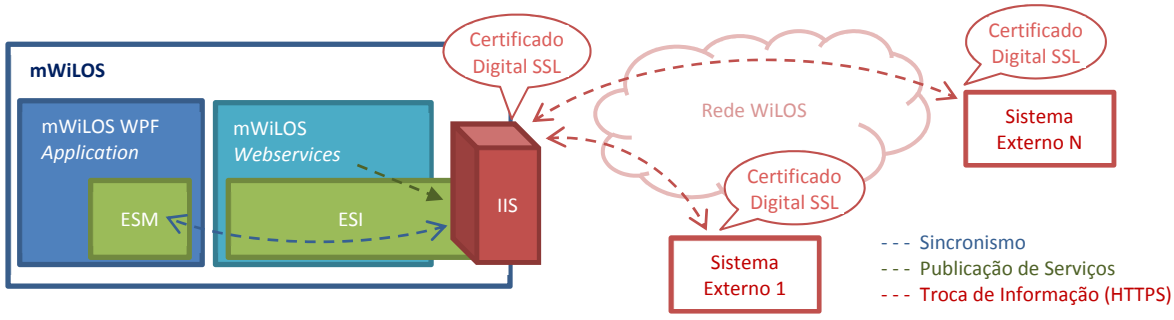


Figura 4.14 - Visualização das interações entre os diversos elementos que compõem ou interagem com os módulos ESM e ESI.

Estrutura da Camada Controlo

As funcionalidades relacionadas com a manipulação de informação da habitação são implementadas pelo HTM. Este módulo é representado pelo *namespace* "_House" que engloba as classes representadas na figura 4.15.

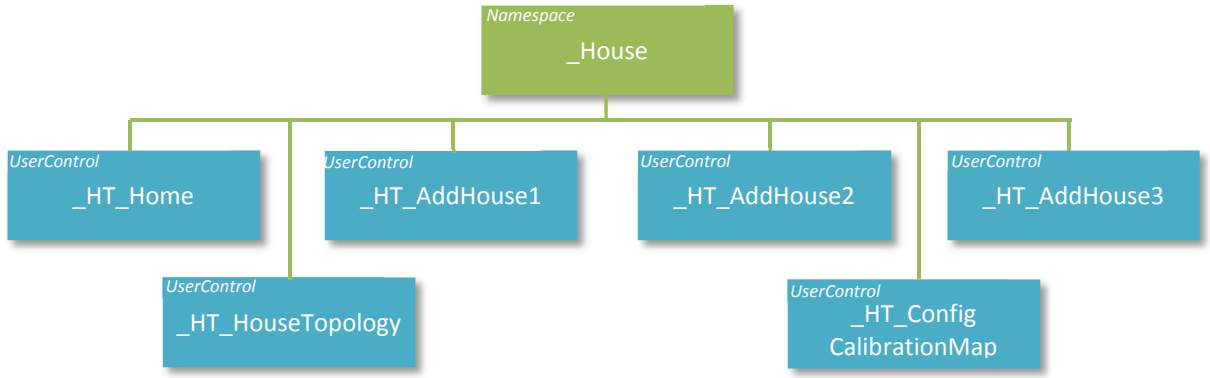


Figura 4.15 - Organização das classes que definem o HTM da camada Controlo do mWiLOS.

Durante o processo de configuração do mWiLOS, este módulo é responsável pela definição da habitação. Para tal, o utilizador descreve o perfil da habitação, através das classes "_HT_AddHouse1" e "_HT_AddHouse2", e define a sua tipologia com auxílio da classe "_HT_AddHouse3". Esta última também permite obter os modelos da tipologia da habitação, que são utilizados pelo SSMM e pelo MM para garantir as restantes funcionalidades do sistema.

A definição da tipologia da habitação implica a definição de 3 elementos: divisões; portas e escala. Durante este processo recorre-se a uma imagem da planta da habitação para auxiliar o utilizador. A adição de novas divisões é realizada a partir da opção "Add Division", que permite o traçar de um polígono, sobre a planta da habitação, de forma a respeitar o formato da divisão. A funcionalidade "Add Door" gera um polígono quadrado que pode ser movido ou redimensionado, de modo a satisfazer a localização e as dimensões da porta a modelar. A ferramenta "Define Scale" apresenta ao

utilizador todos os vértices dos polígonos que definem as divisões. Desta forma, a seleção de 2 vértices permite definir a distância que os separa em metros, de modo a fornecer ao mWiLOS um fator de escala para a toda a habitação.

Após a definição da habitação, a opção "Get Calibration Map" permite configurar os parâmetros que são utilizados para gerar os modelos da habitação (classe "_HT_ConfigCalibrationMap"). O resultado da determinação dos modelos é apresentado ao utilizador através do modelo discreto da área habitacional. Se este não for do agrado do utilizador, basta reconfigurar os parâmetros e gerar um novo mapa de calibração. A pormenorização da computação destes modelos é efetuada no subcapítulo 4.1.3.4.

A classe "_HT_Home" é apresentada ao utilizador sempre que este aceda ao HTM, e a habitação tenha sido previamente configurada. Esta possibilita visualizar ou modificar a informação referente à habitação, como a morada ou contacto, ou consultar a tipologia da habitação, sendo redirecionado para a classe "_HT_HouseTopology". A qualquer altura, o administrador pode reiniciar o processo de definição da habitação, o que implica a recalibração do mapa de potência e a perda da informação referente à localização dos utilizadores.

A gestão e aquisição dos mapas de potência do sinal da habitação são realizados pelo SSMM. O acesso a este módulo apresenta ao utilizador a classe "_SM_Home", que permite visualizar o mapa de calibração e os pontos que ainda estão por calibrar. Esta também permite a definição do número de *APs* existentes na habitação. Este valor é utilizado como ponto de referência para a criação do mapa de potência a adquirir, uma vez que não se garante a alcançabilidade de todos os *APs* em cada ponto do mapa.

A funcionalidade que permite iniciar a calibração do mapa de potência denomina-se "Start Calibration". Esta só pode ser efetuada se um dispositivo móvel sWiLOS, associado a um administrador, estiver no alcance da rede Wi-Fi da habitação. A MSI realiza o envio de uma mensagem para o sWiLOS, que o obriga a assumir a interface gráfica de calibração. O SSMM fica a aguardar a receção dos pontos de calibração que são enviados pelo sWiLOS ao longo do tempo. Durante a recolha dos pontos, tanto o mWiLOS como o sWiLOS podem interromper ou finalizar o processo de calibração, retornando cada um dos sistemas ao seu modo de funcionamento habitual.

A consulta dos vários mapas de potência do sinal, pertencentes a cada *AP*, efetua-se através da classe "_SSM_AccessPointMap". Assim, para cada *AP*, é possível observar a variação da força do sinal ao longo da habitação, em cada um dos sentidos Norte, Sul, Este e Oeste, e de acordo com as duas orientações possíveis para o dispositivo móvel, vertical e horizontal. Os mecanismos implementados

apenas permitem visualizar 1 mapa de cada vez. No entanto, o utilizador pode escolher até 4 *APs* de modo a comparar a variação da potência do sinal entre *APs*.

O módulo Utilizadores (UM) gere maioria da informação associada aos utilizadores do WiLOS. As classes que definem este módulo encontram-se agrupadas pelo *namespace* "_Users", cuja estrutura pode ser consultada na figura 4.16.

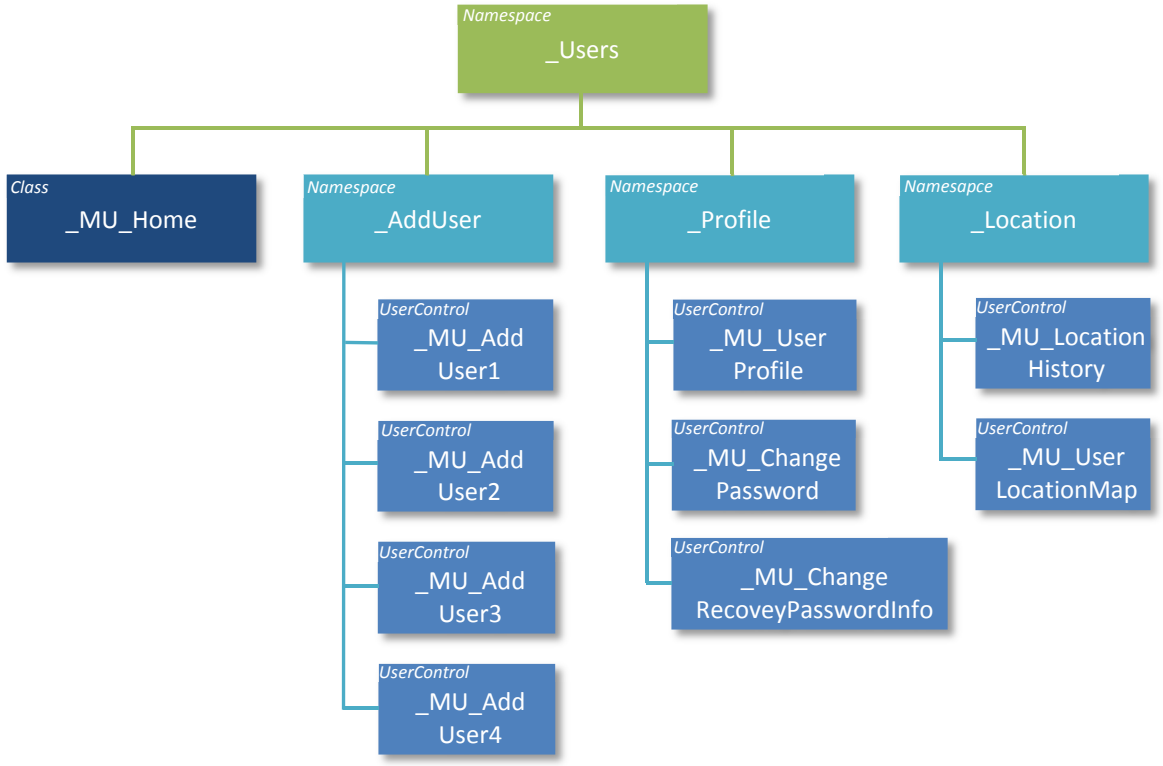


Figura 4.16 - Organização das classes que definem o UM da camada Controlo do mWiLOS.

A classe "_MU_Home" permite a visualização de todos os utilizadores registados, bem como o acesso às funcionalidades de adição de novos utilizadores, eliminação ou desativação de utilizadores existentes, consulta de perfis ou emparelhamento de dispositivos móveis. Desta forma, esta classe define a página inicial apresentada a um utilizador que consulte o UM.

A criação de um novo utilizador do WiLOS é efetuada pelas classes "_MU_AddUser1" e "_MU_AddUser2". Na primeira define-se o perfil generalista de um utilizador, através de dados como o seu nome, data de nascimento e nacionalidade. A segunda permite criar o perfil de utilizador WiLOS, de modo a atribuir a um utilizador um nome virtual, palavra-passe e nível de acesso. Uma imagem também pode ser associada ao utilizador para definir o seu avatar dentro do WiLOS. A classe "_MU_AddUser3" é utilizada para indicar se o novo utilizador foi criado com sucesso ou não. Se a resposta for positiva, é dada a opção de emparelhar um dispositivo móvel com o utilizador recém-criado.

O processo de emparelhamento de um dispositivo a um utilizador é iniciado pela classe "_MU_AddUser4". Este indica à MSI que o mWiLOS está recetível a pedidos de associação por parte do sWiLOS. Caso contrário, todos os pedidos de emparelhamento efetuados são automaticamente rejeitados pela MSI. Durante este processo ocorre troca de informação entre ambos os subsistemas, de modo a que o mWiLOS registe os dados fornecidos pelo sWiLOS que permitem identificar inequivocamente o seu dispositivo na rede. O fenómeno contrário também ocorre, isto é, o sWiLOS recebe informação que permite o seu funcionamento e acesso ao mWiLOS no futuro. A sequência da informação trocada entre mWiLOS e sWiLOS durante o emparelhamento é detalhada no subcapítulo 4.1.4.4.

A consulta dos vários perfis atribuídos a um utilizador do WiLOS é realizada a partir da classe "_UM_UserProfile". Os 3 perfis que podem ser visualizados são o perfil de utilizador, o perfil WiLOS e o perfil do dispositivo móvel. Esta classe também permite a edição dos dados associados aos perfis do utilizador, exceto o perfil do dispositivo móvel. As classes "_MU_ChangePassword" e e"_MU_ChangeRecoveryPasswordInfo" auxiliam no processo de edição de dados, através da alteração da palavra-passe do utilizador ou da pergunta e resposta utilizadas para repor a palavra-passe.

A classe "_MU_LocationHistory" permite a consulta do histórico de localização de um utilizador. Desta forma pode-se verificar todas as divisões ocupadas por um utilizador, sempre que o seu dispositivo móvel se encontrar ligado à rede WiLOS. Esta classe implementa mecanismos de filtragem de informação, de modo a serem apresentadas as localizações referentes a um determinado instante temporal, ou a consulta das alturas do dia em que o utilizador ocupou uma determinada divisão.

Caso se deseje uma maior interação com o histórico de posições de um utilizador, recorre-se à classe "_MU_UserLocationMap". Nesta, um utilizador define o intervalo temporal que deseja analisar, e o UM gera um mapa com todas as localizações do utilizador ocorridas nesse espaço de tempo. Se a informação apresentada não permitir verificar o percurso efetuado pelo utilizador, devido ao excesso de informação associado ao tamanho do intervalo, existe um mecanismo reconstrução temporal. Este, à semelhança de um vídeo, percorre todos os instantes que compõem o intervalo definido, permitindo visualizar o percurso efetuado pelo utilizador.

A monitoração dos utilizadores do WiLOS, bem como a configuração de todos os mecanismos que contribuem para esse processo, é efetuada através do MM. A visualização em tempo real da localização dos utilizadores é proporcionada pela classe "_MS_Home". A taxa de atualização da interface gráfica foi definida para um intervalo de 3 segundos. Esta classe também permite o acesso às funcionalidades de configuração implementadas pela classe "_MS_MonitorConfiguration". O *namespace* que implementa todas as classes constituintes do MM encontra-se estruturado de acordo com figura 4.17.

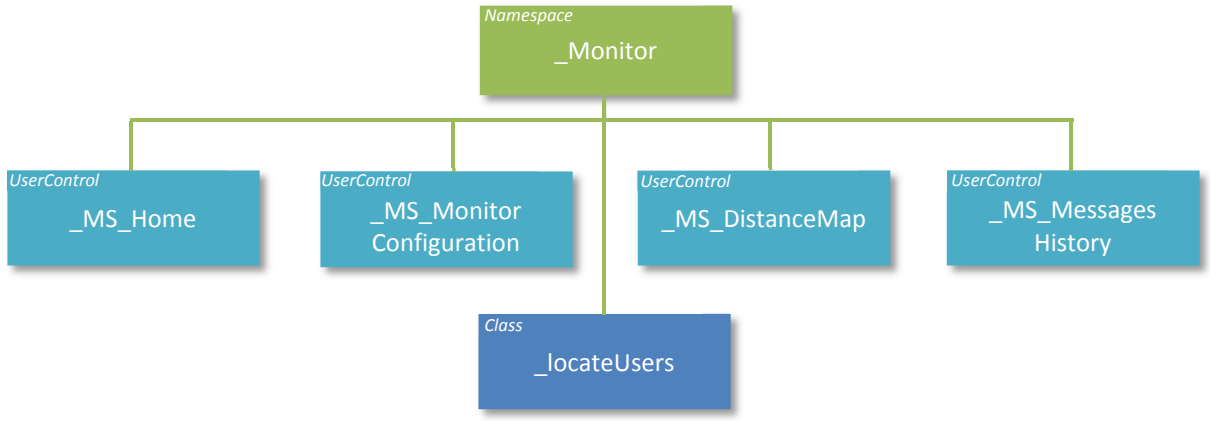


Figura 4.17 - Organização das classes que definem o MM da camada Controlo do mWiLOS.

As várias configurações do MM que podem ser realizadas são a definição do tempo de amostragem, escolha do algoritmo, ou algoritmos, a utilizar durante a monitoração, a definição do valor de 'k' do algoritmo de k-*NNS* e do valor de distância máxima que um utilizador consegue percorrer entre 2 pontos consecutivos do mapa de calibração (tendo em conta o tempo de amostragem). Como medida preventiva, esta classe também permite reiniciar a MSI caso se verifique a existência de alguma anomalia na comunicação mWiLOS-sWiLOS. A consulta de todo o histórico de mensagens trocadas entre estes 2 subsistemas é outra funcionalidade integrante do MM. À semelhança da classe "_MU_UserLocationHistory", a classe "_MS_MessagesHistory" permite visualizar e filtrar as mensagens de acordo com o seu tipo, remetente, destinatário ou data de envio.

Uma vez que o modelo de distância da habitação é utilizado pelo MM para efetuar a localização de utilizadores, optou-se por implementar os mecanismos que permitem a sua consulta no MM. Assim, a classe "_MS_DistanceMap" disponibiliza ao utilizador um mapa da tipologia da habitação com os respetivos pontos de calibração associados. A seleção de um ponto do mapa automaticamente apresenta ao utilizador todas as distâncias desse ponto aos restantes pontos do mapa de calibração. Caso o utilizador deseje, pode também visualizar a influência do filtro de distância no modelo de distância. Esta funcionalidade destaca todos os pontos que se encontram dentro da distância definida em "_MS_MonitorConfiguration", aquando da seleção de um ponto do mapa de calibração.

A classe "_locateUsers" é composta por toda a lógica e mecanismos utilizados para se efetuar a interação com a MSI, recolha dos valores de *RSSI* fornecidos pelo sWiLOS, comparação com os valores do ISM e determinação da localização dos utilizadores. A lógica por detrás desta classe é apresentada no subcapítulo 4.1.3.4, referente ao modelo comportamental do mWiLOS.

A gestão e consulta das conversações dos utilizadores do WiLOS é realizada através do CSM. A MSI recebe as mensagens trocadas entre os diversos dispositivos sWiLOS, e o mWiLOS processa a

informação através do CSM. Desta forma, o CSM cria novas conversações e efetua o redirecionamento de mensagens. As classes que definem este módulo estão organizadas segundo o *namespace* apresentado na figura 4.18.

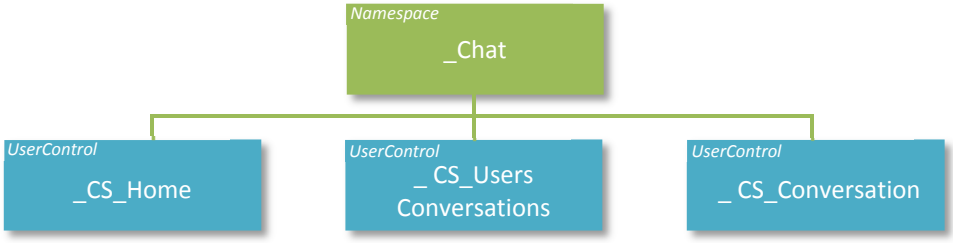


Figura 4.18 - Organização das classes que definem o CSM da camada Controlo do mWiLOS.

A classe "_CS_Home" é exclusiva do administrador, efetuando a listagem de todos os utilizadores de modo a permitir um acesso mais rápido a conversações específicas. O histórico de todas as conversações ocorridas entre 2 utilizadores do WiLOS é gerado pela classe "_CS_UsersConversations". Esta possui mecanismos de filtragem de informação, de forma a reduzir a quantidade de conversações apresentadas ao utilizador. Nesta classe, um utilizador pode também requerer a eliminação de uma conversação. Como já referido, a conversação deixa de ser acessível para o utilizador, mas a sua remoção definitiva do ISM só se efetua após a confirmação de todos os utilizadores que nela participam. A visualização de uma conversação, e de todas as mensagens trocadas, efetua-se a partir da classe "_CS_Conversation". A disponibilização de mecanismos de pesquisa permite a identificação de uma determinada expressão existente nas mensagens.

A gestão dos sistemas com privilégios para aceder aos serviços externos do WiLOS é efetuada pelo módulo Serviços Externos (ESM). Este módulo recorre à ESI para sincronizar os serviços disponíveis com os serviços registados no ISM. Desta forma é possível definir o estado de um serviço específico, bem como quais os sistemas que lhe podem aceder. O *namespace* "_ExternalServices", que implementa o ESM, encontra-se estruturado de acordo com a figura 4.19.

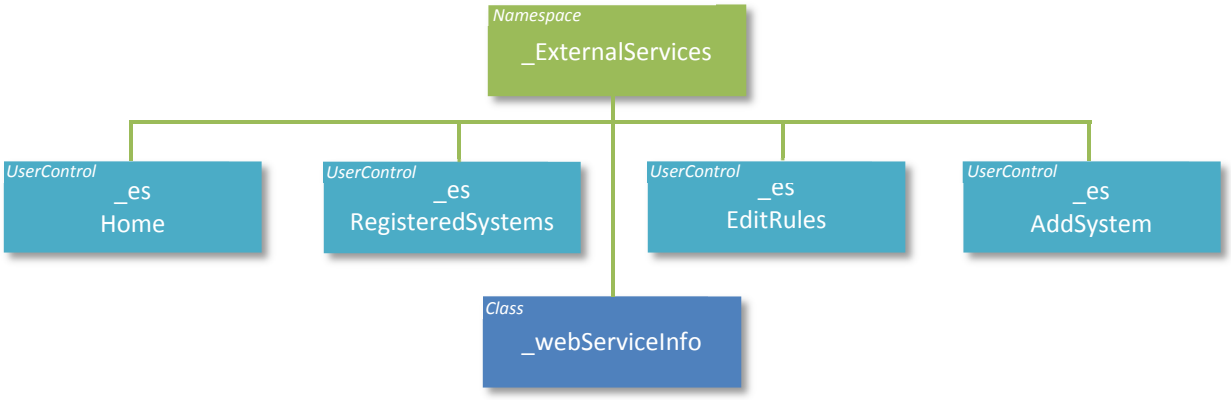


Figura 4.19 - Organização das classes que definem o ESM da camada Controlo do mWiLOS.

A classe "_esHome" implementa as funcionalidades de pedido de sincronismo e listagem de serviços. O processo de sincronismo recorre à classe "_webServiceInfo" para analisar o ficheiro WSDL e extrair a descrição de todos os serviços disponibilizados. Por sua vez, a visualização da lista de sistemas externos registados no WiLOS é efetuada pela classe "_esRegisteredSystems". Esta classe permite ainda aceder às classes "_esAddSystem" e "_esEditRules".

A adição de um novo sistema ao WiLOS, através da definição de um nome e a respetiva palavra-passe, é efetuada através da classe "_esAddSystem". Estes dados são utilizados para verificar a identidade e os privilégios de um sistema externo durante o acesso a um serviço. A classe "_esEditRules" permite ao utilizador associar ou dissociar os serviços a um sistema registado, de modo a limitar o conteúdo disponibilizado a um determinado sistema.

As informações recolhidas durante a execução do mWiLOS, ou inferidas a partir da localização dos utilizadores, são disponibilizadas pelo SM através de elementos estatísticos. A enumeração das funcionalidades deste módulo encontra-se representada num dos diagramas de casos de uso do anexo A. O *namespace* que define o SM encontra-se estruturado segundo a figura 4.20.

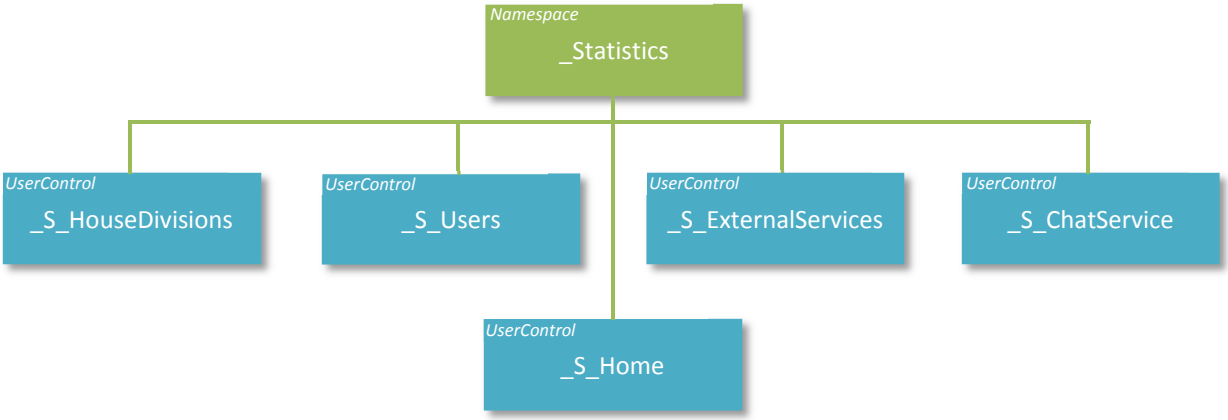


Figura 4.20 - Organização das classes que definem o SM da camada Controlo do mWiLOS.

Cada uma das classes de "_Statistics" agrupa um conjunto de dados estatísticos de acordo com o tipo de informação manipulada. A classe "_S_HouseDivisions" apresenta os dados estatísticos referentes à tipologia da habitação. Assim, informações como a divisão mais utilizada, distribuição da taxa de ocupação por divisão ou a taxa de ocupação da habitação ao longo do tempo, são apresentadas por esta classe. De um modo semelhante, a classe "_S_Users" demonstra qual o utilizador que passa mais tempo dentro da habitação, qual é aquele que percorre uma maior distância ou qual a distribuição da taxa de ocupação de um utilizador pelas divisões da habitação.

As estatísticas referentes aos serviços externos e ao serviço de mensagens são funcionalidades extras do WiLOS. A "_S_ExternalServices" apresenta informação trivial como o sistema externo que efetua mais acessos, o número de pedidos efetuados ao mWiLOS ou o serviço externo mais vezes consultado. A classe "_S_ChatService" permite a visualização do número total de mensagens enviadas, de quem é que recorre mais a este serviço dentro da habitação e da distribuição de mensagens por utilizador.

Todas as classes mencionadas são acessíveis a partir da classe "_S_Home", que define a página de entrada no SM. Esta também reúne um conjunto mínimo de dados estatísticos que permite analisar o ponto de situação do sistema e da habitação. A figura 4.39 apresenta a UI que implementa "_S_Home", bem como alguns exemplos de dados estatísticos que podem ser consultados pelos utilizadores.

Os gráficos do mWiLOS são construídos a partir da biblioteca amCharts, versão 1.0 para WPF, disponibilizada em www.amCharts.com. No entanto, a partir de 2010 a amCharts descontinuou a versão WPF e deixou de lhe fornecer suporte. Mesmo assim, não deixa de ser uma ferramenta poderosa para a criação de gráficos interativos com um visual apelativo. Atualmente, a amChart apenas fornece esta ferramenta para implementar gráficos em páginas da internet através de JavaScript ou HTML5.

Estrutura da Camada Entidade

A camada Entidade é composta pelo ISM e pela BD do mWiLOS. Todas as funcionalidades que permitem a gestão da BD através do ISM são implementadas pela classe "_mWiLOS_Database". Assim, operações básicas como a consulta de informação, inserção, atualização e remoção de dados, são executáveis através deste módulo. Para além destas operações, outras funcionalidades mais complexas e necessárias para o funcionamento do mWiLOS, como a aplicação do algoritmo k-*NNS* independentemente do número de *APs* do sistema, também são acessíveis através do ISM. As figuras 4.21 e 4.22 ilustram a classe "_mWiLOS_Database" e todas as subclasses que fazem parte da sua estrutura.

A classe "_mWiLOS_Database" define os elementos necessários para se efetuar a ligação com a BD, como o nome da BD a selecionar, a sua localização e palavra-passe. Esta também possui funcionalidades para a criação, remoção ou recriação da BD do mWiLOS. As subclasses pertencentes a "_mWiLOSDatabase" agrupam as restantes funcionalidades que permitem a manipulação das tabelas da BD, resultantes da concretização da DER do subcapítulo 4.1.1.3.

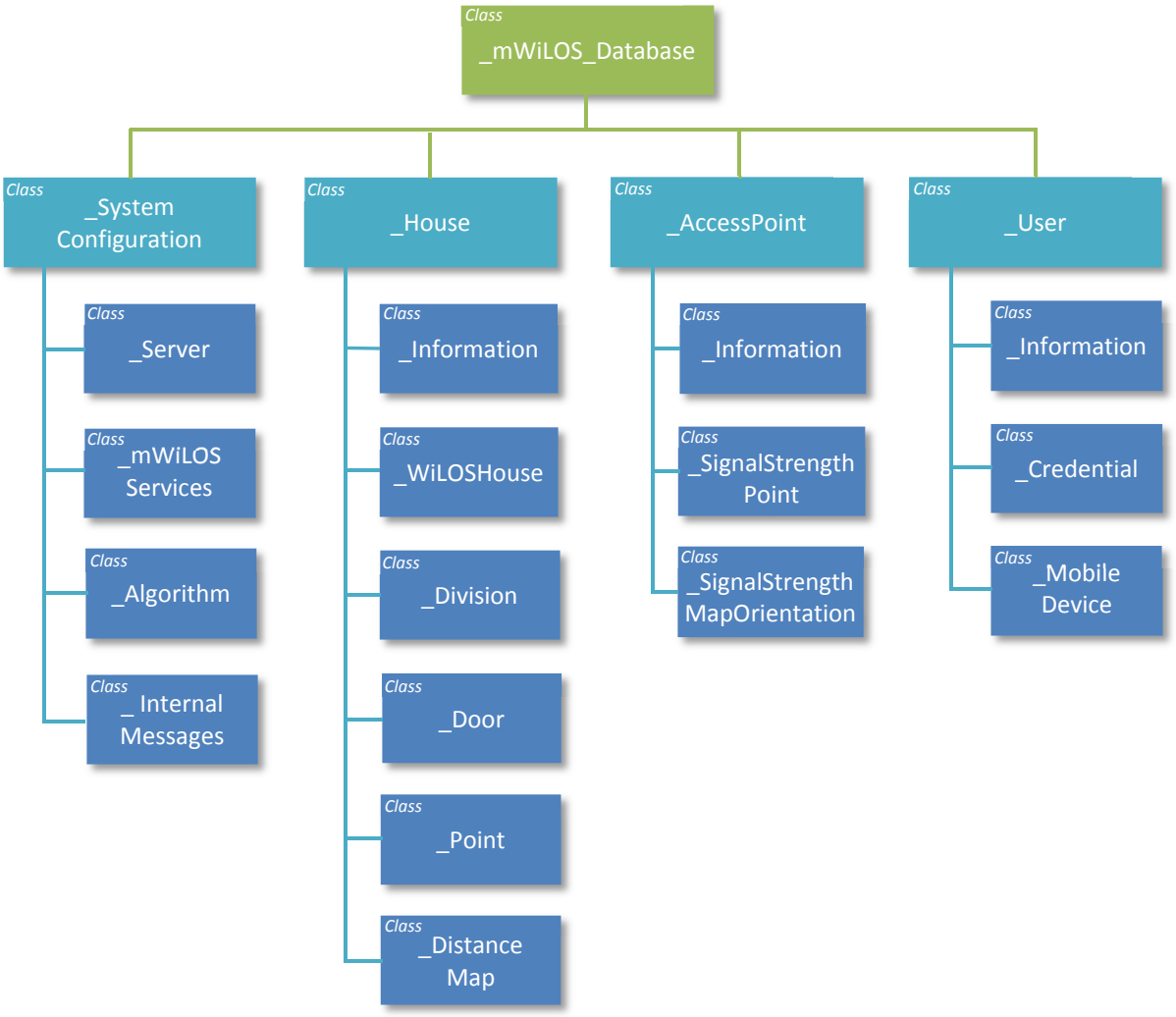


Figura 4.21 - Organização das classes que definem o ISM da camada Entidade do mWiLOS.

Estas subclasses encontram-se organizadas de forma semelhante à DER. Assim, a classe "_House" permite a manipulação das entidades "*House*", "*House WiLOS*", "*House Door*", "*House Division*", "*House Point*" e "*House Point Distance Map*". Desta forma, esta classe gere a informação referente à área "*House Information*". A área "*User Information*" é acedida através da classe "_User", que recorre às subclasses "_Information", "_Credential" e "_MobileDevice" para interagir com as entidades "*User*", "*User Credential*" e "*User Mobile Device*", respetivamente. De forma equivalente, as classes "_SystemConfiguration", "_ExternalServices" e "_ChatService" afetam as áreas "*mWiLOS Configuration*", "*External Services*" e "*Chat Service*", e as entidades que as compõem.

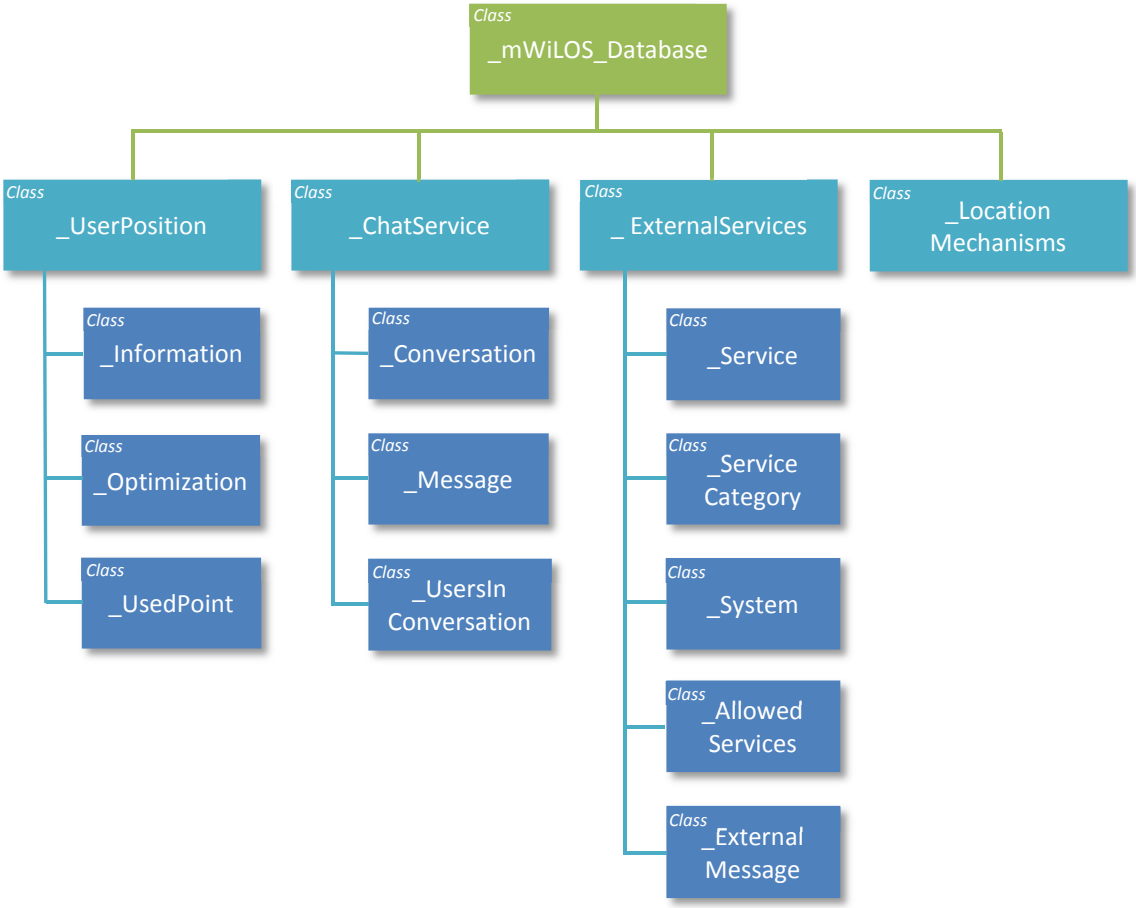


Figura 4.22 - Organização das classes que definem o ISM da camada Entidade do mWiLOS (continuação).

A manipulação das entidades que constituem a área "*Positioning*" é efetuada através de duas classes: "_AccessPoint" e "_UserPosition". Esta divisão permite a "_AccessPoint" interagir com as entidades que definem os mapas de potência do sinal para cada *AP*, nomeadamente "*Access Point*", "*Signal Strength Point*" e "*Signal Strength Map Orientation*". Os mecanismos de gestão das entidades "*User Position*", "*User Position Used Points*" e "*User Position Optimization*", relacionadas com o posicionamento de um utilizador, ficam a cargo de "_UserPosition" e das suas subclasses.

Por fim, a subclasse "_LocationMechanisms" implementa toda a lógica que permite a aplicação dos vários algoritmos, baseados no k-*NNS*, à BD através de *Queries MySQL*. A opção de implementar o próprio algoritmo de localização no ISM deve-se ao tipo de tecnologias por ele disponibilizadas, mais precisamente a linguagem MySQL e o SGBD. A linguagem MySQL, variação da linguagem SQL, é uma linguagem simples, declarativa e fácil de usar. O SGBD é responsável pela gestão das BDs e dos pedidos a elas efetuados, oferecendo mecanismos eficazes de pesquisa e afetação das várias tabelas que as constituem. Uma vez que o próprio algoritmo k-*NNS* consiste num processo de pesquisa, a utilização da combinação MySQL e SGBD permite passar parte do processamento para o SGBD, e usufruir de todas as vantagens supracitadas.

Os 4 procedimentos implementados pela classe "_LocationMechanisms" são descritos na lista seguidamente apresentada. No subcapítulo 4.1.3.4 descreve-se o comportamento de cada uma destas funcionalidades, de modo a estimar-se uma localização para a posição atual do utilizador.

_location_NN

Efetua uma pesquisa simples através de todos os pontos definidos na entidade "*Signal Strength Point*" de modo a selecionar os 'k' valores de *RSSI* com maior semelhança aos valores de *RSSI* fornecidos. Dos 'k' valores obtidos, apenas aquele que se repete mais vezes é que é retornado (repetição por divisão).

Parâmetros de Entrada

nAccessPoints: Número inteiro que indica o número de *APs* do sistema
idUser: Identificador do utilizador que se pretende localizar
infoReceived: Objeto do tipo "List<List<object>>" contendo os valores de *RSSI* associados a cada *AP* do sistema. Exemplo: { {idAccessPoint1, RSSI1} , {idAccesPoint2, RSSI2} }.

Resposta

Um objeto do tipo "List<object>" composto por: idHouseDivision, xAbs, yAbs, idSignalStrengthPoint1, selectionFactor1, idSignalStrengthPoint2, selectionFactor2, ... , idSignalStrengthPointN, selectionFactorN.

_location_NN_DeviceOrientation

Efetua uma pesquisa simples através de todos os pontos definidos na entidade "*Signal Strength Point*" de modo a selecionar os 'k' valores de *RSSI* com maior semelhança aos valores de *RSSI* fornecidos. A pesquisa apenas incide sobre os pontos que definem a orientação vertical ou horizontal do dispositivo móvel. Dos 'k' valores obtidos, apenas aquele que se repete mais vezes é que é retornado (repetição por divisão).

Parâmetros de Entrada

nAccessPoints: Número inteiro que indica o número de *APs* do sistema
idUser: Identificador do utilizador que se pretende localizar
infoReceived: Objeto do tipo "List<List<object>>" contendo os valores de *RSSI* associados a cada *AP* do sistema. Exemplo: { {idAccessPoint1, RSSI1} , {idAccesPoint2, RSSI2} }.
idMapOrientation: Identificador que representa a orientação pretendida durante o algoritmo de pesquisa

Resposta

Um objeto do tipo "List<object>" composto por: idHouseDivision, xAbs, yAbs, idSignalStrengthPoint1, selectionFactor1, idSignalStrengthPoint2, selectionFactor2, ..., idSignalStrengthPointN, selectionFactorN.

_location_NN_DistanceMap

Efetua uma pesquisa através de todos os pontos definidos na entidade "*Signal Strength Point*" de modo a selecionar os 'k' valores de *RSSI* com maior semelhança aos valores de *RSSI* fornecidos. Recorre-se também ao mapa de distância e ao valor que define a distância máxima que um utilizador consegue percorrer entre dois pontos para se filtrar os resultados. Esta função necessita de um valor de localização passado, isto é, 't-1'. Dos 'k' valores obtidos, apenas aquele que se repete mais vezes é que é retornado (repetição por divisão).

Parâmetros de Entrada

nAccessPoints: Número inteiro que indica o número de *APs* do sistema
idUser: Identificador do utilizador que se pretende localizar
infoReceived: Objeto do tipo "List<List<object>>" contendo os valores de *RSSI* associados a cada *AP* do sistema. Exemplo: { {idAccessPoint1, RSSI1} , {idAccesPoint2, RSSI2} }.
idHousePointLastPosition: O identificador do ponto do mapa de calibração referente à localização do utilizador no instante 't-1' (entidade "*House Point*")
proximityRange: Distância máxima percorrida por um utilizador entre dois pontos (metros)

Resposta

Um objeto do tipo "List<object>" composto por: idHouseDivision, xAbs, yAbs, idSignalStrengthPoint1, selectionFactor1, idSignalStrengthPoint2, selectionFactor2, ... , idSignalStrengthPointN, selectionFactorN.

_location_NN_DeviceOrientation_DistanceMap

Efetua uma pesquisa através de todos os pontos definidos na entidade "*Signal Strength Point*" de modo a selecionar os 'k' valores de *RSSI* com maior semelhança aos valores de *RSSI* fornecidos. Utiliza todas as informações apresentadas nas funções anteriores para limitar a janela de resultados. Dos 'k' valores obtidos, apenas aquele que se repete mais vezes é que é retornado (repetição por divisão).

Parâmetros de Entrada

nAccessPoints: Número inteiro que indica o número de APS do sistema
idUser: Identificador do utilizador que se pretende localizar
infoReceived: Objeto do tipo "List<List<object>>" contendo os valores de *RSSI* associados a cada *AP* do sistema. Exemplo: { {idAccessPoint1, RSSI1} , {idAccesPoint2, RSSI2} }.
idMapOrientation: Identificador que representa a orientação pretendida durante o algoritmo de pesquisa
idHousePointLastPosition: O identificador do ponto do mapa de calibração referente à localização do utilizador no instante 't-1' (entidade "*House Point*")
proximityRange: Distância máxima percorrida por um utilizador entre dois pontos (metros)

Resposta

Um objeto do tipo "List<object>" composto por: idHouseDivision, xAbs, yAbs, idSignalStrengthPoint1, selectionFactor1, idSignalStrengthPoint2, selectionFactor2, ... , idSignalStrengthPointN, selectionFactorN.

4.1.3.3 Modelo de Dados – Visão Física

A implementação do modelo de dados definido para o ISM requer a escolha dos tipos de dados que melhor se adequam aos atributos que caracterizam as várias entidades. Deste modo, para todos os dados que implicam a utilização de caracteres alfanuméricos para descrever uma característica ou condição de um atributo recorre-se ao tipo "VARCHAR". Exemplos deste tipo de dados são elementos como nomes, moradas, números de telefone, correio eletrónico, etc.. O tipo "INTEGER" é utilizado para representar atributos como números inteiros. Entre estes encontram-se as chaves de todas as entidades, configurações de algoritmos e tempos de amostragem. Caso não seja possível representar números somente através da sua parte inteira, utiliza-se o tipo "FLOAT". A maioria dos atributos com esta característica estão relacionados com valores de erro, dimensões da habitação ou fatores de escala.

A caracterização de elementos capazes de assumir somente dois estados é realizada através do tipo "BOOL". O estado de um serviço (acessível ou não acessível), de um utilizador (ativo ou não ativo) ou a veracidade de um atributo ("*scaleDefinedOnX*", entidade "*House WiLOS*") são alguns exemplos da utilização deste tipo de dados. Toda a informação relacionada com datas é definida através do tipo "DATETIME". Este permite armazenar dias, mês, ano, horas, minutos e segundos num formato facilmente manipulável quer pela BD, quer pelo mWiLOS.

Para atributos com características mais incomuns, devido à natureza dos seus dados, é necessária a utilização de tipos mais abstratos. O primeiro tipo é o "LONGBLOB", capaz de conter até 4 GB de informação. Este é utilizado pela BD do mWiLOS para armazenar uma sequência de bits que representa uma imagem de um utilizador (entidade "*User*", atributo "*avatarImage*") ou da tipologia da habitação (entidade "*House*", atributos "*houseImage*" e "*houseTopology*").

O segundo tipo é o "LONGTEXT", utilizado para armazenar os objetos que representam graficamente uma divisão (atributo "*object*", entidade "*House Division*") ou uma porta de uma divisão (atributo "*object*", entidade "*House Door*"). Antes da inserção do objeto gráfico na BD é realizada a sua conversão para um objeto XAML, um formato que pode ser arquivado sob a forma de caracteres alfanuméricos. Optou-se pela utilização de "LONGTEXT" em vez da utilização de "VARCHAR" por motivos de segurança. Na fase inicial do projeto ainda não estava definido qual seria o objeto gráfico a utilizar para representar uma divisão ou uma porta. Também não havia noção do tamanho que este objeto poderia ocupar na BD. Assim recorreu-se a um tipo de dados que fosse capaz de garantir uma maior margem de manobra. Na figura 4.23 apresenta-se o DER descrito no subcapítulo 4.1.1.3, bem como os tipos de dados utilizados para definir cada um dos seus atributos.

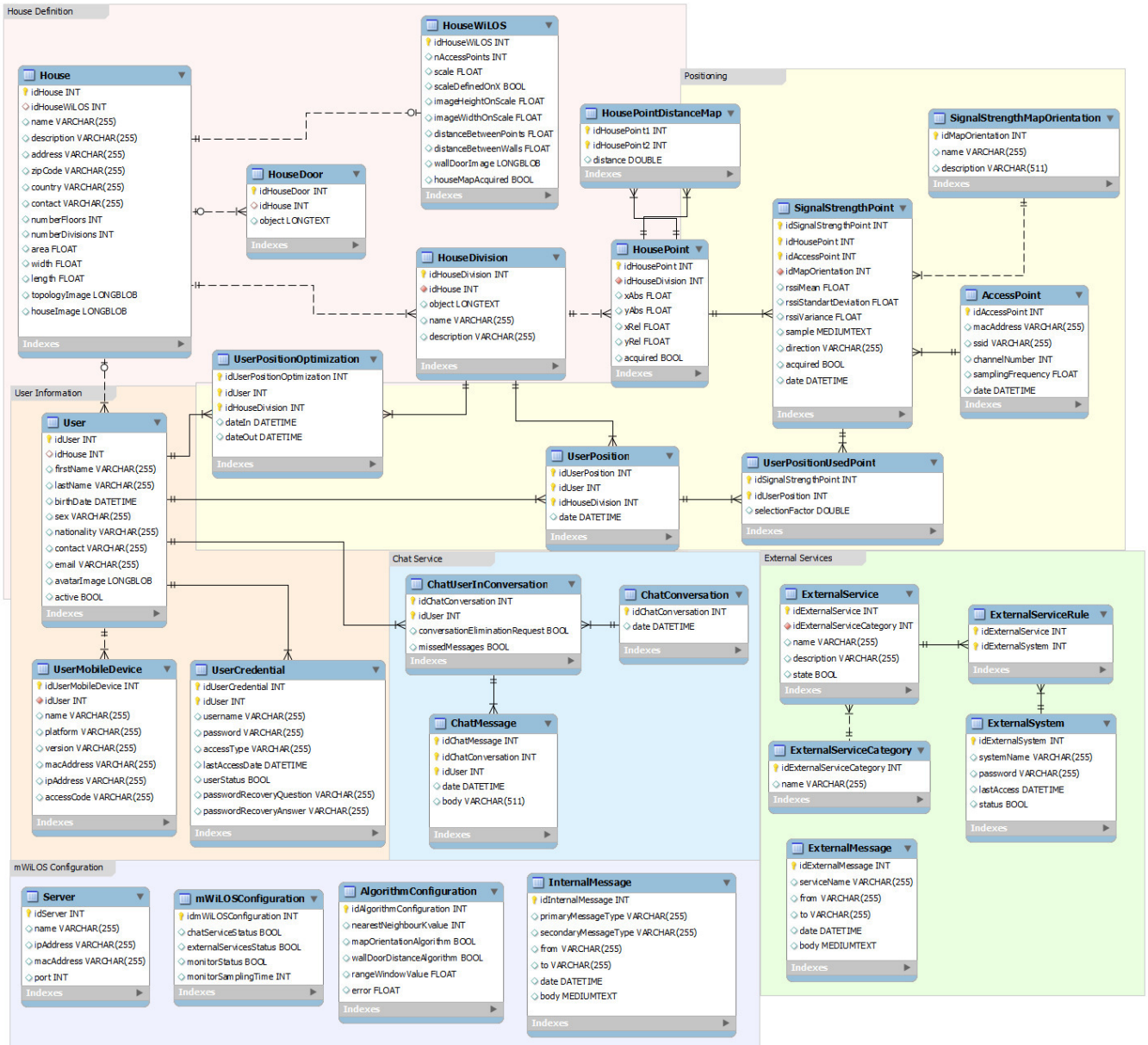


Figura 4.23 - Diagrama de entidades e relacionamentos do mWiLOS – Visão Física.

Existe também uma peculiaridade nos atributos utilizados para armazenar as palavras-passe dos utilizadores ou sistemas. De forma a garantir segurança da informação do mWiLOS, as palavras-passe não são simplesmente armazenadas na BD. Se tal fosse verdade, uma fuga de informação resultaria na descoberta da senha utilizada por um utilizador durante o processo de autenticação. Desta forma, antes da inserção destes elementos na BD, recorre-se ao algoritmo de encriptação MD5 para produzir um código ilegível e dificilmente recuperável. Assim, mesmo que ocorra fuga de informação, a palavra-passe não fica comprometida. Esta funcionalidade é garantida pelo ISM, na camada Entidade.

Uma vez que o mWiLOS possui funcionalidades de cálculo e determinação de distâncias e dimensões da habitação, é de se esperar que alguns atributos da BD obedeçam a um formato de unidades pré-estabelecido. A tabela 4.3 enumera estes atributos, bem como as respetivas unidades ou valores esperados para garantir o bom funcionamento do mWiLOS.

Atributo	Unidade
area, width, length	m^2
distanceBetweenPoints *distanceBetweenWalls* *distance* *rangeWindowValue*	m
scale	pixel/m

Atributo	Unidades
imageHeightOnScale *imageWidthOnScale* *xAbs, yAbs, xRel, yRel*	pixéis
monitorSamplingTime	ms
rssiMean	dBm

Tabela 4.1 - Listagem de atributos que necessitam informação num formato específico ou respeitando certas unidades.

O bom funcionamento do mWiLOS também é assegurado pelas funcionalidades automáticas implementadas na BD. Para tal, desenvolveu-se o *Event* "Verificar Utilizador Eliminado" que é evocado com uma periodicidade diária. A sua função é verificar se as conversações de um utilizador eliminado já foram removidas, de modo a concluir o processo de eliminação.

O processo de eliminação de um utilizador consiste na eliminação de todos os dados a ele associados do ISM, incluindo o seu histórico de localização. No entanto, as conversações não dependem somente deste utilizador, mas também de todos os utilizadores que nelas participam. Para garantir a integridade da informação disponibilizada pelo CSM, as entidades "*User*" e "*User Credentials*" continuam a conter a informação essencial do utilizador, caso existam conversações a ele associadas. Só após a remoção da última conversação pertencente ao utilizador, é que o *Event* elimina os dados existentes nestas duas entidades.

4.1.3.4 Modelo Comportamental

O funcionamento do mWiLOS só é garantido através da cooperação entre os diversos módulos que definem a sua arquitetura. A interação entre mWiLOS, sWiLOS e utilizador também é determinante para que o WiLOS efetue o seu propósito: a localização dos utilizadores dentro da habitação e a disponibilização dos seus serviços. Deste modo, o utilizador ao interagir com a UI desencadeia eventos que evocam as funcionalidades implementadas nos módulos da camada Controlo. Por sua vez, estes interagem entre si, com o ISM (camada Entidade) e com as MSI e ESI, para satisfazerem os pedidos do utilizador, quer através da UI ou do sWiLOS, e dos sistemas externos.

Em termos de modelo comportamental, o SSMM e o CSM possuem um nível de interação com o sWiLOS muito mais delicado do que os restantes módulos, devido à necessidade de se respeitar o protocolo de comunicação da MSI. O SSMM é responsável por iniciar o processo de calibração do mapa de potência do sinal mas, a partir desse ponto, passa a comportar-se como um elemento passivo, onde o sWiLOS e o utilizador se encarregam de obter os pontos de calibração. O CSM efetua o papel de supervisor do serviço de mensagens de modo a criar conversações, armazenar e reencaminhar mensagens trocadas e apresentar históricos de conversação ao utilizador, quer através do mWiLOS, quer a partir do sWiLOS. A sua existência depende da interação entre utilizadores através do sWiLOS. Desta forma, os mecanismos comportamentais associados a estes módulos são descritos no subcapítulo 4.1.4.4, referente ao modelo comportamental do sWiLOS.

Relativamente ao UM, o único mecanismo que desperta interesse comportamental é o processo de emparelhamento de um dispositivo móvel com o mWiLOS. Mais uma vez, a cumplicidade existente entre mWiLOS e o sWiLOS necessita da introdução do sWiLOS antes de se abordar este processo, à semelhança dos módulos supracitados.

O SM recorre às funcionalidades do ISM para construir os elementos estatísticos a apresentar ao utilizador. Os dados armazenados na BD são filtrados ou agrupados de acordo com a informação que se pretende obter através de *Queries MySQL*. Deste modo, parte do processamento da informação passa para o SGBD, reduzindo os recursos computacionais utilizados pelo mWiLOS. A informação resultante das *Queries* é tratada pelo SM e apresentada ao utilizador na interface gráfica disponibilizada pela UI. Em termos comportamentais, o SM apenas acede ao ISM e apresenta dados na UI, não introduzindo nenhum comportamento complexo no sistema.

Seguidamente detalham-se os modelos comportamentais dos módulos mais pertinentes do mWiLOS: a MSI, responsável pela comunicação sWiLOS-mWiLOS; o HTM, utilizado para a obtenção dos modelos comportamentais; o MM, que determina a localização dos utilizadores, e o ESM e a ESI, utilizados para gerir e disponibilizar os serviços de localização para sistemas externos ao WiLOS.

Interface Mestre-Escravo

A MSI do mWiLOS implementa as funcionalidades de um servidor TPC/IP, de modo a garantir a comunicação com todos os sWiLOS registados na plataforma WiLOS. Como detalhado no subcapítulo 4.1.3.2, a classe "_Server" possui 3 procedimentos cruciais para a gestão da comunicação, o "TCP Listenner Thread", o "Handle Communication Thread" e o "Check Client Connection". O primeiro aguarda a chegada de novos clientes à rede e efetua o estabelecimento da ligação. O segundo fica responsável pela ligação, de formar a analisar e tratar todos os pedidos e respostas vindos do sWiLOS. O terceiro verifica periodicamente se a ligação existente entre sWiLOS e mWiLOS ainda é válida. Os diagramas de atividade da figura 4.24 permitem analisar com maior detalhe o encadeamento de ambos os procedimentos.

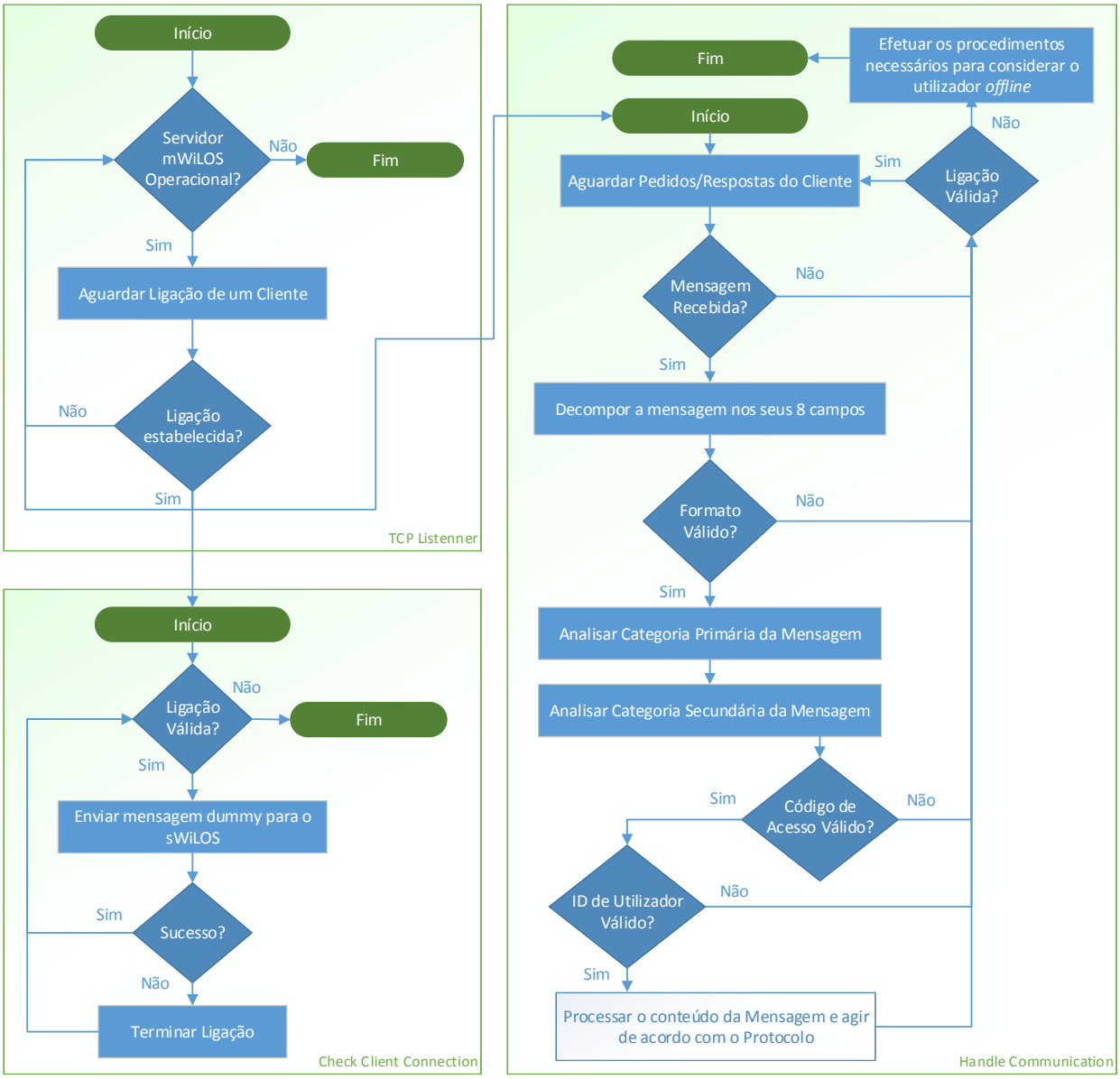


Figura 4.24 - Diagramas de atividade referentes aos procedimentos "TCP Listenner", "Handle Communication" e "Check Client Connection".

O procedimento "TCP Listenner" verifica constantemente se a MSI está operacional, de modo a aguardar novas ligações. Caso se verifique a ocorrência de comunicação através da porta definida para o mWiLOS, este lança uma instância do procedimento “Handle Communication“, para gerir a comunicação com o possível cliente, e uma instância do procedimento “Check Client Connection”. De seguida, o “TCP Listenner” retorna ao estado de espera, de forma a receber eventuais dispositivos sWiLOS.

A fase de espera de pedidos ou respostas vindas do sWiLOS, implementado no “Handle Communication”, é um mecanismo reativo que verifica a receção de dados através do *socket* TCP/IP atribuído ao sWiLOS. A receção de uma mensagem é efetuada em duas partes. A primeira parte da mensagem contém a informação referente ao tamanho da mensagem eminente. A segunda é a mensagem propriamente dita, que respeita o tamanho indicado na primeira parte.

De seguida, efetua-se a decomposição da mensagem nos seus campos, de acordo com o protocolo de comunicação estabelecido para o sWiLOS e o mWiLOS (subcapítulo 4.1.4.2). Esta permite validar o formato da mensagem enviado e facilita o processo de análise da informação recebida. Desta forma, as ações “Analisar Categoria Primária da Mensagem” e “Analisar Categoria Secundária da Mensagem” reencaminham o mecanismo de processamento da mensagem para a área onde se encontram implementadas as operações a realizar para cada tipo de mensagem recebida. Por norma, cada uma das áreas valida o sWiLOS através da verificação do seu identificador e do código de acesso utilizado durante a comunicação. O processamento do conteúdo da mensagem vai depender da natureza da mesma e do que foi definido pelo protocolo de comunicação.

O “Check Client Connection” é um procedimento simples que tenta enviar periodicamente uma mensagem “vazia” através da ligação estabelecida. Isto deve-se ao facto dos mecanismos reativos existentes nos *sockets* não possuírem a capacidade para detetar a queda de uma ligação caso se encontrem em modo de espera (procedimento “Handle Communication”). Porém, caso se tente utilizar o *socket* de forma ativa, isto é, através da solicitação do envio de dados, o *socket* reage imediatamente na presença de uma ligação inválida. Uma vez que o *socket* está a ser gerido por ambos os procedimentos, este “pequeno truque” permite que o “Handle Communication” termine a ligação inválida com o respetivo sWiLOS, em vez de aguardar infinitamente a receção de informação. Como no momento de atribuição do *socket* a um cliente também se atribuí um semáforo, garante-se que não ocorrem problemas de acesso simultâneo durante todas as operações de escrita e leitura do respetivo *socket*.

Note-se que o “TCP Listenner” verifica constantemente se a MSI está operacional. No entanto, apesar de esta condição estar somente destacada neste procedimento, também condiciona o

funcionamento de “Handle Communication” e “Check Client Connection”. O facto de a MSI deixar de estar operacional obriga ao encerramento de todas as ligações em execução. Se tal fenómeno ocorrer, o sWiLOS, a partir de “Handle Communication”, procede à desativação do utilizador, libertando todos os recursos por ele utilizados e atualizando o seu estado no ISM. Por outro lado, se o utilizador saiu da rede de forma tradicional, isto é, se desligou o dispositivo ou saiu do alcance do servidor mWiLOS, o “Handle Communication”, para além de libertar recursos, informa os outros utilizadores que alguém saiu da plataforma WiLOS.

O protocolo de comunicação da MSI, utilizado para efetuar a comunicação entre o sWiLOS e o mWiLOS, é descrito detalhadamente no subcapítulo 4.1.4.2. No entanto, a figura 4.25 ilustra um exemplo de comunicação realizado entre estes 2 subsistemas, de modo a oferecer ao leitor uma melhor noção de como se efetua a troca de mensagens através desta interface.

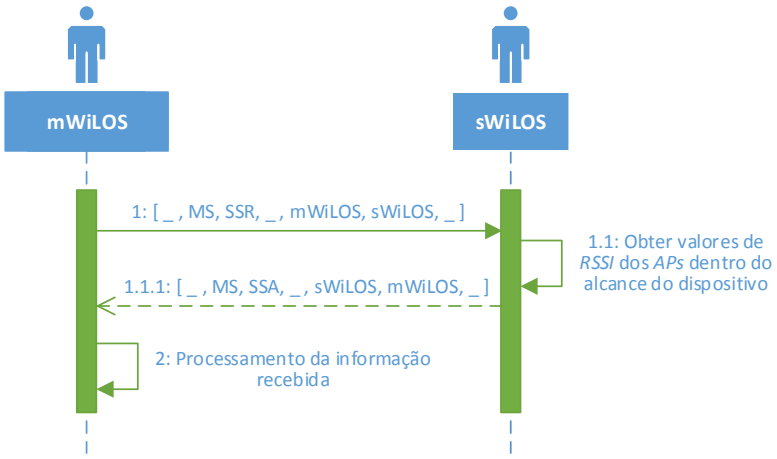


Figura 4.25 - Diagrama de sequência referente a um pedido de localização (mWiLOS) e respetiva resposta (sWiLOS).

O diagrama de sequência apresentado detalha as ações realizadas pelos mWiLOS e pelo sWiLOS durante o processo de monitoração de um dispositivo móvel. Deste modo, admitindo que a ligação entre sWiLOS e mWiLOS foi estabelecida com sucesso, o mWiLOS questiona periodicamente o sWiLOS quanto à sua localização. Para tal, utiliza a mensagem “MS-SSR” do protocolo de comunicação definido. O sWiLOS, para responder ao pedido recebido, recolhe os valores de *RSSI* verificados pelo dispositivo naquele instante e responde ao mWiLOS com uma mensagem “MS-SSA”. O mWiLOS analisa a mensagem recebida através do procedimento “Handle Communication” e reencaminha a informação para o módulo Monitoração, que determina a localização do utilizador.

Módulo "Tipologia da Habitação"

O processo de definição da tipologia da habitação é um elemento essencial para se obterem os diversos modelos que definem a habitação. Como descrito em 4.1.3.2, o HTM permite através da UI o desenho das divisões e das portas que constituem a tipologia da habitação. Cada divisão é representada por um elemento do tipo "Polygon", um desenho geométrico composto por vários pontos, de modo a fornecer uma vasta gama de configurações possíveis para a divisão. Uma porta é modelada pelo tipo "Rectangle", podendo ser movida ou redimensionada consoante a necessidade do utilizador. O desenho das divisões e das portas ocorrem sobre dois painéis de desenho disponibilizados pelo XAML/C#, os "Canvas". Desta forma, as divisões são definidas no "Canvas Divisions" e as portas no "Canvas Doors". É através da análise dos elementos que constituem cada uma das "Canvas" que se obtém os modelos a utilizar no processo de calibração e de monitoração.

A obtenção de cada um destes modelos ocorre por etapas. Primeiro obtém-se o modelo da tipologia da habitação, seguido do modelo discreto da área habitacional e, por fim, o modelo de distância da habitação. A figura 4.26 descreve o diagrama de atividade referente à definição do modelo da tipologia da habitação.

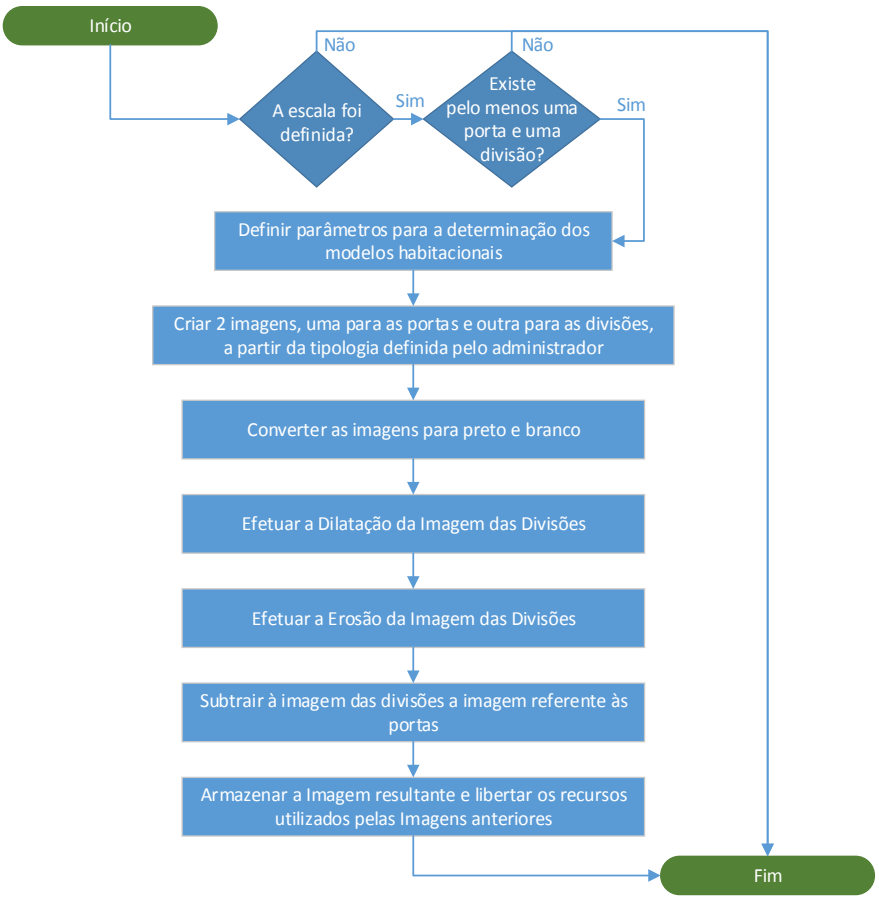


Figura 4.26 - Diagrama de atividade referente à obtenção do Modelo da Tipologia da Habitação.

Em primeiro lugar recorre-se às “Canvas” para se obter um modelo que represente fielmente a tipologia da habitação, isto é, algo que descreva onde começa e acaba cada parede de uma divisão, e qual a área por onde o utilizador se pode deslocar livremente. Apesar das “Canvas” possuírem esta informação, estas impõem um formato muito específico ao modelo. Para além disso, sempre que se deseje inferir mais alguma informação, é necessária uma análise “Polygon” a “Polygon”, e a todos os seus pontos, o que torna o processo exaustivo. Deste modo, optou-se por definir um outro modelo para a tipologia da habitação, mais simples e facilmente utilizável por outros sistemas. Este consiste numa imagem a preto e branco que representa todas as características do modelo supracitadas.

Esta imagem é obtida através de mecanismos de processamento de imagem, como se observa no diagrama da figura 4.26. Cada uma das “Canvas” pode ser convertida numa imagem colorida, contendo todos os elementos por ela definidos. Desta forma obtêm-se 2 imagens, uma imagem composta pelos contornos dos diversos “Polygons”, de modo a modelar as paredes das divisões, e uma imagem que representa os objetos do tipo “Rectangle”, definindo a localização e o tamanho das portas.

A conversão de ambas as imagens obtidas para preto e branco simplifica a sua análise e respetivo processamento. A primeira operação a realizar é a dilatação da imagem referente às divisões. Uma vez que é o utilizador a desenhar cada uma das divisões, podem ocorrer situações em que uma divisão não se encontre precisamente ao lado da sua adjacente. O processo de dilatação permite expandir os contornos das parede (pixéis pretos), de modo a tentar eliminar as ocorrências descritas. Para se recuperar a dimensão natural das paredes das divisões efetua-se o processo inverso, a erosão. A erosão não volta a separar as divisões porque apenas processa todos os pixéis brancos. Se os contornos entre divisões se juntaram, então esta operação não influência o resultado da dilatação.

Por fim, subtrai-se a imagem contendo as portas à imagem referente às divisões. Deste modo, todas as interseções ocorridas entre ambas as imagens são removidas (pixéis brancos), e a imagem resultante apenas contém a morfologia da habitação com a área transitável pelo utilizador. As verificações iniciais efetuadas pelo processo servem para garantir que as “Canvas” e a informação referente à escala da habitação, bem como todos os parâmetros necessários para a aquisição dos modelos da habitação, se encontram definidos. Só assim é que se garante a execução com segurança dos processos de obtenção dos modelos.

O próximo passo consiste na computação dos pontos do mapa de calibração, para se obter o modelo discreto da área habitacional. Mais uma vez recorre-se à “Canvas Divisions”, e às suas divisões, para se determinar qual a distribuição uniforme de pontos por divisão mais adequada. A determinação dos pontos de calibração necessita da definição de 3 valores: a escala da habitação; a distância entre pontos e a distância entre os pontos e as paredes da divisão. A escala fornece ao HTM a relação existente entre o número de pixéis e as dimensões reais da habitação (em metros). Os outros valores

são utilizados para se definir uma grelha de pontos de calibração uniforme para cada divisão, desde que se respeitem os valores introduzidos pelo utilizador e a tipologia da mesma.

Para tal, cada divisão existente em "Canvas Division" é decomposta nos pontos que definem o seu objeto "Polygon". Estes são analisados para se inferirem as dimensões da divisão. Através dos 3 valores supracitados e das dimensões obtidas, calcula-se o número de pontos possíveis para cada direção da divisão e a sua distribuição. Os pontos são posteriormente validados e armazenados no ISM (entidade "*House Point*"), de forma a compor o modelo discreto da habitação. Este processo encontra-se detalhado no diagrama de atividade da figura 4.27.

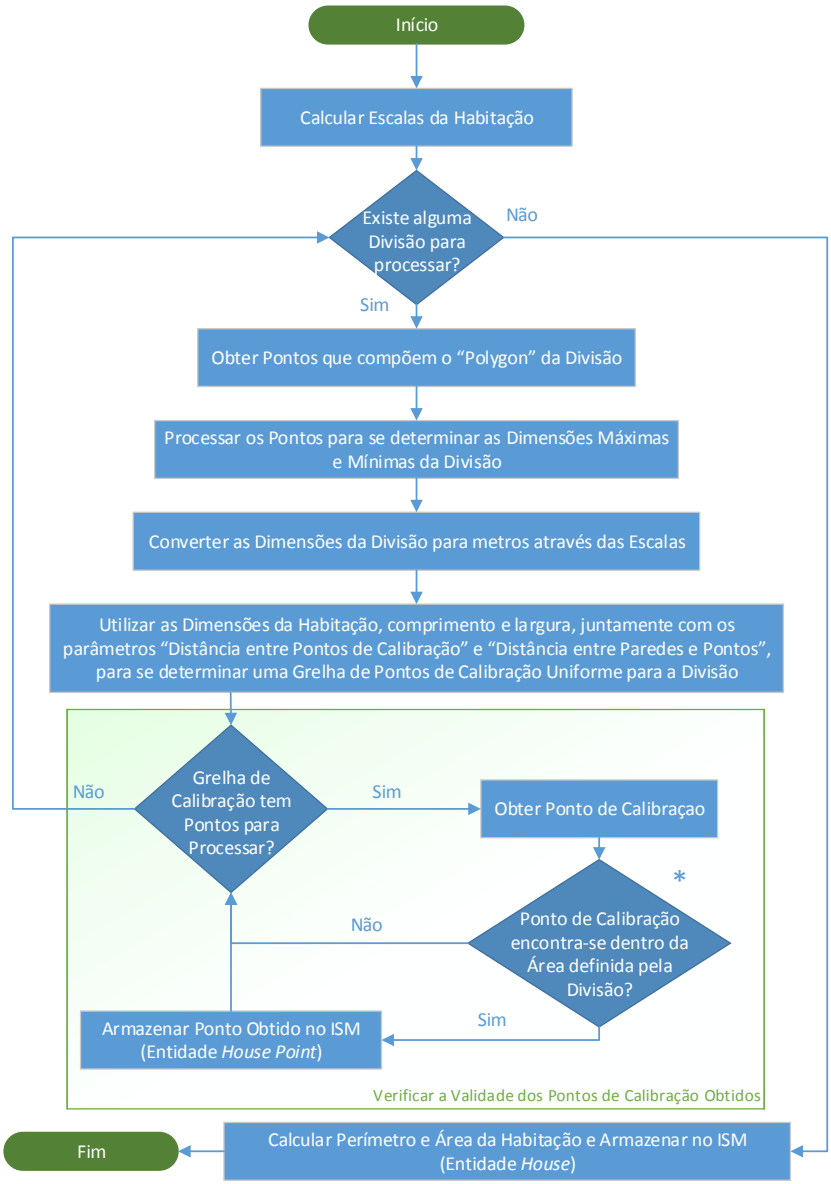


Figura 4.27 - Diagrama de atividade relativo à Computação dos Pontos de Calibração.

Neste diagrama, a única operação que pode suscitar maior curiosidade é a condição "Ponto de Calibração encontra-se dentro da Área definida pela Divisão?". Durante o cálculo das dimensões de uma divisão, apenas se extraem a largura e o comprimento máximo por ela definidos. Estes valores são então conjugados com as escalas da habitação, a distância entre pontos e a distância entre pontos e paredes, para se determinar uma grelha de pontos com distribuição uniforme ao longo da largura e do comprimento da divisão. Como se pode concluir, esta grelha possui uma morfologia retangular, que pode não coincidir com a morfologia da divisão, como demostrado na figura 4.28.

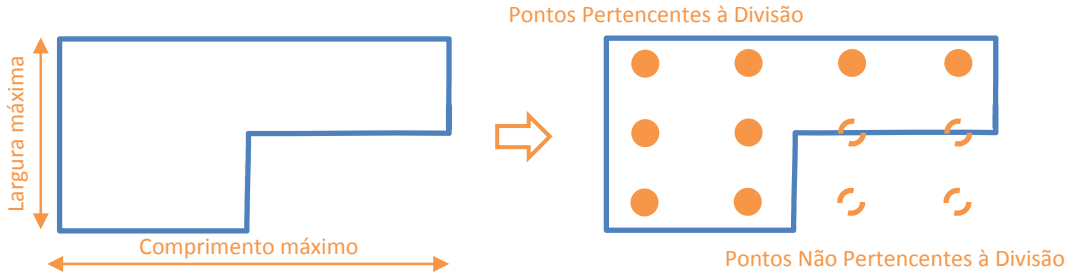


Figura 4.28 - Problemática da determinação de uma grelha uniforme de pontos para uma divisão "não retangular".

A condição definida permite verificar se, de facto, todos os pontos obtidos pertencem à área definida pela divisão, de forma a respeitar a sua morfologia e garantir que todos os pontos gerados são válidos. A função "isInPolygon" é responsável por determinar se o ponto de calibração obtido encontra-se dentro do objeto "Polygon" que define a divisão. Esta função foi disponibilizada pelo autor Daniel Kuppitz em Microsoft Developer Network, através do tópico "*Determine if the point is in the polygon, C#*".

O último modelo que falta determinar é o modelo de distância da habitação. Como referido no subcapítulo 4.1.1.3, este modelo é representado por um grafo cujos nós são dados pelos pontos de calibração. No entanto, antes de se determinar a distância mínima existente entre os vários nós, de modo a se obter o modelo pretendido, é necessário definir as ligações, e respetivas distâncias, que unem os vários pontos do mapa de calibração. Para tal recorre-se ao modelo da tipologia da habitação (imagem a preto e branco) e ao modelo discreto da habitação.

Para cada ponto de calibração verifica-se quais são os pontos que podem ser considerados vizinhos, isto é, que se encontram a uma distância máxima pré-definida. Neste caso, considerou-se que um ponto vizinho situa-se a uma distância máxima dada pelo dobro da distância entre pontos (variável obtida durante a aquisição do modelo discreto da habitação). No entanto, não basta a proximidade para se definir uma ligação entre nós, a alcançabilidade é outro requisito fundamental. Por outras palavras, uma ligação entre nós só é válida se não se verificar a existência de uma parede da habitação a separá-los. Afinal, um utilizador não é capaz de atravessar as paredes, mas sim

contorná-las. Daí a necessidade da utilização do modelo da tipologia da habitação para determinar um mapa de distância que limite a área de movimentação de um utilizador.

A determinação da existência de uma parede entre pontos efetua-se a partir de processamento de imagem. Uma vez que a localização dos pontos em análise é conhecida, basta percorrer a distância que os separa em pixéis, verificando a existência de alguma parede (pixéis pretos). O sentido e direção do varrimento é determinado pela própria localização dos pontos. Este processo é efetuado para todos os pontos de calibração e os seus possíveis vizinhos. Portanto, a ocorrência de muitos pontos no mapa de calibração, obriga a um elevado processamento computacional para a criação do grafo necessário para se inferir o modelo de distância. Todos os passos necessários para a determinação deste modelo são apresentados no diagrama da figura 4.29.

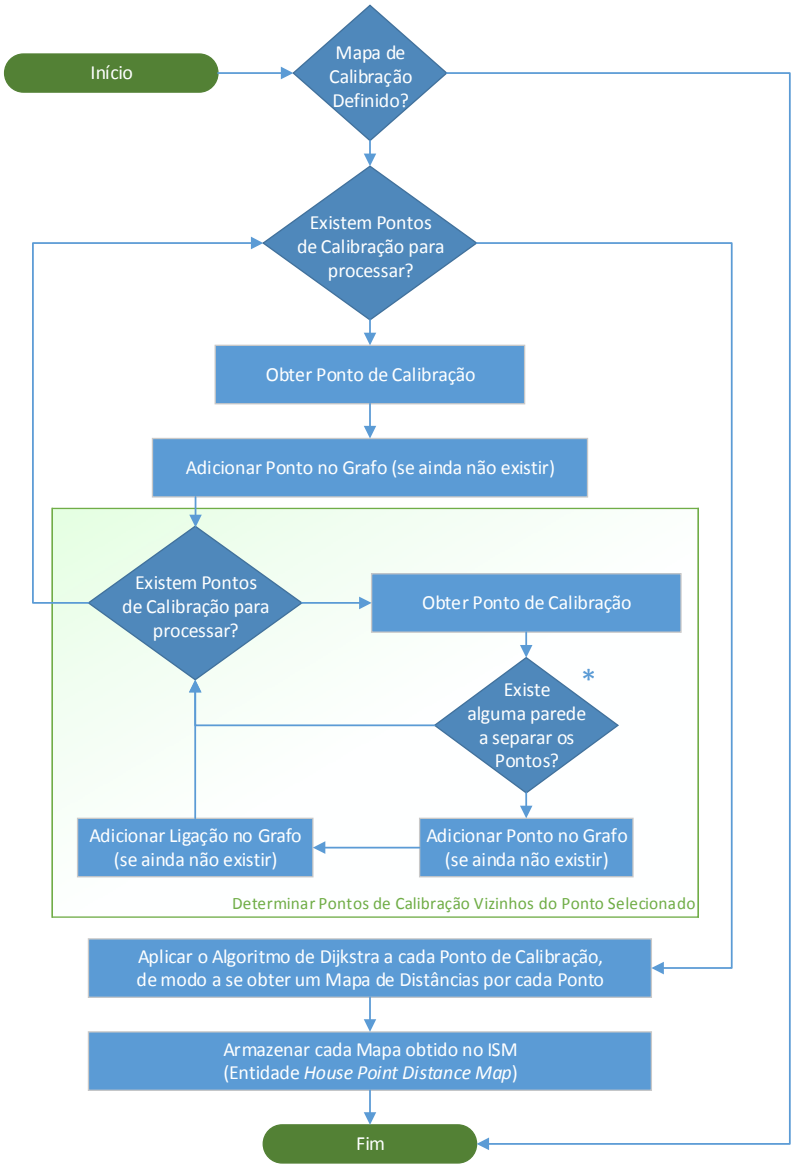


Figura 4.29 - Diagrama de atividade referente à determinação do Modelo de Distância da Habitação.

Uma vez obtido o grafo da habitação, determina-se a menor distância existente entre cada nó do grafo e todos os outros nós que o constituem. Cada um destes mapas de distância é depois utilizado durante o processo de monitoração, para se determinar a localização dos utilizadores através do algoritmo k-*NNS* com filtro de distância aplicado. A determinação da distância mínima entre nós do grafo é realizada a partir do algoritmo de Dijkstra.

O algoritmo de Dijkstra é um método que resolve o problema da procura do caminho mais curto em grafos com distâncias não negativas. O seu objetivo é analisar um grafo a partir de um nó inicial e determinar a menor distância necessária para se atingirem os restantes nós do grafo. O algoritmo original, desenvolvido pelo Holandês Edsger Dijkstra, apenas se preocupa com o conceito de distância, e não propriamente com o caminho efetuado para lá chegar. Este método é conhecido por ser um dos mais eficientes para se determinar a solução para este tipo de problemas.

Para se poder aplicar este algoritmo são necessárias 3 condições: um grafo composto por nós e ligações direcionais ou não direcionais; todas as ligações do grafo têm de possuir distâncias, ou pesos, não negativos e o grafo deve ser ligado, isto é, tem de existir pelo menos uma ligação válida entre os nós do grafo. O algoritmo pode ser analisado como um mecanismo de propagação, que parte do nó inicial e analisa sequencialmente todos os seus vizinhos, de modo a inferir o caminho mais curto até cada nó do grafo. A sua execução é descrita pelos seguintes passos:

1. Definem-se 2 conjuntos, um de nós visitados (V) e um de nós por visitar (NV);
2. Inicialmente, todos os nós encontram-se no conjunto NV, e atribui-se um valor de distância 'infinito' a cada um deles. Este valor indica que ainda não se sabe a distância deste nó ao nó inicial. O nó inicial é a única exceção, sendo-lhe atribuído o valor de distância '0'.
3. Procura-se o nó do conjunto NV com menor valor de distância, de modo a torná-lo no nó a analisar. Na primeira iteração, o nó escolhido é o nó inicial.
4. Verifica-se a distância existente entre o nó escolhido e os seus vizinhos, através das ligações existentes entre eles.
5. Soma-se a distância atribuída ao nó com a distância da ligação existente entre os nós vizinhos. Se o resultado for inferior ao valor de distância previamente atribuído ao nó, efetuar a sua atualização. Caso contrário, não realizar nenhuma alteração.
6. Após o processamento de todos os nós vizinhos, considerar o nó em análise visitado, movendo-o do conjunto NV para o conjunto V.
7. Repetir o passo 3 até que o conjunto NV não possua nenhum nó ou não existam ligações que permitam atingir os nós em falta.

A figura 4.30 mostra um pequeno exemplo da aplicação do algoritmo de Dijkstra a um grafo não negativo, ligado e composto por 4 nós e 5 ligações.

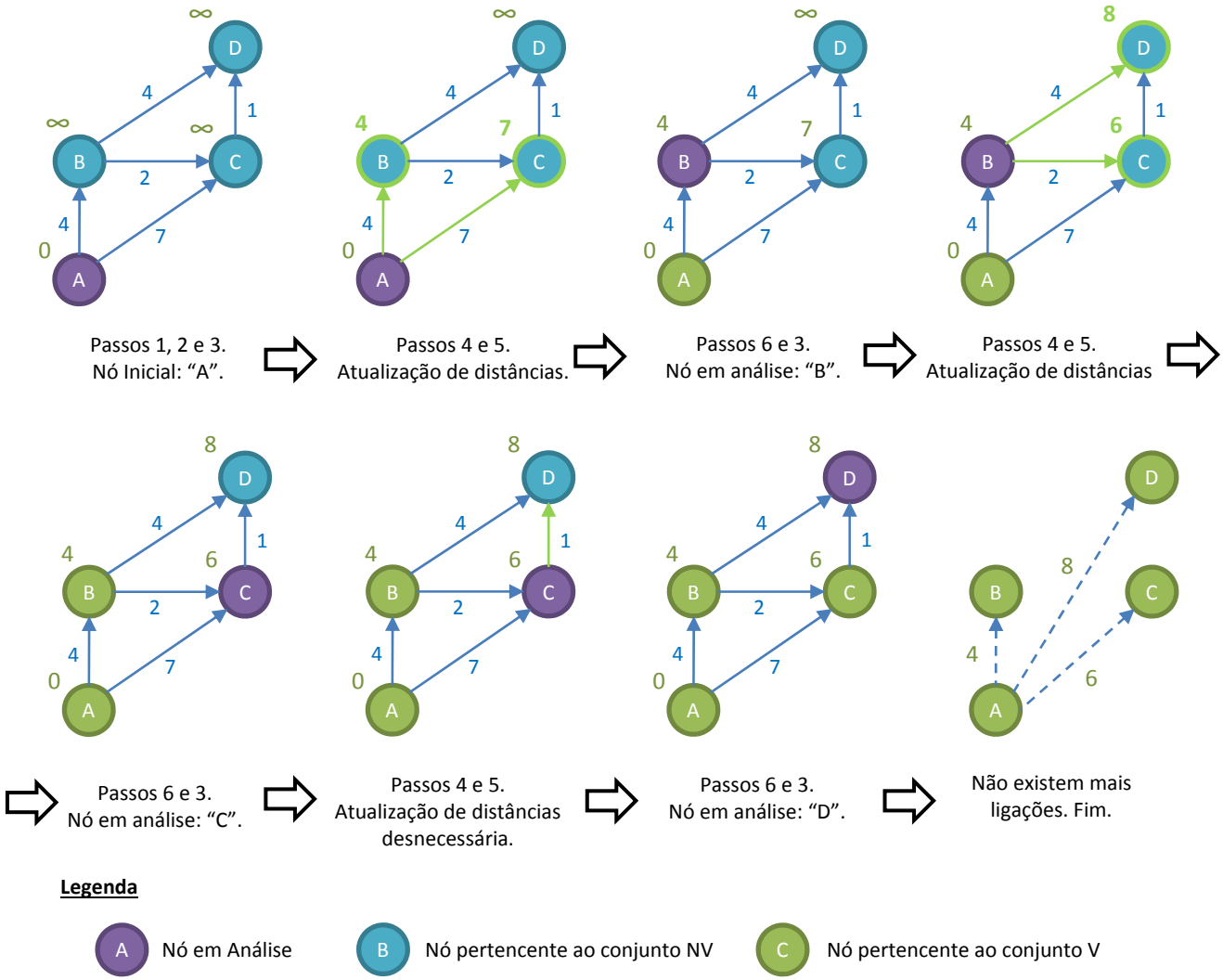


Figura 4.30 - Aplicação do Algoritmo de Dijkstra a um grafo composto por 4 nós e 5 ligações.

Este algoritmo é bastante utilizado nas áreas de robótica e planeamento de trajetos, onde o caminho mais curto é normalmente utilizado para poupar recursos, como a bateria ou o combustível, ou para se atingir o destino mais rapidamente. A sua popularidade origina diferentes variações da sua implementação, e em diversas linguagens de programação. Assim, sua eficiência, popularidade e simplicidade, são apenas algumas das vantagens que levaram à adoção deste algoritmo. A utilização deste algoritmo no WiLOS recorre à classe "Dijkstra_Algorithm", disponibilizada pelo utilizador James Jiao em sourceforge.net, através do tópico "*Dijkstra's Algorithm C# Implementation*".

Para além do algoritmo de Dijkstra, esta também implementa mecanismos para a construção de grafos, sendo portanto utilizada durante a definição do grafo de distâncias da habitação. A classe teve de ser adaptada de modo a satisfazer as necessidades do HTM e dos tipos de dados por ele geridos. Algumas dessas adaptações consistiram em tornar a classe compatível com distâncias decimais e

tornar os mecanismos de construção de grafos mais eficientes, através da verificação de ligações entre nós repetidas.

A aplicação das funcionalidades desta classe a cada nó do grafo permite obter um grafo de distâncias mínimas para cada nó. Deste modo, a extração de todas as ligações de cada grafo, e respetivo armazenamento na entidade "*House Points Distance Map*", permite definir o modelo de distância da habitação. Este pode posteriormente ser consultado através do MM, uma vez que é utilizado por um dos algoritmos de localização durante a determinação da posição do utilizador.

A figura 4.31 ilustra os 4 processos supracitados, que possibilitam a aquisição dos modelos da habitação.

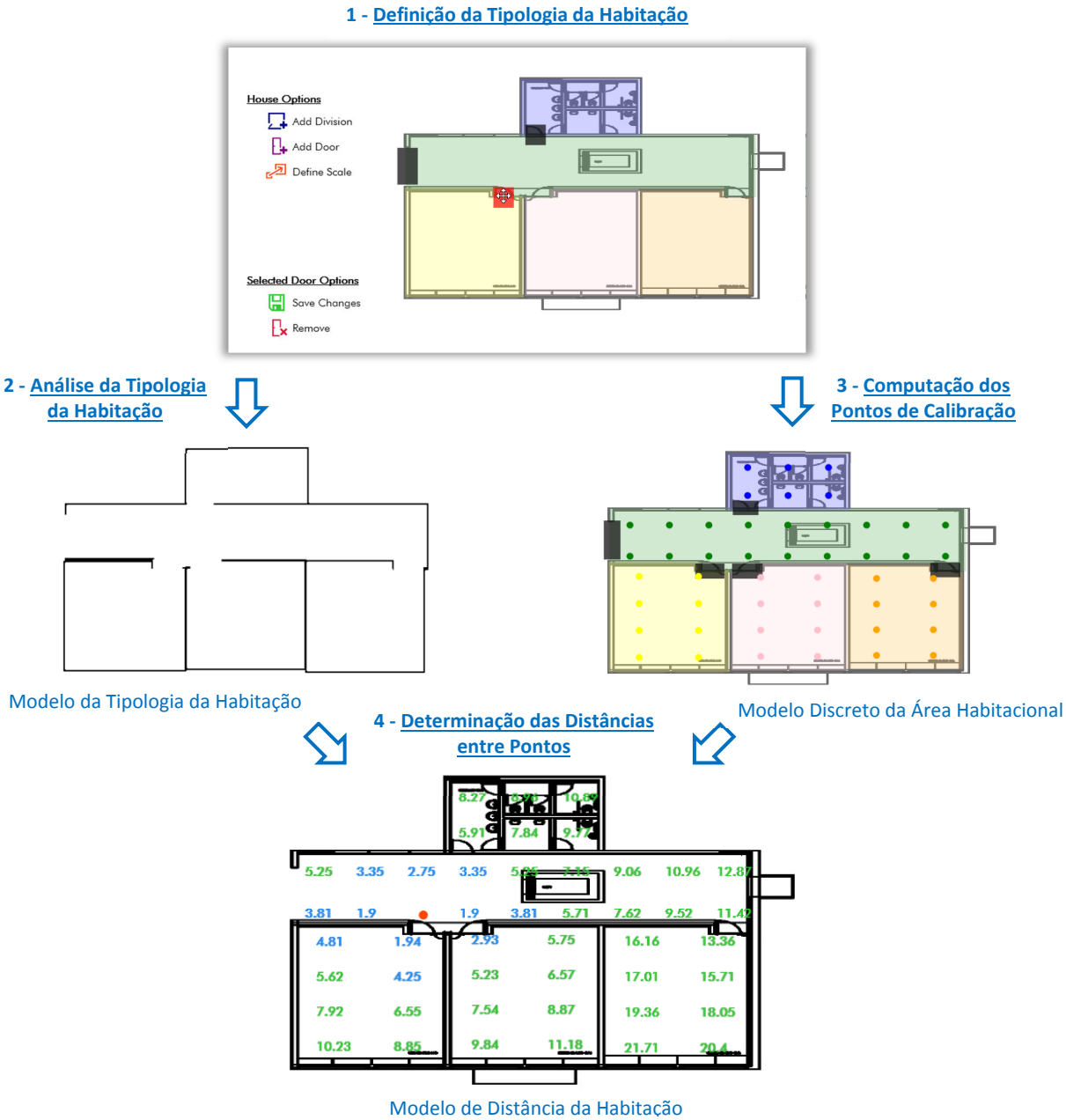


Figura 4.31 - Visualização global da definição dos modelos utilizados para descrever a habitação no sistema WiLOS.

Módulo "Monitoração", MSI e ISM

O processo de monitoração implica uma relação muito forte entre o MM, a MSI e o ISM. Todo o encadeamento de ações que torna a monitoração possível, desde a requisição de informação dos dispositivos móveis até à determinação da sua localização, encontra-se detalhado no diagrama de sequência da figura 4.32.

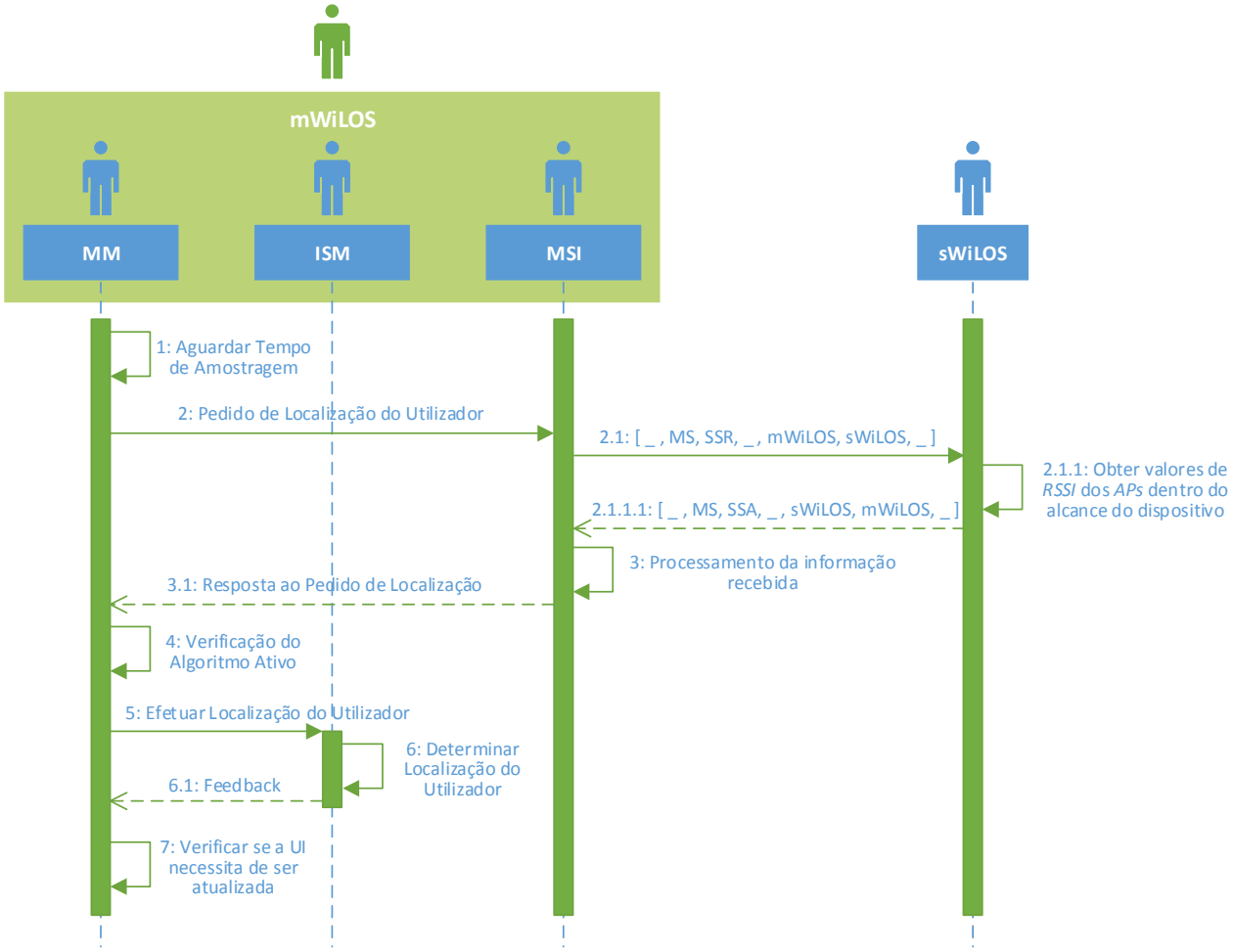


Figura 4.32 - Diagrama de sequência referente ao processo de monitoração de um dispositivo sWiLOS.

O MM, através da classe "_locateUsers", disponibiliza várias funcionalidades necessárias para a execução das duas fases de monitoração. A requisição de informação dos dispositivos móveis é realizada através do procedimento "Send Request For Location". Este interage com a MSI de modo a questionar periodicamente todos os utilizadores ativos acerca da sua localização, como verificado nas ações 1, 2 e 2.1 do diagrama de sequência apresentado. O funcionamento interno deste procedimento obedece ao diagrama de atividade da figura 4.33.

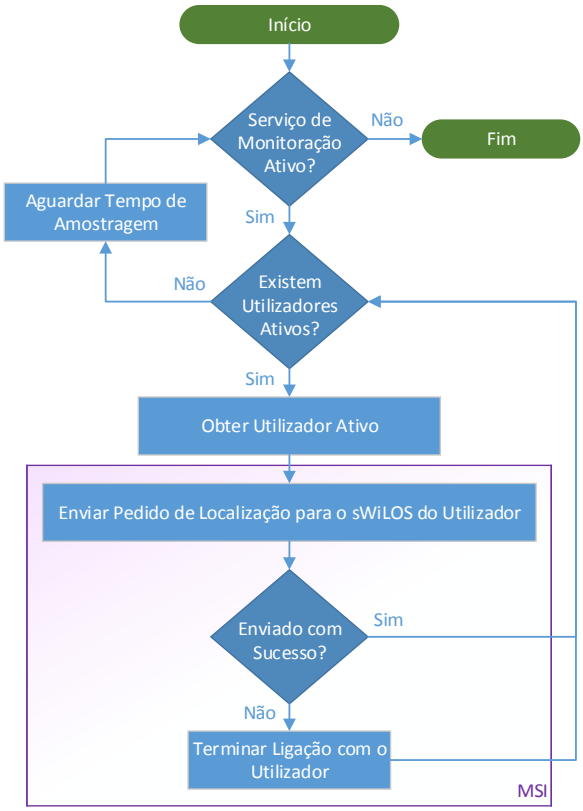


Figura 4.33 - Diagrama de atividade do procedimento "Send Request For Location".

Este procedimento verifica constantemente o estado do serviço de monitoração, uma vez que o utilizador pode desativá-lo a qualquer momento. De seguida obtém todos os utilizadores ativos e efetua um pedido de localização através da MSI. A ocorrência de um erro durante a comunicação significa que a ligação estabelecida pelo *socket* deixou de ser válida, e termina-se a ligação com o utilizador. A MSI fica responsável por libertar os recursos utilizados pelo dispositivo do utilizador.

A fase de estimação da localização dos utilizadores inicia-se quando o dispositivo móvel sWiLOS responde ao pedido de localização realizado pelo mWiLOS (ação 2.1.1.1 da figura 4.32). A informação recebida na MSI contém os dados relativos aos *APs* detetados e aos valores de *RSSI* a eles associados. No entanto, os mecanismos de localização implementados no MM necessitam de uma amostra que contenha todos os *APs* registados no WiLOS, detetados ou não. Daí a necessidade de processar a informação recebida através do sWiLOS, dada pela ação 3.

Uma vez que o mWiLOS dispõe de vários algoritmos de localização, é necessário verificar quais deles é que se encontram ativos, de modo a evocar o procedimento de localização correto. A tarefa de decisão é efetuada pela função "Location Thread" (ação 4), que recorre a um dos procedimentos implementados no ISM para determinar a localização atual do utilizador (ação 5). Se a UI do mWiLOS permitir, as novas localizações dos utilizadores são atualizadas na interface gráfica disponibilizada pelo MM (ação 7).

A subclasse "_LocationMechanisms", definida no ISM, disponibiliza as funcionalidades "_location_NN", "_location_NN_DeviceOrientation", "_location_NN_DistanceMap" e "_location_MM_DeviceOrientation_DistanceMap", como referido no subcapítulo 4.1.3.2. O diagrama de atividade representado na figura 4.34 demonstra os passos realizados por estes procedimentos para se determinar a localização de um utilizador (ação 6). O esclarecimento deste diagrama é efetuado através do procedimento "_location_NN" (anexo N), uma vez que se trata do algoritmo base do sistema. As restantes funcionalidades não passam de pequenas variações deste, incorporando na *Query MySQL* mais uma ou outra condições, de modo a variar o tipo de pesquisa efetuado e, consequentemente, os resultados obtidos.

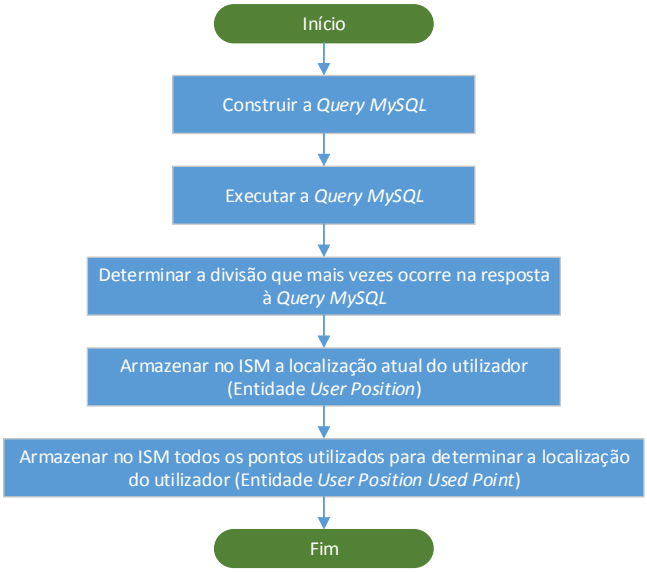


Figura 4.34 - Funcionamento interno dos procedimentos definidos na subclasse "_LocationMechanisms.

Para se explicar a construção da *Query MySQL* é necessário voltar a introduzir o conceito por detrás do algoritmo k-*NNS*, e a sua aplicação no WiLOS.

O algoritmo k-*NNS* procura verificar qual é o valor com maior número de ocorrências, dentro de um conjunto de 'k' valores, que mais se aproxima do valor pretendido. Desta forma, o primeiro passo a efetuar é a obtenção de um conjunto de 'k' valores que cumpram os requisitos. Para se inferir a proximidade existente entre valores, recorre-se ao método da distância Euclidiana a '*d*' dimensões (4.1).

$$J_e[k,l] = \sqrt{\sum_{i=1}^{d}(x_{ik} - x_{il})^2} \quad (4.1)$$

Aplicando estes conceitos ao caso em estudo, os valores que se pretendem comparar são os valores de *RSSI* verificados pelo dispositivo móvel e os valores de *RSSI* que definem os mapas de potência do sinal, adquiridos durante a calibração do sistema. Estes encontram-se armazenados na entidade "*Signal Strength Point*" e associados aos pontos de calibração da entidade "*House Points*", permitindo determinar as divisões dos valores de *RSSI* obtidos. Definiu-se também, na fase de modelação do WiLOS, que todos os pontos que constituem os mapas de potência, Norte, Sul, Este e Oeste, encontram-se elegíveis para a aplicação do algoritmo k-*NNS*. É através deste raciocínio que se definiu a *Query MySQL 1*, referente à determinação do valor de *RSSI* mais semelhante ao valor de *RSSI* verificado através de 1 *AP*.

Query MySQL 1 - Determinação do valor de *RSSI* mais semelhante ao valor de *RSSI* fornecido por 1 *AP*

```
SELECT ap1.idHousePoint, hp.idDivision, ap1.direction,
        ABS(ap1.rssiMean - RSSI_AP1)
FROM signalStrengthPoints AS ap1
JOIN housePoint AS hp
ON hp.idHousePoint = ap1.idHousePoint
WHERE (ap1.direction = "_north
    OR ap1.direction = "_south"
    OR ap1.direction = "_east"
    OR ap1.direction = "_west)
    AND ap1.idAccessPointMap = idAccessPointMap1
ORDER BY SQRT( POW( ABS(ap1.rssiMean - RSSI_AP1), 2) ) ASC
LIMIT 0, K;
```

Os valores que se encontram destacados na *Query* são elementos fornecidos ao procedimento "_location_NN" durante a execução do processo de monitoração. Desta forma, a aplicação da distância Euclidiana a todos os elementos de "*Signal Strength Point*", admitindo todas as orientações cardeais, permite organizar todos os valores de *RSSI* de acordo com a sua proximidade com o valor de *RSSI* fornecido. A disposição dos resultados sob a forma ascendente, isto é, dos valores mais próximos para os mais distantes, associada à limitação do número de resultados devolvidos pela *Query*, permite obter o conjunto 'k' pretendido.

No entanto, o objetivo principal dos mecanismos de localização do WiLOS é a escalabilidade da infraestrutura da rede Wi-Fi da habitação. Assim, o procedimento "_location_NN" deve ser capaz de determinar o posicionamento de um dispositivo móvel através de 1, 2, 3 ou *N APs* que constituem a rede WiLOS. Para tal, a formulação da *Query MySQL* tem de ser dinâmica, de modo a acompanhar o número de *APs*, e as coordenadas de localização que eles representam ($RSSI_{AP1,}$ $RSSI_{AP2}$, …, $RSSI_{APN}$). Desta forma, por cada *AP* registado no WiLOS, tem de se efetuar um *JOIN* e interligar os diversos mapas de potência de sinal definidos por cada *AP*, como demonstrado na *Query MySQL* 2.

Query MySQL 2 - Determinação dos valores de *RSSI* mais semelhantes aos valores de *RSSI* fornecidos por 2 *APs*

```
SELECT ap1.idHousePoint, hp.idDivision, ap1.direction,
        ABS(ap1.rssiMean - RSSI_AP1), ABS(ap2.rssiMean - RSSI_AP2)
FROM signalStrengthPoints AS ap1
JOIN signalStrengthPoints AS ap2
ON (ap1.idHousePoint = ap2.idHousePoint AND ap2.idAccessPointMap = idAccessPointMap2)
JOIN housePoint AS hp
ON hp.idHousePoint = ap1.idHousePoint
WHERE ((ap1.direction = "_north" AND ap2.direction = "_north")
    OR (ap1.direction = "_south" AND ap2.direction = "_south")
    OR (ap1.direction = "_east" AND ap2.direction = "_east")
    OR (ap1.direction = "_west" AND ap2.direction = "_west"))
    AND ap1.idAccessPointMap = idAccessPointMap1
ORDER BY SQRT( POW( ABS(ap1.rssiMean - RSSI_AP1), 2) + POW( ABS(ap2.rssiMean - RSSI_AP2), 2) ) ASC
LIMIT 0, K;
```

A existência de vários *APs* implica a procura dos valores de *RSSI* na mesma entidade mas com condições diferentes. Deste modo, a operação de *JOIN* para o "ap2" permite interligar todos os pontos que definem os mapas de potência do "ap1" com os mapas de potência do "ap2". Esta ligação realiza-se a partir de um elemento que ambos têm em comum, os pontos de calibração dados por "*House Point*". Porém, não basta interligar todos os mapas de potência de cada *AP*. O mecanismo de procura tem de ser coerente, isto é, não deve comparar a coordenada de valores de *RSSI* fornecida ($RSSI_{AP1}$, $RSSI_{AP2}$) com os valores de *RSSI* provenientes de mapas de potência com orientações diferentes (e.g., $RSSI_{AP1}$ Norte, $RSSI_{AP2}$ Sul). As condições definidas pelo *WHERE* garantem que os mapas de potência são organizados de acordo com as suas orientações cardeais.

Assim, a construção dinâmica da *Query MySQL*, aplicada pelo procedimento "_location_NN", consiste em acrescentar à *Query* todos os *APs* registados no sistema, através da adição de valores de RSSI para comparar, elementos *JOIN* e outras condições, de modo a cumprir os requisitos expostos.

A execução da *Query* resulta na obtenção de um conjunto de 'k' valores que mais se adequam aos valores de *RSSI* recebidos do sWiLOS. A aplicação do algoritmo k-*NNS* consiste em verificar qual é a divisão que mais vezes ocorre nesse conjunto. Em caso de empate, considera-se a divisão que mais vezes se repete e que possua um valor de erro menor. Se mesmo assim se verificar uma colisão, a primeira divisão encontrada é aquela que se atribui ao utilizador.

O resultado da localização é armazenado na entidade "*User Position*". Todos os pontos devolvidos pela *Query*, utilizados para determinar a posição do utilizador, são arquivados na entidade "U*ser Position Used Points*", de modo a ficarem associados ao resultado da localização. A atualização ou criação de elementos na entidade "*User Position Optimization*" efetua-se automaticamente, sempre que uma nova entrada for inserida em "*User Position*". Não se recorre a nenhum *trigger* para o efeito,

apenas verifica-se se a entrada antiga associada ao utilizador corresponde à mesma divisão ou não. Se corresponder, não se efetua nenhuma alteração. Caso contrário, atualiza-se a sua data de saída e cria-se uma nova entrada associada à nova divisão.

Os dados referentes à localização dos utilizadores podem depois ser consultados e processados, de forma a se criarem históricos de localização, através do UM, ou elementos estatísticos que facilitem a sua análise, a partir do SM.

Módulo "Serviços Externos" e Interface com os Serviços Externos

A implementação, gestão e disponibilização de serviços a sistemas externos são efetuados pelo ESM e pela ESI. Como descrito no subcapítulo 4.1.3.2, é necessária a repartição das funcionalidades em dois projetos diferentes devido às diferenças tecnológicas associadas à disponibilização dos serviços e à sua interface de gestão. A solução desenvolvida para se interligar o ESM e a ESI recorre a um mecanismo de sincronização dos serviços disponibilizados e implementados na ESI, com os serviços geridos através do ESM.

O mecanismo de sincronismo é um processo bastante simples, detetando apenas os serviços externos do WiLOS existentes no IIS e adicionando ao ISM aqueles que ainda não fazem parte da BD. Note-se que o objetivo deste trabalho não consiste na implementação de mecanismos de sincronismo de alto nível. Isto é, o ideal seria que o ESM fosse capaz de detetar alterações nos serviços extremos disponibilizados, de modo a reassociar a informação existente na BD com a configuração de serviços atual. Desta forma, caso algum serviço externo fosse removido ou substituído por outro, o ESM seria capaz de identificar esse acontecimento e agir de forma a garantir a integridade do ESI.

A solução implementada garante a funcionalidade da ESI e do ESM com uma qualidade de serviço mínima. Caso sejam efetuadas alterações nos serviços externos, o utilizador tem de reconfigurar o ESM para garantir o seu funcionamento, bem como o funcionamento dos sistemas que dependem da informação do WiLOS. O encadeamento de ações necessário para o sincronismo de serviços é ilustrado na figura 4.35.

Um serviço disponibilizado pela ESI, para além de satisfazer o pedido efetuado pelo sistema externo, tem de garantir a privacidade e segurança da informação do WiLOS. Uma vez que o ESM permite a gestão de sistemas externos autorizados a aceder a estes serviços, é possível implementar mecanismos que verifiquem a autenticidade e a autorização de um sistema durante o seu consumo. O funcionamento interno de um serviço disponibilizado pelo WiLOS obedece à estrutura do diagrama de atividade da figura 4.36.

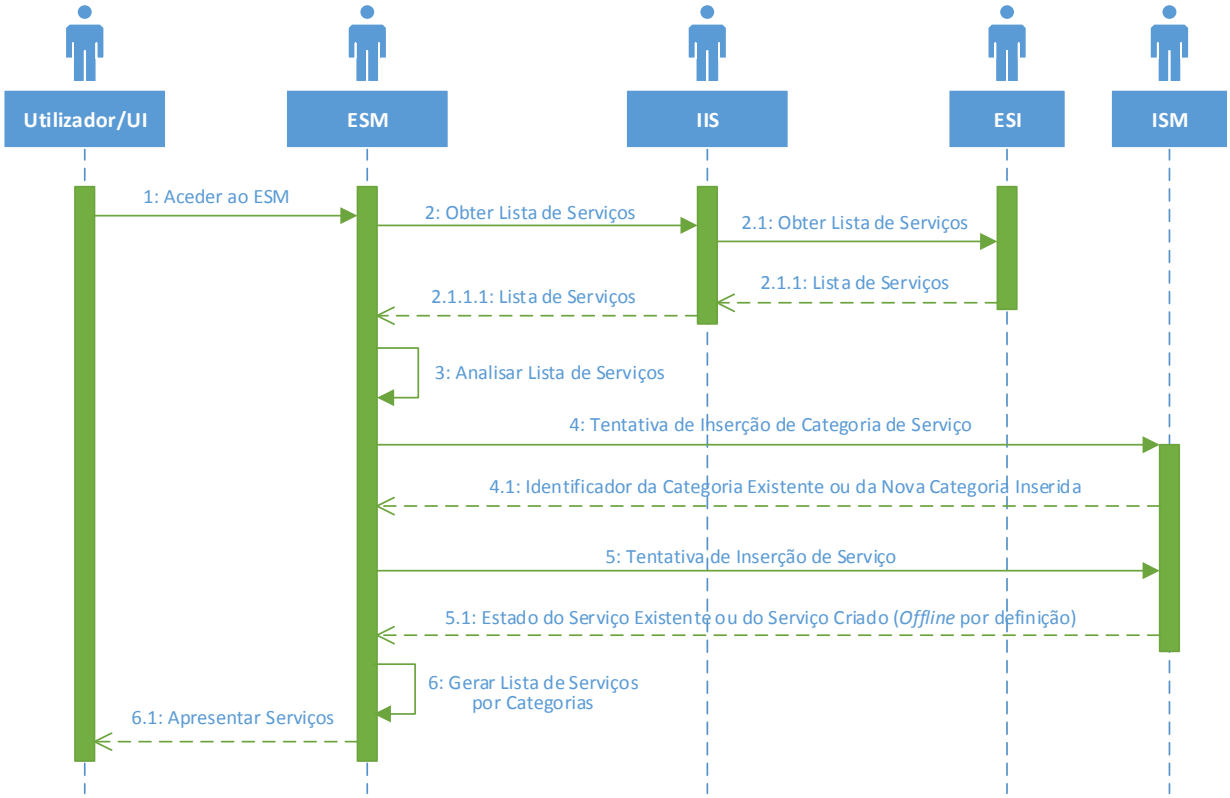


Figura 4.35 - Diagrama de sequência relativo ao mecanismo de sincronismo de Serviços Externos do mWiLOS.

As respostas aos serviços externos são efetuadas utilizando um vetor de dois objetos. A primeira posição do vetor possui um valor "inteiro" que é utilizado para indicar se ocorreu algum erro durante a sua execução (campo erro). Se não se verificar nenhum erro, a segunda posição do vetor irá conter a informação requisitada ao serviço pelo sistema externo (campo informação).

O campo erro pode assumir 4 valores. O valor '0' indica que a operação foi realizada com sucesso. A indisponibilidade de um serviço é dada pelo valor '1'. A ocorrência do valor '2' significa que o sistema externo não é reconhecido pelo WiLOS. Por fim, o valor '3' é utilizado para indicar que o sistema não possui privilégios para aceder ao serviço em questão.

A verificação da autorização e autenticação de um sistema externo no mWiLOS é realizada através da utilização de um cabeçalho SOAP especial (classe "_ExternalSystemCredentials"). Este cabeçalho define 2 campos adicionais: o nome do sistema e uma palavra-passe. Estes valores são comparados com os sistemas registados no ISM do mWiLOS, de modo a garantir que a informação é acedida somente por sistemas conhecidos. No entanto, um sistema autenticado não significa que possua autorização para aceder a toda a informação disponibilizada. Esta distinção entre autenticação e autorização permite a definição dos valores de erro '2' e '3'.

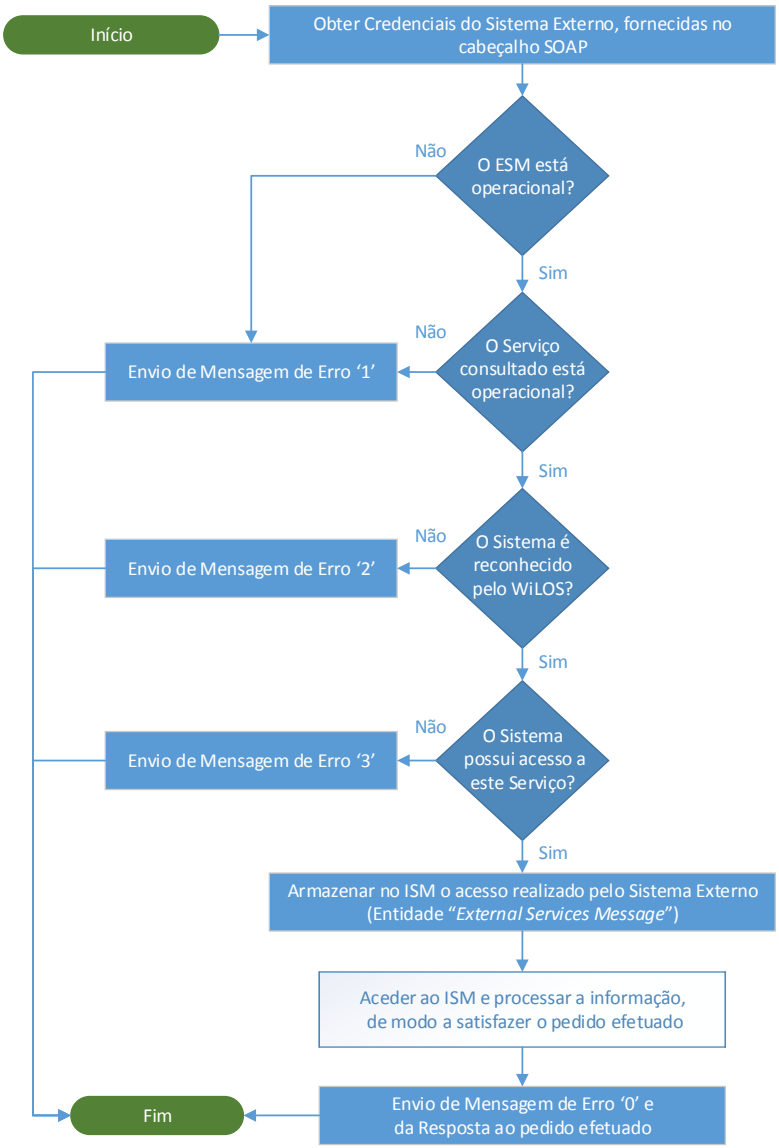


Figura 4.36 - Diagrama de atividade referente ao funcionamento interno de um serviço externo disponibilizado pelo WiLOS.

Para normalizar os serviços externos, todas as respostas a pedidos são efetuadas sob a forma de objetos XML no campo informação. Para se extrair a informação do objeto XML é necessário saber o tipo de informação que este representa. Os tipos de objetos possíveis de serem trocados com WiLOS encontram-se implementados na classe "_WiLOS_Objects". Desta forma, sempre que a resposta de um serviço for positiva, o sistema externo utiliza esta classe, ou uma semelhante, para converter a resposta XML no objeto correto. Esta classe também implementa mecanismos que efetuam a conversão inversa, isto é, do objeto do tipo "WiLOS" para um objeto XML. Esta conversão é necessária para se enviar o objeto original para outro sistema através do serviço externo. Uma lista com todos os serviços externos do WiLOS, e respetivas respostas, é disponibilizada para consulta no anexo I. A implementação destes serviços é realizada através da classe "_WiLOS_ExternalServices".

Como referido no subcapítulo 4.1.3.2, toda a informação trocada entre o WiLOS e os sistemas externos é efetuada através de uma ligação segura, implementada no IIS através de um certificado digital SSL. Um exemplo de implementação e de consumo de um *webservice* é realizado no anexo L.

4.1.4 Implementação – sWiLOS

O funcionamento do WiLOS depende de 2 subsistemas, o mWiLOS e o sWiLOS. Ao longo deste subcapítulo detalha-se a implementação do sWiLOS através da definição das tecnologias utilizadas para o seu desenvolvimento. As soluções tecnológicas escolhidas são aplicadas aos diferentes módulos que compõem a sua arquitetura, de modo a garantir que o sWiLOS cumpra os objetivos para os quais foi projetado.

4.1.4.1 Tecnologias Adotadas

Do mesmo modo que para o mWiLOS, recorre-se ao Microsoft Visual Studio (MVS) como plataforma de desenvolvimento. No entanto, a implementação do sWiLOS implica a utilização da versão 2008 do MVS. Esta versão é a única que ainda permite o desenvolvimento de aplicações móveis compatíveis com o sistema operativo Windows Mobile 6.5, desde que se possuam as bibliotecas necessárias (Windows Mobile 6 SDKs e Windows Mobile 6.5.3 DTK). A escolha deste sistema operativo deve-se a dois fatores. O primeiro é a necessidade de se aceder ao *hardware* do dispositivo móvel para se implementar as funcionalidades desejadas, mais nomeadamente o acesso ao módulo de Wi-Fi. No Windows Mobile 6.5 esta potencialidade já tinha sido confirmada em trabalhos realizados por outros autores. O segundo fator é o material disponibilizado pela universidade para efetuar a implementação e teste do WiLOS. A existência de 3 *smartphones* HTC HD Mini com o sistema operativo referido acabou por ser um fator determinante.

A gestão e organização da informação no sWiLOS também é efetuada a partir de bases de dados (BD). Ao contrário do mWiLOS, o sistema de gestão de bases de dados (SGBD) utilizado é do tipo SQL, fornecido pelo SQL Server CE. A utilização de dois SGBD diferentes deve-se à inexistência de SGBD MySQL compatíveis com o Windows Mobile 6.5. Os mecanismos de comunicação utilizados pelo sWiLOS para manipular a BD são disponibilizados pelo próprio MVS.

Como já mencionado, a comunicação entre mWiLOS e sWiLOS ocorre através do protocolo TCP/IP. O sWiLOS funciona como Escravo, sendo capaz de estabelecer uma ligação com o mWiLOS de modo a ser alvo de monitoração e efetuar pedidos ao Mestre. O protocolo TCP/IP garante a entrega e a ordem de todos os dados trocados entre ambos os subsistemas.

4.1.4.2 Pormenorização do Modelo Arquitetural

O sWiLOS consiste numa aplicação do tipo *Windows Form* que, tal como o mWiLOS, encontra-se estruturada em nível de apresentação e nível comportamental. Ambos são implementados através da linguagem C#, de modo a definir a interface gráfica da UI e as funcionalidades dadas pelos restantes módulos da camada Interface e camada Controlo. A interligação entre os dois níveis ocorre através de eventos, que podem ser originados pelo utilizador (UI), pelo mWiLOS (MSI) ou pelo próprio sWiLOS, de forma a garantir a execução de todo o WiLOS.

A camada Entidade, composta pelo módulo ISM, é implementada pela classe "_sWiLOS_Database" e pela BD do sWiLOS, localizada no SGBD SQL do dispositivo móvel. Esta classe define todas as funcionalidades que permitem aceder e gerir a BD, recorrendo às linguagens de programação C# e SQL. A BD do sWiLOS resulta da concretização do DER do subcapítulo 4.1.3.2.

Estrutura da Camada Interface

A interface gráfica, dada pelo módulo UI, deve ser intuitiva e facilmente manipulável pelo polegar do utilizador. Note-se que se trata de uma aplicação executada num dispositivo móvel, um aparelho que atualmente possui à sua disposição um ecrã tátil de dimensões relativamente reduzidas. Este não só limita a quantidade de informação que pode ser disponibilizada ao utilizador de uma só vez, como também implica a definição de um modelo de apresentação de dados melhor estruturado. As várias classes responsáveis pela estruturação da UI, bem como das páginas iniciais apresentadas ao utilizador, pertencem ao *namespace* "_Interface", que pode ser consultado na figura 4.37.

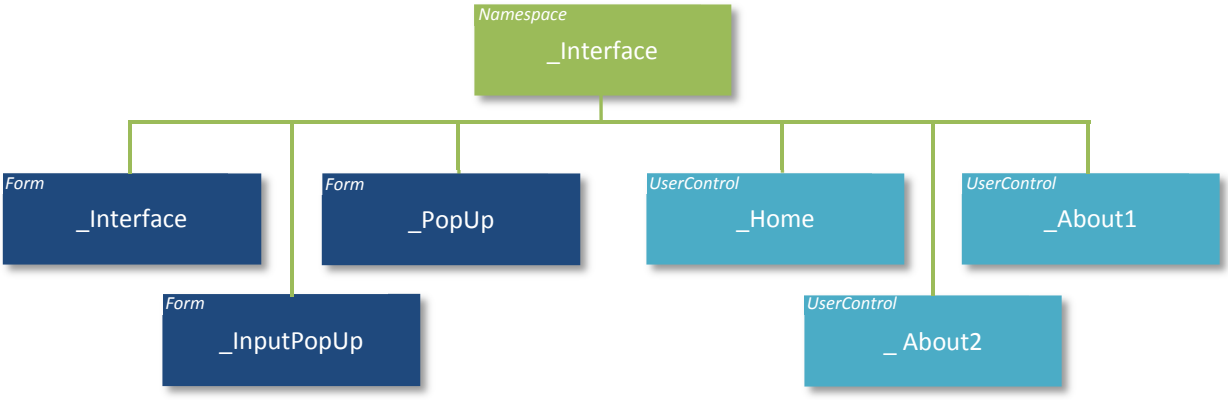


Figura 4.37 - Organização das classes que definem o *layout* gráfico da UI e a página de entrada do sWiLOS.

A classe "_Interface" define a janela de interação principal com o sWiLOS. A barra de navegação permite ao utilizador aceder aos diversos módulos que constituem a camada Controlo do sWiLOS, e às funcionalidades por eles implementadas. Todas as classes do tipo "*UserControl*", que definem a

camada Controlo, possuem um *layout* gráfico único conforme os mecanismos disponibilizados ao utilizador. Alguns os exemplos de interface gráfica do sWiLOS podem ser consultados no anexo G.

Durante a execução do sWiLOS pode ser necessário que o utilizador efetue uma tomada de decisão, insira algum valor necessário para a execução de uma determinada funcionalidade ou, simplesmente, ser alertado para algum erro ou problema que ocorra no sWiLOS. Para tal, as classes "_PopUp" e "_InputPopUp" definem uma janela de menores dimensões que se destaca da janela de interação principal, para providenciar uma das operações supracitadas. Após a sua execução, a janela é automaticamente fechada e o sWiLOS volta a disponibilizar a janela de interação principal.

As classes "_Home", "_About1" e "_About2" representam a página inicial apresentada ao utilizador, bem como algumas informações referentes ao projeto WiLOS e os seus autores. Estas não implementam funcionalidades de nenhum módulo específico, apenas disponibilizam informação trivial ao utilizador.

A comunicação entre sWiLOS e mWiLOS ocorre através da "Interface Mestre-Escravo" (MSI). A MSI do sWiLOS implementa os mecanismos típicos de um cliente TCP/IP. Desta forma, o sWiLOS pode tentar estabelecer uma ligação com o mWiLOS e aguardar a receção de eventuais pedidos. Esta interface também possibilita o envio de pedidos para o mWiLOS, que podem ou não ser realizados dependendo do tipo de pedido e do nível de ocupação do mWiLOS. A classe utilizada para definir as funcionalidades de cliente TCP/IP denomina-se "_Client". As mensagens passíveis de serem trocadas através da MSI encontram-se implementadas na classe "_CommunicationProtocol".

Os vários tipos de mensagens utilizados pelo sWiLOS ou pelo mWiLOS obrigam à definição de um protocolo de comunicação, de modo a garantir a integridade da comunicação. Este, para além de definir a estrutura de uma mensagem, também descreve quem é que envia ou recebe determinado tipo de mensagens, bem como a sequência que se deve respeitar para garantir as funcionalidades das MSI de ambos os subsistemas. Uma mensagem trocada entre sWiLOS e mWiLOS obedece à estrutura seguidamente apresentada.

```
Código de Acesso /* Tipo Primário /* Tipo Secundário /* Data /* Remetente /* Destinatário /* Conteúdo
```

O primeiro campo da mensagem recorre a um identificador de acesso para a garantir que a mensagem está a ser enviada/recebida por uma entidade autenticada e autorizada. Este identificador é atribuído durante o processo de emparelhamento inicial entre o sWiLOS e o mWiLOS, sendo atualizado sempre que um dispositivo móvel válido se associe ao mWiLOS.

O campo "Tipo de Mensagem Primário" permite categorizar as mensagens de acordo com a natureza da informação tratada ou com as funcionalidades que podem ser por elas desencadeadas. Por exemplo, todas as mensagens que lidam com a informação dos utilizadores têm o tipo de mensagem primário "MU", *Manage Users*. Todas as mensagens trocadas durante o processo de emparelhamento são do tipo primário "DS", *Device Setup*. O identificador secundário de tipo de mensagem, dado pelo terceiro campo, permite especificar uma mensagem dentro da categoria de mensagem principal. Desta forma, as mensagens tornam-se específicas, sendo utilizadas para efetuar uma determinada ação, como obter algum tipo de informação ou responder a um determinado pedido.

O quarto campo da mensagem possui a indicação horária referente à data de envio da mesma. Consoante a natureza do tipo de mensagem, os campos 5 e 6 possuem informação relacionada com o remetente e o destinatário. Esta informação é de grande importância para os serviços de troca de mensagens, por exemplo. O último campo possui o conteúdo da mensagem. Devido à variabilidade deste elemento, a sua dimensão varia consoante o tipo de pedido realizado e a informação necessária para o satisfazer. Antes do envio de uma mensagem através da MSI, efetua-se o cálculo do tamanho da mesma, de modo a indicar ao recetor a quantidade de informação que vai ser transmitida.

Seguidamente apresentam-se as listas que discriminam todas as categorias principais e secundárias que podem ser trocadas pela MSI, bem como o seu conteúdo. As várias sequências de mensagens necessárias para o funcionamento do WiLOS são apresentadas no subcapítulo 4.1.4.4, referente ao modelo comportamental do sWiLOS. Aqui pode-se consultar o processo de emparelhamento de um dispositivo móvel com maior detalhe, ou como se processa o reencaminhamento de uma mensagem para outro dispositivo sWiLOS.

Categorias Primárias de Mensagens

DS - Device Setup

- Mensagens para emparelhamento, autenticação e troca de informação de configuração entre o dispositivo móvel (sWiLOS) e o servidor (mWiLOS).

MU - Manage Users

- Mensagens para gestão de eventos e informação dos utilizadores da rede WiLOS.

CS - Chat Service

- Mensagens dedicadas ao serviço de troca de mensagens entre utilizadores do WiLOS.

SSM - Signal Strength Map

- Mensagens trocadas durante o processo de definição do mapa de potência do sinal da rede WiLOS.

MS - Monitoring Service

- Mensagens que garantem a determinação do posicionamento de um utilizador dentro da rede WiLOS.

SS - Statistics Service

- Mensagens utilizadas para satisfazer os pedidos de informação estatística por parte dos utilizadores dos dispositivos móveis (sWiLOS).

HT - House Topology

- Mensagens para gestão de informação relacionada com a habitação.

Categorias Secundárias de *DS, Device Setup*

DSS - Device Setup Start

Mensagem utilizada para notificar o mWiLOS que um dispositivo encontra-se disponível para efetuar o processo de emparelhamento.

Enviada por: sWiLOS
Data de Envio: Obrigatória
Conteúdo: Vazio

Recebida por: mWiLOS
Identificador: 0

DSF - Device Setup Finish

Mensagem utilizada para notificar o mWiLOS que um dispositivo encontra-se configurado e que o processo de emparelhamento pode terminar.

Enviado por: sWiLOS (-1)
Data de Envio: Obrigatória
Conteúdo: Vazio

Recebido por: mWiLOS
Identificador: 0

PCR - Pairing Code Request

Mensagem utilizada para indicar ao sWiLOS que o mWiLOS aguarda a introdução do código de emparelhamento por parte do utilizador.

Enviado por: mWiLOS
Data de Envio: Obrigatória
Conteúdo: Vazio

Recebido por: sWiLOS (-1)
Identificador: 0

UIR - User Information Request

Mensagem enviada pelo sWiLOS de modo a responder ao "PCR" e a transmitir ao mWiLOS a informação referente ao dispositivo móvel. É efetuado um pedido de informação ao mWiLOS referente aos dados do utilizador dono do dispositivo móvel.

Enviado por: sWiLOS (-1)
Data de Envio: Obrigatória
Conteúdo: Dados do Dispositivo

Recebido por: mWiLOS
Identificador: Código Introduzido

UIA - User Information Answer

Mensagem enviada para o sWiLOS como resposta ao "UIR". É atribuído ao dispositivo do utilizador num novo código gerado pelo mWiLOS.

Enviado por: mWiLOS
Data de Envio: Obrigatória
Conteúdo: Código Atribuído e Dados do Utilizador

Recebido por: sWiLOS (-1)
Identificador: Código Introduzido

HIR - House Information Request

Mensagem utilizada pelo sWiLOS para requisitar ao mWiLOS a informação relativa à habitação e aos utilizadores existentes.

Enviado por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Conteúdo: Vazio

Recebido por: mWiLOS
Identificador: Código Atribuído

HIA - House Information Answer

Mensagem enviada para o sWiLOS como resposta ao "HIR". É transmitida ao sWiLOS toda a informação habitacional e relacionada com os utilizadores do WiLOS necessária para o seu funcionamento.

Enviado por: mWiLOS
Data de Envio: Obrigatória
Conteúdo: Dados da Habitação e dos Utilizadores

Recebido por: sWiLOS (ID Utilizador)
Identificador: Código Atribuído

Categorias Secundárias de *MU, Manage Users*

UIR - User Information Request

Mensagem utilizada pelo sWiLOS de modo a obter informações referentes a um utilizador.

Enviado por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Conteúdo: ID do Utilizador a atualizar

Recebido por: mWiLOS
Identificador: Código Atribuído

UIA - User Information Answer

Mensagem de resposta à mensagem "UIA". O mWiLOS transmite ao sWiLOS a informação do utilizador requisitado.

Enviado por: mWiLOS
Data de Envio: Obrigatória
Conteúdo: Dados do Utilizador

Recebido por: sWiLOS (ID Utilizador)
Identificador: Código Atribuído

UON - User Online

Mensagem utilizada para alertar a entrada de um utilizador na rede WiLOS. Caso o mWiLOS seja recetor, significa que um utilizador deseja regressar à rede. Caso o sWiLOS seja recetor, então um utilizador acabou de entrar na rede.

Enviado por: mWiLOS
Recebido por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: ID do Utilizador Online

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Vazio

UOFF - User Offline

Mensagem de aviso responsável por alertar o mWiLOS e o sWiLOS que um determinado utilizador saiu da rede. Caso seja o mWiLOS o recetor, significa que um utilizador saiu da rede voluntariamente. Caso seja o sWiLOS o recetor, então um utilizador acabou de sair da rede.

Enviado por: mWiLOS
Recebido por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: ID do Utilizador Offline

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Vazio

USL - Users List

Mensagem utilizada pelo sWiLOS para a obtenção da lista de todos os utilizadores existentes no WiLOS.

Enviado por: mWiLOS
Recebido por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Lista de Utilizadores

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Vazio

UVR - User Verification Request

Após a tentativa de entrada de um utilizador na rede WiLOS (mensagem "UON"), o mWiLOS efetua um pedido de informação ao sWiLOS de modo a poder proceder à sua autenticação.

Enviado por: mWiLOS
Data de Envio: Obrigatória
Conteúdo: Vazio

Recebido por: sWiLOS (ID Utilizador)
Identificador: Código Atribuído

UVA - User Verification Answer

Mensagem de resposta à mensagem "UVR". O sWiLOS envia para o mWiLOS o seu nome de utilizador e o endereço MAC do dispositivo, de modo a continuar com o processo de autenticação.

Enviado por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Conteúdo: Nome de Utilizador e endereço MAC

Recebido por: mWiLOS
Identificador: Código Atribuído

UA - User Accepted

Mensagem de resposta à mensagem "UVA". Caso o mWiLOS confirme a identidade do sWiLOS, este aceita-o na rede WiLOS e atribuí-lhe um novo código de acesso.

Enviado por: mWiLOS
Data de Envio: Obrigatória
Conteúdo: Vazio

Recebido por: sWiLOS (ID Utilizador)
Identificador: Código Atribuído

Categorias Secundárias de *HT, House Topology*

HI - House Information

Mensagem utilizada pelo sWiLOS para a obtenção dos dados referentes à habitação. O sWiLOS efetua o pedido e o mWiLOS responde com as informações desejadas.

Enviado por: mWiLOS
Recebido por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Dados da Habitação

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Vazio

Categorias Secundárias de *SSM, Signal Strength Map*

SSB - Signal Strength Map Begin

Mensagem utilizada para indicar ao sWiLOS que o mWiLOS necessita de efetuar a calibração do mapa de potência do sinal da habitação.

Enviado por: mWiLOS
Data de Envio: Obrigatória
Conteúdo: Vazio

Recebido por: sWiLOS (ID Utilizador)
Identificador: Código Atribuído

SSF - Signal Strength Map Finish

Mensagem enviada para o mWiLOS ou para o sWiLOS de modo a indicar que o processo de calibração do mapa de potência do sinal foi concluído. Cabe ao mWiLOS verificar se este foi concluído com sucesso ou se ainda faltam calibrar alguns pontos.

Enviado por: mWiLOS
Recebido por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Vazio

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Vazio

SSH - Signal Strength Map House Points

Mensagem utilizada no início do processo de calibração do mapa de potência do sinal da habitação. Envia para o sWiLOS toda a informação referente aos pontos que definem a tipologia da habitação.

Enviado por: mWiLOS
Data de Envio: Obrigatória
Conteúdo: Pontos que definem a Tipologia da Habitação.

Recebido por: sWiLOS (ID Utilizador)
Identificador: Código Atribuído

SSN - Signal Strength Map North

Envio para o sWiLOS dos pontos necessários para efetuar a calibração do mapa de potência do sinal no sentido Norte.

Enviado por: mWiLOS
Data de Envio: Obrigatória
Conteúdo: Pontos que compõem o mapa de potência do sinal “Norte“

Recebido por: sWiLOS (ID Utilizador)
Identificador: Código Atribuído

SSS - Signal Strength Map South

Envio para o sWiLOS dos pontos necessários para efetuar a calibração do mapa de potência do sinal no sentido Sul.

Enviado por: mWiLOS
Data de Envio: Obrigatória
Conteúdo: Pontos que compõem o mapa de potência do sinal “Sul“

Recebido por: sWiLOS (ID Utilizador)
Identificador: Código Atribuído

SSE - Signal Strength Map East

Envio para o sWiLOS dos pontos necessários para efetuar a calibração do mapa de potência do sinal no sentido Este.

Enviado por: mWiLOS
Data de Envio: Obrigatória
Conteúdo: Pontos que compõem o mapa de potência do sinal “Este“

Recebido por: sWiLOS (ID Utilizador)
Identificador: Código Atribuído

SSW - Signal Strength Map West

Envio para o sWiLOS dos pontos necessários para efetuar a calibração do mapa de potência do sinal no sentido Oeste.

Enviado por: mWiLOS
Recebido por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Pontos que compõem o mapa de potência do sinal "Oeste"

SS - Sample Send

Mensagem utilizada para o envio de uma amostra da força do sinal, obtida num determinado ponto da habitação durante um determinado tempo.

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
D Data ate de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Pontos Amostrados e Valores do Sinal Obtidos

API - Access Point

Mensagem enviada pelo sWiLOS durante o processo de calibração do mapa de potência para transmitir a informação referente a um novo ponto de acesso detetado.

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Dados do Ponto de Acesso Detetado

Categorias Secundárias de MS, Monitoring Service

SSR - Signal Strength Request

Pedido realizado pelo mWiLOS ao sWiLOS para a obter a força do sinal verificado pelos dispositivos móveis.

Enviado por: mWiLOS
Recebido por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Vazio

SSA - Signal Strength Answer

Resposta ao pedido "SSR". O sWiLOS envia para o mWiLOS os valores de força do sinal verificado pelo dispositivo móvel.

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Endereços MAC e força do sinal dos pontos de acesso detetados

Categorias Secundárias de *CS, Chat Service*

SON - Service Online

Mensagem utilizada para determinar se o serviço de mensagens encontra-se disponível (sWiLOS - mWiLOS) ou para responder afirmativamente a um pedido de estado do serviço de mensagens (mWiLOS - sWiLOS).

Enviado por: mWiLOS
Recebido por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Vazio

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Vazio

SOFF - Service Offline

Resposta ao pedido "SON" do sWiLOS. O mWiLOS informa que o serviço de troca de mensagens encontra-se inoperacional.

Enviado por: mWiLOS
Data de Envio: Obrigatória
Conteúdo: Vazio

Recebido por: sWiLOS (ID Utilizador)
Identificador: Código Atribuído

ANC - Add New Conversation

Esta mensagem é utilizada para requisitar ao mWiLOS a criação de uma nova conversação através da informação enviada. O mWiLOS recebe o pedido e reencaminha a mensagem com a informação atualizada para todos os utilizadores de destino (incluindo o sWiLOS que realiza o pedido).

Enviado por: sWiLOS
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Conteúdo da Conversação e Utilizadores

Enviado por: sWiLOS (ID Utilizador 1)
Recebido por: sWiLOS (ID Utilizador 2)
Data de Envio: Obrigatória
Identificador: Código Atribuído (Destinatário)
Conteúdo: Conteúdo da Conversação

AUC - Add User to Conversation

Mensagem utilizada quando se pretende adicionar um novo utilizador a uma conversação existente. O sWiLOS envia para o mWiLOS qual o utilizador a adicionar e este utiliza essa informação para enviar a conversação para o novo destinatário. A partir deste momento todas as mensagens são também partilhadas com o novo utilizador.

Enviado por: mWiLOS
Data de Envio: Obrigatória
Conteúdo: Identificador da Conversação e Identificador do Novo Utilizador

Recebido por: sWiLOS (ID Utilizador)
Identificador: Código Atribuído

SMS - Short Message Send

O envio de uma mensagem de texto para uma conversação, e respetivos utilizadores, é realizado através deste tipo de mensagem. O sWiLOS envia a informação para o mWiLOS, que se encarrega de a reencaminhar para os utilizadores da conversação.

Enviado por: sWiLOS (ID Utilizador 1)
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído

Enviado por: sWiLOS (ID Utilizador 1)
Recebido por: sWiLOS (ID Utilizador 2, 3, …)
Data de Envio: Obrigatória
Identificador: Código Atribuído (Destinatário)

LCL - Lost Conversation List

Sempre que um utilizador entra na rede WiLOS, o mWiLOS verifica se este possui novas mensagens recebidas e envia-as para o utilizador. Pode ocorrer a situação em que o sWiLOS não possui a conversação desejada, sendo obrigado a requisitar informações referentes a essa conversação (mensagem do tipo "CNV").

Enviado por: mWiLOS
Data de Envio: Obrigatória
Conteúdo: Os identificadores das conversações e as mensagens não lidas

Recebido por: sWiLOS (ID Utilizador)
Identificador: Código Atribuído

CNV - Get a Conversation

A requisição da informação de uma conversação e de toda a informação a ela associada (utilizadores intervenientes e mensagens) é efetuada através desta mensagem. Desta forma, o sWiLOS pode obter os dados referentes a uma conversação desconhecida.

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: ID da Conversação

Enviado por: mWiLOS
Recebido por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Dados da Conversação

GCNV - Get Conversations

Esta mensagem é utilizada para se obterem as diversas conversações de um utilizador. O sWiLOS envia a informação refente aos filtros utilizados para a seleção das conversações e o mWiLOS devolve uma lista de 10 conversações que preenchem os requisitos. É também devolvido o número total de conversações que podem ser consultadas.

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Filtros de Seleção

Enviado por: mWiLOS
Recebido por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Lista de Conversações

ACNV - All Conversations

Mensagem que permite ao sWiLOS obter todas as conversações realizadas pelo seu utilizador. O sWiLOS executa o pedido e o mWiLOS devolve a lista de todas as conversações, mensagens e utilizadores intervenientes.

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Vazio

Enviado por: mWiLOS
Recebido por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Todas as Conversações

DCNV - Delete Conversations

Caso um utilizador deseje remover conversações através do mWiLOS ou do sWiLOS, esta mensagem é enviada, ou para o mWiLOS ou para o sWiLOS, de modo a sincronizar informação e a requerer a eliminação das conversações selecionadas.

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: IDs da Conversações

Enviado por: mWiLOS
Recebido por: c sWiLOS WiLOS (ID Utilizador)
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: IDs da Conversações

Categorias Secundárias de *SS, Statistics Service*

HDS - House and Divisions Statistics

Mensagem desenhada para transmitir ao sWiLOS as estatísticas de natureza habitacional, construídas a partir dos dados recolhidos durante o funcionamento do WiLOS. As estatísticas enviadas são de natureza habitacional.

Enviado por: mWiLOS
Recebido por: cWi sWiLOS LOS (ID Utilizador)
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Estatísticas Geradas

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
Data ate de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Vazio

US - Users Statistics

Mensagem desenhada para transmitir ao sWiLOS as estatísticas relacionadas com os utilizadores, construídas a partir dos dados recolhidos durante o funcionamento do WiLOS. As estatísticas enviadas permitem analisar o comportamento dos utilizadores.

Enviado por: mWiLOS
Recebido por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Estatísticas Geradas

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Vazio

ESS - External Service Statistics

Mensagem desenhada para transmitir ao sWiLOS as estatísticas referentes aos serviços externos, construídas a partir dos dados recolhidos durante o funcionamento do WiLOS

Enviado por: mWiLOS
Recebido por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Estatísticas Geradas

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Vazio

CSS- Chat Service Statistics

Mensagem desenhada para a transmitir ao sWiLOS as estatísticas referentes ao serviço de troca de mensagens, construídas a partir dos dados recolhidos durante o funcionamento do WiLOS.

Enviado por: mWiLOS
Recebido por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Estatísticas Geradas

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Vazio

GCS - Get Chart Statistics

Mensagem desenhada para a transmitir ao sWiLOS determinados gráficos e elementos visuais, construídos a partir dos dados recolhidos durante o funcionamento do WiLOS.

Enviado por: mWiLOS
Recebido por: sWiLOS (ID Utilizador)
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Elementos Visuais Gerados

Enviado por: sWiLOS (ID Utilizador)
Recebido por: mWiLOS
Data de Envio: Obrigatória
Identificador: Código Atribuído
Conteúdo: Elemento pretendido

Estrutura da Camada Controlo

As funcionalidades relacionadas com a manipulação de informação da habitação são implementadas pelo HTM. Este módulo é representado pelo *namespace* "_House" e engloba as classes aprestadas na figura 4.38.

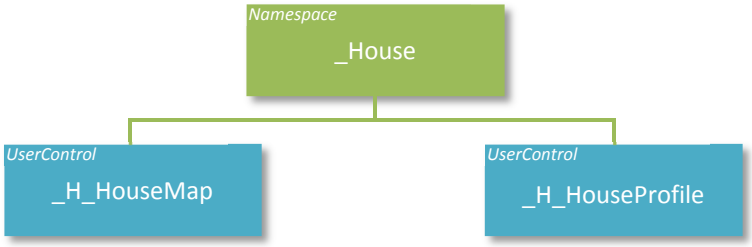


Figura 4.38 - Organização das classes que constituem o HTM da camada Controlo do sWiLOS.

O HTM apenas permite ao utilizador a consulta de informação. Desta forma, a classe "_H_HouseMap" possibilita a visualização da tipologia da habitação, enquanto a classe "_H_HouseProfile" disponibiliza os dados referentes ao perfil da habitação (morada, contactos, etc.).

A classe "_H_HouseMap" também é utilizada para se visualizar o resultado do processo de monitorização do WiLOS, isto é, a localização dos utilizadores na habitação. A sincronização da informação referente à habitação com o mWiLOS é realizada sempre que um utilizador entra na rede, através da MSI. As localizações atuais dos utilizadores não são sincronizadas automaticamente, devendo o utilizador requerer a visualização deste tipo de informação.

A gestão dos utilizadores registados no WiLOS é realizada pelo UM. Este módulo permite visualizar todos os utilizadores existentes, bem como os seus perfis WiLOS. A listagem de utilizadores é implementada pela classe "_U_Home", enquanto o perfil de utilizador, o perfil WiLOS e o perfil do dispositivo móvel são disponibilizados por "_U_UserProfile", "_U_WiLOSProfile" e "_U_DeviceProfile", respetivamente. O *namespace* "_Users", que agrupa as classes citadas, encontra-se estruturado de acordo com a figura 4.39.

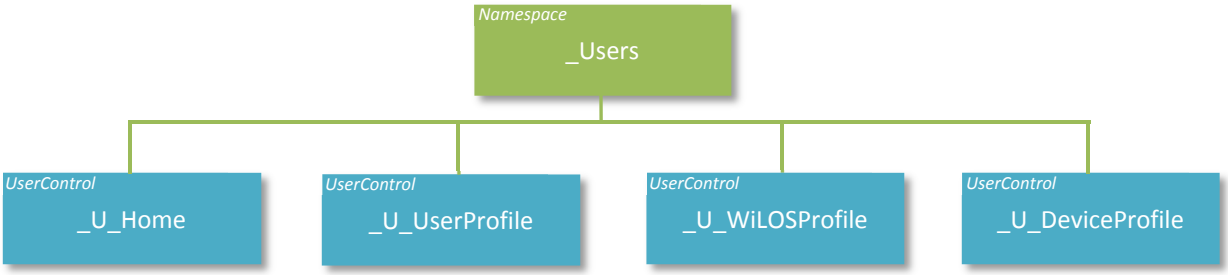


Figura 4.39 - Organização das classes que definem o UM da camada Controlo do sWiLOS.

A classe "_U_Home" também possibilita ao utilizador o acesso ao serviço de mensagens e ao módulo que o implementa, o CSM. A troca de mensagens entre utilizadores, ou uma conversação, é efetuada através da classe "_C_Conversation". Caso o utilizador deseje consultar o histórico de conversações efetuadas, a classe "_C_History" recorre à MSI para requisitar as conversações antigas ao mWiLOS, e à UI para apresentá-las ao utilizador. A visualização de uma destas conversações é realizada pela classe "_C_History_Conversation". O *namespace* que define o CSM, bem como as suas classes, encontra-se representado na figura 4.40.

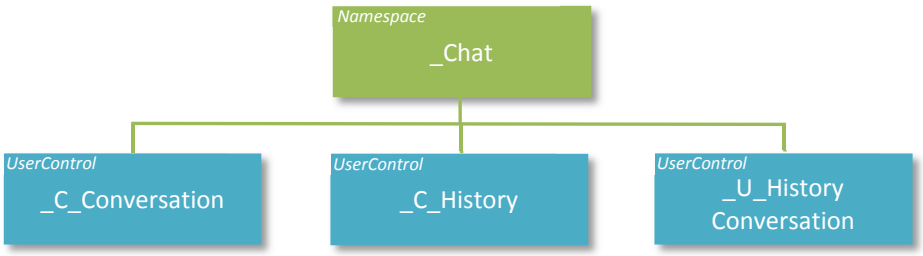


Figura 4.40 - Organização das classes que definem o CSM da camada Controlo do sWiLOS.

O SSMM, o MM e a MSI do sWiLOS são implementados através do *namespace* "_Monitor", e engloba as funcionalidades relativas à calibração do mapa de potência do sinal da habitação e à recolha dos valores de *RSSI* evidenciados pelo dispositivo móvel. Para além disso, também possui os elementos que possibilitam o emparelhamento do sWiLOS com o mWiLOS. Na fase de modelação propôs-se que o emparelhamento fosse da responsabilidade do UM mas, por motivos de organização da aplicação, optou-se pela sua definição neste *namespace*. A estrutura do *namespace* "_Monitor" é apresentada na figura 4.41.

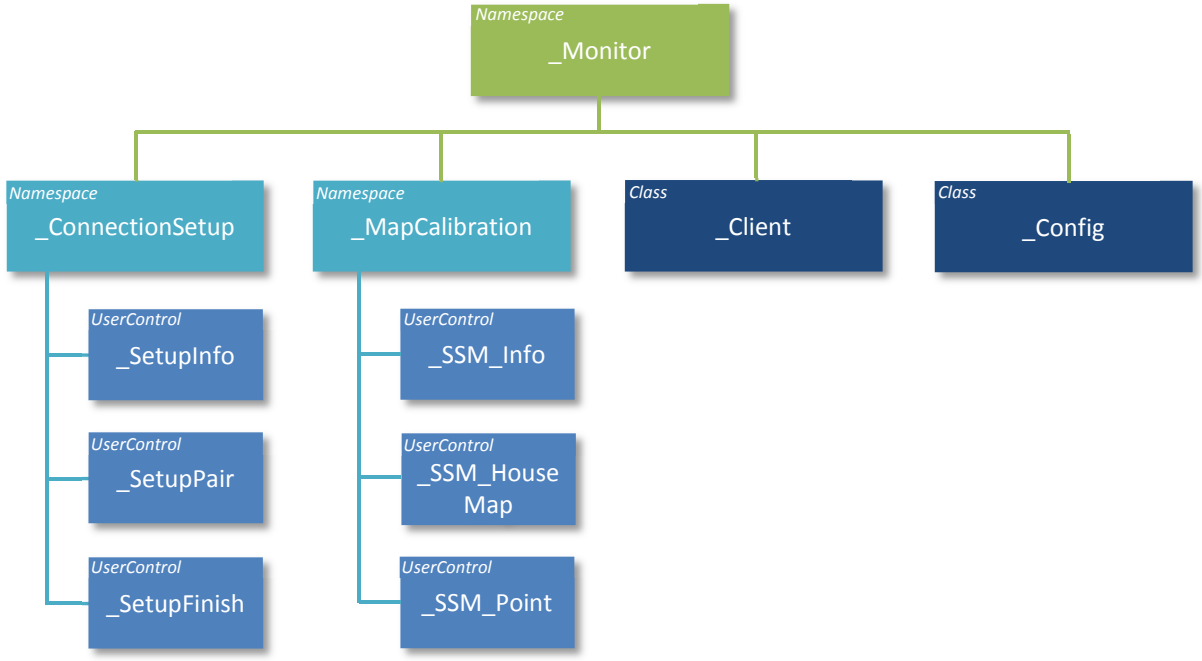


Figura 4.41 - Organização das classes que definem o MM, o SSMM a MSI do sWiLOS.

O emparelhamento de um dispositivo recorre às classes "_SetupInfo", "_SetupPair" e "_SetupFinish" do *namespace* "_ConnectionSetup". A primeira classe é a página inicial apresentada ao utilizador quando este executa o sWiLOS, caso o dispositivo nunca tenha sido emparelhado com o mWiLOS. A segunda classe é utilizada durante o processo de troca de mensagens entre sWiLOS e mWiLOS, mediada pelo utilizador, para se realizar o emparelhamento do dispositivo. A classe "_setupFinish" mostra ao utilizador o resultado da operação, de modo a proceder ou re-efetuar o emparelhamento.

A classe "_Client", como já descrito, implementa os mecanismos cliente TCP/IP que determinam o funcionamento da MSI. Também incorpora as funcionalidades que permitem a recolha e o envio dos valores de *RSSI* detetados pelo dispositivo, sempre que o mWiLOS os solicitar. A classe "_Config" permite a configuração dos dados relativos ao servidor do mWiLOS, de modo a garantir a comunicação entre as MSI de ambos os subsistemas. O uso da biblioteca Smart Device Framework, disponibilizada

pela OpenNETCF Consulting, permite o acesso ao módulo Wi-Fi do dispositivo móvel, de modo a se detetarem os *APs* pertencentes à habitação, e os valores de *RSSI* a eles associados. A determinação da orientação do dispositivo móvel é realizada através da Windows Mobile Unified Sensor API, disponibilizada em sensorapi.codeplex.com, que recorre ao acelerómetro incorporado no dispositivo para esse fim.

A calibração do WiLOS, ou obtenção dos mapas de potência do sinal da habitação, realiza-se através das classes do *namespace* "_MapCalibration". Uma pequena descrição do processo de calibração é efetuada pela classe "_SSM_Info". O mapa de calibração da habitação é apresentado através da classe "_SSM_House_Map". Aqui, o utilizador pode consultar todos os pontos calibrados, bem como aqueles ainda por calibrar. A seleção de um ponto do mapa redireciona o utilizador para a classe "_SSM_Point". A informação referente à distância deste ponto ao canto superior direito da divisão, ao qual pertence, fornece ao utilizador uma referência geográfica adicional.

O processo de calibração, para um ponto, consiste na recolha dos valores de *RSSI* verificados pelo dispositivo para cada orientação cardeal e vertical/horizontal. Cada orientação obriga a que o utilizador se encontre estacionário durante 10 s, de modo a se obterem 10 valores de *RSSI* representativos de cada orientação. Este conjunto de amostras é enviado para o mWiLOS que infere o seu valor médio e desvio padrão, de modo a armazenar toda a informação relativa ao ponto e compor o mapa de potência do sinal. Durante este processo são dadas ao utilizador várias informações para garantir que este executa todas as orientações e confirme a sua posição antes da recolha de dados.

A visualização da informação recolhida e gerada pelo WiLOS sob a forma de elementos estatísticos é efetuada pelo módulo Estatísticas. Todos os elementos disponibilizados pelo mWiLOS são possíveis de ser consultados pelo sWiLOS. Para tal, o SM efetua pedidos de informação através da MSI, satisfazendo as necessidades do utilizador. A figura 4.42 ilustra o *namespace* que define o SM, bem como as classes que o constituem.

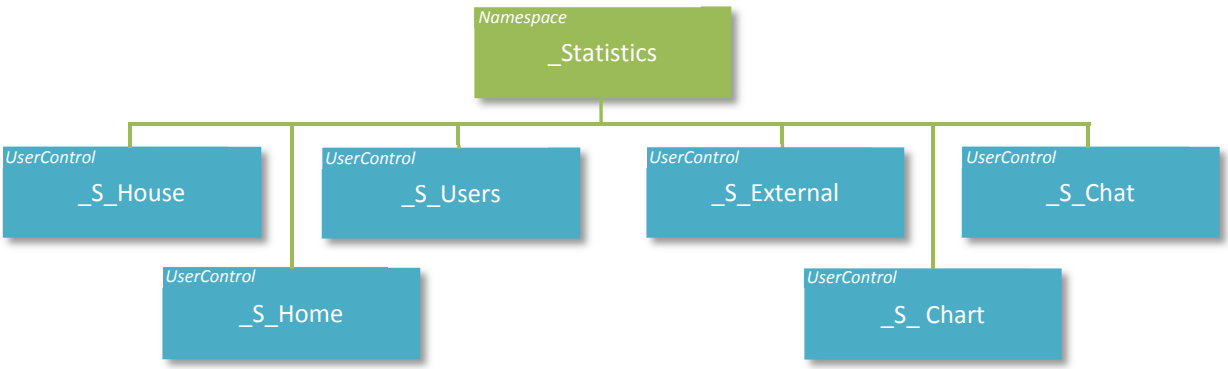


Figura 4.42 - Organização das classes que definem o SM da camada Controlo do sWiLOS.

A classe "_S_Home" é a página inicial deste módulo, sendo utilizada para aceder às diversas estatísticas relacionadas com a habitação, utilizadores, serviço de mensagens e serviços externos. Cada uma destas áreas é implementada pelas classes "_S_House", "_S_Users", "_S_Chat" e "_S_External", que permitem visualizar a informação estatística mais trivial. A consulta de um gráfico específico requer o auxílio da classe "_S_Chart", que possui mecanismos para descodificar os dados recebidos através da MSI e apresentar a informação ao utilizador.

Estrutura da Camada Entidade

A gestão, organização e integridade de toda a informação do sWiLOS ocorre ao nível da camada Entidade. Para tal, o ISM implementa todas as funcionalidades de consulta, inserção, atualização e remoção de dados da BD do sWiLOS. As várias classes que constituem o ISM obedecem à estrutura representada na figura 4.43.

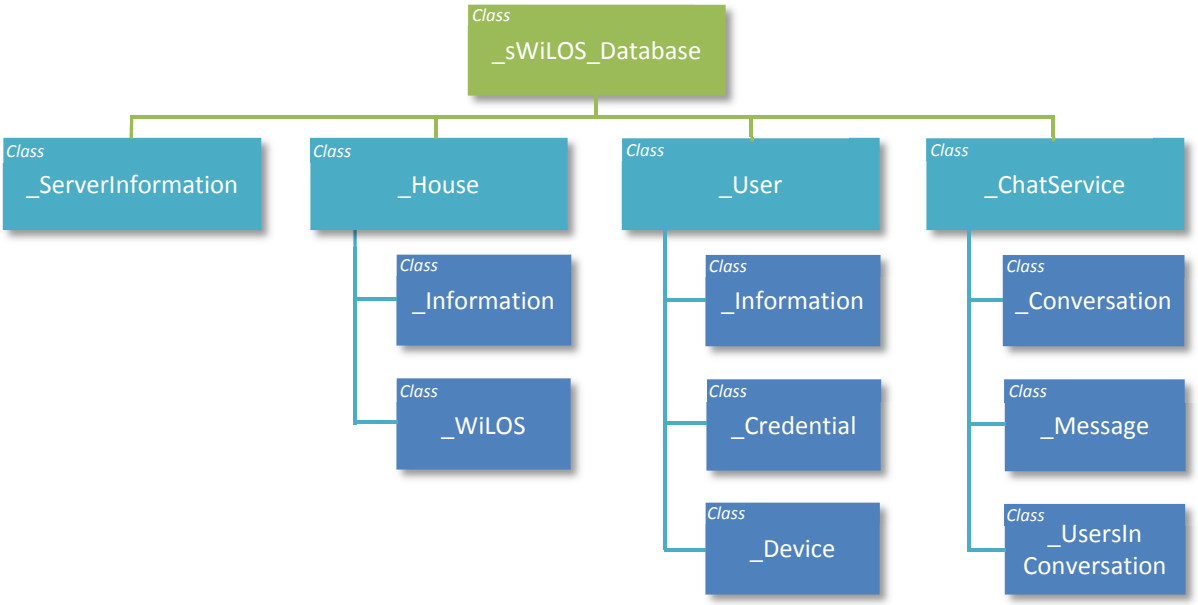


Figura 4.43 - Organização das classes que definem o ISM da camada Entidade do sWiLOS.

Uma vez que as entidades manipuladas pelo ISM do sWiLOS são as mesmas que as entidades do mWiLOS, só que em menor número, existe uma equivalência direta entre as classes aqui apresentadas e as classes do subcapítulo 4.1.3.2. Desta forma, as subclasses "_Information" e "_WiLOS" da classe "_House" são responsáveis pelas funções relativas às entidades "*House*" e "*House WiLOS*" da área "*House Information*". A área "*User Information*", composta pelas entidades "*User*", "*User Credential*" e "*User Mobile Device*", é gerida pela classe "_User" e as suas subclasses. As mensagens e conversações referentes ao serviço de mensagens são da responsabilidade das classes "_Conversation", "_Message" e "_UsersInConversation", pertencentes à classe "_ChatService". Por último, a classe "_ServerInformation" atua sobre a entidade "*Server*", permitindo a manipulação da informação referente ao servidor do mWiLOS.

4.1.4.3 Modelo de Dados – Visão Física

Uma vez que o DER do sWiLOS consiste numa réplica simplificada do DER do mWiLOS, os tipos de dados estabelecidos para os atributos das entidades do sWiLOS são os seus equivalentes SQL. Desta forma, atributos que implicam cadeias de caracteres alfanuméricos são descritos pelo tipo "NVARCHAR". O tipo "INT" é utilizado em atributos cuja informação é dada por números inteiros, como as chaves das entidades implementadas. Os elementos que representam dimensões ou áreas recorrem ao tipo "FLOAT", uma vez que um número inteiro não confere precisão suficiente.

O tipo "TINYINT" é utilizado para caracterizar atributos que assumem dois estados, enquanto os elementos temporais são representados pelo tipo "NVARCHAR". Como as imagens armazenadas no sWiLOS são cópias de menor dimensão do mWiLOS, o tipo "IMAGE" é escolhido para armazenar esta informação. A associação dos tipos de dados descritos com os atributos das entidades do sWiLOS encontra-se representada na figura 4.44. Uma vez a plataforma MySQL Workbench não permite o desenho de DER do tipo SQL, a figura 4.44 apresenta os atributos equivalentes em MySQL.

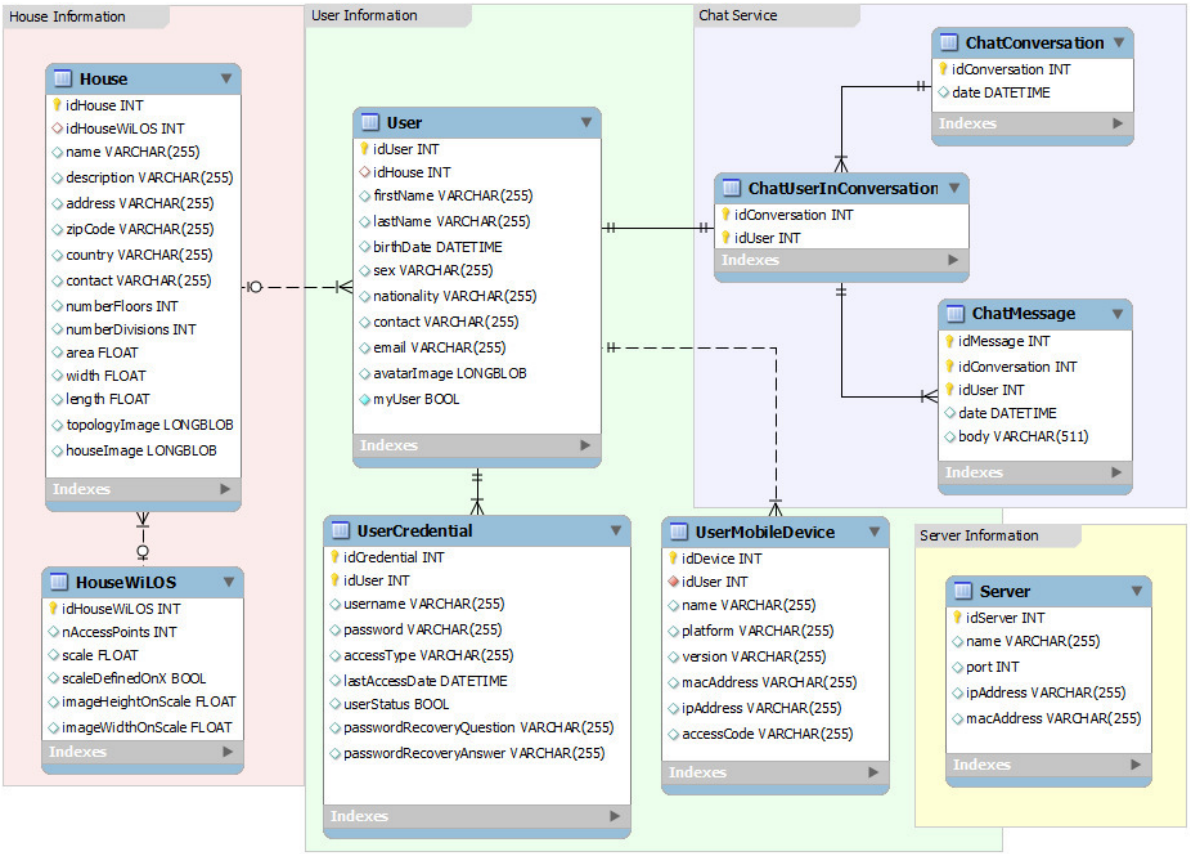


Figura 4.44 - Diagrama de entidades e relacionamentos do sWiLOS – Visão Física.

4.1.4.4 Modelo Comportamental

Ao longo deste subcapítulo detalham-se os modelos comportamentais da MSI e dos MM, SSMM e CSM. Em termos comportamentais, o HTM, o UM e o SM não introduzem grande complexidade no sistema. Estes módulos são simplesmente responsáveis por apresentar a informação armazenada no ISM através da UI. Caso a informação esteja desatualizada, o utilizador pode requerer a sua atualização, o que implica a utilização da MSI para a realização do pedido ao mWiLOS e receção da respetiva resposta, como demonstrado no diagrama de sequência da figura 4.45.

Por outro lado, a MSI é responsável por gerir toda comunicação realizada pelo sWiLOS e o mWiLOS, através do protocolo descrito em 4.1.3.2. Portanto, esta é crucial para garantir o funcionamento do sistema, com destaque para os seus mecanismos de ligação com o servidor e receção de mensagens. O MM implementas as funcionalidades que permitem inferir os valores de *RSSI* dos *APs* detetados pelo dispositivo móvel, bem como para realizar o emparelhamento do dispositivo com o mWiLOS. O SSMM recorre a funções semelhantes às do MM para efetuar a calibração dos mapas de potência do sinal da habitação. Por fim, as conversações entre utilizadores implicam a criação de conversações e o envio de mensagens entre sWiLOS, sendo pertinente a demonstração do seu funcionamento.

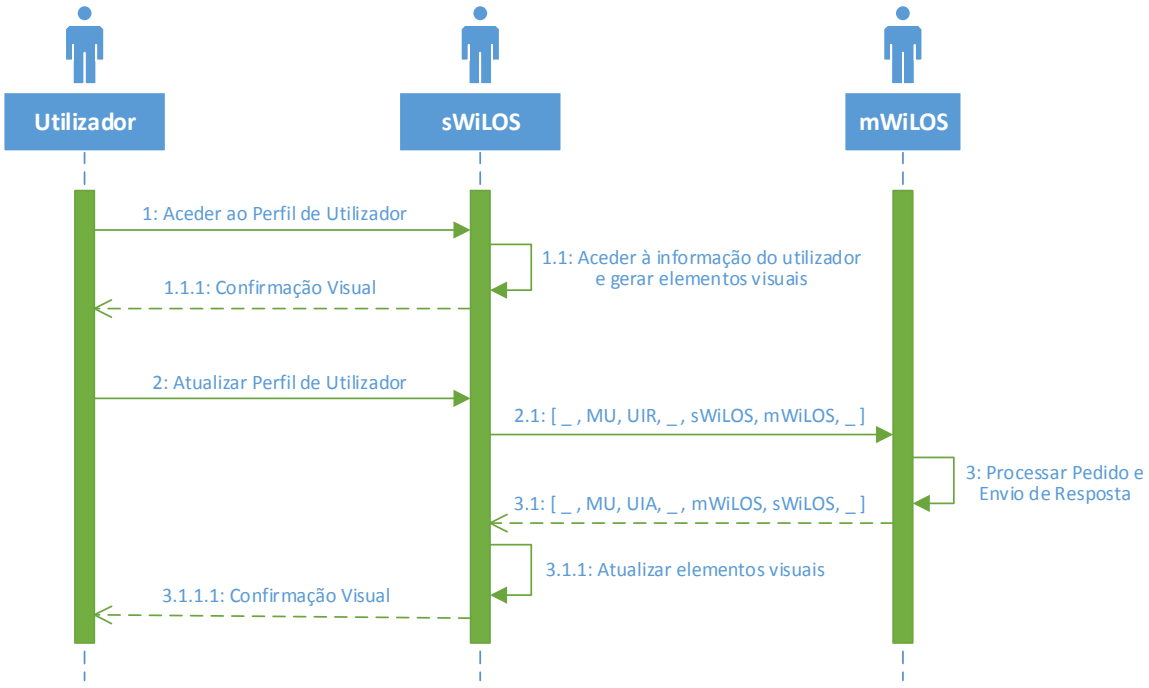


Figura 4.45 - Atualização do perfil de um utilizador através das interações entre utilizador, sWiLOS (UI, UM e MSI) e mWiLOS.

O exemplo da figura 4.45 permite ilustrar o processo de pedido de informação ao mWiLOS. Como citado em 4.1.3.2, um utilizador consegue visualizar o seu perfil, registado no ISM, através da classe "_U_UserProfile". No entanto, por algum motivo, este pode não conter a informação mais atual. A

atualização dos dados do utilizador podem ser efetuados através de um pedido de atualização na UI. Esta recorre ao UM, e subsequentemente à MSI, para comunicar com o mWiLOS de modo a se obter a informação desejada, como descrito no diagrama de sequência. As linhas de vida dos atores estão sempre ativas, uma vez que no sWiLOS e no mWiLOS a MSI encontra-se sempre disponível para a receção e envio de pedidos. O utilizador desencadeia a ação de pedido de informação através da interação com o sWiLOS, daí a sua atividade ao longo de todo o processo.

Interface Mestre-Escravo

A classe "_Client", responsável por implementar a MSI, possui mecanismos que tornam o sWiLOS num cliente TCP/IP do mWiLOS. O cumprimento do protocolo de comunicação descrito em 4.1.4.2 implica, não só uma sequência ordenada de pedidos e respostas, como também a verificação da autenticidade de cada uma das partes durante o processamento de uma mensagem. O diagrama de sequência da figura 4.46 permite evidenciar esse facto. Esta ilustra o processo de estabelecimento de uma ligação com o mWiLOS, admitindo que o dispositivo já fazia parte do sistema WiLOS.

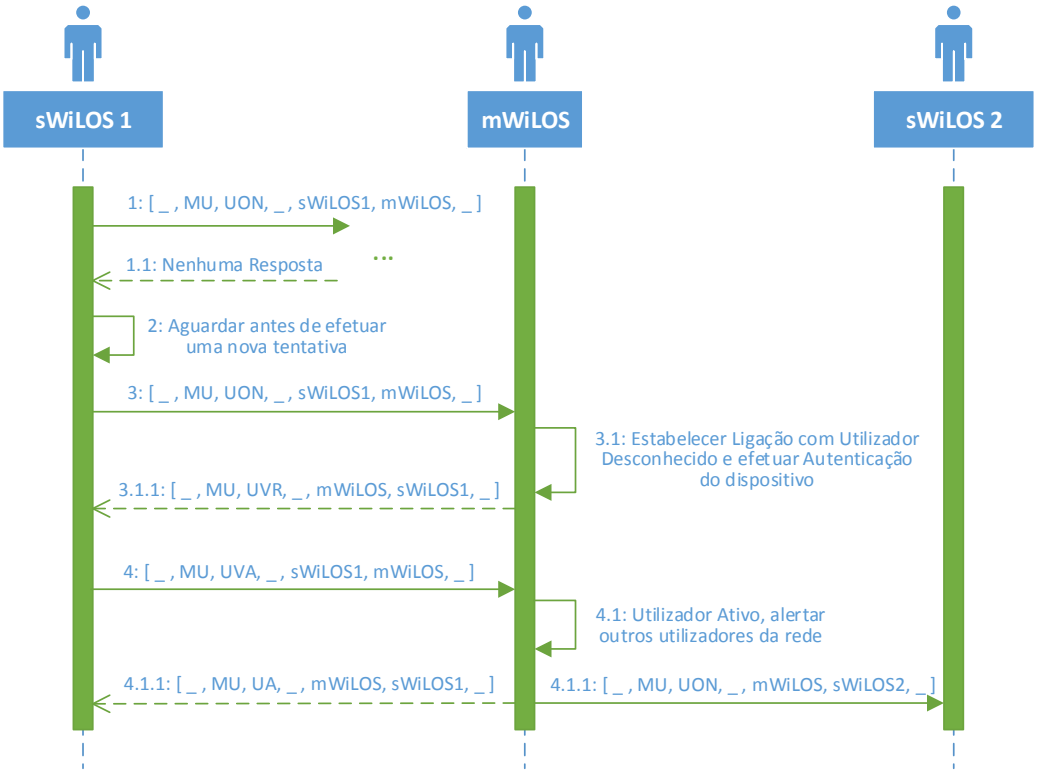


Figura 4.46 - Processo de estabelecimento de uma ligação do sWiLOS com o mWiLOS.

Inicialmente o dispositivo móvel pode não se encontrar no alcance do servidor do mWiLOS, o que implica repetir periodicamente o pedido de estabelecimento de ligação até o mWiLOS responder positivamente. O mWiLOS só responderá ao pedido se os campos "Código de Acesso" e "ID de Utilizador" forem coincidentes com os valores registados no ISM. Mesmo assim, é requisitada mais informação ao sWiLOS para garantir a sua autenticidade. Do mesmo modo, se o sWiLOS verificar alguma irregularidade na resposta inicial do mWiLOS, como código ou o nome de servidor, a comunicação cessa imediatamente. Deste modo evita-se o envio de informações sigilosas a sistemas desconhecidos.

O processo de autenticação do sWiLOS começa com a resposta à mensagem "UM-UVR" do mWiLOS com uma mensagem "UM-UVA". Nesta envia-se a informação referente ao nome de utilizador e o endereço MAC do dispositivo móvel para garantir acesso ao mWiLOS. Se o utilizador for válido, o mWiLOS procede à ativação do utilizador na rede, avisando os outros utilizadores, e responde ao sWiLOS com um novo código de acesso, a ser utilizado nas comunicações futuras (mensagem "UM-UA").

Todos estes mecanismos de tentativa de estabelecimento de ligação, espera de pedidos e respostas do mWiLOS, verificação da autenticidade do servidor, análise do formato da mensagem e respetivo processamento do seu conteúdo, obedece ao diagrama de atividades representado na figura 4.47. A primeira condição verificada é se o sWiLOS possui uma ligação TCP/IP válida com um servidor. Só através desta é que se pode determinar se o servidor é quem afirma ser, de modo a se iniciar o processo de autenticação supracitado. Se não existir nenhuma ligação, então a MSI do sWILOS tenta periodicamente estabelecer uma ligação sempre que se encontra dentro do alcance da rede WiLOS.

A fase de espera de pedidos ou respostas vindas do mWiLOS é um mecanismo reativo que, através da escuta do *socket*, verifica se alguma informação foi recebida através da ligação. Caso tal se verifique, a receção de uma mensagem, para além de ser precedida pelo tamanho da própria, implica a sua decomposição nos 8 campos definidos no protocolo. Esta decomposição permite validar o formato da mensagem enviada e facilita o processo de análise da informação recebida. Desta forma, as ações "Analisar Categoria Primária da Mensagem" e "Analisar Categoria Secundária da Mensagem" reencaminham o mecanismo de processamento para a área onde se encontram implementadas as operações a realizar para cada tipo de mensagem recebida. Por norma, cada uma das áreas valida o servidor através da verificação do seu nome e código de acesso durante a comunicação.

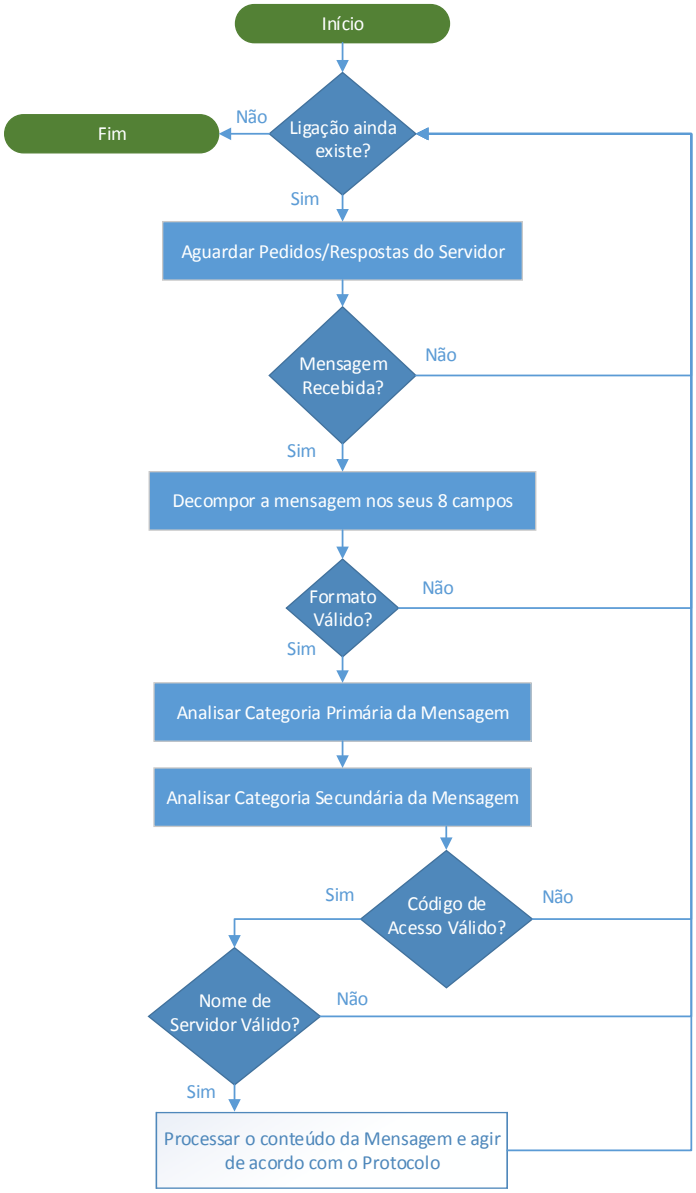


Figura 4.47 - Diagrama de atividade do funcionamento interno da MSI do sWiLOS.

Módulo "Monitoração" e MSI

O MM não efetua propriamente a monitoração do dispositivo móvel, mas implementa funcionalidades que permitem o emparelhamento do sWiLOS com o mWiLOS e a recolha dos valores de *RSSI* verificados pelo dispositivo. O processo de emparelhamento implica grande interatividade entre o utilizador, sWiLOS e mWiLOS, como demonstrado no diagrama da figura 4.48.

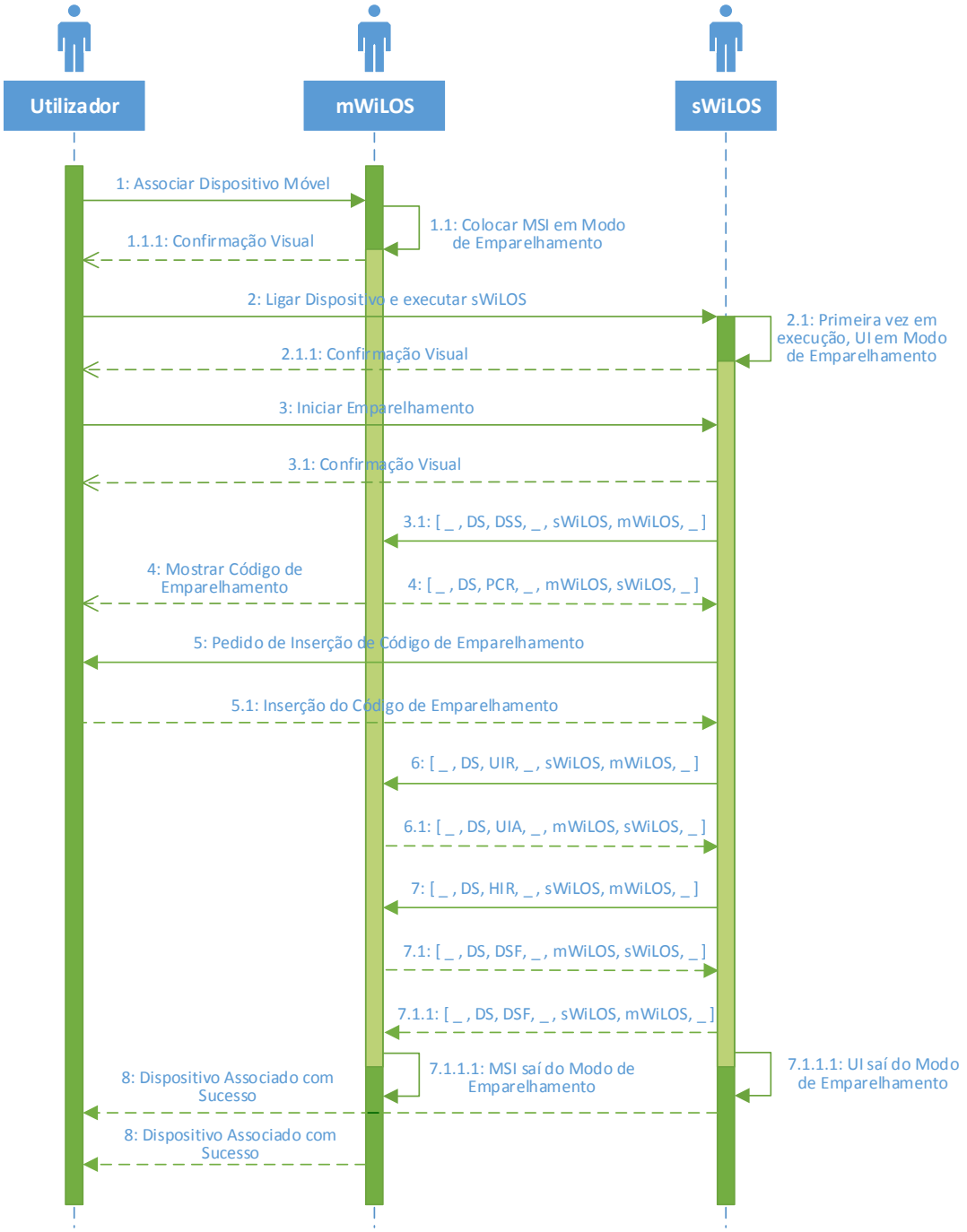


Figura 4.48 - Diagrama de sequência referente ao processo de emparelhamento de um dispositivo móvel com o mWiLOS.

O processo de emparelhamento consiste numa troca de informação mediada entre o utilizador, o sWiLOS e o mWiLOS. Durante o processo de criação de um novo utilizador no mWiLOS, o utilizador pode optar por associar um dispositivo, ou efetuar essa operação mais tarde. Se optar por associar um dispositivo, o mWiLOS entra num estado de espera, de modo a receber um pedido de emparelhamento do sWiLOS. A partir deste ponto, todos os intervenientes obedecem à sequência demonstrada, fornecendo tanto ao sWiLOS como ao mWiLOS a informação necessária para ambos se identificarem mais tarde na rede WiLOS. A interface gráfica apresentada pelo sWiLOS ao utilizador durante este

processo encontra-se ilustrada na figura G.4 do anexo G: "Interface Gráfica - sWiLOS", referente ao *namespace* "_Monitor" e as suas classes. A informação trocada entre ambos os subsistemas é descrita no protocolo de comunicação apresentado subcapítulo 4.1.4.2, durante a descrição da implementação da MSI.

A monitoração do sWiLOS exige que este responda periodicamente aos pedidos de localização efetuados pelo mWiLOS. A localização do sWiLOS é determinada a partir dos valores de *RSSI* recolhidos pelo dispositivo, referentes aos *APs* da rede WiLOS detetados no instante de receção do pedido. A recolha destes valores recorre à biblioteca *Smart Device Framework*, que permite o acesso aos vários módulos de comunicação com rede existentes no dispositivo, incluído o módulo Wi-Fi. Para além disso, a biblioteca disponibiliza funcionalidades para verificar quais os *APs* na vizinhança, recolher os seus dados e até mesmo tentar estabelecer uma ligação com um dos *APs*.

Os dados dos *APs* utilizados durante a fase de monitoração são o nome da rede sem fios ou o *SSID* (*Service Set IDentifier*), o endereço MAC e o valor de *RSSI* associado. O primeiro permite identificar quais os *APs* que pertencem à rede WiLOS. O segundo identifica inequivocamente o *AP* dentro da rede. Por fim, o valor de *RSSI* contém a informação relativa à potência do sinal utilizada pelo mWiLOS para estimar a localização do utilizador. Este processo efetua-se de acordo com o diagrama de atividades da figura 4.49.

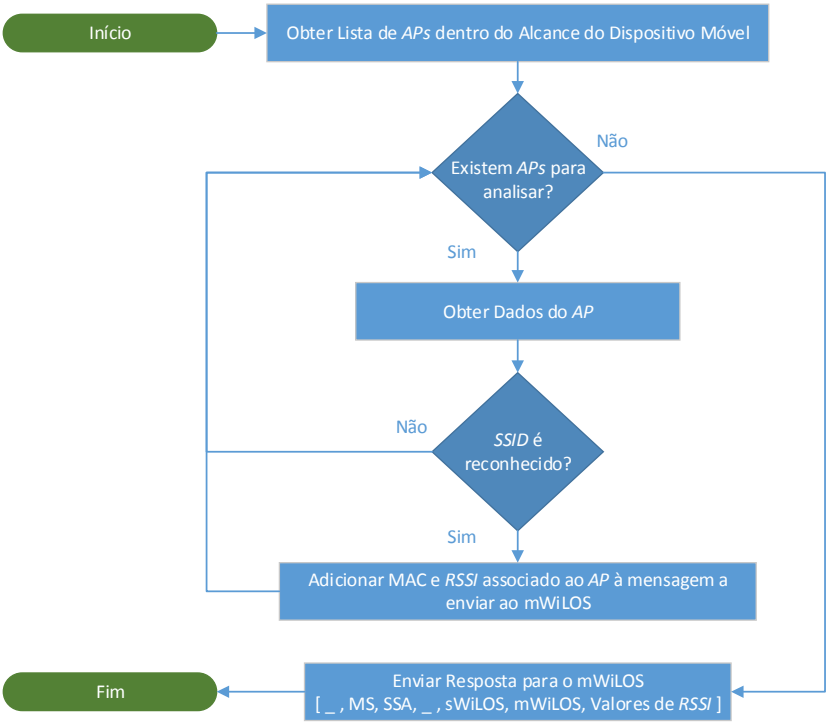


Figura 4.49 - Diagrama de atividade relativo à resposta ao pedido de localização efetuado pelo mWiLOS (MS-SSR).

Módulo "Mapa de Potência" e MSI

Para se efetuar o processo de calibração, o sWiLOS recebe do mWiLOS o mapa de calibração, os dados dos *APs* da rede WiLOS previamente detetados e a informação relativa aos pontos já calibrados. O utilizador recorre ao mapa de calibração para se situar sobre os pontos por calibrar e realizar a sua aquisição. O diagrama da figura 4.49 detalha o processo de calibração de um ponto, introduzido no subcapítulo 4.1.4.2 através da pormenorização da implementação do SSMM.

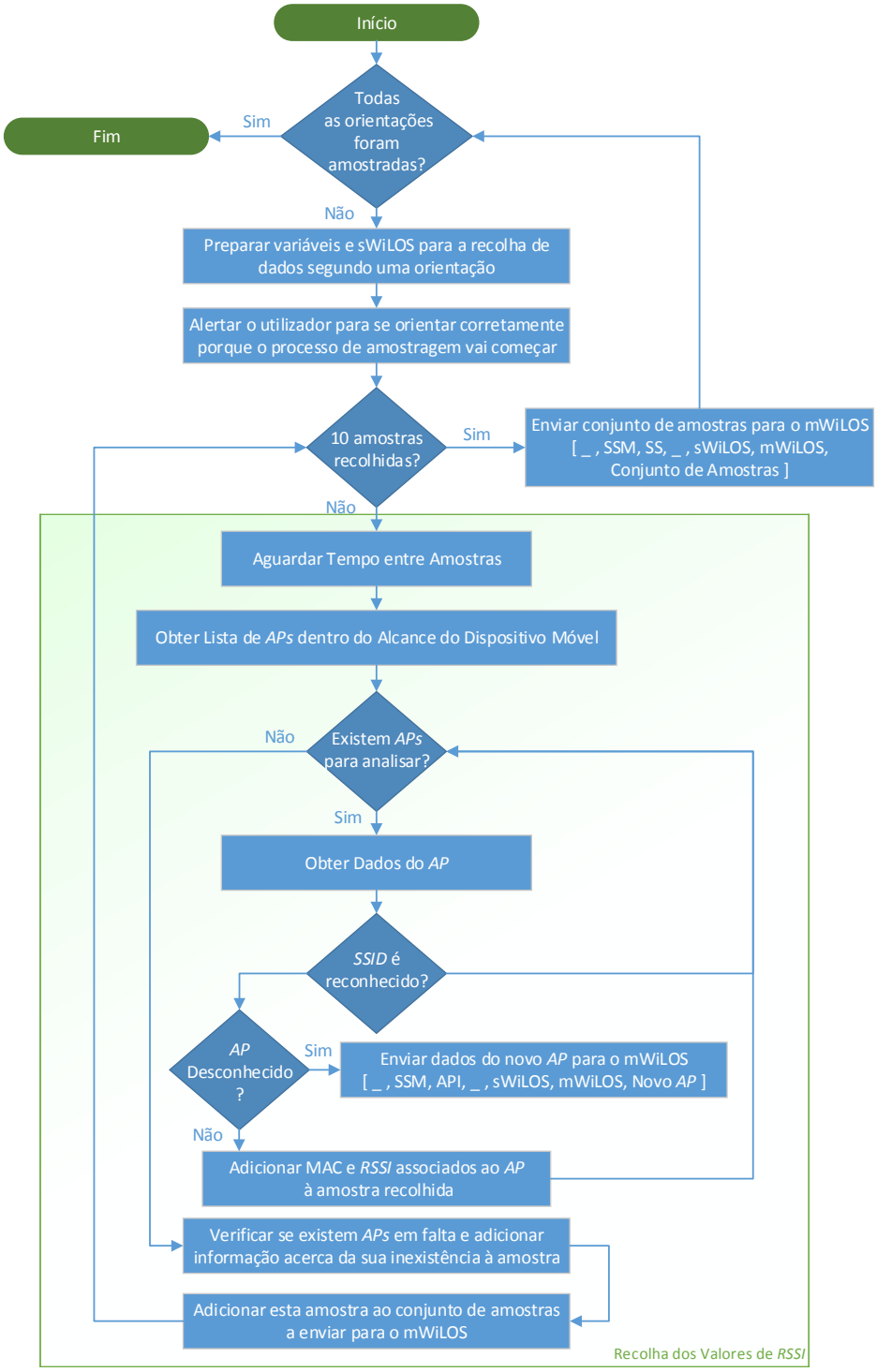


Figura 4.50 - Diagrama de atividade referente ao processo de calibração de um ponto do mapa de calibração da habitação.

Este processo consiste na recolha de um conjunto de amostras que permita mais tarde identificar inequivocamente o ponto através de algoritmos de comparação/vizinhança, como o algoritmo k-*NNS*. Um conjunto de amostras é composto por 10 valores de *RSSI*, de modo a representar uma das direções e orientações de calibração definidas no processo de modelação. Assim, as combinações de direções Norte, Sul, Este e Oeste com as orientações Vertical e Horizontal, obrigam a que o processo de calibração de um ponto ocorra 8 vezes, de forma a se obterem 80 valores de *RSSI* para cada ponto. Em vez de se enviarem todos os valores que constituem o ponto numa única mensagem, efetua-se a sua repartição por 8 mensagens. Cada uma destas contém apenas um dos 8 conjuntos de amostras obtidos (mensagem "SSM-SS"). Deste modo reduz-se a quantidade de informação enviada somente por mensagem, poupando os recursos da MSI do sWiLOS.

De forma semelhante ao processo de aquisição de valores de *RSSI* implementado pelo MM, efetua-se um ciclo que analisa todos os *APs* dentro do alcance do dispositivo móvel. No entanto, durante o processo de calibração, o sWiLOS apenas conhece o número de *APs* que constituem a rede WiLOS e os dados dos *APs* previamente detetados. Isto significa que, durante a aquisição de um ponto de calibração, é possível a identificação de novos *APs* pertencentes à rede WiLOS, mas que ainda não fazem parte do ISM do WiLOS. Se tal fenómeno ocorrer, o sWiLOS envia para o mWiLOS a mensagem "SSM-API" com os dados do novo *AP*, de modo a armazenar a sua informação para operações futuras.

Módulo "Serviço de Mensagens" e MSI

A utilização do serviço de mensagens do sWiLOS implica a criação de conversações entre utilizadores, troca de mensagens e consulta de histórico de conversações. Apesar de o ISM e a MSI estarem preparados para realizar uma conversação com mais do que 2 utilizadores, o CSM e a UI só implementam funcionalidades para uma conversação a 2 pessoas. Ainda assim, a criação de uma nova conversação necessita de obedecer ao protocolo de comunicação e à sequência de mensagens definida pelo diagrama da figura 4.51, de modo a garantir a troca de mensagens entre os 2 utilizadores associados à conversação.

O diagrama de sequência apresentado na figura 4.52 refere-se ao processo de envio de uma mensagem, admitindo a existência de uma conversação. A consulta do histórico de conversações de um utilizador consiste na execução de pedidos de informação simples ao mWiLOS, que seguem o mesmo mecanismo do diagrama da figura 4.45, apresentado no início deste subcapítulo.

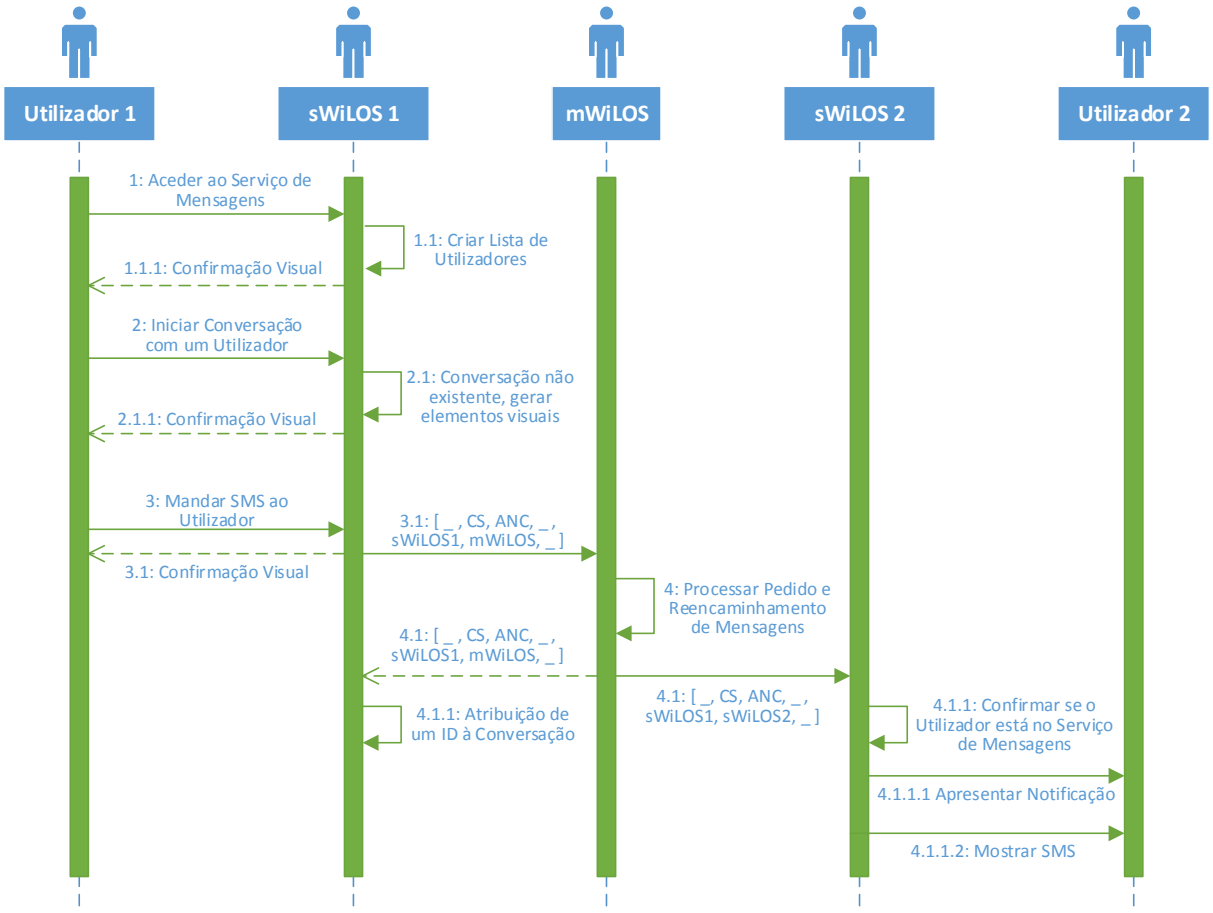


Figura 4.51 - Diagrama de sequência relativo ao processo de criação de uma nova conversação entre 2 utilizadores WiLOS.

O processo de criação de uma conversação inicia-se no CSM do sWiLOS. Quando um utilizador abre uma janela de conversação com outro utilizador, o CSM possui mecanismos que verificam se o ISM contém alguma conversação associada a estes utilizadores. Se existir uma conversação ativa, as mensagens trocadas são automaticamente apresentadas na janela de conversação, e recorre-se ao diagrama da figura 4.52 para se realizar a troca de mensagens. Caso contrário, o CSM regista que uma nova conversação está para ocorrer, de modo a enviar pela MSI um eventual pedido de nova conversação (mensagem "CS-ANC"). Este só chega a ser enviado se, de facto, o utilizador tentar estabelecer comunicação com o outro utilizador, isto é, através do envio de uma mensagem.

O pedido de uma nova conversação possui a informação referente à nova conversação, incluindo os utilizadores nela registados e a primeira mensagem enviada. Por norma, o mWiLOS acede ao pedido de conversação imediatamente e devolve ao remetente somente o identificador atribuído à conversação. No outro lado, o utilizador destinatário recebe toda a informação referente à conversação (identificador inclusive). O CSM deste fica responsável por apresentar a mensagem ao utilizador, ou simplesmente notificá-lo da receção de uma nova mensagem.

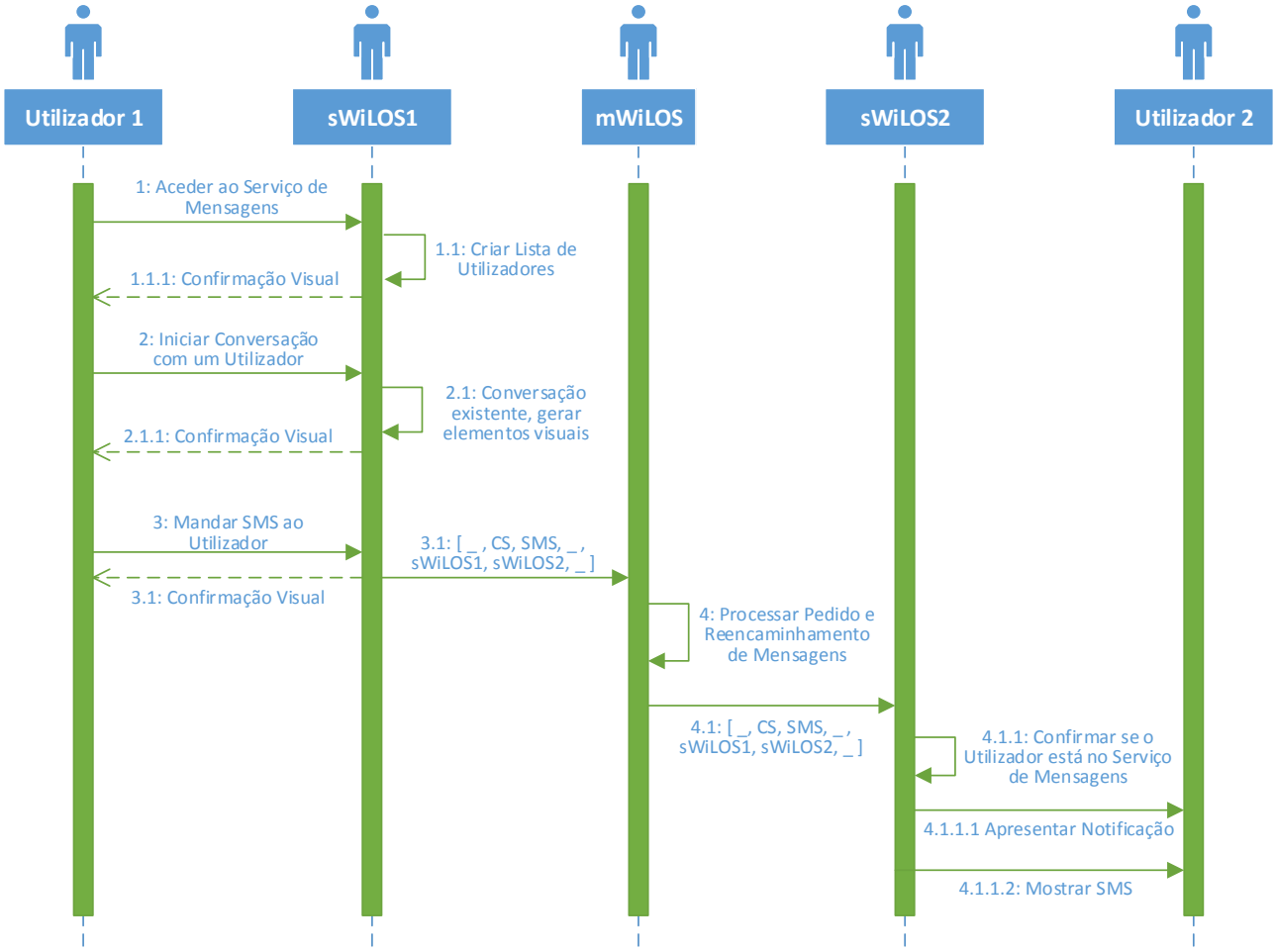


Figura 4.52 - Diagrama de sequência relativo ao processo de envio de uma mensagem a um utilizador.

A troca de mensagens entre dispositivos sWiLOS obedece a um encadeamento de mensagens muito semelhante ao do processo de criação de uma nova conversação. A única diferença reside no tipo de mensagem MSI utilizada durante a comunicação sWiLOS e mWiLOS. Desta vez utiliza-se uma mensagem do tipo "CS-SMS", que contém a informação do remetente, conversação de destino e conteúdo da mensagem enviada. O mWiLOS apenas associa a mensagem recebida à respetiva conversação registada no ISM, e reencaminha-a para todos os utilizadores associados à conversação.

4.2 HUEPS

O HUEPS é um sistema de monitoração energética que recorre a serviços de localização para efetuar uma distribuição justa do consumo energético de uma habitação pelos seus ocupantes. O objetivo é verificar até que ponto é possível utilizar o modelo de ocupação dos habitantes e o modelo de consumo energético da habitação para determinar os perfis energéticos de cada utilizador e das divisões que compõem a habitação.

O serviço de localização utilizado pelo HUEPS será fornecido pelo WiLOS. Desta forma, os modelos de ocupação/localizações atuais de cada utilizador são fornecidos ao HUEPS em tempo real. Uma vez que o WiLOS já possui muita informação referente à habitação e aos seus utilizadores, propõe-se que o HUEPS seja um consumidor de informação, necessitando apenas de sincronizar os dados já existentes com o WiLOS. A redução de mecanismos de inserção de nova informação por parte do utilizador acaba por simplificar o sistema. Isto permite ao HUEPS dedicar-se à interação com o aparelho de medição do consumo energético da habitação e à correlação do consumo obtido com o posicionamento dos utilizadores.

4.2.1 Modelo Concetual – HUEPS

4.2.1.1 Modelo Funcional

O HUEPS, como mencionado, é maioritariamente um sistema de geração automática e consulta de informação. Sendo assim, não possui muitos mecanismos que permitam ao utilizador introduzir nova informação diretamente. Para esse efeito recorre-se ao WiLOS, através do qual é efetuada a sincronização de informação com o HUEPS. Esta informação engloba a tipologia da habitação (planta da habitação com as suas divisões distintas), bem como outras informações complementares (exemplos: morada, contacto e descrição da habitação). Para além disso, só os utilizadores registados no WiLOS podem usufruir das funcionalidades do HUEPS, o que reforça o recurso a sincronismo de informação em vez da introdução de elementos diretamente no HUEPS. Alguns exemplos de informação relativa ao utilizador partilhados são o seu identificador WiLOS, nome, sexo, idade e contato.

No WiLOS os ocupantes da habitação criam um perfil de utilizador de modo a definirem o seu nome de utilizador, nível de acesso e palavra-passe. Isto garante a privacidade dos utilizadores e a segurança do sistema através da restrição da informação visível e das funcionalidades de configuração disponibilizadas a um determinado tipo de utilizador. Este perfil de utilizador é replicado no HUEPS para que o utilizador continua a possuir os mesmos privilégios e acesso à informação disponibilizada.

Uma vez que o HUEPS usufrui de informação disponibilizada por outros sistemas, neste caso o WiLOS, deseja-se também que o HUEPS partilhe alguma da sua informação para o mundo exterior. Para salvaguardar a privacidade dos dados fornecidos, o HUEPS tem que possuir mecanismos que permitam a gestão de sistemas externos e quais os dados que estes podem consultar, à semelhança do WiLOS.

Como o objetivo principal do HUEPS é a atribuição de consumos aos utilizadores do WiLOS, este necessita de dois elementos essenciais: a localização de cada utilizador dentro da habitação e o

consumo energético global da habitação. O posicionamento de cada um dos ocupantes é fornecido em tempo real pelo WiLOS. Caso seja necessário, o HUEPS pode requerer ao WiLOS informação adicional que permita reduzir ambiguidades durante o processo de atribuição de consumo. O consumo energético global da habitação é obtido através da interação com um medidor "inteligente" (*Smart Meter*), que possibilita a recolha e consulta da variação do consumo energético através de um computador. O medidor a utilizar para garantir estas funcionalidades é a *Energy Box* (*EB*) da EDP. A referência deste equipamento na fase de modelação é importante uma vez que é a partir deste que se definem as operações possíveis de executar sobre a consumo energético recolhido.

A *EB* disponibiliza diversos dados (corrente, potência, tarifários, etc.) que podem ser organizados de modo a construir gráficos ilustrativos da evolução do consumo de uma habitação ao longo do tempo. Esta ferramenta por si só é bastante útil pois permite aos utilizadores a consciencialização dos gastos efetuados ao longo do dia, semana, mês e ano. Também permite detetar períodos do dia em que o consumo é maior do que o esperado, obrigando os ocupantes a mudar o seu comportamento ou a verificar a causa desse excesso. Como acréscimo, a determinação dos perfis de cada utilizador e divisão possibilitam um maior detalhe da distribuição do consumo energético da habitação. Estes podem ser representados através de diversos elementos estatísticos, como um gráfico de distribuição energética de um utilizador por divisão, ou qual foi o utilizador com maior consumo energético numa determinada semana, desde que a sua interpretação ocorra de forma natural e intuitiva.

A comunicação entre a *EB* e o computador obedece a um protococo de comunicação série, que possibilita a escrita e/ou leitura dos registos de memória da *EB*. Este tipo de interação é típico dos *Smart Meters*, daí o HUEPS possuir mecanismos a definição das características do protocolo série utilizado durante a comunicação. Este também permite a gestão dos registos do *Smart Meter* que podem ser acedidos pelo computador anfitrião. Outra informação relevante para o HUEPS é o tempo de amostragem utilizado para se obterem os valores referentes ao consumo energético da habitação. Sendo assim, a configuração deste valor é uma funcionalidade obrigatória no HUEPS.

Como os dados relativos ao tarifário ou ao fornecedor de energia disponibilizados pela *EB* podem não ser suficientes para a determinação dos custos energéticos, o HUEPS possibilita a introdução, consulta e remoção deste tipo de informação. Desta forma, cabe ao utilizador verificar se a informação relativa ao tarifário está atualizada ou, caso contrário, definir os períodos do dia onde o valor monetário do consumo por hora é maior ou menor. Isto possibilita ao HUEPS apresentar aos utilizadores uma estimativa de quanto é que foi gasto ao longo do mês. Como incentivo à consciencialização ecológica dos ocupantes, o HUEPS possibilita a configuração da taxa de emissão de CO2 por cada W de potência consumida. Esta métrica permite mostrar aos utilizadores as emissões de CO2 relacionadas com o consumo energético verificado na habitação.

As várias funcionalidades disponibilizadas pelo HUEPS podem ser consultadas nos diagramas de casos de uso representados nas figuras 4.53, 4.54 e no anexo C.

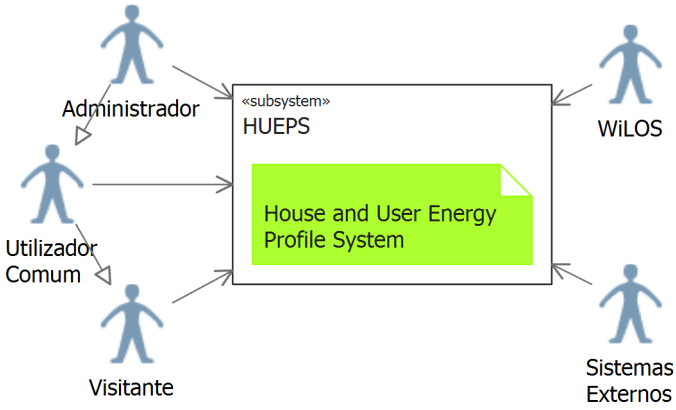


Figura 4.53 - Diagrama de casos de uso do HUEPS – Atores que interagem com o HUEPS.

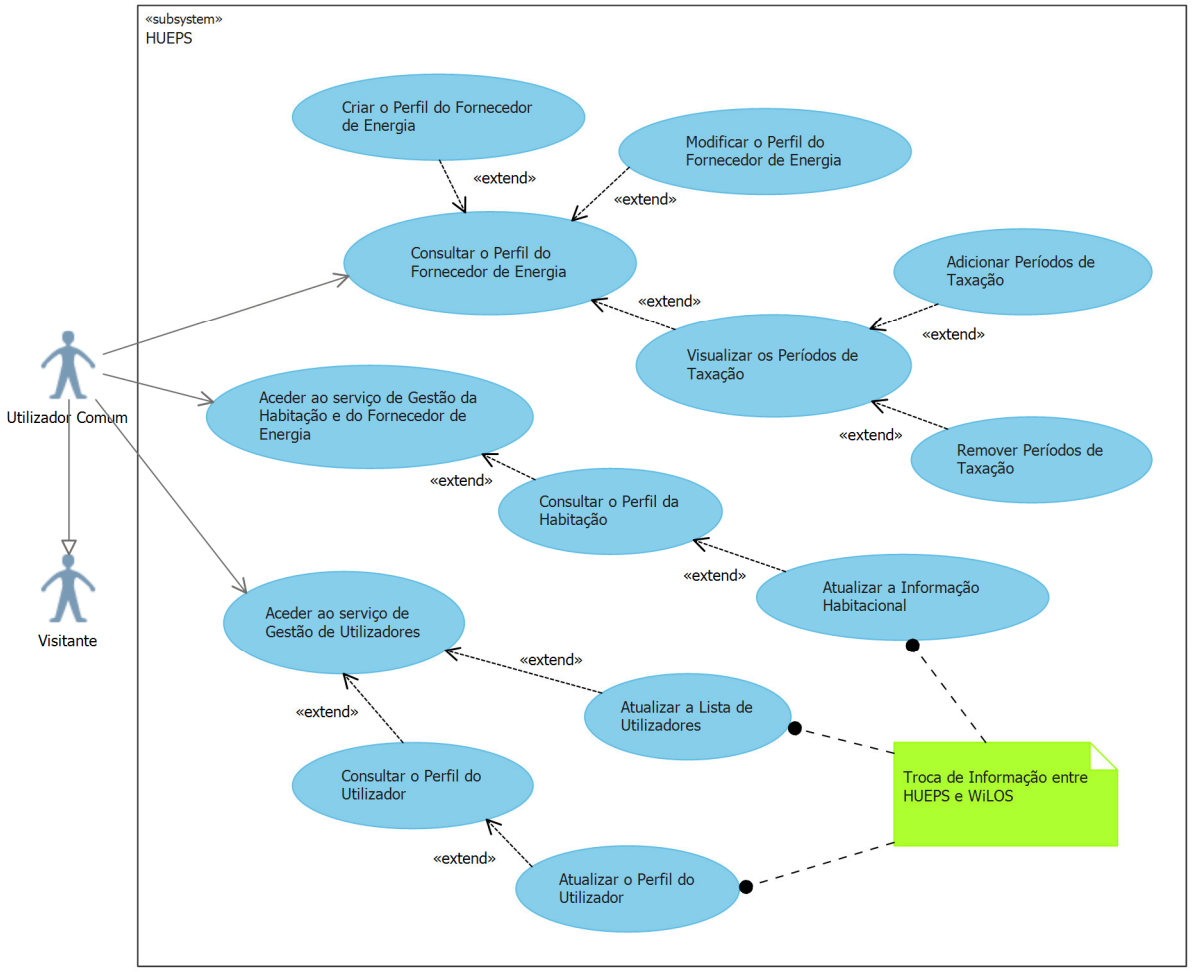


Figura 4.54 - Diagrama de casos de uso do HUEPS – Funcionalidades disponibilizadas ao Utilizador Comum.

4.2.1.2 Modelo Arquitetural

O modelo arquitetural do HUEPS é composto pelas camadas "Interface", "Controlo" e "Entidade", como ilustrado na figura 4.55.

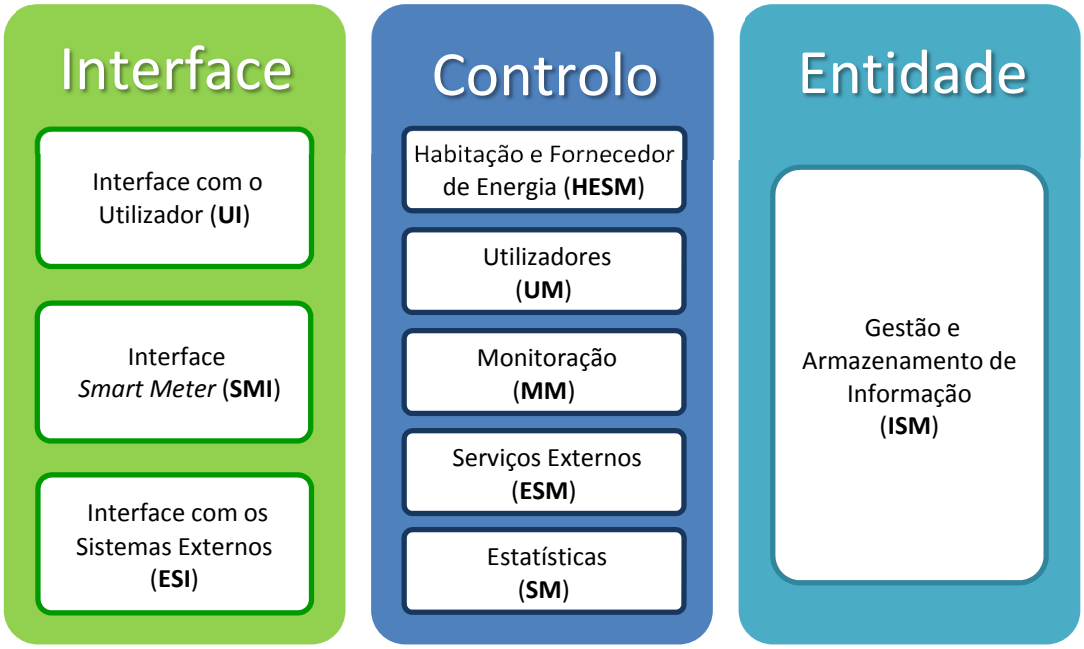


Figura 4.55 - Modelo arquitetural do HUEPS. Estruturação em 3 camadas: Interface; Controlo e Entidade.

Interface

Toda a informação adquirida pelo HUEPS (de forma interna ou externa) pode ser consultada pelos seus utilizadores. Estes também podem manipular alguma da informação, através da introdução e remoção de dados, de modo a garantir a integridade e fidedignidade do HUEPS. Estes exemplos de interação direta com o utilizador são facultados pelo módulo "Interface com o Utilizador" (***U**ser **I**nterface*). As interações indiretas dos utilizadores com o HUEPS ocorrem através da utilização dos aparelhos elétricos da habitação, o que origina flutuações no consumo. Só através destas, e do posicionamento dos utilizadores, é que se torna possível a existência deste sistema.

A comunicação entre o HUEPS e a *EB* é realizada através do módulo "Interface *Smart Meter*" (***S**mart **M**eter **I**nterface*). Esta interface permite o acesso aos registos da *EB,* de modo a extrair-se a informação referente ao consumo da habitação. É também possível aceder a outros dados por ela disponibilizados, tais como o contrato efetuado com o fornecedor de energia ou o tarifário em vigor. A camada Entidade é evocada para se obter a informação relativa aos registos da *EB* que são necessários para o funcionamento do HUEPS. Para além disso, é necessária a existência de uma ligação "física" que interligue a *EB* ao computador onde o HUEPS se encontra instalado.

O módulo "Interface com os Sistemas Externos" (***E**xternal **S**ystems **I**nterface*) permite ao HUEPS trocar informação com outros sistemas que usufruam da rede WiLOS. É através desta interface que ocorre o sincronismo de informação com o WiLOS, bem como a receção periódica do posicionamento dos seus utilizadores. Toda a informação recebida é encaminhada para a camada Entidade que fica responsável pelo seu armazenamento e organização.

Controlo

Cada módulo da camada Controlo é desenhado de forma a realizar uma funcionalidade específica ou lidar com dados que podem ser agrupados de acordo com a sua natureza. Deste modo, o módulo "Habitação e Fornecedor de Energia" (***H**ouse and **E**nergy **S**upplier **M**odule*) efetua a interligação entre a UI e a camada Entidade, permitindo aos utilizadores consultarem os dados referentes à habitação e à sua tipologia. Também é possível gerir os dados do fornecedor de energia e do tarifário a aplicar à habitação, de modo a se obterem estimativas do custo monetário referente ao consumo verificado.

A consulta e visualização da informação relativa aos utilizadores do HUEPS são realizadas através do módulo "Utilizadores" (***U**sers **M**odule*). Os mecanismos para a modificação dos dados de um utilizador ficam entregues ao WiLOS que, como supracitado, fica encarregue de sincronizar a informação com o HUEPS (ou vice-versa).

O algoritmo que correlaciona o posicionamento dos utilizadores com o consumo global da habitação é executado pelo módulo "Monitoração" (***M**onitor **M**odule*). O processo de monitoração ocorre automaticamente e em tempo real, caso o serviço de monitoração esteja ativo. À medida que um utilizador, ou utilizadores, se deslocam dentro da habitação, as suas ações desencadeiam variações no consumo energético. Estas variações podem ser associadas às divisões onde ocorrem, de modo a estabelecer um perfil energético por divisão. Dependendo do número de utilizadores dentro da divisão, torna-se possível atribuir uma parcela desse consumo energético a cada utilizador. Estas parcelas de consumo energético permitem definir o perfil energético de um utilizador. Para determinar qual a parcela do consumo energético da divisão a atribuir a cada utilizador, o HUEPS baseia-se em 11 premissas.

A figura 4.56 ilustra diversas situações que permitem extrair essas premissas de um modo mais intuitivo.

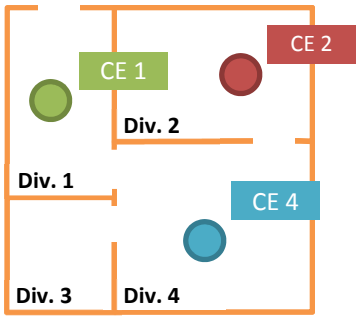
Legenda

- - - - Consumo Energético por Divisão (CE)

- - - - Posição ou CE Anterior

- Utilizador

- Deslocação

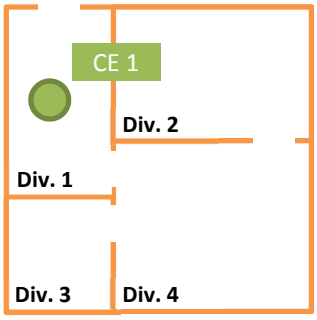


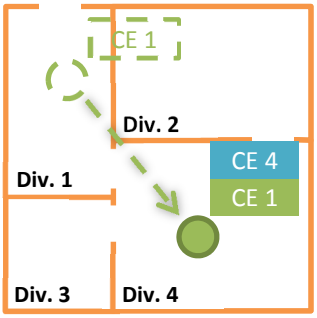
Premissa 1

Cada utilizador da habitação deve ser capaz de originar eventos que afetem o consumo energético global.

Premissa 2

Se apenas existir um utilizador na habitação, o consumo energético total é da sua responsabilidade.





Premissa 3

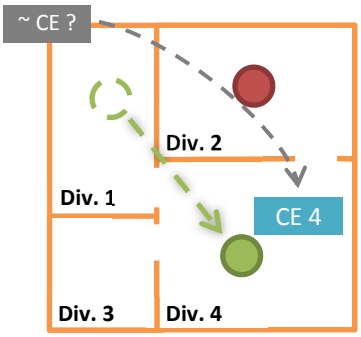
Os consumos energéticos por divisão mantêm a sua individualidade.

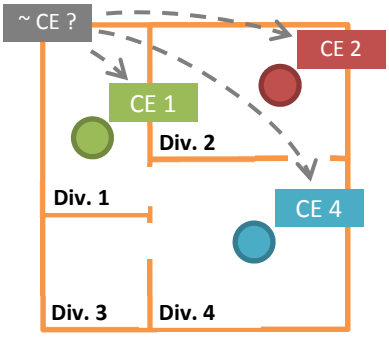
Premissa 4

Se um utilizador abandonou a divisão e esta se encontrar vazia, o consumo referente a essa divisão continua associado ao utilizador.

Premissa 5

Se ocorrer movimentação de utilizadores entre divisões durante uma variação de consumo energético (~ CE), essa variação é associada, preferencialmente, a esses utilizadores.



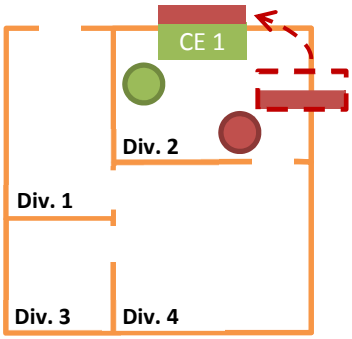


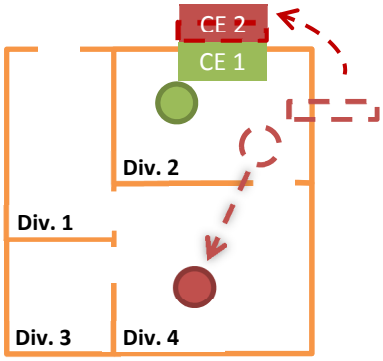
Premissa 6

Se ocorrer uma variação de consumo e não se verificar o deslocamento de nenhum utilizador, esse consumo é repartido por todos os utilizadores.

Premissa 7

Caso se verifique que o consumo energético da divisão é comum aos utilizadores, repartir apenas o consumo dessa divisão e não o consumo total associado a cada utilizador.



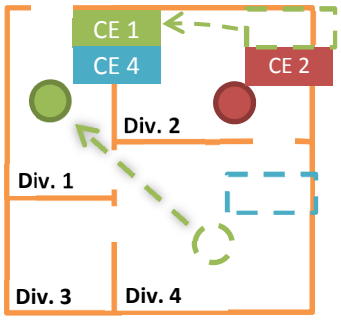


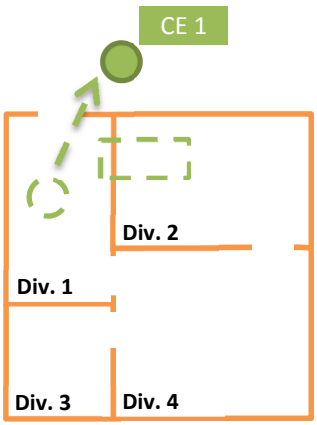
Premissa 8

Se um utilizador abandonar uma divisão e esta ainda possuir ocupantes, o consumo da divisão é alocado somente aos utilizadores presentes na divisão.

Premissa 9

Caso um utilizador entre numa divisão com consumo energético associado a outro utilizador, e esse utilizador não se encontre na divisão, então o consumo dessa divisão passa a ser da responsabilidade do novo utilizador.





Premissa 10

Se existir algum consumo energético remanescente na habitação quando todos os utilizadores a abandonam, esse consumo é da responsabilidade do último ocupante a sair da habitação.

Premissa 11

Se existir consumo energético na habitação não atribuído durante a entrada de utilizadores na habitação, o consumo é repartido pelos utilizadores.

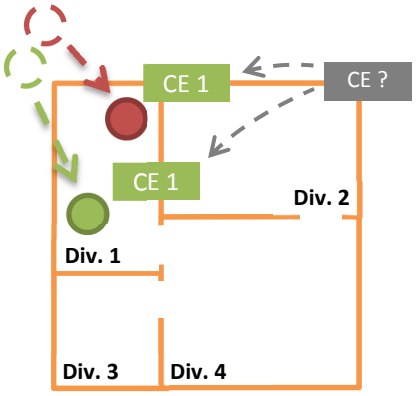


Figura 4.56 - Premissas que definem a atribuição e distribuição de consumo energético no HUEPS.

Algumas premissas são controversas na medida em que os utilizadores podem sentir-se injustiçados. Afinal, o consumo energético corrente pode não ter origem coletiva (luz, televisão, etc.) e ser apenas má gestão de um determinado aparelho elétrico por um dos ocupantes. Como não será implementado nenhum método para determinar o tipo de aparelho utilizado, que permitiria fazer esta distinção entre o coletivo e o individual, estas premissas permitem fazer uma distribuição de consumo simplista.

Outra falha resultante da aplicação destas premissas verifica-se ao nível da atribuição do consumo por divisão. Tenha-se como exemplo o ato de colocar uma máquina de lavar roupa a trabalhar. Se a variação ocorre na divisão 1, então o consumo referente à máquina de lavar fica associado a essa divisão e ao utilizador (ou utilizadores) que nela se encontram. No entanto, uma pessoa não fica à espera que a máquina acabe a lavagem e, provavelmente, desloca-se para outra divisão. Se o consumo da máquina de lavar fosse constante ao longo do tempo não haveria problema. Porém, sabe-se que o consumo desta varia ao longo do tempo, consoante a fase de lavagem atual. Mais uma vez, como o HUEPS não reconhece aparelhos e não tem noção da existência deste tipo de comportamentos, vai assumir que as variações de consumo são originadas pelos utilizadores nas diversas divisões. Desta forma, o consumo energético da divisão 1 fica incorretamente distribuído pelas restantes divisões, pelo menos durante o período de funcionamento da máquina de lavar roupa. Outras situações

semelhantes são passíveis de ocorrer, mas o erro é aceitável tendo em conta os objetivos deste projeto.

Para além disso, existe a problemática do consumo residual da habitação, isto é, o consumo energético originado por eletrodomésticos como o frigorífico. Estes aparelhos encontram-se em funcionamento constante e costumam possuir diversos modos de funcionamento, originando um consumo energético residual não constante ao longo do tempo. Como o objetivo deste trabalho não engloba a deteção de cargas, a simplificação deste problema passa por tratar o consumo residual como um consumo normal, causado pela interação dos utilizadores com a habitação.

Por outro lado, estas premissas levam ao desenvolvimento da consciencialização ecológica. Uma vez que as pessoas não querem ser responsáveis por um consumo que não lhes pertence, estas encarregam-se de desligar o aparelho em questão (mudança de comportamento). Outra possibilidade é a de alertarem o individuo responsável pelo aparelho, de modo a influenciar o comportamento desse utilizador.

O processo oposto à atribuição de consumos aos utilizadores é a dissociação de consumos energéticos. Esta é efetuada sempre que se verifique uma variação negativa no consumo energético global da habitação. O principal objetivo é encontrar o consumo energético atribuído a um utilizador, ou utilizadores, que mais se aproxime da variação energética registada. Uma vez que a premissa 6, e todas as premissas a ela similares, possuem maior probabilidade de introduzir erro na fase de distribuição de consumo, existe uma preferência inicial para se dissociarem as distribuições originadas por estas premissas. Desta forma, espera-se reduzir o erro introduzido por elas a longo prazo, não permitindo a sua existência durante um período de tempo relativamente extenso.

Para se verificar a distribuição energética efetuada pelo HUEPS, o MM permite visualizar toda a informação que está a ser obtida e gerada em tempo real. A posição atual de cada utilizador, o consumo global da habitação, de cada divisão e de cada utilizador podem ser visualizados na UI à medida que são atualizados. Para além do processo de monitoração em si, este módulo tem de garantir uma boa gestão dos recursos da *EB* através da SMI. Operações como a consulta e adição de registos, configuração do protocolo de comunicação e teste da ligação do HUEPS à *EB* são facultados e supervisionados pelo MM.

O controlo e gestão da ESI são efetuados através do módulo “Serviços Externos” (***E****xternal* ***S****ervices* ***M****odule*). Como a informação existente no HUEPS pode ser de carácter sigiloso, os sistemas externos têm de ser configurados no ESM, de modo a conseguirem aceder aos serviços disponibilizados. Assim é possível configurar quais as informações que podem ser consultadas por cada sistema,

personalizando o nível de interação desejado. A gestão da ligação com o WiLOS é também realizada através deste módulo.

O módulo "Estatística" (***S**tatistics **M**odule*) é responsável por organizar a informação referente a todo o consumo energético verificado na habitação. Esta informação pode ser apresentada sobre a forma de imagens, gráficos ou outros meios visuais que satisfaçam os pedidos dos utilizadores. A consulta do histórico mensal da habitação, ou de um determinado utilizador, torna-se rapidamente numa ferramenta importante para que os habitantes mudem os seus comportamentos energéticos. São utilizadas a camada Entidade e a UI para a criação e apresentação dos elementos descritos.

Entidade

A camada Entidade é composta pelo módulo "Gestão e Armazenamento de Informação" (***I**nformation and **S**torage **M**odule*). Este módulo desempenha as mesmas funções que os ISMs do mWiLOS e do sWiLOS, de modo a possibilitar o armazenamento, gestão e coerência da informação tratada pelo HUEPS. A informação é acedida pela camada Controlo para garantir as funcionalidades de cada módulo. A sua estruturação é realizada através de uma BD modelada pelo DER do subcapítulo 4.2.1.3.

4.2.1.3 Modelo de Dados

A organização da informação do HUEPS obedece ao modelo definido pelo DER representado na figura 4.57. Este pode ser seccionado em 6 áreas de acordo com o tipo de informação a manipular: "*Smart Meter Information*"; "*House and Energy Supplier Information*"; "*Users Information*"; "*Energy Consumption Information*"; "*External Services*" e "HUEPS *Configuration*".

A área "*House and Energy Supplier*" é responsável pela modelação dos elementos relativos à habitação e à sua topologia. Uma vez que se pretende efetuar o sincronismo de informação entre WiLOS e HUEPS, as entidades "*House*", "*House Door*", "*House Division*" e "*House Image Info*" herdam os atributos das entidades "*House*", "*House Door*", "*House Division"* e "*House WiLOS*", previamente definidos no DER do mWiLOS no subcapítulo 4.1.1.3. Para além disso, esta área também representa o fornecedor de energia da habitação, através da entidade "*Energy Supplier*", e as tarifas energéticas atualmente aplicadas (entidade "*Energy Supplier Fare Detail"*).

Do mesmo modo que a área "*House and Energy Supplier*", a área "*User Information*" possui entidades diretamente equivalentes às entidades da área "*User Information*" do DER do mWiLOS. Desta forma, todos os dados referentes aos utilizadores existentes no WiLOS são facilmente associados ao HUEPS, onde as credenciais "WiLOS" continuam a ser utilizadas para se realizar o acesso ao sistema.

A entidade "*User Location*" armazena todas as localizações dos utilizadores requisitadas ao WiLOS, que são utilizadas pelo HUEPS para se efetuar a distribuição de consumo energético.

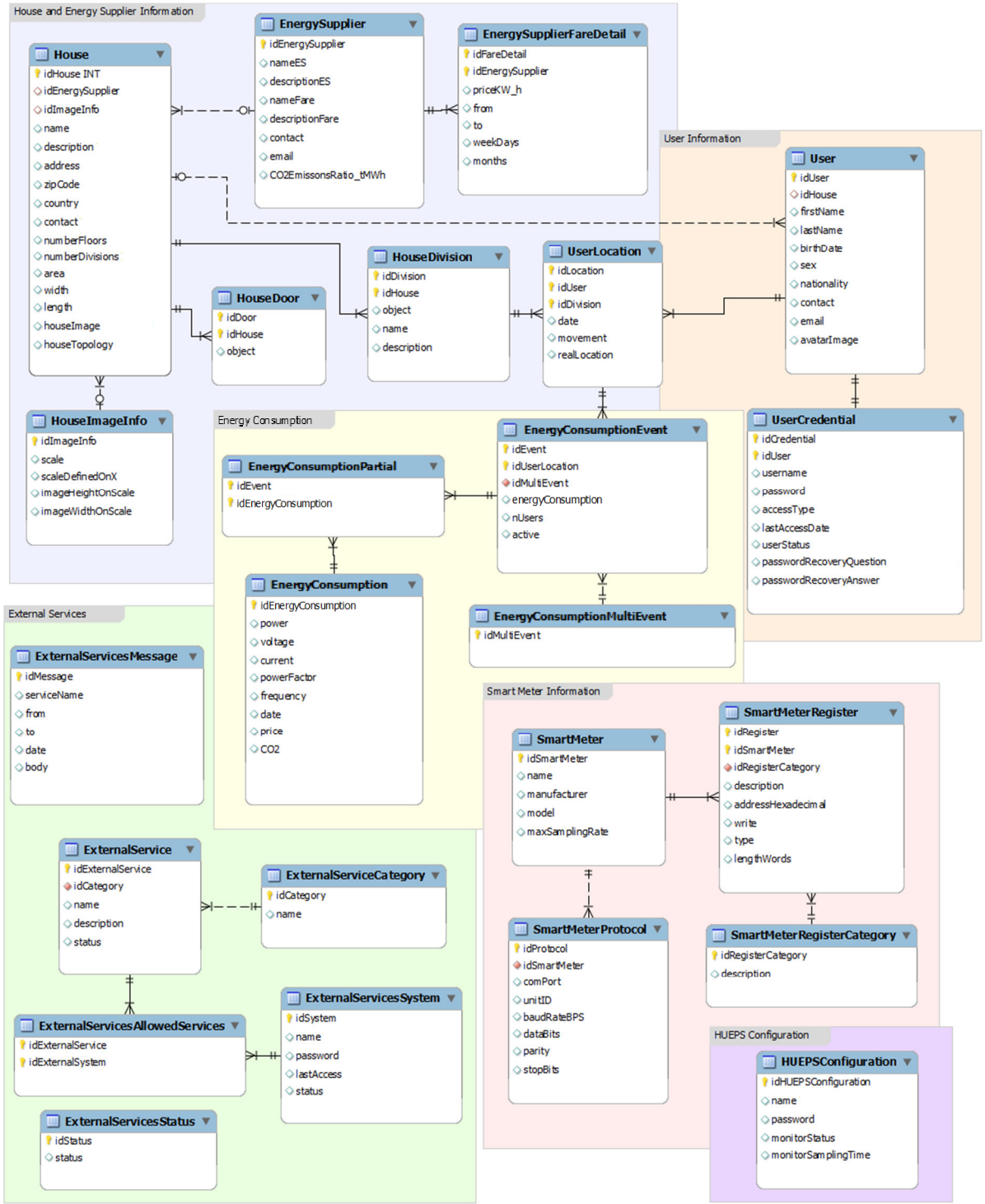


Figura 4.57 - Diagrama de entidades e relacionamentos do HUEPS – Visão Lógica.

A área "*Energy Consumption*" é responsável por toda a informação energética que circula no HUEPS. A entidade "*Energy Consumption*" define uma amostra de consumo energético habitacional, através de características como a corrente, a tensão e o fator potência. O conceito de evento energético, isto é, qualquer ação realizada por um utilizador, numa determinada divisão, que origine uma variação energética, é representada pela entidade "*Energy Consumption Event*". Uma vez que certos eventos ocorrem devido às ações de múltiplos utilizadores, recorre-se à entidade "*Energy Consumption Multi Event*" para os diferenciar. Por fim, a entidade "*Energy Cosumption Partial*" efetua a ligação entre os eventos ativos e as amostras recolhidas em tempo real, de modo a tornar toda a informação energética coerente.

Para se analisar o consumo energético habitacional é necessária a utilização de um *Smart Meter* (*SM*). A modelação deste aparelho é realizada pela área "*Smart Meter Information*". A informação recolhida por um *SM* é disponibilizada através de registos de memória (entidade "*Smart Meter Register*"), cuja atualização depende do tempo de amostragem máximo definido pelo fabricante. O acesso aos registos é efetuado através de um protocolo de comunicação série (entidade "*Smart Meter Protocol*"), podendo a comunicação resultar na aquisição dos valores de consumo energético, ou na configuração do próprio *SM*. Uma vez que um *SM* pode ser constituído por muitos registos, a entidade "*Smart Meter Register Category*" permite catalogá-los de acordo com a informação manipulada.

A área "*External Services*" do HUEPS é uma réplica da área "*External Services*" do mWiLOS. Assim, a entidade "*External System*" permite efetuar a autenticação de um sistema externo, através da definição de um sistema válido no HUEPS. A associação de serviços (entidade "*External Service*") a sistemas externos é efetuada pela entidade "*External Service Rule*". Por fim, todas as mensagens trocadas entre HUEPS e sistemas externos são armazenadas na entidade "*External Services Message*".

A configuração do funcionamento interno do HUEPS é realizada pela área "*HUEPS Configuration*" e a sua única entidade. Nesta definem-se as credenciais utilizadas pelo HUEPS durante a troca de informação com os sistemas externos, como o WiLOS. Alguns parâmetros que permitem definir o modo de funcionamento do módulo Monitoração, como o seu estado e tempo de amostragem, são também aqui representados.

Uma maior pormenorização das escolhas efetuadas para cada entidade dentro de cada área apresentada no DER da figura 4.57 é disponibilizada no anexo E.

4.2.2 Implementação - HUEPS

A implementação do HUEPS consiste na materialização do modelo concetual do subcapítulo 4.2.1. A concretização deste sistema implica a escolha de soluções tecnológicas para a sua implementação, bem como as suas aplicações em cada módulo, de forma a garantir o funcionamento do HUEPS.

4.2.2.1 Tecnologias Adotadas

O HUEPS recorre maioritariamente às mesmas tecnologias utilizadas para o desenvolvimento do mWiLOS. Desta forma, utiliza-se a linguagem C# para implementar todas as camadas da arquitetura do HUEPS exceto o módulo "Interface com o Utilizador", que usa a linguagem XAML para definir as suas funcionalidades. O ambiente de desenvolvimento utilizado continua a ser o Microsoft Visual Studio 2012.

Toda a informação gerada e recebida pelo HUEPS é armazenada numa base de dados. Esta BD encontra-se a cargo do SGBD MySQL fornecido pelo XAMPP. A comunicação efetuada pelos HUEPS e o SGDB é mediada pelo MySQL Connector/.Net, tal como no mWiLOS. O MySQL Workbench também é utilizado para criar a BD do HUEPS, assim como para testar e validar os mecanismos MySQL implementados.

Os serviços externos do HUEPS são disponibilizados na rede WiLOS através de *webservices*. Isto possibilita a qualquer sistema externo devidamente autorizado e autenticado o acesso à informação cedida pelo HUEPS. Os mecanismos de divulgação de serviços e de interoperabilidade inerentes aos *webservices* voltam a ser determinantes para a continuação da utilização desta tecnologia para disponibilizar informação para o exterior. Os *webservices* do HUEPS encontram-se alojados no servidor do IIS fornecido pela Microsoft com o Windows 7 ou 8 (versão "Professional" ou "Ultimate"), à semelhança dos serviços externos do WiLOS.

A comunicação entre o HUEPS e a *EB* é realizada através de um conversor USB – RS445/422. Como a *EB* só possui disponível uma porta RJ-42, é também necessária a utilização de um adaptador RJ-42 – RS445/422 para concluir a ligação. Posteriormente, o protoloco Modbus permite o acesso aos registos da *EB* a partir desta ligação.

A figura 4.58 ilustra a configuração experimental que possibilita o funcionamento do HUEPS. Uma vez que a aplicação do HUEPS numa habitação é algo demasiado ambicioso, pois implicaria a colocação da *EB* no quadro elétrico da habitação em questão, optou-se por construir uma estrutura que simule um quadro elétrico. Esta estrutura de madeira possui três interruptores, onde são ligados os diversos aparelhos elétricos da habitação, e um disjuntor de 16 A, que é utilizado para limitar a corrente máxima que pode existir no sistema, como medida de precaução.

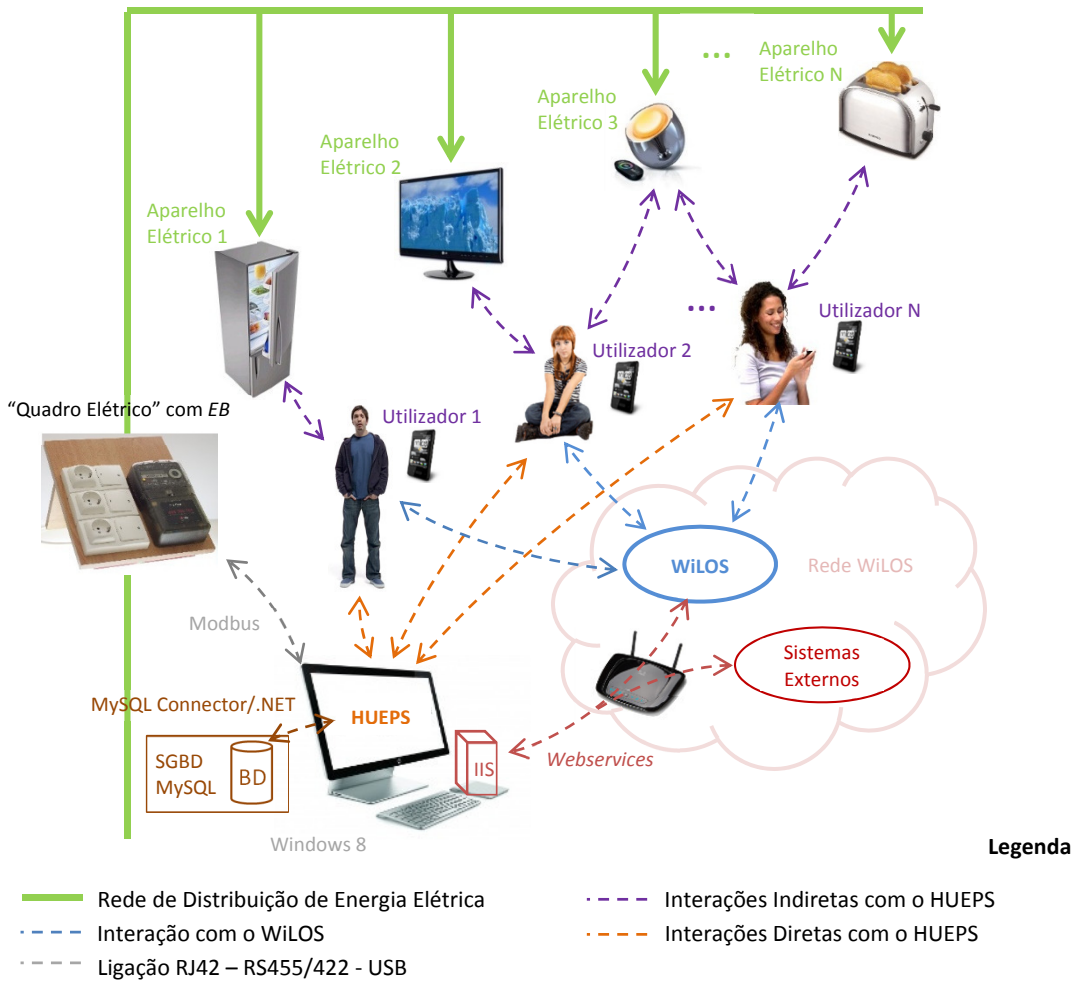


Figura 4.58 - Configuração experimental do HUEPS e atores intervenientes.

4.2.2.2 Pormenorização do Modelo Arquitetural

O HUEPS é uma aplicação WPF que obedece à mesma estrutura do mWiLOS. Esta é composta pelo nível de apresentação, definido em XAML, e pelo nível comportamental, implementado em C#. O módulo UI do HUEPS é concretizado através do nível de apresentação, sendo constituído pela interface gráfica apresentada ao utilizador. O nível comportamental define a camada Controlo e os restantes módulos da camada Interface, de modo a implementar todas as funcionalidades do HUEPS.

A camada Entidade, e o módulo ISM, são implementados pelo *namespace* "_HUEPS_Database" e pela BD do HUEPS. Este define todas as classes e as funcionalidades que permitem aceder e gerir a BD. Esta BD resulta da concretização da DER do subcapítulo 4.2.1.3. A comunicação entre o HUEPS e o SGBD MySQL é efetuada através do MySQL Connector/.Net.

Estrutura da Camada Interface

O objetivo inicial do HUEPS era a interação com o utilizador através do computador e de um dispositivo móvel, como o *smartphone* do utilizador. No entanto, só foi possível a conclusão da versão "computador" dentro do limite de tempo previsto. Ainda assim, a interface gráfica resultante possui uma grande influência de uma eventual versão "*smartphone*". Esta é gerida e implementada através da UI, obedecendo à estruturação gráfica da figura 4.59

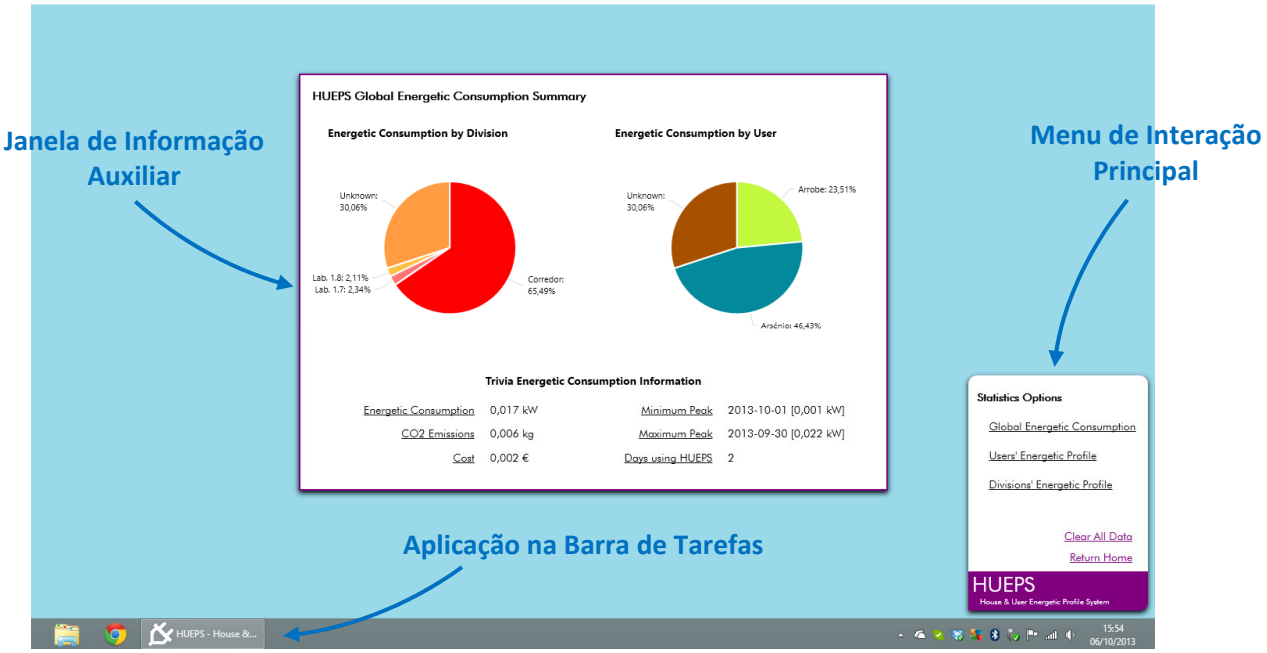


Figura 4.59 - Organização e estruturação da interface gráfica do HUEPS, disponibilizada pela UI.

O menu de interação principal, como o nome indica, é a interface gráfica primária do HUEPS. Esta permite a navegação ao longo de todas as áreas do HUEPS, de modo a se visualizar e manipular a informação disponibilizada por cada área. A estrutura base deste menu é implementada pela classe "_interfaceMenu", pertencente ao *namespace* "_Interface".

A variação do *layout* gráfico da UI ocorre conforme a ação a ser realizada: navegação; consulta de informação; manipulação de informação ou híbrido. Estes *layouts* são implementados pelas classes que definem os módulos da camada Controlo, onde cada módulo oferece um conjunto específico de funcionalidades, tornando cada *layout* único. Alguns destes encontram-se disponíveis para consulta no anexo H.

Caso seja necessária a interação com um maior volume de informação, recorre-se à janela de informação auxiliar, implementada pela classe "_interfaceWindow". Esta é especialmente útil para casos como o do módulo Estatística (camada Controlo), permitindo visualizar a informação gerada no

HUEPS sob a forma de gráficos. Todos os estilos e tipos de letra utilizados para melhorar a aparência da interface gráfica encontram-se definidos pelos dicionários XAML "_styles" e "_comboBoxStyles". A figura 4.60 ilustra a organização do *namespace* que define as interfaces da UI mencionadas.

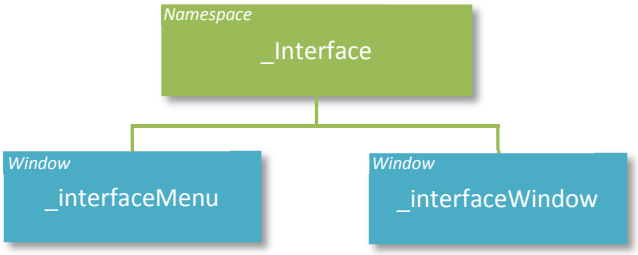


Figura 4.60 - Organização das classes que definem o *layout* gráfico da UI.

Ao longo deste subcapítulo serão referenciadas classes do tipo "UserControl" como "_monitorHome" e "_monitorHomeOptions". Nestes casos, o sufixo "Options" serve para diferenciar qual a classe que recorre ao menu de interação principal ou à janela de informação auxiliar para apresentar a sua informação. Deste modo, todas as classes com sufixo "Options" definem no menu de interação principal todas as opções que permitem atuar ou manipular, de alguma forma, a janela de informação auxiliar (neste caso, a classe "_monitorHome"). Se nenhuma associação entre a classe "_monitorHome" e algum sufixo for explicitamente apresentada, como "Options" ou "Stats", então é correto assumir que se trata de um "UserControl" executado no "menu de interação principal".

A interação do HUEPS com a *EB* é realizada através da SMI. Esta é definida pelas classes "_modbus" e "_modbusResponseConversion". Ambas as classes são alcançáveis pelo *namespace* "_ModbusCommunication", como representado na figura 4.61.

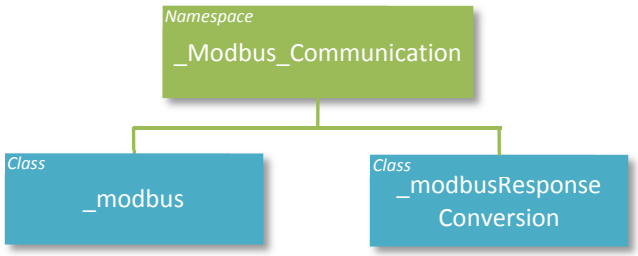


Figura 4.61 - Organização das classes que constituem a SMI da camada Interface.

A classe "_modbus" implementa os mecanismos necessários para a comunicação série através do protoloco Modbus. O seu funcionamento implica a definição de parâmetros como o ritmo de transmissão, número de bits para dados, o tipo de codificação utilizado, bits de paridade, etc., a partir da interface gráfica do HUEPS. O passo seguinte consiste em abrir a ligação e consultar a informação

dos registos da *EB*. A conversão dos dados recolhidos para o tipo de dados adequado vai depender do registo consultado. Esta conversão é realizada a partir das funções definidas na classe "_modbusResponseConversion". A classe "_modbus" é uma adaptação do trabalho disponibilizado pelo utilizador "distantcity" em www.codeproject.com.

A ESI do HUEPS é implementada da mesma forma que a ESI do WiLOS. A disponibilização dos serviços através de *webservices* exige a divisão do sistema em dois projetos, sendo depois realizada a sincronização de serviços entre a ESI e o ESM, de modo a se efetuar a gestão dos mesmos. Os serviços externos do HUEPS também são publicados no servidor fornecido pelo IIS, de forma a ser disponibilizados na rede WiLOS.

Uma vez que os dados trocados entre os sistemas podem conter informações sigilosas, como os dados dos utilizadores do HUEPS ou dos próprios sistemas, torna-se necessário definir mecanismos que garantam a segurança da informação em trânsito. Assim, tal como para a ESI do mWiLOS, utiliza-se um certificado digital SSL para implementar uma ligação segura HTTPS. Por motivos de simplificação, reutilizou-se o certificado digital criado para o WiLOS.

Estrutura da Camada Controlo

As funcionalidades relacionadas com a manipulação de informação da habitação e do fornecedor de energia são implementadas pelo HESM. Este módulo é representado pelo *namespace* "_HouseAndSupplier" e engloba as classes e *namespaces* da figura 4.62.

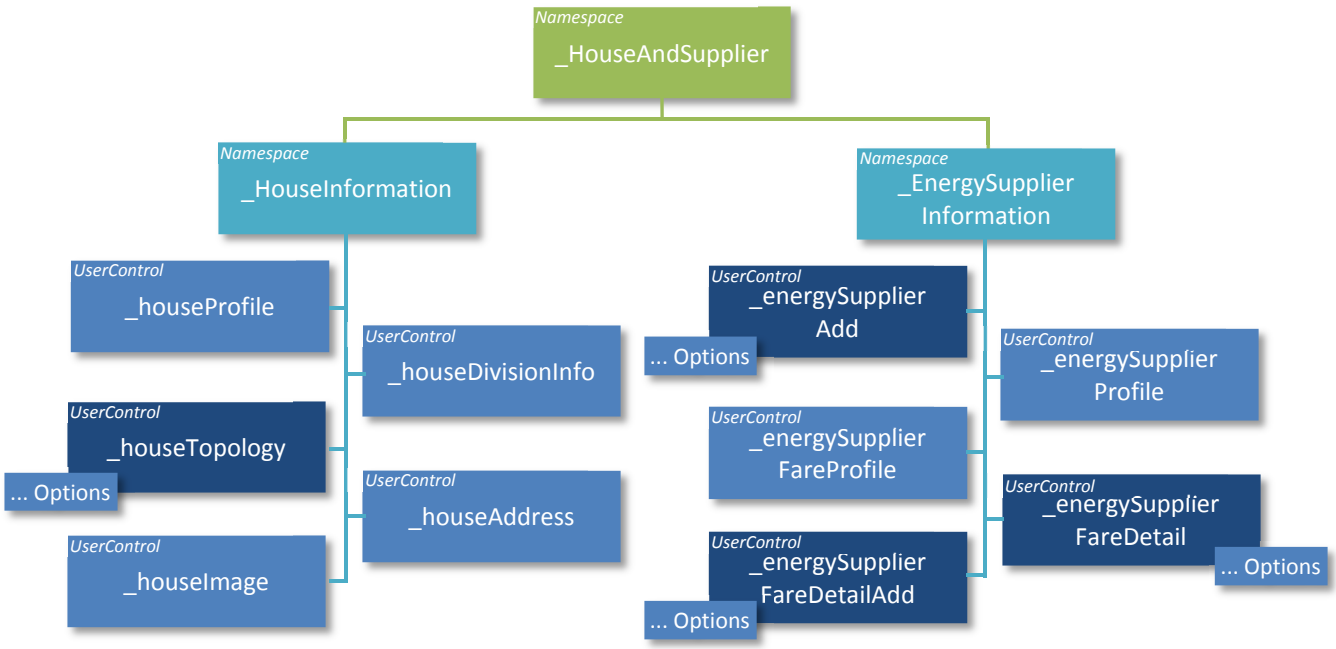


Figura 4.62 - Organização das classes que constituem o HESM da camada Controlo do HUEPS.

O HUEPS não permite a manipulação da informação referente à habitação. Desta forma, as classes pertencentes a "_houseInformation" apenas permitem a consulta da informação sincronizada entre o HUEPS e o WiLOS. A sincronização de informação pode ser forçada pelo utilizador, obrigando a utilização da ESI da camada Interface. Os dados relativos ao perfil da habitação, morada e sua tipologia são depois recolhidos da BD através do ISM (camada Entidade), e apresentados na camada Interface a partir de "_houseProfile", "_houseAddress", "_houseImage", "_houseTopology" e "_houseDivisionInfo".

A informação do fornecedor de energia já possui uma maior dinâmica, uma vez que o HUEPS permite a definição do perfil do fornecedor de energia, bem com a gestão do tarifário aplicado à habitação. Os elementos mais importantes a considerar são a definição da taxa de emissões de CO2 e a definição das diversas tarifas que compõem o tarifário. Estes permitem a determinação das emissões de CO2 e do custo monetário associados ao consumo energético verificado na habitação. A taxa de CO2 pode ser introduzida em "_energySupplierAdd", caso não exista nenhum perfil armazenado na BD, ou em "_energySupplierProfile". A lista de tarifas que compõem o tarifário podem ser consultadas em "_energySupplierFareDetail". Por fim, o processo de adição de uma nova tarifa é efetuada por "_energySupplierFareDetailAdd".

A gestão de utilizadores do HUEPS é efetuada pelo módulo Utilizadores. As classes "_usersHome", "_usersProfile" e "_usersCredentials" são utilizadas para definir o *namespace* "_Users" que implementa o UM (figura 4.63). A primeira classe permite a visualização de todos os utilizadores registados no HUEPS. A segunda e a terceira são utilizadas para consultar o perfil e a conta de um utilizador em particular. Mais uma vez, o HUEPS apenas atua como consumidor de informação, não permitindo a alteração dos dados relativos aos utilizadores. Caso se queira consultar os dados mais atuais disponibilizados pelo WiLOS, a sincronização de informação é efetuada pela ESI.

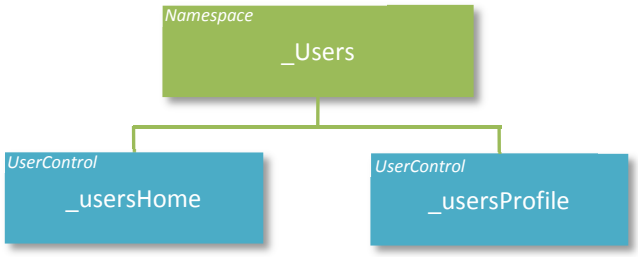


Figura 4.63 - Organização das classes que definem o UM da camada Controlo do HUEPS.

O módulo Monitoração controla toda a informação e mecanismos que permitem a aquisição e distribuição do consumo energético da habitação. Tarefas como a definição do protocolo Modbus, gestão de registos da *EB*, históricos de mensagens trocadas e visualização em tempo real da

distribuição energética podem ser acedidas através deste módulo. O *namespace* "_Monitor" agrupa as classes que definem o MM, obedecendo à estrutura indicada na figura 4.64.

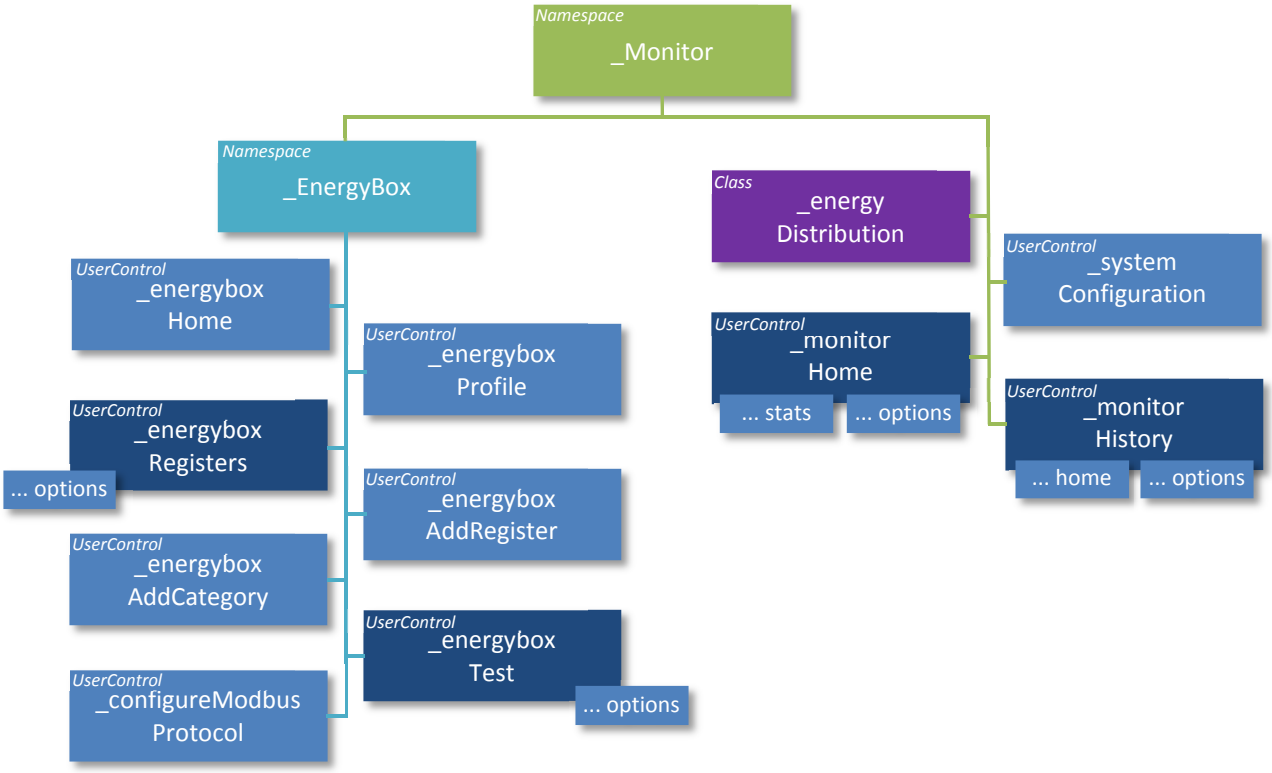


Figura 4.64 - Organização das classes que definem o MM da camada Controlo do HUEPS.

Todas as funcionalidades relativas à gestão e configuração da *EB* são implementadas pelo *namespace* "_EnergyBox". As classes "_energyboxRegisters", "_energyboxAddRegister" e "_energyboxAddRegister" permitem a visualização dos registos e categorias de registos já definidos, bem como a adição de novos elementos ou a remoção daqueles já existentes. A comunicação entre a *EB* e o HUEPS pode ser testada através de "_energyboxTest". A acessibilidade desta depende da definição prévia dos registos da *EB* e o do protocolo Modbus (classe "_configureModbusProtocol").

A classe "_energyboxHome", para além de implementar o menu de navegação que permite o acesso às funcionalidades da *EB*, também possibilita a definição do tempo de amostragem a utilizar durante o processo de aquisição do consumo energético da habitação. A definição e visualização do perfil da *EB* fica a cargo da classe "_energyboxProfile".

Só após a configuração da *EB* é que se torna possível a ativação do serviço de monitoração. A classe "_monitorHomeOptions" disponibiliza ao utilizador a opção para ativar ou desativar o serviço conforme a necessidade. Quando o serviço se encontra ativo, a visualização da distribuição energética em tempo real passa a ser acessível. Esta é implementada pelas classes "_monitorHome" e

"_monitorHomeStats". A classe "_monitorHomeOptions" também permite o acesso ao histórico de mensagens trocadas entre o HUEPS e a *EB* (classe "_monitorHistory") e ao menu de configuração da *EB* (classe "_energyBoxHome").

A classe "_energyDistribution" é a chave de todo o módulo Monitoração. Esta implementa os mecanismos que permitem efetuar a distribuição do consumo energético pelas divisões e pelos utilizadores da habitação. O seu funcionamento é detalhado no subcapítulo 4.2.2.4.

A definição e gestão dos sistemas com privilégios para aceder aos serviços externos do HUEPS é efetuada pelo módulo Serviços Externos. Este módulo recorre à ESI para sincronizar os serviços disponíveis com os serviços registados no ISM (camada Entidade). Desta forma é possível definir o estado de um serviço especifico, bem como quais os sistemas que lhe podem aceder. O *namespace* "_ExternalServices", que implementa o ESM, encontra-se estruturado de acordo com a figura 4.65.

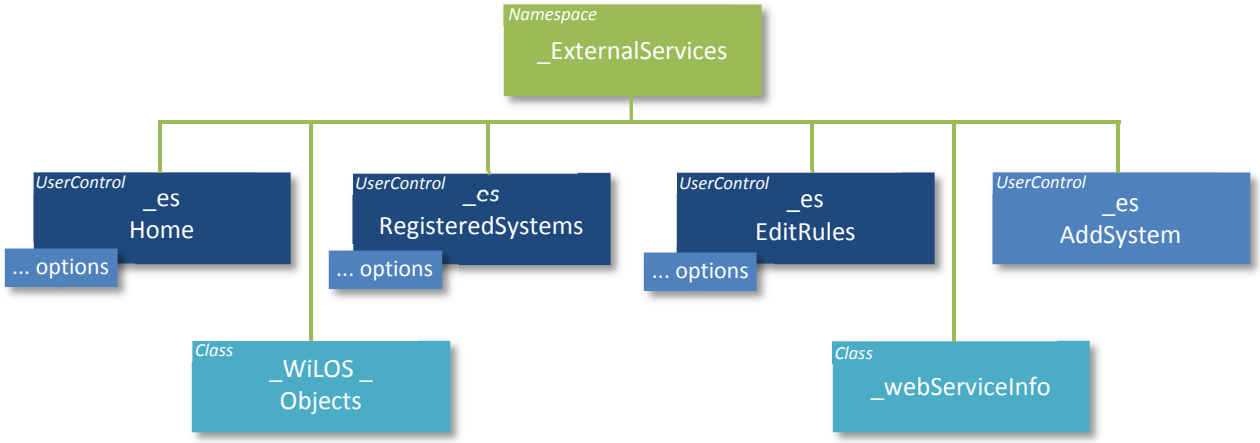


Figura 4.65 - Organização das classes que definem o ESM da camada Controlo do HUEPS.

Todas as classes pertencentes a este *namespace* oferecem exatamente as mesmas funcionalidades que o *namespace* "_ExternalServices" do mWiLOS, detalhado no subcapítulo 4.1.3.2. A única diferença encontra-se na adaptação da UI implementada pelas classes "_esHome", "_esRegisteredSystems", "_esEditRules" e "_esAddSystem" para o novo formato gráfico disponibilizado pelo HUEPS. Uma lista de todos os serviços externos disponibilizados pelo HUEPS é apresentada no anexo J.

Uma vez que o HUEPS sincroniza informação com o WiLOS, a utilização de um serviço externo implica a receção de uma resposta XML equivalente à informação desejada. Para se obter o objeto correto, isto é, um objeto do mesmo tipo que a informação recebida, recorre-se à classe "_WiLOS_Objects" para realizar a conversão. O tipo de objeto para o qual se deseja converter vai depender do serviço externo consultado.

As funcionalidades que permitem a visualização da distribuição do consumo energético da habitação encontram-se implementadas pelo módulo Estatísticas. A informação referente ao consumo, custos e emissões de CO2 é apresentada ao utilizador sob a forma de gráficos ou outros elementos estatísticos, de modo a facilitar a sua interpretação. O *namespace* que define o SM encontra-se estruturado segundo a figura 4.66.

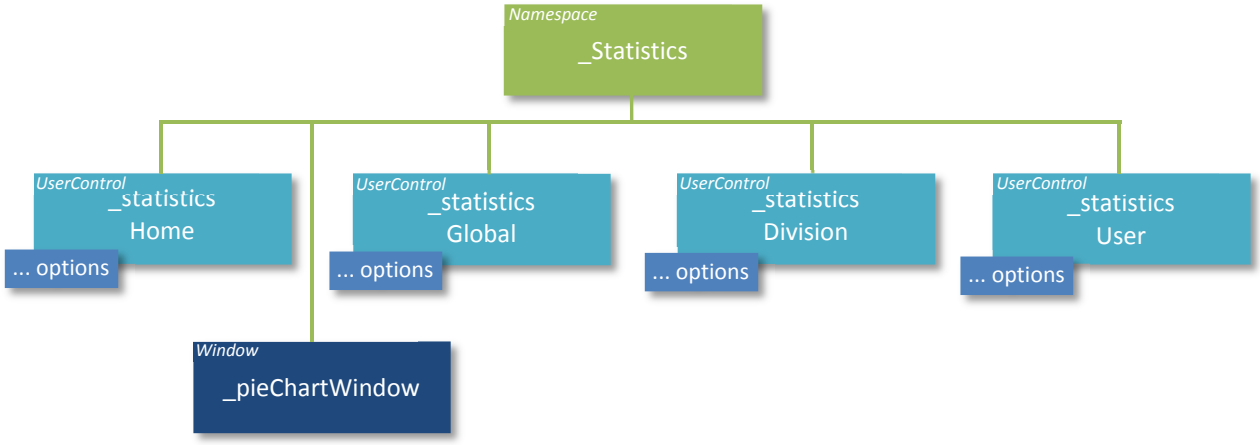


Figura 4.66 - Organização das classes que definem o SM da camada Controlo do HUEPS.

Os históricos de consumo energético são representados por gráficos de barras ou de linhas, enquanto a distribuição de consumo por utilizadores ou por divisões é ilustrado a partir de gráficos circulares. Estes gráficos podem ser consultados através das classes "_statisticsGlobal", "_statisticsDivision", "_statisticsDivision" e "_statisticsHome".

A geração dos gráficos citados implica a definição do intervalo de tempo que se deseja analisar. Desta forma, pode-se visualizar o consumo global relativo a um ano ou consumo de um utilizador ao longo de um determinado mês. A informação temporal é fornecida através da interação do utilizador com a UI das classes "_statisticsGlobalOptions", "_statisticsDivisionOptions" e "_statisticsUserOptions". A classe "_pieChartWindow" é utilizada quando se pretende analisar com maior detalhe a distribuição do consumo energético verificado num determinado dia, mês ou ano. A sua acessibilidade só é possível caso a análise do consumo energético esteja a ser realizada a partir de um gráfico de barras numa das 3 classes supracitadas.

Do mesmo modo que para o mWiLOS, os gráficos do HUEPS são gerados através da biblioteca amCharts, versão 1.0 para WPF, disponibilizada em www.amCharts.com. Um exemplo de pesquisa estatística é disponibilizado no anexo M.

Estrutura da Camada Entidade

A camada Entidade é composta pelo ISM e pela BD do HUEPS. O ISM implementa todas as funcionalidades que permitem a gestão da BD, através da classe "_HUEPS_Database". Assim, operações básicas como a consulta de informação, inserção, atualização e remoção de dados, são executáveis através deste módulo. Para além destas operações, outras funcionalidades mais complexas e necessárias para o funcionamento do HUEPS, como a consulta do consumo energético de um determinado utilizador a partir de 4 entidades, também são acessíveis através do ISM. Uma lista destas funcionalidades mais específicas é fornecida no anexo L. As figuras 4.67 e 4.68 ilustram a classe "_HUEPS_Database" e todas as subclasses que implementam as funcionalidades de gestão da BD.

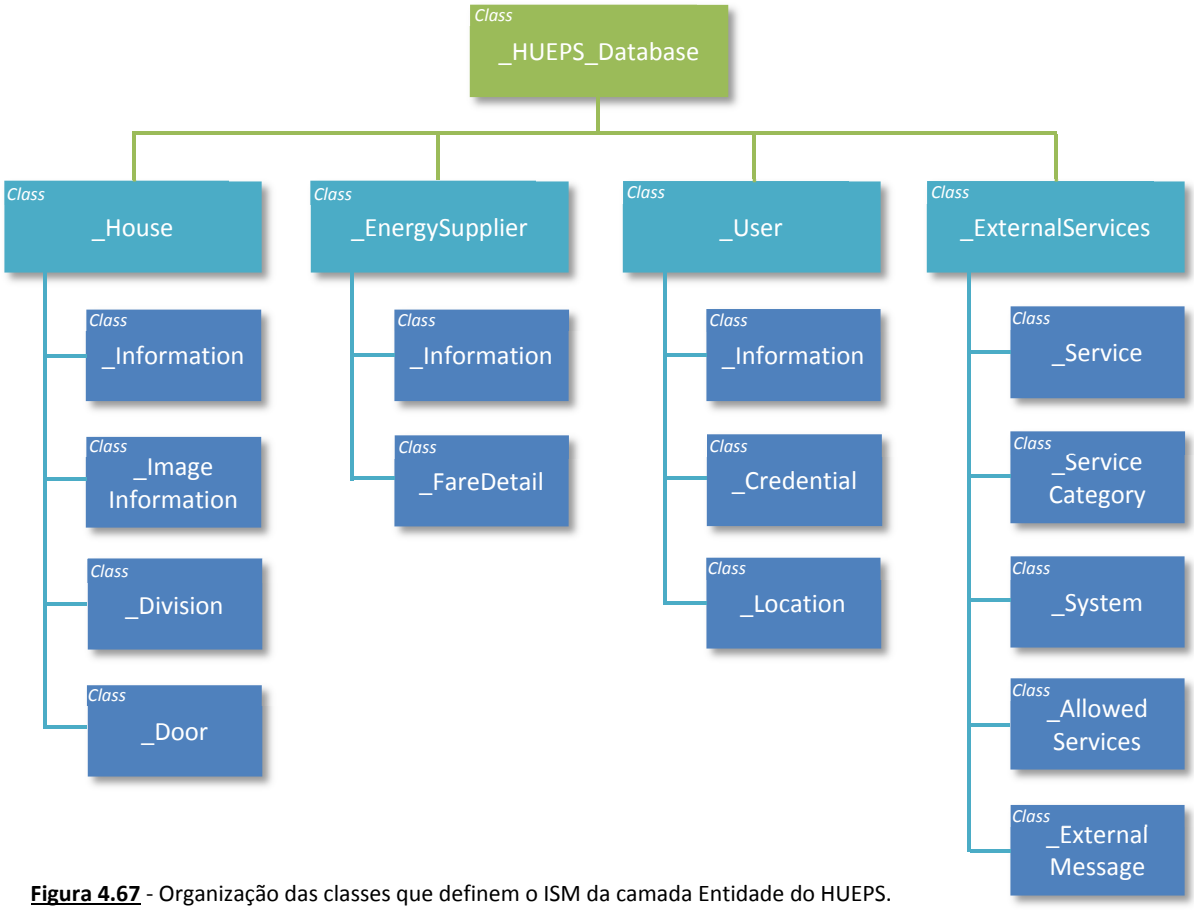


Figura 4.67 - Organização das classes que definem o ISM da camada Entidade do HUEPS.

A classe "_HUEPS_Database" permite definir os elementos necessários para efetuar a ligação com a BD. Também possui funcionalidades para a criação, remoção ou recriação da BD do HUEPS. As restantes subclasses responsáveis por manipular as tabelas da BD são acessíveis a partir do segundo nível desta classe.

Estas subclasses encontram-se organizadas de forma semelhante ao DER apresentada no subcapítulo 4.2.1.3. Assim, a classe "_House" permite a manipulação das entidades "*House*", "*House Door*", "*House Division*" e "*House Image Info*", enquanto a classe "_EnergySupplier" fica responsável

pelas funcionalidades das entidades "*Energy Supplier*" e "*Energy Supplier Fare Detail*". Estas duas classes gerem a informação da área "*House and Energy Supplier Information*". A área "*User Information*" é acedida através da classe "_User" que recorre às subclasses "_Information", "_Credential" e "_Location" para interagir com as entidades "*User*", "*User Credential*" e "*User Location*", respetivamente. De forma equivalente, as classes "_SmartMeter", "_ExternalServices" e "_HUEPS_Configuration" afetam as áreas "*Smart Meter Information*", "*External Services*" e "*HUEPS Configuration*", e as entidades que as compõem.

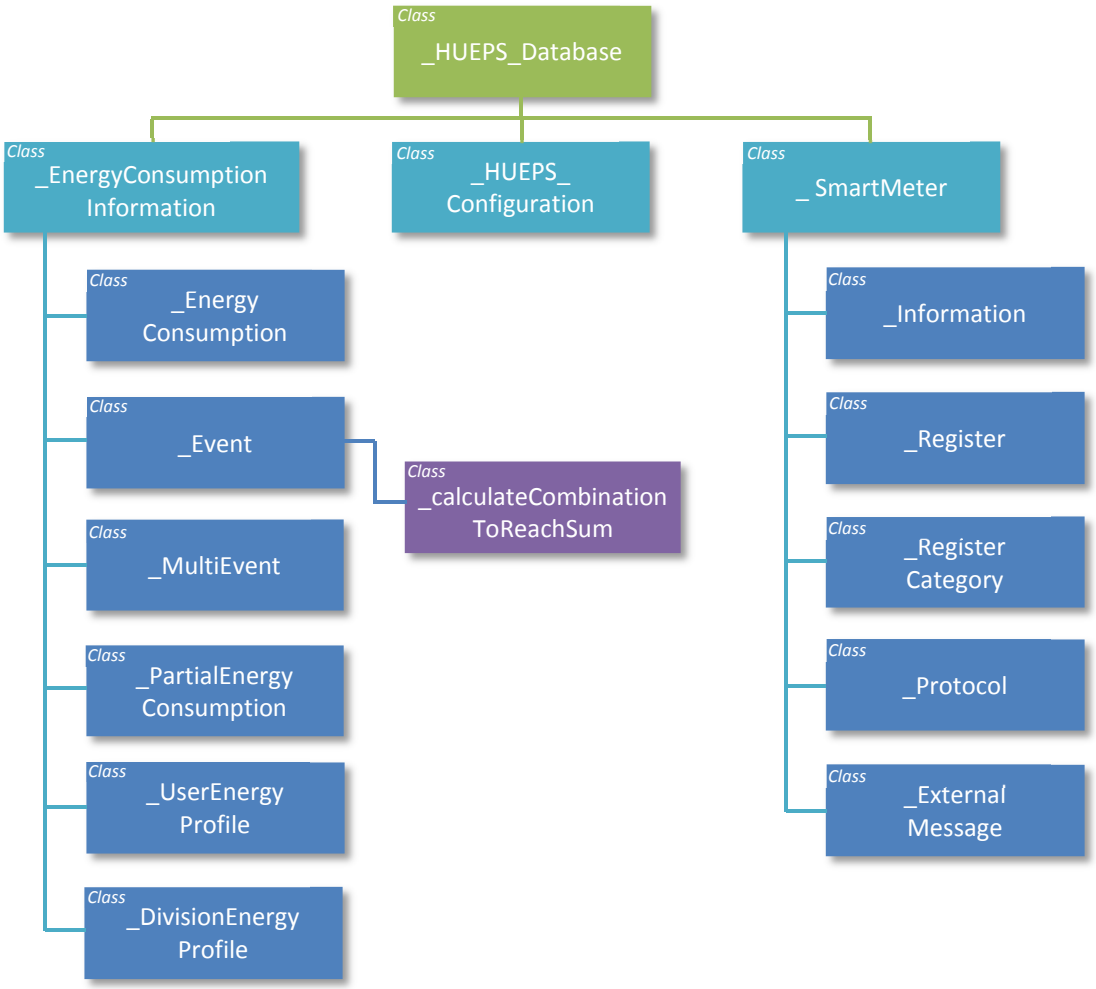


Figura 4.68 - Organização das classes que definem o ISM da camada Entidade do HUEPS (continuação).

A classe "_EnergyConsumptionInformation possibilita a manipulação das entidades da área "*Energy Consumption Information*" através das subclasses "_EnergyConsumption", "Event", "_multiEvent" e "_PartialEnergyConsumption". Para além disso, as subclasses "_UserEnergyProfile" e "_DivisionEnergyProfile" implementam funcionalidades específicas para determinar o consumo energético efetuado por uma divisão ou utilizador, ou verificar a distribuição do consumo energético da habitação pelos seus utilizadores num determinado intervalo de tempo.

Destas classes destaca-se a manipulação de eventos energéticos através da classe "_Event". Esta, como todas as outras, engloba as funcionalidades básicas para criação, atualização, remoção e desativação de elementos da BD. No entanto, a desativação de um evento é realizada através da comparação do valor da variação energética verificada com o valor do consumo energético atribuído a um determinado evento ativo. Porém, podem surgir situações em que a variação energética não seja suportada apenas por um evento. Para estes casos é necessário determinar se existe uma combinação de eventos cuja soma de consumo energético seja coincidente com a variação verificada. Para tal recorre-se à classe "_calculateCombinationToReachSum".

Esta classe lida com a problemática da soma dos subconjuntos, que consiste na determinação de um subconjunto não-vazio de valores cuja soma é '0', a partir de um conjunto inicial de valores inteiros. Outra maneira de se observar este problema é a seguinte: dado um conjunto de valores inteiros e um inteiro 's', existe algum subconjunto cuja soma resulte em 's'? O maior desafio deste problema traduz-se na complexidade inerente à determinação de todas as combinações possíveis e o tempo necessário para as alcançar. Os parâmetros que determinam a complexidade da pesquisa são o número de variáveis utilizadas no processo de decisão, isto é, o número de elementos do conjunto inicial, e o número de bits necessários para resolver o problema.

Existem várias técnicas que podem ser implementadas para determinar as soluções possíveis. Neste caso, utiliza-se um algoritmo recursivo que constrói um diagrama de árvore com todas as combinações válidas, isto é, cuja soma não exceda o valor 's' pretendido. Uma vez que se está a lidar com variações e consumos energéticos, o algoritmo deve manipular números decimais em vez de inteiros. Também se assume uma determinada margem de erro durante o processo de comparação, uma vez que não se consegue eliminar o ruído associado à rede elétrica. Apenas as soluções que verifiquem as condições descritas são selecionadas para posterior utilização pelos mecanismos de desativação.

A classe "_calculateCombinationToReachSum" é uma adaptação do trabalho disponibilizado em "http://stackoverflow.com" pelos utilizadores "Manuel Salvadores" e "Keith Beller" no tópico "*Finding all possible combinations of numbers to reach a given sum*".

4.2.2.3 Modelo de Dados – Visão Física

Para concretizar o modelo de dados proposto no subcapítulo 4.2.1.3 para o HUEPS, é necessário definir o tipo de dados que cada atributo representa. Na figura 4.69 podem-se visualizar as entidades que pertencem ao modelo de dados, bem como as relações existentes entre elas. O DER apresentado também define os tipos de dados atribuídos a cada um dos seus elementos.

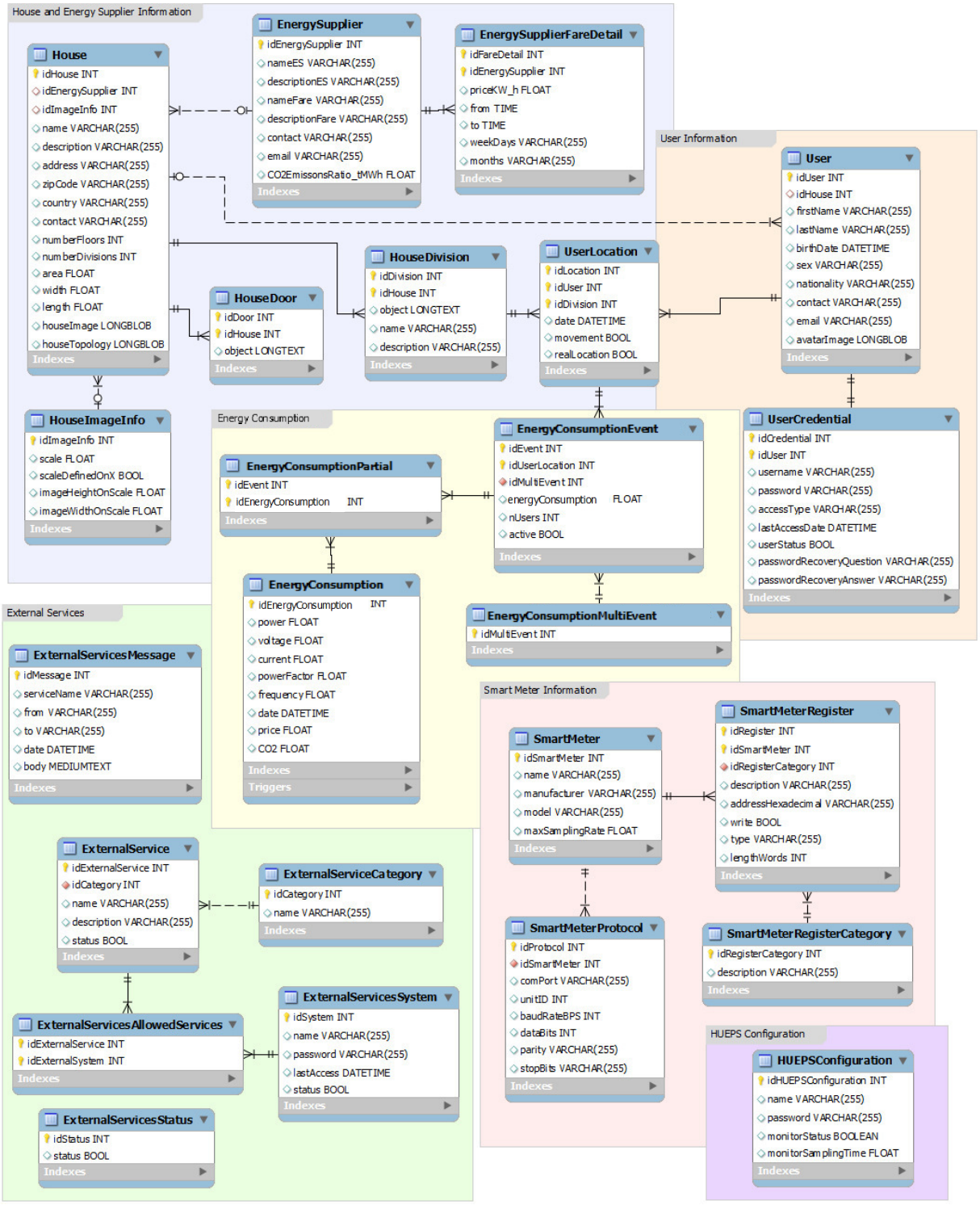


Figura 4.69 - Diagrama de entidades e relacionamentos do HUEPS – Visão Física.

Uma vez que o HUEPS possui funcionalidades de cálculo e determinação de consumos energéticos, emissões de CO2 e custos energéticos, é de se esperar que alguns atributos da BD obedeçam a um formato de unidades pré-estabelecido. A tabela 4.5 destaca estes atributos, bem como as respetivas unidades ou valores esperados para garantir o bom funcionamento do HUEPS.

Atributo	Unidade
CO2EmissionRatio_tMWh	ton/MW-hora (ou eq.)
priceKW_h	€/KW-hora (ou eq.)
power	KW
voltage	V
current	mA
frequency	Hz
weekDays	"Mon", "Tue", "Mon, Tue", …

Atributo	Unidades
monitorSamplingTime	ms
comPort	COM1, COM2, …
baudRateBPS	bit/s
dataBits	bits
stopBits	*enum System.IO.Ports.StopBits*
Parity	*enum System.IO.Ports.Parity*
months	"Jan", "Feb", "Jan, Feb", …

Tabela 4.2 - Listagem de atributos que necessitam informação num formato específico ou respeitando certas unidades.

O bom funcionamento do HUEPS também é assegurado pelas funcionalidades automáticas implementadas na BD. Esta possui dois *triggers* (anexo O) associados à entidade "*Energy Consumption*", que são evocados de modo a garantir a integridade da informação das entidades "*Energy Consumption*" e "*Energy Consumption Partial*".

O primeiro *trigger*, denominado "Obter Valores de CO2 e Custos", atua antes da criação de uma nova entrada na entidade "*Energy Consumption*". Este consulta as tabelas "*Energy Supplier*" e "*Energy Supplier Fare Detail*", de modo a obter os valores da taxa de emissão de CO2 e do custo monetário em vigor durante a aquisição do consumo energético. Estes são associados à amostra recolhida, que fica registada na BD, permitindo a coerência da informação referente aos custos energéticos e às emissões de CO2. Note-se que estes valores variam ao longo do tempo, devido a flutuações da economia, adoção de novas tecnologias, etc.. Desta forma, a acoplação destes valores à própria amostra, permite mais tarde a alteração do tarifário e da taxa de emissões CO2 sem afetar os valores previamente definidos.

Após a inserção desta amostra, o *trigger* "Verificar Eventos Ativos" é acionado. Este verifica todos os eventos ativos na entidade "*Energy Consumption Event*" e efetua a sua propagação para a entidade "*Energy Consumption Partial*". Esta propagação permite associar os identificadores dos eventos ativos com o identificador da amostra obtida, de modo a indicar que naquele instante apenas se encontram ativos aqueles eventos. Isto possibilita a determinação da duração de um determinado evento, que por sua vez permite inferir o consumo e os custos totais associados a um utilizador e a uma divisão.

4.2.2.4 Modelo Comportamental

A distribuição do consumo energético da habitação, bem como as restantes funcionalidades do HUEPS, só são possíveis devido à cooperação entre os diversos módulos das camadas que definem a sua arquitetura. O utilizador, ao interagir com a UI, desencadeia eventos que evocam os serviços implementados nos módulos da camada Controlo. Por sua vez, estes interagem entre si, com o ISM (camada Entidade) e com as interfaces SMI e ESI, de modo a satisfazer os pedidos do utilizador e a realizar o objetivo para o qual o HUEPS foi desenvolvido.

Em termos de modelo comportamental, o HESM, o UM e o SM não introduzem grande complexidade no sistema. Estes módulos apenas gerem a troca de informação entre a UI, a camada Controlo e o ISM para efetuarem a consulta ou modificação da informação requisitada pelo utilizador (exemplo da figura 4.70). O ESM do HUEPS possui o mesmo comportamento do ESM implementado pelo mWiLOS, já detalhado no subcapítulo 4.1.3.4.

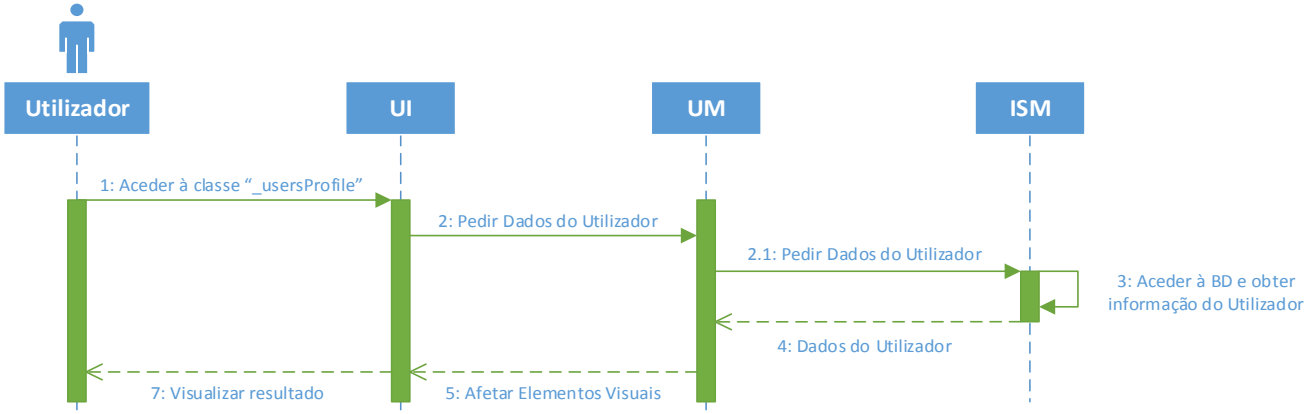


Figura 4.70 - Diagrama de sequência referente à visualização do perfil de um utilizador HUEPS.

Os comportamentos mais interessantes ocorrem nos mecanismos de aquisição e distribuição de consumo energético do MM, que implicam a interação entre SMI, MM e ISM, de forma a satisfazer as premissas definidas no subcapítulo 4.2.1.1. Estes comportamentos são o principal alvo de estudo deste subcapítulo, aprofundando as funcionalidades implementas pela classe "_energyConsumption".

Módulo Monitoração

Para se efetuar a distribuição do consumo energético da habitação é necessário recolher periodicamente o consumo energético global (CEG) e a localização dos utilizadores dentro da habitação (LU). O tempo decorrido entre iterações (tempo de amostragem) depende do tempo de processamento máximo utilizado pelo algoritmo de distribuição de consumo energético. O tempo de amostragem também não deve ser superior ao tempo de amostragem do WiLOS, de modo a evitar perda de informação referente à localização dos utilizadores. A comparação dos valores de CEG e LU

entre iterações é a chave para se efetuar a distribuição do consumo energético. O diagrama da figura 4.71 ilustra a rotina inicialmente utilizada para realizar a recolha destes valores.

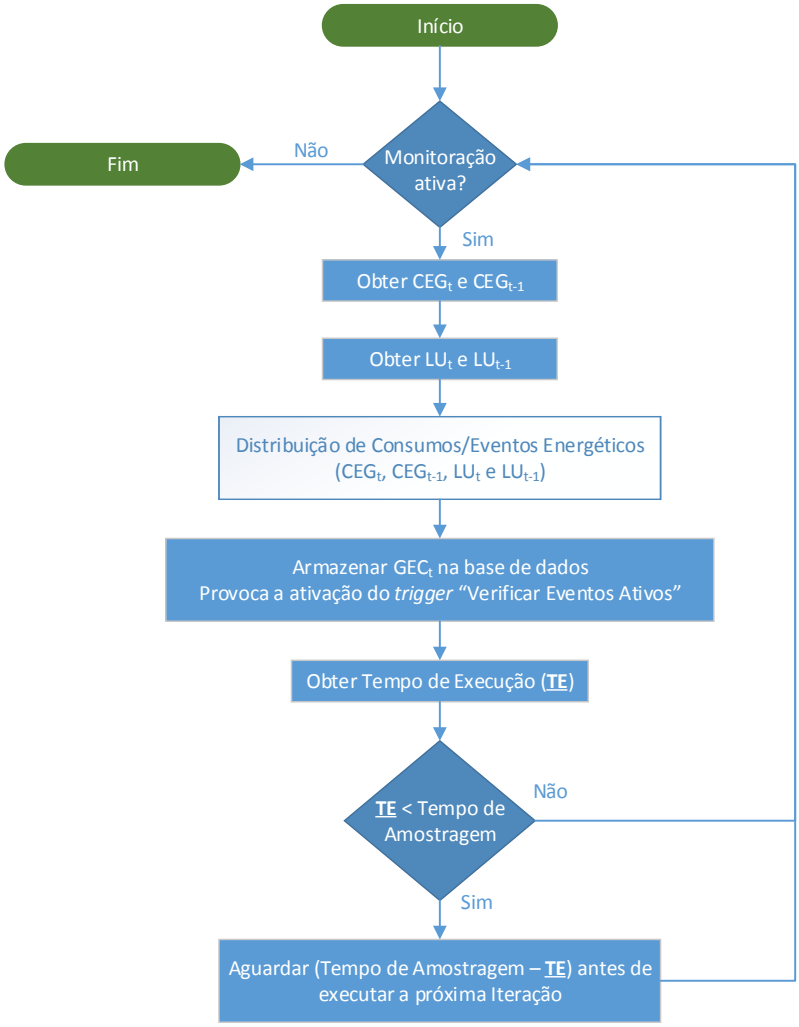


Figura 4.71 - Diagrama de atividade da rotina de Recolha de Dados e Distribuição de Consumo Energético.

No início de cada iteração é verificado se o módulo Monitoração encontra-se ativo, de modo a terminar a execução desta rotina. Seguidamente obtêm-se os valores de consumo energético global corrente da *Energy Box* (CEG_t) e de consumo enérgico global anterior da base de dados (CEG_{t-1}). Através do WiLOS são recolhidas as localizações atuais e anteriores dos utilizadores (LU_t e LU_{t-1}, respetivamente). Caso não existam dados relativos ao instante 't-1', o valor de CEG_{t-1} é nulo e admite-se que nenhum utilizador se encontrava dentro da habitação. Esta informação é fornecida ao algoritmo de Distribuição de Consumos/Eventos Energéticos que analisa os dados e atua de acordo com as premissas definidas no subcapítulo 4.2.1.1.

Após o processamento dos dados e criação, desativação e/ou realocação de eventos energéticos, armazena-se o valor de CEG_t na base de dados. Esta ação desencadeia o *trigger* "Verificar Eventos Ativos" descrito no subcapítulo 4.2.2.3. Por fim, calcula-se o tempo que demorou a execução da rotina

e utiliza-se este valor para aguardar até à próxima interação. Se, por algum motivo, o tempo de execução excedeu o previsto, a próxima iteração inicia imediatamente.

Esta rotina foi inicialmente utilizada para testar o algoritmo de Distribuição de Consumo Energético, mas os resultados obtidos não foram satisfatórios. Como mencionado, os valores de CEG_t e CEG_{t-1} são cruciais para o algoritmo. Através destes consegue-se determinar a variação de consumo energético global (CEGv) que é utilizado para associar, dissociar e realocar o consumo enérgico dos utilizadores. Esta abordagem possui a vantagem de conseguir detetar todas as variações que ocorrem entre dois instantes dados pelo tempo de amostragem do HUEPS. No entanto, esta vantagem torna-se uma desvantagem tendo em conta a natureza do sistema. O exemplo das figuras 4.72 e 4.73 é utilizado como caso de estudo para este problema.

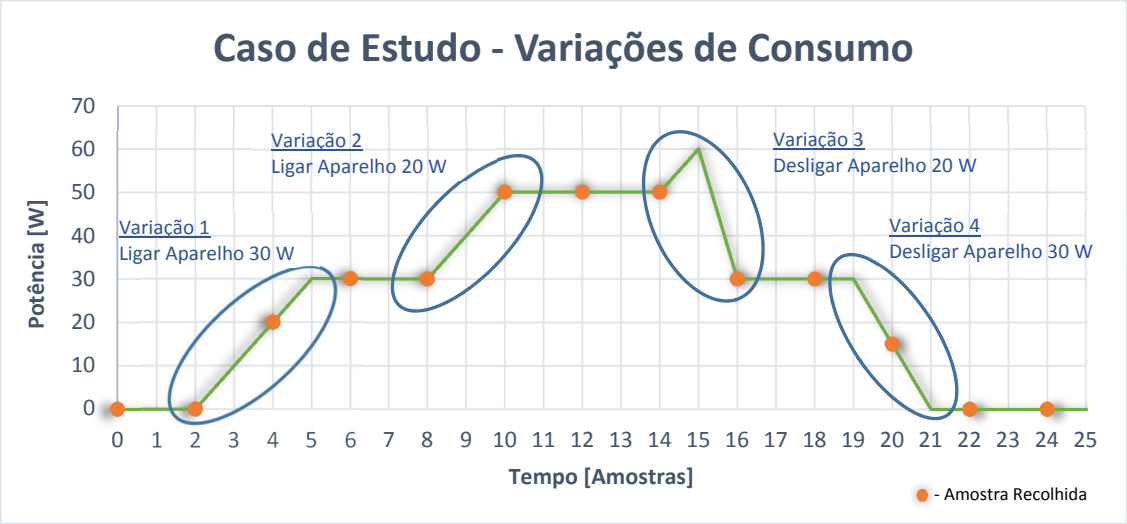


Figura 4.72 - Caso de estudo para a problemática da rotina de Recolha de Dados e Distribuição de Consumo Energético apresentada. Curva do CEG e Variações Energéticas causados por um Utilizador.

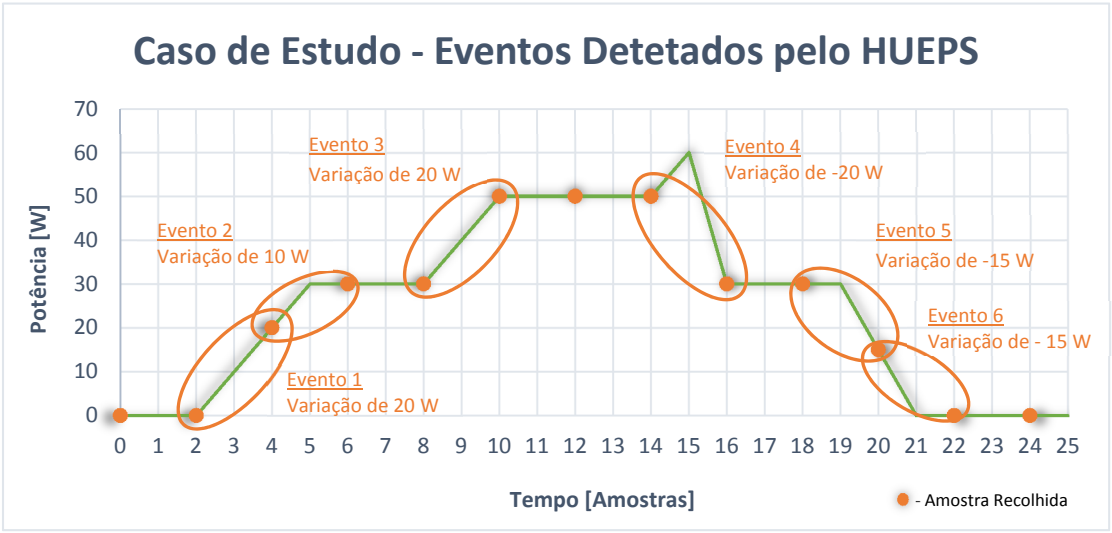


Figura 4.73 - Caso de estudo para a problemática da rotina de Recolha de Dados e Distribuição de Consumo Energético apresentada. Curva de CEG e Eventos Energéticos atribuídos pelo HUEPS.

Na figura 4.72 observa-se o comportamento da rede elétrica da habitação, resultante da interação com um utilizador. Este ligou e desligou dois aparelhos elétricos provocando as 4 variações de consumo representadas. A recolha de dados efetuada pelo HUEPS foi realizada com um tempo de amostragem de 2 s. Na figura 4.73 visualizam-se os eventos energéticos que o HUEPS categorizou. Como se pode verificar, a constante amostragem deteta todas as variações graduais do consumo energético. Caso o período de amostragem coincida com o estado transitório, não é possível para o HUEPS individualizar a variação 1, originando os eventos 1 e 2. Este erro repete-se na variação 4, sendo gerados os eventos 5 e 6. No entanto algumas variações são bem categorizadas, tais com a variação 2 (evento 3) e 3 (evento 4).

Para além da dificuldade na atribuição de eventos, surgem vários problemas durante a fase de desativação e atualização de eventos. A variabilidade da variação energética detetada pelo HUEPS torna menos provável a existência de eventos ativos compatíveis com os decréscimos de consumo energético verificados. As variações 1 e 4, resultantes da interação com o mesmo aparelho, comprovam esta situação. Os eventos 1 e 2 possuem valores de consumo próximos dos valores dos eventos 5 e 6 mas não são equiparados (10 e 20 W vs. 15 e 15 W, respetivamente). Se existisse outro evento ativo com o valor de 15 W, este possuiria menor erro face à variação verificada e seria desativado em vez do evento 5 ou 6.

Para contornar este problema desenvolveu-se outro mecanismo para a determinação de variações energéticas. Note-se que não é possível determinar variações simultâneas, isto é, caso dois ou mais aparelhos sejam ligados ou desligados simultaneamente ou em instantes ligeiramente desfasados, não é possível isolar a variação causada por cada um. Desta forma, não se torna muito interessante a deteção constante de variações na rede elétrica. Propõe-se então uma rotina que só efetue a distribuição do consumo energético após a ocorrência de uma variação e desta estabilizar. A figura 4.74 representa a nova rotina de Recolha de Dados e Distribuição de Consumo Energético implementada pelo módulo Monitoração.

A grande diferença entre as duas rotinas reside na verificação inicial efetuada aos valores de CEG_t e CEG_{t-1}. Se os valores forem díspares, então está-se perante uma variação de consumo e deve-se aguardar pela próxima iteração. Somente quando os valores de CEG_t e CEG_{t-1} voltarem a se equiparar é que se aplica o algoritmo de Distribuição de Consumo Energético. No entanto, a informação relativa ao instante 't-1' não é utilizada pelo algoritmo para efetuar a distribuição. Para tal, este recorre ao consumo energético global verificado antes da ocorrência de uma variação (CEG_{t-n}) e às localizações dos utilizadores verificadas nesse instante (LU_{t-n}). Desta forma utiliza-se toda a informação anterior e posterior à variação, independentemente do número de variações sucessivas ou simultâneas

ocorridas. Após o processamento da informação atualizam-se os respetivos valores de CEG_{t-n}, CEG_{t-1} e LU_{t-n} para a próxima iteração.

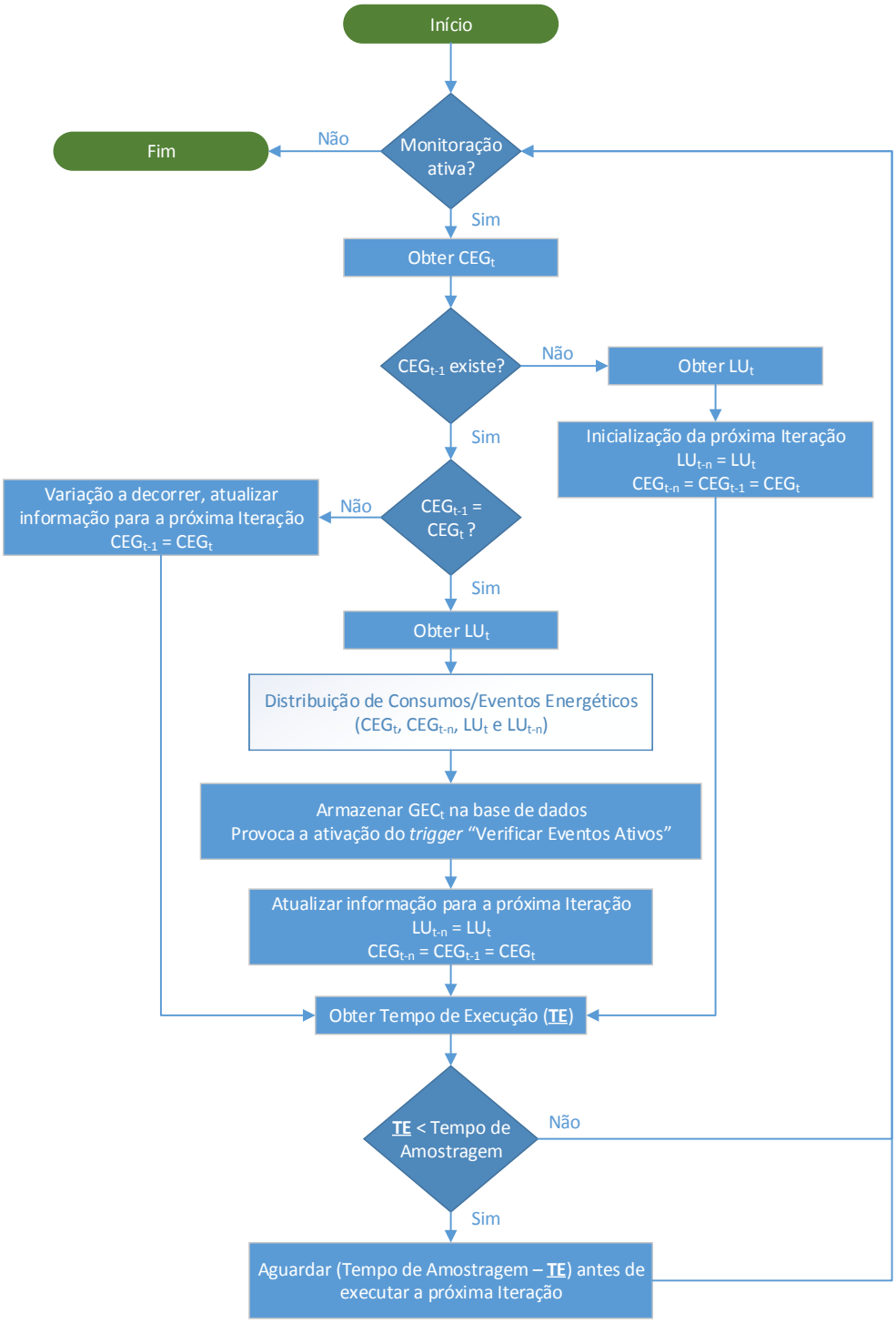


Figura 4.74 - Diagrama de atividade da nova rotina de Recolha de Dados e Distribuição de Consumo Energético.

Todas as comparações de valores de consumo energético realizadas por este algoritmo têm em consideração uma margem de tolerância a definir pelo administrador do HUEPS (CEG_{tol}). Esta margem

é necessária uma vez que a rede elétrica possui num certo nível de ruído, que acaba por influenciar os valores de consumo recolhidos. O valor de CEG_{tol} também determina o valor mínimo a partir do qual se considera a existência de uma variação de consumo energético.

O algoritmo de Distribuição de Consumos/Eventos Energéticos utiliza os valores de CEG_{t-n} e CEG_t para subdividir a problemática da distribuição de consumos em 3 estados: "Criação e Atualização de Eventos"; "Atualização de Eventos devido à Movimentação de Utilizadores" e "Desativação e Atualização de Eventos". Para além destes estados, outra situação a considerar é a "Entrada de Utilizadores na Habitação". A alcançabilidade de cada um destes estados é descrita pelo diagrama da figura 4.75.

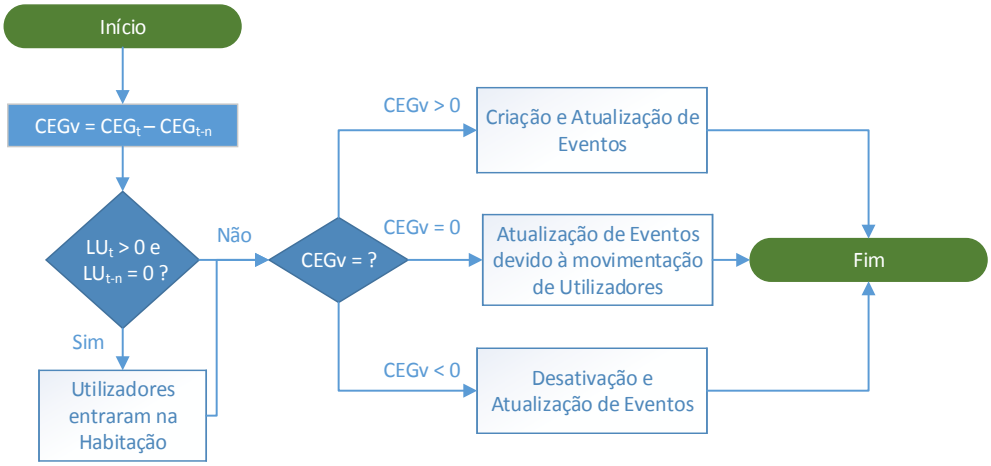


Figura 4.75 - Diagrama de atividade do algoritmo de Distribuição de Consumos/ Eventos Energéticos.

Os valores de CEG_t e CEG_{t-n} permitem determinar a variação do consumo energético global (CEGv). No entanto, antes de se proceder à análise do valor de CEGv, é conveniente determinar se o HUEPS encontra-se numa fase de espera, isto é, se nenhum utilizador se encontra presentemente na habitação. Quando um utilizador entra ou reentra na habitação podem existir eventos previamente atribuídos ou consumos energéticos que nunca chegaram a ser alocados a um determinado utilizador. Estes casos são previamente tratados de modo a tornar coerente o valor de CEG_t com o consumo energético total resultante dos eventos ativos. A figura 4.76 descreve a lógica implementada por este estado, "Utilizadores entraram na Habitação".

As premissas 9 e 11 são concretizadas neste estado. Desta forma, se existirem eventos atribuídos a algum utilizador diferente daqueles que acabaram de entrar na habitação, estes são repartidos pelos utilizadores recém-chegados, tendo em conta as suas posições atuais. Caso existam eventos pertencentes aos utilizadores recém-chegados, estes não são afetados uma vez que os próximos estados lidam com as restantes premissas associadas à distribuição do consumo energético

relacionado com movimentação. Por fim, verifica-se se existia algum CEG na habitação *priori* à chegada dos utilizadores. Se tal se verificar, então o consumo residual da habitação não se encontra atribuído a nenhum utilizador e deve ser repartido pelos utilizadores presentes.

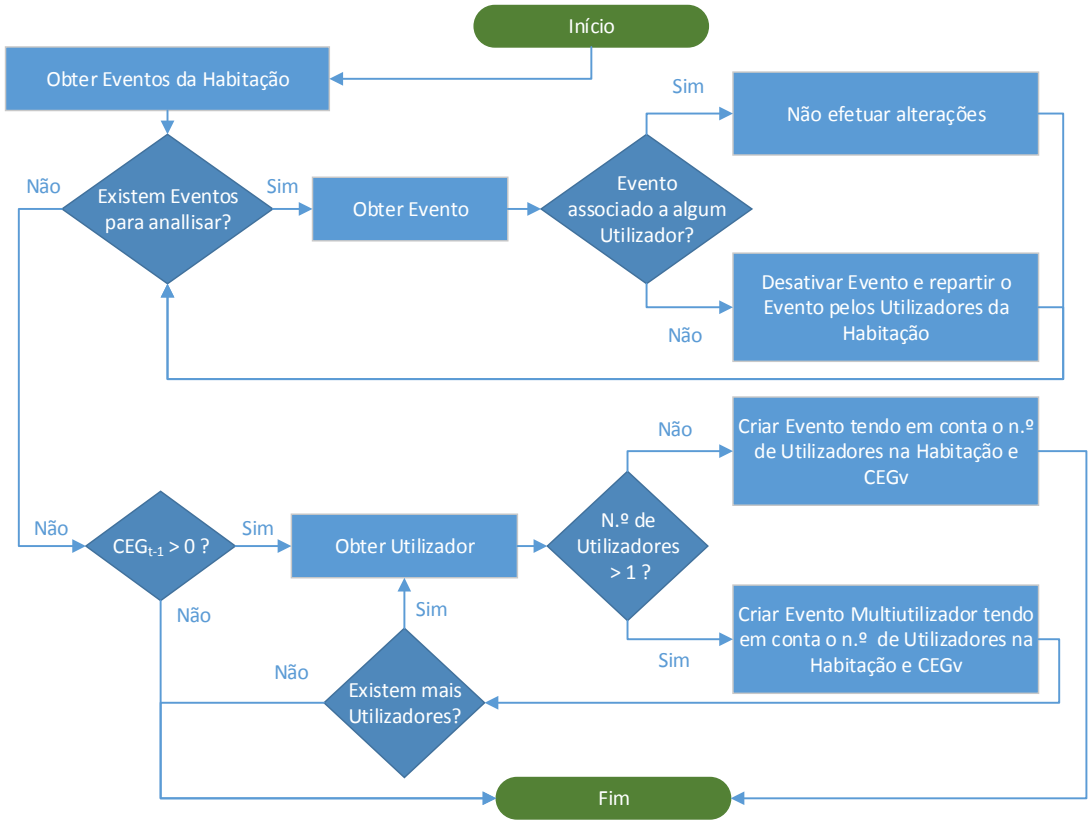


Figura 4.76 - Diagrama de atividade do estado "Utilizadores entraram na Habitação".

Após a verificação e execução do estado de espera, é necessário distribuir o consumo energético pelos utilizadores do HUEPS tendo em conta a sua movimentação e o valor de CEGv. Um valor de CEGv nulo indica que não ocorreu nenhuma interação entre os utilizadores e a rede elétrica. Neste caso apenas se verifica se os utilizadores se deslocaram entre divisões ou se saíram do sistema. Estas movimentações podem obrigar à realocação de eventos energéticos, de modo a serem cumpridas as premissas 2, 4, 7, 8, 9 e 10. A lógica que sustenta este estado encontra-se implementada segundo o diagrama de atividade da figura 4.77. O código que permite a sua execução é disponibilizado pelo anexo P.

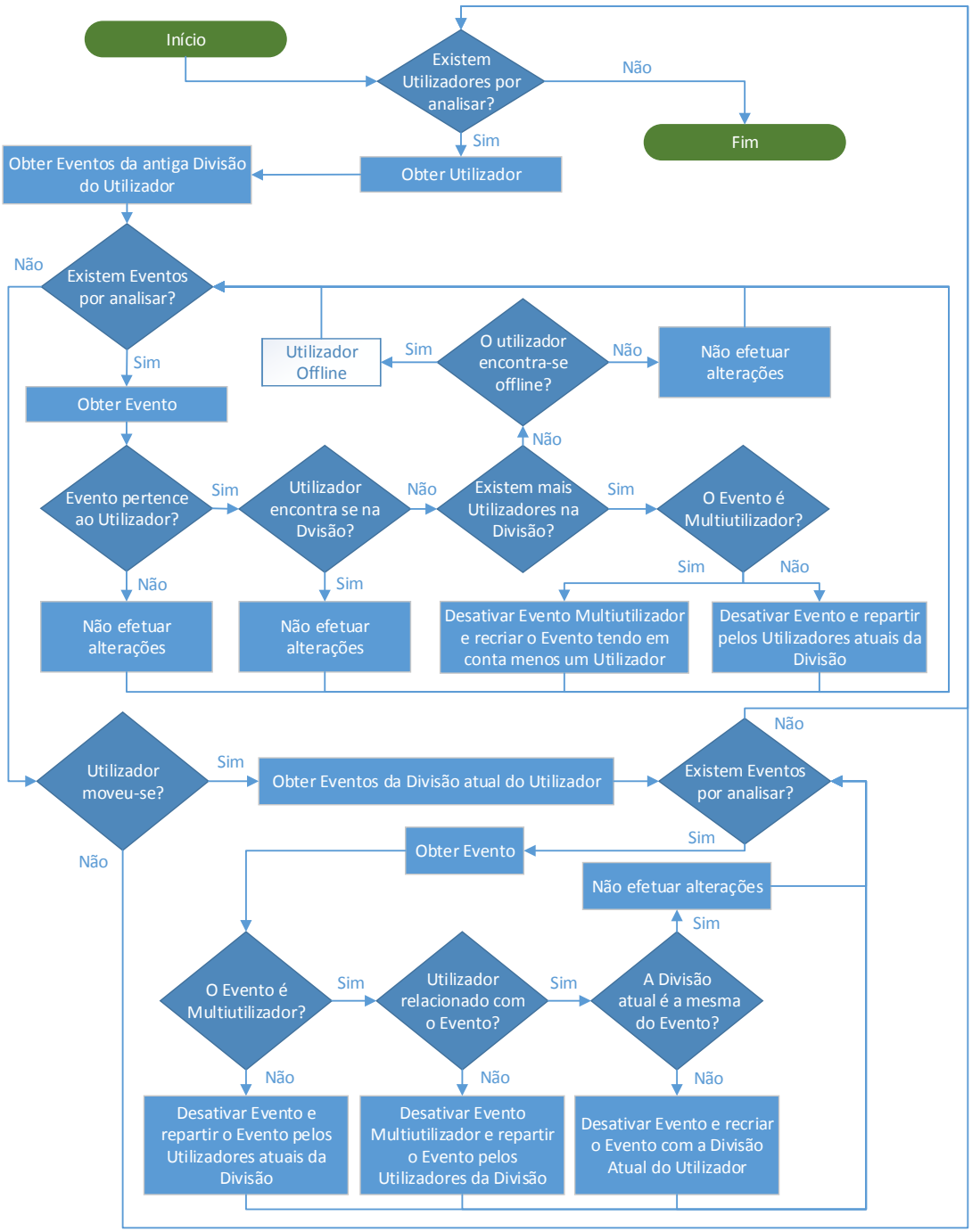


Figura 4.77 - Diagrama de atividade do estado "Atualização de Eventos devido à Movimentação de Utilizadores".

Como se pode observar, este estado percorre todos os utilizadores detetados dentro da habitação. O diagrama também pode ser divido em duas partes, uma que analisa os eventos da antiga divisão do utilizador e outra que verifica os eventos da divisão atual, caso o utilizador se tenha deslocado.

A análise dos eventos da divisão anterior incide apenas nos eventos pertences ao utilizador selecionado. Caso o utilizador se encontre dentro da mesma divisão do evento, não se efetua nenhuma

alteração porque a pessoa responsável por aquele evento ainda se encontra presente. Por outro lado, se o utilizador já não se encontra nessa divisão, torna-se necessário averiguar se a presença de mais utilizadores. Se tal se verificar, o consumo da divisão passa a ser da responsabilidade desses utilizadores (Premissa 8). No entanto, se a divisão não possuir ocupantes, tem de se determinar se o utilizador saiu do sistema. Neste contexto é necessário distribuir todos os eventos a ele associados pelos restantes utilizadores da habitação (figura 4.78). Contudo, este pode apenas se ter deslocado de uma divisão para outra sem desligar qualquer aparelho. Desta forma fica responsável pelo consumo existente na divisão abandonada (Premissa 4 ou 10).

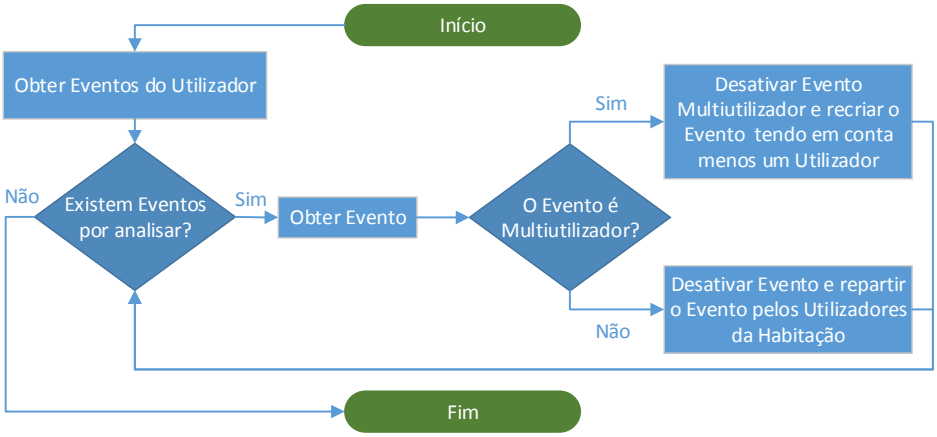


Figura 4.78 - Diagrama de atividade do processo "Utilizadores *Offline*".

A análise dos eventos da divisão atual só ocorre caso o utilizador possua movimento. A estacionariedade do utilizador significa que a divisão atual é a mesma que a divisão anterior, previamente analisada pela lógica supracitada. Para cada evento da divisão atual verifica-se a sua multiplicidade. Um evento pertencente apenas a um utilizador é facilmente distribuído por todos os utilizadores presentes na divisão (Premissa 7 ou 9). Por outro lado, um evento multiutilizador pode estar, ou não, relacionado com o utilizador em análise. Se o utilizador não se relacionar com o evento, efetua-se a sua repartição pelos ocupantes da divisão (Premissa 7 ou 9), senão é necessário deduzir se a divisão do evento coincide com a posição atual do utilizador. Se esta condição for verdadeira, então este evento já foi processado e não se devem efetuar alterações. O contrário implica que o utilizador encontrava-se presente noutra divisão quando ocorreu o evento multiutilizador. Desta forma atualiza-se somente a divisão da parcela do evento associado ao utilizador, uma vez que o evento multiutilizador ainda se encontra ativo.

O estado "Atualização de Eventos devido à Movimentação de Utilizadores" serve apenas para gerir os eventos energéticos dos utilizadores da habitação quando o valor de CEGv é nulo. A sua utilidade implica a existência de eventos energéticos armazenados no HUEPS. Para efetuar a criação

de eventos, o estado "Criação e Atualização de Eventos" executa a lógica do diagrama de atividade da figura 4.79. Este estado é alcançado caso o valor CEGv seja positivo, indicando que ocorreu um conjunto de variações energéticas que resultou num aumento de consumo energético na habitação. Desta forma, é necessário determinar qual a divisão onde poderá ter acontecido esse aumento e quais os utilizadores responsáveis, de modo a repartir o consumo pelos respetivos intervenientes.

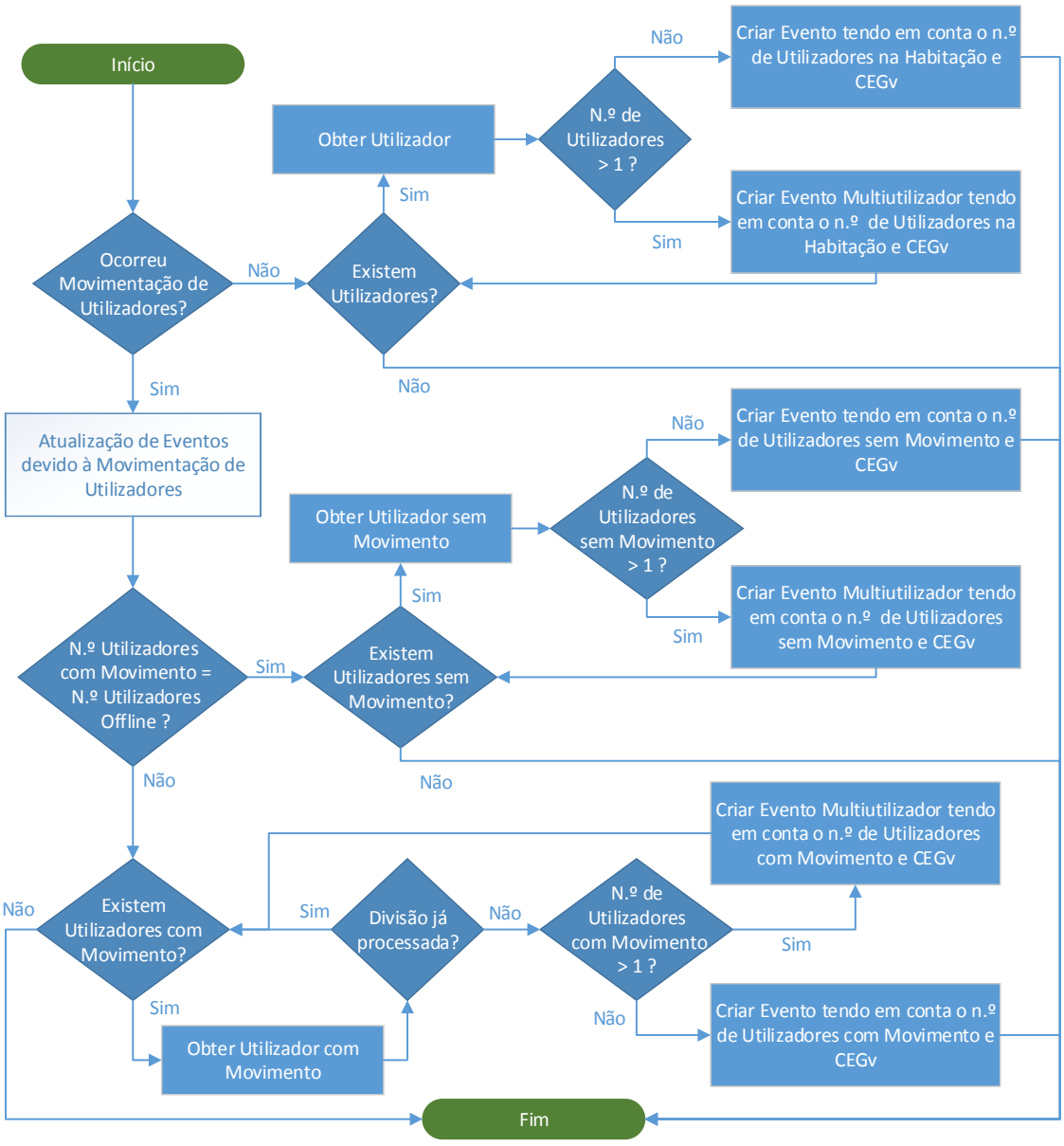


Figura 4.79 - Diagrama de atividade do estado "Criação e Atualização de Eventos".

A primeira condição verificada é a existência de movimentação de utilizadores. Se nenhum utilizador se moveu, então não existe nenhuma preferência por utilizador e/ou divisão. Desta forma cria-se um evento, multiutilizador ou não, recorrendo ao número de utilizadores ativos para distribuir equitativamente o consumo verificado pelas posições atuais dos utilizadores (premissa 6). Por outro

lado, se ocorreu movimentação de algum utilizador, assume-se que estes foram responsáveis pela variação do consumo, como indicado pela premissa 5. No entanto, antes de se atribuir o consumo a estes utilizadores, aplica-se o estado "Atualização de Eventos devido à Movimentação de Utilizadores". Deste modo, os eventos energéticos ativos já se encontram organizados de acordo com as posições atuais dos utilizadores e pode-se proceder à criação de novos eventos.

A criação de novos eventos incide apenas sobre os utilizadores com movimentação. Este grupo pode ser constituído por utilizadores que se descolocaram de uma divisão para outra ou para fora da habitação, ficando *offline*. Se o número de utilizadores com movimento e *offline* for igual, está-se perante a situação "todos os utilizadores que se moveram saíram da habitação". Assim, a distribuição do valor de CEGv não afeta estes utilizadores, mas sim aqueles que ainda se encontram dentro da habitação e não efetuaram qualquer deslocamento. Porém, se existir pelo menos um utilizador ativo e com movimento, o novo consumo é repartido por eles. Durante este processo de repartição, verifica-se se uma divisão específica já foi processada. Esta validação permite evitar a duplicação de eventos para diferentes utilizadores dentro da mesma divisão.

Por fim, se o valor de CEGv for negativo, é necessário determinar quais os eventos ativos (ou evento) mais semelhantes à variação de consumo energética ocorrida, de forma a desativá-los ou atualizá-los. Estas operações são implementadas através do estado "Desativação e Atualização de Eventos" representado pelo diagrama da figura 4.80.

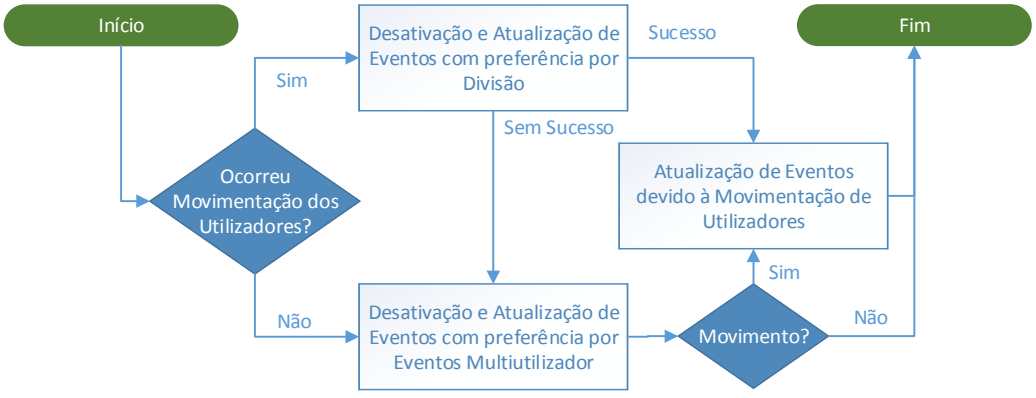


Figura 4.80 - Diagrama de atividade do estado "Desativação e Atualização de Eventos".

Mais uma vez, a existência de movimentação de utilizadores é utilizada para selecionar a melhor abordagem a adotar. Esta permite obter as divisões onde se verificou deslocamento, que podem ser usadas para filtrar o conjunto inicial de eventos mais verossemelhantes com CEGv. Este processo, representado na figura 4.80 como "Desativação e Atualização de Eventos com preferência por Divisão", obedece ao diagrama de atividades da figura 4.81.

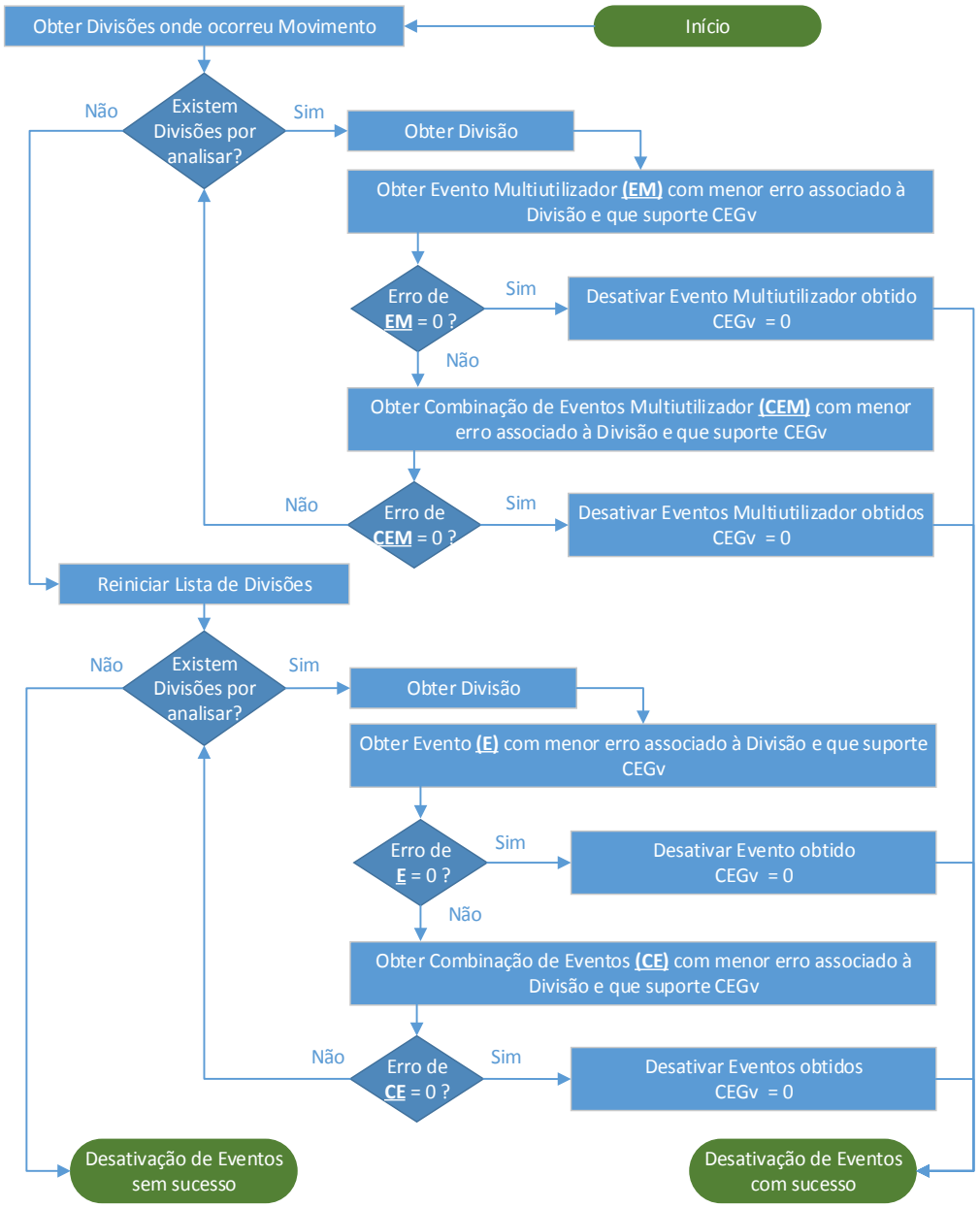


Figura 4.81 - Diagrama de atividade do estado "Desativação e Atualização de Eventos com preferência por Divisão".

Inicialmente dá-se preferência aos eventos multiutilizador de uma determinada divisão, originados nas mesmas condições e com um determinado erro inerente (teoricamente). A sua desativação é preferível face aos eventos criados sob condições mais restritas, devido à maior probabilidade destes eventos possuírem maior correlação com a variação que os originou. Se nenhum evento multiutilizador preencher os requisitos, verifica-se se existe uma combinação de eventos multiutilizador cuja soma de consumo energético coincida com o valor de CEGv. Para tal aplica-se a teoria da resolução do problema da soma dos subconjuntos, através da classe "*calculateCombinationsToReachSum.cs*", referida no subcapítulo 4.2.2.2. Se mesmo assim não se

conseguir obter uma correspondência, é efetuada novamente uma pesquisa mas restringindo a procura a eventos não-multiutilizador da divisão.

A determinação da melhor correspondência obedece a dois parâmetros. Em primeiro lugar, o valor de CEGv tem de ser suportado na totalidade pelo consumo energético do(s) evento(s) (CEE) (4.2). Se esta condição puder ser satisfeita por diversos eventos ou diferentes combinações de eventos, seleciona-se aquele que possui o menor valor de erro relativo (4.3). Ao serem cumpridas ambas as condições, o resultado assume automaticamente um valor de erro nulo, para facilitar o seu processamento ao longo da execução do algoritmo.

$$CEE - CEG_{tol} < CEGv < CEE + CEG_{tol} \quad (4.2) \qquad \delta_{relativo} = \frac{|CEE - CEGv|}{CEGv} \quad (4.3)$$

Por outro lado, a ausência de movimento implica o desconhecimento total de onde poderá ter ocorrido a variação e quem a causou. Desta forma não é necessário filtrar os eventos por divisão e apenas atribui-se uma preferência inicial por eventos multiutilizador. Este processo denomina-se "Desativação e Atualização de Eventos com preferência por Eventos Multiutilizador". Ao contrário do processo "Desativação e Atualização de Eventos com preferência por Divisão", este processo efetua a desativação e/ou atualização de eventos mesmo que não se obtenha uma correspondência perfeita. Não se deve esquecer que ocorreu uma diminuição de consumo energético na habitação, e essa variação tem de se refletir no HUEPS de modo a garantir a integridade da informação. Deste modo, após a execução de "Desativação e Atualização de Eventos com preferência por Divisão", verifica-se se a operação foi realizada com sucesso. Caso contrário, recorre-se a "Desativação e Atualização de Eventos com preferência por Eventos Multiutilizador" para concluir a tarefa. A figura 4.82 ilustra o diagrama de atividade referente a este processo.

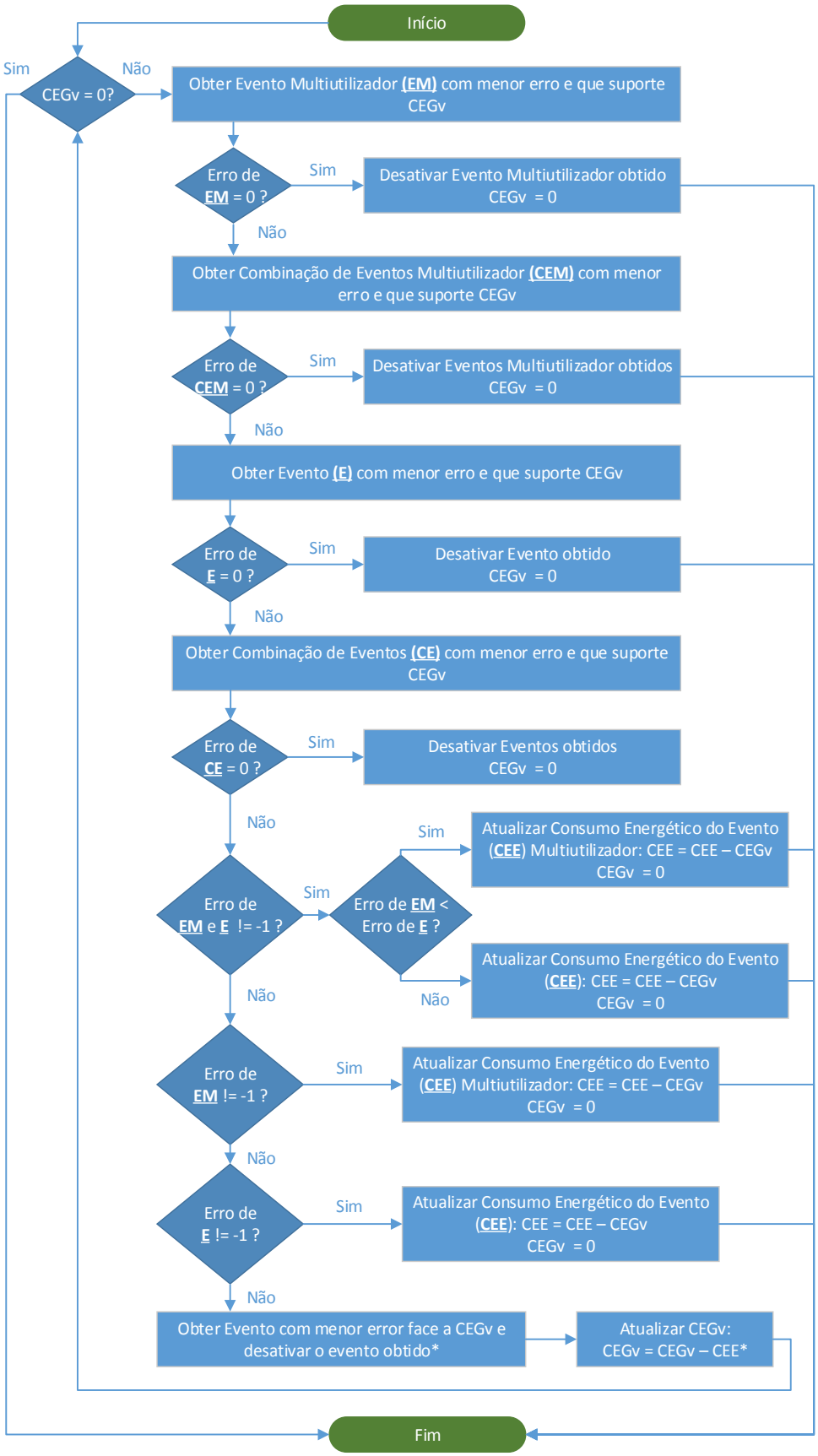


Figura 4.82 - Diagrama de atividade do processo "Desativação e Atualização de Eventos com preferência por Eventos Multiutilizador".

Tal como em "Desativação e Atualização de Eventos com preferência por Divisão", primeiro verifica-se se existe um evento multiutilizador, uma combinação de eventos multiutilizador, um evento simples ou uma combinação de eventos simples que cumpra as condições (4.2) e (4.3). Desta vez pretende-se encontrar todos os eventos possíveis que suportem CEGv, e não apenas correspondências "perfeitas". No entanto, caso se encontre uma correspondência, o evento é imediatamente desativado e o processo termina. Se apenas se obtiveram eventos possíveis, tem de se analisar qual deles é que possui menor erro face ao valor de CEGv, de modo a proceder-se à atualização do seu valor de CEE. O pior caso ocorre quando não são devolvidos quaisquer casos possíveis.

$$CEE - CEG_{tol} < CEGv \quad (4.3)$$

O não retorno de casos possíveis indica que não existe nenhum evento que suporte o valor de CEGv. Desta forma, é necessário reduzir o valor de CEGv gradualmente de acordo com os eventos ativos no HUEPS. Efetua-se uma procura excluindo as combinações de eventos e recorrendo somente à condição (4.2) para se obter o evento mais próximo. O valor de CEE obtido é subtraído ao valor de CEGv, que é utilizado na próxima iteração do processo. Este ciclo repete-se até que o valor de CEGv seja nulo.

O reajuste de eventos de acordo com a movimentação dos utilizadores, caso esta tenha ocorrido, só é realizado após a desativação/atualização de eventos. Esta solução permite associar a redução do consumo energético aos eventos existentes antes de um determinado utilizador sair da respetiva divisão. Note-se que é mais provável um utilizador desligar um aparelho enquanto abandona uma divisão do que durante a entrada.

Capítulo 5: Validação e Resultados Experimentais

Neste capítulo sujeita-se o WiLOS e o HUEPS a diferentes casos de estudo, de modo a testar e validar cada uma das aplicações desenvolvidas. Numa primeira fase, ambos os sistemas são analisados separadamente para determinar o seu desempenho. A análise inicial incide sobre o WiLOS uma vez que este implementa os serviços de localização que são utilizados para garantir o funcionamento do HUEPS.

A validação do WiLOS é executada em várias etapas. Como se trata de um sistema de localização dependente de uma rede sem fios, primeiramente efetua-se o estudo do comportamento do WiLOS utilizando diferentes configurações para a rede. Ao variar o número de *APs* e a sua distribuição, consegue-se inferir a configuração que origina uma precisão mais elevada, garantindo a cobertura total da zona de implementação. Seguidamente analisa-se o desempenho do algoritmo k-*NNS* através da variação do número de vizinhos 'k'. O objetivo é determinar o menor valor de 'k' que alcance uma boa precisão sem comprometer a eficiência do algoritmo (maior número de vizinhos, implica maior poder computacional). Por fim, após o estabelecimento da configuração da rede e do valor de 'k', testam-se os diferentes algoritmos desenvolvidos para verificar se existe uma melhoria no desempenho do WiLOS.

O estudo do desempenho do HUEPS, de forma independente do WiLOS, é realizado através de um simulador. Este simulador é utilizado para garantir o fornecimento de um serviço de localização sem erros. De outro modo, qualquer erro externo ao HUEPS, poderia comprometer os resultados recolhidos durante os testes. A validação do HUEPS passa pela interação de vários utilizadores com a "habitação" de forma controlada. Desta forma consegue-se determinar se o HUEPS está a comportar-se de acordo com o modelo comportamental do subcapítulo 4.2.2.4 e extrair o erro resultante da distribuição das variações energéticas pelos utilizadores, e respetivas divisões.

5.1 Análise Experimental do WiLOS

5.1.1 Influência da configuração da rede sem fios

Para determinar a melhor configuração da rede sem fios, foi necessário definir uma zona de implementação capaz de simular uma habitação. Desta forma, o ambiente de simulação do WiLOS foi implementado no 1º andar do Departamento de Engenharia Eletrotécnica da Faculdade de Ciências e Tecnologias, Universidade Nova de Lisboa. A área utilizada, aproximadamente 133 m^2, permite simular uma habitação com 5 divisões.

Uma vez que o WiLOS recorre ao método de impressão digital para efetuar a localização dos utilizadores, é necessário obter um mapa de calibração para o sistema. Esta tarefa é facilitada pelo WiLOS, que oferece mecanismos para a aquisição deste mapa. Sendo assim, de acordo com a tipologia do cenário e com os parâmetros introduzidos (*Wall Distance* = 0.2 m*; Space Between Points*: 1.85 m e *Image Processing Configuration* = Valores Pré-Definidos), obtém-se o mapa de calibração dado pela figura 5.1.

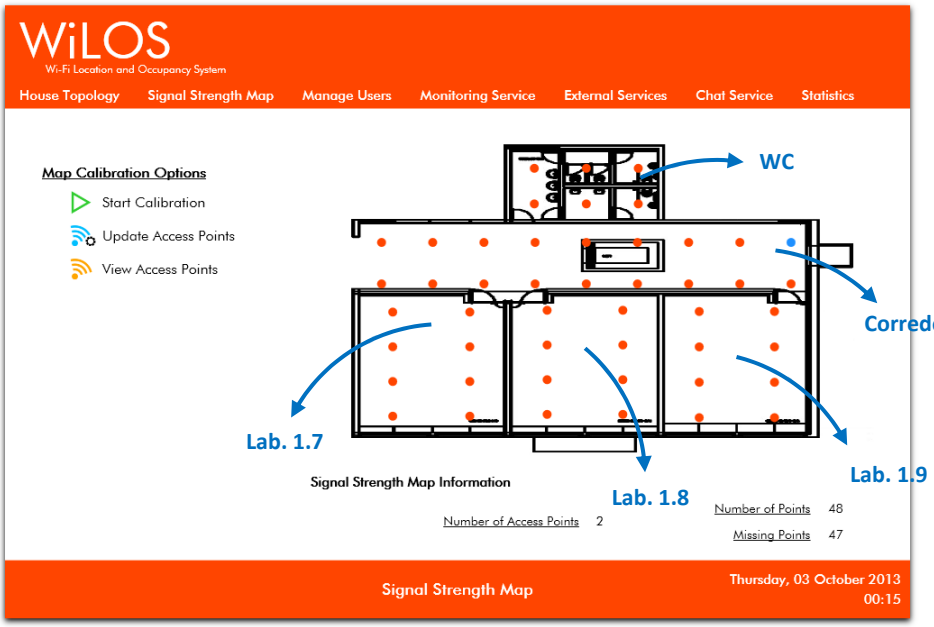


Figura 5.1 - Visualização da tipologia da zona de implementação e dos 48 pontos que constituem o mapa de calibração.

A influência da configuração da rede é determinada a partir da análise de 5 cenários possíveis. Para cada cenário efetuou-se a calibração do sistema através da recolha de 10 valores de *RSSI* referentes aos *APs* mais próximos, nas 4 direções Norte, Sul, Este e Oeste, para cada ponto de calibração (um total de 768 valores recolhidos). Este processo é realizado duas vezes, uma com o telemóvel na mão, outra com o telemóvel no bolso. O objetivo é a obtenção de 2 mapas de calibração, de modo a comparar não só a influência da distribuição dos *APs*, como também a orientação do próprio

dispositivo móvel. Após a aquisição de todos os mapas de potência do sinal, a verificação do desempenho do WILOS é realizada através da execução do percurso de testes definido na figura 5.2.

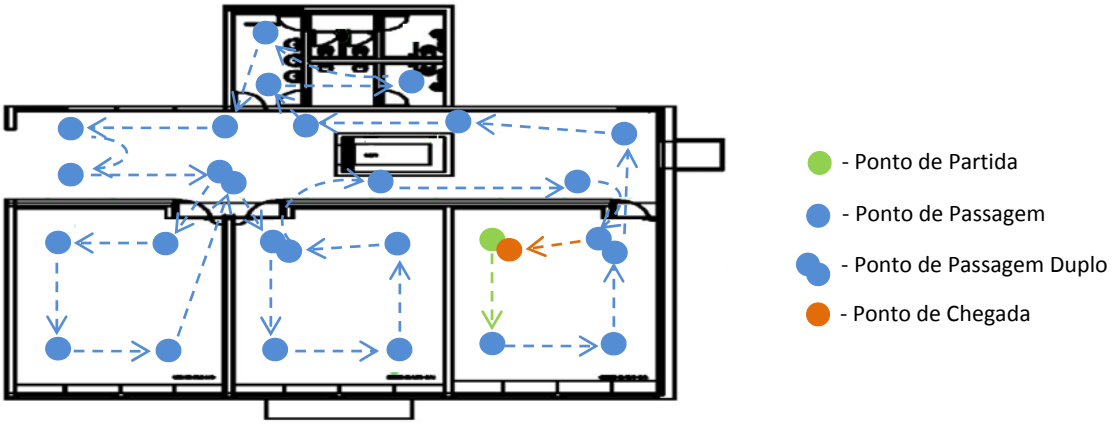


Figura 5.2 - Percurso de testes utilizado para analisar o desempenho do WiLOS em tempo real.

O utilizador caminha ao longo do percurso, parando em cada ponto durante 15 s. Como o serviço de monitoração encontra-se configurado com um valor de 'k' igual a 4 e um tempo de amostragem de 3 s, isto garante a recolha de 5 amostras em cada ponto do percurso. Os dados recolhidos durante a movimentação do utilizador entre pontos também são usados para determinar a precisão do WiLOS. Este percurso é repetido 3 vezes para cada cenário, de modo a garantir que os dados obtidos são consistentes e não apenas resultados do mero acaso. Nesta fase apenas se recorrem aos mapas de potência obtidos com o dispositivo móvel na mão. Os resultados destes ensaios são apresentados nas tabelas 5.1 - 5.6. A figura 5.3 ilustra os 5 cenários em estudo.

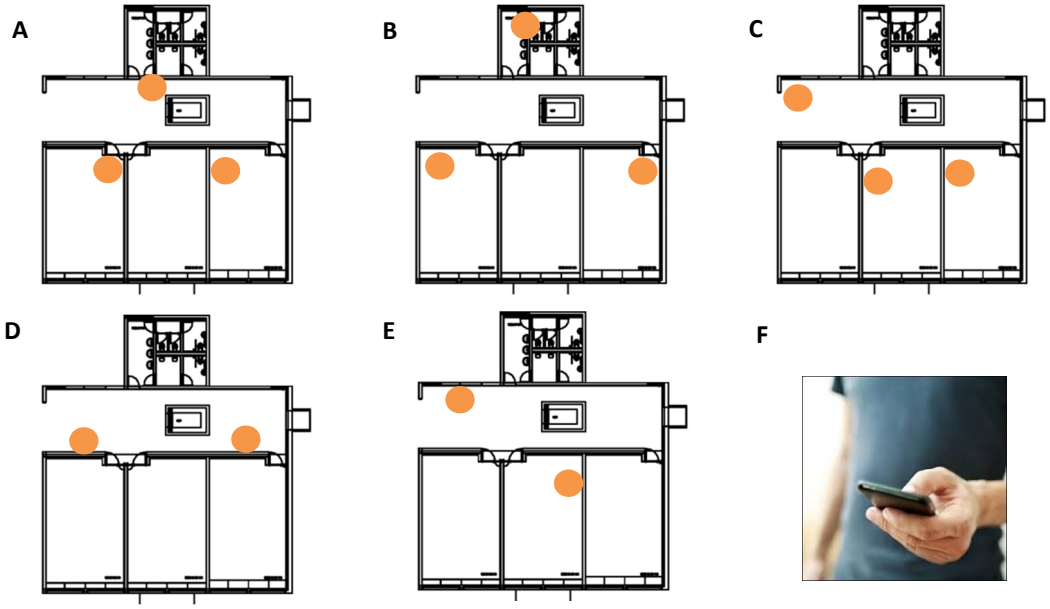


Figura 5.3 - (**A**)-(**E**): Cenários de estudo da influência da configuração da rede na precisão do WiLOS, onde (**A**)-(**C**) são configurações com 3 *APs* e (**D**)-(**E**) com 2. (**F**): Orientação do dispositivo móvel horizontal ou "na mão".

Divisão	*Pontos Recolhidos*	*Precisão Média [%]*	*Desvio Padrão [%]*
Lab. 1.9	152	94.79	2.02
Lab. 1.8	139	69.98	9.09
Lab. 1.7	99	77.61	10.45
WC	75	60.55	11.58
Corredor	219	58.99	10.40
Total	**684**	**72.11**	**5.90**

Tabela 5.1 - Precisão do WiLOS no cenário A. Resultados dos ensaios realizados com o dispositivo na mão.

Divisão	*Pontos Recolhidos*	*Precisão Média [%]*	*Desvio Padrão [%]*
Lab. 1.9	147	88.50	3.71
Lab. 1.8	130	73.73	6.14
Lab. 1.7	96	63.91	11.12
WC	73	55.75	14.12
Corredor	214	75.88	9.10
Total	**660**	**74.18**	**4.68**

Tabela 5.2 - Precisão do WiLOS no cenário B. Resultados dos ensaios realizados com o dispositivo na mão.

Divisão	*Pontos Recolhidos*	*Precisão Média [%]*	*Desvio Padrão [%]*
Lab. 1.9	145	95.17	1.23
Lab. 1.8	120	95.00	6.61
Lab. 1.7	100	77.82	12.82
WC	78	80.76	13.32
Corredor	202	87.69	6.44
Total	**645**	**88.37**	**4.84**

Tabela 5.3 - Precisão do WiLOS no cenário C. Resultados dos ensaios realizados com o dispositivo na mão.

Divisão	*Pontos Recolhidos*	*Precisão Média [%]*	*Desvio Padrão [%]*
Lab. 1.9	161	90.90	7.17
Lab. 1.8	131	38.20	4.93
Lab. 1.7	98	60.09	5.58
WC	74	46.15	17.63
Corredor	232	82.28	3.48
Total	**696**	**68.93**	**1.83**

Tabela 5.4 - Precisão do WiLOS no cenário D. Resultados dos ensaios realizados com o dispositivo na mão.

Divisão	*Pontos Recolhidos*	*Precisão Média [%]*	*Desvio Padrão [%]*
Lab. 1.9	152	53.96	13.84
Lab. 1.8	137	80.66	7.93
Lab. 1.7	109	48.86	17.26
WC	77	64.90	3.76
Corredor	215	85.14	1.83
Total	**690**	**69.18**	**3.03**

Tabela 5.5 - Precisão do WiLOS no cenário E. Resultados dos ensaios realizados com o dispositivo na mão.

Cenário	*Pontos Recolhidos*	*Precisão Média [%]*	*Desvio Padrão [%]*
A	684	72.11	5.90
B	660	74.18	4.68
C	645	88.37	4.84
D	696	68.93	1.83
E	690	69.18	3.03

Tabela 5.6 - Comparação da Precisão do WiLOS nos cenários em estudo.

Pela análise dos resultados, verifica-se que o cenário que melhor aproveita o potencial do WiLOS é o cenário "C", garantindo uma precisão média de 88.37 %. Como esperado, as configurações de rede compostas por 3 *APs* apresentam melhores resultados do que os cenários constituídos somente por 2. A distribuição de 2 *APs* em habitações com várias divisões (> 3), não permite a criação de mapas de potência do sinal com detalhe suficiente. Existem muitas zonas da área de implementação que partilham os mesmos valores de *RSSI*, no que se traduz na incapacidade de identificar inequivocamente todos os pontos que definem o mapa de calibração. Mesmo assim, os cenários "D" e "C" conseguem determinar a localização do utilizador com uma precisão superior a 65 %.

Relativamente às configurações com 3 *APs*, os cenários "A" e "B" possuem os seus *APs* distribuídos de forma simétrica, através de uma disposição em triângulo. A existência de um 3º *AP* contribui para

o aumento do desempenho do sistema, mas este nem sempre é superior a 75 %. A configuração do cenário "A" dispõe os *APs* de forma assimétrica (Figura 5.3), proporcionando um desempenho que excedeu as expectativas, com uma precisão nunca inferior a 85 %. A determinação se a simetria foi, de facto, um fator chave para o aumento do desempenho, implicaria a realização de mais testes com diferentes configurações de 3 *APs*. Apenas com estes ensaios não se possui informação suficiente para inferir um motivo válido, podendo este aumento estar relacionado com a própria tipologia da habitação, ou com a qualidade dos *APs* utilizados para definir a rede WiLOS.

A partir deste ponto, todos os futuros ensaios referentes ao desempenho do WiLOS, e HUEPS, são realizados através da configuração da rede dada pelo cenário "C".

5.1.2 Determinação do valor de 'k' para o algoritmo k-*NNS*

Após a determinação da topologia da rede que oferece melhor precisão durante o processo de monitoração, verifica-se a influência do valor de 'k', variável do algoritmo k-*NNS*, no desempenho do sistema. No capítulo 3, vários autores estudaram a influência deste valor relativamente à precisão ao nível do ponto de calibração. Para o WiLOS apenas se deseja uma precisão ao nível da divisão. Desta forma, confirma-se se o desempenho dos sistemas existentes garantem resultados semelhantes perante as condições apresentadas.

Para este estudo volta-se a replicar as condições de teste para a determinação da configuração da rede. Deste modo, recorre-se ao percurso da figura 5.2 para se realizarem 5 ensaios para cada valor de 'k' analisado, com um tempo de amostragem de 3 s. A variação do valor de 'k' ao longo dos ensaios permite verificar o desempenho do algoritmo k-*NNS* quando se utiliza uma quantidade diferente de vizinhos durante o processo de comparação. A realização destes ensaios originou um total de 7478 amostras. Os resultados são apresentados na figura 5.4.

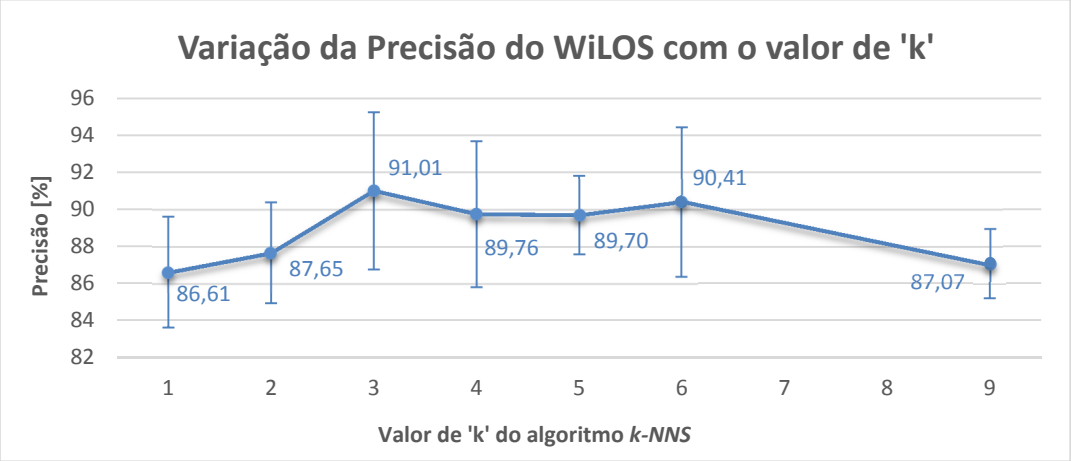


Figura 5.4 - Variação da Precisão do algoritmo k-*NNS* no WiLOS, devido à variação do valor de 'k'.

O algoritmo k-*NNS* implica sempre algum tipo de compromisso. Quando o número de vizinhos utilizado é demasiado elevado, o processamento computacional aumenta e começam a surgir erros relacionados com "maus vizinhos", diminuindo o desempenho do sistema. Por outro lado, a utilização de poucos vizinhos pode não garantir que o valor determinado, neste caso a posição, seja correto. De acordo com a Figura 5.4, um desempenho ligeiramente superior obtém-se quando o valor de 'k' é igual a 3 (91.01 %). No entanto, uma precisão semelhante é alcançada quando se aplicam 4, 5 ou 6 vizinhos ao algoritmo (≈ 90 %). Uma queda no desempenho ocorre para os valores 1 e 2, e para valores de 'k' muito elevados, como o valor 9 (≈ 87 %). A informação recolhida reforça o conceito de compromisso imposto pelo algoritmo k-*NNS*, referente à influência do número de vizinhos na precisão do sistema. Os resultados obtidos são similares àqueles verificados em [53,57].

Como se pode constatar, a maioria dos ensaios demonstra um desvio padrão considerável (≈ 4 %). Estes ensaios foram efetuados em tempo real, com a replicação das mesmas condições e as mesmas orientações do utilizador em cada ponto de passagem do percurso de teste. O dispositivo móvel manteve-se na mão do utilizador numa posição horizontal, a mesma posição utilizada para calibrar o WiLOS. Todos estes cuidados foram tomados para minimizar a influência de fatores externos, que poderiam comprometer a análise dos resultados. Desta forma, os elementos que originam este desvio têm de coexistir com o WiLOS, ao longo da sua zona de implementação. Este erro pode ser justificado pela variabilidade do sinal de potência emitido pelos *APs*. O sinal transmitido sofre reflexão, difração e perdas por propagação, causadas pela infraestrutura do edifício. Estes fenómenos acabam por introduzir um erro aleatório no WiLOS. Os ensaios mostram que na maioria dos casos a precisão é bastante boa, mas nem sempre isso se verifica.

Outro motivo para a origem deste desvio pode estar relacionado com a movimentação dos utilizadores ao longo da zona de implementação. O corpo humano é maioritariamente composto por água, sendo capaz de atenuar o sinal durante a sua propagação. Este efeito torna-se mais evidente caso a pessoa se encontre diretamente entre o dispositivo móvel e o *AP*. Para além disso, a rede WiLOS não é a única infraestrutura sem fios existente no edifício. Outras redes compostas por outros *APs*, ou a presença de dispositivos que partilham a mesma largura de banda que a norma 802.11 (e.g., teclados sem fios e dispositivos Bluetooth), podem causar flutuações e interferências no WiLOS, contribuindo para o valor do desvio padrão verificado durante os testes.

5.1.3 Análise dos algoritmos desenvolvidos

A validação dos algoritmos desenvolvidos implica a existência de um ponto de partida. Neste caso, o ponto de partida será o algoritmo base do WiLOS, o algoritmo k-*NNS*. O sujeito de testes realiza 10 vezes o percurso da figura 5.2, 5 vezes com o dispositivo móvel na mão e 5 com o dispositivo no bolso.

Uma vez que o WiLOS está configurado para utilizar somente os mapas de potência adquiridos com o dispositivo móvel na mão, espera-se que a orientação vertical/horizontal do dispositivo influencie a precisão do sistema. Os resultados destes ensaios podem ser consultados na tabela 5.7 e na figura 5.5.

Algoritmo	*Orientação do Dispositivo*	*Mínimo [%]*	*Máximo [%]*	*Precisão Média [%]*	*Precisão Média s/ Extremos [%]*	*Desvio Padrão [%]*
k-*NNS*	Mão	87.20	97.27	**91.01**	90.20	4.24
	Bolso	81.61	85.80	**84.06**	84.29	1.58

Tabela 5.7 - Sumário estatístico dos dados recolhidos do WiLOS utilizando o algoritmo k-*NNS*.

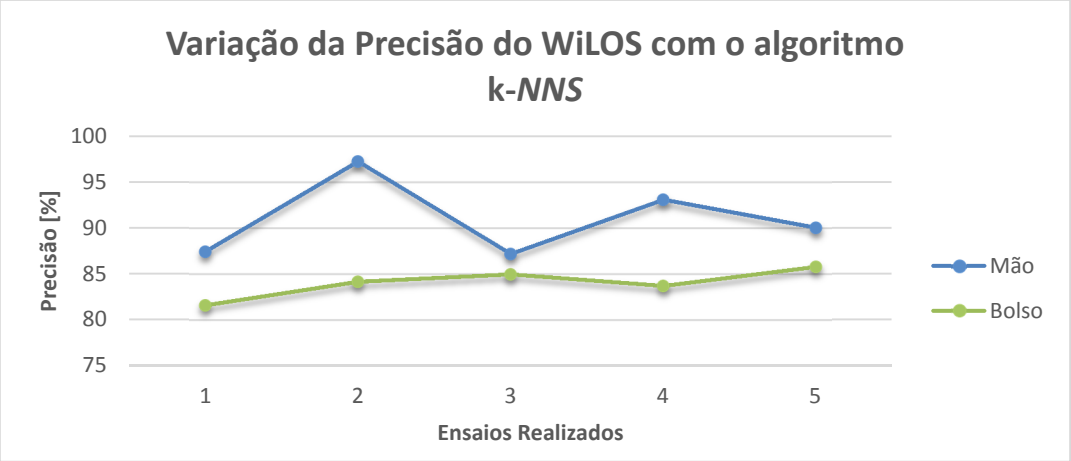


Figura 5.5 - Variação da Precisão do WiLOS através do algoritmo k-*NNS*.

Como esperado, caso o *smartphone* esteja dentro do bolso do utilizador numa posição vertical, a precisão do WiLOS decresce ligeiramente (≈ 6 %). No entanto, se se tiver em conta o desvio padrão associado aos ensaios, este valor não se torna muito relevante. Se a queda de desempenho fosse mais acentuada, o valor não seria de estranhar. Note-se que os mapas de potência utilizados durante estes ensaios foram adquiridos com o dispositivo numa posição horizontal, isto é na mão. Desta forma, as amostras recolhidas em tempo real não correspondem exatamente às posições definidas no WiLOS durante a calibração.

A solução implementada para atenuar este efeito consiste em indicar ao WiLOS a orientação do dispositivo móvel sempre que é efetuado um pedido de localização. Esta informação é obtida através do acelerómetro do *smartphone* e permite ao WiLOS alternar entre todos os mapas de potência obtidos, mão e bolso, de modo a aplicar o algoritmo k-*NNS* ao mapa mais adequado. Os dados relativos a estes ensaios, algoritmo k-*NNS* com compensação de orientação, encontram-se representados na tabela 5.8 e na figura 5.6.

Algoritmo	*Orientação do Dispositivo*	*Mínimo [%]*	*Máximo [%]*	*Precisão Média [%]*	*Precisão Média s/ Extremos [%]*	*Desvio Padrão [%]*
k-*NNS* + Orientação	Mão	85.31	89.96	**87.88**	88.05	1.68
	Bolso	79.52	89.84	**84.58**	84.52	4.15

Tabela 5.8 - Sumário estatístico dos dados recolhidos do WiLOS utilizando o algoritmo k-*NNS* com compensação de orientação.

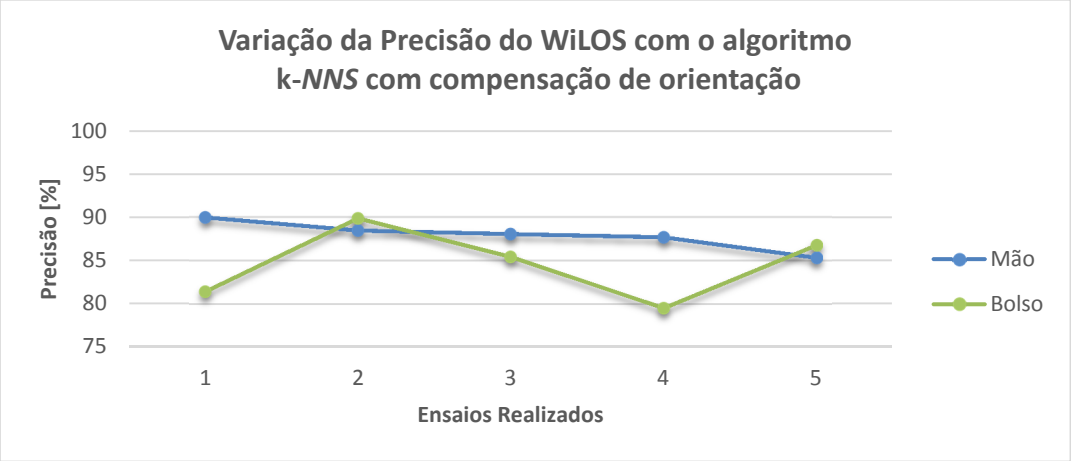


Figura 5.6 - Variação da precisão do WiLOS através do algoritmo k-*NNS* com compensação de orientação.

A análise dos dados permite verificar que não ocorreu nenhuma melhoria significativa da precisão do WiLOS nos ensaios com o dispositivo no bolso. Por outro lado, visualiza-se uma queda no desempenho relativamente aos ensaios efetuados com o *smartphone* na mão (≈ 4%). Através do WiLOS confirmou-se que o acelerómetro devolvia resultados precisos acerca da orientação do dispositivo móvel. O WiLOS também utilizou toda a informação de forma conveniente, recorrendo aos mapas corretos para determinar a posição do utilizador. Desta forma, a queda de desempenho verificada só é justificada pela variabilidade do sinal. A indiferença existente entre a utilização de mapas de potência "horizontais" ou "verticais", para catalogar os valores de *RSSI* nos ensaios com o dispositivo no bolso, podem estar relacionados com os diferentes ganhos direcionais da antena do dispositivo móvel. Este, quando colocado muito próximo do utilizador, causa uma queda no ganho considerável, o que pode originar a aquisição de mapas de potência "verticais" com menor detalhe e que não permitem melhorar a precisão do WiLOS.

Uma análise mais detalhada a todos os ensaios realizados demonstra que os valores de desvio padrão ainda possuem valores significativos (< 5%), tenho em conta que se está a tentar alcançar uma precisão superior a 90 %. Sabe-se que a própria rede Wi-Fi e a zona de implementação são bastante dinâmicas e propensas a interferências, o que confere alguma variabilidade ao WiLOS. Para além disso, mesmo recorrendo a uma rede sem fios composta por 3 *APs*, existem alguns locais da zona de

implementação que possuem valores de *RSSI* muito semelhantes. Isto não seria um problema se este fenómeno apenas ocorresse dentro da mesma divisão. No entanto, uma análise aos mapas de potência disponibilizados pelo WiLOS permite verificar que este problema ocorre em divisões distintas, como ilustrado na figura 5.7.

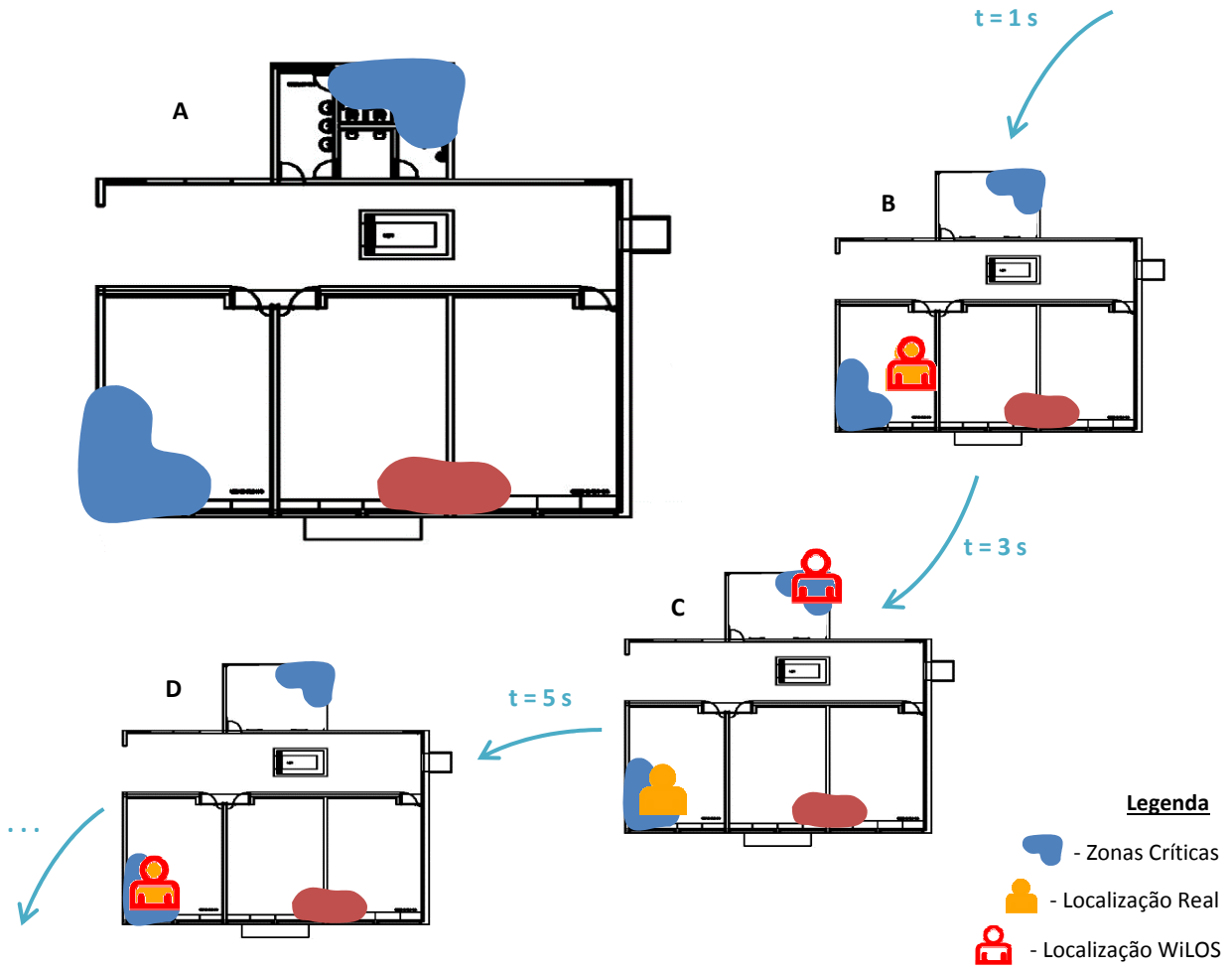


Figura 5.7 - (**A**): Zonas Críticas dos mapas de potência obtidos pelo WiLOS. (**B**), (**C**) e (**D**): Sequência de amostras recolhidas ao longo do tempo e evolução da localização real do utilizador face à localização obtida pelo WiLOS.

A existência de locais com valores de *RSSI* comuns em divisões distintas, denominadas "zonas críticas", origina erros de localização desnecessários. Como se pode visualizar no caso em estudo (figura 5.7), as zonas críticas, para além de estarem estabelecidas em divisões diferentes, encontram-se separadas por uma distância considerável. Deste modo, é pouco provável que um utilizador consiga alternar entre estas zonas durante um período de tempo relativamente curto, definido pelo tempo de amostragem do WiLOS. Para evitar estas situações, mecanismos de seguimento ou de previsão da localização futura do utilizador podem ser implementados para tentar minimizar o problema. No caso do WiLOS, o filtro de distância permite limitar a distância máxima que um utilizador pode percorrer entre dois instantes, de modo a evitar a ocorrência do fenómeno de "teletransporte" entre zonas críticas.

A validação deste filtro é realizada com o acelerómetro desativo, para evitar a introdução de informação extra no WiLOS. O percurso é realizado mais 10 vezes, com o dispositivo na mão e no bolso, de forma a se obterem os resultados da tabela 5.9 e da figura 5.8.

Algoritmo	*Orientação do Dispositivo*	*Mínimo [%]*	*Máximo [%]*	*Precisão Média [%]*	*Precisão Média s/ Extremos [%]*	*Desvio Padrão [%]*
k-*NNS* + Filtro de Distância	Mão	86.03	95.73	90.00	89.40	3.99
	Bolso	76.79	93.80	86.18	86.76	7.05

Tabela 5.9 - Sumário estatístico dos dados recolhidos do WiLOS utilizando o algoritmo k-*NNS* com filtro de distância aplicado.

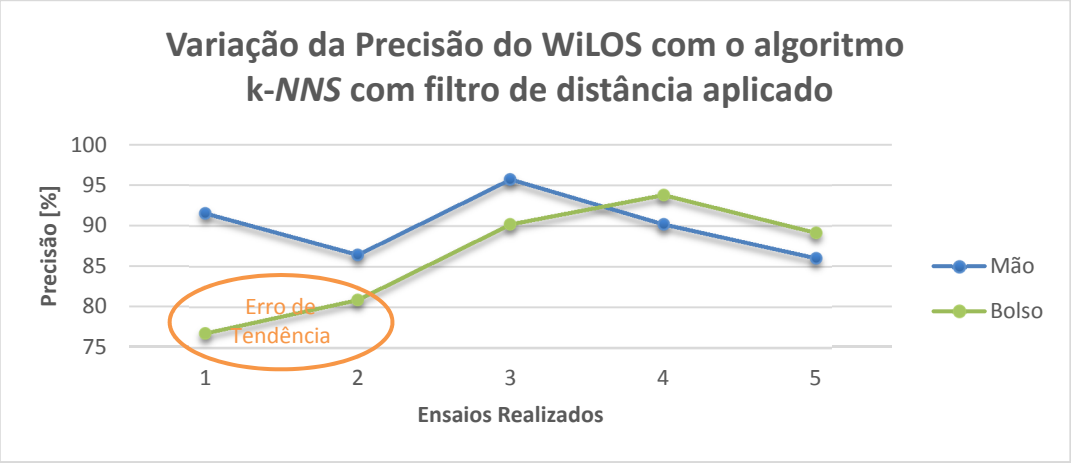


Figura 5.8 - Variação da precisão do WiLOS através do algoritmo k-*NNS* com filtro de distância aplicado..

Os dados referentes aos ensaios realizados com o dispositivo na mão demonstram que o filtro de distância não chegou a influenciar o desempenho do sistema, garantindo um nível de precisão na ordem dos 90 %. Por outro lado, os testes efetuados com o *smartphone* no bolso mostram que a aplicação deste filtro é "uma faca de dois gumes". Os ensaios 3 a 5 demonstram bons resultados, incluindo o melhor ensaio verificado para um teste com o dispositivo no bolso. Isto deve-se ao facto de o filtro conseguir limitar os pontos utilizados pelo algoritmo k-*NNS* e restringir o utilizador a uma certa área de deslocamento, de modo a eliminar o problema das zonas críticas. Porém, esta restrição pode originar um problema ainda mais grave que a variabilidade do sistema, a ocorrência de erros de tendência, como verificado nos ensaios 1 e 2.

Um erro de tendência consiste num erro introduzido num sistema como consequência de atos de favoritismo ou condicionamento segundo um certo padrão. Neste caso, sempre que um utilizador se desloca ao longo da habitação, as posições válidas dependem da posição anterior do utilizador, dos valores de *RSSI* verificados e da distância máxima que este pode percorrer entre dois pontos. Desta forma, sempre que se inicia o processo de localização, está-se a introduzir uma tendência no

posicionamento do utilizador, uma vez que este só se pode deslocar ao longo de um número restrito de pontos. Se os valores de *RSSI* forem fiéis e a posição anterior do utilizador for correta, o algoritmo é capaz de garantir o seguimento do utilizador. No entanto, o WiLOS é um sistema bastante variável, logo não se garante que os valores de *RSSI* sejam sempre fidedignos.

Durante o desenvolvimento do algoritmo k-*NNS* com filtro de distância aplicado, teve-se em conta a possibilidade da existência de erros de tendência associados à restrição de pontos. Para tal, verifica-se se o erro resultante da aplicação da distância Euclidiana excede um valor pré-determinado. Deste modo, após a ocorrência sucessiva de 3 erros, aplica-se somente o algoritmo k-NNS para se obter uma localização do utilizador sem condicionamentos (subcapítulo 4.1.3.4). O problema deste método é que, se ocorrer um erro de tendência, o WiLOS demora pelo menos 3 amostras para o detetar. Se o período de amostragem for 3 s, isto corresponde a 9 s para se tentar corrigir a posição atual do utilizador.

Para além disso, se os valores de *RSSI* entre duas áreas forem próximos, isto torna as áreas por eles definidas válidas para deslocação, e o algoritmo k-*NNS* com filtro de distância aplicado não se apercebe da existência de um erro. Esta situação ocorreu nos ensaios 1 e 2, que originou um decréscimo no desempenho no WiLOS, demonstrado pelas precisões de 76.79 % e 80.91 % obtidas nos respetivos ensaios. A figura 5.9 ilustra os dois casos onde o erro de tendência influenciou negativamente o desempenho do sistema.

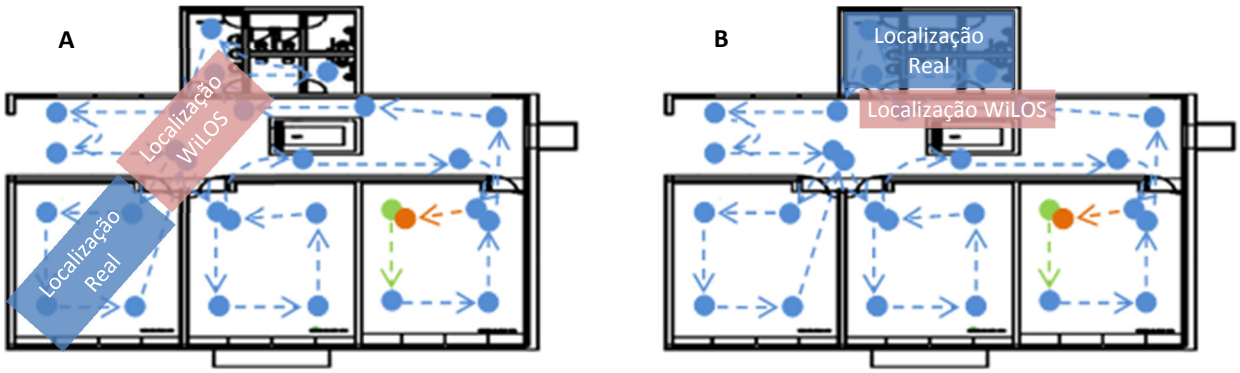


Figura 5.9 - Visualização das zonas onde existe probabilidade de ocorrência de erro de tendência. (**A**): Ensaio 1. (**B**) Ensaio 2.

No ensaio 1 todas as amostras referentes à divisão "Lab. 1.7" foram catalogadas com "Corredor", da mesma forma que as amostras de "WC" foram associadas à divisão "Corredor" no ensaio 2. Como se tratam de zonas com valores de *RSSI* próximos, o algoritmo nunca chegou a detetar o erro. Só quando o utilizador se deslocou para a divisão "Lab. 1.8" (ensaio 1), ou para as proximidades do *AP* localizado no "Corredor" (ensaio 2), é que o WiLOS voltou a acompanhar corretamente a localização do utilizador.

Por fim, verificou-se a influência do acelerómetro, e respetiva compensação de orientação do dispositivo móvel, no algoritmo k-*NNS* com filtro de distância aplicado. Os dados relativos aos ensaios efetuados encontram-se apresentados na tabela 5.10 e na figura 5.10.

Algoritmo	*Orientação do Dispositivo*	*Mínimo [%]*	*Máximo [%]*	*Precisão Média [%]*	*Precisão Média s/ Extremos [%]*	*Desvio Padrão [%]*
k-*NNS* + Filtro de Distância + Orientação	Mão	83.72	94.44	**88.66**	88.38	4.98
	Bolso	87.27	98.11	**92.46**	92.30	3.97

Tabela 5.10 - Sumário estatístico dos dados recolhidos do WiLOS utilizando o algoritmo k-*NNS* com filtro de distância aplicado e compensação de orientação.

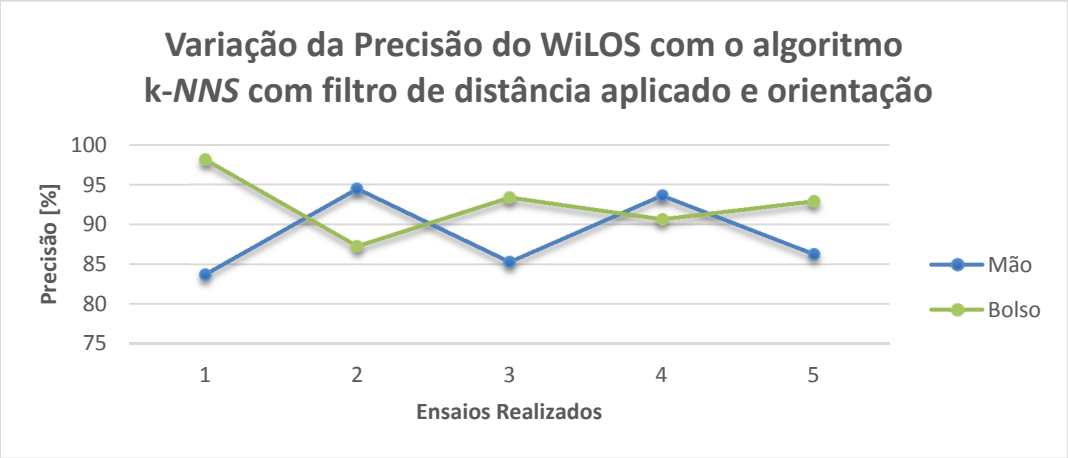


Figura 5.10 - Variação da precisão do WiLOS através do algoritmo k-*NNS* com filtro de distância aplicado e compensação de orientação.

Como se pode verificar, o algoritmo de k-*NNS* com filtro de distância aplicado possui um grande potencial para melhorar o processo de localização, caso se elimine o problema de tendência. Os resultados obtidos para o dispositivo móvel no bolso possuem 4 ensaios com valores de precisão superiores a 90 %, enquanto os ensaios com o dispositivo na mão mantêm-se dentro da gama de valores verificada ao longo de todos os algoritmos (> 85 %).

A figura 5.11 apresenta o resumo de todos os ensaios realizados para se determinar o desempenho do WiLOS, e dos algoritmos de localização por ele disponibilizados. Basicamente a variação dos algoritmos pouco influencia a precisão do sistema quando o telemóvel se encontra na mão do utilizador. No pior dos casos, o algoritmo k-*NNS* com filtro de distância aplicado introduz algum erro considerável. Relativamente aos testes com o *smartphone* no bolso, verifica-se uma melhoria gradual no processo de estimação da localização do utilizador. A aplicação de todas as técnicas de localização em conjunto (k-*NNS*, compensação de orientação e filtro de distância) é a que apresenta os maiores valores de precisão para o caso em estudo. Mais uma vez, a inconsistência do método k-

NNS com filtro de distância aplicado é verificada através da existência de um desvio padrão elevado (7.05 %).

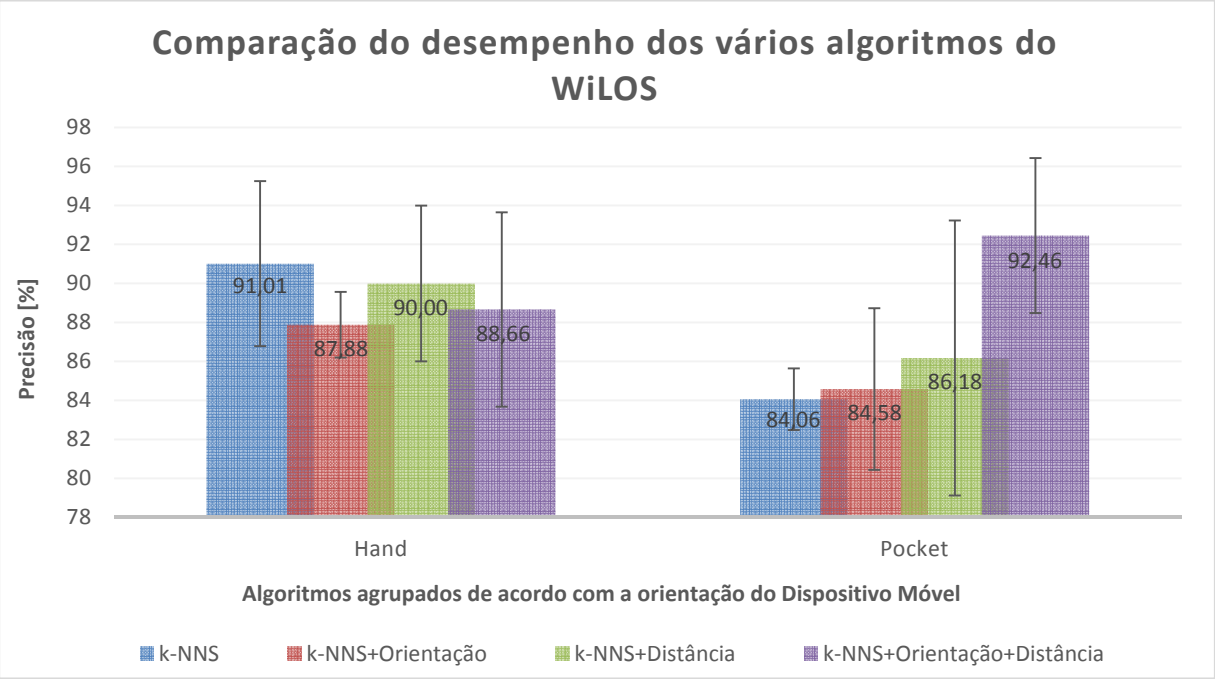


Figura 5.11 - Resumo dos ensaios efetuados para a verificação e estudo do desempenho do WiLOS.

5.2 Análise Experimental do HUEPS

Para se efetuarem os vários ensaios necessários para a validação do HUEPS e dos seus mecanismos de distribuição energética, desenvolveu-se um simulador de localização WiLOS. Este apresenta apenas as funcionalidades mínimas necessárias para a criação de utilizadores, monitoração e disponibilização de serviços de localização. Desta forma, pode-se introduzir até 6 utilizadores no simulador, através da definição dos seus nomes de utilizador e identificadores da BD, ativar o mecanismo de monitoração, que recolhe as localizações virtuais dos utilizadores de 3 em 3 s, e variar a posição dos utilizadores ao longo do mapa da habitação. O HUEPS apenas tem de atuar como atuaria com o WiLOS, e aceder aos serviços externos disponibilizados pelo simulador (*webservices*). A interface gráfica disponibilizada pelo simulador WiLOS é apresentada na figura 5.12.

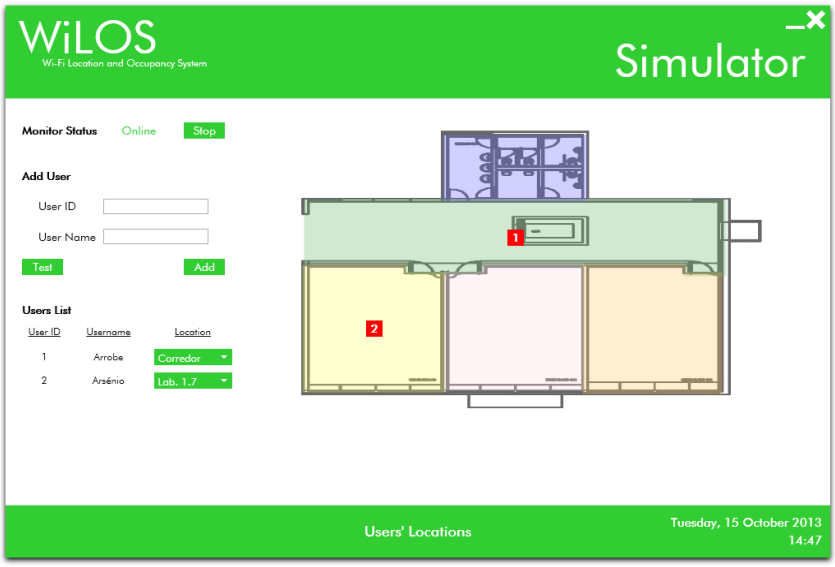


Figura 5.12 - Interface gráfica disponibilizada pelo simulador WiLOS.

Uma vez que a aplicação do HUEPS num dos quadros elétricos do Departamento de Engenharia Eletrotécnica é algo demasiado ambicioso, preparou-se um ambiente de simulação energético composto por 9 aparelhos elétricos. Esta montagem experimental pode ser visualizada na figura 5.13. Os aparelhos utilizados encontram-se "virtualmente" distribuídos pelas diferentes divisões que constituem a zona de implementação. Na figura 5.13 também são apresentadas as diferentes potências consumidas por estes dispositivos, e a respetiva associação de potências/equipamentos a cada divisão.

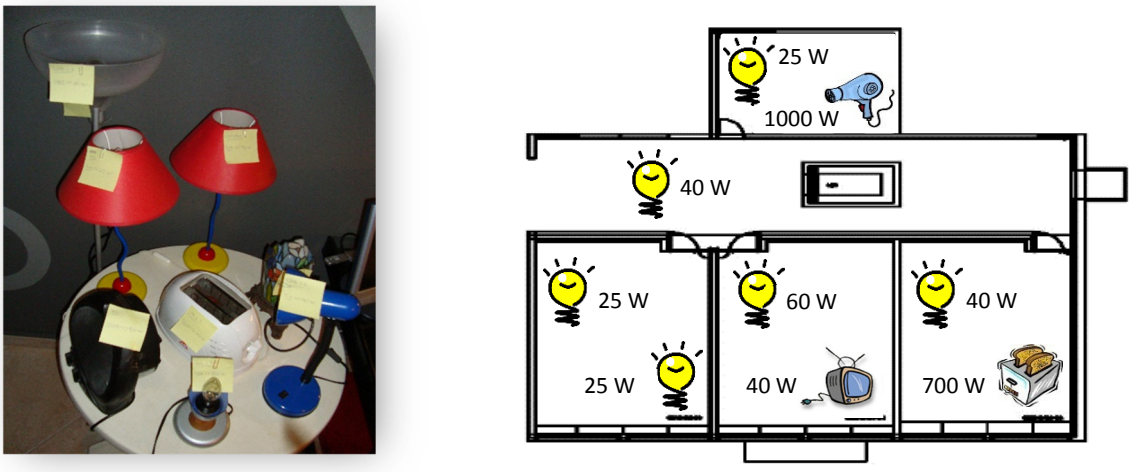


Figura 5.13 - Montagem experimental que simula a rede elétrica da habitação e respetiva distribuição de equipamentos.

A primeira validação do HUEPS passou pela verificação do funcionamento do MM em tempo real, de forma a confirmar se os eventos mais básicos tratados pelas premissas eram corretamente distribuídos. Deste modo simularam-se alguns casos simples, como a entrada de um utilizador dentro

de uma divisão enquanto uma luz era ligada, ou a ocorrência de uma variação quando dois utilizadores se encontravam na mesma divisão, confirmando as atribuições realizadas pelo HUEPS. Esta fase apenas serviu para verificar a implementação do MM antes de se proceder à análise do desempenho do HUEPS, através da execução de um percurso mais complexo.

O estudo do desempenho do HUEPS efetua-se através da execução de 2 percursos distintos por 2 utilizadores. Estes deslocam-se entre divisões e originam eventos energéticos, que podem ser comuns a ambos, e o HUEPS efetua a sua distribuição. Em termos de simulação, isto implica a utilização do simulador WiLOS para "obrigar" os utilizadores a realizarem o percurso enquanto se manobra a montagem experimental, de modo a afetar corretamente os equipamentos elétricos de cada divisão. A linha de tempo referente ao percurso de teste, com todas as ações a realizar pelos utilizadores, bem como as premissas em estudo, é detalhada na figura 5.14. As tabelas 5.11 e 5.12 apresentam os resultados obtidos para a execução de 1 ensaio.

Divisão	*CE Real [Wh]*	*CE Esperado [Wh]*	*CE Atribuído [Wh]*	*Erro CE Real vs. CE Esperado [%]*	*Erro CE Esperado vs. CE Atribuído [%]*
WC	8.85	1.26	0.36	85.77	71.80
Corredor	4.83	9.72	3.77	50.27	61.19
Lab. 1.7	0.42	0.42	0.46	0.00	9.83
Lab. 1.8	2.50	9.58	9.78	73.91	2.04
Lab. 1.9	9.25	4.79	11.61	48.27	58.01
Total	25.85	25.85	25.98	0.00	0.49

Tabela 5.11 - Distribuição do Consumo Energético (CE) por Divisão.

Utilizadores	*CE Real [Wh]*	*CE Esperado [Wh]*	*CE Atribuído [Wh]*	*Erro CE Real vs. CE Esperado [%]*	*Erro CE Esperado vs. CE Atribuído [%]*
Utilizador 1	11.31	10.64	9.27	5.99	12.80
Utilizador 2	12.04	15.22	16.71	20.88	8.91
Total	25.85	25.85	25.98	0.00	0.49

Tabela 5.12 - Distribuição do Consumo Energético (CE) por Utilizador.

Nas tabelas 5.11 e 5.12, a coluna "CE Real" representa o consumo energético teórico esperado, calculado a partir das potências teóricas de cada aparelho e tendo em conta uma distribuição de consumo correta. A coluna "CE Esperado" descreve o consumo energético teórico que deve ser atribuído pelas premissas do HUEPS, sem ter em conta a dissociação de eventos. A distribuição de consumo energético efetuada pelo HUEPS, em tempo real, é apresentada na coluna "CE Atribuído". Por último, as duas últimas colunas, "Erro CE Real vs. CE Esperado" e "Erro CE Esperado vs. CE Atribuído", indicam o erro normalizado, em valor absoluto, existente entre os valores de consumo

energético reais e esperados, e os valores de consumo atribuídos face aos consumos esperados, respetivamente.

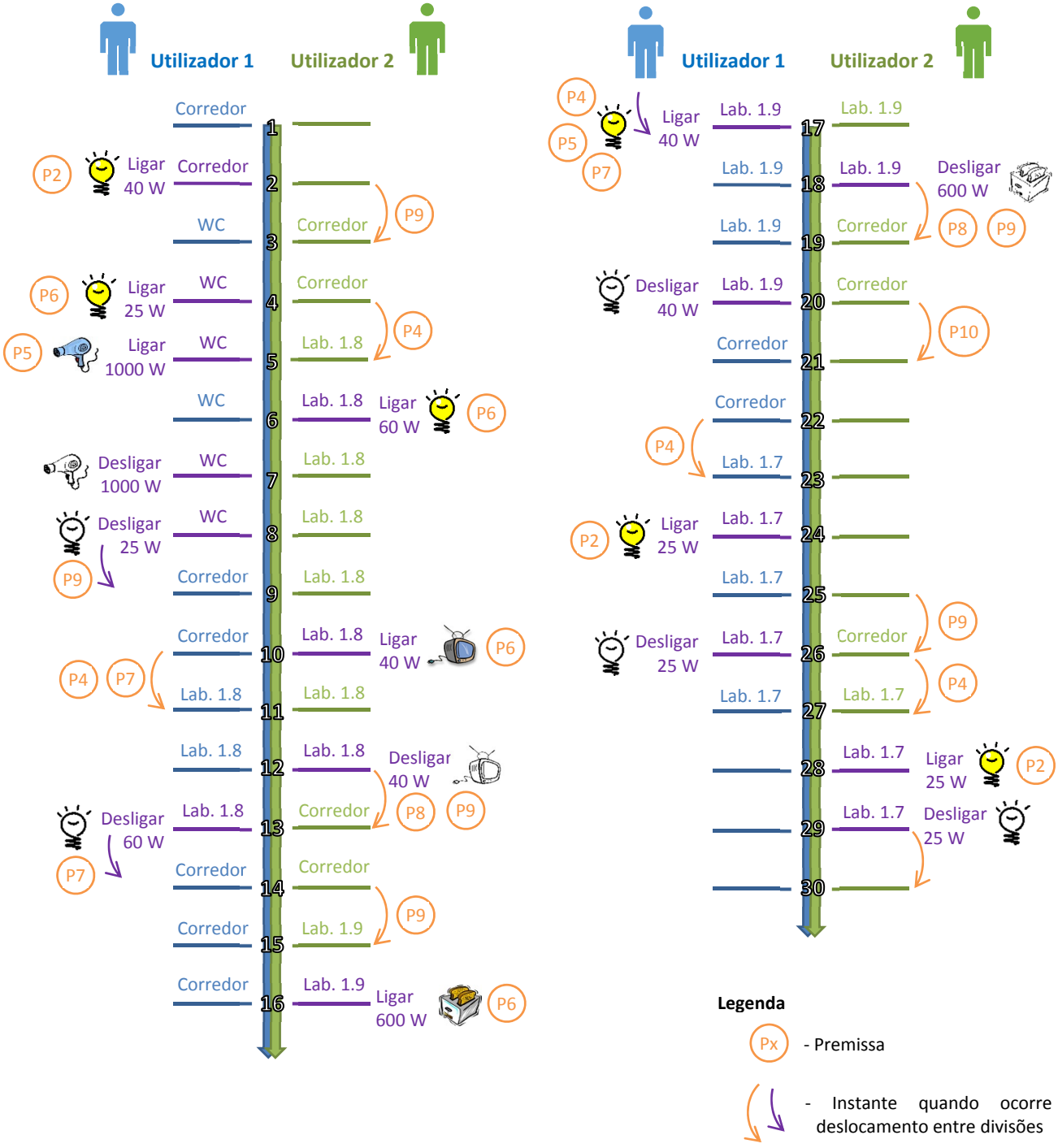


Figura 5.14 - Linha temporal relativa aos percursos e ações efetuadas pelos utilizadores ao longo da habitação.

A coluna "CE Real vs. CE Esperado" serve para demonstrar que a aplicação das próprias premissas introduzem um erro significativo no HUEPS, verificando-se um erro médio de 51.65 % na distribuição de consumos por divisões e um erro médio de 13.43 % para a associação de eventos energéticos a

utilizadores. Destacam-se situações semelhantes à do ponto 5 da figura 5.14, cuja "Premissa 5" efetua uma má distribuição do consumo energético. Neste caso, o utilizador 1 é responsável pela variação de consumo energético (1000 W), mas como o utilizador 2 é que se encontra em movimento, ele é que fica associado a esse evento. Deste modo, esta e outras premissas, justificam a existência do erro de 85.77 % entre o CE Real e o CE Esperado para a divisão "WC" (tabela 5.11). Sendo assim, o objetivo destes ensaios é verificar se a aplicação das premissas, obtenção do posicionamento de utilizadores e dissociação de eventos estão a funcionar como previsto, através da análise dos valores da coluna "CE Esperado vs. CE Atribuído".

A análise desta coluna permite verificar que ocorreu uma distribuição de consumo satisfatória para as divisões "Lab. 1.8" e "Lab. 1.7" (erro < 10 %), e uma atribuição de eventos pouco precisa para as divisões "Lab. 1.9", "WC" e "Corredor" (erro > 55 %). Uma análise mais cuidada ao ensaio em questão permite verificar que as causas de erro para estas divisões ocorreram devido à remanescência de movimento entre amostras consecutivas para o utilizador 2, originando a atribuição de um maior número de eventos às divisões por ele ocupadas. Desta forma, consumos que deveriam ter sido associados ao "WC" ficaram associados ao "Lab. 1.8", justificando um valor de CE atribuído ao "WC" menor que o esperado. Outras situações semelhantes, como a atribuição do consumo ligado ao "Corredor" ao "Lab. 1.9", contribuiriam para o aumento do erro.

A remanescência de movimento é um elemento que deve ser analisado com bastante atenção e depende dos tempos de amostragem praticados. Caso a janela de movimento seja muito curta, torna-se difícil associar eventos energéticos à ocorrência de movimento, especialmente durante a ocorrência de uma variação energética. No entanto, a utilização de uma janela maior, como é o caso aqui representado, dificulta a distinção de eventos de acordo com a estacionariedade dos utilizadores, mesmo recorrendo a um ambiente de simulação relativamente controlado.

Para além disso, o HUEPS possui uma margem de tolerância de 8 W, não sendo capaz de detetar variações energéticas abaixo deste valor. Esta opção é utilizada para que o ruído da rede elétrica, ou o comportamento não constante do consumo elétrico associado aos aparelhos elétricos, não cause a ocorrência de falsos eventos no HUEPS. Porém, este fator acaba por introduzir algum erro em várias fases do funcionamento do HUEPS. Uma delas é durante o transitório que ocorre durante o ato de ligar um aparelho elétrico. Quando se liga um equipamento elétrico, este não atinge logo o seu valor de consumo elétrico constante, possuindo uma fase crescente, um pico e só depois a estabilização. Caso o HUEPS detete um valor de consumo pertencente à fase entre o pico e a estabilização, ou outro que se encontre dentro desta margem, o evento é válido e pode ser registado no HUEPS. Assim, um aparelho que consuma 40 W pode oscilar entre os 32 e os 48 W, uma vez que essa variação é tolerada

pelo HUEPS. Este erro ao fim de alguns minutos, e aplicado a diversos aparelhos, influência negativamente a atribuição do consumo energético correto a um evento.

O mecanismo de dissociação de consumos também insere um erro de origem aleatória no HUEPS. Define-se que este erro é de origem aleatória porque depende dos eventos ativos, dos valores de consumo energético a eles atribuídos e do valor da variação negativa de consumo habitacional verificada num determinado instante. Mesmo recorrendo à definição de prioridades para a dissociação de eventos, de modo a suportar a variação verificada, podem existir valores que causem ambiguidade no HUEPS. Um caso simples é a existência de dois eventos com 60 W e ocorrer uma variação negativa com o mesmo valor. De acordo com o subcapítulo 4.2.2.4, caso não se saiba qual a divisão onde ocorreu essa variação, tentar-se-á desativar o evento mais verossemelhante, que poderá não ser o evento correto.

A aplicação da margem de tolerância de 8 W a este mecanismo também não favorece o processo de dissociação de eventos. Esta pode originar situações em que, em vez de se desativar completamente o evento que mais se adequa à variação ocorrida, apenas se atualiza o seu valor de consumo energético, de modo a compensar essa variação. Para além disso, o uso desta margem também reforça a problemática da fase de transição-pico-estabilização, originando a desativação, ou afetação, do evento incorreto. A permanência deste tipo de eventos "parasitas" ao longo do funcionamento do HUEPS insere discrepâncias no consumo energético, que contribuem para a redução do seu desempenho.

A análise da distribuição do consumo energético pelos utilizadores 1 e 2 mostra a ocorrência de um erro relativamente baixo, como verificado na tabela 5.12 (< 13 %). Porém, a análise previamente efetuada à tabela 5.11 revela que a distribuição desse consumo, pelas divisões por eles ocupadas, foi realizada de forma incorreta. Assim deduz-se que o valor de erro obtido tratou-se de uma coincidência. Deste modo, observa-se que não é possível efetuar uma distribuição de consumos energéticos para cada utilizador, e divisão, com um nível de detalhe elevado. Afinal, as premissas aplicam regras simples para realizarem uma distribuição de consumo de acordo com a qualidade da informação recebida. Os dados referentes ao posicionamento, à movimentação atual dos utilizadores e às variações energéticas ocorridas, não fornecem informação suficiente para se associar e dissociar o consumo corretamente.

Capítulo 6: Conclusões e Trabalhos Futuros

Neste capítulo efetua-se uma síntese geral de todo o trabalho desenvolvido, de acordo com os objetivos propostos. As contribuições dos sistemas implementados, incluindo as contribuições para a investigação, também são aqui apresentadas. Por último, uma análise mais detalhada ao trabalho permite inferir o ponto de situação, as melhorias que podem ser realizadas e possíveis trabalhos futuros.

6.1 Síntese Geral

A problemática do aumento da procura energética a nível mundial é um assunto atual, que tenta encontrar várias soluções para a sua satisfação, sem colocar em causa a sustentabilidade do planeta. Uma delas consiste na utilização de fontes de energia renováveis, de forma a compensar as novas necessidades energéticas, ou substituir a produção efetuada por processos menos ecológicos. Outro método é o reaproveitamento da energia consumida, seja através do aumento da eficiência energética dos aparelhos elétricos, ou a partir de mudanças comportamentais nos consumidores. Este trabalho foca-se nesta última opção, fornecendo aos ocupantes de uma habitação uma ferramenta que permite analisar o consumo energético global da habitação, bem como das suas divisões e dos seus utilizadores.

Para tal, o sistema divide-se em dois subsistemas: o WiLOS, responsável por determinar a localização dos utilizadores dentro da habitação, e o HUEPS, que efetua a correlação da informação energética da habitação com o posicionamento dos utilizadores. Desta forma, o trabalho desenvolvido acaba por fornecer aos ocupantes da habitação, não só um sistema de monitoração energética habitacional, como um serviço de localização.

O WiLOS recorre aos dispositivos móveis dos utilizadores e a uma rede sem fios, composta por vários pontos de acesso, para determinar em tempo real a localização dos ocupantes da habitação. As suas funcionalidades possibilitam a modelação da habitação, calibração do sistema e gestão de utilizadores, de modo a fornecer ao WiLOS a informação necessária para o seu funcionamento. A análise do desempenho do WiLOS consistiu na realização de um percurso pré-estabelecido ao longo da zona de implementação, para se verificar o comportamento dos algoritmos de localização disponibilizados. A análise de resultados demonstra que este sistema garante uma precisão por divisão superior a 85 %, independentemente do algoritmo utilizado. Os melhores resultados são apresentados pelo algoritmo k-*NNS* com filtro de distância aplicado e compensação de orientação do dispositivo móvel, capaz de estimar a localização de um utilizador com uma precisão média de 90.56 %.

O HUEPS acede ao quadro elétrico de uma habitação através de um aparelho de aquisição de dados. Este papel é efetuado pela *Energy Box* fornecida pela EDP e pela Janz-Contar, que permite recolher os valores de corrente e potência consumidos pela habitação. Através da lógica definida no subcapítulo 4.2.2.4, correlaciona-se as variações energéticas verificadas com posicionamento atual dos utilizadores, facultado pelo WiLOS. Deste modo, o HUEPS efetua uma estimação dos perfis energéticos dos utilizadores e das divisões, cujo nível de precisão varia de acordo com as condições que lhe são impostas em tempo real. No caso em estudo, se considerar-se uma situação favorável, o HUEPS efetua uma distribuição dos consumos por divisão com um erro médio de aproximadamente 50 %, e uma atribuição de consumos por utilizador com um erro próximo de 20 %. Desta forma conclui-se que a informação relativa ao posicionamento dos utilizadores, associada às premissas definidas, é insuficiente para que o HUEPS efetue uma distribuição precisa da energia consumida na habitação.

6.2 Contribuição para a Investigação

Existem inúmeros sistemas de localização, que recorrem a diversos métodos e tecnologias para determinar o posicionamento de um utilizador. O método de impressão digital é um dos mais simples, sendo portanto um dos mais usados para a implementação deste tipo de sistemas. A inovação ocorre na metodologia utilizada para se correlacionarem os valores de *RSSI* armazenados com os valores verificados em tempo real. O WiLOS recorre ao algoritmo k-*NNS* para obter uma precisão ao nível da divisão, uma vez que o posicionamento com uma resolução bastante elevada não é necessário para o âmbito deste trabalho.

Uma técnica simples para se tentar melhorar a precisão do sistema recorre ao auxílio do acelerómetro do dispositivo móvel do utilizador, que fornece informação adicional acerca da orientação do dispositivo. Deste modo, através de mais mapas de calibração para as várias orientações do dispositivo, consegue-se compensar ligeiramente o erro introduzido. Pela literatura verificada, esta metodologia ainda não tinha sido aplicada para melhorar a eficiência deste tipo de sistemas.

A aplicação de um método de seguimento que utiliza a tipologia da habitação não é algo inovador. No entanto, o processo de aquisição de modelos para descrever a habitação, nomeadamente o modelo de distância obtido através de processamento de imagem e da aplicação do algoritmo de Dijsktra durante a fase de calibração, nunca foi documentado. Este, associado à definição da distância máxima que um utilizador consegue percorrer entre pontos, permite obter um modelo rudimentar do deslocamento do utilizador dentro da habitação. O seu objetivo é limitar a movimentação do utilizador entre divisões, de modo a melhorar o desempenho do WiLOS.

Por fim, o conceito de um sistema de monitoração energética habitacional que recorre a um serviço de localização para obter perfis energéticos para os utilizadores e para as divisões em tempo real (HUEPS), é uma ferramenta inovadora e bastante criativa. O facto de o HUEPS ser um sistema não intrusivo, isto é, apenas analisa o consumo energético habitacional num só ponto, sem ser necessária a introdução de mecanismos adicionais para a recolha de dados energéticos ao longo da habitação, torna o sistema mais confortável para o utilizador final. Porém, a menor quantidade de informação disponibilizada torna o desenvolvimento do sistema mais desafiante. A correlação entre os dados energéticos e a localização dos utilizadores é a chave para se garantir, no futuro, uma definição de perfis energéticos precisa.

O trabalho desenvolvido deu origem à publicação do seguinte artigo científico: Arrobe, D. M.; Martins, J.F.; Lima, C.; "*Locating and monitoring tenants in PV based buildings*", publicado em *2nd International Conference on Renewable Energy Research and Applications (ICRERA 2013)*, Espanha, Madrid, outubro 2013. Este artigo é disponibilizado para consulta através do anexo R. Para além disso, o sistema integrado WiLOS e HUEPS possui potencial para a publicação de mais artigos, estando prevista a escrita de um artigo de revista.

6.3 Trabalhos Futuros

O WiLOS é um sistema de localização ambicioso, com bastante margem de manobra. Em primeiro lugar, a aquisição de um modelo da tipologia da habitação recorre a um mecanismo simples que implica a introdução de uma planta da habitação e o traçar das respetivas divisões e portas. Seria interessante que somente o processamento da imagem da planta fosse suficiente para determinar a disposição das divisões, portas e suas dimensões. Outra funcionalidade interessante seria a incorporação de um mecanismo que analisasse um ficheiro do tipo CAD e extraísse a informação da habitação necessária ao WiLOS.

O algoritmo de processamento utilizado para a construção automática do grafo da área transitável da habitação, utilizado para se determinar o modelo de distância, pode ser melhorado. Atualmente o mecanismo apenas efetua uma pesquisa entre pontos em linha reta, de modo a verificar a existência de uma parede entre 2 pontos vizinhos. No entanto, podem existir casos em que ambos os pontos estão bastante próximos de uma porta, e um pixel pode fazer a diferença entre ligar ou não ligar ambos os pontos. Estas situações podem introduzir erro no modelo de distância, e influenciar o desempenho do WiLOS. Uma solução consiste em incorporar uma margem de tolerância durante o processamento de imagem (≈ 70 cm), de modo a verificar se foi detetada uma parede isolada, ou uma zona próxima de uma porta.

O algoritmo de localização k-*NNS* com filtro de distância aplicado é bastante volátil, devido à introdução de tendências durante a determinação da posição de um utilizador. Este tanto pode ser uma mais-valia, permitindo o seguimento do utilizador, como uma desvantagem, limitando a sua movimentação numa divisão incorreta. Deste modo, tem de se encontrar uma solução para atenuar o efeito negativo deste algoritmo. Uma solução possível implica a introdução de outra variável no WiLOS, à semelhança da orientação do dispositivo móvel, dada pelo acelerómetro. Hoje em dia, os *smartphones* são dotados não só de acelerómetros, como também de giroscópios e magnetómetros. A combinação destes elementos permite a utilização do dispositivo móvel como uma bússola, de forma a inferir a orientação do dispositivo relativamente à linha do horizonte. Com esta informação, o WiLOS saberia em que sentido e direção é que um utilizador se desloca, de forma a prever a sua localização futura e, possivelmente, eliminar o erro de tendência.

Outro fator bastante importante é a privacidade de um utilizador WiLOS [77]. Apesar de a sua localização poder ser consultada por outros sistemas, de modo a originar novos serviços, isso não implica que todos os utilizadores do WiLOS devam ser capazes de aceder a essa informação. Assim, a incorporação de mecanismos que permitam a privacidade do utilizador dentro do próprio WiLOS não deve ser descartada. No entanto, esta funcionalidade não deverá impedir um administrador de consultar a localização de um utilizador oculto, desde que o utilizador cuja privacidade esteja a ser invadida seja alertado dessa ocorrência.

Relativamente à rede sem fios, a disposição dos *APs* é uma variável com grande influência no sistema. A sua variação consegue aumentar a precisão do WiLOS em 20 %, tendo em conta os casos em estudo. Porém, o processo de determinação da melhor configuração da rede é bastante moroso, devido à necessidade de calibração do WiLOS para cada configuração, e respetiva análise do seu desempenho. Um trabalho futuro que envolvesse o desenvolvimento de um método mais prático e relativamente preciso para este processo seria uma mais-valia para o WiLOS.

Durante a análise do desempenho do WiLOS, todos os ensaios foram realizados por *APs* de marcas diferentes, e somente capazes de usufruir da variante ‘b’ da norma 802.11. Esta limitação adequava-se à situação, uma vez que os dispositivos móveis utilizados só recorrem a esta norma para comunicar com a rede sem fios. Atualmente existem no mercado equipamentos, tanto dispositivos móveis como *APs*, que utilizam normas mais avançadas e que garantem uma melhor cobertura da rede sem fios. A aplicação do WiLOS numa rede com este tipo de equipamentos poderia alterar o desempenho do sistema, sendo pertinente o seu estudo.

Ainda relacionado com a problemática do equipamento, note-se que todos os testes foram efetuados com dispositivos móveis do mesmo modelo e, consequentemente, da mesma marca. Isto

significa que os seus emissores/recetores Wi-Fi possuem desempenhos semelhantes. Caso se deseje comercializar ou implementar o WiLOS numa habitação, não se espera que todos os seus ocupantes possuam o mesmo equipamento. Uma vez que o processo de calibração do WiLOS é efetuado utilizando apenas um dispositivo móvel como referência, caso o desempenho entre equipamentos seja díspar, cada utilizador vai ter um certo erro associado durante o processo de monitoração. Desta forma, o WiLOS necessita de um módulo que permita efetuar a calibração das antenas Wi-Fi de todos os dispositivos móveis da rede, utilizando como referência o dispositivo responsável pela aquisição do mapa de potência do sinal.

Devido ao prazo estipulado para o projeto e desenvolvimento de ambos os sistemas, WiLOS e HUEPS, só foi possível verificar o seu desempenho em conjunto através de simulações. Desta forma, é necessário validar o comportamento obtido pelo HUEPS num ambiente não controlado, com recurso ao verdadeiro serviço de localização oferecido pelo WiLOS.

A melhoria do processo de distribuição de consumo energético habitacional implica a utilização de mais informação. O posicionamento dos utilizadores e as variações energéticas ocorridas na habitação são insuficientes para tornar o processo justo e preciso. Deste modo, a associação do HUEPS a um sistema de reconhecimento de cargas [32] permitiria a categorização das variações energéticas de acordo com os aparelhos elétricos intervenientes. Este nível de detalhe garante uma distribuição muito mais cuidada e minuciosa das variações energéticas causadas pelos ocupantes. Para além disso, a localização dos utilizadores associada à variação energética causada por um determinado dispositivo, pode ser utilizada para inferir a localização dos aparelhos existentes na habitação. Este tipo de mecanismos de aprendizagem, ao fim de algum tempo, contribui para melhorar a qualidade dos perfis energéticos.

Numa fase inicial do projeto, o HUEPS possuía uma componente móvel para permitir aos seus utilizadores a consulta da informação energética em qualquer lugar. No entanto, o tempo despendido para o desenvolvimento do WiLOS impossibilitou a sua implementação. A portabilidade e associação do perfil energético de um utilizador ao seu dispositivo móvel introduz no HUEPS novas funcionalidades.

Considere-se um cenário onde o WiLOS e o HUEPS encontram-se implementados em todas as zonas habitacionais e empresariais. Neste cenário, sempre que um utilizador entra dentro de uma habitação, quer dele ou de outrem, o seu dispositivo móvel regista-se automaticamente na rede WiLOS. A portabilidade do perfil energético do utilizador permite ao HUEPS daquela habitação efetuar a distribuição do consumo energético por todos os seus ocupantes, incluindo o “novo” utilizador. Aqui, o seu perfil energético pode ser utilizado para determinar as suas preferências ou rotinas, de modo a

garantir a atualização do perfil com precisão. Este processo ocorreria em todas as habitações e no próprio local de trabalho do utilizador. Desta forma, o perfil energético tornar-se-ia mais completo, representando o consumo energético diário de uma pessoa, em vez do perfil energético adquirido somente a nível habitacional.

A informação disponibilizada pelo WiLOS e HUEPS também pode ser utilizada por sistemas de automação e domótica, de modo a verificar a existência de utilizadores nas imediações, ou correlacionar o consumo energético verificado com as suas rotinas energéticas. Estes dados possibilitam uma melhor gestão da energia da habitação, através da redução de desperdícios.

Por último, poder-se-ia determinar se a população estaria interessada em possuir este tipo de sistemas nas suas habitações, através da elaboração de inquéritos estatísticos. Um estudo referente à influência destes sistemas no dia-a-dia dos seus ocupantes também seria interessante. Deste modo verificar-se-ia se, de facto, a utilização deste tipo de ferramentas permite ao utilizador desenvolver consciencialização energética, originando mudanças comportamentais.

Bibliografia

[1] *Beyond Petroleum* BP, "*BP Energy Outlook 2030*", Londres, Reino Unido, janeiro 2012.

[2] U.S. Energy Information Administration, "*Annual Energy Outlook 2012 with Projections to 2035*", Washington DC, Estados Unidos da América, 25 de junho de 2012.

[3] Ruhl, C., Appleby, P., Naumov, A., et al., "*Economic Development and the Demand for Energy: A Historical Perspective on the Next 20 Years*", novembro 2012.

[4] Renata Branco, "Combustíveis Fósseis: Vantagens e Desvantagens", Manutenção & Suprimentos, Brasil, 10 de maio de 2011, *online*.
Disponível em http://www.manutencaoesuprimentos.com.br.

[5] Internacional Energy Agency, "*Energy Technology Perspectives 2010: Scenarios & Strategies to 2050*", Paris, França, 2010.

[6] GALP Energia, "*Uma atividade sustentável: A sustentabilidade explicada*", Sustentabilidade e Responsabilidade Corporativa, Portugal 2012, *online*.
Disponível em http://www.galpenergia.com.

[7] GALP Energia, "*Projeto Smart GALP*", Sustentabilidade e Eficiência Energética, Portugal, 2012.
Disponível em http://www.mysmartgalp.com e em http://www.galpenergia.com.

[8] Energias de Portugal, Programa ECO, "*Guia prático para a casa eficiente*", "*O que é a eficiência energética*", "*Grandes marcos das alterações climáticas*", "*Eficiência energética e energias renováveis*", Portugal, 2012, *online*.
Disponível em http://www.eco.edp.pt.

[9] Energias de Portugal, Projeto TWIST, "Eficiência Energética", "Alterações Climatéricas", Portugal, 2012, *online*.
Disponível em http://www.twitst.edp.pt.

[10] Energias de Portugal Distribuição, "*InovGrid, Next Generation* Grid", Seminário sobre a *Inovação para a Competitividade em Serviços de água*, Porto, Portugal, 9 de fevereiro de 2012.

[11] GALP Energia, "*No controlo do aquecimento Global", "A eficiência energética na sua casa e escritório*", Sustentabilidade e Eficiência Energética, Portugal, 2012, *online*.
Disponível em http://www.galpenergia.com.

[12] Energias de Portugal, Projeto TWIST, "*Eficiência Energética de Edifícios*", Portugal, 26 de Maio de 2012, *online*.

Disponível em http://twisters.blogs.sapo.pt/13593.html.

[13] B2B Energias Renováveis, "*ISA e EDP discutem solução para famílias pouparem energia*", Portugal, 7 de dezembro de 2012.
Disponível em http://www.renewablesb2b.com.

[14] Redes Energéticas Nacionais REN, "*Sustentabilidade e Compromissos*", Portugal, 2012.
Disponível em http://www.ren.pt.

[15] Energias de Portugal, "*Dicas de Eficiência Energética: Viva a sua casa com uma energia mais sustentável*", Odivelas, Portugal, 2012.

[16] BizEE Software Ltd, "*What is Energy Awareness?*", Energy Lens, Reino Unido, 2012.
Disponível em http://www.energylens.com.

[17] BizEE Software Ltd, "*The Energy Saving Meaning?*", Energy Lens, Reino Unido, 2012.
Disponível em http://www.energylens.com.

[18] U.S. Energy Information Administration, "*Energy Kids: Using & Saving Energy – Saving Energy*", Estados Unidos da América.
Disponível em http://www.eia.gov/kids/index.cfm.

[19] Climate Literacy and Energy Awareness Network CLEAN, "*The Principle of Energy Awareness*", 18 de janeiro de 2012.
Disponível em http://cleanet.org.

[20] Instituto Nacional de Estatística de Portugal, I.P., Direção Geral de Energia e Geologia de Portugal, "*Inquérito ao Consumo de Energia no Sector Doméstico 2010*", Lisboa, Portugal, Edição de 2011.

[21] Lei n.º 51-A/2011 de 30 de setembro, publicada no Diário da República, 1ª série - N.º 189 - 30 setembro de 2011.

[22] Lei n.º 51-A/2011 de 30 de setembro, publicada no Diário da República, 1ª série - N.º 189 - 30 setembro de 2011.

[23] Alahmad, M.; Zulfiqar, M.F.; Hasna, H.; Sharif, H.; Sordiashie, E.; Aljuhaishi, N.A., "Green and Sustainable Technologies for the Built Environment" *Developments in E-systems Engineering (DeSE), 2011* , Páginas 521-526, 6-8 de dezembro de 2011.

[24] Spagnolli A.; Corradi N.; Gamberini L.; Hoggan E.; Jacucci G.; Katzeff C.; et al., "Eco-Feedback on the Go: Motivating Energy Awareness", *Computer*, Volume 44, Número 5, Páginas 38-45, maio 2011.

[25] Sundramoorthy, V.; Cooper, G.; Linge, N.; Qi Liu, "Domesticating Energy-Monitoring Systems: Challenges and Design Concerns," *Pervasive Computing, IEEE*, Volume 10, Número 1, Páginas 20-27, janeiro-março de 2011.

[26] Ghidini, G.; Das, S.K., "Improving home energy efficiency with E^2Home: A Web-based application for integrated electricity consumption and contextual information visualization," *Smart Grid Communications (SmartGridComm), 2012 IEEE Third International Conference on*, Páginas 471-475, 5-8 de novembro 2012.

[27] Stawitz, C.C.; McGehee, H.S.; Devlin, C.M.; Tan, F.; Wong, S., "Increasing awareness of residential energy consumption: Data analysis and presentation for ecoMOD, a sustainable housing initiative," *Systems and Information Engineering Design Symposium, 2008. SIEDS 2008. IEEE*, Páginas 55-59, 25 de abril de 2008.

[28] Marusic, L.; Skocir, P.; Petric, A.; Jezic, G., "Home-in-Palm — A mobile service for remote control of household energy consumption," *Telecommunications (ConTEL), Proceedings of the 2011 11th International Conference on* , vol., no., pp.109,116, 15-17 June 2011

[29] Junbo Wang; Cheng, Zixue; Lei Jing; Ozawa, Yota; Yinghui Zhou, "A location-aware lifestyle improvement system to save energy in smart home," *Awareness Science and Technology (iCAST), 2012 4th International Conference on* , vol., no., pp.109,114, 21-24 Aug. 2012.

[30] Stragier, J. ; Hauttekeete, L. ; De Marez, L. ; Derboven, J. ;Laporte, L., "Household Energy Use and Creating Awareness: Opportunities for ICT", *Emerging Intelligent Data and Web Technologies (EIDWT), 2012 Third International Conference on*, Pages 276-280, September 2012, in press.

[31] Zhong Bocheng, "Design of Building Energy Monitoring and Management System," *Business Computing and Global Informatization (BCGIN), 2012 Second International Conference on* , vol., no., pp.645,648, 12-14 Oct. 2012

[32] Martins, J.F.; Lopes, R.; Lima, C.; Romero-Cadaval, E.; Vinnikov, D., "A novel nonintrusive load monitoring system based on the S-Transform," *Optimization of Electrical and Electronic Equipment (OPTIM), 2012 13th International Conference on* , vol., no., pp.973,978, 24-26 May 2012.

[33] Hsueh-Hsien Chang; Ching-Lung Lin, "A New Method for Load Identification of Nonintrusive Energy Management System in Smart Home," *e-Business Engineering (ICEBE), 2010 IEEE 7th International Conference on* , vol., no., pp.351,357, 10-12 Nov. 2010.

[34] Sawyer, R.L.; Anderson, J.M.; Foulks, E.L.; Troxler, J.O.; Cox, R.W., "Creating low-cost energy-management systems for homes using non-intrusive energy monitoring devices," *Energy Conversion Congress and Exposition, 2009. ECCE 2009. IEEE* , vol., no., pp.3239,3246, 20-24 Sept. 2009.

[35] Bier, T.; Abdeslam, D.O.; Merckle, J.; Benyoucef, D., "Smart meter systems detection & classification using artificial neural networks," *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society* , vol., no., pp.3324,3329, 25-28 Oct. 2012;

[36] Kwang-Soon Choi; Yang-Keun Ahn; Young-Choong Park; Woo-Chool Park; Hae-Moon Seo; Kwang-Mo Jung; Kyeung-Hak Seo, "Architectural Design of Home Energy Saving System Based on Realtime Energy-Awareness," *Ubiquitous Information Technologies & Applications, 2009. ICUT '09. Proceedings of the 4th International Conference on* , vol., no., pp.1,5, 20-22 Dec. 2009.

[37] Jihua Ye; Qi Xie; Yaohong Xiahou; Chunlan Wang, "The research of an adaptive smart home system," *Computer Science & Education (ICCSE), 2012 7th International Conference on* , vol., no., pp.882,887, 14-17 July 2012.

[38] Liu Zhi-Gang; Huang Wei, "The design of smart home system based on Wi-Fi," *Computational Problem-Solving (ICCP), 2012 International Conference on* , vol., no., pp.454,456, 19-21 Oct. 2012.

[39] Gao Chong; Ling Zhihao; Yuan Yifeng, "The research and implement of smart home system based on Internet of Things," *Electronics, Communications and Control (ICECC), 2011 International Conference on* , vol., no., pp.2944,2947, 9-11 Sept. 2011.

[40] Ramlee, R.A.; Othman, M.A.; Leong, M.H.; Ismail, M.M.; Ranjit, S.S.S., "Smart home system using android application," *Information and Communication Technology (ICoICT), 2013 International Conference of* , vol., no., pp.277,280, 20-22 March 2013.

[41] Depuru, S.S.S.R.; Lingfeng Wang; Devabhaktuni, V.; Gudi, N., "Smart meters for power grid — Challenges, issues, advantages and status," *Power Systems Conference and Exposition (PSCE), 2011 IEEE/PES* , vol., no., pp.1,7, 20-23 March 2011.

[42] Vukmirović, S.; Erdeljan, A.; Kulic, Filip; Luković, S., "Software architecture for Smart Metering systems with Virtual Power Plant," *MELECON 2010 - 2010 15th IEEE Mediterranean Electrotechnical Conference* , vol., no., pp.448,451, 26-28 April 2010.

[43] Chuyuan Wei; Yongzhen Li, "Design of energy consumption monitoring and energy-saving management system of intelligent building based on the Internet of things,"*Electronics,*

Communications and Control (ICECC), 2011 International Conference on , vol., no., pp.3650,3652, 9-11 Sept. 2011.

[44] Fuller, R.; Christie, J.; Nichols, J.; et al., "*A Highly Flexible and Scalable System for Location Determination of Wireless Devices*"', *IEEE Position and Location National Symposium*, Páginas 233-239, 2002

[45] Joy, V.; Laxman, P.V., "*Smart Spaces: Indoor Wireless Location Management System", The 2007 International Conference on Next Generation Mobile Applications, services and* Technologies, Páginas 261-266, setembro 2007

[46] Beale, R., "*Improving the accuracy and reliability of wireless location Systems: a case study*", *7th Mexican International Conference on Artificial Intelligence (MICAI)*, Páginas 383-387, outubro 2008

[47] Golden, S.A.; Bateman, S.S., "*Sensor Measurements for Wi-Fi Location with Emphasis on Time-of-Arrival Ranging*", *IEEE Transactions on Mobile Computing*, Vol. 6, N. 10, Páginas 1185-1198, outubro 2007

[48] Shang, J.; Yu, S.; Zhu, L., "*Location-Aware Systems for Short-range Wireless Networks*", *International Symposium on Computer Network and Multimedia Technology,* Páginas 1-5, janeiro 2009

[49] Lin, J.; Chu, W.; Gong, T.; et al., "*Integrating BLIP into Location-aware System: A Service-Oriented Method*", *Fourth International Conference on Computer Sciences and Convergence Information Technology*, Páginas 144-148, novembro 2009

[50] Zhao, Y.; Zhou, H.; Li, M.; Kong, R., "*Implementation of Indoor Positioning System based on Location Fingerprinting in Wireless Networks*", *4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, Páginas 1-4, outubro 2008

[51] Ito, S.; Kawaguchi, N., "*Bayesian Based Location Estimation System Using Wireless LAN*", *Third IEEE International Conference on Pervasive Computing and Communications Workshops*, Páginas 273-278, março 2005

[52] Emery, M.; Denko, M.K., "IEEE 802.11 WLAN Based Real-Time Location Tracking in Indoor and Outdoor Environments," *Electrical and Computer Engineering, 2007. CCECE 2007. Canadian Conference on* , vol., no., pp.1062,1065, 22-26 April 2007

[53] Farivar, R.; Wiczer, D.; et al., "*A Statistical Study on the Impact of Wireless Signals' Behavior on Location Estimation Accuracy in 802.11 Fingerprinting Systems*", *IEEE International Symposium on Computing & Processing (Hardware/Software)*, Páginas 1-8, maio 2009

[54] Figuera, C.; Rojo-Álvarez, J.L.; et al., "*Time-Space Sampling and Mobile Device Calibration for WiFi Indoor Location Systems", IEEE Transactions on Mobile Computing*, Vol. 10, N. 7, Páginas 913-926, julho 2011

[55] Bagci, F.; Kluge F.; et al., "*LocSens – An Indoor Location Tracking System using Wireless Sensors", Proceedings of 17th International Conference on Computer Communications and Networks* (ICCCN), Páginas 1-5, agosto 2008

[56] Zhao, M.; Yang, H.; Liu, J.; et al., "*Directional Wi-Fi Based Indoor Location System for Emergency*", *7th International Conference on Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing (UIC/ATC)*, Páginas 501-502, outubro 2010

[57] Bahl, P.; Padmanabhan, V.N., "RADAR: an in-building RF-based user location and tracking system," *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* , vol.2, no., pp.775,784 vol.2, 2000

[58] Ahmed, U.; Gavrilov, A.; Sungyoung Lee; Young-Koo Lee, "Context-Aware Fuzzy ArtMap for Received Signal Strength Based Location Systems," *Neural Networks, 2007. IJCNN 2007. International Joint Conference on* , vol., no., pp.2740,2745, 12-17 Aug. 2007

[59] Ishii, J.; Tamura, Y.; Asama, H., "*Filter Design by Using Map Information on Wireless-LAN Location Awareness System*", *International Conference on Mechatronics and Automation* (ICMA), Páginas 2967- 2972, agosto 2009

[60] Paul, A.S.; Wan, E.A., *"Wi-Fi based Indoor Localization and Tracking using Sigma-Point Kalman Filtering Methods", IEEE/ION Position, Location and Navigation Symposium*, Páginas 646-659, maio 2008

[61] Po-Hsuan Tseng; Chao-Lin Chen; Kai-Ten Feng, "An Unified Kalman Tracking Technique for Wireless Location Systems," *Wireless Pervasive Computing, 2007. ISWPC '07. 2nd International Symposium on* , vol., no., pp.,, 5-7 Feb. 2007

[62] Chin-Der Wann; Ming-Hui Lin, "Data fusion methods for accuracy improvement in wireless location systems,"*Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE* , vol.1, no., pp.471,476 Vol.1, 21-25 March 2004

[63] Fonseca, J.; Moura, A.D., "Classificadores - Métodos de Vizinhanças e Funções de Distâncias", Sistemas de Informação Médica (SIM), UNL-FCT, Caparica, Portugal, 2012

[64] Wicklin, R., "What is Mahalanobis distance?", The DO Loop - Data Analysis and Statistical Thinking, 15 February 2012, online.

Disponível em http://blogs.sas.com/content/iml/2012/02/15/what-is-mahalanobis-distance/

[65] Egea-Lopez, E.; Martinez-Sala, A.; Monzo-Sanchez, F.; Garcia-Haro, J., "A test-bed for a wireless identification and location system for industrial items," *Communications, Computers and signal Processing, 2003. PACRIM. 2003 IEEE Pacific Rim Conference on* , vol.1, no., pp.280,283 vol.1, 28-30 Aug. 2003

[66] Kumar, A., "Location of mobile user in an indoor wireless system operating in 2 GHz frequency band," *Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on* , vol., no., pp.294,297, 17-20 May 2009

[67] Quoc Cuong Nguyen; Dongil Shin; Dongkyoo Shin; Juhan Kim, "Real-Time Human Tracker Based on Location and Motion Recognition of User for Smart Home," *Multimedia and Ubiquitous Engineering, 2009. MUE '09. Third International Conference on* , vol., no., pp.243,250, 4-6 June 2009

[68] Tesoriero, R.; Gallud, J.A.; Lozano, M.; Penichet, V. M R, "A Location-Aware System Using RFID and Mobile Devices for Art Museums," *Autonomic and Autonomous Systems, 2008. ICAS 2008. Fourth International Conference on* , vol., no., pp.76,81, 16-21 March 2008

[69] Chenchen Zhang; Jianling Qi; Haihang Yu, "Design for downhole wireless location system based on ZIGBEE wireless sensor network," *Computer Science and Network Technology (ICCSNT), 2011 International Conference on* , vol.3, no., pp.1679,1681, 24-26 Dec. 2011

[70] Shouwei Gao; Haihang Wang; Hairun Wang; Gang Wang, "A low-cost wireless location system based on Zigbee technology," *Transportation, Mechanical, and Electrical Engineering (TMEE), 2011 International Conference on* , vol., no., pp.1665,1668, 16-18 Dec. 2011

[71] Quigley, A.; Ward, B.; Ottrey, C.; Cutting, D.; Kummerfeld, R., "BlueStar, a privacy centric location aware system," *Position Location and Navigation Symposium, 2004. PLANS 2004* , vol., no., pp.684,689, 26-29 April 2004

[72] Lashkari, A.H.; Parhizkar, B.; Ngan, M.N.A., "WIFI-Based Indoor Positioning System," *Computer and Network Technology (ICCNT), 2010 Second International Conference on* , vol., no., pp.76,78, 23-25 April 2010

[73] "IEEE 802.11g and IEEE 802.11n Standards", IEEE Standard Association – IEEE 802.11TM, Wireless LANs, online
Disponível em http://standards.ieee.org/about/get/802/802.11.html

[74] Shih-Hau Fang; Tsung-Nan Lin, "Projection-Based Location System via Multiple Discriminant Analysis in Wireless Local Area Networks," *Vehicular Technology, IEEE Transactions on* , vol.58, no.9, pp.5009,5019, Nov. 2009

[75] Sybase, "*Statistics: The Growth Of Mobile Into 2011*", setembro 2010
Disponível em http://www.youtube.com/user/SybaseInc?feature=watch

[76] Online Psychology Degree, "*Getting in Bed With Gadgets, Your Technology is Keeping you Awake*", novembro 2012
Disponível em http://www.onlinepsychologydegree.net/2012/11/12/sleeping-with-gadgets/

[77] Kim, E.; Plummer, M.; Hiltz, S.R.; Jones, Q., "Perceived Benefits and Concerns of Prospective Users of the SmartCampus Location-Aware Community System Test-bed," *System Sciences, 2007. HICSS 2007. 40th Annual Hawaii International Conference on* , vol., no., pp.19,19, Jan. 2007

Anexos

Anexo A: Diagramas de Casos de Uso - mWiLOS

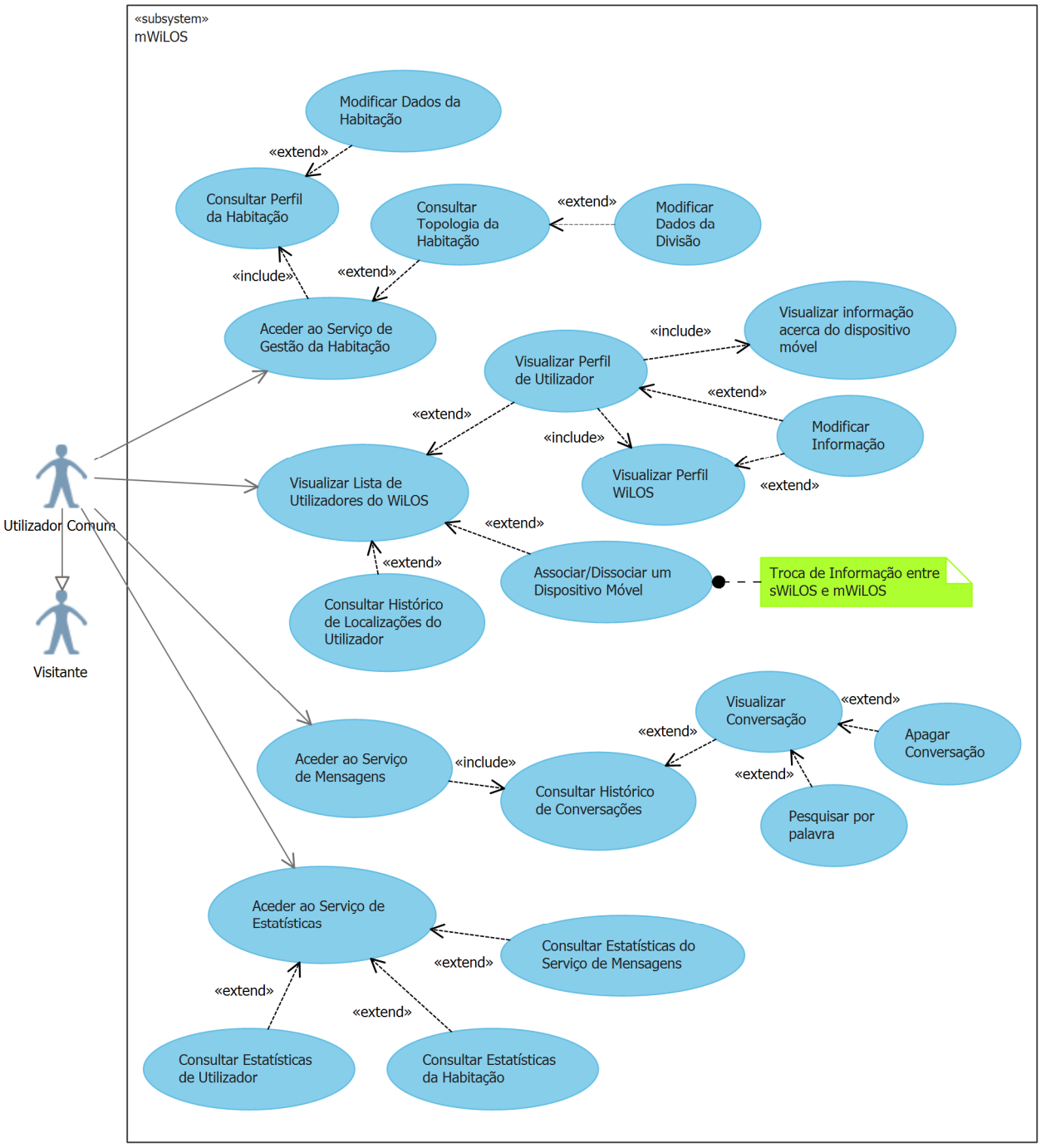


Figura A.1 - Diagrama de casos de uso do mWiLOS – Funcionalidades oferecidas ao Utilizador Comum.

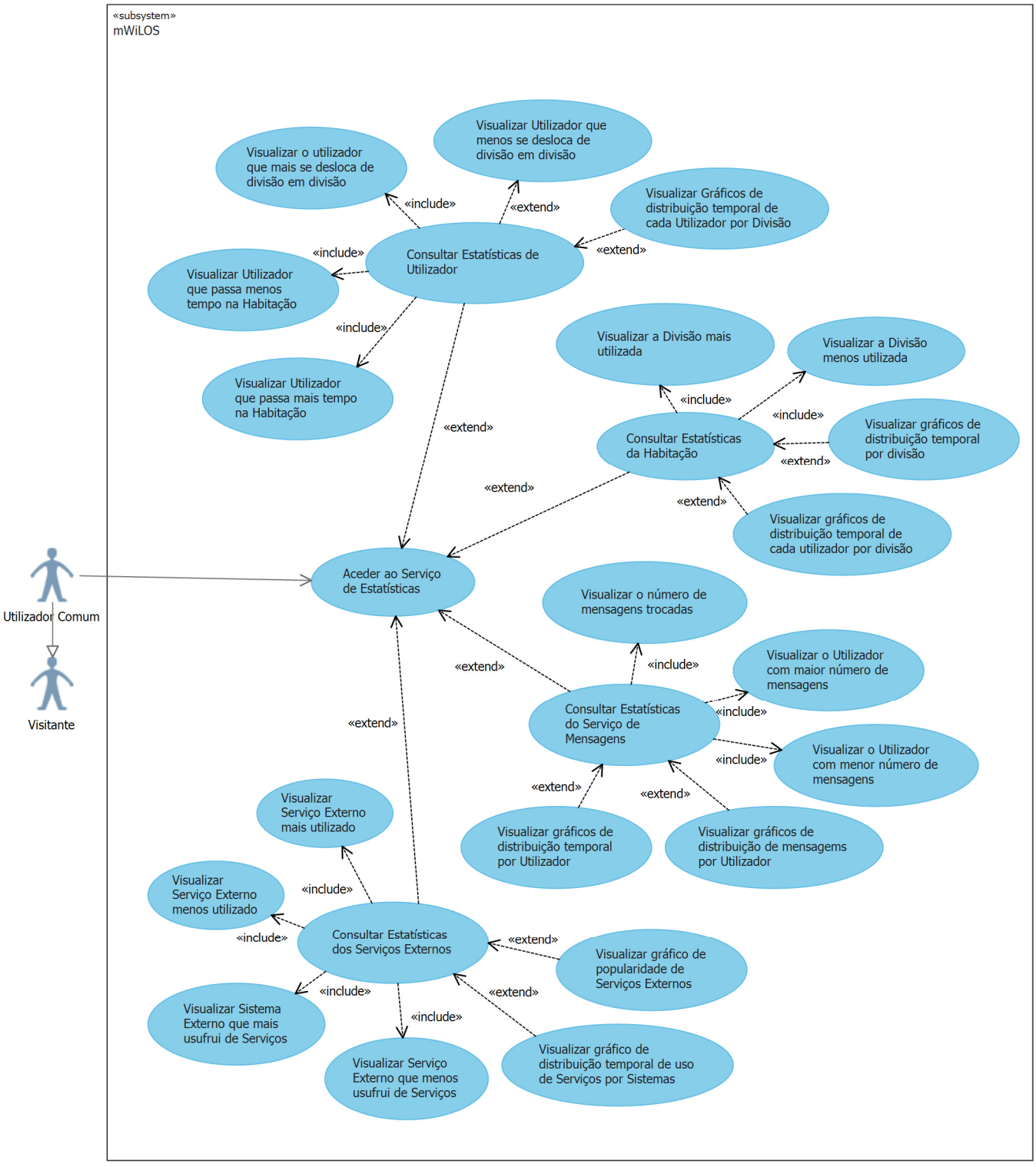


Figura A.2 - Diagrama de casos de uso do mWiLOS – Funcionalidades específicas de estatística.

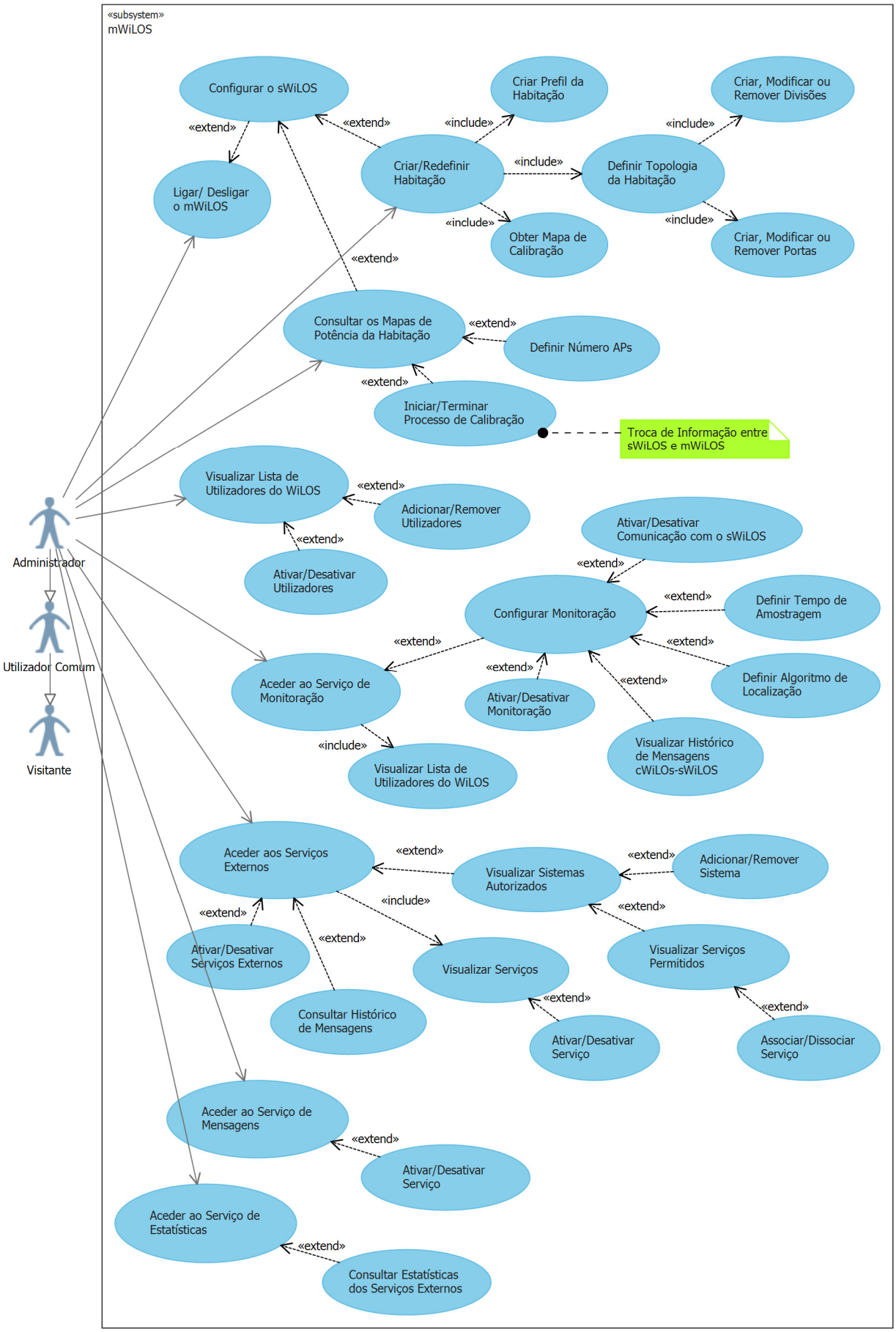


Figura A.3 - Diagrama de casos de uso do mWiLOS – Funcionalidades disponibilizadas ao Administrador.

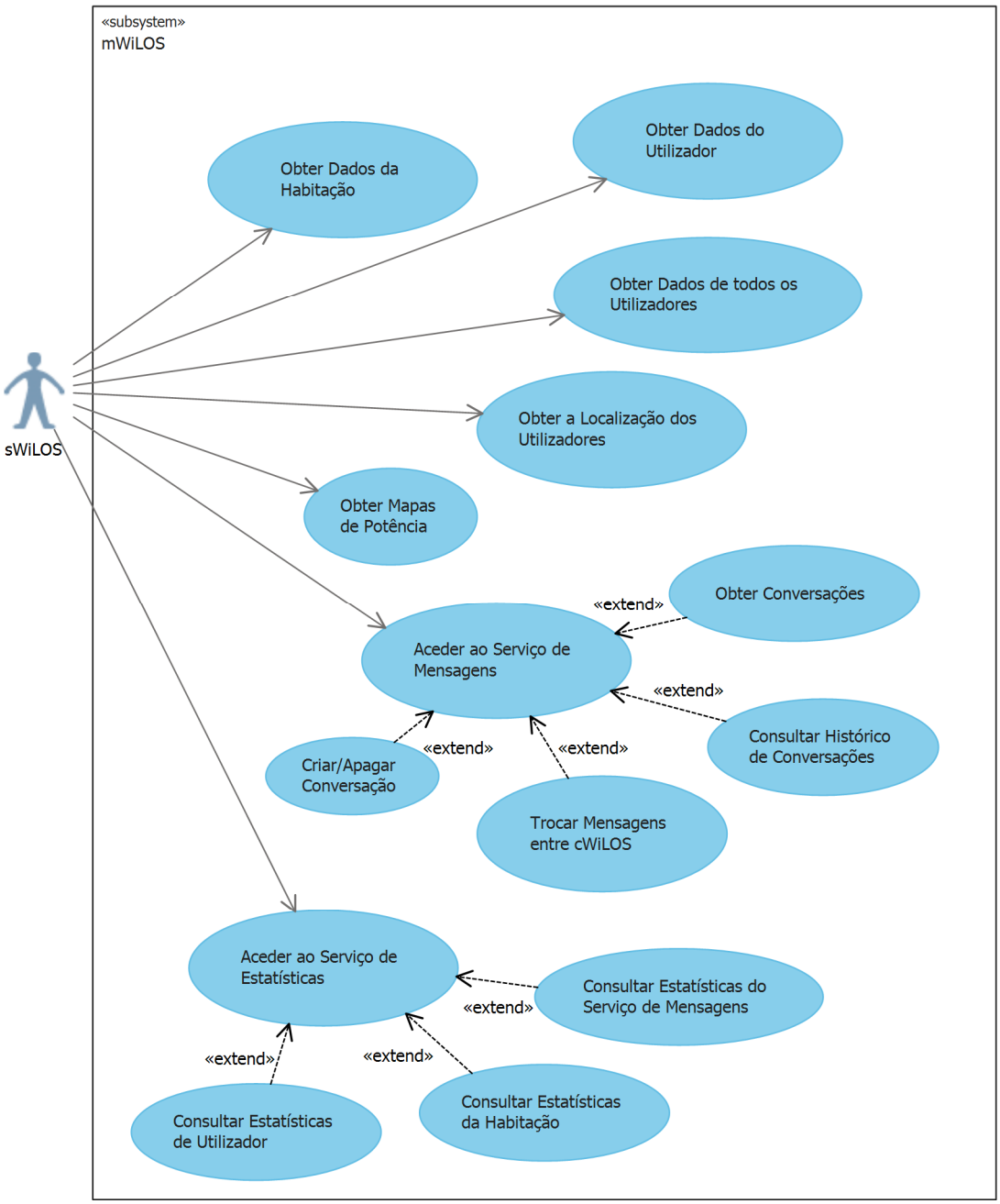


Figura A.4 - Diagrama de casos de uso do mWiLOS – Funcionalidades disponibilizadas ao sWiLOS.

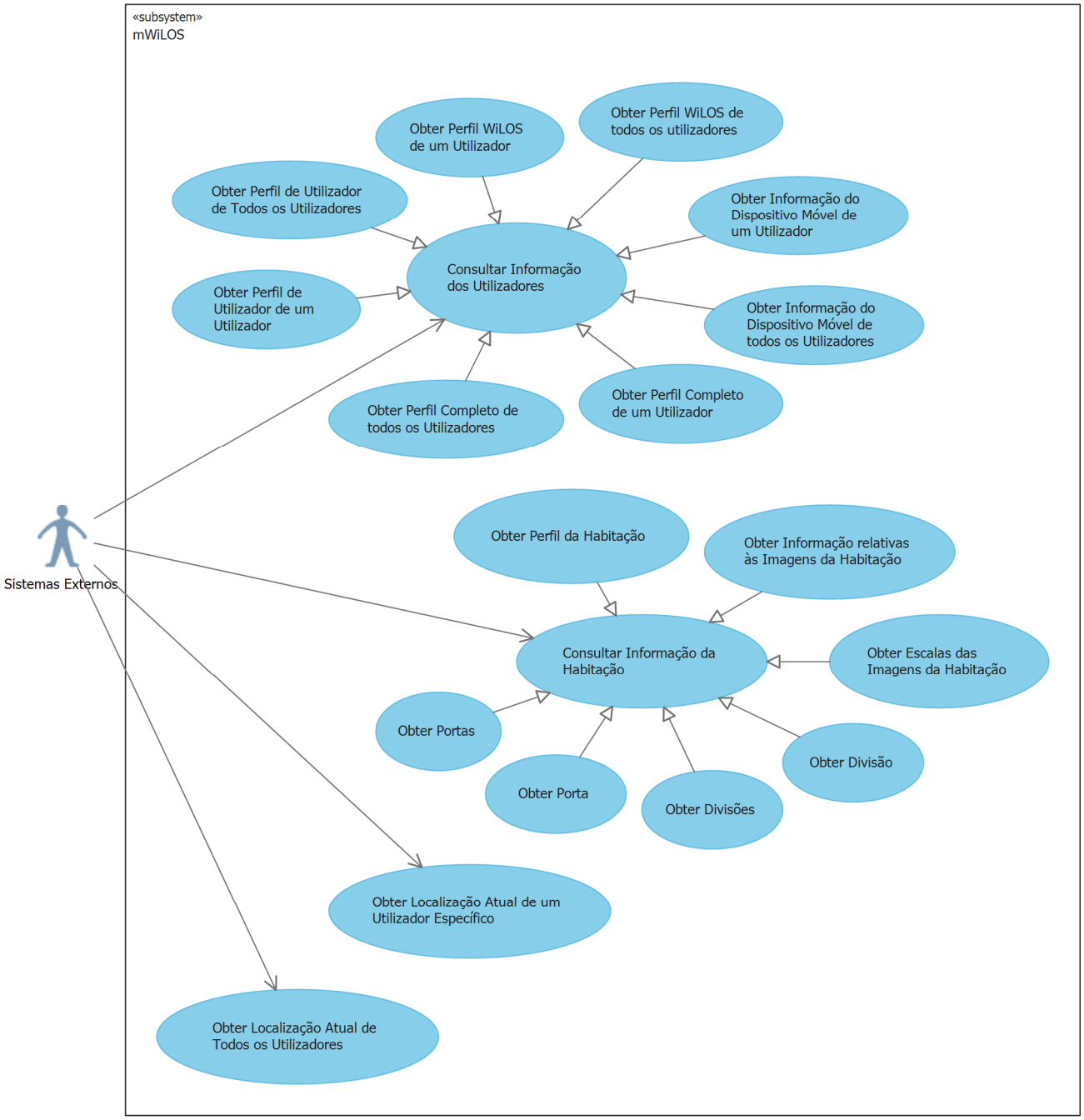


Figura A.5 - Diagrama de casos de uso do mWiLOS – Funcionalidades oferecidas aos Sistemas Externos.

Anexo B: Diagramas de Casos de Uso - sWiLOS

Figura B.1 - Diagrama de casos de uso do sWiLOS – Funcionalidades disponibilizadas ao Utilizador Comum.

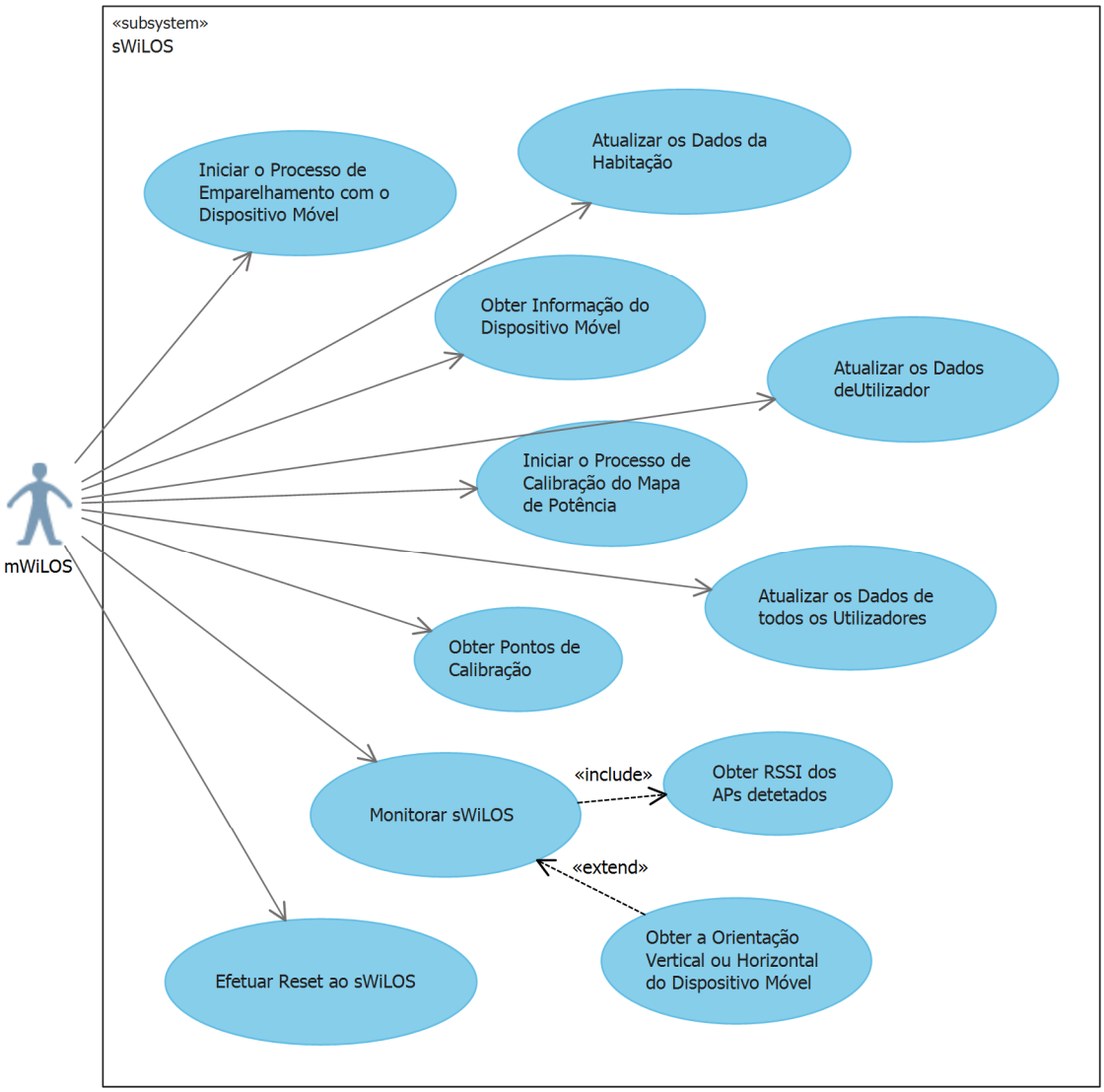


Figura B.2 - Diagrama de casos de uso do sWiLOS – Funcionalidades disponibilizadas ao mWiLOS.

Anexo C: Diagramas de Casos de Uso - HUEPS

Figura C.1- Diagrama de casos de uso do HUEPS – Funcionalidades oferecidas ao Utilizador Comum.

Figura C.2 - Diagrama de casos de uso do HUEPS – Funcionalidades oferecidas ao Administrador.

Figura C.3 - Diagrama de casos de uso do HUEPS – Funcionalidades disponibilizadas aos Sistemas Externos.

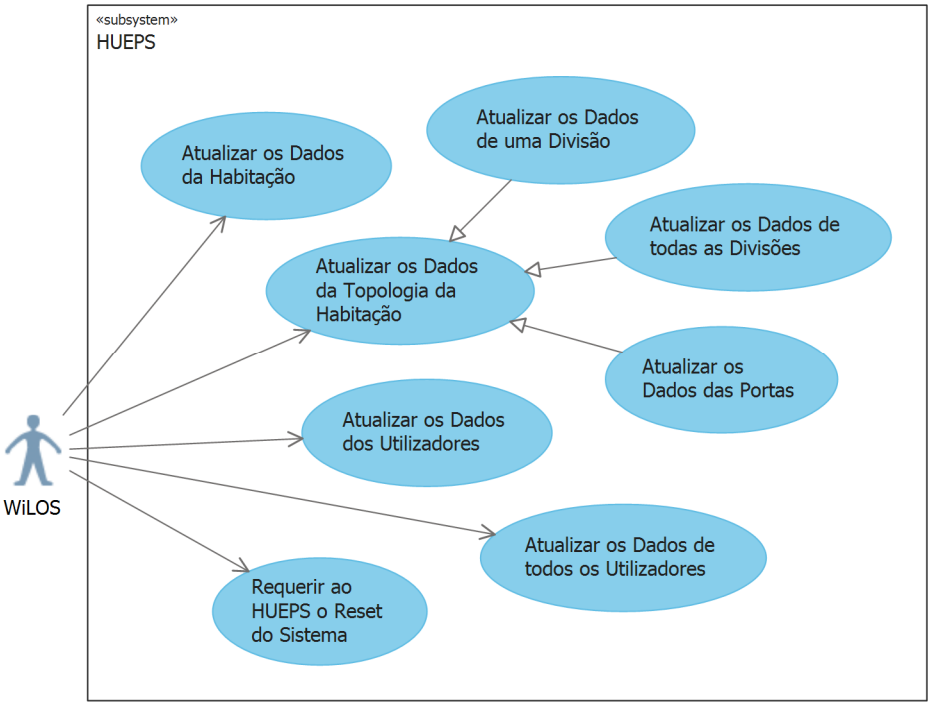


Figura C.4 - Diagrama de casos de uso do HUEPS – Funcionalidades oferecidas ao WiLOS.

Anexo D: Pormenorização do DER - mWiLOS

A estruturação e organização da informação são essenciais, não só para facilitar o seu acesso e consulta, como também para correlacionar diversos elementos que permitem a criação de nova informação. Assim, o modelo de dados a aplicar ao ISM é representado pelo diagrama de entidades e relacionamentos (DER) da figura D.1. O DER pode ser repartido em 6 secções: "*House Information*"; "*User Information*"; "*Positioning*"; "*Chat Service*"; "*External Services*" e "*mWiLOS Configuration*". Esta divisão ocorre devido à natureza dos dados que são manipulados pelo WiLOS.

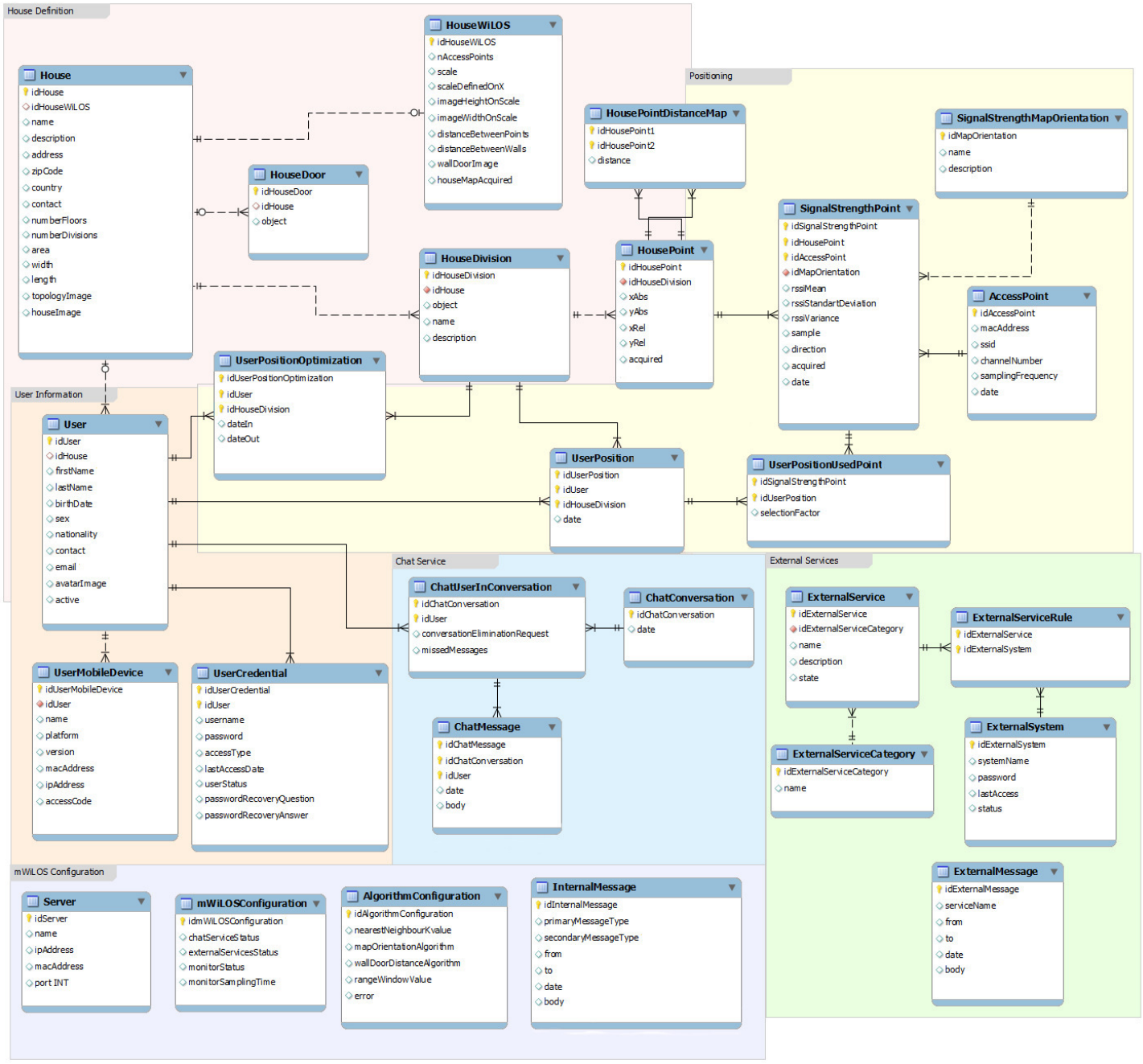


Figura D.1 - Diagrama de entidades e relacionamentos do mWiLOS – Visão Lógica.

A área "*House Information*" engloba todos os dados referentes a uma habitação e os seus constituintes. A informação mais generalista referente a uma habitação é representada pela entidade "*House*". Aqui, elementos como a morada, descrição da habitação, contacto e imagens referentes à tipologia da habitação podem ser armazenados para consulta futura. As dimensões da habitação e outros dados referentes à sua tipologia, como o número de divisões, também se encontram definidos nesta entidade.

A tipologia da habitação é composta essencialmente por portas e divisões. Desta forma, estes elementos são essenciais para se definir o modelo da tipologia da habitação. A entidade "*House Door*" caracteriza uma porta, armazenando o objeto utilizado para a sua modelação. De modo semelhante, a entidade "*House Division*" descreve uma divisão, não só através do objeto que a modela, como também a partir da atribuição de um nome à divisão e respetiva descrição.

No entanto, o processo de calibração necessita de uma representação discreta da área da habitação, algo que não é oferecido pelo modelo estabelecido. Porém, o modelo descrito por "*House Division*" e "*House Door*", pode ser analisado de forma a se extrair um modelo discreto da habitação. Cada ponto da habitação obtido possui uma posição absoluta e uma posição relativa, sendo representado pela entidade "*House Point*". A posição absoluta é em relação ao canto superior esquerdo da imagem referente à tipologia da habitação. A posição relativa corresponde à distância existente entre o ponto e o canto superior esquerdo da divisão ao qual pertence. Todos os pontos que compõem o modelo discreto da habitação representam o mapa de calibração do WiLOS. Os modelos idealizados encontram-se representados na figura D.2.

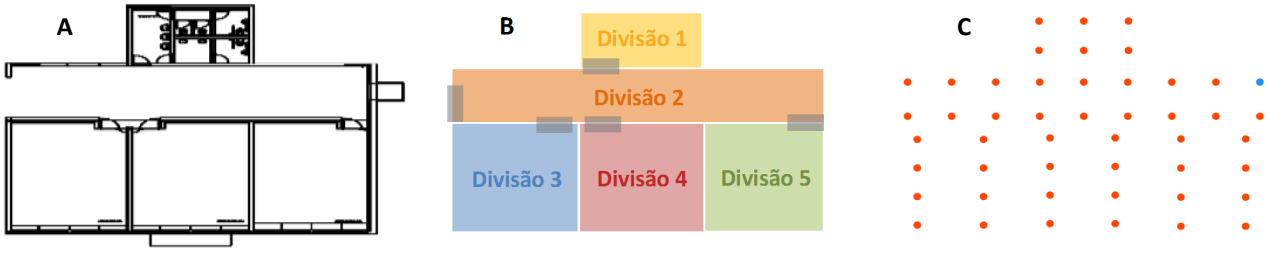


Figura D.2 - (**A**): Tipologia Real da Habitação. (**B**): Modelo de divisões e portas. (**C**): Modelo discreto da habitação, ou mapa de calibração, que se extrai do modelo (**B**).

Os processos utilizados para se extrair os modelos apresentados na figura D2 implicam a definição de determinados parâmetros. A entidade "*House WiLOS*" fica encarregue de modelar essa informação, sendo os seus atributos justificados no subcapítulo 4.1.3, uma vez que surgem das metodologias aplicadas durante a fase de implementação.

O mapa de calibração do WiLOS possui dois propósitos. O primeiro é possibilitar a calibração do sistema através da recolha dos mapas de potência do sinal ao longo da habitação. O segundo é a obtenção de um modelo que permite descrever as distâncias dentro da habitação. Este modelo é utilizado pelo algoritmo k-*NNS* com filtro de distância aplicado para condicionar o conjunto 'k' resultante, através da distância máxima que uma pessoa consegue percorrer entre dois pontos.

O modelo de distância é representado através do conceito de grafo, onde um grafo é composto por vários nós ligados entre si por um coeficiente de distância. A sua análise permite inferir todas as combinações de nós possíveis, e quais as distâncias resultantes. Como o objetivo deste modelo é obter uma referência a partir da qual se pode limitar a movimentação do utilizador, apenas interessam as menores distâncias alcançáveis a partir de cada nó. Neste caso, cada ponto do mapa de calibração define um nó, e esse nó encontra-se ligado a outro nó pela distância mínima que os separa. Cada uma destas ligações é definida na entidade "*House Point Distance Map*". A figura D.3 e as tabelas D.1 e D.2 exemplificam o conceito de grafo para o mapa de calibração, todas as combinações de nós possíveis e as ligações que são relevantes para o WiLOS.

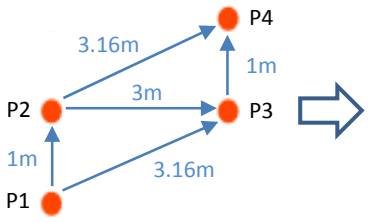


Figura D.3 - Grafo ilustrativo do mapa de calibração do WiLOS.

Partida	Ligações	Chegada	Distância [m]
P1	P2	P2	1
	P3	P3	3.16
	P2,P4	P4	4.16
	P2,P3,P4	P4	5
	P3,P4	P4	4.16

Tabela D.1 - Distâncias entre o nó "P1" e os restantes nós, resultantes da análise do grafo da figura D.3.

Partida	Chegada	Distância [m]
P1	P2	1
	P3	3.16
	P4	4.16
P2	…	…

Tabela D.2 - Entidade "*House Point Distance Map*". Distâncias mínimas entre o nó "P1" e os restantes nós do grafo, extraídas a partir da tabela D.1.

A aquisição de mapas de potência de sinal implica a existência de uma infraestrutura Wi-Fi composta por um ou vários *APs*. A informação referente a cada *AP*, como o endereço MAC, identificador de rede ou o canal utilizado para transmitir o sinal, é representada pela entidade "*Access Point*". Como os mapas de potência obtidos são relativos a um determinado *AP*, estes dados são necessários para as fases de calibração e monitoração do WiLOS.

A entidade "*Signal Strength Point*" representa um ponto pertencente aos vários mapas de potência do sinal recolhidos durante a fase de calibração. Cada um destes pontos possui uma referência para o ponto de calibração correspondente, dado por "*House Point*", e uma referência para o *AP* ao qual pertence o valor de *RSSI* recolhido. Para permitir a reutilização desta entidade noutras metodologias, como a inferência probabilística, define-se que um ponto é composto por valor médio, desvio padrão e variância de *RSSI*. Em todo o caso, o atributo "*sample*" armazena todos os dados

recolhidos durante a aquisição do ponto, de modo a fornecer maior liberdade na fase de implementação. A informação referente à orientação do dispositivo móvel durante a calibração é efetuada através de uma referência à entidade "*Signal Strength Map Orientation*". Esta determina as várias orientações válidas que um dispositivo móvel pode adquirir. No caso do WiLOS, estas consistem na orientação vertical, ou dispositivo móvel no bolso, e na orientação horizontal, dada pelo dispositivo móvel na mão do utilizador.

A monitoração do utilizador ao longo do tempo permite a criação de um histórico de localizações. Cada localização é representada por um utilizador e por uma divisão por ele ocupada num determinado instante temporal. A entidade "*User Position*" relaciona as entidades "*User*" e "*House Division*" de forma a representar o conceito de localização citado. Por outro lado, a entidade "*User Position Used Point*" efetua a ponte entre as entidades "*User Position*" e "*Signal Strength Point*", de modo a registar os pontos que foram utilizados durante o processo de decisão para inferir a localização do utilizador. Esta entidade também armazena o valor atribuído pelo algoritmo de localização durante o processo de decisão, como o valor resultante da aplicação de uma métrica, permitindo a sua consulta e validação por parte do utilizador.

A entidade "*User Position Optimization*" é utilizada para simplificar a visualização do posicionamento de um utilizador. A análise direta da entidade "*User Position*" torna-se fastidiosa devido à elevada quantidade de informação originada pelo serviço de monitoração. Porém, se um utilizador estiver sempre na mesma divisão, torna-se mais atrativo saber que este entrou na divisão 1 às 12:30 e saiu às 13:30. Admitindo um tempo de amostragem de 3 s, a análise da entidade "*User Position*" para o mesmo período de tempo equivaleria à visualização de 1200 entradas.

A secção "*User Information*" é constituída pelas entidades "*User*", "*User Credential*" e "*User Mobile Device*". A entidade "*User*" representa um utilizador no seu sentido mais generalista. A sua caracterização é efetuada por elementos como o seu nome, idade, contacto, sexo, data de nascimento, entre outros que permitem a sua identificação. No entanto, a acesso ao WiLOS não deve ser efetuado por qualquer utilizador. A entidade "*User Credential*" permite o registo de um utilizador no WiLOS através da definição de um nome de utilizador, palavra-passe e outras informações relevantes para os mecanismos de autenticação do WiLOS. Dentro do próprio WiLOS devem existir diferentes níveis de acesso às funcionalidades implementadas, de modo a limitar a informação e os mecanismos de configuração que podem ser acedidos por cada utilizador. Desta forma, a entidade "*User Credential*" torna-se necessária para garantir a privacidade, segurança e integridade do sistema.

Como citado ao longo dos capítulos 3 e 4, a deteção de um utilizador na rede do WiLOS não é efetuada através da localização do próprio utilizador, mas sim através do dispositivo móvel que se

encontra na sua posse. Este aparelho é caracterizado por um nome atribuído pelo utilizador, um sistema operativo, um endereço MAC distinto e um endereço IP, que se encontra disponível sempre que o dispositivo se liga à rede. Estes elementos tornam o dispositivo único para o WiLOS, sendo a entidade "*User Mobile Device*" responsável por modelar esta informação.

A implementação de um serviço de mensagens no WiLOS também exige uma organização estrutural da informação, efetuada pela área "*Chat Service*" e as suas entidades. Para além disso, alguns conceitos devem ser previamente definidos, tal como o conceito de conversação.

Uma conversação consiste numa troca de mensagens sequencial que ocorre entre dois ou mais utilizadores. Nesta, cada utilizador consegue visualizar as suas próprias mensagens, bem como as mensagens enviadas pelos outros intervenientes. Deste modo não existe o conceito de destinatário, uma vez que todos os participantes recebem a mensagem. Uma mensagem é constituída por um "corpo" (i.e., o conteúdo da mensagem), um remetente, uma data de envio e uma conversação à qual pertence. Esta mensagem é modelada pela entidade "*Chat Message*". A associação de vários utilizadores a uma conversação é realizada pela entidade "*Chat Users In Conversation*". Esta também permite verificar se um utilizador possui mensagens por ler, ou se a conversação encontra-se em lista de espera para ser eliminada. A eliminação de uma conversação só pode ocorrer após todos os seus intervenientes terem requisitado a sua remoção.

Para simplificar alguns mecanismos relacionados com as conversações, define-se que ocorre apenas uma conversação ao longo do dia entre os mesmos utilizadores. Assim, todas as mensagens trocadas entre os utilizadores dessa conversação ficam automaticamente associadas a uma conversação diária. No final do dia essa conversação é armazenada e uma nova é automaticamente gerada, de forma transparente ao utilizador. A entidade "*Chat Conversation*" é composta por um identificador único e pela data de criação da conversação. Este identificador é associado às restantes entidades da área "*Chat Service*", de modo a identificar inequivocamente todos os elementos comuns a uma conversação.

A zona "*External Services Information*" é responsável por organizar os serviços disponibilizados pelo mWiLOS e quais os sistemas externos que lhes podem aceder. A definição dos serviços é efetuada através da entidade "*External Service*", que possui como características o nome do serviço e a sua descrição. Estes serviços, de acordo com a natureza da informação que manipulam, podem ser catalogados recorrendo à entidade "*External Service Category*".

A acessibilidade de um serviço depende de certas condicionantes. Um administrador do WiLOS possui privilégios para desabilitar o serviço ou definir quais são os sistemas que podem usufruir da sua informação. O estado de funcionamento de um serviço (ativo ou desabilitado) é representado pela

característica "*status*" em "*External Service*". A definição do acesso personalizado ao serviço é realizada pela entidade "*External Services Allowed Systems*". No entanto, um sistema externo só pode aceder aos serviços do mWiLOS se estiver registado e autenticado. Assim, a entidade "*External Services' System*" utiliza o nome do sistema e a sua palavra-passe para garantir a segurança da informação do mWiLOS.

Um histórico de mensagens trocadas entre o mWiLOS e os sistemas externos é uma ferramenta útil na medida em que permite verificar o resultado da interação entre os serviços e os sistemas. Desta forma é possível validar o funcionamento do módulo "Serviços Externos" e detetar acessos não autorizados ao mWiLOS. A entidade "*External Services' Message*" modela as características de uma mensagem, como o remetente, conteúdo e data de envio, de forma a arquivar a informação enviada aos sistemas externos.

Por último, a área "*mWiLOS Configuration*" engloba as entidades responsáveis pelo funcionamento interno do mWiLOS. A entidade "*Server*" define a informação que permite ao sWiLOS identificar e aceder remotamente ao mWiLOS através da rede Wi-Fi. Tipicamente esta informação é composta pelo endereço IP, MAC e pela porta da rede por onde é efetuada a comunicação. Todas as mensagens trocadas entre mWiLOS e sWiLOS, quer durante o processo de monitoração, quer para a satisfação de pedidos do utilizador, são armazenadas com o intuito de se obter um histórico de comunicação. Este possui como objetivo confirmar se os mecanismos de comunicação estão a funcionar corretamente após a implementação do WiLOS. Uma mensagem trocada é composta pelos elementos típicos de uma mensagem, como o remetente, o destinatário, a data de envio e o conteúdo da mensagem. Para além disso, são reservados dois campos para definir a categoria principal e secundária da mensagem. Uma mensagem é representada pela entidade "*Internal Message*".

A configuração do estado de alguns módulos da camada Controlo do mWiLOS é efetuada através da entidade "*mWiLOS Configuration*". Deste modo é possível ativar ou desativar o serviço de mensagens fornecido pelo CSM, disponibilizar a informação para sistemas externos através do ESM, e configurar o mecanismo de monitoração executado pelo MM. A configuração do MM inclui a definição do tempo de amostragem a utilizar durante o processo de monitoração.

Os algoritmos de localização do WiLOS são configurados pela entidade "*Algorithm Configuration*". A manipulação desta entidade permite selecionar qual o algoritmo, ou algoritmos, a utilizar durante o processo de monitoração. Esta entidade também permite a definição do valor de 'k' a aplicar ao algoritmo k-*NNS* e a distância máxima que um utilizador consegue percorrer entre dois pontos. Este último é utilizado pelo algoritmo k-*NNS* com filtro de distância aplicado, de modo a limitar a movimentação do utilizador a partir do modelo de distância (entidade "*House Point Distance Map*").

Anexo E: Pormenorização do DER - HUEPS

A organização da informação do HUEPS obedece ao modelo definido pelo DER representado na figura E.1. Este pode ser seccionado em 6 áreas de acordo com o tipo de informação a manipular: "*Smart Meter Information*"; "*House and Energy Supplier Information*"; "*Users Information*"; "*Energy Consumption Information*"; "*External Services*" e "HUEPS *Configuration*".

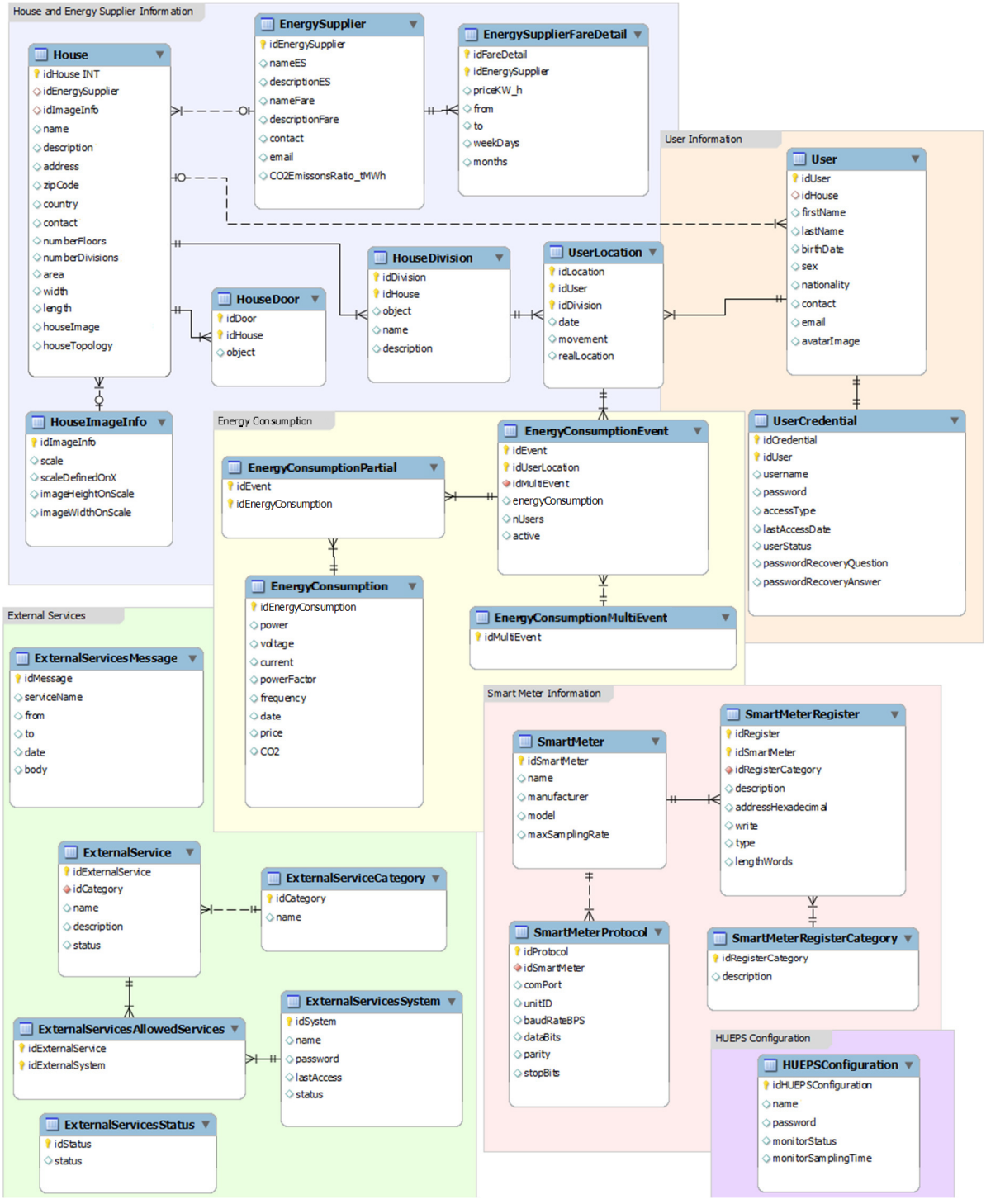


Figura E.1 - Diagrama de entidades e relacionamentos do HUEPS – Visão Lógica.

A área "*Smart Meter Information*" destina-se a armazenar a informação referente ao medidor "inteligente", neste caso a *EB*. Desta forma, a entidade "*Smart Meter*" modela o aparelho em si, bem como as suas características mais relevantes para o HUEPS. Traços como o tempo de amostragem máximo permitido, nome, modelo e fabricante do aparelho são aqui representados. Para além disso, a comunicação entre o HUEPS e a *EB* ocorre segundo um protocolo estabelecido pelo fabricante. Uma vez que a *EB* utiliza o protocolo Modbus para comunicar, a entidade "*Smart Meter Protocol*" possui os atributos que definem este protocolo, tais como o número de bits utilizados para troca de dados, o ritmo de transmissão e o tipo de codificação a aplicar.

O protocolo de comunicação permite aceder aos diferentes registos que compõem a *EB*. Estes registos podem servir para consultar informação ou modificar o comportamento do aparelho. De acordo com o documento fornecido pela EDP, definiu-se a entidade "*Smart Meter Register*". Esta permite organizar os diferentes registos existentes no aparelho, bem como o tipo de dados que podem ser lidos e/ou escritos a partir deles. A entidade "*Smart Meter Register Category*" possibilita ainda catalogar esses registos de acordo com as suas funcionalidades específicas.

A modelação dos dados referentes à habitação e ao fornecedor de energia é efetuada através da área "*House and Energy Supplier Information*". Esta é constituída pelas entidades "*House*", "*House Division*", "*House Door"*, "*House Image Info*", "*Energy Supplier*" e "*Energy Supplier Fare Detail*". As 3 primeiras herdam as mesmas características das entidades "*House*", "*House Division"* e "*House Door*" do mWiLOS (subcapítulo 4.1.1.3). O sincronismo de informação efetuado entre ambos os sistemas leva à replicação da informação do WiLOS no HUEPS. Deste modo, as entidades mencionadas são coincidentes. A entidade "*House Image Info*" permite definir as características da imagem referente à tipologia da habitação. Uma vez que esta é obtida através do WiLOS, dados como o tamanho da imagem durante a obtenção de escalas são importantes para se replicar corretamente esta imagem e outros elementos a ela associados no HUEPS.

As restantes entidades representam o fornecedor de energia e as tarifas que estão a ser atualmente aplicadas na habitação. Assim, em "*Energy Supplier*" descrevem-se dados mais triviais, tais como o nome do fornecedor de energia, o seu contato e qual o nome do tarifário aplicado. O valor de emissões de CO2 associadas ao consumo energético também se encontra aqui representado. Como este valor é divulgado pelo fornecedor de energia periodicamente, achou-se mais adequado a sua definição nesta entidade. A entidade "*Energy Supplier Fare Detail*" permite definir os dias da semana e os períodos horários em que é taxada uma determinada quantia monetária por consumo efetuado.

A secção "*Users Information*" contém a informação relativa aos utilizadores e à sua localização. Os dados que permitem identificar inequivocamente um utilizador encontram-se representados pela

entidade "*User*". O perfil de utilizador, que define o seu nome de utilizador, nível de acesso ao sistema, palavra-passe, etc., é modelado pela entidade "*User Credential*". Estas entidades são coerentes com as entidades "*User*" e "*User Credential*" do modelo de dados do WiLOS, sendo também alvo de sincronismo entre ambos os sistemas (HUEPS e WiLOS). Deste modo reutiliza-se o perfil de utilizador do WiLOS no HUEPS, garantindo o mesmo nível de acesso. As localizações de cada utilizador são também fornecidas pelos WiLOS e ficam registadas na entidade "*User Location*".

A zona "*External Services Information*" é responsável por organizar os serviços disponibilizados pelo HUEPS e quais os sistemas externos que lhes podem aceder. A definição dos serviços é efetuada através da entidade "*External Service*", que possui como características o nome do serviço e a sua descrição. Estes serviços, de acordo com a natureza da informação que manipulam, podem ser catalogados recorrendo à entidade "*External Service Category*".

A acessibilidade de um serviço depende de certas condicionantes. O administrador do HUEPS possui privilégios para desabilitar o serviço ou definir quais são os sistemas que podem usufruir da sua informação. O estado de funcionamento de um serviço (ativo ou desabilitado) é representado pela característica "*status*" em "*External Service*". A definição do acesso personalizado ao serviço é realizada pela entidade "*External Services Allowed Systems*". No entanto, um sistema externo só pode aceder aos serviços do HUEPS se estiver registado e autenticado. Assim, a entidade "*External Services System*" utiliza o nome do sistema e a sua palavra-passe para garantir a segurança da informação do HUEPS.

Um histórico de mensagens trocadas entre o HUEPS e os sistemas externos é uma ferramenta útil na medida em que permite verificar o resultado da interação entre os serviços e os sistemas. Desta forma é possível validar o funcionamento do módulo "Serviços Externos" e detetar acessos não autorizados ao HUEPS. A entidade "*External Services Message*" modela as características de uma mensagem, como o remetente, conteúdo e data de envio, de forma a arquivar a informação enviada aos sistemas externos.

A entidade "*HUEPS Configuration"* é utilizada para indicar o estado do sistema e algumas configurações essenciais ao seu funcionamento. O estado serviço de monitoração e a sua frequência de amostragem são alguns exemplos de configuração. Os campos "*name*" e "*password*" definem as credenciais que o HUEPS utiliza para se autenticar nos sistemas externos pertencentes à rede WiLOS.

O registo de todos os consumos energéticos verificados na habitação é possível através das entidades que definem a área "*Energy Consumption Information*". O consumo global da habitação, resultante da recolha constante de amostras, fica registado em "*Energy Consumption*". Aqui, dados como a frequência da rede energética, valor de tensão aplicado, corrente e potência fornecidos são

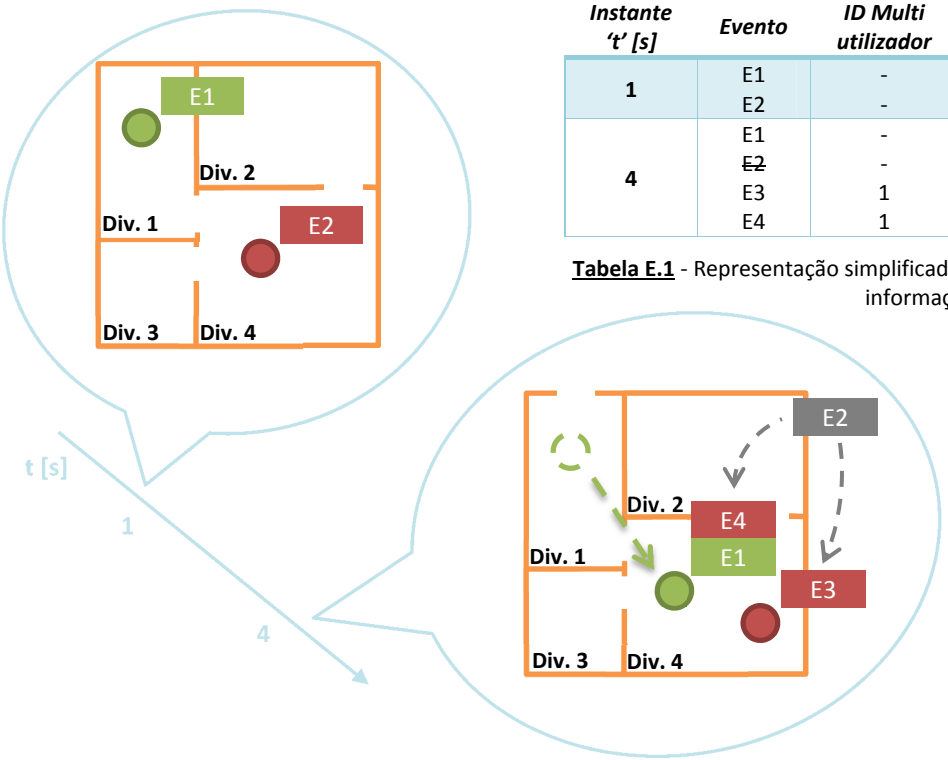
armazenados para consulta posterior. As entidades "*Energy Consumption Partial*", "*Energy Consumption Event*" e "*Energy Consumption Multi Event*" necessitam da introdução do conceito "Evento" e "Evento Multiutilizador" para demonstrar a sua importância no HUEPS.

Um "Evento" é qualquer ação inicial que origine a atribuição de uma variação de consumo energético a um utilizador numa determinada divisão. Estas ações são descritas pelas premissas do subcapítulo 4.2.1.2, sendo na sua maioria relacionadas com a movimentação de utilizadores. Assim, o objetivo de um evento é modelar o consumo verificado numa divisão e associá-lo somente a um utilizador. No entanto, existem situações em que uma ação obriga à repartição do consumo por vários utilizadores. Nestes casos, os eventos passam a denominar-se "Eventos Multiutilizador". A individualidade destes eventos continua a ser garantida e apenas se recorre a um identificador comum único para os interligar. Nas situações mais extremas pode ser necessário definir uma localização virtual de um utilizador, de modo a acomodar a atribuição ou a repartição de um evento. Para tal cria-se um elemento na entidade "*User Location*", afetando o valor do atributo "*realLocation*" de forma a indicar que aquela localização não é a posição atual do utilizador.

Um evento encontra-se ativo até ao instante em que se verifique uma variação de consumo energético negativo, cujo valor equipara-se ao valor de consumo energético associado ao evento (ou eventos nas situações "Multiutilizador"). Outro caso que obriga a terminação de um evento ocorre quando um evento futuro impossibilita a coexistência de eventos.

A definição de um evento no HUEPS é realizada pela entidade "*Energy Consumption Event*". Esta possui como características a variação de consumo verificada em kW ("*Energy Consumption*"), o número de utilizadores associado ao evento ("*nUsers*"), a referência da posição do utilizador ("*idUserLocation*"), a referência da multiplicidade de utilizadores no evento ("*idMultiEvent*") e o estado do evento, ativo ou desativo ("*status*"). O campo "*nUsers*" é utilizado para se obter a parcela de consumo energético do evento atribuído a um utilizador.

A entidade "*Energy Consumption Multi Event*" possibilita a gestão dos identificadores dos eventos multiutilizador, de modo a originar novos identificadores únicos e garantir a integridade dos eventos. A entidade "*Energy Consumption Partial*" efetua o elo de ligação entre os eventos ativos e o consumo energético global da habitação. Durante a amostragem do consumo energético deve-se verificar quais os eventos ativos e popular "*Energy Consumption Partial*" com as referências destes eventos. Desta forma, o consumo dos eventos ativos é registado ao longo do tempo e pode ser utilizado para determinar a duração total do evento e o consumo a ele associado. Seguidamente apresenta-se um caso de estudo para eliminar qualquer ambiguidade relativa às entidades descritas.



Instante 't' [s]	*Evento*	*ID Multi utilizador*	*Consumo Parcial [W]*	*Consumo Total [W]*
1	E1 E2	- -	70 30	100
4	E1 ~~E2~~ E3 E4	- - 1 1	70 ~~30~~ 15 15	100

Tabela E.1 - Representação simplificada das entidades em estudo e da informação a elas associada no decorrer do caso de estudo da figura E.2.

Figura E.2 - Caso de estudo referente à evolução do posicionamento dos utilizadores ao longo do tempo e respetiva distribuição de eventos.

No instante inicial (t = 1 s) os utilizadores encontram-se em divisões distintas e com dois eventos ativos, E1 e E2. Como estes eventos encontram-se ativos, os consumos a eles associados são registados em "Consumo Parcial", sendo coerentes com o consumo total verificado na habitação. No instante seguinte (t = 4 s) verifica-se que os utilizadores encontram-se na mesma divisão (Div. 4). De acordo com a premissa 7, o consumo de uma divisão é repartido por todos os ocupantes da divisão. Como o evento E2 possui a informação referente ao consumo da divisão em comum, este é utilizado para gerar os eventos E3 e E4. Cada um destes eventos fica associado a um utilizador de forma a individualizar o consumo. O identificador multiutilizador '1' é atribuído a E3 e E4 para interligar ambos os eventos. O evento E2 é desativado devido à sua incompatibilidade com os dois eventos criados. Uma vez que neste instante só E1, E3 e E4 se encontram ativos, o consumo parcial registará somente os valores de consumo a eles associados.

Como se pode verificar, a manipulação da informação nestas entidades é crucial para se efetuar a distribuição do consumo energético pelos utilizadores e pelas divisões da habitação.

Anexo F: Interface Gráfica - mWiLOS

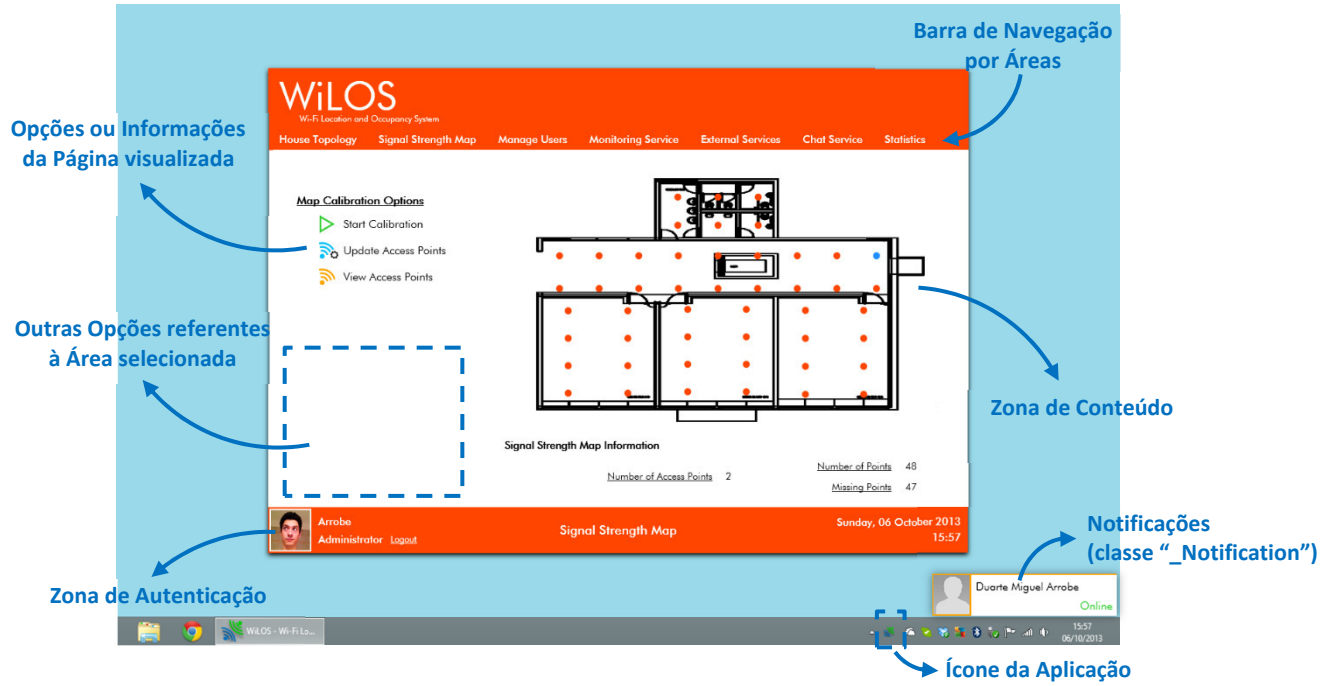


Figura F.1 - Organização e estruturação da interface gráfica do mWiLOS, disponibilizada pelo módulo UI.

A barra de navegação varia o número de áreas do mWiLOS disponibilizadas ao utilizador consoante o seu nível de acesso. As páginas que compõem as diversas áreas do mWiLOS também são dinâmicas, restringindo ou disponibilizando funcionalidades de acordo o utilizador autenticado. A maioria destas funcionalidades são disponibilizadas nas zonas "Opções ou Informações da Página visualizada" e "Outras Opções referentes à Área selecionada".

O ícone da aplicação permite a execução do WiLOS sem utilizar muitos recursos da barra de ferramentas do sistema operativo. Se o mWiLOS estiver a interagir com o utilizador, a aplicação é visível na barra de ferramentas. Caso o utilizador minimize a aplicação, esta desaparece da barra de ferramentas, sendo apenas alcançável através do seu ícone. Desta forma, o mWiLOS continua a monitorar os utilizadores do WiLOS e a barra de tarefas fica livre para outras aplicações.

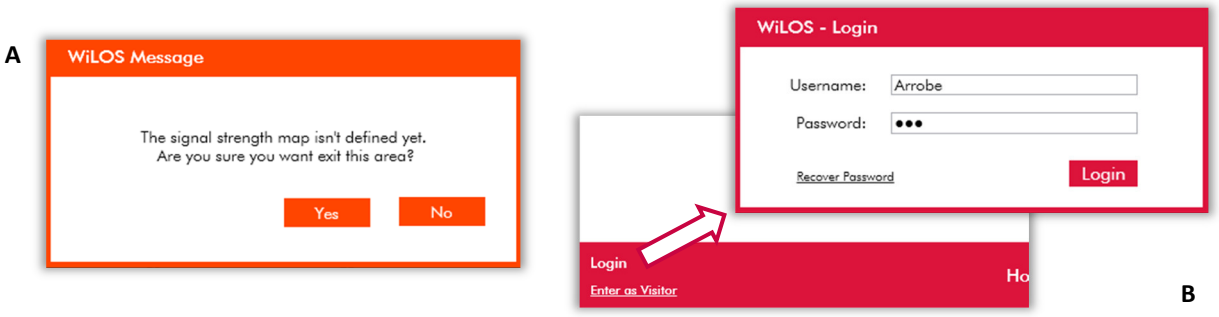


Figura F.2 - (**A**): Utilização das classes "_popUp" e "_popUpMessage" para se efetuar uma tomada de decisão. (**B**): Processo de autenticação. Utilização de "_popUp" e "_H_Login" para a introdução de credenciais de um utilizador.

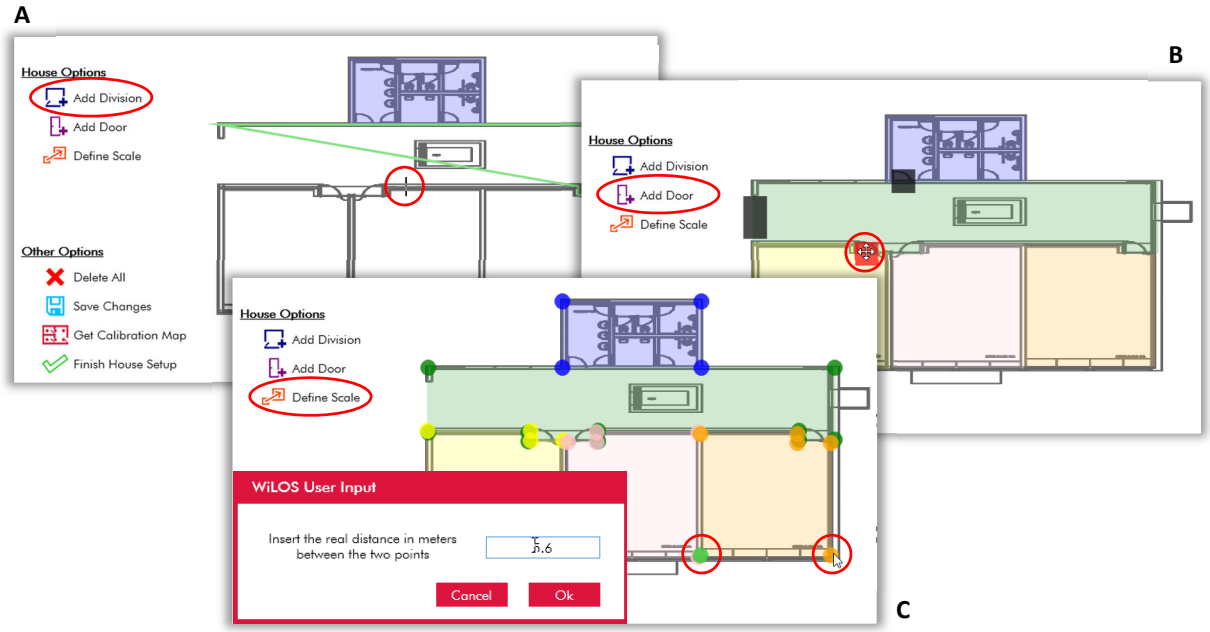


Figura F.3 - Visualização das opções para a definição da tipologia disponibilizadas pela classe "_HT_Add_House3". (**A**): Traçar das divisões da habitação. (**B**): Criação e manipulação de portas. (**C**): Definição da escala da habitação.

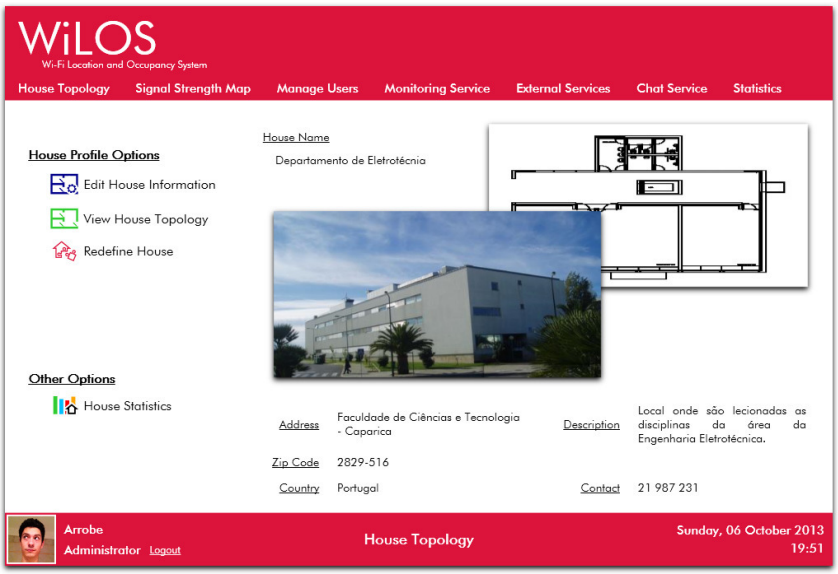


Figura F.4 - Página inicial apresentada ao utilizador durante o acesso à área "*House Topology*", implementada pelo HTM.

Figura F.5 - Interface gráfica implementada pela UI da classe ""_SSM_AccessPointMap". Visualização dos mapas de potência do sinal obtidos para cada *AP*.

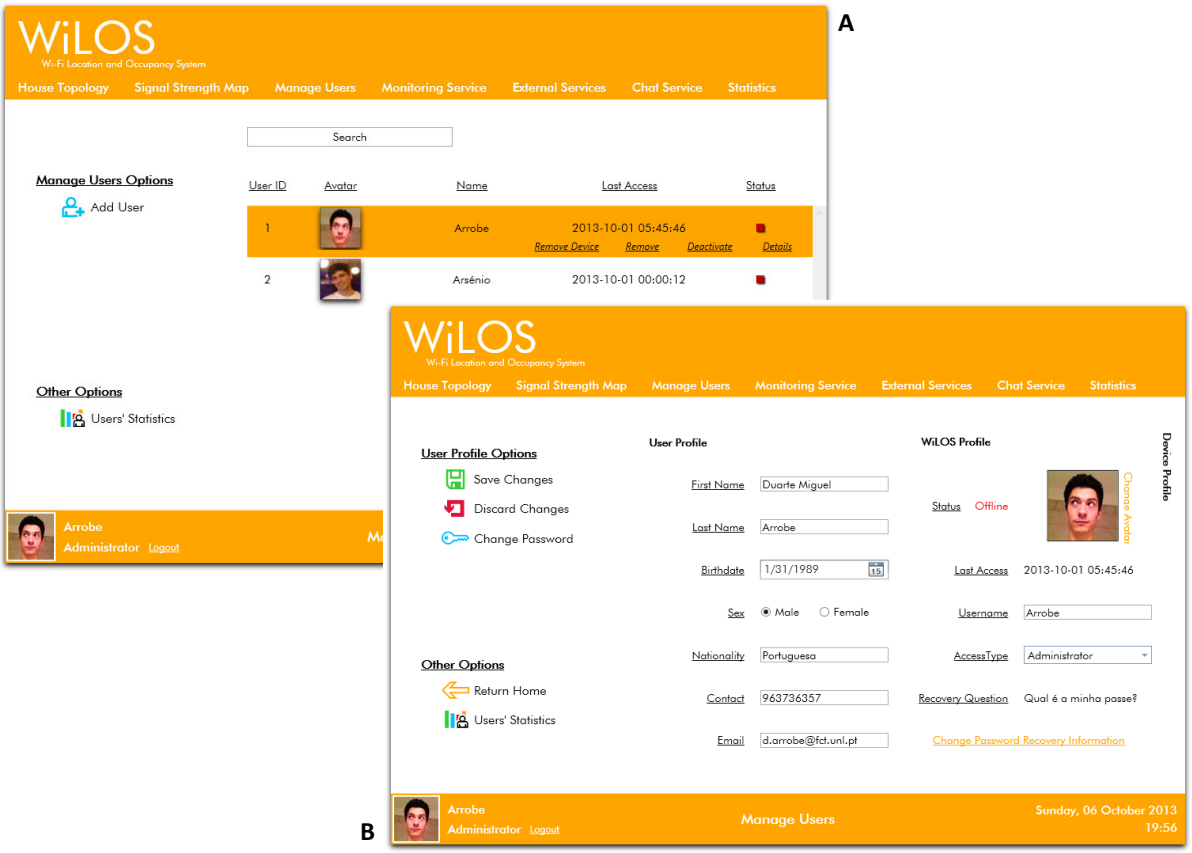


Figura F.6 - Exemplos da interface gráfica disponibilizada pelo UM através da UI. (**A**): Classe "_MU_Home", visualização de 2 utilizadores registados no WiLOS. (**B**): Classe "_MU_UserProfile", visualização e edição dos perfis do utilizador.

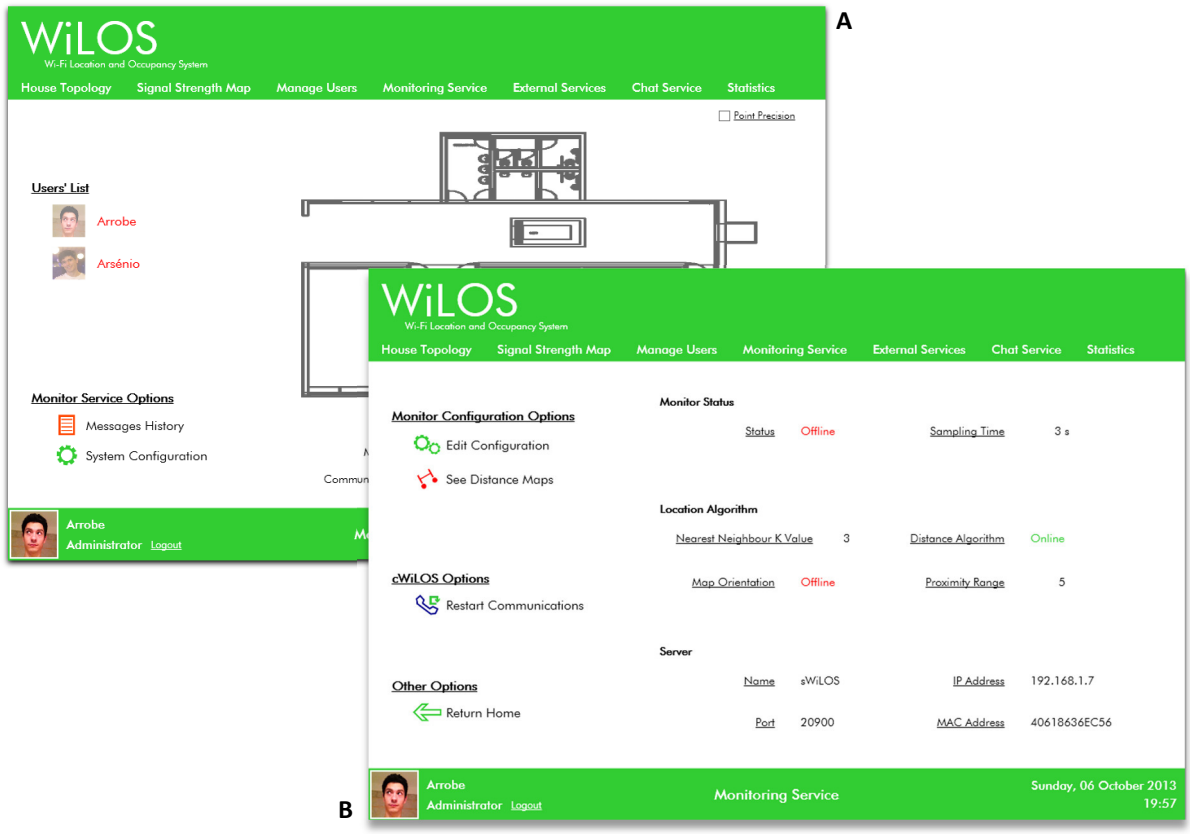


Figura F.7 - Visualização da UI disponibilizada pelas classes "_MS_MonitorHome" (**A**) e "_MS_MonitorConfiguration" (**B**).

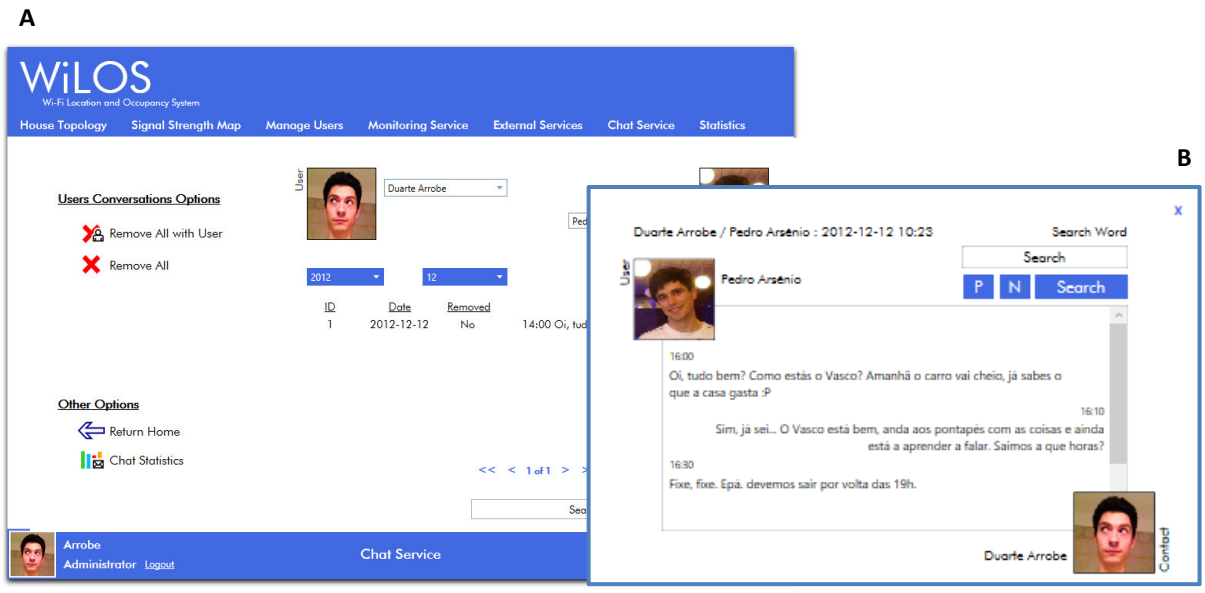


Figura F.8 - Exemplos da interface gráfica disponibilizada pelo CSM através da UI. (**A**): Classe "_CS_UsersConversations", listagem das conversações efetuadas entre 2 utilizadores. (**B**): Classe "_CS_Conversation", visualização de uma conversação.

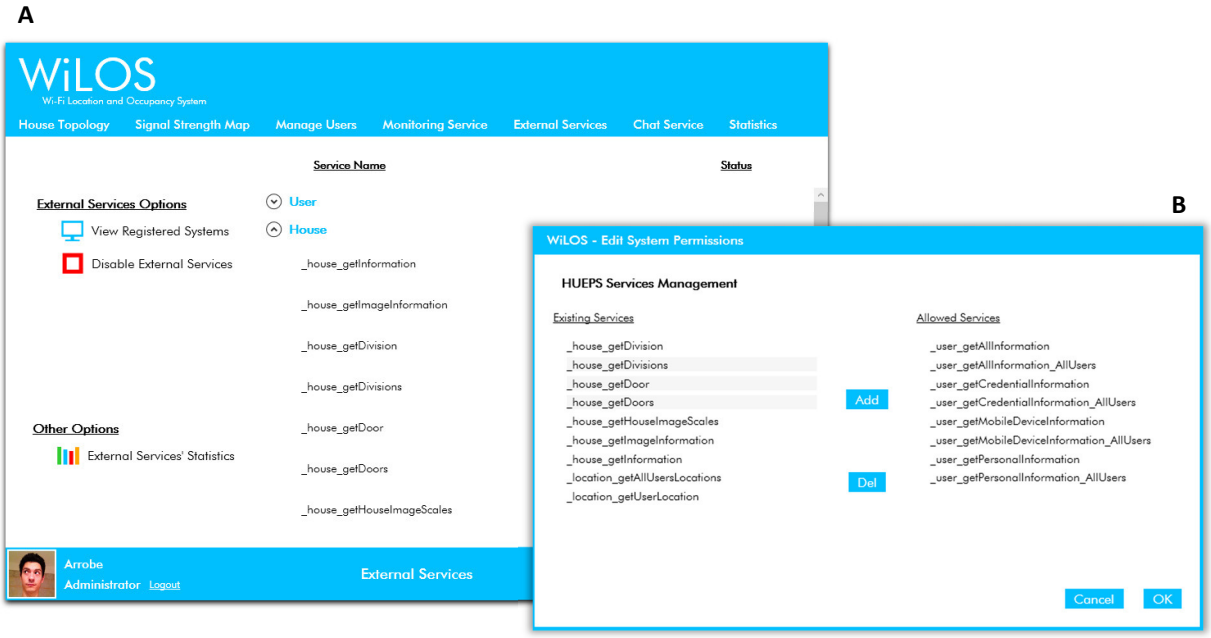


Figura F.9 - Visualização da UI disponibilizada pelas classes "_esHome" (**A**) e "_esEditRules" (**B**).

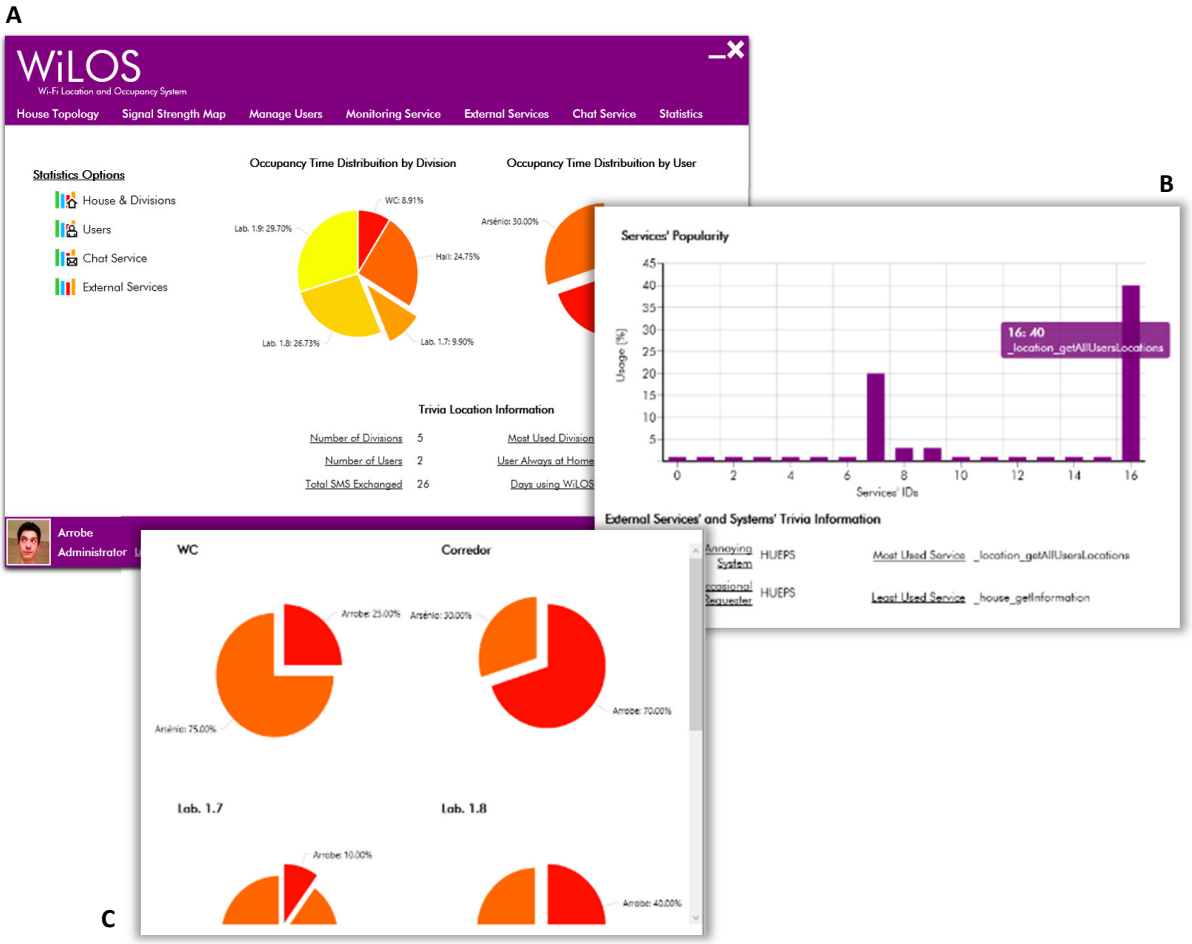


Figura F.10 - Visualização de vários elementos estatísticos gerados pelo SM e apresentados ao utilizador através da UI. (**A**): *Layout* da interface gráfica da área "Estatística" dado por "_S_Home". (**B**): Gráficos de barras usados para demonstrar popularidade ou utilização, neste caso, a popularidade dos serviços externos disponibilizados pelo WiLOS. (**C**) Gráficos circulares utilizados para representar distribuições.

Anexo G: Interface Gráfica - sWiLOS

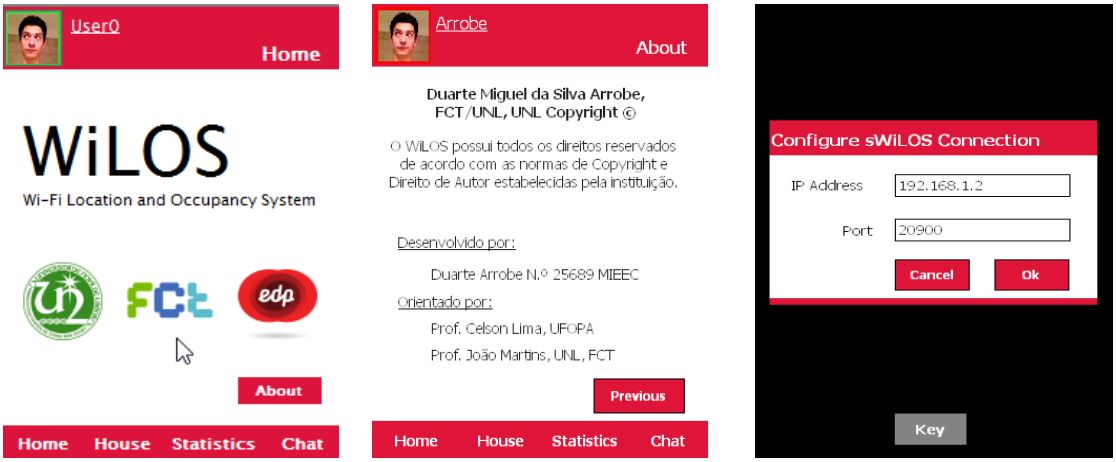


Figura G.1 - Organização das classes que definem o *layout* gráfico da UI e a página de entrada do sWiLOS.

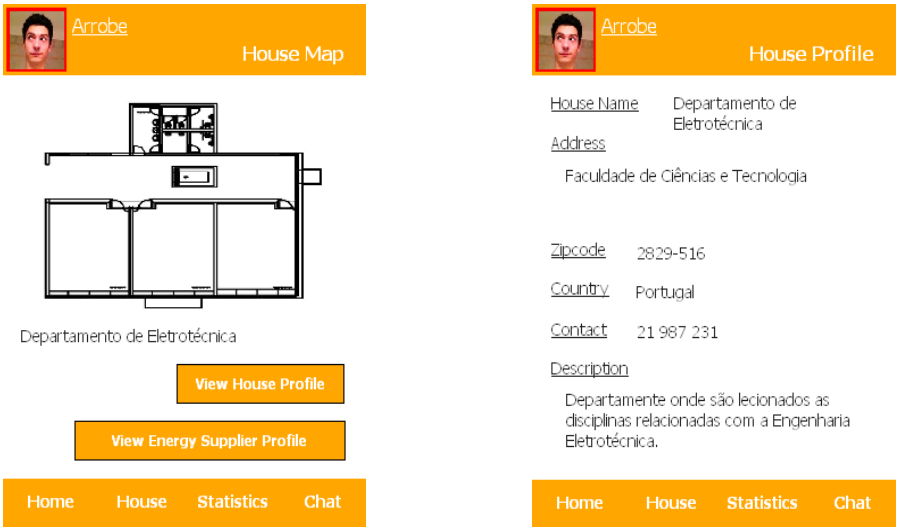


Figura G.2 - Visualização da UI implementada pelas classes "_H_HouseMap" e "_H_HouseProfile".

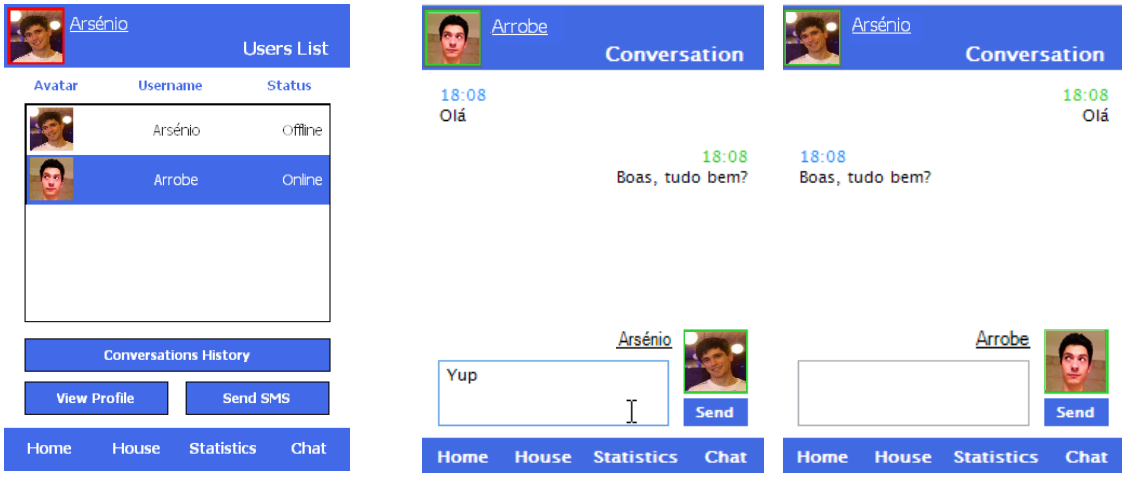


Figura G.3 - Visualização da interface gráfica disponibilizada pelas classes "_U_Home" e "_C_Conversation".

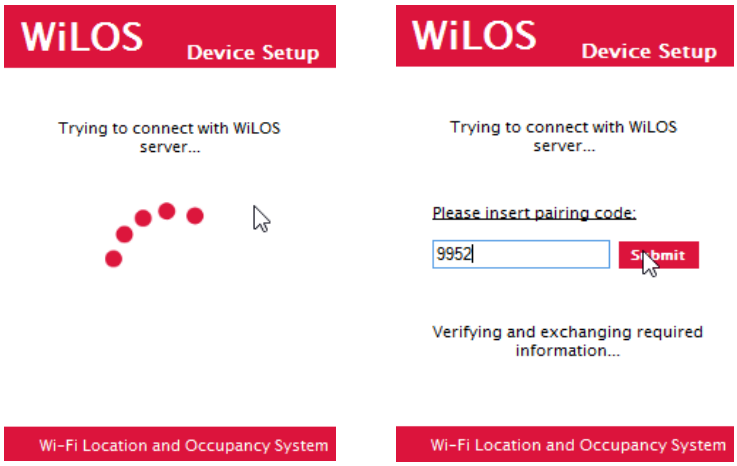


Figura G.4 - Processo de emparelhamento de um dispositivo móvel sWiLOS através da classe "_SetupPair.

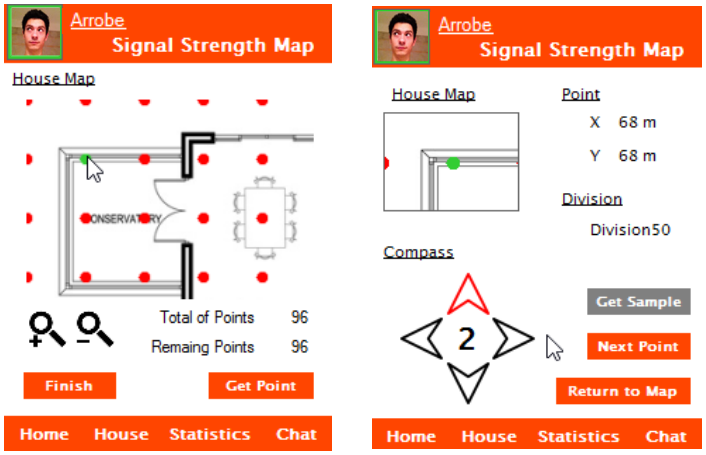


Figura G.5 - Interface gráfica utilizada para se efetuar a calibração do mapa de sinal de potência. Da esquerda para a direita: classe "_SSM_HouseMap" e classe "_SSM_Point".

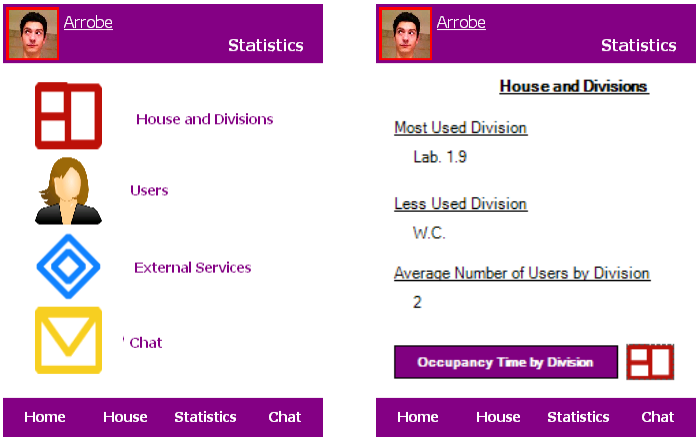


Figura G.6 - Exemplos da UI disponibilizada pelo SM através das classes "_S_Home" e "_S_House".

Anexo H: Interface Gráfica - HUEPS

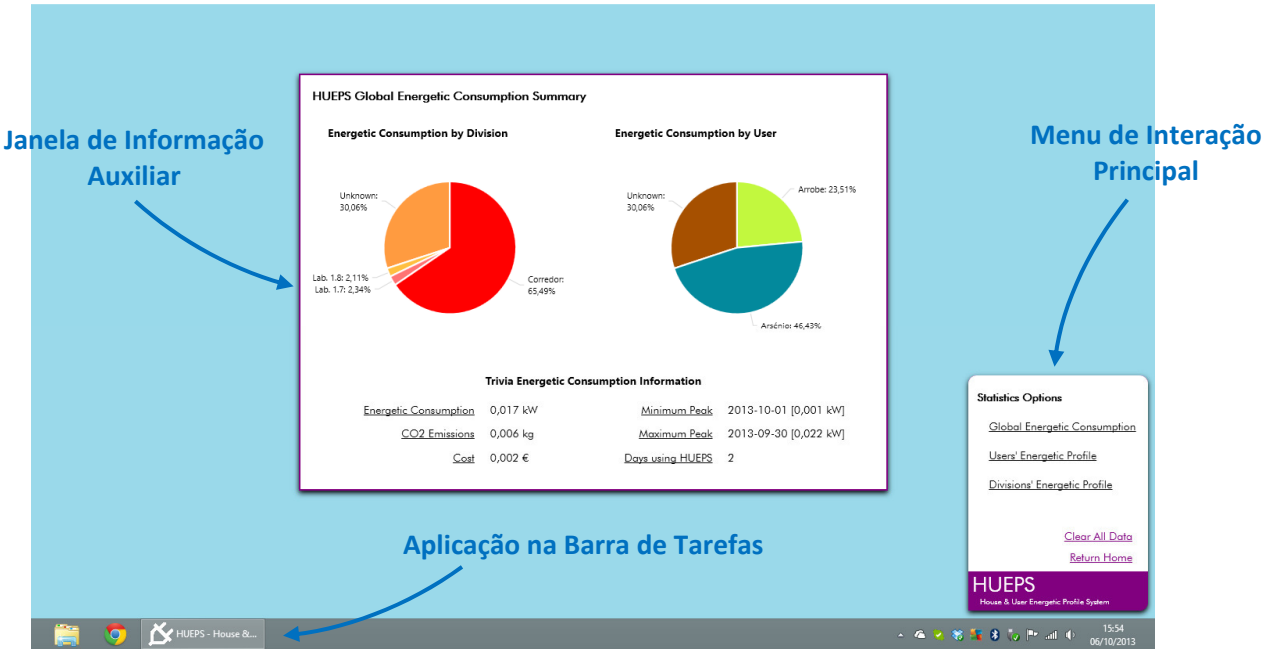


Figura H.1 - Organização e estruturação da interface gráfica do HUEPS, disponibilizada pela UI.

O ícone da aplicação é utilizado para indicar que o HUEPS se encontra em execução no computador. Também permite minimizar ou encerrar o HUEPS sempre que o utilizador desejar. Note-se que o fim da execução do HUEPS não afeta o estado do ESM da camada Controlo. Caso se deseje que o HUEPS deixe de fornecer os seus serviços a outras entidades externas, o utilizador terá que se deslocar à respetiva área e requerer a desativação do ESM.

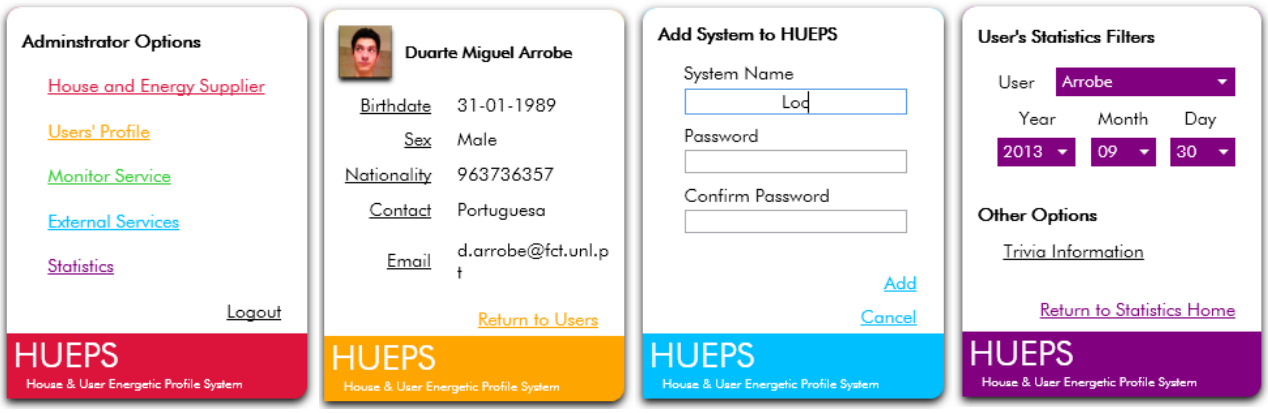


Figura H.2 - *Layouts* gráficos possíveis para o menu de interação principal do HUEPS. Da esquerda para a direita, navegação, consulta de informação, criação de informação e híbrido.

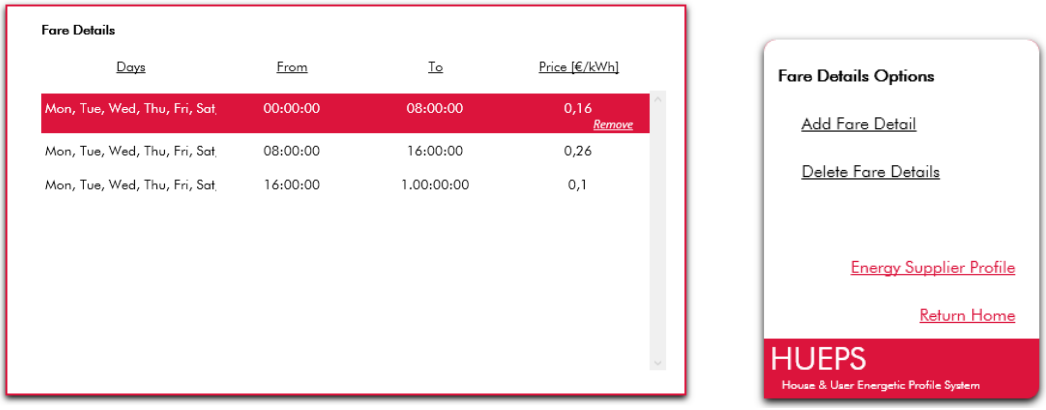


Figura H.3 - Gestão de tarifas que compõem o tarifário aplicado à habitação. Interação entre a UI e o HESM efetuada através das classes "_energySupplierFareDetail" e "_energySupplierFareDetailOptions".

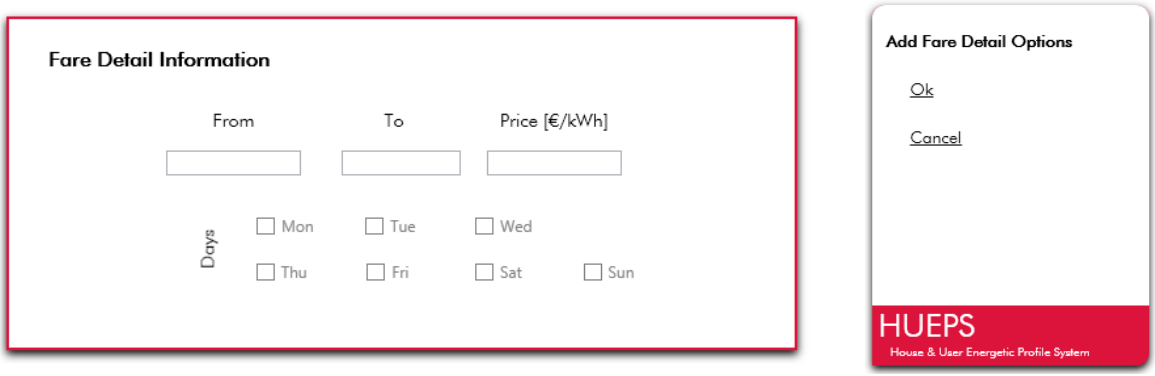


Figura H.4 - Adição de uma nova tarifa ao tarifário. Interação entre a UI e o HESM efetuada através das classes "_energySupplierFareDetailAdd" e "_energySupplierFareDetailAddOptions".

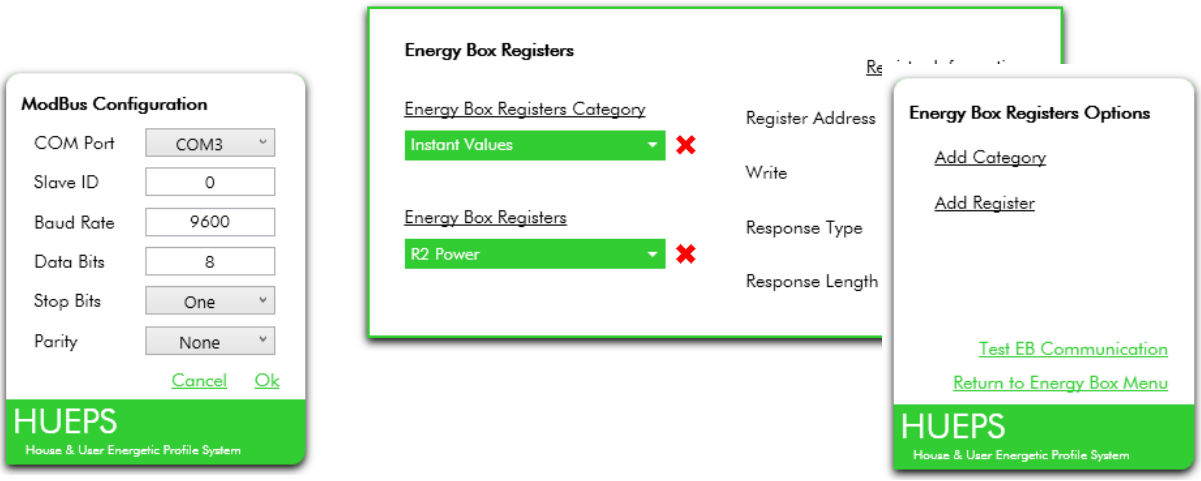


Figura H.5 - Interface gráfica implementada pela UI que permite o acesso às funcionalidades do MM. Classes "_configureModbusProtocol" (esquerda), "_energyboxRegisters" e "_energyboxRegistersOptions" (direita).

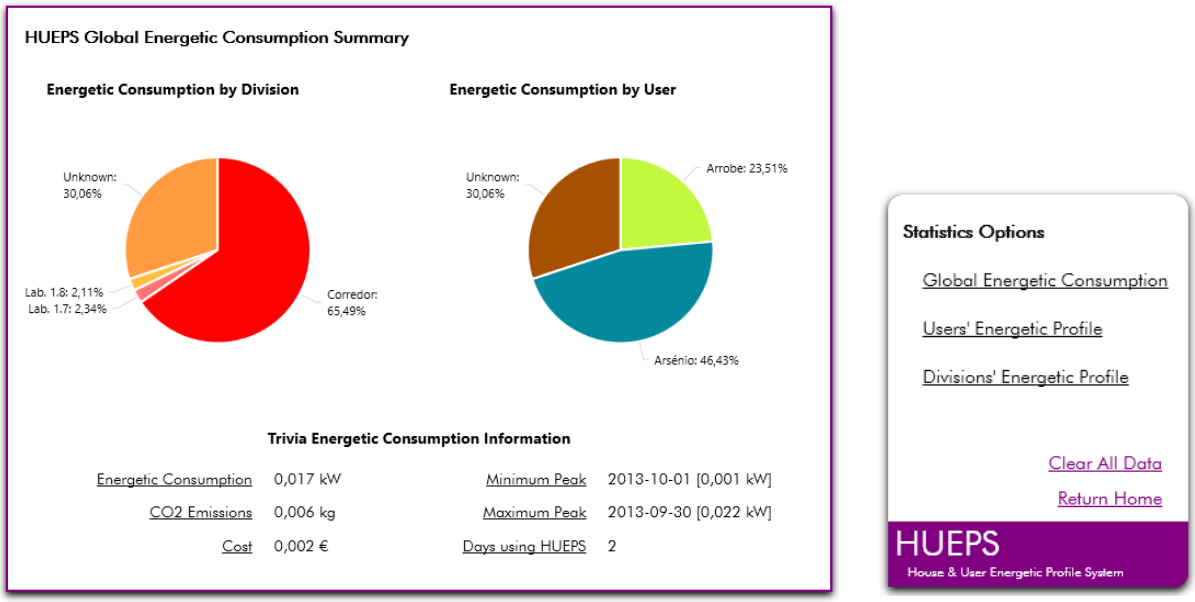


Figura H.6 - Visualização da UI implementada pelas classes "_statisticsHome" e "_statisticsHomeOptions".

Figura H.7 - Visualização da UI disponibilizada pelas classes "_statisticsUser" e "_statisticsUserOptions". Opções para a visualização da informação relativa ao consumo energético de um utilizador: Gráfico de Linha e Gráfico de Barras.

Anexo I: Lista de Serviços Externos - WiLOS

User

_user_getPersonalInformation

Envia os dados referentes à informação pessoal de um utilizador WiLOS específico para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idUser: Identificador do utilizador que se pretende consultar

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "User Personal Information", definido na classe _WiLOS_Objects.cs

_user_getPersonalInformation_AllUsers

Envia os dados referentes à informação pessoal de todos os utilizadores WiLOS para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

Nenhum

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "*List*< User Personal Information >", definido na classe _WiLOS_Objects.cs

_user_getCredentialInformation

Envia os dados referentes às credenciais WiLOS de um utilizador WiLOS específico para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idUser: Identificador do utilizador que se pretende consultar

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "User Credential Information", definido na classe _WiLOS_Objects.cs

_user_getCredentialInformation_AllUsers

Envia os dados referentes às credenciais WiLOS de todos os utilizadores WiLOS para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

Nenhum

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "*List*<User Credential Information>", definido na classe _WiLOS_Objects.cs

_user_getMobileDeviceInformation

Envia os dados referentes ao dispositivo móvel de um utilizador WiLOS específico para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idUser: Identificador do utilizador que se pretende consultar

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "User Mobile Information", definido na classe _WiLOS_Objects.cs

_user_getMobileDeviceInformation_AllUsers

Envia os dados referentes aos dispositivos móveis de todos os utilizadores WiLOS para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

Nenhum

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "*List*<User Mobile Information>", definido na classe _WiLOS_Objects.cs

_user_getAllInformation

Envia toda a informação associada a um utilizador WiLOS específico para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idUser: Identificador do utilizador que se pretende consultar

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "*List*<User, Credential, Mobile>", definido na classe _WiLOS_Objects.cs

_user_getAllInformation_AllUsers

Envia toda a informação de todos os utilizadores WiLOS para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

Nenhum

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "*List*<*List*<User, Credential, Mobile>>", definido na classe _WiLOS_Objects.cs

House

_house_getInformation

Envia os dados referentes à habitação WiLOS (incluindo imagens) para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

Nenhum

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "House Information", definido na classe _WiLOS_Objects.cs

_house_getImageInformation

Envia os dados referentes às escalas das imagens da habitação obtidas através do WiLOS para um sistema externo. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idHouse: Identificador da habitação que se deseja consultar

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "House Scales Information", definido na classe _WiLOS_Objects.cs

_house_getDivision

Envia os dados referentes a uma divisão específica da habitação WiLOS para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idDivision: Identificador da divisão que se deseja consultar

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Division Information", definido na classe _WiLOS_Objects.cs

_house_getDivisions

Envia os dados referentes a a todas as divisões da habitação WiLOS para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

Nenhum

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "*List*<Division Information>", definido na classe _WiLOS_Objects.cs

_house_getDoor

Envia os dados referentes a uma porta específica da habitação WiLOS para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idDoor: Identificador da porta que se deseja consultar

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Door Information", definido pela classe _WiLOS_Objects.cs

_house_getDoors

Envia os dados referentes a todas as portas da habitação WiLOS para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

Nenhum

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "*List*<Door Information>", definido na classe _WiLOS_Objects.cs

User Location

_location_getUserLocation

Envia os dados referentes à última localização de um utilizador WiLOS para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idUser: Identificador do utilizador cujas localizações se pretendem obter

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "User Location Information", definido na classe _WiLOS_Objects.cs

_location_getUserPreviousDivisionID

Envia os dados referentes à última localização de um utilizador, relativamente a uma determinada divisão, para um sistema externo. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idUser: Identificador do utilizador cujas localizações se pretendem obter

int idDivision: Identificador da divisão utilizada como referência

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "User Location Information", definido na classe _WiLOS_Objects.cs

_location_getAllUsersLocationslocation

Envia os dados referentes às últimas localizações de todos os utilizadores WiLOS para um sistema externo. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

Nenhum

Resposta

Um vetor de 2 posições: *ans[2]*.

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "*Object*{ *List*<User Location Information> as *Current Locations*, List<User Location Information> *as Previous Locations*}", definido na classe WiLOS_Objects.cs

Anexo J: Lista de Serviços Externos - HUEPS

User

_user_getUser

Envia os dados referentes a um utilizador HUEPS específico para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idUser: Identificador do utilizador que se pretende consultar

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "User", definido na classe _HUEPS_Objects.cs

_user_getAllUsers

Envia os dados referentes a todos os utilizadores HUEPS para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

Nenhum

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "User List", definido na classe _HUEPS_Objects.cs

Energy Supplier

_energySupplier_getInformation

Envia os dados referentes ao fornecedor de energia da habitação HUEPS para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

Nenhum

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Energy Supplier Information", definido na classe _HUEPS_Objects.cs

_energySupplier_getFareDetail

Envia os dados referentes a uma tarifa específica, aplicada na habitação HUEPS, para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idFareDetail: Identificador da tarifa que se pretende consultar

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Energy Supplier Fare Detail", definido na classe _HUEPS_Objects.cs

_energySupplier_getEnergySupplier

Envia todos os dados referentes ao fornecedor de energia da habitação, incluindo tarifas, para um sistema externo. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

Nenhum

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Energy Supplier", definido na classe _HUEPS_Objects.cs

House

_house_getDivision

Envia os dados referentes uma divisão específica da habitação HUEPS para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idDivision: Identificador da divisão que se pretende consultar

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Division", definido na classe _HUEPS_Objects.cs

_house_getDivisions

Envia os dados referentes a todas as divisões da habitação HUEPS para o sistema externo que requisitou este serviço. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

Nenhum

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Division List", definido na classe _HUEPS_Objects.cs

Global Energy Consumption

_globalEnergyConsumption_getYears

Envia a informação temporal (anos) que permite a consulta dos dados estatísticos referentes ao consumo global da habitação HUEPS. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

Nenhum

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Time _List (Years)", definido na classe _HUEPS_Objects.cs

_globalEnergyConsumption_getMonths

Envia a informação temporal (meses) que permite a consulta dos dados estatísticos referentes ao consumo global da habitação HUEPS. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

string year: Ano utilizado como referência temporal

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Time _List (Months)", definido na classe _HUEPS_Objects.cs

_globalEnergyConsumption_getDays

Envia a informação temporal (dias) que permite a consulta dos dados estatísticos referentes ao consumo global da habitação HUEPS. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

string year: Ano utilizado como referência temporal
string month: Mês utilizado como referência temporal

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Time _List (Days)", definido na classe _HUEPS_Objects.cs

_globalEnergyConsumption_getTime

Envia toda a informação temporal que permite a consulta dos dados estatísticos referentes ao consumo global da habitação HUEPS. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

Nenhum

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Time Time Object", definido na classe _HUEPS_Objects.cs

_globalEnergyConsumption_getConsumptionByDate

Calcula e envia a distribuição energética global da habitação, utilizado como ponto de referência indicações temporais. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

string year: Referência temporal
string day: Referência temporal
string month: Referência temporal
string _minutesResolution: Resolução dos intervalos de tempo

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')
ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Energy Consumption List", definido na classe _HUEPS_Objects.cs

_globalEnergyConsumption_getConsumptionByUser

Calcula e envia a distribuição energética global da habitação por utilizador, utilizado como ponto de referência indicações temporais. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

string year: Referência temporal
string day: Referência temporal
string month: Referência temporal
string _hourMinute: "HH:mm" específicos
string _minutesResolution: Resolução dos intervalos de tempo

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')
ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Energy Consumption List", definido na classe _HUEPS_Objects.cs

_globalEnergyConsumption_getConsumptionByDivision

Calcula e envia a distribuição energética global da habitação por divisão, utilizado como ponto de referência indicações temporais. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

string year: Referência temporal
string day: Referência temporal
string month: Referência temporal
string _hourMinute: "HH:mm" específicos
string _minutesResolution: Resolução dos intervalos de tempo

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')
ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Energy Consumption List", definido na classe _HUEPS_Objects.cs

User Energy Consumption

_userEnergyConsumption_getYears

Envia a informação temporal (dias) que permite a consulta dos dados estatísticos referentes ao consumo de um utilizador HUEPS. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idUser: Identificador do utilizador a consultar

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Time _List (Years)", definido na classe _HUEPS_Objects.cs

_userEnergyConsumption_getMonths

Envia a informação temporal (meses) que permite a consulta dos dados estatísticos referentes ao consumo de um utilizador HUEPS. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idUser: Identificador do utilizador a consultar
string year: Ano utilizado como referência temporal

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Time _List (Months)", definido na classe _HUEPS_Objects.cs

_userEnergyConsumption_getDays

Envia a informação temporal (dias) que permite a consulta dos dados estatísticos referentes ao consumo de um utilizador HUEPS. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idUser: Identificador do utilizador
string year: Referência temporal *string month*: Referência temporal

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Time _List (Days)", definido na classe _HUEPS_Objects.cs

_userEnergyConsumption_getTime

Envia toda a informação temporal que permite a consulta dos dados estatísticos referentes ao consumo de um utilizador HUEPS. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idUser: Identificador do utilizador

Resposta

Um vetor de 2 posições: *ans[2].*

ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')

ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Time Time Object", definido na classe _HUEPS_Objects.cs

_userEnergyConsumption_getConsumption

Calcula e envia a distribuição energética de um utilizador HUEPS, utilizado como ponto de referência indicações temporais. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idUser: Identificador do utilizador
string year: Referência temporal
string month: Referência temporal
string day: Referência temporal
string _minutesResolution: Resolução dos intervalos

Resposta

Um vetor de 2 posições: *ans[2].*
ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')
ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Energy Consumption List", definido na classe _HUEPS_Objects.cs

_userEnergyConsumption_getUserConsumptionByDivision

Calcula e envia a distribuição energética de um utilizador HUEPS por divisão, utilizado como ponto de referência indicações temporais. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idUser: Identificador do utilizador
string year: Referência temporal
string month: Referência temporal
string day: Referência temporal
string _hourMinute: "HH:mm" específicos
string _minutesResolution: Resolução dos intervalos

Resposta

Um vetor de 2 posições: *ans[2].*
ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')
ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Energy Consumption List", definido na classe _HUEPS_Objects.cs

Division Energy Consumption

_divisionEnergyConsumption_getYears

Envia a informação temporal (anos) que permite a consulta dos dados estatísticos referentes ao consumo de uma divisão HUEPS. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idDivision: Identificador da divisão a consultar

Resposta

Um vetor de 2 posições: *ans[2].*
ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')
ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Time _List (Years)", definido na classe _HUEPS_Objects.cs

_divisionEnergyConsumption_getMonths

Envia a informação temporal (meses) que permite a consulta dos dados estatísticos referentes ao consumo de uma divisão HUEPS. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idDivision: Identificador da divisão
string year: Ano utilizado como referência temporal

Resposta

Um vetor de 2 posições: *ans[2].*
ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')
ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Time _List (Months)", definido na classe _HUEPS_Objects.cs

_divisionEnergyConsumption_getDays

Envia a informação temporal (dias) que permite a consulta dos dados estatísticos referentes ao consumo de uma divisão HUEPS. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idDivision: Identificador da divisão
string year: Referência temporal *string month*: Referência temporal

Resposta

Um vetor de 2 posições: *ans[2].*
ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')
ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Time _List (Days)", definido na classe _HUEPS_Objects.cs

_divisionEnergyConsumption_getTime

Envia toda a informação temporal que permite a consulta dos dados estatísticos referentes ao consumo de uma divisão HUEPS. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idDivision: Identificador da divisão a consultar

Resposta

Um vetor de 2 posições: *ans[2].*
ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')
ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Time Time Object", definido na classe _HUEPS_Objects.cs

_divisionEnergyConsumption_getConsumption

Calcula e envia a distribuição energética de uma divisão HUEPS, utilizado como ponto de referência indicações temporais. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idDivision: Identificador da divisão *string year*: Referência temporal
string month: Referência temporal *string day*: Referência temporal
string _minutesResolution: Resolução dos intervalos

Resposta

Um vetor de 2 posições: *ans[2].*
ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')
ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Energy Consumption List", definido na classe _HUEPS_Objects.cs

_divisionEnergyConsumption_getDivisionConsumptionByUser

Calcula e envia a distribuição energética de uma divisão HUEPS por utilizador, utilizado como ponto de referência indicações temporais. São verificadas a autenticidade e o o nível de autorização do sistema externo.

Parâmetros de Entrada

int idDivision: Identificador da divisão *string year*: Referência temporal
string month: Referência temporal *string day*: Referência temporal
string _hourMinute: "HH:mm" específicos *string _minutesResolution*: Resolução dos intervalos

Resposta

Um vetor de 2 posições: *ans[2].*
ans[0] – Código de Erro (0 - Sucesso ; 1 - Serviço *Offline* ; 2 - Erro 'Autenticação' ; 3 - Erro 'Autorização')
ans[1] – '*Null*' ou uma *string* XML que encapsula um objeto do tipo "Energy Consumption List", definido na classe _HUEPS_Objects.cs

Anexo K: *Webservices*

Definição do *Webservice* "_location_getAllUsersLocations", WiLOS Simulator

```
[WebMethod(Description = "Returns: [0] - 0 'Success', 1 'Service Offline' , 2 'Authetification
   Error' , 3 'Not Authorized' || [1] - 'XML Object{List-User Location Information- as
   CurrentLocations, List-User Location Information- as PreviousLocations}' String or null ")]
[SoapHeader("_externalCredentials")]
public object[] _location_getAllUsersLocations(DateTime _currentDate)
{
   // Confirms if the external system has access to the web service
   int _errorCode = _DatabaseSimulator._ExternalServices._Rule._checkSystemAuthentification(
                    _externalCredentials._systemName, _externalCredentials._password,
                    "_location_getUsersLocations");

   if (_errorCode != 0)
      return new object[2] { _errorCode, null };

   // Access Granted
   _DatabaseSimulator._ExternalServices._System._updateLastAccess(_externalCredentials._systemName,
      DateTime.Now.ToString());

   // Get users list
   List<int> _usersID = _DatabaseSimulator._UserLocation._selectAllUsers();
   if (_usersID == null)
      return new object[2] { 0, null };
   if (_usersID.Count == 0)
      return new object[2] { 0, null };

   #region Variables

   List<_WiLOS_HUEPS_Objects._UserLocation> _previousUsersLocations = new
      List<_WiLOS_HUEPS_Objects._UserLocation>();
   List<_WiLOS_HUEPS_Objects._UserLocation> _currentUsersLocations = new
      List<_WiLOS_HUEPS_Objects._UserLocation>();
   _WiLOS_HUEPS_Objects._UserLocation _currentPosition;
   List<object> _currentPositionList;
   _WiLOS_HUEPS_Objects._UserLocation _previousPosition;
   List<object> _previousPositionList;
   DateTime _date = _currentDate;
   object[] _positions = new object[2];
   DateTime _dateReceived;

   #endregion

   foreach (int _userID in _usersID)
   {
      // Get Previous location
      _positions = _DatabaseSimulator._UserLocation._selectLastUsersPosition(_userID, _date, 8);

      if (_positions[0] == null && _positions[1] == null)
      {
         // User Offline
         _currentPosition = new _WiLOS_HUEPS_Objects._UserLocation();
         _previousPosition = new _WiLOS_HUEPS_Objects._UserLocation();
         _currentUsersLocations.Add(_currentPosition);
         _previousUsersLocations.Add(_previousPosition);

         continue;
      }

      // At least one location exists
      _currentPositionList = (List<object>)_positions[0];
      _currentPosition = new _WiLOS_HUEPS_Objects._UserLocation();
      _currentPosition._idUserLocation = Convert.ToInt32(_currentPositionList[0]);
      _currentPosition._idUser = Convert.ToInt32(_currentPositionList[1]);
      _currentPosition._idDivision = Convert.ToInt32(_currentPositionList[2]);
      _dateReceived = Convert.ToDateTime(_currentPositionList[3]);
      _currentPosition._date = _dateReceived.ToString("yyyy-MM-dd HH:mm:ss");
      if (DateTime.Compare(_date.AddSeconds(-5), _dateReceived) <= 0)
```

```
        {
            // User still online
            _previousPosition = new _WiLOS_HUEPS_Objects._UserLocation();
            if (_positions[1] == null)
            {
                // User was offline
                _currentPosition._movement = true;
                _previousPosition._idUser = _currentPosition._idUser;
            }
            else
            {
                // User was online, therefore he has a previous location
                _previousPositionList = (List<object>)_positions[1];
                _previousPosition._idUserLocation = Convert.ToInt32(_previousPositionList[0]);
                _previousPosition._idUser = Convert.ToInt32(_previousPositionList[1]);
                _previousPosition._idDivision = Convert.ToInt32(_previousPositionList[2]);
                _previousPosition._date = Convert.ToDateTime(_previousPositionList[3]).ToString("yyyy-
                                          MM-dd HH:mm:ss");

                // Check if movement occured between divisions
                if (_previousPosition._idDivision != _currentPosition._idDivision)
                    _currentPosition._movement = true;
            }
        }
        else
        {
            // User is now offline, movement occured
            _previousPosition = _currentPosition;
            _currentPosition = new _WiLOS_HUEPS_Objects._UserLocation();
            _currentPosition._idUser = _previousPosition._idUser;
            _currentPosition._movement = true;
        }

        // Add info to lists
        _currentUsersLocations.Add(_currentPosition);
        _previousUsersLocations.Add(_previousPosition);

    }

    // Check data
    if (_currentUsersLocations.Count == 0)
        return new object[2] { 0, null };

    // Create the object to serialize
    _WiLOS_HUEPS_Objects._UsersLocationsList _ull = new _WiLOS_HUEPS_Objects._UsersLocationsList();
    _ull._currentUsersLocations = _currentUsersLocations;
    _ull._previousUsersLocations = _previousUsersLocations;

    // Serialize and "send"
    string _xmlObject = _xmlOperations._serializeToXML(_ull);

    return new object[2] { 0, _xmlObject };

    // End Function
}
```

Exemplo de Consumo de um *Webservice*

```
   using HUEPS._ExternalServices;
   using System.Net;
   using System.Security.Cryptography.X509Certificates;

...

   List<_WiLOS_HUEPS_Objects._UserLocation> _usersLocationsCurrent = null;
   object[] _dataReceived = null;         // Object to store the webservice result

   try
   {
     // Validate Certificate
     ServicePointManager.ServerCertificateValidationCallback = new
        System.Net.Security.RemoteCertificateValidationCallback(CheckValidationResult);

     // Set Service Authorization and Authentication Header
     WiLOSSimulatorExternalServices._WiLOS_Simulator_ExternalServices _WiLOS_Services = new ~
        WiLOSSimulatorExternalServices._WiLOS_Simulator_ExternalServices();  // Simulator
     WiLOSSimulatorExternalServices._ExternalSystemCredentialsValue = this._HUEPS_Credentials;

     // Get data
     _dataReceived = _WiLOS_Services._location_getAllUsersLocations(DateTime.Now);
   }
   catch (Exception _exc)
   {
      System.Diagnostics.Debug.WriteLine("Couldn't get users' locations, error: " + _exc.Message);
   }

   // Check if the operation was successful
   if (_dataReceived != null)
   {
      int _result = Convert.ToInt32(_dataReceived[0]);
      if (_result == 0)
      {
         _WiLOS_HUEPS_Objects._UsersLocationsList _ull = null;
         if (_dataReceived[1] != null)
         {
            // All good, get users' locations
            _ull = (_WiLOS_HUEPS_Objects._UsersLocationsList)_xmlOperations._deserializeToObject(
                 Convert.ToString(_dataReceived[1]), typeof(_WiLOS_HUEPS_Objects._UsersLocationsList));
            _usersLocationsCurrent = _ull._currentUsersLocations;
         }
      }
   }

...
```

Anexo L: Lista de Funcionalidades da Base de Dados - _HUEPS_Database.cs

_HUEPS_Database

```
/// <summary>
/// <para> Creates the HUEPS Database </para>
/// <returns>True or False, depending on the operation success</returns>
public static bool _database_create()


/// <summary>
/// <para> Emptys all the tables in HUEPS database, resetting the auto-increment values </para>
/// </summary>
/// <returns>True or False, depending on the operation success</returns>
public static bool _database_empty()
```

_EnergyConsumptionInformation

_EnergyConsumption

```
/// <summary>
/// <para> Selects all entries </para>
/// <para> Returns: A list of lists, each sub list contains: int _idEnergyConsumption, float _power,
          float _voltage, float _current, float _powerFactor, float _frequency, string _date, float
          _price </para>
/// </summary>
public static List<List<object>> _selectAll()


/// <summary>
/// <para> Selects all information of an Energy Consumption Entry </para>
/// <para> Returns: A list: int _idEnergyConsumption, float _power, float _voltage, float _current,
          float _powerFactor, float _frequency, string _date, float _price </para>
/// </summary>
/// <param name="_idEnergyConsumption">Identifier of the entry to select</param>
public static List<object> _select(int _idEnergyConsumption)


/// <summary>
/// <para> </para>
/// <para> Returns: </para>
/// </summary>
/// <param name="_date"> </param>
public static List<List<object>> _selectCurrentEnergyConsumptionByDivision(DateTime _date)

/// <summary>
/// <para> </para>
/// <para> Returns: </para>
/// </summary>
/// <param name="_date"> </param>
public static double _selectCurrentEnergyConsumption(DateTime _date)


/// <summary>
/// <para> Selects last energy consumption entry </para>
/// <para> Returns: Active power in kWh </para>
/// </summary>
/// <param name="_date">The date to be used as time comparison parameter</param>
public static float _selectPreviousConsumption(DateTime _date)


/// <summary>
/// <para> Selects Trivia Information from Energy Consumption </para>
/// <para> Returns: A list: Max Consumption Day, Max Consumption, Min Consumption Day, Min Consumption,
Total Days </para>
/// </summary>
public static List<object> _selectTriviaInformation()
```

```
/// <summary>
/// <para> Gets the energy consumption of the entire household organized by the given time filters
    </para>
/// <para> Returns a List of Lists, each list containing: </para>
/// <para> - All parameters 'null': year, total active power [kW], total cost [€], CO2 Emission Ratio
          [kg] </para>
/// <para> - Month and Day  'null': month, ...  </para>
/// <para> - Only Day 'null'      : day, ... </para>
/// <para> - No parameter 'null'  : date, ... </para>
/// </summary>
/// <param name="_year"> If not 'null' allows to define the desired year information </param>
/// <param name="_month"> If not 'null' allows to define the desired year information </param>
/// <param name="_day"> If not 'null' allows to define the desired day information  </param>
/// <param name="_minutesInterval"> Specifies the minute interval to be used at day resolution </param>
public static List<List<object>> _selectStats(string _year, string _month, string _day, int
                                               _minutesInterval)

/// <summary>
/// <para> Gets the energy consumption of the entire household organized by the existing users. The
          filters are used to refine the results </para>
/// <para> Returns a List of Lists, each list containing: idUser, username, total active power [kW],
          total cost [€], CO2 Emission Ratio [kg] </para>
/// </summary>
/// <param name="_year"> If not 'null' allows to define the desired year information </param>
/// <param name="_month"> If not 'null' allows to define the desired year information </param>
/// <param name="_day"> If not 'null' allows to define the desired day information  </param>
/// <param name="_minutesInterval"> Specifies the minute interval to be used at day resolution </param>
public static List<List<object>> _selectGlobalByUserDistribution(string _year, string _month, string
                                                    _day, string _hourMinute, int _minutesInterval)

/// <summary>
/// <para> Gets the energy consumption of the entire household organized by the existing divisions.
          The filters are used to refine the results </para>
/// <para> Returns a List of Lists, each list containing: idDivision, division name, total active power
          [kW], total cost [€], CO2 Emission Ratio [kg] </para>
/// </summary>
/// <param name="_year"> If not 'null' allows to define the desired year information </param>
/// <param name="_month"> If not 'null' allows to define the desired year information </param>
/// <param name="_day"> If not 'null' allows to define the desired day information  </param>
/// <param name="_minutesInterval"> Specifies the minute interval to be used at day resolution </param>
public static List<List<object>> _selectGlobalByDivisionDistribution(string _year, string _month,
                                               string _day, string _hourMinute, int _minutesInterval)

/// <summary>
/// <para> Gets the energy consumption of the entire household that has not been distribued to any
          user or division. The filters are used to refine the results </para>
/// <para> Returns a List of Lists, each list containing: date, total active power [kW], total cost
          [€], CO2 Emission Ratio [kg] </para>
/// </summary>
/// <param name="_year"> If not 'null' allows to define the desired year information </param>
/// <param name="_month"> If not 'null' allows to define the desired year information </param>
/// <param name="_day"> If not 'null' allows to define the desired day information  </param>
/// <param name="_minutesInterval"> Specifies the minute interval to be used at day resolution </param>
public static List<List<object>> _selectGlobalNoOneDistribution(string _year, string _month, string
                                                    _day, string _hourMinute, int _minutesInterval)

/// <summary>
/// <para> Gets a list of years available </para>
/// </summary>
public static List<object> _selectYears()


/// <summary>
/// <para> Gets a list of months available </para>
/// </summary>
/// <param name="_year"> The year to limit the search </param>
public static List<object> _selectMonths(string _year)


/// <summary>
/// <para> Gets a list of the days available on a certain month of the year </para>
/// </summary>
public static List<object> _selectDays(string _year, string _month)
```

```
/// <summary>
/// <para> Selects the information according to the given filters </para>
/// <para> Returns: A list of lists, each sub list contains: _idEnergyConsumption, float _power,
          float _voltage, float _current, float _powerFactor, float _frequency, string _date </para>
/// </summary>
/// <param name="_year"> The year to be used as filter </param>
/// <param name="_month"> The month to be used as filter </param>
/// <param name="_day"> The day to be used as filter </param>
/// <param name="_startTime"> The start of the time interval to be used as filter </param>
/// <param name="_endTime"> The end of the time interval to be used as filter </param>
/// <param name="_page"> The page to get from the database </param>
/// <param name="_rowsToGet"> The number of items to retrieve from the database </param>
public static List<List<object>> _selectFilter(string _year, string _month, string _day, string
                                        _startTime, string _endTime, int _page, int _rowsToGet)


/// <summary>
/// <para> Gets the number of rows according to the given filters </para>
/// <para> Returns: An int value </para>
/// </summary>
/// <param name="_year"> The year to be used as filter </param>
/// <param name="_month"> The month to be used as filter </param>
/// <param name="_day"> The day to be used as filter </param>
/// <param name="_startTime"> The start of the time interval to be used as filter </param>
/// <param name="_endTime"> The end of the time interval to be used as filter </param>
public static int _getRows(string _year, string _month, string _day, string _startTime, string
                           _endTime)


/// <summary>
/// <para> Creates a new Energy Consumption Entry </para>
/// <para> Returns: The new user identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_power"> The power valuebeing consumed by the house </param>
/// <param name="_voltage">  The voltage imposed by the energy network </param>
/// <param name="_current"> The current value being consumed by the house </param>
/// <param name="_powerFactor"> The power factor value read </param>
/// <param name="_frequency"> The frequency of the energy network </param>
/// <param name="_date"> The date of the sample aquisition </param>
public static int _insert(float _power, float _voltage, float _current, float _powerFactor, float
                          _frequency, DateTime _date)


/// <summary>
/// <para> Creates a new Energy Consumption Entry </para>
/// <para> Returns: The new user identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_idEnergyConsumption">Identifier of the Entry to create</param>
/// <param name="_power"> The power valuebeing consumed by the house </param>
/// <param name="_voltage">  The voltage imposed by the energy network </param>
/// <param name="_current"> The current value being consumed by the house </param>
/// <param name="_powerFactor"> The power factor value read </param>
/// <param name="_frequency"> The frequency of the energy network </param>
/// <param name="_date"> The date of the sample aquisition </param>
public static int _insert(int _idEnergyConsumption, float _power, float _voltage, float _current,
                          float _powerFactor, float _frequency, string _date)


/// <summary>
/// <para> Updates all fields of an Energy Consumption Entry </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idEnergyConsumption">Identifier of the Entry to be updated</param>
/// <param name="_power">The new 'power' value </param>
/// <param name="_voltage">The new 'voltage' value </param>
/// <param name="_current">The new 'current' value </param>
/// <param name="_powerFactor">The new 'powerFactor' value </param>
/// <param name="_frequency">The new 'frequency' value </param>
/// <param name="_date">The new 'date' value </param>
public static bool _update(int _idEnergyConsumption, float _power, float _voltage, float _current,
                           float _powerFactor, float _frequency, string _date)
```

```
/// <summary>
/// <para> Updates the field 'power' of an Energy Consumption Entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idEnergyConsumption">Identifier of the Entry to be updated</param>
/// <param name="_power">The new 'power' value </param>
public static bool _updatePower(int _idEnergyConsumption, float _power)


/// <summary>
/// <para> Updates the field 'voltage' of an Energy Consumption Entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idEnergyConsumption">Identifier of the Entry to be updated</param>
/// <param name="_voltage">The new 'voltage' value </param>
public static bool _updateVoltage(int _idEnergyConsumption, float _voltage)

/// <summary>
/// <para> Updates the field 'current' of an Energy Consumption Entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idEnergyConsumption">Identifier of the Entry to be updated</param>
/// <param name="_current">The new 'current' value </param>
public static bool _updateCurrent(int _idEnergyConsumption, float _current)


/// <summary>
/// <para> Updates the field 'powerFactor' of an Energy Consumption Entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idEnergyConsumption">Identifier of the Entry to be updated</param>
/// <param name="_powerFactor">The new 'powerFactor' value </param>
public static bool _updatePowerFactor(int _idEnergyConsumption, float _powerFactor)


/// <summary>
/// <para> Updates the field 'frequency' of an Energy Consumption Entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idEnergyConsumption">Identifier of the Entry to be updated</param>
/// <param name="_frequency">The new 'frequency' value </param>
public static bool _updateFrequency(int _idEnergyConsumption, float _frequency)


/// <summary>
/// <para> Updates the field 'date' of an Energy Consumption Entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idEnergyConsumption">Identifier of the Entry to be updated</param>
/// <param name="_date">The new 'date' value </param>
public static bool _updateDate(int _idEnergyConsumption, string _date)


/// <summary>
/// <para> Updates the field 'price' of an Energy Consumption Entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idEnergyConsumption">Identifier of the Entry to be updated</param>
/// <param name="_price">The new 'price' value </param>
public static bool _updatePrice(int _idEnergyConsumption, float _price)


/// <summary>
/// <para> Updates the field 'CO2' of an Energy Consumption Entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idEnergyConsumption">Identifier of the Entry to be updated</param>
/// <param name="_CO2">The new 'CO2' value </param>
public static bool _updateCO2(int _idEnergyConsumption, float _CO2)
```

```
/// <summary>
/// <para> Deletes an Energy Consumption Entry</para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name ="_idEnergyConsumption">Identifier of the entry to remove </param>
public static bool _delete(int _idEnergyConsumption)


/// <summary>
/// <para> Deletes all entries </para>
/// <para> Returns: True or False, depending on the operation success </para> </para>
/// </summary>
public static bool _deleteAll()
```

_Event

```
/// <summary>
/// <para> Selects all events </para>
/// <para> Returns: A list of lists, each sub list contains: int idEvent, int idLocation, id
          MultiEvent, float energyConsumption, int nUsers, bool active, string deactivatedTime,
          string date, int iduser, string username, int iddivision, string division name, bool
          movement </para>
/// </summary>
public static List<List<object>> _selectAll()

/// <summary>
/// <para> Selects all information of an energyConsumptionEvent </para>
/// <para> Returns: A list: int idEvent, int idLocation, id MultiEvent, float energyConsumption, int
          nUsers, bool active, string deactivatedTime, string date, int iduser, string username, int
          iddivision, string division name, bool movement  </para>
/// </summary>
/// <param name="_idEvent">Identifier of the user to select</param>
public static List<object> _select(int _idEvent)


/// <summary>
/// <para> Gets a list of years available </para>
/// </summary>
public static List<object> _selectYears()


/// <summary>
/// <para> Gets a list of months available </para>
/// </summary>
/// <param name="_year"> The year to limit the search </param>
public static List<object> _selectMonths(string _year)


/// <summary>
/// <para> Gets a list of the days available on a certain month of the year </para>
/// </summary>
public static List<object> _selectDays(string _year, string _month)


/// <summary>
/// <para> Selects the information according to the given filters </para>
/// <para> Returns: A list of lists, each sub list contains: int idEvent, int idLocation, id
          MultiEvent, float energyConsumption, int nUsers, bool active, string deactivatedTime,
          string date, int iduser, string username, int iddivision, string division name, bool
          movement </para>
/// </summary>
/// <param name="_year"> The year to be used as filter </param>
/// <param name="_month"> The month to be used as filter </param>
/// <param name="_day"> The day to be used as filter </param>
/// <param name="_startTime"> The start of the time interval to be used as filter </param>
/// <param name="_endTime"> The end of the time interval to be used as filter </param>
/// <param name="_divisionID"> The division to be used as filter </param>
/// <param name="_userID"> The user to be used as filter </param>
/// <param name="_page"> The page to get from the database </param>
/// <param name="_rowsToGet"> The number of items to retrieve from the database </param>
public static List<List<object>> _selectFilter(string _year, string _month, string _day, string
          _startTime, string _endTime, string _divisionID, string _userID,  int _page, int _rowsToGet)
```

```
/// <summary>
/// <para> Gets the number of rows according to the given filters </para>
/// <para> Returns: An int value </para>
/// </summary>
/// <param name="_year"> The year to be used as filter </param>
/// <param name="_month"> The month to be used as filter </param>
/// <param name="_day"> The day to be used as filter </param>
/// <param name="_startTime"> The start of the time interval to be used as filter </param>
/// <param name="_endTime"> The end of the time interval to be used as filter </param>
public static int _getRows(string _year, string _month, string _day, string _startTime, string
                           _endTime, string _divisionID, string _userID)

/// <summary>
/// <para> Creates a new Event </para>
/// <para> Returns: The new user identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_idEvent"> The identifier of the energyConsumptionEvent to create </param>
/// <param name="_idLocation"> The identifier of the user's location which caused this
                               energyConsumptionEvent </param>
/// <param name="_energyConsumption"> The energy consumption which caused this
                                      energyConsumptionEvent </param>
/// <param name="_active"> The state of this energyConsumptionEvent, if still occuring or not
    </param>
/// <param name="_nUsers"> The number of users associated with this energyConsumptionEvent. Used to
               calculate the correct amount of energy consumption associated with each user </param>
public static int _insert(int _idEvent, int _idLocation, int _idMultiEvent, float
                          _energyConsumption, int _nUsers, bool _active)

/// <summary>
/// <para> Creates a new Event </para>
/// <para> Returns: The new user identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_idLocation"> The identifier of the user's location which caused this
                               energyConsumptionEvent </param>
/// <param name="_energyConsumption"> The energy consumption which caused this
                                      energyConsumptionEvent </param>
/// <param name="_active"> The state of this energyConsumptionEvent, if still occuring or not
    </param>
/// <param name="_nUsers"> The number of users associated with this energyConsumptionEvent. Used to
               calculate the correct amount of energy consumption associated with each user </param>
public static int _insert(int _idLocation, int _idMultiEvent, float _energyConsumption, int _nUsers,
                          bool _active)

/// <summary>
/// <para> Creates a new Event </para>
/// <para> Returns: The new user identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_userLocation"> The user location object to be associated with this
                                 energyConsumptionEvent </param>
/// <param name="_energyConsumption"> The energy consumption which caused this
                                      energyConsumptionEvent </param>
/// <param name="_active"> The state of this energyConsumptionEvent, if still occuring or not
    </param>
/// <param name="_nUsers"> The number of users associated with this energyConsumptionEvent. Used to
               calculate the correct amount of energy consumption associated with each user </param>
public static int _insert(_WiLOS_HUEPS_Objects._UserLocation _userLocation, int _idMultiEvent, float
                          _energyConsumption, int _nUsers, bool _active)

/// <summary>
/// <para> Updates all fields of an Event </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idEvent">Identifier of the Event to be updated</param>
/// <param name="_idMultiEvent">The new 'idMultiEvent' value</param>
/// <param name="_idLocation">The new 'identifier of the user's location' value </param>
/// <param name="_energyConsumption">The new 'energy consumption' value </param>
/// <param name="_active">The new 'active' value </param>
/// <param name="_nUsers">The new '_nUsers' value </param>
/// <param name="_deactivatedTime">The new 'deactivatedTime' value</param>
public static bool _update(int _idEvent, int _idLocation, int _idMultiEvent, float
                           _energyConsumption, int _nUsers, bool _active, DateTime _deactivatedTime)
```

```
/// <summary>
/// <para> Updates the field 'idLocation' of an Event </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idEvent">Identifier of the Event to be updated</param>
/// <param name="_idLocation">The new 'identifier of the user's location' value </param>
public static bool _updateUserLocationID(int _idEvent, int _idLocation)


/// <summary>
/// <para> Updates the field 'energyConsumption' of an Event </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idEvent">Identifier of the Event to be updated</param>
/// <param name="_energyConsumption">The new 'energy consumption' value </param>
public static bool _updateEnergyConsumption(int _idEvent, float _energyConsumption)


/// <summary>
/// <para> Updates the field 'active' of an Event </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idEvent">Identifier of the Event to be updated</param>
/// <param name="_active">The new 'active' value </param>
public static bool _updateActive(int _idEvent, bool _active)


/// <summary>
/// <para> Updates the field 'idMultiEvent' of an Event </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idEvent">Identifier of the Event to be updated</param>
/// <param name="_idMultiEvent">The new 'idMultiEvent' value </param>
public static bool _updateMultiEventID(int _idEvent, int _idMultiEvent)


/// <summary>
/// <para> Updates the field 'nUsers' of an Event </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idEvent">Identifier of the Event to be updated</param>
/// <param name="_nUsers">The new 'nUsers' value </param>
public static bool _updateNUsers(int _idEvent, int _nUsers)

/// <summary>
/// <para> Updates the field 'deactivatedTime' of an Event </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idEvent">Identifier of the Event to be updated</param>
/// <param name="_deactivatedTime">The new 'deactivatedTime' value </param>
public static bool _updateDeactivatedTime(int _idEvent, DateTime _deactivatedTime)


/// <summary>
/// <para> Deletes an Event </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name ="_idEvent">Identifier of the Event to be eliminated</param>
public static bool _delete(int _idEvent)


/// <summary>
/// <para> Deletes the existing entries in Event Entity </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
public static bool _deleteAll()

// *************************** General Events *****************************

/// <summary>
/// <para> Selects all house active events </para>
/// <para> Returns: A list of _Monitor._energyDistribution._eventClass objects  </para>
/// </summary>
public static List<_Monitor._energyDistribution._eventClass> _selectActiveEvents()
```

```
/// <summary>
/// <para> Selects all information of an energyConsumptionEvent </para>
/// <para> Returns: A list: int idEvent, int idLocation, float energyConsumption, bool active, int
          nUsers, string date, int iduser, string username, int iddivision, string division name, bool
          movement </para>
/// </summary>
/// <param name="_idDivision"> Identifier of the division to be used as filter </param>
/// <param name="_active"> The status of the desired events (Active = 1, Not Active = 0) </param>
public static List<_Monitor._energyDistribution._eventClass> _selectWithDivisionID(int _idDivision,
                                                                                    bool _active)

/// <summary>
/// <para> Selects all information of an energyConsumptionEvent </para>
/// <para> Returns: A list: int idEvent, int idLocation, float energyConsumption, bool active, int
          nUsers, string date, int iduser, string username, int iddivision, string division name, bool
          movement  </para>
/// </summary>
/// <param name="_idUser"> Identifier of the user to be used as filter </param>
/// <param name="_active"> The status of the desired events (Active = 1, Not Active = 0) </param>
public static List<_Monitor._energyDistribution._eventClass> _selectWithUserID(int _idUser, bool
                                                                                _active)

/// <summary>
/// <para> Deactivates all active Events </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <returns> True or False, depending on the operation success </returns>
public static bool _deactivateAllActiveEvents()

// ************************ Single User Events ******************************

/// <summary>
/// <para> </para>
/// <para> Returns: </para>
/// </summary>
/// <param name="_idDivision"> </param>
/// <param name="_target"> </param>
/// <param name="_margin"> </param>
public static calculateCombinationsToReachSum._result _selectSingleEventsSumBestMatch(int _idDivision,
                                                                                      float _target, float _margin)

/// <summary>
/// <para> </para>
/// <para> Returns: </para>
/// </summary>
/// <param name="_target"> </param>
/// <param name="_margin"> </param>
public static calculateCombinationsToReachSum._result _selectSingleEventsSumBestMatch(float _target,
                                                                                      float _margin)

/// <summary>
/// <para> Tries to deactivate a perfect user energy consumption energyConsumptionEvent of a
          certain division (a non multi-user energyConsumptionEvent) </para>
/// <para> Returns: If unsuccessful, returns the ID of the closest multi-user energyConsumptionEvent
          or '-1', meaning that no matches can be found. </para>
/// </summary>
/// <param name="_divisionID"> The identifier of the division to be used as filter </param>
/// <param name="_powerkWh"> The active power value used for comparison </param>
/// <param name="_powerMargin"> The tolerance in kWh to be used during active power comparison </param>
public static _Monitor._energyDistribution._eventSimpleClass _deactivateSingleUserEvent(int
                                                  _divisionID, float _powerkWh, float _powerMargin)

/// <summary>
/// <para> Tries to deactivate a perfect user energy consumption energyConsumptionEvent that
         supports the given consumption variation (a non multi-user energyConsumptionEvent) </para>
/// <para> Returns: If unsuccessful, returns the closest multi-user energyConsumptionEvent or '-1',
          meaning that no matches can be found. </para>
/// </summary>
/// <param name="_powerkWh"> The active power value used for comparison </param>
/// <param name="_powerMargin"> The tolerance in kWh to be used during active power comparison </param>
public static _Monitor._energyDistribution._eventSimpleClass _deactivateSingleUserEvent(float
                                                                  _powerkWh, float _powerMargin)
```

```
/// <summary>
/// <para> Deactivates single user energy consumption energyConsumptionEvent associated with the
          given ID (a non multi-user energyConsumptionEvent) </para>
/// <para> Returns: True or False depending on the operation success </para>
/// </summary>
/// <param name="_idEvent"> The identifier of the energyConsumptionEvent </param>
public static bool _deactivateSingleUserEvent(int _eventID)


// ************************ Multi User Events ******************************


/// <summary>
/// <para> </para>
/// <para> Returns: </para>
/// </summary>
/// <param name="_target"> </param>
/// <param name="_margin"> </param>
public static _Monitor._energyDistribution._eventClass _selectMultiEvent(int _idMultiEvent, int
                                                                          _idUser)


/// <summary>
/// <para> </para>
/// <para> Returns: </para>
/// </summary>
/// <param name="_target"> </param>
/// <param name="_margin"> </param>
public static calculateCombinationsToReachSum._result _selectMultiEventsSumBestMatch(int _idDivision,
                                                                    float _target, float _margin)


/// <summary>
/// <para> </para>
/// <para> Returns: </para>
/// </summary>
/// <param name="_target"> </param>
/// <param name="_margin"> </param>
public static calculateCombinationsToReachSum._result _selectMultiEventsSumBestMatch(float _target,
                                                                          float _margin)


/// <summary>
/// <para> Updates the field 'energyConsumption' of a Multi Event (all the events belonging to this
          multi-energyConsumptionEvent) </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idMultiEvent">Identifier of the Multi-Event to be updated</param>
/// <param name="_energyConsumption">The new '_idMultiEvent consumption' value </param>
public static bool _updateMultiEventEnergyConsumption(int _idMultiEvent, float
                                                      _energyConsumption)


/// <summary>
/// Reorganiza os eventos tendo em conta que o idUser fornecido deixou de fazer parte do multi-evento
/// </summary>
/// <param name="_idMultiEvent"></param>
/// <param name="_idUser"></param>
/// <param name="_factor"></param>
public static void _multiEventReallocationUser(int _idMultiEvent, int _idUser, int _factor)


/// <summary>
/// Reorganiza os eventos tendo em conta que o utilizador mudou de divisão mas continua a fazer parte
    do multi-evento
/// </summary>
/// <param name="_idEvent"></param>
/// <param name="_idMultiEvent"></param>
/// <param name="_idLocation"></param>
public static void _multiEventReallocationDivision(int _idEvent, int _idMultiEvent, int _idLocation)
```

```
/// <summary>
/// <para>Tries to deactivate a perfect multi user energy consumption energyConsumptionEvent of a
        certain division (a multi-user energyConsumptionEvent has several events associated)
    </para>
/// <para>If unsuccessful, returns the ID of the closest multi-user energyConsumptionEvent or '-1',
        meaning that no matches can be found. </para>
/// </summary>
/// <param name="_divisionID">The identifier of the division to be used as filter </param>
/// <param name="_powerkWh">The active power value used for comparison </param>
/// <param name="_powerMargin">The tolerance in kWh to be used during active power comparison </param>
public static _Monitor._energyDistribution._eventSimpleClass _deactivateMultiUserEvent(int
                                                        _divisionID, float _powerkWh, float _powerMargin)

/// <summary>
/// <para>Tries to deactivate a perfect multi user energy consumption energyConsumptionEvent
        where the given  variation fits (a multi-user energyConsumptionEvent has several events
        associated) </para>
/// <para>If unsuccessful, returns the ID of the closest multi-user energyConsumptionEvent or '-1',
        meaning that no matches can be found. </para>
/// </summary>
/// <param name="_powerkWh"> The active power value used for comparison </param>
/// <param name="_powerMargin"> The tolerance in kWh to be used during active power comparison </param>
public static _Monitor._energyDistribution._eventSimpleClass _deactivateMultiUserEvent(float _powerkWh,
float _powerMargin)


/// <summary>
/// <para>Deactivates a multi user energy consumption energyConsumptionEvent associated with the
        given ID (a multi-user energyConsumptionEvent has several events associated) </para>
/// <para>Returns: True or False depending on the operation success </para>
/// </summary>
/// <param name="_idMultiEvent"> Identifier of the multi energyConsumptionEvent to deactivate </param>
public static bool _deactivateMultiUserEvent(int _idMultiEvent)


/// <summary>
/// <para> </para>
/// <para> Returns: </para>
/// </summary>
/// <param name="_powerkWh"> </param>
public static float _deactivateEvent(float _powerkWh)
```

_MultiEvent

```
/// <summary>
/// <para>Selects all multievents identifiers </para>
/// <para>Returns: A list: int idMultiEvent </para>
/// </summary>
public static List<object> _selectAll()


/// <summary>
/// <para>Creates a new MultiEvent Identifier </para>
/// <para>Returns: The new identifier as an int, or -1 if an error occurs</para>
/// </summary>
public static int _insert()


/// <summary>
/// <para> Creates a new MultiEvent Identifier </para>
/// <para> Returns: The new identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_idMultiEvent"> The identifier to insert </param>
public static int _insert(int _idMultiEvent)


/// <summary>
/// <para> Deletes a MultiEvent Identifier </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name ="_idMultiEvent">Identifier to be eliminated</param>
public static bool _delete(int _idMultiEvent)
```

```
/// <summary>
/// <para> Deletes the existing entries in MultiEvent Entity </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
public static bool _deleteAll()
```

_PartialEnergyConsumption

```
/// <summary>
/// <para> Selects all Partial Energy Consumption entries </para>
/// <para> Returns: A list of lists, each sub list contains: int idEvent, int idEnergyConsumption
    </para>
/// </summary>
public static List<List<object>> _selectAll()
```

```
/// <summary>
/// <para> Selects Partial Energy Consumption entries </para>
/// <para> Returns: A list of lists, each sub list contains: int idEvent, int idEnergyConsumption
    </para>
/// </summary>
/// <param name="_idEvent">The energyConsumptionEvent identifier to be used as a filter</param>
public static List<List<object>> _selectWithEventID(int _idEvent)
```

```
/// <summary>
/// <para> Selects Partial Energy Consumption entries </para>
/// <para> Returns: A list of lists, each sub list contains: int idEvent, int idEnergyConsumption
    </para>
/// </summary>
/// <param name="_idEnergyConsumption"> The energy consumption identifier to be used as a
                                        filter</param>
public static List<List<object>> _selectWithEnergyConsumptionID(int _idEnergyConsumption)
```

```
/// <summary>
/// <para> Creates a new Partial Energy Consumption entry</para>
/// <para> The new entry identifier as an int or -1 if an error occurs </para>
/// </summary>
/// <param name ="_idEnergyConsumption"> Identifier of the energy consumption entry in "Energy
                                         Consumption" Entity </param>
/// <param name ="_idEvent"> Identifier of the energyConsumptionEvent entry in "Event" Entity </param>
public static int _insert(int _idEvent, int _idEnergyConsumption)
```

```
/// <summary>
/// <para> Deletes an entry </para>
/// <para> True or False, depending on the operation success </para>
/// </summary>
/// <param name ="_idEnergyConsumption"> Identifier of the energy consumption associated with the
                                         entry to remove </param>
/// <param name ="_idEvent"> Identifier of the energyConsumptionEvent associated with the entry to
                             remove </param>
public static bool _delete(int _idEvent, int _idEnergyConsumption)
```

```
/// <summary>
/// <para> Deletes Partial Energy Consumption entries </para>
/// <para> True or False, depending on the operation success </para>
/// </summary>
/// <param name ="_idEnergyConsumption">Identifier of the energy consumption entry to be used as
                                        filter </param>
public static bool _deleteWithEnergyConsumptionID(int _idEnergyConsumption)
```

```
/// <summary>
/// <para> Deletes Partial Energy Consumption entries </para>
/// <para> True or False, depending on the operation success </para>
/// </summary>
/// <param name ="_idEvent">Identifier of the energyConsumptionEvent to be used as a filter</param>
public static bool _deleteWithEventID(int _idEvent)



/// <summary>
/// <para> Deletes all entries in Partial Energy Consumption Entity </para>
/// <para> Returns: True or False, depending on the operation success </para> </para>
/// </summary>
public static bool _deleteAll()
```

_UserEnergyProfile

```
/// <summary>
/// <para> Gets users list with associated energy consumption </para>
/// <para> Returns: A List of Lists, each list containing: int idUser, string username </para>
/// </summary>
public static List<List<object>> _selectUsers()



/// <summary>
/// <para> Gets the user energy consumption organized by the existing divisions. The filters are
          used to refine the results </para>
/// <para> Returns a List of Lists, each list containing: idDivision, division name, total active power
          [kW], total cost [€], CO2 Emission Ratio [kg] </para>
/// </summary>
/// <param name="_idUser"> The identifier of the desired user </param>
/// <param name="_year"> If not 'null' allows to define the desired year information </param>
/// <param name="_month"> If not 'null' allows to define the desired year information </param>
/// <param name="_day"> If not 'null' allows to define the desired day information  </param>
/// <param name="_minutesInterval"> Specifies the minute interval to be used at day resolution </param>
public static List<List<object>> _selectByDivisionDistribution(int _idUser, string _year, string
                                        _month, string _day, string _hourMinute, int _minutesInterval)



/// <summary>
/// <para> Gets the energy consumption of the given user organized by the given time filters </para>
/// <para> Returns a List of Lists, each list containing: </para>
/// <para> - All parameters 'null': year, total active power [kW], total cost [€], CO2 Emission Ratio
            [kg] </para>
/// <para> - Month and Day  'null': month, ...  </para>
/// <para> - Only Day 'null'      : day, ... </para>
/// <para> - No parameter 'null'  : date, ... </para>
/// </summary>
/// <param name="_idUser"> The identifier of the user whose energy profile is going to be selected
    </param>
/// <param name="_year"> If not 'null' allows to define the desired year information </param>
/// <param name="_month"> If not 'null' allows to define the desired year information </param>
/// <param name="_day"> If not 'null' allows to define the desired day information  </param>
/// <param name="_minutesInterval"> Specifies the minute interval to be used at day resolution </param>
public static List<List<object>> _selectStats(int _idUser, string _year, string _month, string _day,
                                        int _minutesInterval)



/// <summary>
/// <para> Gets the energy consumption of the the desired user arranged by month </para>
/// <para> Returns: A List of Lists, each list containing: Time, total active power [kW], total cost
          [€], CO2 Emission Ratio [kg] </para>
/// </summary>
/// <param name="_idUser"> The identifier of the user whose energy profile is going to be selected
    </param>
/// <param name="_year"> Defines the desired year </param>
public static List<List<object>> _selectByYear(int _idUser, string _year)
```

```
/// <summary>
/// <para> Gets the energy consumption of the the desired user arranged by day </para>
/// <para> Returns: A List of Lists, each list containing: day, total active power [kW], total cost
        [€], CO2 Emission Ratio [kg] </para>
/// </summary>
/// <param name="_idUser"> The identifier of the user whose energy profile is going to be selected
    </param>
/// <param name="_year"> Defines the desired year </param>
/// <param name="_month"> Defines the desired month </param>
public static List<List<object>> _selectByMonth(int _idUser, string _year, string _month)


/// <summary>
/// <para> Gets the energy consumption of the the desired user arranged by minutes </para>
/// <para> Returns: A List of Lists, each list containing: Time, total active power [kW], total cost
        [€], CO2 Emission Ratio [kg] </para>
/// </summary>
/// <param name="_idUser"> The identifier of the user whose energy profile is going to be selected
</param>
/// <param name="_year"> Defines the desired year </param>
/// <param name="_month"> Defines the desired month </param>
/// <param name="_day"> Defines the desired day </param>
/// <param name="_minutesInterval"> Specifies the minute interval to be used as day resolution </param>
public static List<List<object>> _selectByDay(int _idUser, string _year, string _month, string _day,
                                               int _minutesInterval)

/// <summary>
/// <para> Gets a list of available years for the energy consumption of the given user </para>
/// </summary>
/// <param name="_idUser"> The identifier of the user </param>
public static List<object> _selectYears(int _idUser)


/// <summary>
/// <para> Gets a list of available months of a certain year for the energy consumption of the given
        user </para>
/// </summary>
/// <param name="_idUser"> The identifier of the user </param>
/// <param name="_year"> The year to be used as filter </param>
/// <returns> Gets a list of months [int] </returns>
public static List<object> _selectMonths(int _idUser, string _year)


/// <summary>
/// <para> Gets a list of available days of certain year-month for the energy consumption of the
        given user </para>
/// </summary>
/// <param name="_idUser"> The identifier of the user </param>
/// <param name="_year"> The year to be used as filter </param>
/// <param name="_month"> The month to be used as filter </param>
public static List<object> _selectDays(int _idUser, string _year, string _month)
```

_DivisionEnergyProfile

```
/// <summary>
/// <para> Gets divisions list with associated energy consumption </para>
/// <para> Returns: A List of Lists, each list containing: int idDivision, string divisionName </para>
/// </summary>
public static List<List<object>> _selectDivisions()


/// <summary>
/// <para> Gets the energy consumption of the the desired user arranged by year </para>
/// <para> Returns: A List of Lists, each list containing: year, total active power [kW], total cost
        [€], CO2 Emission Ratio [kg] </para>
/// </summary>
/// <param name="_idDivision"> The identifier of the division whose energy profile is going to be
    </param>
public static List<List<object>> _selectByAll(int _idDivision)
```

```
/// <summary>
/// <para> Gets the energy consumption of the given user organized by the given time filters </para>
/// <para> Returns a List of Lists, each list containing: </para>
/// <para> - All parameters 'null': year, total active power [kW], total cost [€], CO2 Emission Ratio
           [kg] </para>
/// <para> - Month and Day  'null': month, ...  </para>
/// <para> - Only Day 'null'      : day, ... </para>
/// <para> - No parameter 'null'  : date, ... </para>
/// </summary>
/// <param name="_idDivision"> The identifier of the division whose energy profile is going to be
                              selected </param>
/// <param name="_year"> If not 'null' allows to define the desired year information </param>
/// <param name="_month"> If not 'null' allows to define the desired year information </param>
/// <param name="_day"> If not 'null' allows to define the desired day information  </param>
/// <param name="_minutesInterval"> Specifies the minute interval to be used at day resolution </param>
public static List<List<object>> _selectStats(int _idDivision, string _year, string _month, string
                                               _day, int _minutesInterval)


/// <summary>
/// <para> Gets the division energy consumption organized by the existing users. The filters are
          used to refine the results </para>
/// <para> Returns a List of Lists, each list containing: idUser, username, total active power [kW],
          total cost [€], CO2 Emission Ratio [kg] </para>
/// </summary>
/// <param name="_idDivision"> Identifier of the desired division </param>
/// <param name="_year"> If not 'null' allows to define the desired year information </param>
/// <param name="_month"> If not 'null' allows to define the desired year information </param>
/// <param name="_day"> If not 'null' allows to define the desired day information  </param>
/// <param name="_minutesInterval"> Specifies the minute interval to be used at day resolution </param>
public static List<List<object>> _selectByUserDistribution(int _idDivision, string _year, string
                                         _month, string _day, string _hourMinute, int _minutesInterval)


/// <summary>
/// <para> Gets the energy consumption of the the desired user arranged by month </para>
/// <para> Returns: A List of Lists, each list containing: Time, total active power [kW], total cost
          [€], CO2 Emission Ratio [kg] </para>
/// </summary>
/// <param name="_idDivision"> The identifier of the division whose energy profile is going to be
                              selected </param>
/// <param name="_year"> Defines the desired year </param>
public static List<List<object>> _selectByYear(int _idDivision, string _year)



/// <summary>
/// <para> Gets the energy consumption of the the desired user arranged by day </para>
/// <para> Returns: A List of Lists, each list containing: day, total active power [kW], total cost
          [€], CO2 Emission Ratio [kg] </para>
/// </summary>
/// <param name="_idDivision"> The identifier of the division whose energy profile is going to be
                              selected </param>
/// <param name="_year"> Defines the desired year </param>
/// <param name="_month"> Defines the desired month </param>
public static List<List<object>> _selectByMonth(int _idDivision, string _year, string _month)



/// <summary>
/// <para> Gets the energy consumption of the the desired user arranged by minutes </para>
/// <para> Returns: A List of Lists, each list containing: Time, total active power [kW], total cost
          [€], CO2 Emission Ratio [kg] </para>
/// </summary>
/// <param name="_idDivision"> The identifier of the division whose energy profile is going to be
                              selected </param>
/// <param name="_year"> Defines the desired year </param>
/// <param name="_month"> Defines the desired month </param>
/// <param name="_day"> Defines the desired day </param>
/// <param name="_minutesInterval"> Specifies the minute interval to be used as day resolution </param>
public static List<List<object>> _selectByDay(int _idDivision, string _year, string _month, string
                                               _day, int _minutesInterval)
```

```
/// <summary>
/// <para> Gets a list of available years for the energy consumption of the given user </para>
/// </summary>
/// <param name="_idDivision"> The identifier of the division </param>
public static List<object> _selectYears(int _idDivision)



/// <summary>
/// <para> Gets a list of available months of a certain year for the energy consumption of the given
        user </para>
/// </summary>
/// <param name="_idDivision"> The identifier of the division </param>
/// <param name="_year"> The year to be used as filter </param>
public static List<object> _selectMonths(int _idDivision, string _year)



/// <summary>
/// <para> Gets a list of available days of certain year-month for the energy consumption of the
        given user </para>
/// </summary>
/// <param name="_idDivision"> The identifier of the division </param>
/// <param name="_year"> The year to be used as filter </param>
/// <param name="_month"> The month to be used as filter </param>
public static List<object> _selectDays(int _idDivision, string _year, string _month)
```

_EnergySupplier

_Information

```
/// <summary>
/// <para> Selects all entries from Energy Supplier Entity </para>
/// <para> Returns: A list of lists. Each list contains: int idEnergySupplier, string
        nameEnergySupplier, string descriptionEnergySupplier, string nameFare, string
        descriptionFare, string contact, string email, float CO2EmissionsRatio_tMWh</para>
/// </summary>
public static List<List<object>> _selectAll()



/// <summary>
/// <para> Selects an entry from Energy Supplier Entity </para>
/// <para> Returns: A list: int idEnergySupplier, string nameEnergySupplier, string
        descriptionEnergySupplier, string nameFare, string descriptionFare, string contact, string
        email, float CO2EmissionsRatio_tMWh</para>
/// </summary>
/// <param name="_id">The identifier of the Energy Supplier to select</param>
public static List<object> _select(int _id)



/// <summary>
/// <para> Creates an Energy Supplier Entry with the given information </para>
/// <para> Returns: The energy Supplier identifier as an int, or -1 if an error occurs </para>
/// </summary>
/// <param name="_nameES">Name of the Energy Supplier</param>
/// <param name="_descriptionES">Description of the company</param>
/// <param name="_nameFare">Name of the Current Fare</param>
/// <param name="_descriptionFare">Description of the Fare</param>
/// <param name="_contact">The company phone number in case of assistance</param>
/// <param name="_email">The email address of the organization</param>
public static int _insert(string _nameES, string _descriptionES, string _nameFare, string
                    _descriptionFare, string _contact, string _email, float _CO2EmissionsRatio_tMWh)
```

```
/// <summary>
/// <para> Creates an Energy Supplier Entry with the given information </para>
/// <para> Returns: The energy Supplier identifier as an int, or -1 if an error occurs </para>
/// </summary>
/// <param name="_idEnergySupplier">The identifier of the Energy Supplier to create</param>
/// <param name="_nameES">Name of the Energy Supplier</param>
/// <param name="_descriptionES">Description of the company</param>
/// <param name="_nameFare">Name of the Current Fare</param>
/// <param name="_descriptionFare">Description of the Fare</param>
/// <param name="_contact">The company phone number in case of assistance</param>
/// <param name="_email">The email address of the organization</param>
public static int _insert(int _idEnergySupplier, string _nameES, string _descriptionES, string
     _nameFare, string _descriptionFare, string _contact, string _email, float _CO2EmissionsRatio_tMWh)


/// <summary>
/// <para> Updates all fields in the entity Energy Supplier </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_id">The Energy Supplier Identifier value of the entry to update</param>
/// <param name="_nameES">The new Energy Supplier Name value</param>
/// <param name="_descriptionES">The new Energy Supplier Description value</param>
/// <param name="_nameFare">The new Fare Name value</param>
/// <param name="_descriptionFare">The new Fare Description value</param>
/// <param name="_email">The new Email value</param>
/// <param name="_contact">The new Contact value</param>
public static bool _update(int _id, string _nameES, string _descriptionES, string _nameFare, string
                     _descriptionFare, string _email, string _contact, float _CO2EmissionsRatio_tMWh)

/// <summary>
/// <para> Updates the Energy Supplier Name of the given entry </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_id">The Energy Supplier Identifier value of the entry to update</param>
/// <param name="_name">The new Energy Supplier Name value</param>
public static bool _updateEnergySupplierName(int _id, string _name)


/// <summary>
/// <para> Updates the Energy Supplier Description of the given entry </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_id">The Energy Supplier Identifier value of the entry to update</param>
/// <param name="_description">The new Description value of the Energy Supplier</param>
public static bool _updateEnergySupplierDescription(int _id, string _description)


/// <summary>
/// <para> Updates the Energy Supplier Fare Name of the given entry </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_id">The Energy Supplier Identifier value of the entry to update</param>
/// <param name="_name">The new 'nameFare'</param>
public static bool _updateFareName(int _id, string _name)


/// <summary>
/// <para> Updates the Energy Supplier Fare Description of the given entry </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_id">The Energy Supplier Identifier value of the entry to update</param>
/// <param name="_description">The new 'descriptionFare'</param>
public static bool _updateFareDescription(int _id, string _description)


/// <summary>
/// <para> Updates the Energy Supplier Contact of the given entry </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_id">The Energy Supplier Identifier value of the entry to update</param>
/// <param name="_contact">The new Contact value</param>
public static bool _updateContact(int _id, string _contact)
```

```
/// <summary>
/// <para> Updates the Energy Supplier Email of the given entry </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_id">The Energy Supplier Identifier value of the entry to update</param>
/// <param name="_email">The new Email value</param>
public static bool _updateEmail(int _id, string _email)


/// <summary>
/// <para> Updates the Energy Supplier CO2 Emissions Ratio [tMWh] of the given entry </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_id">The Energy Supplier Identifier value of the entry to update</param>
/// <param name="_CO2EmissionsRatio_tMWh">The new 'CO2 Emissions Ratio [tMWh]' value</param>
public static bool _updateCO2EmissionsRatio_tMWh(int _id, float _CO2EmissionsRatio_tMWh)


/// <summary>
/// <para> Delete an entry in Energy Supplier Entity </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_id">The Energy Supplier Identifier value of the entry to update</param>
public static bool _delete(int _id)


/// <summary>
/// <para> Deletes the existing entry in Energy Supplier Entity </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
public static bool _deleteAll()
```

_FareDetail

```
/// <summary>
/// <para> Selects all fare details from the Energy Supplier Fare Detail Entity related to an energy
          supplier </para>
/// <para> Returns: A list of lists, each sub list contains: int idFareDetail, int idEnergySupplier,
          float priceKW_h, string from, string to, string weekDays</para>
/// </summary>
/// <param name="_idEnergySupplier">The identifier of the Energy Supplier</param>
public static List<List<object>> _selectAll(int _idEnergySupplier)


/// <summary>
/// <para> Selects all fare details from the Energy Supplier Fare Detail Entity </para>
/// <para> Returns: A list of lists, each sub list contains: int idFareDetail, int idEnergySupplier,
          float priceKW_h, string from, string to, string weekDays</para>
/// </summary>
public static List<List<object>> _selectAll()


/// <summary>
/// <para> Selects all information from an Energy Supplier Fare Detail Entry </para>
/// <para> Returns: A list: int idFareDetail, int idEnergySupplier, float priceKW_h, string from,
          string to, string weekDays </para>
/// </summary>
public static List<object> _select(int _idFareDetail)


/// <summary>
/// <para> Selects the fare price to be applied </para>
/// <para> Returns: float prite [€/h] or -1 if an error occurs </para>
/// </summary>
public static float _select(DateTime _dt)
```

```
/// <summary>
/// <para> Creates an Energy Supplier Fare Detail Entry with the given information </para>
/// <para> Returns: The fare detail identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_idEnergySupplier">The identifier of the Energy Supplier</param>
/// <param name="_priceKW_h">Price of energy during that period</param>
/// <param name="_from">The beginning point of the period</param>
/// <param name="_to">The ending point of the period</param>
/// <param name="_weekDays">The string value representing the days of the week of the taxing
                           period</param>
/// <param name="_months">The string value representing the months of the year of the taxing
                          period</param>
public static int _insert(int _idEnergySupplier, float _priceKW_h, string _from, string _to, string
                          _weekDays, string _months)

/// <summary>
/// <para> Creates an Energy Supplier Fare Detail Entry with the given information </para>
/// <para> Returns: The fare detail identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_idFareDetail"> The identifier of the Fare Detail to insert</param>
/// <param name="_idEnergySupplier">The identifier of the Energy Supplier</param>
/// <param name="_priceKW_h">Price of energy during that period</param>
/// <param name="_from">The beginning point of the taxing period</param>
/// <param name="_to">The ending point of the taxing period</param>
/// <param name="_weekDays">The string value representing the days of the week of the taxing
                           period</param>
/// <param name="_months">The string value representing the months of the year of the taxing
                          period</param>
public static int _insert(int _idFareDetail, int _idEnergySupplier, float _priceKW_h, string _from,
                          string _to, string _weekDays, string _months)


/// <summary>
/// <para> Updates all the information of an Energy Supplier Fare Detail Entry </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idFareDetail">The Fare Identifier of the entry to be updated</param>
/// <param name="_idEnergySupplier">The new identifier of the Energy Supplier value </param>
/// <param name="_priceKW_h">The new 'price of energy during that period' value </param>
/// <param name="_from">The new 'beginning point of the taxing period' value </param>
/// <param name="_to">The new 'ending point of the taxing period' value </param>
/// <param name="_weekDays">The new 'string representing the days of the week of the taxing period'
                           value </param>
/// <param name="_months">The new 'string representing the months of the year of the taxing period'
                          value</param>
public static bool _update(int _idFareDetail, int _idEnergySupplier, float _priceKW_h, string _from,
                           string _to, string _weekDays, string _months)

/// <summary>
/// <para> Updates the field 'idEnergySupplier' of an Energy Supplier Fare Detail Entry </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idEnergySupplier">The new Energy Supplier Identifier value</param>
/// <param name="_idFareDetail">The Fare Identifier of the entry to be updated</param>
public static bool _updateEnergySupplierID(int _idFareDetail, int _idEnergySupplier)


/// <summary>
/// <para> Updates the field 'priceKW_h' of an Energy Supplier Fare Detail Entry </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idFareDetail">The Fare Identifier of the entry to be updated</param>
/// <param name="_priceKW_h">The new 'price kW/h' value</param>
public static bool _updatePriceKW_h(int _idFareDetail, float _priceKW_h)


/// <summary>
/// <para> Updates the field 'from' of an Energy Supplier Fare Detail Entry </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idFareDetail">The Fare Identifier of the entry to be updated</param>
/// <param name="_from">The new 'beginning point of the taxing period' value</param>
public static bool _updateFrom(int _idFareDetail, string _from)
```

```
/// <summary>
/// <para> Updates the field 'to' of an Energy Supplier Fare Detail Entry </para>
/// </summary>
/// <param name="_idFareDetail">The Fare Identifier of the entry to be updated</param>
/// <param name="_to">The new 'ending point of the taxing period' value</param>
public static bool _updateTo(int _idFareDetail, string _to)


/// <summary>
/// <para> Updates the field 'weekDays' of an Energy Supplier Fare Detail Entry </para>
/// </summary>
/// <param name="_idFareDetail">The Fare Identifier of the entry to be updated</param>
/// <param name="_weekDays">The new 'weekDays' value</param>
public static bool _updateWeekDays(int _idFareDetail, string _weekDays)


/// <summary>
/// <para> Updates the field 'months' of an Energy Supplier Fare Detail Entry </para>
/// </summary>
/// <param name="_idFareDetail">The Fare Identifier of the entry to be updated</param>
/// <param name="_months">The new 'months' value</param>
public static bool _updateMonths(int _idFareDetail, string _months)


/// <summary>
/// <para> Deletes the existing entry in Energy Supplier Fare Detail Entity </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idFareDetail">The Fare Identifier of the entry to be updated</param>
public static bool _delete(int _idFareDetail)


/// <summary>
/// <para> Deletes the all entries in Energy Supplier Fare Detail Entity </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
public static bool _deleteAll()
```

_House

_Information

```
/// <summary>
/// <para> Selects all information from one House </para>
/// <para> Returns: A list: int idHouse, int idEnergySupplier, int idImageInfo, string name, string
           description, string address, string zipCode, string country, string contact, int
           numberFloors, int numberDivisions, float area, float width, float length, byte[] houseImage
    </para>
/// </summary>
/// <param name="_idHouse">The identifier of the house to select</param>
public static List<object> _select(int _idHouse)


/// <summary>
/// <para> Selects all houses </para>
/// <para> Returns: A list of lists containing: int idHouse, int idEnergySupplier, int idImageInfo,
           string name, string description, string address, string zipCode, string country, string
           contact, int numberFloors, int numberDivisions, float area, float width, float length,
           byte[] houseImage </para>
/// </summary>
public static List<List<object>> _selectAll()


/// <summary>
/// <para> Checks if the given house exists on the database </para>
/// <para> Returns: True or False depending on the success of the operation </para>
/// </summary>
/// <param name="_idHouse"> The identifier of the house to check </param>
public static bool _exists(int _idHouse)
```

```
/// <summary>
/// <para> Checks if there is any house information </para>
/// <para> Returns: True if a house exists, false if it doens't </para>
/// </summary>
public static bool _houseExists()



/// <summary>
/// <para> Creates a house with the all the information </para>
/// <para> The house identifier as an 'Int' or '-1' if an error occurs </para>
/// </summary>
/// <param name="_idEnergySupplier"> Identifier of the house energy supplier </param>
/// <param name="_name"> THe house's name </param>
/// <param name="_description"> The house's Description </param>
/// <param name="_address"> The house's Street Location </param>
/// <param name="_zipCode"> The house's Zipcode </param>
/// <param name="_country"> The house's Country Location </param>
/// <param name="_contact"> The number of the house's telephone </param>
/// <param name="_numberFloors"> Number of floors of the house </param>
/// <param name="_numberDivisions"> Number of divisions of the house </param>
/// <param name="_area"> The house's terrain area </param>
/// <param name="_width"> The house's width </param>
/// <param name="_length"> The house's length </param>
/// <param name="_houseImage"> The house's footprint image</param>
/// <param name="_scale"> The house scale obtained from one of its walls </param>
/// <param name="_scaleDefinedOnX"> The value representing if the scale was defined from the X Axis or
                                    not </param>
/// <param name="_imageHeightOnScale"> The image Height during the scale definition (used for scaling
                                       purposes) </param>
/// <param name="_imageWidthOnScale"> The image Width during the scale definition (used for scaling
                                      purposes) </param>
public static int _insert(int _idEnergySupplier, int _idImageInfo, string _name, string _description,
 string _address, string _zipCode, string _country, string _contact, int _numberFloors, int
 _numberDivisions, float _area, float _width, float _length, byte[] _houseImage, byte[] _houseTopology)



/// <summary>
/// <para> Creates a house with the all the information </para>
/// <para> The house identifier as an 'Int' or '-1' if an error occurs </para>
/// </summary>
/// <param name="_idHouse">The identifier of the house to insert</param>
/// <param name="_idEnergySupplier"> Identifier of the house energy supplier </param>
/// <param name="_name"> THe house's name </param>
/// <param name="_description"> The house's Description </param>
/// <param name="_address"> The house's Street Location </param>
/// <param name="_zipCode"> The house's Zipcode </param>
/// <param name="_country"> The house's Country Location </param>
/// <param name="_contact"> The number of the house's telephone </param>
/// <param name="_numberFloors"> Number of floors of the house </param>
/// <param name="_numberDivisions"> Number of divisions of the house </param>
/// <param name="_area"> The house's terrain area </param>
/// <param name="_width"> The house's width </param>
/// <param name="_length"> The house's length </param>
/// <param name="_houseImage"> The house's footprint image</param>
/// <param name="_scale"> The house scale obtained from one of its walls </param>
/// <param name="_scaleDefinedOnX"> The value representing if the scale was defined from the X Axis or
                                    not </param>
/// <param name="_imageHeightOnScale"> The image Height during the scale definition (used for scaling
                                       purposes) </param>
/// <param name="_imageWidthOnScale"> The image Width during the scale definition (used for scaling
                                      purposes) </param>
public static int _insert(int _idHouse, int _idEnergySupplier, int _idImageInfo, string _name, string
  _description, string _address, string _zipCode, string _country, string _contact, int _numberFloors,
  int _numberDivisions, float _area, float _width, float _length, byte[] _houseImage, byte[]
  _houseTopology)
```

```
/// <summary>
/// <para> Updates the house information, given the correct identifier </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse">The identifier for this house</param>
/// <param name="_idEnergySupplier"> Identifier of the house energy supplier </param>
/// <param name="_name"> THe house's name </param>
/// <param name="_description"> The house's Description </param>
/// <param name="_address"> The house's Street Location </param>
/// <param name="_zipCode"> The house's Zipcode </param>
/// <param name="_country"> The house's Country Location </param>
/// <param name="_contact"> The number of the house's telephone </param>
/// <param name="_numberFloors"> Number of floors of the house </param>
/// <param name="_numberDivisions"> Number of divisions of the house </param>
/// <param name="_area"> The house's terrain area </param>
/// <param name="_width"> The house's width (X axis) </param>
/// <param name="_length"> The house's length (Y axis) </param>
/// <param name="_houseImage"> The house's footprint image</param>
/// <param name="_scale"> The house scale obtained from one of its walls </param>
/// <param name="_scaleDefinedOnX"> The value representing if the scale was defined from the X Axis or
                                    not </param>
/// <param name="_imageHeightOnScale"> The image Height during the scale definition (used for scaling
                                       purposes) </param>
/// <param name="_imageWidthOnScale"> The image Width during the scale definition (used for scaling
                                      purposes) </param>
public static bool _update(int _idHouse, int _idEnergySupplier, int _idImageInfo, string _name, string
   _description, string _address, string _zipCode, string _country, string _contact, int _numberFloors,
   int _numberDivisions, float _area, float _width, float _length, byte[] _houseImage, byte[]
   _houseTopology)


/// <summary>
/// <para> Updates the field 'idEnergySupplier' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier of the house to be updated</param>
/// <param name="_idEnergySupplier"> The new 'house energy supplier identifier' value </param>
public static bool _updateEnergySupplierID(int _idHouse, int _idEnergySupplier)


/// <summary>
/// <para> Updates the field 'name' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier of the house to be updated</param>
/// <param name="_name"> The new 'house name' value </param>
public static bool _updateName(int _idHouse, string _name)


/// <summary>
/// <para> Updates the field 'description' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier of the house to be updated</param>
/// <param name="_description">The new 'house description' value</param>
public static bool _updateDescription(int _idHouse, string _description)


/// <summary>
/// <para> Updates the field 'address' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier of the house to be updated </param>
/// <param name="_address">The new 'house's Street Location' value </param>
public static bool _updateAddress(int _idHouse, string _address)


/// <summary>
/// <para> Updates the field 'zipCode' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier of the house to be updated</param>
/// <param name="_zipCode">The new 'house's Zipcode' value</param>
public static bool _updateZipCode(int _idHouse, string _zipCode)
```

```
/// <summary>
/// <para> Updates the field 'country' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier of the house to be updated</param>
/// <param name="_country">The new 'house's Country Location' value</param>
public static bool _updateCountry(int _idHouse, string _country)


/// <summary>
/// <para> Updates the field 'contact' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier of the house to be updated</param>
/// <param name="_contact">The new 'number of the house's telephone' value</param>
public static bool _updateContact(int _idHouse, string _contact)


/// <summary>
/// <para> Updates the field 'numberFloors' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier of the house to be updated</param>
/// <param name="_numberFloors">The new 'house number of floors' value</param>
public static bool _updateNumberFloors(int _idHouse, int _numberFloors)


/// <summary>
/// <para> Updates the field 'numberDivisions' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier of the house to be updated</param>
/// <param name="_numberDivisions">The new 'house number of divisions' value</param>
public static bool _updateNumberDivisions(int _idHouse, int _numberDivisions)


/// <summary>
/// <para> Updates the field 'area' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier of the house to be updated</param>
/// <param name="_area">The new 'house terrain area' value</param>
public static bool _updateArea(int _idHouse, float _area)


/// <summary>
/// <para> Updates the field 'width' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier of the house to be updated</param>
/// <param name="_width">The new 'house width (X axis) value'</param>
public static bool _updateWidth(int _idHouse, float _width)


/// <summary>
/// <para> Updates the field 'length' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier of the house to be updated</param>
/// <param name="_length"> The new 'housep's length (Y Axis)' value</param>
public static bool _updateLength(int _idHouse, float _length)


/// <summary>
/// <para> Updates the field 'houseImage' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier of the house to be updated</param>
/// <param name="_houseImage">The new 'house's image' value</param>
public static bool _updateHouseImage(int _idHouse, byte[] _houseImage)
```

```
/// <summary>
/// <para> Updates the field 'houseYopology' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier of the house to be updated</param>
/// <param name="_houseTopology">The new 'house's topology image' value</param>
public static bool _updateHouseTopology(int _idHouse, byte[] _houseTopology)


/// <summary>
/// <para> Deletes an entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idHouse"> The identifier of the house to remove </param>
public static bool _delete(int _idHouse)


/// <summary>
/// <para> Deletes all entries </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
public static bool _deleteAll()
```

_ImageInformation

```
/// <summary>
/// <para> Selects all information from one House Image Info </para>
/// <para> Returns: A list: int idImageInfo, float scale, bool scaleDefiniedOnX, float
          imageHeightOnScale, float imageWidthOnScale </para>
/// </summary>
/// <param name="_idHouse"> The identifier of the house connected with the house image information
    </param>
public static List<object> _select_houseID(int _idHouse)


/// <summary>
/// <para> Selects all information from one House Image Info </para>
/// <para> Returns: A list: int idImageInfo, float scale, bool scaleDefiniedOnX, float
          imageHeightOnScale, float imageWidthOnScale </para>
/// </summary>
/// <param name="_idImageInfo"> The identifier of the house image info to select </param>
public static List<object> _select_houseImageInfoID(int _idImageInfo)


/// <summary>
/// <para> Selects all House Image Infos </para>
/// <para> Returns: A list of lists containing: int idImageInfo, float scale, bool
          scaleDefiniedOnX, float imageHeightOnScale, float imageWidthOnScale </para>
/// </summary>
public static List<List<object>> _selectAll()


/// <summary>
/// <para> Checks if the given house image info exists on the database </para>
/// <para> Returns: True or False depending on the success of the operation </para>
/// </summary>
/// <param name="_idImageInfo"> The identifier of the house image info to check </param>
public static bool _exists(int _idImageInfo)


/// <summary>
/// <para> Creates a house image info with the all the information </para>
/// <para> The house image info identifier as an 'Int' or '-1' if an error occurs </para>
/// </summary>
/// <param name="_scale"> The house scale obtained from one of its walls </param>
/// <param name="_scaleDefinedOnX"> The value representing if the scale was defined from the X Axis or
                                    not </param>
/// <param name="_imageHeightOnScale"> The image Height during the scale definition (used for scaling
                                       purposes) </param>
/// <param name="_imageWidthOnScale"> The image Width during the scale definition (used for scaling
                                      purposes) </param>
public static int _insert(float _scale, bool _scaleDefinedOnX, float _imageHeightOnScale, float
                          _imageWidthOnScale)
```

```
/// <summary>
/// <para> Creates a house image info with the all the information </para>
/// <para> The house identifier as an 'Int' or '-1' if an error occurs </para>
/// </summary>
/// <param name="_idImageInfo">The identifier of the house image info to insert</param>
/// <param name="_scale"> The house scale obtained from one of its walls </param>
/// <param name="_scaleDefinedOnX"> The value representing if the scale was defined from the X Axis or
                                    not </param>
/// <param name="_imageHeightOnScale"> The image Height during the scale definition (used for scaling
                                       purposes) </param>
/// <param name="_imageWidthOnScale"> The image Width during the scale definition (used for scaling
                                      purposes) </param>
public static int _insert(int _idImageInfo, float _scale, bool _scaleDefinedOnX, float
                          _imageHeightOnScale, float _imageWidthOnScale)

/// <summary>
/// <para> Updates the fields on house image info Entity </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier connected to the house image info entry to be
                            updated</param>
/// <param name="_scale"> The new 'house scale' value </param>
/// <param name="_scaleDefinedOnX"> The new value representing if the scale was defined on the X Axis
                                    not </param>
/// <param name="_imageHeightOnScale">The new 'image height when the house scale was defined'
                                      value</param>
/// <param name="_imageWidthOnScale"> The new 'image width when the house scale was defined' value
</param>
public static bool _update_houseID(int _idHouse, float _scale, bool _scaleDefinedOnX, float
                                   _imageHeightOnScale, float _imageWidthOnScale)

/// <summary>
/// <para> Updates the fields on house image info Entity </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idImageInfo"> The house image info identifier of the entry to be updated</param>
/// <param name="_scale"> The new 'house scale' value </param>
/// <param name="_scaleDefinedOnX"> The new value representing if the scale was defined on the X Axis
                                    or not </param>
/// <param name="_imageHeightOnScale">The new 'image height when the house scale was defined'
                                      value</param>
/// <param name="_imageWidthOnScale"> The new 'image width when the house scale was defined' value
    </param>
public static bool _update_houseImageInfoID(int _idImageInfo, float _scale, bool _scaleDefinedOnX,
                                            float _imageHeightOnScale, float _imageWidthOnScale)

/// <summary>
/// <para> Updates the field 'Scale' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier related with the house image info to be updated
    </param>
/// <param name="_scale"> The new 'scale' value </param>
public static bool _updateScale_houseID(int _idHouse, float _scale)


/// <summary>
/// <para> Updates the field 'Scale' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house image info identifier of the entry to be updated</param>
/// <param name="_scale"> The new 'scale' value </param>
public static bool _updateScale_houseImageInfoID(int _idImageInfo, float _scale)


/// <summary>
/// <para> Updates the field '_scaleDefinedOnX' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier related with the house image info to be updated
    </param>
/// <param name="_scaleDefinedOnX"> The new value representing if the scale was defined on the X Axis
                                    or not </param>
public static bool _updateScaleDefinedOnX_houseID(int _idHouse, bool _scaleDefinedOnX)
```

```
/// <summary>
/// <para> Updates the field '_scaleDefinedOnX' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idImageInfo"> The house image info identifier of the entry to be updated</param>
/// <param name="_scaleDefinedOnX"> The new value representing if the scale was defined on the X Axis
                                    or not </param>
public static bool _updateScaleDefinedOnX_houseImageInfoID(int _idImageInfo, bool _scaleDefinedOnX)


/// <summary>
/// <para> Updates the field 'imageHeigthOnScale' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier related with the house image info to be updated
</param>
/// <param name="_imageHeightOnScale"> The new 'image height when the house scale was defined' value
</param>
public static bool _updateImageHeigthOnScale_houseID(int _idHouse, float _imageHeightOnScale)


/// <summary>
/// <para> Updates the field 'imageHeigthOnScale' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idImageInfo"> The house image info identifier of the entry to be updated </param>
/// <param name="_imageHeightOnScale"> The new 'image height when the house scale was defined' value
</param>
public static bool _updateImageHeigthOnScale_houseImageInfoID(int _idImageInfo, float
_imageHeightOnScale)


/// <summary>
/// <para> Updates the field 'imageWidthOnScale' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHouse"> The house identifier related with the house image info to be updated
    </param>
/// <param name="_imageWidthOnScale"> The new 'image width when the house scale was defined' value
    </param>
public static bool _updateImageWidthOnScale_houseID(int _idHouse, float _imageWidthOnScale)


/// <summary>
/// <para> Updates the field 'imageWidthOnScale' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idImageInfo"> The house image info identifier of the entry to be updated</param>
/// <param name="_imageWidthOnScale"> The new 'image width when the house scale was defined' value
    </param>
public static bool _updateImageWidthOnScale_houseImageInfoID(int _idImageInfo, float
                                                             _imageWidthOnScale)

/// <summary>
/// <para> Deletes an entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idHouse"> The identifier of the house connect to the House Image Info to remove
    </param>
public static bool _delete_houseID(int _idHouse)

/// <summary>
/// <para> Deletes an entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idImageInfo"> The identifier of the house image info to remove </param>
public static bool _delete_houseImageInfoID(int _idImageInfo)


/// <summary>
/// <para> Deletes all entries </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
public static bool _deleteAll()
```

_Door

```
/// <summary>
/// <para> Selects all doors </para>
/// <para> Returns: A list of lists, each sub list contains: int idDoor, int idHouse, string
          _object</para>
/// </summary>
public static List<List<object>> _selectAll()


/// <summary>
/// <para> Selects all doors from a specific house </para>
/// <para> Returns: A list of lists, each sub list contains: int idDoor, int idHouse, string
_object</para>
/// </summary>
/// <param name="_idHouse">The identifier of the house</param>
public static List<List<object>> _selectAll(int _idHouse)


/// <summary>
/// <para> Selects all information from a door associated with the given identifier </para>
/// <para> Returns: A list: int idDoor, int idHouse, string object<</para>
/// </summary>
/// <param name="_idDoor">Identifier of the door to select</param>
public static List<object> _select(int _idDoor)


/// <summary>
/// <para> Checks if the given house door exists on the database </para>
/// <para> Returns: True or False depending on the success of the operation </para>
/// </summary>
/// <param name="_idDoor"> The identifier of the house door to check </param>
public static bool _exists(int _idDoor)


/// <summary>
/// <para> Creates a new Door </para>
/// <para> Returns: The new door identifier as an int, or -1 if an error occurs </para>
/// </summary>
/// <param name="_idHouse">The identifier of the house where this door belongs</param>
/// <param name="_object">The string value representing the door object</param>
public static int _insert(int _idHouse, string _object)


/// <summary>
/// <para> Creates a new Door </para>
/// <para> Returns: The new door identifier as an int, or -1 if an error occurs </para>
/// </summary>
/// <param name="_idDoor">The identifier of the door's house</param>
/// <param name="_idHouse">The identifier of the house where this door belongs</param>
/// <param name="_object">The string value representing the door object</param>
public static int _insert(int _idDoor, int _idHouse, string _object)


/// <summary>
/// <para> Updates the door information </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idDoor">The identifier of the door's house</param>
/// <param name="_idHouse">The new 'identifier of the house where this door belongs' value</param>
/// <param name="_object">The new 'string value representing the door object' value</param>
public static bool _update(int _idDoor, int _idHouse, string _object)


/// <summary>
/// <para> Updates the field 'idHouse' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idDoor">The identifier of the door's house</param>
/// <param name="_idHouse">The new 'identifier of the house where this door belongs' value</param>
public static bool _updateHouseID(int _idDoor, int _idHouse)
```

```
/// <summary>
/// <para> Updates field 'object' </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idDoor">The identifier of the door's house</param>
/// <param name="_object">The new 'string value representing the door object' value</param>
public static bool _updateObject(int _idDoor, string _object)


/// <summary>
/// <para>Deletes a door, given its identifier</para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name ="_idDoor">Identifier of the door to be eliminated</param>
public static bool _deleteWithDoorID(int _idDoor)


/// <summary>
/// <para>Deletes the doors related to the given house</para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name ="_idHouse">The house identifier of the doors to be eliminated</param>
public static bool _deleteWithHouseID(int _idHouse)


/// <summary>
/// <para> Deletes a door, given the object that defines it</para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_object">The string value representing the door object</param>
public static bool _deleteWithObject(string _object)


/// <summary>
/// <para> Deletes all door in Door Entity </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
public static bool _deleteAll()
```

_Division

```
/// <summary>
/// <para> Selects all House Division Entries </para>
/// <para> Returns: A list of lists, each sub list contains: int idDivision, int idHouse, string
          object, string name, string description </para>
/// </summary>
public static List<List<object>> _selectAll()


/// <summary>
/// <para> Selects all House Division Entries related to a specific House </para>
/// <para> Returns: A list of lists, each sub list contains: int idDivision, int idHouse, string
          object, string name, string description </para>
/// </summary>
/// <param name="_idHouse"> The identifier of the house </param>
public static List<List<object>> _selectAll(int _idHouse)


/// <summary>
/// <para> Selects all House Division Entries related to a specific House </para>
/// <para> Returns: A list of lists, each sub list contains: int idDivision, int idHouse, string
          object, string name, string description </para>
/// </summary>
/// <param name="_idHouse"> The identifier of the house </param>
public static List<_Monitor._monitorHome._divisionStruct> _selectAllv2(int _idHouse)


/// <summary>
/// <para> Selects all information from a House Division Entry</para>
/// <para> Returns: A list: int idDivision, int idHouse, string object, string name, string description
     </para>
/// </summary>
/// <param name="_idDivision">Identifier of the division to select</param>
public static List<object> _select(int _idDivision)
```

```
/// <summary>
/// <para> Checks if the given house division exists on the database </para>
/// <para> Returns: True or False depending on the success of the operation </para>
/// </summary>
/// <param name="_idDivision"> The identifier of the house division to check </param>
public static bool _exists(int _idDivision)


/// <summary>
/// <para> Creates a new House Division </para>
/// <para> Returns: The new division identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_idHouse">The identifier of the division's house</param>
/// <param name="_object">Value representing the object that describes the division physical
                         format</param>
/// <param name="_name">The name of the division</param>
/// <param name="_description">The description of the division</param>
public static int _insert(int _idHouse, string _object, string _name, string _description)


/// <summary>
/// <para> Creates a new House Division </para>
/// <para> Returns: The new division identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_idDivision">The identifier of the division to create</param>
/// <param name="_idHouse">The identifier of the division's house</param>
/// <param name="_object">Value representing the object that describes the division physical
                         format</param>
/// <param name="_name">The name of the division</param>
/// <param name="_description">The description of the division</param>
public static int _insert(int _idDivision, int _idHouse, string _object, string _name, string
                         _description)

/// <summary>
/// <para> Updates all the fields of a House Division </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idDivision">The identifier of the division to create</param>
/// <param name="_idHouse">The new 'identifier of the division's house' value</param>
/// <param name="_object">The new value representing the object that describes the division physical
                         format</param>
/// <param name="_name">The new 'name of the division' value</param>
/// <param name="_description">The new 'description of the division' value</param>
public static bool _update(int _idDivision, int _idHouse, string _object, string _name, string
                          _description)

/// <summary>
/// <para> Updates the field 'idHouse' of a House Division </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idDivision">Identifier of the Division to be updated</param>
/// <param name="_idHouse">The new 'identifier of the division's house' value</param>
public static bool _updateHouseID(int _idDivision, int _idHouse)


/// <summary>
/// <para> Updates the field 'object' of a House Division </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idDivision">Identifier of the Division to be updated</param>
/// <param name="_object">The new value representing the object that describes the division physical
                         format</param>
public static bool _updateObject(int _idDivision, string _object)


/// <summary>
/// <para> Updates the field 'name'of a House Division </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idDivision">Identifier of the Division to be updated</param>
/// <param name="_name">The new 'name of the division' value</param>
public static bool _updateName(int _idDivision, string _name)
```

```
/// <summary>
/// <para> Updates the field 'description' of a House Division </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idDivision">Identifier of the Division to be updated</param>
/// <param name="_description">The new 'description of the division' value</param>
public static bool _updateDescription(int _idDivision, string _description)


/// <summary>
/// <para> Deletes a Division </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name ="_idDivision">Identifier of the division to be eliminated</param>
public static bool _delete(int _idDivision)


/// <summary>
/// <para> Deletes all divisions </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
public static bool _deleteAll()


/// <summary>
/// <para> Deletes all divisions of a specific house </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idHouse"> The identifier of the house </param>
public static bool _deleteAll(int _idHouse)


/// <summary>
/// <para> Checks if there is any division information </para>
/// <para> Returns: True if any division exists, false if it doens't</para>
/// </summary>
/// <param name="_idHouse"> The identifier of the house </param>
public static bool _houseHasDivisions(int _idHouse)
```

_User

_Information

```
/// <summary>
/// <para> Retrives the essencial information to fill the user loggeg in Area (username and avatar)
    </para>
/// <para> Returns: A List of objects: string username, byte[] avatarImage </para>
/// </summary>
/// <param name="_username">The username of the user logged in</param>
public static List<object> _selectLogin(string _username)



/// <summary>
/// <para> Selects all users </para>
/// <para> Returns: A list of lists, each sub list contains: int idUser, int idHouse, string firstname,
           string lastname, string birthDate, string sex, string nationality, string contact, string
           email, Image avatarImage, int idCredential, string username, string accessType, string
           lastAccessDate, bool userStatus </para>
/// </summary>
public static List<List<object>> _selectAll()



/// <summary>
/// <para> Selects all users IDs </para>
/// <para> Returns: A list containing: int idUser1, int idUser2, ... </para>
/// </summary>
public static List<object> _selectAllUsersIDs()
```

```
/// <summary>
/// <para> Selects all information of an user </para>
/// <para> Returns: A list: int idUser, int idHouse, string firstname, string lastname, string
           birthDate, string sex, string nationality, string contact, string email, Image avatarImage,
           bool active </para>
/// </summary>
/// <param name="_idUser">Identifier of the user to select</param>
public static List<object> _select(int _idUser)


/// <summary>
/// <para> Checks if the given user exists on the database </para>
/// <para> Returns: True or False depending on the success of the operation </para>
/// </summary>
/// <param name="_idUser"> The identifier of the user to check </param>
public static bool _exists(int _idUser)


/// <summary>
/// <para> Creates a new User </para>
/// <para> Returns: The new user identifier as an int, or -1 if an error occurs </para>
/// </summary>
/// <param name="_idHouse"> The identifier of the user's house </param>
/// <param name="_firstname"> The first name of a person </param>
/// <param name="_lastname"> The last name of a person </param>
/// <param name="_birthDate"> The date of birth of the user </param>
/// <param name="_sex"> The sex of the user (male or female) </param>
/// <param name="_contact"> The cell phone contact of the user </param>
/// <param name="_nationality"> Nationality of the user </param>
/// <param name="_avatarImage"> An image of the user, used for presentation purposes only </param>
/// <param name="_email"> The electronic address of the user </param>
/// <param name="_active"> The value that indicates if the user is active or not </param>
public static int _insert(int _idHouse, string _firstname, string _lastname, string _birthDate, string
                      _sex, string _nationality, string _contact, string _email, byte[] _avatarImage)

/// <summary>
/// <para> Creates a new User </para>
/// <para> Returns: The new user identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_idUser"> The identifier of the user to create </param>
/// <param name="_idHouse"> The identifier of the user's house </param>
/// <param name="_firstname"> The first name of a person </param>
/// <param name="_lastname"> The last name of a person </param>
/// <param name="_birthDate"> The date of birth of the user </param>
/// <param name="_sex"> The sex of the user (male or female)</param>
/// <param name="_contact"> The cell phone contact of the user </param>
/// <param name="_nationality"> Nationality of the user </param>
/// <param name="_avatarImage"> An image of the user, used for presentation purposes only</param>
/// <param name="_email"> The electronic address of the user </param>
/// <param name="_active"> The value that indicates if the user is active or not </param>
public static int _insert(int _idUser, int _idHouse, string _firstname, string _lastname, string
     _birthDate, string _sex, string _nationality, string _contact, string _email, byte[] _avatarImage)


/// <summary>
/// <para> Updates all fields of an User </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idUser">Identifier of the User to be updated</param>
/// <param name="_idHouse">The new 'identifier of the user's house' value </param>
/// <param name="_firstname">The new 'first name of a person' value </param>
/// <param name="_lastname">The new 'last name of a person' value </param>
/// <param name="_birthDate">The new 'date of birth of the user' value </param>
/// <param name="_sex">The new 'sex of the user (male or female)' value </param>
/// <param name="_contact">The new 'cell phone contact of the user' value </param>
/// <param name="_nationality">The new 'nationality of the user' value </param>
/// <param name="_avatarImage">The new 'image of the user' value </param>
/// <param name="_email"> The new 'electronic address of the user' value </param>
/// <param name="_active"> The new 'active' value </param>
public static bool _update(int _idUser, int _idHouse, string _firstname, string _lastname, string
     _birthDate, string _sex, string _nationality, string _contact, string _email, byte[] _avatarImage)
```

```
/// <summary>
/// <para> Updates the field 'idHouse' of an User </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idUser">Identifier of the User to be updated</param>
/// <param name="_idHouse">The new 'identifier of the user's house' value </param>
public static bool _updateHouseID(int _idUser, int _idHouse)


/// <summary>
/// <para> Updates the field 'idHouse' of all Users </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idHouse">The new 'identifier of the user's house' value </param>
public static bool _updateAllHouseID(int _idHouse)


/// <summary>
/// <para> Updates the field 'firstname' of an User </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idUser">Identifier of the User to be updated</param>
/// <param name="_firstname">The new 'first name of a person' value </param>
public static bool _updateFirstname(int _idUser, string _firstname)


/// <summary>
/// <para> Updates the field 'lastname' of an User </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idUser">Identifier of the User to be updated</param>
/// <param name="_lastname">The new 'last name of a person' value </param>
public static bool _updateLastname(int _idUser, string _lastname)


/// <summary>
/// <para> Updates the field 'birthDate' of an User </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idUser">Identifier of the User to be updated</param>
/// <param name="_birthDate">The new 'date of birth of the user' value </param>
public static bool _updateBirthdate(int _idUser, string _birthDate)


/// <summary>
/// <para> Updates the field 'sex' of an User </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idUser">Identifier of the User to be updated</param>
/// <param name="_sex">The new 'sex of the user (male or female)' value </param>
public static bool _updateSex(int _idUser, string _sex)


/// <summary>
/// <para> Updates the field 'contact' of an User </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idUser">Identifier of the User to be updated</param>
/// <param name="_contact">The new 'cell phone contact of the user' value </param>
public static bool _updateContact(int _idUser, string _contact)


/// <summary>
/// <para> Updates the field 'nationality' of an User </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idUser">Identifier of the User to be updated</param>
/// <param name="_nationality">The new 'nationality of the user' value </param>
public static bool _updateNationality(int _idUser, string _nationality)
```

```
/// <summary>
/// <para> Updates the field 'avatarImage' of an User </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idUser">Identifier of the User to be updated</param>
/// <param name="_avatarImage">The new 'image of the user' value </param>
public static bool _updateAvatarImage(int _idUser, byte[] _avatarImage)


/// <summary>
/// <para> Updates the field 'email' of an User </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idUser">Identifier of the User to be updated</param>
/// <param name="_email"> The new 'electronic address of the user' value </param>
public static bool _updateEmail(int _idUser, string _email)


/// <summary>
/// <para> Deletes an user </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name ="_idUser">Identifier of the User to be eliminated</param>
public static bool _delete(int _idUser)


/// <summary>
/// <para> Deletes an User and his Credentials </para>
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name ="_idUser">Identifier of the User to be eliminated</param>
public static bool _deleteUserAndCredentials(int _idUser)


/// <summary>
/// <para> Deletes all users </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <returns>True or False, depending on the operation success</returns>
public static bool _deleteAll()
```

_Credential

```
/// <summary>
/// <para> Selects all users' userCredential </para>
/// <para> Returns: A list of lists, each sub list contains: int idCredential, int idUser, string
          username, string password, string accessType, string lastAccessDate, bool userState, string
          passwordRecoveryQuestion, string passwordRecoveryAnswer </para>
/// </summary>
public static List<List<object>> _selectAll()


/// <summary>
/// <para> Selects all credential information from an user </para>
/// <para> Returns: A list: int idCredential, int idUser, string username, string password, string
          accessType, string lastAccessDate, bool userState, string passwordRecoveryQuestion, string
          passwordRecoveryAnswer </para>
/// </summary>
/// <param name="_idCredential">Identifier of the User Credential Entry to be selected</param>
public static List<object> _selectWithUserCredentialID(int _idCredential)


/// <summary>
/// <para> Selects all credential information from an user </para>
/// <para> Returns: A list: int idCredential, int idUser, string username, string password, string
          accessType, string lastAccessDate, bool userState, string passwordRecoveryQuestion, string
          passwordRecoveryAnswer </para>
/// </summary>
/// <param name="_username"> The username of the user to be selected</param>
public static List<object> _selectWithUsername(string _username)
```

```
/// <summary>
/// <para> Selects all credential information from an user </para>
/// <para> Returns: A list: int idCredential, int idUser, string username, string password, string
          accessType, string lastAccessDate, bool userState, string passwordRecoveryQuestion, string
          passwordRecoveryAnswer </para>
/// </summary>
/// <param name="_idUser">Identifier of the user who's userCredential are going to be selected</param>
public static List<object> _selectWithUserID(int _idUser)


/// <summary>
/// <para> Checks if the given user credential exists on the database </para>
/// <para> Returns: True or False depending on the success of the operation </para>
/// </summary>
/// <param name="_idCredential"> The identifier of the user credential to check </param>
public static bool _exists(int _idCredential)


/// <summary>
/// <para> Creates a new Credential with the all information </para>
/// <para> Returns: The new entry identifier as an int, or -1 if an error occurs </para>
/// </summary>
/// <param name="_idUser">Identifier of the User</param>
/// <param name="_username">Virtual name of the user in the HUEPS platform</param>
/// <param name="_password">Password that allows the user to access the HUEPS platform</param>
/// <param name="_accessType">The name of the acces type of the user</param>
/// <param name="_lastAccessDate">The last time that an user used HUEPS</param>
/// <param name="_userStatus">Value that allows to see if an user is online or offline</param>
/// <param name="_passwordRecoveryQuestion">Saves the question that allows to obtain the lost
                                            password</param>
/// <param name="_passwordRecoveryAnswer">Saves the answer for the question that allows to obtain the
                                          lost password</param>
public static int _insert(int _idUser, string _username, string _password, string _accessType, string
   _lastAccessDate, bool _userStatus, string _passwordRecoveryQuestion, string _passwordRecoveryAnswer)


/// <summary>
/// <para> Creates a new Credential with the all information </para>
/// <para> Returns: The new entry identifier as an int, or -1 if an error occurs </para>
/// </summary>
/// <param name="_idCredential">The identifier of the credential entry to insert</param>
/// <param name="_idUser">Identifier of the User</param>
/// <param name="_username">Virtual name of the user in the HUEPS platform</param>
/// <param name="_password">Password that allows the user to access the HUEPS platform</param>
/// <param name="_email">The electronic address of the user</param>
/// <param name="_accessType">The name of the acces type of the user</param>
/// <param name="_lastAccessDate">The last time that an user used HUEPS</param>
/// <param name="_userStatus">Value that allows to see if an user is online or offline</param>
/// <param name="_passwordRecoveryQuestion">Saves the question that allows to obtain the lost
                                            password</param>
/// <param name="_passwordRecoveryAnswer">Saves the answer for the question that allows to obtain the
                                          lost password</param>
public static int _insert(int _idCredential, int _idUser, string _username, string _password, string
        _accessType, string _lastAccessDate, bool _userStatus, string _passwordRecoveryQuestion, string
                                                                    _passwordRecoveryAnswer)

/// <summary>
/// <para> Updates all fields </para>
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name="_idCredential">Identifier of the User Credential to be updated</param>
/// <param name="_idUser">Identifier of the User</param>
/// <param name="_username">The new 'username' value</param>
/// <param name="_password">The new 'password' value</param>
/// <param name="_accessType">The new 'accessType' value</param>
/// <param name="_lastAccessDate">The new 'lastAccessDate' value</param>
/// <param name="_userStatus">The new 'userStatus' value</param>
/// <param name="_passwordRecoveryQuestion">The new 'passwordRecoveryQuestion' value</param>
/// <param name="_passwordRecoveryAnswer">The new 'passwordRecoveryAnswer' value</param>
public static bool _updateWithUserCredentialID(int _idCredential, int _idUser, string _username, string
                      _password, string _accessType, string _lastAccessDate, bool _userStatus, string
                                          _passwordRecoveryQuestion, string _passwordRecoveryAnswer)
```

```
/// <summary>
/// <para> Updates all fields </para>
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name="_idUser">Identifier of the User to be updated </param>
/// <param name="_username">The new 'username' value</param>
/// <param name="_password">The new 'password' value</param>
/// <param name="_accessType">The new 'accessType' value</param>
/// <param name="_lastAccessDate">The new 'lastAccessDate' value</param>
/// <param name="_userStatus">The new 'userStatus' value</param>
/// <param name="_passwordRecoveryQuestion">The new 'passwordRecoveryQuestion' value</param>
/// <param name="_passwordRecoveryAnswer">The new 'passwordRecoveryAnswer' value</param>
public static bool _updateWithUserID(int _idUser, string _username, string _password, string
_accessType, string _lastAccessDate, bool _userStatus, string _passwordRecoveryQuestion, string
                                                                        _passwordRecoveryAnswer)


/// <summary>
/// <para> Updates the field 'idUser' </para>
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name="_idCredential">Identifier of the Credential to be updated</param>
/// <param name="_idUser">The new 'idUser' value</param>
public static bool _updateUserID(int _idCredential, int _idUser)



/// <summary>
/// <para> Updates the field 'username' </para>
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name="_idUser">Identifier of the User</param>
/// <param name="_username">The new 'username' value</param>
public static bool _updateUsernameWithUserID(int _idUser, string _username)



/// <summary>
/// <para> Updates the field 'username' </para>
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name="_idCredential">Identifier of the Credential to be updated</param>
/// <param name="_username">The new 'username' value</param>
public static bool _updateUsernameWithUserCredentialID(int _idCredential, string _username)


/// <summary>
/// <para> Updates the field 'password' </para>
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name="_idUser">Identifier of the User</param>
/// <param name="_password">The new 'password' value</param>
public static bool _updatePasswordWithUserID(int _idUser, string _password)


/// <summary>
/// <para> Updates the field 'password' </para>
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name="_idCredential">Identifier of the Credential to be updated</param>
/// <param name="_password">The new 'password' value</param>
public static bool _updatePasswordWithUserCredentialID(int _idCredential, string _password)


/// <summary>
/// <para> Updates the field 'accessType' </para>
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name="_idUser">Identifier of the User</param>
/// <param name="_accessType">The new 'accessType' value</param>
public static bool _updateAccessTypeWithUserID(int _idUser, string _accessType)
```

```
/// <summary>
/// <para> Updates the field 'accessType' </para>
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name="_idCredential">Identifier of the Credential to be updated</param>
/// <param name="_acessType">The new 'accessType' value</param>
public static bool _updateAccessTypeWithUserCredentialID(int _idCredential, string _accessType)


/// <summary>
/// <para> Updates the field 'lastAccessDate' </para>
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name="_idUser">Identifier of the User</param>
/// <param name="_lastAccessDate">The new 'lastAccessDate' value</param>
public static bool _updateLastAccessDateWithUserID(int _idUser, string _lastAccessDate)


/// <summary>
/// <para> Updates the field 'lastAccessDate' </para>
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name="_idCredential">Identifier of the Credential to be updated</param>
/// <param name="_lastAccessDate">The new 'lastAccessDate' value</param>
public static bool _updateLastAccessDateWithUserCredentialID(int _idCredential, string _lastAccessDate)


/// <summary>
/// <para> Updates the field 'userStatus' in the entity User Credential, entry given by the respective
        user identifier </para>
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name="_idUser">Identifier of the User</param>
/// <param name="_userStatus">The new 'userStatus' value</param>
public static bool _updateUserStatusWithUserID(int _idUser, bool _userStatus)


/// <summary>
/// <para> Updates the field 'userStatus' </para>
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name="_idCredential">Identifier of the Credential to be updated</param>
/// <param name="_userStatus">The new 'userStatus' value</param>
public static bool _updateUserStatusWithUserCredentialID(int _idCredential, bool _userStatus)


/// <summary>
/// <para> Updates the field 'passwordRecoveryQuestion' </para>
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name="_idUser">Identifier of the User</param>
/// <param name="_passwordRecoveryQuestion">The new 'passwordRecoveryQuestion' value</param>
public static bool _updatePasswordRecoveryQuestionWithUserID(int _idUser, string
                                                    _passwordRecoveryQuestion)

/// <summary>
/// <para> Updates the field 'passwordRecoveryQuestion' </para>
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name="_idCredential">Identifier of the Credential to be updated</param>
/// <param name="_passwordRecoveryQuestion">The new 'passwordRecoveryQuestion' value</param>
public static bool _updatePasswordRecoveryQuestionWithUserCredentialID(int _idCredential, string
                                                                _passwordRecoveryQuestion)

/// <summary>
/// <para> Updates the field 'passwordRecoveryAnswer' </para>
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name="_idUser">Identifier of the User</param>
/// <param name="_passwordRecoveryAnswer">The new 'passwordRecoveryAnswer' value</param>
public static bool _updatePasswordRecoveryAnswerWithUserID(int _idUser, string _passwordRecoveryAnswer)
```

```
/// <summary>
/// <para> Updates the field 'passwordRecoveryAnswer'
/// <para> Returns: True or False, depending of the operation success </para>
/// </summary>
/// <param name="_idCredential">Identifier of the Credential to be updated</param>
/// <param name="_passwordRecoveryAnswer">The new 'passwordRecoveryAnswer' value</param>
public static bool _updatePasswordRecoveryAnswerWithUserCredentialID(int _idCredential, string
                                                                     _passwordRecoveryAnswer)

/// <summary>
/// <para> Deletes an entry in User Credential Entity </para>
/// <para> Returns: True or false, depending on the sucess of the operation </para>
/// </summary>
/// <param name ="_idCredential">Identifier of the Credential to be eliminated</param>
public static bool _deleteWithUserCredentialID(int _idCredential)


/// <summary>
/// <para> Deletes an entry in User Credential Entity </para>
/// <para> Returns: True or false, depending on the sucess of the operation </para>
/// </summary>
/// <param name ="_idUser">Identifier of the User to be eliminated</param>
public static bool _deleteWithUserID(int _idUser)


/// <summary>
/// <para> Deletes all entries </para>
/// <para> Returns: True or false, depending on the sucess of the operation </para>
/// </summary>
public static bool _deleteAll()


/// <summary>
/// <para> Checks if the user is registered in the database an if his password is correct </para>
/// <para> Returns: The accessType of the user or null if the an error occurs </para>
/// </summary>
/// <param name="_username">User Virtual Name on the HUEPS platform</param>
/// <param name="_password">User password to be checked</param>
public static string _checkUser(string _username, string _password)


/// <summary>
/// <para> Gets the password recovery question of a specific user </para>
/// <para> Returns: The password recovery question of the user or null if an error occurs</para>
/// </summary>
/// <param name="_username"> The HUEPS/HUEPS 'username' of the user </param>
public static string _selectRecoveryQuestion(string _username)


/// <summary>
/// <para> Confirms if the user answer matches the one stored o the database </para>
/// <para> Returns: True or false, depending on the sucess of the operation </para>
/// </summary>
/// <param name="_username"> The HUEPS/HUEPS 'username' of the user</param>
/// <param name="_answer">  The answer given by the user</param>
public static bool _checkUserRecoveryAnswer(string _username, string _answer)
```

_Location

```
/// <summary>
/// <para> Selects all Users Positions from the Entity User Location </para>
/// <para> Returns: A list of lists, each sub list contains: int idLocation, int idUser, int
          idDivision, string date, bool movement, bool realLocation</para>
/// </summary>
public static List<List<object>> _selectAll()


/// <summary>
/// <para> Selects all information from an User Location associated with the given identifier </para>
/// <para> Returns: A list: int idLocation, int idUser, int idDivision, string date, bool movement,
          bool realLocation </para>
/// </summary>
/// <param name="_idLocation">Identifier of the entry to be selected</param>
public static List<object> _select(int _idLocation)
```

```
/// <summary>
/// <para> Selects the last known user position inside the house </para>
/// <para> Returns: A list: int idLocation, int idUser, int idDivision, string date, bool movement,
          bool realLocation </para>
/// </summary>
/// <param name="_idUser"> Identifier of the user </param>
public static List<object> _selectLastUserLocation(int _idUser)


/// <summary>
/// <para> </para>
/// <para> Returns: </para>
/// </summary>
public static List<List<object>> _selectAllLastUsersLocation()


/// <summary>
/// <para> Creates a new user position entry </para>
/// <para> Returns: The new entry identifier as an int, or -1 if an error occurs </para>
/// </summary>
/// <param name="_idLocation"> The identifier of the entry to created </param>
/// <param name="_idUser"> The identifier of the user to be associated with this position </param>
/// <param name="_idDivision"> The identifier of the division where the user is </param>
/// <param name="_date"> The date and time of the acquired position </param>
/// <param name="_movement"> Indicates if the user stayed still on the division or if some movement
                            occured </param>
public static int _insert(int _idLocation, int _idUser, int _idDivision, string _date, bool _movement,
                          bool _realLocation = true)

/// <summary>
/// <para> Creates a new user location entry </para>
/// <para> Returns: The new entry identifier as an int, or -1 if an error occurs </para>
/// </summary>
/// <param name="_idUser">The identifier of the user to be associated with this position</param>
/// <param name="_idDivision">The identifier of the division where the user is</param>
/// <param name="_date">The date and time of the acquired position</param>
/// <param name="_movement"> Indicates if the user stayed still on the division or if some movement
                            occured </param>
public static int _insert(int _idUser, int _idDivision, string _date, bool _movement, bool
                          _realLocation = true)

/// <summary>
/// <para> Updates all fields in the entity User Location, entry given by the respective identifier
</para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idLocation">Identifier of the User Location Entry to be updated</param>
/// <param name="_idUser">The new 'identifier of the user to be associated with this position'
value</param>
/// <param name="_idDivision">The new 'identifier of the division where the user is' value</param>
/// <param name="_date"> The new 'date and time of the acquired position' value </param>
/// <param name="_movement"> The new 'movement' value</param>
public static bool _update(int _idLocation, int _idUser, int _idDivision, string _date, bool _movement,
                           bool _realLocation)

/// <summary>
/// <para> Updates the field 'idUser' in the entity User Location, entry given by the respective
          identifier </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idLocation">Identifier of the User Location Entry</param>
/// <param name="_idUser">The new 'identifier of the user to be associated with this position'
                          value</param>
public static bool _updateUserID(int _idLocation, int _idUser)


/// <summary>
/// <para> Updates the field 'idDivision' in the entity User Location, entry given by the respective
          identifier </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idLocation">Identifier of the User Location Entry</param>
/// <param name="_idDivision">The new 'identifier of the division where the user is' value</param>
public static bool _updateDivisionID(int _idLocation, int _idDivision)
```

```
/// <summary>
/// <para> Updates the field 'date' in the entity User Location, entry given by the respective
        identifier </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idLocation"> Identifier of the User Location Entry </param>
/// <param name="_date"> The new 'date and time of the acquired position' value </param>
public static bool _updateDate(int _idLocation, string _date)


/// <summary>
/// <para> Updates the field 'movement' in the entity User Location, entry given by the respective
        identifier </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idLocation"> Identifier of the User Location Entry </param>
/// <param name="_movement"> The new 'movement' value </param>
public static bool _updateMovement(int _idLocation, bool _movement)


/// <summary>
/// <para> Updates the field 'realLocation' in the entity User Location, entry given by the respective
        identifier </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idLocation"> Identifier of the User Location Entry </param>
/// <param name="_realLocation"> The new 'realLocation' value </param>
public static bool _updateRealLocation(int _idLocation, bool _realLocation)


/// <summary>
/// <para> Deletes entry in User Location </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name ="_idLocation">Identifier of the entry to be eliminated</param>
public static bool _delete(int _idLocation)


/// <summary>
/// <para> Deletes entries in User Location associated with the given user. </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name ="_idUser">The identifier of the user to delete</param>
public static bool _deleteWithUserID(int _idUser)


/// <summary>
/// <para> Deletes all entries in User Location Entity </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
public static bool _deleteAll()
```

_ExternalServices

_Service

```
/// <summary>
/// <para> Selects all External Services from the Entity External Services </para>
/// <para> Returns: A list of lists, each sub list contains: int idExternalService, int idCategory,
        string name, string description, bool status </para>
/// </summary>
/// <param name="_categoryID"> True - Category ID, False - Category Name </param>
public static List<List<object>> _selectAll(bool _categoryID = false)


/// <summary>
/// <para> Selects all External Services with a specific External Service Status </para>
/// <para> Returns: A list of lists, each sub list contains: int idExternalService, int idCategory,
        string name, string description, bool status </para>
/// </summary>
/// <param name="_status"> The External Service Status used to filter the results </param>
public static List<List<object>> _selectAll(bool _status, bool _categoryID = false)
```

```
/// <summary>
/// <para> Selects an External Service </para>
/// <para> A list that contains: int idExternalService, int idCategory, string name, string
        description, bool status </para>
/// </summary>
/// <param name="_idExternalService"> Identifier of the External Service to select </param>
public static List<object> _select(int _idExternalService, bool _categoryID = false)


/// <summary>
/// <para> Selects an External Service </para>
/// <para> A list that contains: int idExternalService, int idCategory, string name, string
        description, bool status </para>
/// </summary>
/// <param name="_name"> The name of the External Service to select </param>
public static List<object> _select(string _name, bool _categoryID = false)


/// <summary>
/// <para> Creates a new External Service to be used </para>
/// <para> The new entry identifier as an int or -1 if an error occurs </para>
/// </summary>
/// <param name="_idExternalService"> The identifier of the External Service to create </param>
/// <param name="_idCategory"> The identifier of the current status of the External Service </param>
/// <param name="_name"> The name or abreviation of the External Service </param>
/// <param name="_description"> The description of the External Service </param>
/// <param name="_status"> Used to activate or deactivate the service </param>
public static int _insert(int _idExternalService, int _idCategory, string _name, string _description,
                          bool _status)


/// <summary>
/// <para> Creates a new External Service to be used. Checks if the service already exists. </para>
/// <para> An object list, {Service ID, Service Status}, Service ID = -1 if an error occurs. </para>
/// </summary>
/// <param name="_idCategory"> The identifier of the current status of the External Service </param>
/// <param name="_name"> The name or abreviation of the External Service </param>
/// <param name="_description"> The description of the External Service </param>
/// <param name="_status"> Used to activate or deactivate the service </param>
public static object[] _insert(int _idCategory, string _name, string _description, bool _status)


/// <summary>
/// <para> Updates all fields of an External Service </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idExternalService"> Identifier of the External Service to be updated</param>
/// <param name="_idCategory"> The new 'idCategory' value </param>
/// <param name="_name"> The new 'name' value </param>
/// <param name="_description"> The new 'description' value </param>
/// <param name="_status"> The new 'status' value </param>
public static bool _update(int _idExternalService, int _idCategory, string _name, string _description,
                           bool _status)


/// <summary>
/// <para> Updates all fields of an External Service using its Name as key </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idCategory"> The new 'idCategory' value </param>
/// <param name="_name"> The name of the service to update </param>
/// <param name="_description"> The new 'description' value </param>
public static bool _update(string _name, int _idCategory, string _description, bool _status)


/// <summary>
/// <para> Updates the 'name' field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idExternalService">Identifier of the External Services</param>
/// <param name="_name">The new 'name' value</param>
public static bool _updateName(int _idExternalService, string _name)
```

```
/// <summary>
/// <para> Updates the 'externalServiceCategoryID' field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_name"> Identifier of the External Services </param>
/// <param name="_externalServiceCategoryID"> The new '_externalServiceCategoryID' value </param>
public static bool _updateCategoryID(int _idExternalService, int _externalServiceCategoryID)


/// <summary>
/// <para> Updates the 'externalServiceCategoryID' field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_name"> The name of the External Service to update </param>
/// <param name="_externalServiceCategoryID"> The new '_externalServiceCategoryID' value </param>
public static bool _updateCategoryID(string _name, int _externalServiceCategoryID)


/// <summary>
/// <para> Updates the 'description' field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_name"> Identifier of the External Services </param>
/// <param name="_description"> The new 'description' value </param>
public static bool _updateDescription(int _idExternalService, string _description)


/// <summary>
/// <para> Updates the 'description' field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_name"> The name of the External Service to update </param>
/// <param name="_description"> The new 'description' value </param>
public static bool _updateDescription(string _name, string _description)


/// <summary>
/// <para> Updates the 'idExternalServiceState' field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_name"> Identifier of the External Services </param>
/// <param name="_status"> The new 'status' value </param>
public static bool _updateStatus(int _idExternalService, bool _status)


/// <summary>
/// <para> Updates the 'idExternalServiceState' field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_name"> Identifier of the External Services </param>
/// <param name="_status"> The new 'status' value </param>
public static bool _updateState(string _name, bool _status)


/// <summary>
/// <para> Deletes an External Service </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name ="_idExternalService"> Identifier of the external service to be eliminated </param>
public static bool _delete(int _idExternalService)


/// <summary>
/// <para> Deletes an External Service </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name ="_name"> The name of the external service to be eliminated </param>
public static bool _delete(string _name)

/// <summary>
/// <para> Deletes the existing entries in External Service Entity </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
public static bool _deleteAll()
```

_Rule

```
/// <summary>
/// <para> Selects all existing External Service Rules </para>
/// <para> Returns a list of lists, each sub list contains: int idExternalService, int idSystem </para>
/// </summary>
public static List<List<object>> _selectAll()


/// <summary>
/// <para> Selects an External Service Rule </para>
/// <para> Returns: A list that contains: int idExternalService, int idSystem </para>
/// </summary>
/// <param name="_idAllowedServices">Identifier of the Existing Rule</param>
public static List<List<object>> _selectWithService(int _idExternalService, bool _externalSystemID =
                                                    false)

/// <summary>
/// <para> Selects an External Service Rule </para>
/// <para> Returns: A list that contains: int idExternalService, int idSystem </para>
/// </summary>
/// <param name="_idAllowedServices">Identifier of the Existing Rule</param>
public static List<List<object>> _selectWithSystemAllowedRules(int _idSystem, bool _externalServiceID =
                                                               false)

/// <summary>
/// <para> Selects an External Service Rule </para>
/// <para> Returns: A list that contains: int idExternalService, int idCategory, string name, string
          description, bool status </para>
/// </summary>
/// <param name="_idAllowedServices">Identifier of the Existing Rule</param>
public static List<List<object>> _selectWithSystemExistingServices(int _idSystem)


/// <summary>
/// <para> Creates a new External Service Rule </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idExternalService"> Identifier of the existing service </param>
/// <param name="_idSystem"> Identifier of the existing system </param>
public static bool _insert(int _idExternalService, int _idSystem)


/// <summary>
/// <para> Creates a new External Service Rule </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_externalServiceName"> Name of the existing service </param>
/// <param name="_idSystem"> Identifier of the existing system </param>
public static bool _insert(string _externalServiceName, int _idSystem)


/// <summary>
/// <para> Deletes a specific entry </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name ="_idExternalService"> Identifier of the existing service </param>
/// <param name ="_idSystem"> Identifier of the existing system </param>
public static bool _delete(int _idExternalService, int _idSystem)


/// <summary>
/// <para> Deletes all entries associated with that service </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name ="_idExternalService"> Identifier of the existing service </param>
public static bool _deleteWithService(int _idExternalService)


/// <summary>
/// <para> Deletes all entries associated with that system </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name ="_idSystem"> Identifier of the existing system </param>
public static bool _deleteWithSystem(int _idSystem)
```

```
/// <summary>
/// <para> Deletes the existing entries in External Service Rule Entity </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
public static bool _deleteAll()


/// <summary>
/// <para> Checks if the external system is allowed to access to WiLOS services </para>
/// <para> Returns: 0 - Success </para>
/// <para> Returns: 1 - Service Offline </para>
/// <para> Returns: 2 - System No Authenticated </para>
/// <para> Returns: 3 - System Not Authorized </para>
/// </summary>
/// <param name="_systemName"> The name of the system to check on WiLOS database </param>
/// <param name="_password"> The password used by the system for authetification </param>
/// <param name="_serviceName"> The name of the service to be used </param>
public static int _checkSystemAuthentification(string _systemName, string _password, string _serviceName)
```

_System

```
/// <summary>
/// <para> Selects all External System </para>
/// <para> Returns: A list of lists, each sub list contains: int idSystem, string systemName, string
          password, string lastAccess, bool status </para>
/// </summary>
public static List<List<object>> _selectAll()


/// <summary>
/// <para> Selects External System according to the selection criteria </para>
/// <para> Returns: A list of lists, each sub list contains: int idSystem, string systemName, string
          password, string lastAccess, bool status </para>
/// </summary>
/// <param name="_start">The starting point of the selection</param>
/// <param name="_count">The number of rows to obtain</param>
/// <param name="_filter">The term of comparison in "systemName" field (it can be null)</param>
public static List<List<object>> _selectByPage(int _start, int _count, string _filter = null)


/// <summary>
/// <para> Returns the max number of rows that can be selected </para>
/// </summary>
/// <param name="_filter">The term of comparison in "systemName" field (it can be null)</param>
public static int _selectByPage_getRows(string _filter = null)


/// <summary>
/// <para> Selects an External System </para>
/// <para> Returns: A list containing: int idSystem, string systemName, string password, string
          lastAccess, bool status </para>
/// </summary>
/// <param name="_idSystem"> The identifier of the External System </param>
public static List<object> _select(int _idSystem)


/// <summary>
/// <para> Creates a new External System </para>
/// <para> The new entry identifier as an int or -1 if an error occurs </para>
/// </summary>
/// <param name="_idSystem"> Identifier of the External System to be create </param>
/// <param name="_systemName"> The name of the system to create </param>
/// <param name="_password"> The password used to authenticate the system </param>
/// <param name="_lastAccess"> The last time the system accessed HUEPS </param>
/// <param name="_status"> Used to turn on or off this system access </param>
public static int _insert(int _idSystem, string _systemName, string _password, string _lastAccess, bool
                          _status)
```

```
/// <summary>
/// <para> Creates a new External System </para>
/// <para> The new entry identifier as an int or -1 if an error occurs </para>
/// </summary>
/// <param name="_systemName"> The name of the system to create </param>
/// <param name="_password"> The password used to authenticate the system </param>
/// <param name="_lastAccess"> The last time the system accessed HUEPS </param>
/// <param name="_status"> Used to turn on or off this system access </param>
public static int _insert(string _systemName, string _password, string _lastAccess, bool _status)


/// <summary>
/// <para> Updates all fields of an External System </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idSystem"> Identifier of the External System to be updated </param>
/// <param name="_name"> The name of the system to create </param>
/// <param name="_password"> The password used to authenticate the system </param>
/// <param name="_lastAccess"> The last time the system accessed HUEPS </param>
/// <param name="_status"> Used to turn on or off this system access </param>
public static bool _update(int _idSystem, string _name, string _password, string _lastAccess, bool
                           _status)


/// <summary>
/// <para> Updates all fields of an External System using its Name as key </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_systemName"> The name of the system to update </param>
/// <param name="_password"> The password used to authenticate the system </param>
/// <param name="_lastAccess"> The last time the system accessed HUEPS </param>
/// <param name="_status"> Used to turn on or off this system access </param>
public static bool _update(string _systemName, string _password, string _lastAccess, bool _status)


/// <summary>
/// <para> Updates the 'name' field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idSystem"> Identifier of the External System </param>
/// <param name="_systemName"> The new 'systemName' value </param>
public static bool _updateSystemName(int _idSystem, string _systemName)


/// <summary>
/// <para> Updates the 'password' field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idSystem"> Identifier of the External System </param>
/// <param name="_password"> The new 'password' value </param>
public static bool _updatePassword(int _idSystem, string _password)


/// <summary>
/// <para> Updates the 'password' field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_systemName"> The name of the External System to update </param>
/// <param name="_password"> The new '_password' value </param>
public static bool _updatePassword(string _systemName, string _password)


/// <summary>
/// <para> Updates the 'lastAccess' field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idSystem"> Identifier of the External System to update </param>
/// <param name="_lastAccess"> The new 'lastAccess' value </param>
public static bool _updateLastAccess(int _idSystem, string _lastAccess)
```

```
/// <summary>
/// <para> Updates the 'lastAccess' field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_systemName"> The name of the External System to update </param>
/// <param name="_lastAccess"> The new 'lastAccess' value </param>
public static bool _updateLastAccess(string _systemName, string _lastAccess)


/// <summary>
/// <para> Updates the 'status' field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idSystem"> Identifier of the External System </param>
/// <param name="_status"> The new 'status' value </param>
public static bool _updateStatus(int _idSystem, bool _status)


/// <summary>
/// <para> Updates the 'status' field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_systemName"> Name of the External System to update </param>
/// <param name="_status"> The new 'status' value </param>
public static bool _updateStatus(string _systemName, bool _status)


/// <summary>
/// <para> Deletes an External System Entry </para>
/// <para> True or False, depending on the operation success </para>
/// </summary>
/// <param name ="_idSystem"> Identifier of the external system to be eliminated </param>
/// <param name="_all"> True - Removes All Info, False - Trys to remove only on External System Entity
    </param>
public static bool _delete(int _idSystem, bool _all = true)


/// <summary>
/// <para> Deletes an External System Entry </para>
/// <para> True or False, depending on the operation success </para>
/// </summary>
/// <param name ="_systemName"> Name of the external system to be eliminated </param>
/// <param name="_all"> True - Removes All Info, False - Trys to remove only on External System Entity
    </param>
public static bool _delete(string _systemName, bool _all = true)


/// <summary>
/// <para> Deletes All External System Entries </para>
/// <para> True or False, depending on the operation success </para>
/// </summary>
public static bool _deleteAll()
```

_ServiceCategory

```
/// <summary>
/// <para> Selects all External Service Category </para>
/// <para> Returns: A list of lists, each sub list contains: int idCategory, string name </para>
/// </summary>
public static List<List<object>> _selectAll()


/// <summary>
/// <para> Selects an External System </para>
/// <para> Returns: A list containing: int idCategory, string name </para>
/// </summary>
/// <param name="_idCategory"> The identifier of the External Service Category </param>
public static List<object> _select(int _idCategory)
```

```
/// <summary>
/// <para> Creates a new External System </para>
/// <para> The new entry identifier as an int or -1 if an error occurs </para>
/// </summary>
/// <param name="_idSystem"> Identifier of the External System to be create </param>
/// <param name="_name"> The name of the system to create </param>
/// <param name="_password"> The password used to authenticate the system </param>
/// <param name="_lastAccess"> The last time the system accessed HUEPS </param>
/// <param name="_status"> Used to turn on or off this system access </param>
public static int _insert(int _idCategory, string _name)


/// <summary>
/// <para> Creates a new External Service Category, but first confirms if the the category already
          exists </para>
/// <para> Returns: The new entry identifier as an int or the existing identifier or -1 if an error
          occurs </para>
/// </summary>
/// <param name="_name"> The name of the category to create </param>
public static int _insert(string _name)


/// <summary>
/// <para> Updates all fields of an External Service Category </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idCategory"> Identifier of the External Service Category to be updated </param>
/// <param name="_name"> The name of the category to create </param>
public static bool _update(int _idCategory, string _name)


/// <summary>
/// <para> Deletes an External Service Category Entry </para>
/// <para> True or False, depending on the operation success </para>
/// </summary>
/// <param name ="_idSystem"> Identifier of the external system to be eliminated </param>
public static bool _delete(int _idCategory)


/// <summary>
/// <para> Deletes All External Service Category Entries </para>
/// <para> True or False, depending on the operation success </para>
/// </summary>
public static bool _deleteAll()
```

_ExternalMessage

```
/// <summary>
/// <para> Selects all External Messages </para>
/// <para> Returns: A list of lists, each sub list contains: int idMessage, string serviceName, string
          from, string to, string date, string body </para>
/// </summary>
public static List<List<object>> _selectAll()


/// <summary>
/// <para> Selects an External Message </para>
/// <para> Returns a list that contains: int idMessage, string serviceName, string from, string to,
          string date, string body </para>
/// </summary>
/// <param name="_idMessage"> Identifier of the External Message to select </param>
public static List<object> _select(int _idMessage)


/// <summary>
/// <para> Gets a list of years available </para>
/// </summary>
public static List<object> _selectYears()
```

```
/// <summary>
/// <para> Gets a list of months available </para>
/// </summary>
/// <param name="_year"> The year to limit the search </param>
public static List<object> _selectMonths(string _year)



/// <summary>
/// <para> Gets a list of the days available on a certain month of the year </para>
/// </summary>
public static List<object> _selectDays(string _year, string _month)



/// <summary>
/// <para> Selects the information according to the given filters </para>
/// <para> Returns: A list of lists, each sub list contains: int idMessage, string serviceName, string
          from, string to, string date, string body </para>
/// </summary>
/// <param name="_year"> The year to be used as filter </param>
/// <param name="_month"> The month to be used as filter </param>
/// <param name="_day"> The day to be used as filter </param>
/// <param name="_startTime"> The start of the time interval to be used as filter </param>
/// <param name="_endTime"> The end of the time interval to be used as filter </param>
/// <param name="_serviceName"> The service name to be used as filter </param>
/// <param name="_page"> The page to get from the database </param>
/// <param name="_rowsToGet"> The number of items to retrieve from the database </param>
public static List<List<object>> _selectFilter(string _year, string _month, string _day, string
                          _startTime, string _endTime, string _serviceName, int _page, int _rowsToGet)



/// <summary>
/// <para> Gets the number of rows according to the given filters </para>
/// <para> Returns: An int value </para>
/// </summary>
/// <param name="_year"> The year to be used as filter </param>
/// <param name="_month"> The month to be used as filter </param>
/// <param name="_day"> The day to be used as filter </param>
/// <param name="_startTime"> The start of the time interval to be used as filter </param>
/// <param name="_endTime"> The end of the time interval to be used as filter </param>
/// <param name="_serviceName"> The service name to be used as filter </param>
public static int _getRows(string _year, string _month, string _day, string _startTime, string
                          _endTime, string _serviceName)



/// <summary>
/// <para> Creates a new External Message to be used </para>
/// <para> Returns: The new entry identifier as an int or -1 if an error occurs </para>
/// </summary>
/// <param name="_serviceName"> The name of the service consumed </param>
/// <param name="_from"> The system who used the service </param>
/// <param name="_to"> The system consulted </param>
/// <param name="_date"> The date when the service was consumed </param>
/// <param name="_body"> The content received </param>
public static int _insert(string _serviceName, string _from, string _to, string _date, string _body)



/// <summary>
/// <para> Creates a new External Message to be used </para>
/// <para> Returns: The new entry identifier as an int or -1 if an error occurs </para>
/// </summary>
/// <param name="_idMessage"> The identifier of the External Message to create </param>
/// <param name="_serviceName"> The name of the service consumed </param>
/// <param name="_from"> The system who used the service </param>
/// <param name="_to"> The system consulted </param>
/// <param name="_date"> The date when the service was consumed </param>
/// <param name="_body"> The content received </param>
public static int _insert(int _idMessage, string _serviceName, string _from, string _to, string _date,
                          string _body)
```

```
/// <summary>
/// <para> Updates all fields </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idMessage"> Identifier of the External Message to update </param>
/// <param name="_serviceName"> The new 'serviceName' value </param>
/// <param name="_from"> The new 'from' value </param>
/// <param name="_to"> The new 'to' value </param>
/// <param name="_date"> The new 'date' value </param>
/// <param name="_body"> The new 'body' value </param>
public static bool _update(int _idMessage, string _serviceName, string _from, string _to, string _date,
                           string _body)

/// <summary>
/// <para> Updates service name field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idMessage"> Identifier of the External Message to update </param>
/// <param name="_serviceName"> The new 'serviceName' value </param>
public static bool _updateServiceName(int _idMessage, string _serviceName)


/// <summary>
/// <para> Updates from field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idMessage"> Identifier of the External Message to update </param>
/// <param name="_from"> The new 'from' value </param>
public static bool _updateFrom(int _idMessage, string _from)


/// <summary>
/// <para> Updates 'to' field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idMessage"> Identifier of the External Message to update </param>
/// <param name="_to"> The new 'to' value </param>
public static bool _updateTo(int _idMessage, string _to)


/// <summary>
/// <para> Updates 'date' field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idMessage"> Identifier of the External Message to update </param>
/// <param name="_date"> The new 'date' value </param>
public static bool _updateDate(int _idMessage, string _date)


/// <summary>
/// <para> Updates 'body' field </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idMessage"> Identifier of the External Message to update </param>
/// <param name="_body"> The new 'body' value </param>
public static bool _updateBody(int _idMessage, string _body)


/// <summary>
/// <para> Deletes an External Message Entry </para>
/// <para> True or False, depending on the operation success </para>
/// </summary>
/// <param name ="_idMessage"> Identifier of the external message to be eliminated </param>
public static bool _delete(int _idMessage)


/// <summary>
/// <para> Deletes All External Message Entries </para>
/// <para> True or False, depending on the operation success </para>
/// </summary>
public static bool _deleteAll()
```

_HUEPSConfiguration

```
/// <summary>
/// <para> Selects all HUEPS Configurations </para>
/// <para> Returns: A list of lists, each sub list contains: int idHUEPSConfiguration, string name,
          bool password, string monitorStatus, float monitorSamplingTime </para>
/// </summary>
public static List<List<object>> _selectAll()

/// <summary>
/// <para> Selects all information of an HUEPS Configuration </para>
/// <para> Returns: A list: int idHUEPSConfiguration, string name, string password, bool monitorStatus,
          float monitorSamplingTime</para>
/// </summary>
/// <param name="_idHUEPSConfiguration"> Identifier of the HUEPS Configuration to select </param>
public static List<object> _select(int _idHUEPSConfiguration)

/// <summary>
/// <para> Creates a new HUEPS Configuration Entry </para>
/// <para> Returns: The new HUEPS Configuration identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_name"> The name of the HUEPS System </param>
/// <param name="_password"> The password used by HUEPS to authentificate on other systems  </param>
/// <param name="_monitorStatus"> The status of the HUEPS monitor service </param>
/// <param name="_monitorSamplingTime"> The sampling time used to acquire the house's energy information
                                        in miliseconds </param>
public static int _insert(string _name, string _password, bool _monitorStatus, float _monitorSamplingTime,
                          bool _externalServicesStatus)

/// <summary>
/// <para> Creates a new HUEPS Configuration Entry </para>
/// <para> Returns: The new HUEPS Configuration identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_idHUEPSConfiguration"> Identifier of the HUEPS Configuration to create</param>
/// <param name="_name"> The name of the HUEPS System </param>
/// <param name="_password"> The password used by HUEPS to authentificate on other systems  </param>
/// <param name="_monitorStatus"> The status of the HUEPS monitor service </param>
/// <param name="_monitorSamplingTime"> The sampling time used to acquire the house's energy information
                                        in miliseconds </param>
public static int _insert(int _idHUEPSConfiguration, string _name, string _password, bool _monitorStatus,
                          float _monitorSamplingTime, bool _externalServicesStatus)

/// <summary>
/// <para> Updates all fields of a specific HUEPS Configuration </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idHUEPSConfiguration"> Identifier of the HUEPS Configuration Entry to be updated
    </param>
/// <param name="_name"> The new 'name' value </param>
/// <param name="_password"> The new 'password' value </param>
/// <param name="_monitorStatus"> The new ' monitorStatus' value </param>
/// <param name="_monitorSamplingTime"> The new 'monitorSamplingTime' value (miliseconds) </param>
/// <param name="_externalServicesStatus"> The new 'External Services Status' value </param>
public static bool _update(int _idHUEPSConfiguration, string _name, string _password, bool _monitorStatus,
                           float _monitorSamplingTime, bool _externalServicesStatus)

/// <summary>
/// <para> Updates the field 'name' of an HUEPS Configuration Entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idHUEPSConfiguration"> Identifier of the HUEPS Configuration Entry to be updated</param>
/// <param name="_name"> The new 'name' value </param>
public static bool _updateName(int _idHUEPSConfiguration, string _name)

/// <summary>
/// <para> Updates the field 'password' of an HUEPS Configuration Entry </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHUEPSConfiguration"> Identifier of the HUEPS Configuration Entry to be updated
    </param>
/// <param name="_password"> The new 'password' value </param>
public static bool _updatePassword(int _idHUEPSConfiguration, string _password)
```

```
/// <summary>
/// <para> Updates the field 'monitor Status' of an HUEPS Configuration Entry </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHUEPSConfiguration"> Identifier of the HUEPS Configuration Entry to be updated
    </param>
/// <param name="_monitorStatus"> The new 'monitor Status' value </param>
public static bool _updateMonitorStatus(int _idHUEPSConfiguration, bool _monitorStatus)


/// <summary>
/// <para> Updates the field 'monitor Status' of all HUEPS Configuration Entries </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_monitorStatus"> The new 'monitor Status' value </param>
public static bool _updateMonitorStatus(bool _monitorStatus)


/// <summary>
/// <para> Updates the field 'Sampling Time' of HUEPS Monitor Service </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHUEPSConfiguration"> Identifier of the HUEPS Configuration entry to be updated
    </param>
/// <param name="_monitorSamplingTime"> The new 'max sampling rate' value (miliseconds) </param>
public static bool _updateMonitorSamplingTime(int _idHUEPSConfiguration, float _monitorSamplingTime)


/// <summary>
/// <para> Updates the field 'Sampling Time' of HUEPS Monitor Service (all entries) </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_monitorSamplingTime"> The new 'max sampling rate' value (miliseconds) </param>
public static bool _updateMonitorSamplingTime(float _monitorSamplingTime)


/// <summary>
/// <para> Updates the field 'External Services Status' of HUEPS </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idHUEPSConfiguration"> Identifier of the HUEPS Configuration entry to be updated
    </param>
/// <param name="_externalServicesStatus"> The new 'External Services Status' value </param>
public static bool _updateExternalServicesStatus(int _idHUEPSConfiguration, bool _externalServicesStatus)


/// <summary>
/// <para> Updates the field 'External Services Status' of HUEPS (all entries) </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_externalServicesStatus"> The new 'External Services Status' value </param>
public static bool _updateExternalServicesStatus(bool _externalServicesStatus)


/// <summary>
/// <para> Deletes an HUEPS Configuration Entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name ="_idSmartMeter"> Identifier of the HUEPS Configuration to be eliminated </param>
public static bool _delete(int _idHUEPSConfiguration)


/// <summary>
/// <para> Deletes the existing entries in HUEPS Configuration Entity </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
public static bool _deleteAll()
```

_SmartMeter

_Information

```
/// <summary>
/// <para> Selects all Smart Meters </para>
/// <para> Returns: A list of lists, each sub list contains: int idSmartMeter, string name, string
          manufacturer, string model, float maxSamplingRate </para>
/// </summary>
public static List<List<object>> _selectAll()


/// <summary>
/// <para> Selects all information of a Smart Meter </para>
/// <para> Returns: A list: int idSmartMeter, string name, string manufacturer, string model, float
          maxSamplingRate</para>
/// </summary>
/// <param name="_idSmartMeter">Identifier of the Smart Meter to select</param>
public static List<object> _select(int _idSmartMeter)


/// <summary>
/// <para> Creates a new Smart Meter Entry </para>
/// <para> Returns: The new smart meter identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_name">The name of the smart meter</param>
/// <param name="_manufacturer">The company who built the smart meter</param>
/// <param name="_model">The model version of the smart meter </param>
/// <param name="_maxSamplingRate">The maximum sample rate given by the smart meter </param>
public static int _insert(string _name, string _manufacturer, string _model, float _maxSamplingRate)


/// <summary>
/// <para> Creates a new Smart Meter Entry </para>
/// <para> Returns: The new smart meter identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_idSmartMeter">Identifier of the Smart Meter to create</param>
/// <param name="_name">The name of the smart meter</param>
/// <param name="_manufacturer">The company who built the smart meter</param>
/// <param name="_model">The model version of the smart meter </param>
/// <param name="_maxSamplingRate">The maximum sample rate given by the smart meter </param>
public static int _insert(int _idSmartMeter, string _name, string _manufacturer, string _model, float
                          _maxSamplingRate)

/// <summary>
/// <para> Updates all fields of a specific Smart Meter </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idSmartMeter">Identifier of the Smart Meter to be updated</param>
/// <param name="_name">The new 'name' value</param>
/// <param name="_manufacturer">The new 'manufacturer' value</param>
/// <param name="_model">The new ' model' value</param>
/// <param name="_maxSamplingRate">The new 'max sampling rate' value </param>
public static bool _update(int _idSmartMeter, string _name, string _manufacturer, string _model, float
                           _maxSamplingRate)

/// <summary>
/// <para> Updates the field 'name' of a Smart Meter </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idSmartMeter">Identifier of the Smart Meter to be updated</param>
/// <param name="_name">The new 'name' value </param>
public static bool _updateName(int _idSmartMeter, string _name)


/// <summary>
/// <para> Updates the field 'manufacturer' of a Smart Meter </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idSmartMeter">Identifier of the Smart Meter to be updated</param>
/// <param name="_manufacturer">The new 'manufacturer' value </param>
public static bool _updateManufacturer(int _idSmartMeter, string _manufacturer)
```

```
/// <summary>
/// <para> Updates the field 'model' of a Smart Meter </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idSmartMeter">Identifier of the Smart Meter to be updated</param>
/// <param name="_model">The new 'model' value </param>
public static bool _updateModel(int _idSmartMeter, string _model)


/// <summary>
/// <para> Updates the field 'maxSamplingRate' of a Smart Meter </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idSmartMeter">Identifier of the Smart Meter to be updated</param>
/// <param name="_maxSamplingRate">The new 'max sampling rate' value </param>
public static bool _updateMaxSamplingRate(int _idSmartMeter, float _maxSamplingRate)


/// <summary>
/// <para> Deletes a Smart Meter Entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name ="_idSmartMeter">Identifier of the Smart Meter to be eliminated</param>
public static bool _delete(int _idSmartMeter)


/// <summary>
/// <para> Deletes the existing entries in Smart Meter Entity </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
public static bool _deleteAll()
```

_Register

```
/// <summary>
/// <para> Selects all Smart Meter Registers </para>
/// <para> Returns: A list of lists, each sub list contains: int idRegister, int idSmartMeter, int
          idRegisterCategory, string description, string addressHexadecimal, bool write, string type,
          int lengthWords </para>
/// </summary>
public static List<List<object>> _selectAll()


/// <summary>
/// <para> Selects all Smart Meter Registers from a Smart Meter</para>
/// <para> Returns: A list of lists, each sub list contains: int idRegister, int idSmartMeter, int
          idRegisterCategory, string description, string addressHexadecimal, bool write, string type,
          int lengthWords </para>
/// </summary>
/// <param name="_idSmartMeter">The identifier of the smart meter</param>
public static List<List<object>> _selectAllWithSmartMeterID(int _idSmartMeter)


/// <summary>
/// <para> Selects all Smart Meter Registers from a Smart Meter Register Category</para>
/// <para> Returns: A list of lists, each sub list contains: int idRegister, int idSmartMeter, int
          idRegisterCategory, string description, string addressHexadecimal, bool write, string type,
          int lengthWords </para>
/// </summary>
/// <param name="_idRegisterCategory">The identifier of the smart meter register catagory</param>
public static List<List<object>> _selectAllWithSmartMeterRegisterCategoryID(int _idRegisterCategory)


/// <summary>
/// <para> Selects all information of a Smart Meter Register</para>
/// <para> Returns: A list: int idRegister, int idSmartMeter, int idRegisterCategory, string
          description, string addressHexadecimal, bool write, string type, int lengthWords</para>
/// </summary>
/// <param name="_idRegister">Identifier of the Smart Meter Regsiter to select</param>
public static List<object> _select(int _idRegister)
```

```
/// <summary>
/// <para> Creates a new Smart Meter Entry </para>
/// <para> Returns: The new smart meter register identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_idSmartMeter">Identifier of the Smart Meter of this register</param>
/// <param name="_idRegisterCategory">Identifier of the Category of this Smart Meter Register</param>
/// <param name="_description">The description of this register</param>
/// <param name="_addressHexadecimal">Register Address in Hexadecimal Format</param>
/// <param name="_write">Bool value representing if this is a write or a read register</param>
/// <param name="_type">The type of the value read from the register</param>
/// <param name="_lengthWords">The size of the returned value of the register in words</param>
public static int _insert(int _idSmartMeter, int _idRegisterCategory, string _description, string
                          _addressHexadecimal, bool _write, string _type, int _lengthWords)

/// <summary>
/// <para> Creates a new Smart Meter Register Entry </para>
/// <para> Returns: The new smart meter register identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_idRegister">Identifier of the Smart Meter Register to create</param>
/// <param name="_idSmartMeter">Identifier of the Smart Meter of this register</param>
/// <param name="_idRegisterCategory">Identifier of the Category of this Smart Meter Register</param>
/// <param name="_description">The description of this register</param>
/// <param name="_addressHexadecimal">Register Address in Hexadecimal Format</param>
/// <param name="_write">Bool value representing if this is a write or a read register</param>
/// <param name="_type">The type of the value read from the register</param>
/// <param name="_lengthWords">The size of the returned value of the register in words</param>
public static int _insert(int _idRegister, int _idSmartMeter, int _idRegisterCategory, string
                 _description, string _addressHexadecimal, bool _write, string _type, int _lengthWords)

/// <summary>
/// <para> Updates all fields of a specific Smart Meter Register</para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idRegister">Identifier of the Smart Meter Register to update</param>
/// <param name="_idSmartMeter">The new 'Identifier of the Smart Meter of this register' value</param>
/// <param name="_idRegisterCategory">The new 'Identifier of the Category of this Smart Meter Register'
                                     value</param>
/// <param name="_description">The new 'description of this register' value</param>
/// <param name="_addressHexadecimal">The new 'Register Address in Hexadecimal Format' value</param>
/// <param name="_write">The new bool value representing if this is a write or a read register</param>
/// <param name="_type">The new type of the value read from the register</param>
/// <param name="_lengthWords">The new size of the returned value of the register in words</param>
public static bool _update(int _idRegister, int _idSmartMeter, int _idRegisterCategory, string
                 _description, string _addressHexadecimal, bool _write, string _type, int _lengthWords)

/// <summary>
/// <para> Updates the field 'idSmartMeter'</para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idRegister">Identifier of the Smart Meter Register to be updated</param>
/// <param name="_idSmartMeter">The new 'Identifier of the Smart Meter of this register' value</param>
public static bool _updateSmartMeterID(int _idRegister, int _idSmartMeter)


/// <summary>
/// <para> Updates the field 'idRegisterCategory'</para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idRegister">Identifier of the Smart Meter Register to update</param>
/// <param name="_idRegisterCategory">The new 'Identifier of the Category of this Smart Meter Register'
                                     value</param>
public static bool _updateSmartMeterRegisterCategoryID(int _idRegister, int _idRegisterCategory)


/// <summary>
/// <para> Updates the field 'description'</para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idRegister">Identifier of the Smart Meter Register to update</param>
/// <param name="_description">The new 'description of this register' value</param>
public static bool _updateDescription(int _idRegister, string _description)
```

```
/// <summary>
/// <para> Updates the field 'addressHexadecimal'</para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idRegister">Identifier of the Smart Meter Register to update</param>
/// <param name="_addressHexadecimal">The new 'Register Address in Hexadecimal Format' value</param>
public static bool _updateAddressHexadecimal(int _idRegister, string _addressHexadecimal)

/// <summary>
/// <para> Updates the field 'write'</para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idRegister">Identifier of the Smart Meter Register to update</param>
/// <param name="_write">The new bool value representing if this is a write or a read register</param>
public static bool _updateWrite(int _idRegister, bool _write)

/// <summary>
/// <para> Updates the field 'type'</para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idRegister">Identifier of the Smart Meter Register to update</param>
/// <param name="_type">The new type of the value read from the register</param>
public static bool _updateType(int _idRegister, string _type)

/// <summary>
/// <para> Updates the field 'lengthWords'</para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idRegister">Identifier of the Smart Meter Register to update</param>
/// <param name="_lengthWords">The new size of the returned value of the register in words</param>
public static bool _updateLengthWords(int _idRegister, int _lengthWords)

/// <summary>
/// <para> Deletes a Smart Meter Register Entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name ="_idSmartMeter"> Identifier of the Smart Meter that owns this registers </param>
/// <param name="_idRegister"> Identifier of the Smart Meter Register to be eliminated </param>
public static bool _delete(int _idSmartMeter, int _idRegister)

/// <summary>
/// <para> Deletes the existing entries in Smart Meter Register Entity </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
public static bool _deleteAll()
```

_RegisterCategory

```
/// <summary>
/// <para> Selects all Smart Meter Registers Category</para>
/// <para> Returns: A list of lists, each sub list contains: int idRegisterCategory, string description
    </para>
/// </summary>
public static List<List<object>> _selectAll()

/// <summary>
/// <para> Selects all information of a Smart Meter Register Category</para>
/// <para> Returns: A list: int idRegisterCategory, string description</para>
/// </summary>
/// <param name="_idRegisterCategory">Identifier of the Smart Meter Regsiter Category to select</param>
public static List<object> _select(int _idRegisterCategory)

/// <summary>
/// <para> Creates a new Smart Meter Register Entry </para>
/// <para> Returns: The new smart meter register category identifier as an int, or -1 if an error
          occurs</para>
/// </summary>
/// <param name="_description">The description of this register</param>
public static int _insert(string _description)
```

```
/// <summary>
/// <para> Creates a new Smart Meter Register Entry </para>
/// <para> Returns: The new smart meter register category identifier as an int, or -1 if an error
          occurs</para>
/// </summary>
/// <param name="_idRegisterCategory">Identifier of the Category of this Smart Meter Register</param>
/// <param name="_description">The description of this register</param>
public static int _insert(int _idRegisterCategory, string _description)



/// <summary>
/// <para> Updates all fields of a specific Smart Meter Register Category</para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idRegisterCategory">The Identifier of the Smart Meter Register Category to be
                                    updated</param>
/// <param name="_description">The new 'description of this register category' value</param>
public static bool _update(int _idRegisterCategory, string _description)



/// <summary>
/// <para> Deletes a Smart Meter Register Category Entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name ="_idRegisterCategory">Identifier of the Smart Meter Register Category to be
                                     eliminated</param>
public static bool _delete(int _idRegisterCategory)



/// <summary>
/// <para> Deletes the existing entries in Smart Meter Register Category Entity </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
public static bool _deleteAll()
```

_Protocol

```
/// <summary>
/// <para> Selects all Smart Meter Protocols </para>
/// <para> Returns: A list of lists, each sub list contains: int _idProtocol, int _idSmartMeter, string
       comPort, int _unitID, int _baudRateBPS, int _dataBits, string _parity, string _stopBits</para>
/// </summary>
public static List<List<object>> _selectAll()



/// <summary>
/// <para> Selects all Smart Meter Protocols from a Smart Meter</para>
/// <para> Returns: A list of lists, each sub list contains: int _idProtocol, int _idSmartMeter, string
       comPort, int _unitID, int _baudRateBPS, int _dataBits, string _parity, string _stopBits</para>
/// </summary>
/// <param name="_idSmartMeter">The identifier of the smart meter</param>
public static List<List<object>> _selectAllWithSmartMeterID(int _idSmartMeter)



/// <summary>
/// <para> Selects all information of a Smart Meter Protocol</para>
/// <para> Returns: A list: int _idProtocol, int _idSmartMeter, string comPort, int _unitID, int
          _baudRateBPS, int _dataBits, string _parity, string _stopBits </para>
/// </summary>
/// <param name="_idProtocol">Identifier of the Smart Meter Protocol to select</param>
public static List<object> _select(int _idProtocol)
```

```
/// <summary>
/// <para> Creates a new Smart Meter Protocol </para>
/// <para> Returns: The new smart meter protocol identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_idSmartMeter">The identifier of the smart meter where this protocol is used</param>
/// <param name="_unitID">The identifier of the unit to communicate</param>
/// <param name="_baudRateBPS">The baudrate in bit per second value used during the
                              communication</param>
/// <param name="_dataBits">The number of bits use to transmit data</param>
/// <param name="_parity">Represents if parity is used, or not, in this protocol</param>
/// <param name="_stopBits">The number of stopbits used</param>
public static int _insert(int _idSmartMeter, string _comPort, int _unitID, int _baudRateBPS, int
                          _dataBits, string _parity, string _stopBits)

/// <summary>
/// <para> Creates a new Smart Meter Protocol Entry </para>
/// <para> Returns: The new smart meter protocol identifier as an int, or -1 if an error occurs</para>
/// </summary>
/// <param name="_idProtocol">Identifier of the Smart Meter Protocol to create</param>
/// <param name="_idSmartMeter">The identifier of the smart meter where this protocol is used</param>
/// <param name="_unitID">The identifier of the unit to communicate</param>
/// <param name="_baudRateBPS">The baudrate in bit per second value used during the
                              communication</param>
/// <param name="_dataBits">The number of bits use to transmit data</param>
/// <param name="_parity">Represents if parity is used, or not, in this protocol</param>
/// <param name="_stopBits">The number of stopbits used</param>
public static int _insert(int _idProtocol, int _idSmartMeter, string _comPort, int _unitID, int
                          _baudRateBPS, int _dataBits, string _parity, string _stopBits)

/// <summary>
/// <para> Updates all fields of a specific Smart Meter Protocol</para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idProtocol">Identifier of the Smart Meter Protocol to update</param>
/// <param name="_idSmartMeter">The new 'Smart Meter Identifier' value</param>
/// <param name="_unitID">The new 'unitID' value</param>
/// <param name="_baudRateBPS">The new 'baudRateBPS' value</param>
/// <param name="_dataBits">The new 'dataBits' value</param>
/// <param name="_parity">The new 'parity' value</param>
/// <param name="_stopBits">The new 'stopBits' value</param>
public static bool _update(int _idProtocol, int _idSmartMeter, string _comPort, int _unitID, int
                           _baudRateBPS, int _dataBits, string _parity, string _stopBits)

/// <summary>
/// <para> Updates the field 'idSmartMeter'</para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idProtocol">Identifier of the Smart Meter Protocol to be updated</param>
/// <param name="_idSmartMeter">The new 'Smart Meter Identifier' value</param>
public static bool _updateSmartMeterID(int _idProtocol, int _idSmartMeter)

/// <summary>
/// <para> Updates the field 'comPort' </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idProtocol"> Identifier of the Smart Meter Protocol to update </param>
/// <param name="_comPort"> The new 'unitID' value </param>
public static bool _updateComPort(int _idProtocol, string _comPort)

/// <summary>
/// <para> Updates the field 'unitID'</para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idProtocol">Identifier of the Smart Meter Protocol to update</param>
/// <param name="_unitID">The new 'unitID' value</param>
public static bool _updateUnitID(int _idProtocol, int _unitID)

/// <summary>
/// <para> Updates the field 'baudRateBPS'</para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idProtocol">Identifier of the Smart Meter Protocol to update</param>
/// <param name="_baudRateBPS">The new 'baudRateBPS' value</param>
public static bool _updateBaudRateBPS(int _idProtocol, int _baudRateBPS)
```

```
/// <summary>
/// <para> Updates the field 'dataBits'</para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idProtocol">Identifier of the Smart Meter Protocol to update</param>
/// <param name="_dataBits">The new 'dataBits' value</param>
public static bool _updateDatabits(int _idProtocol, int _dataBits)


/// <summary>
/// <para> Updates the field 'parity'</para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idProtocol">Identifier of the Smart Meter Protocol to update</param>
/// <param name="_parity">The new 'parity' value</param>
public static bool _updateParity(int _idProtocol, string _parity)


/// <summary>
/// <para> Updates the field 'stopBits'</para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idProtocol">Identifier of the Smart Meter Protocol to update</param>
/// <param name="_stopBits">The new 'stopBits' value</param>
public static bool _updateStopbits(int _idProtocol, string _stopBits)


/// <summary>
/// <para> Deletes a Smart Meter Protocol Entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name ="_idProtocol"> Identifier of the Smart Meter Protocol to be eliminated </param>
public static bool _delete(int _idProtocol)


/// <summary>
/// <para> Deletes the existing entries in Smart Meter Protocol Entity </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
public static bool _deleteAll()
```

Anexo M: Operações com a Base de Dados

Inserção de Dados

```
using MySql.Data.MySqlClient;

...

/// <summary>
/// <para> Creates a new Door </para>
/// <para> Returns: The new door identifier as an int, or -1 if an error occurs </para>
/// </summary>
/// <param name="_idDoor">The identifier of the door's house</param>
/// <param name="_idHouse">The identifier of the house where this door belongs</param>
/// <param name="_object">The string value representing the door object</param>
/// <returns>The new door identifier as an int, or -1 if an error occurs</returns>
public static int _insert(int _idDoor, int _idHouse, string _object)
{
   int _ans = -1;

   // Using the connection. Afterwards releases all resources allocated to the connection
   using (MySqlConnection _conn = new MySqlConnection(_databaseConnectionString))
   {
      using (MySqlCommand _sqlCommand = _conn.CreateCommand())
      {
         try
         {
            // Opens Connection
            _conn.Open();

            // Define query
            _sqlCommand.CommandType = System.Data.CommandType.Text;
            _sqlCommand.CommandText = "INSERT INTO houseDoor VALUES(@idDoor, @idHouse, @object);";

            // Set Parameters
            _sqlCommand.Parameters.AddWithValue("@idDoor", _idDoor);
            _sqlCommand.Parameters.AddWithValue("@idHouse", _idHouse);
            _sqlCommand.Parameters.AddWithValue("@object", _object);
            _sqlCommand.Prepare();

            // Execute query
            _sqlCommand.ExecuteNonQuery();
             if (_sqlCommand.LastInsertedId > 0)
                _ans = (int)_sqlCommand.LastInsertedId;
             else
                System.Diagnostics.Debug.WriteLine("Couldn't create a Door", "HUEPS Database
                   Error");

          }
          catch (Exception _e)
          {
             // An Error Occured
             System.Diagnostics.Debug.WriteLine("Couldn't create a Door: commandtext: " +
                _sqlCommand.CommandText + " exception: " + _e, "HUEPS Database Error");
          }
          finally
          {
             // Closes the connection
             _conn.Close();
          }
      }
   }

   return _ans;

   // End Function
}
```

Atualização de Dados

```
using MySql.Data.MySqlClient;

...

/// <summary>
/// <para> Updates the field 'object' of a House Division </para>
/// <para> Returns: True or False, depending on the operation success </para>
/// </summary>
/// <param name="_idDivision">Identifier of the Division to be updated</param>
/// <param name="_object">The new value representing the object that describes the division physical
                         format</param>
/// <returns>True or False, depending on the operation success</returns>
public static bool _updateObject(int _idDivision, string _object)
{
   bool _ans = false;

   // Using the connection. Afterwards releases all resources allocated to the connection
    using (MySqlConnection _conn = new MySqlConnection(_databaseConnectionString))
   {
      using (MySqlCommand _sqlCommand = _conn.CreateCommand())
      {
          try
          {
             // Opens Connection
             _conn.Open();

             // Define query
             _sqlCommand.CommandType = System.Data.CommandType.Text;
             _sqlCommand.CommandText = "UPDATE houseDivision SET object = @object WHERE idDivision =
                                       @idDivision;";

             // Set Parameters
             _sqlCommand.Parameters.AddWithValue("@idDivision", _idDivision);
             _sqlCommand.Parameters.AddWithValue("@object", _object);
             _sqlCommand.Prepare();

             // Execute query
             if (_sqlCommand.ExecuteNonQuery() > 0)
                _ans = true;
             else
                System.Diagnostics.Debug.WriteLine("Couldn't update field 'object' in House Division
                '" + _idDivision + "'", "HUEPS Database Error");

          }
          catch (Exception _e)
          {
             // An Error Occured
             System.Diagnostics.Debug.WriteLine("Couldn't update field 'object' in House Division '"
                + _idDivision + "': commandtext: " +_sqlCommand.CommandText + " exception: " + _e,
                "HUEPS Database Error");
          }
          finally
          {
             // Closes the connection
             _conn.Close();
          }
      }
   }

   return _ans;

   // End Function
}
```

Remoção de Dados

```
using MySql.Data.MySqlClient;

...

/// <summary>
/// <para> Deletes an house entry </para>
/// <para> Returns: True or False, depending on the operation success</para>
/// </summary>
/// <param name="_idHouse"> The identifier of the house to remove </param>
/// <returns> True or False, depending on the operation success </returns>
public static bool _delete(int _idHouse)
{
   bool _ans = false;

   // Using the connection. Afterwards releases all resources allocated to the connection
   using (MySqlConnection _conn = new MySqlConnection(_databaseConnectionString))
   {
      using (MySqlCommand _sqlCommand = _conn.CreateCommand())
      {
         try
         {
            // Opens Connection
            _conn.Open();

            // Define query
            _sqlCommand.CommandType = System.Data.CommandType.Text;
            _sqlCommand.CommandText = "DELETE FROM house WHERE idHouse = @idHouse";

            // Set Parameters
            _sqlCommand.Parameters.AddWithValue("@idHouse", _idHouse);
            _sqlCommand.Prepare();

            // Execute query
            if (_sqlCommand.ExecuteNonQuery() > 0)
               _ans = true;
            else
               System.Diagnostics.Debug.WriteLine("House '" + _idHouse + "' couldn't be deleted.",
                  "HUEPS Database Error");

         }
         catch (Exception _e)
         {
            // An Error Occured
            System.Diagnostics.Debug.WriteLine("House '" + _idHouse + "' couldn't be deleted:
            commandtext: " + _sqlCommand.CommandText + " exception: " + _e, "HUEPS Database Error");

         }
         finally
         {
             // Closes the connection
             _conn.Close();
         }
      }
   }

   return _ans;

   // End Function
}
```

Seleção de Dados

```
using MySql.Data.MySqlClient;

...

/// <summary>
/// <para> Selects all House Points from the House Point Entity </para>
/// <para> Returns: A list of lists, each sub list contains: int _idHousePoint, int
          _idHouseDivision, float _xAbs, float _yAbs, float _xRel, float _yRel, bool _acquired
    </para>
/// </summary>
/// <param name="_nameOrIdDivision">If true returns the division name, if false returns the division
                                    identifier</param>
/// <returns>A list of lists, each sub list contains: int _idHousePoint, int _idHouseDivision, float
             _xAbs, float _yAbs, float _xRel, float _yRel, bool _acquired</returns>
public List<List<object>> _selectAll(bool _nameOrIdDivision)
{
   List<List<object>> _ans = null;
   List<object> _aux = null;
   int _i = 0;

   // Using the connection. Afterwards releases all resources allocated to the connection
   using (MySqlConnection _conn = new MySqlConnection(_databaseConnectionString))
   {
      using (MySqlCommand _command = _conn.CreateCommand())
      {
         try
         {
            // Opens Connection
            _conn.Open();

            // Define query
            _command.CommandType = System.Data.CommandType.Text;
            if (_nameOrIdDivision)
               _command.CommandText = "SELECT h.idHousePoint, d.name, h.xAbs, h.yAbs, h.xRel,
                                       h.yRel, h.acquired FROM housePoint h, houseDivision d WHERE
                                       h.idHouseDivision = d.idHouseDivision;";
            else
               _command.CommandText = "SELECT * FROM housePoint;";

            // Read Data
            using (MySqlDataReader _data = _command.ExecuteReader())
            {
               if (_data.HasRows)
               {
                  _ans = new List<List<object>>();

                  while (_data.Read())
                  {
                     _aux = new List<object>();
                     for (_i = 0; _i < _data.FieldCount; _i++)
                     {
                        _aux.Add(_data[_i]);
                     }
                     _ans.Add(_aux);
                     _i = 0;
                  }
               }
            else
                System.Diagnostics.Debug.WriteLine("Select * From House Point: No data returned.",
                   "WiLOS Database Error");

            // Close data object
            if (!_data.IsClosed)
               _data.Close();

            }
         }
         catch (Exception _e)
         {
            // An Error Occured
            System.Diagnostics.Debug.WriteLine("Select * From House Point: commandtext: " +
```

```
                _command.CommandText + " exception: " + _e, "WiLOS Database Error");
            }
            finally
            {
               // Closes the connection
               _conn.Close();
            }
         }
      }

      return _ans;

      // End Function

   }
```

Consulta de Dados Estatísticos - HUEPS

```
   using MySql.Data.MySqlClient;

   ...

   /// <summary>
   /// <para> Gets the energy consumption of the the desired user arranged by year </para>
   /// <para> Returns: A List of Lists, each list containing: year, total active power [kW], total cost
              [€], CO2 Emission Ratio [kg] </para>
   /// </summary>
   /// <param name="_idDivision"> The identifier of the division whose energy profile is going to be
                                 selected </param>
   /// <returns> A List of Lists, each list containing: year, total active power [kW], total cost [€],
                CO2 Emission Ratio [kg] </returns>
   public static List<List<object>> _selectByAll(int _idDivision)
   {
      List<List<object>> _ans = null;
      List<object> _aux = null;
      int _i = 0;

      // Using the connection. Afterwards releases all resources allocated to the connection
      using (MySqlConnection _conn = new MySqlConnection(_databaseConnectionString))
      {
         using (MySqlCommand _sqlCommand = _conn.CreateCommand())
         {
            try
            {
                // Opens Connection
                _conn.Open();

                // Define query
                _sqlCommand.CommandType = System.Data.CommandType.Text;
                _sqlCommand.CommandText = @"SELECT b._year as _year,
                     SUM(b._activePower_kWh)* hc.monitorSamplingTime/(3600 * 1000) as
                         totalActivePower_kWs,
                     SUM(b._cost)* hc.monitorSamplingTime/(3600 * 1000) as totalCost_Euro,
                     SUM(b._CO2) *hc.monitorSamplingTime / (3600 * 1000) as CO2_Emissions_kg
                  FROM
                     (SELECT YEAR(ec.date) as _year,
                         e.energyConsumption * COUNT(pec.idEvent) / e.nUsers as _activePower_kWh,
                        (e.energyConsumption * COUNT(pec.idEvent) / e.nUsers) * ec.price as _cost,
                        (e.energyConsumption * COUNT(pec.idEvent) / e.nUsers) * ec.CO2 as _CO2
                      FROM energyConsumptionPartial pec, energyConsumptionEvent e, userLocation
                           ul, energyConsumption ec
                      WHERE ul.idDivision = @idDivision AND ul.idLocation = e.idLocation AND e.idEvent
                            = pec.idEvent AND pec.idEnergyConsumption = ec.idEnergyConsumption
                      GROUP BY e.idEvent, _year) b, huepsConfiguration hc
                  GROUP BY _year;";

                // Set Parameters
                _sqlCommand.Parameters.AddWithValue("@idDivision", _idDivision);
                _sqlCommand.Prepare();

                // Read Data
```

```
            using (MySqlDataReader _data = _sqlCommand.ExecuteReader())
            {
               if (_data.HasRows)
               {
                  _ans = new List<List<object>>();

                  while (_data.Read())
                  {
                     _aux = new List<object>();
                     for (_i = 0; _i < _data.FieldCount; _i++)
                     {
                        _aux.Add(_data[_i]);
                     }
                     _ans.Add(_aux);
                     _i = 0;
                  }
               }
               else
                  System.Diagnostics.Debug.WriteLine("Get Division " + _idDivision + " Energy
                     Profile by Years: No data returned.", "HUEPS Database Error");

               // Close data object
               if (!_data.IsClosed)
                  _data.Close();
               }

            }
            catch (Exception _e)
            {
               // An Error Occured
               System.Diagnostics.Debug.WriteLine("Get Division " + _idDivision + " Energy
                  Profile by Years: commandtext: " + _sqlCommand.CommandText + " exception: " + _e,
                  "HUEPS Database Error");
            }
            finally
            {
               // Closes the connection
               _conn.Close();
            }
         }
      }

      return _ans;

      // End Function
   }
```

Anexo N: Funcionalidade “_location_NN”

```
/// <summary>
/// <para> Does a simples search through all signalStrengthPoint in order to select the '_k' rssi
          coordinates with the mininum distance to the give rssi value </para>
/// <para> From those '_k' rssi values, only the one who occurs most times is returned (repetition
          by division) </para>
/// <para> Returns:  A list containing: idHouseDivision, xAbs, yAbs, SSP1, SelectionFactor1, SSP2,
          SelectionFactor2, ... , minSelectionFactor</para>
/// </summary>
/// <param name="_nAccessPoints"> The number of existing access points </param>
/// <param name="_receivedInfo"> A List of Lists of objects that contains: object1 {idAccessPoint1,
                                 rssi1}, object2 {idAccessPoint2, rssi2}, ... </param>
/// <param name="_k"> The number of neighbours to select </param>
/// <param name="_userID"> The identifier of the user </param>
public List<object> _location_NN(int _nAccessPoints, List<List<object>> _receivedInfo, int _k,
                                 int _userID)
{
   return _location_NN_MapOrientation(_nAccessPoints, _receivedInfo, _k, _userID, 2);

   // End Function
}


/// <summary>
/// <para> Does a simples search through all signalStrengthPoint in order to select the '_k' rssi
          coordinates with the mininum distance to the give rssi value</para>
/// <para> From those '_k' rssi values, only the one who occurs most times is returned (repetition
          by division)</para>
/// <para> Returns:  A list containing: int idHouseDivision, float xAbs, float yAbs, int idSSP1,
          float SelectionFactor1, int idSSP2, float SelectionFactor2, ... </para>
/// </summary>
/// <param name="_nAccessPoints">The number of existing access points</param>
/// <param name="_receivedInfo">A List of Lists of objects that contains: object1 {idAccessPoint1,
                               rssi1}, object2 {idAccessPoint2, rssi2}, ... </param>
/// <param name="_k">The number of neighbours to select</param>
/// <param name="_idMapOrientation"> 1 - Vertical Map, 2 - Horizontal Map</param>
/// <param name="_userID">The identifier of the user</param>
public List<object> _location_NN_MapOrientation(int _nAccessPoints, List<List<object>>
                                          _receivedInfo, int _k, int _userID, int _idMapOrientation)
{
   List<List<object>> _ans = null;
   List<object> _aux = null;
   int _i = 0;
   int _j = 0;
   int _n;
   bool _divisionExists = false;
   int _iString = 0;
   string _auxAP = String.Empty;
   List<List<object>> _divisionCountList = new List<List<object>>();
   List<object> _divisionCount;
   bool _error = false;

   try
   {
      #region Query Construction

      string _sqlQuery = "SELECT h.idHouseDivision, h.xAbs, h.yAbs";

      //Set Several AP rssi values return
      for (_i = 0; _i < _nAccessPoints; _i++)
      {
         _auxAP = _formatStringValue(_iString);
         _iString = _incrementValue(_iString);
         _sqlQuery += ", " + _auxAP + ".idSignalStrengthPoint, " + "ABS(" + _auxAP + ".rssiMean - "
                     + _receivedInfo[_i][1] + ")";
      }
      _i = 0;
      _iString = 0;

      string _auxAP1 = _formatStringValue(_iString);
      _iString = _incrementValue(_iString);
      _sqlQuery += " FROM signalStrengthPoint AS " + _auxAP1;
```

```
//Make the APs Join
for (_i = 1; _i < _nAccessPoints; _i++)
{
   _auxAP = _formatStringValue(_iString);
   _iString = _incrementValue(_iString);
   _sqlQuery += " JOIN signalStrengthPoint AS " + _auxAP + " ON (" + _auxAP1 + ".idHousePoint
                = " + _auxAP + ".idHousePoint AND " + _auxAP + ".idAccessPoint = " +
                _receivedInfo[_i][0] + ")";
}
_i = 0;
_iString = 0;

_sqlQuery += " JOIN housePoint AS h ON h.idHousePoint = " + _auxAP + ".idHousePoint";

_sqlQuery += " WHERE ((";

//North
for (_i = 0; _i < _nAccessPoints; _i++)
{
   _auxAP = _formatStringValue(_iString);
   _iString = _incrementValue(_iString);
   if (_i + 1 == _nAccessPoints)
      _sqlQuery += _auxAP + ".direction = '_north'";
   else
      _sqlQuery += _auxAP + ".direction = '_north' AND ";
}
_i = 0;
_iString = 0;

_sqlQuery += ") OR (";

//South
for (_i = 0; _i < _nAccessPoints; _i++)
{
   _auxAP = _formatStringValue(_iString);
   _iString = _incrementValue(_iString);
   if (_i + 1 == _nAccessPoints)
      _sqlQuery += _auxAP + ".direction = '_south'";
   else
      _sqlQuery += _auxAP + ".direction = '_south' AND ";
}
_i = 0;
_iString = 0;

_sqlQuery += ") OR (";

//East
for (_i = 0; _i < _nAccessPoints; _i++)
{
   _auxAP = _formatStringValue(_iString);
   _iString = _incrementValue(_iString);
   if (_i + 1 == _nAccessPoints)
      _sqlQuery += _auxAP + ".direction = '_east'";
   else
      _sqlQuery += _auxAP + ".direction = '_east' AND ";
}
_i = 0;
_iString = 0;

_sqlQuery += ") OR (";

//West
for (_i = 0; _i < _nAccessPoints; _i++)
{
   _auxAP = _formatStringValue(_iString);
   _iString = _incrementValue(_iString);
   if (_i + 1 == _nAccessPoints)
      _sqlQuery += _auxAP + ".direction = '_west'";
   else
      _sqlQuery += _auxAP + ".direction = '_west' AND ";
}
_i = 0;
_iString = 0;
```

```
// Set the last access point
_sqlQuery += ")) AND " + _auxAP1 + ".idAccessPoint = " + _receivedInfo[0][0];

// Set the map orientation
for (_i = 0; _i < _nAccessPoints; _i++)
{
   _auxAP = _formatStringValue(_iString);
   _iString = _incrementValue(_iString);
   _sqlQuery += " AND " + _auxAP + ".idMapOrientation = " + _idMapOrientation;
}
_i = 0;
_iString = 0;

_sqlQuery += " ORDER BY SQRT(";

for (_i = 0; _i < _nAccessPoints; _i++)
{
   _auxAP = _formatStringValue(_iString);
   _iString = _incrementValue(_iString);
   if (_i + 1 == _nAccessPoints)
      _sqlQuery += "POW(ABS(" + _auxAP + ".rssiMean - " + _receivedInfo[_i][1] + "),2)";
   else
      _sqlQuery += "POW(ABS(" + _auxAP + ".rssiMean - " + _receivedInfo[_i][1] + "),2) + ";
}
_i = 0;
_iString = 0;

_sqlQuery += ") ASC LIMIT 0, " + _k + ";";

#endregion

#region Query Execution

// Using the connection. Afterwards releases all resources allocated to the connection
using (MySqlConnection _conn = new MySqlConnection(_databaseConnectionString))
{
   using (MySqlCommand _command = _conn.CreateCommand())
   {
      try
      {
         // Opens Connection
         _conn.Open();

         // Define query
         _command.CommandType = System.Data.CommandType.Text;
         _command.CommandText = _sqlQuery;

         // Execute query
         using (MySqlDataReader _data = _command.ExecuteReader())
         {
            if (_data.HasRows)
            {
               _ans = new List<List<object>>();

               while (_data.Read())
               {
                  _aux = new List<object>();
                  for (_i = 0; _i < _data.FieldCount; _i++)
                  {
                     _aux.Add(_data[_i]);
                  }

                  //Store the divisions that repeat the most
                  _n = _divisionCountList.Count;
                  if (_n == 0)
                  {
                     _divisionCount = new List<object>();
                     _divisionCount.Add(_data[0]);        // Division ID
                     _divisionCount.Add(1);               // Counter
                     _divisionCount.Add(_ans.Count);      // Point position in _ans point list
                     _divisionCountList.Add(_divisionCount);
                  }
                  else
                  {
```

```
                        for (_j = 0; _j < _n; _j++)
                        {
                           _iString = (int)_data[0];
                           if ((int)_divisionCountList[_j][0] == _iString)
                           {
                              _divisionCountList[_j][1] = (int)_divisionCountList[_j][1] + 1;
                              _divisionExists = true;
                           }
                        }
                        _j = 0;

                        if (!_divisionExists)
                        {
                           _divisionCount = new List<object>();
                           _divisionCount.Add(_data[0]);
                           _divisionCount.Add(1);
                           _divisionCount.Add(_ans.Count);
                            _divisionCountList.Add(_divisionCount);
                        }
                        _divisionExists = false;
                     }

                     _ans.Add(_aux);
                     _i = 0;
                  }
                  _data.Close();
               }
               else
               {
                  System.Diagnostics.Debug.WriteLine("Select * From Simple Search:\nNo data
                     returned.", "WiLOS - Database Error");

                  // Error
                  _error = true;
               }
            }
         }
         catch (Exception _e)
         {
            // An Error Occured
            System.Diagnostics.Debug.WriteLine("External Message Entries couldn't be deleted:
               commandtext: " + _command.CommandText + " exception: " + _e, "WiLOS Database
               Error");
         }
         finally
         {
           // Closes the connection
            _conn.Close();
         }
      }
   }

   if (_error)
      return null;

   #endregion

   #region K Nearest Neighbour

   //Select the division that repeats the most and create user location
   int _divisionID = -1;
   int _max = -1;
   _n = _divisionCountList.Count;
   for (_i = 0; _i < _n; _i++)
   {
      if ((int)_divisionCountList[_i][1] > _max)
      {
         _max = (int)_divisionCountList[_i][1];
         _divisionID = (int)_divisionCountList[_i][0];
         _aux = _ans[(int)_divisionCountList[_i][2]];
      }
   }
   _i = 0;
```

```
            //Update User Position
            _UserPosition _userPosition = new _UserPosition();
            int _userPositionID = _userPosition._information._insert(_userID, _divisionID,
                                                                     DateTime.Now.ToString());

            //Store points used to get user position
            for (_i = 0; _i < _k; _i++)
            {
               for (_j = 0; _j < _nAccessPoints; _j++)
               {
                  //Insert each point
                  _userPosition._usedPoint._insert(_userPositionID, (int)_ans[_i][3 + _j * 2],
                                                   (double)_ans[_i][(3 + _j * 2) + 1]);
               }
            }

            return _aux;

            #endregion
         }
         catch (Exception _exc)
         {
            System.Diagnostics.Debug.WriteLine("Select * From Simple Search:\n" + _exc.ToString(), "WiLOS
               - Database Error");

            return null;
         }

         // End Function
      }
```

Anexo O: *Triggers* HUEPS

Obter Valores de CO2 e Custos

```
DELIMITER $$
USE `HUEPSDatabase`$$

CREATE TRIGGER GetCO2andCost_Trigger
    BEFORE INSERT ON EnergyConsumption
    FOR EACH ROW
    BEGIN

    DECLARE PRICE FLOAT;
    DECLARE CO2 FLOAT;

    SELECT energysupplierfaredetail.pricekw_h, energySupplier.CO2EmissionsRatio_tMWh INTO PRICE, CO2
    FROM energysupplierfaredetail, energySupplier
    WHERE (TIME(new.date) BETWEEN energysupplierfaredetail.from AND energysupplierfaredetail.to) AND
          energysupplierfaredetail.weekdays LIKE concat('%',date_format( NEW.date,'%a'),'%');
    SET NEW.price = PRICE, NEW.CO2 = CO2;

    END$$
```

Verificar Eventos Ativos

```
DELIMITER $$
USE `HUEPSDatabase`$$

CREATE TRIGGER CheckActiveEvents_Trigger
    AFTER INSERT ON EnergyConsumption
    FOR EACH ROW
    BEGIN
        #declare variable
        DECLARE e_idEvent INT;
        DECLARE e_idEnergyConsumption INT;
        DECLARE done INT DEFAULT FALSE;

        #declare cursor
        DECLARE cur1 CURSOR FOR SELECT idEvent
        FROM energyConsumptionEvent
        WHERE active = TRUE;

        #declare handle
        DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

        #gets inserted energetic consumption id
        SET e_idEnergyConsumption = NEW.idEnergyConsumption;

        #open cursor
        OPEN cur1;

        #starts the loop
        the_loop: LOOP
            #get the values of each column into our variables
            FETCH cur1 INTO e_idEvent;
            IF done THEN
              LEAVE the_loop;
            END IF;

            #Insert on Energy Consumption Partial
            INSERT INTO energyConsumptionPartial (idEvent, idEnergyConsumption)
            VALUES (e_idEvent, e_idEnergyConsumption);

        END LOOP the_loop;

        CLOSE cur1;

    END$$
```

Anexo P: Rotina “Atualização de Eventos devido à Movimentação de Utilizadores”, HUEPS

```
/// <summary>
/// <para>Function used to rearrange the active energetic events only using the users' locations </para>
/// </summary>
/// <param name="_usersCurrentLocations"> A list with the curent users' locations </param>
/// <param name="_usersPreviousLocations"> A list with the previous users' locations </param>
private void _noVariation(List<_WiLOS_HUEPS_Objects._UserLocation> _usersCurrentLocations,
                          List<_WiLOS_HUEPS_Objects._UserLocation> _usersPreviousLocations)
{
    #region Initializations

    // Lists for storing important info
     List<_eventClass> _events = null;                          // ID event, ID user location, id user,
                                                                   id division, power, factor, date
    List<_WiLOS_HUEPS_Objects._UserLocation> _users = null;
    List<_divisionClass> _divisionsProcessed = new List<_divisionClass>();
    WiLOSSimulatorExternalServices._WiLOS_Simulator_ExternalServices _WiLOS_Services = new
        WiLOSSimulatorExternalServices._WiLOS_Simulator_ExternalServices();

    //
    _WiLOS_HUEPS_Objects._UserLocation _usersCurrentLocation;
    _WiLOS_HUEPS_Objects._UserLocation _usersPreviousLocation;

    // Variables for sequence control
    int _i;
    int _n = _usersCurrentLocations.Count;
    int _usersOffline = 0;
    int _usersMoved = 0;
    _divisionClass _dsResult;
    int _idMultiEvent = -1;

    //
    _i = 0;
    foreach (_WiLOS_HUEPS_Objects._UserLocation _userLocation in _usersCurrentLocations)
    {
        // Count Users Offline
        if (_userLocation._idDivision == -1)
        {
            _HUEPS_Database._User._Credential._updateUserStatusWithUserID(_userLocation._idUser, false);
            _usersOffline++;
        }

        // Check if user entered -> online
        if (_usersPreviousLocations[_i]._idDivision == -1)
            _HUEPS_Database._User._Credential._updateUserStatusWithUserID(_userLocation._idUser, true);

        // Count Users Moving (includes offline users)
        if (_userLocation._movement || _userLocation._idDivision != _usersPreviousLocations[_i]._idDivision
            || _usersPreviousLocations[_i]._movement)
        {
            // Check movement, if idDivision the same, then we must get the previous division id
            if (_userLocation._idDivision == _usersPreviousLocations[_i]._idDivision)
            {
                object[] _WiLOSans=_WiLOS_Services._location_getUserPreviousDivisionID(_userLocation._idUser,
                                   _userLocation._idDivision);

                // Check if the operation was successful
                if (_WiLOSans != null)
                {
                    int _result = Convert.ToInt32(_WiLOSans[0]);
                    if (_result == 0)
                    {
                        if (_WiLOSans[1] != null)
                        {
                            // Last Division found
                            _usersPreviousLocations[_i]._idDivision = Convert.ToInt32(_WiLOSans[1]);
                        }
                        else
                            continue;   // No last division found, No movement
                    }
```

```
            }
        }

        // Forces movement
        _userLocation._movement = true;

        // Increments users
        _usersMoved++;

        // Check division with movement
        _dsResult = _divisionsProcessed.Find(delegate(_divisionClass _ds)
        {
            return (_ds._divisionID == _userLocation._idDivision);
        });

        if (_d<sResult == null && _userLocation._idDivision != -1)
        {
            _dsResult = new _divisionClass();
            _dsResult._divisionID = _userLocation._idDivision;
            _divisionsProcessed.Add(_dsResult);
        }
    }

    _i++;

    // End Loop
}

#endregion

 #region Energy Consumption Reorganization

 // Efetuar a restruturação dos consumos energéticos devido à simples movimentação dos
    utilizadores dentro da habitação

 // Por cada utilizador
 for (_i = 0; _i < _n; _i++)
 {
    // Get user
    _usersCurrentLocation = _usersCurrentLocations[_i];
    _usersPreviousLocation = _usersPreviousLocations[_i];

    #region Energy Consumption Reorganization -> previous divisions

    // Gets events from previous division
    _events = _HUEPS_Database._EnergyConsumptionInformation._Event._selectWithDivisionID(
              _usersPreviousLocation._idDivision, true);     // ID division, active
    if (_events != null)
    {
        // Para cada evento da divisão anterior
        foreach (_eventClass _event in _events)
        {
            // Evento é Meu?
            if (_event._userID == _usersCurrentLocation._idUser)
            {
                // Ainda estou na mesma divisão?
                if (_usersCurrentLocation._movement)
                {
                    _users = _getDivisionUsers(_usersCurrentLocations, _usersPreviousLocation._idDivision);

                    // Está lá alguém?
                    if (_users.Count > 0)
                    {
                        // Evento Multi-Utilizador?
                        if (_event._multiEventID > 1)
                        {
                            // Desativa evento multiutilizador (e eventos associados) e cria novos eventos
                               tendo em conta o novo fator com menos um utilizador

                            _HUEPS_Database._EnergyConsumptionInformation._Event._multiEventReallocationU
                                ser(_event._multiEventID, _event._userID, _event._factor);
                        }
                        else
                        {
```

```
                // Desativa evento
                _HUEPS_Database._EnergyConsumptionInformation._Event._updateActive(_event._event
                    ID, false);

                // Confirma o número de utilizadores para determinar se o evento é multiutilizador
                _idMultiEvent = 1;
                if (_users.Count > 1)
                    _idMultiEvent = HUEPS_Database._EnergyConsumptionInformation._MultiEvent.
                                    _insert();

                // Reparte consumo pelos utilizadores atuais dentro da divisão
                foreach (var _user in _users)
                {
                    _HUEPS_Database._EnergyConsumptionInformation._Event._insert(_user,
                        _idMultiEvent, _event._power, _users.Count, true);
                }
            }
        }
        else
        {
            // Ver se utilizador está ofline
            #region User Offline

            // Confirm if user is still inside the house
            if (_usersCurrentLocation._idDivision == -1)
            {
                // The user exited the network... equal energy consumption distribution for the
                    rest of the users

                // Obter Eventos do utilizador
                _events = _HUEPS_Database._EnergyConsumptionInformation._Event._selectWithUserID
                          (_usersPreviousLocation._idUser, true);     // ID division, active

                if (_events != null)
                {
                    foreach (_eventClass _eventUser in _events)
                    {
                        // Evento é Multiutilizador  ?
                        if (_eventUser._multiEventID == 1)
                        {
                            //Verificar o número total de utilizadores (alguém pode ter saído da rede)
                            int _usersCount = _usersCurrentLocations.Count - _usersOffline;

                            if (_usersCount == 0)
                            {
                                // No Users inside, this user continues to be responsible for the
                                    Consumption

                                _idMultiEvent = -1;
                            }
                            else if (_usersCount == 1)
                            {
                                // Only one valid user
                                _idMultiEvent = 1;
                            }
                            else
                            {
                                // Several users
                                _idMultiEvent = _HUEPS_Database._EnergyConsumptionInformation.
                                                _MultiEvent._insert();
                            }

                            if (_idMultiEvent != -1)
                            {
                                foreach (_WiLOS_HUEPS_Objects._UserLocation _userLocation in
                                    _usersCurrentLocations)
                                {
                                    if (_userLocation._idDivision != -1)
                                    {
                                        // Criar localização virtual
                                        int _idUserLocation = _HUEPS_Database._User._Location.
                                            _insert(_userLocation._idUser, _eventUser._divisionID,
                                            DateTime.Now.AddSeconds(-1).ToString("yyyy-MM-dd HH:mm:ss"),
                                            false, false);
```

```
                                    // Repartir evento
                                    _HUEPS_Database._EnergyConsumptionInformation._Event.
                                        _insert(_idUserLocation, _idMultiEvent, _eventUser._power,
                                        _usersCount, true);
                                }
                            }

                            // Desativa evento
                            _HUEPS_Database._EnergyConsumptionInformation._Event._updateActive
                                (_eventUser._eventID, false);

                        }
                    }
                    else
                    {
                        // Desativa evento multiutilizador (e eventos associados) e cria novos
                            eventos tendo em conta o novo fator com menos um utilizador

                        _HUEPS_Database._EnergyConsumptionInformation._Event._multiEventRealloc
                            ationUser(_eventUser._multiEventID, _eventUser._userID,
                            _eventUser._factor);
                    }

                }
            }

            break;
        }

        #endregion
    }
  }
}

#endregion

#region Energy Consumption Reorganization -> current divisions (movement triggered)

if (_usersCurrentLocation._movement || _usersPreviousLocation._movement)
{
    // Gets events from current division
    _events = _HUEPS_Database._EnergyConsumptionInformation._Event._selectWithDivisionID(
              _usersCurrentLocation._idDivision, true);     // ID division, active

    if (_events != null)
    {
        // Por cada evento da divisão atual
        foreach (_eventClass _event in _events)
        {
            // Evento Multiutilizador?
            if (_event._multiEventID > 1)
            {
                // Utilizador relacionado com o evento
                _Monitor._energyDistribution._eventClass _ess = _HUEPS_Database.
                    _EnergyConsumptionInformation._Event._selectMultiEvent(_event._multiEventID,
                    _usersCurrentLocation._idUser);
                if (_ess._eventID != -1)
                {
                    if (_ess._divisionID != _usersCurrentLocation._idDivision)
                    {
                        // Recriar todos os eventos mas alterando a divisão deste evento para a divisão
                            atual

                        _HUEPS_Database._EnergyConsumptionInformation._Event._multiEventReallocationDivi
                        sion(_ess._eventID, _ess._multiEventID, _ess._userLocationID);
                    }
                }
                else
                {
                    // Confirma o número de utilizadores para determinar se o evento é multiutilizador
                    _idMultiEvent = 1;
```

```
                _users = _getDivisionUsers(_usersCurrentLocations, _usersCurrentLocation.
                         _idDivision);

                if (_users.Count > 1)
                    _idMultiEvent = _HUEPS_Database._EnergyConsumptionInformation._MultiEvent.
                                    _insert();

                // Desativa o evento multiutilizador
                _HUEPS_Database._EnergyConsumptionInformation._Event._deactivateMultiUserEvent
                    (_ess._multiEventID);

                // Repartir pelos utilizadores que se encontram nesta divisão
                foreach (var _user in _users)
                {
                    _HUEPS_Database._EnergyConsumptionInformation._Event._insert(_user,
                        _idMultiEvent, _ess._power, _users.Count, true);
                }
            }
        }
        else
        {
            // Desativa evento
            _HUEPS_Database._EnergyConsumptionInformation._Event._updateActive(_event._eventID,
                false);

            // Confirma o número de utilizadores para determinar se o evento é multiutilizador
            _idMultiEvent = 1;
            _users = _getDivisionUsers(_usersCurrentLocations, _usersCurrentLocation._idDivision);
            if (_users.Count > 1)
                _idMultiEvent = _HUEPS_Database._EnergyConsumptionInformation._MultiEvent
                                ._insert();

            // Reparte consumo pelos utilizadores atuais dentro da divisão
            foreach (var _user in _users)
            {
                _HUEPS_Database._EnergyConsumptionInformation._Event._insert(_user,
                    _idMultiEvent, _event._power, _users.Count, true);
            }
        }
    }
}
else
{
    #region User Offline

    // Confirm if user is still inside the house
    if (_usersCurrentLocation._idDivision == -1)
    {
        // The user exited the network... equal energy consumption distribution for the rest of
        the users

        // Obter Eventos do utilizador
        _events = _HUEPS_Database._EnergyConsumptionInformation._Event._selectWithUserID(
                  _usersPreviousLocation._idUser, true);     // ID division, active

        if (_events != null)
        {
            foreach (_eventClass _eventUser in _events)
            {
                // Evento é Multiutilizador  ?
                if (_eventUser._multiEventID == 1)
                {
                    // Verificar o número total de utilizadores (alguém pode ter saído da rede)
                    int _usersCount = _usersCurrentLocations.Count - _usersOffline;
                    if (_usersCount == 0)
                    {
                        // No Users inside, this user continues to be responsible for the consumption
                        _idMultiEvent = -1;
                    }
                    else if (_usersCount == 1)
                    {
                        // Only one valid user
                        _idMultiEvent = 1;
                    }
```

```
                        else
                        {
                            // Several users
                            _idMultiEvent = _HUEPS_Database._EnergyConsumptionInformation._MultiEvent.
                                            _insert();
                        }

                        if (_idMultiEvent != -1)
                        {
                            foreach (_WiLOS_HUEPS_Objects._UserLocation _userLocation in
                                      _usersCurrentLocations)
                            {
                                if (_userLocation._idDivision != -1)
                                {
                                    // Criar localização virtual
                                    int _idUserLocation = _HUEPS_Database._User._Location._insert(
                                        _userLocation._idUser, _eventUser._divisionID,
                                        DateTime.Now.AddSeconds(-1).ToString("yyyy-MM-dd HH:mm:ss"), false,
                                        false);

                                    // Repartir evento
                                    _HUEPS_Database._EnergyConsumptionInformation._Event._insert(
                                    _idUserLocation, _idMultiEvent, _eventUser._power, _usersCount, true);
                                }
                            }

                            // Desativa evento
                            _HUEPS_Database._EnergyConsumptionInformation._Event._updateActive(
                                _eventUser._eventID, false);
                        }
                    }
                    else
                    {
                        // Desativa evento multiutilizador (e eventos associados) e cria novos eventos
                           tendo em conta o novo fator com menos um utilizador

                        _HUEPS_Database._EnergyConsumptionInformation._Event._multiEventReallocationU
                            ser(_eventUser._multiEventID, _eventUser._userID, _eventUser._factor);
                    }
                }
            }
        }

        #endregion

    }
}

#endregion

// End Function
}
```

Anexo Q: Exemplos XAML

_Interface.xaml, WiLOS

```
<Window x:Class="WiLOS._Interface"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:m="clr-namespace:WiLOS"
        xmlns:tb="http://www.hardcodet.net/taskbar"
        Title="WiLOS - Wi-Fi Location and Occupancy System" Name="windowWiLOS"
        Height="700" Width="1024" AllowsTransparency="True" WindowStyle="None" Background="Transparent"
        Closing="_windowClosing" SizeChanged="windowWiLOS_SizeChanged"
        Activated="windowWiLOS_Activated" ResizeMode="CanMinimize" Icon="_Images/Imagem2.ico"
        >
    <!-- Load Dictionaries -->
    <Window.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="/_Dictionaries/WiLOSstyles.xaml"/>
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </Window.Resources>
    <!-- Define WiLOS Window Animations -->
    <Window.Triggers>
        <EventTrigger RoutedEvent="Window.Loaded">
            <BeginStoryboard>
                <Storyboard>
                    <DoubleAnimation Storyboard.TargetName="windowWiLOS"
                                     Storyboard.TargetProperty="Opacity"
                                     From="0.0" To="1.0" Duration="0:0:0.5" />
                </Storyboard>
            </BeginStoryboard>
        </EventTrigger>
    </Window.Triggers>
    <!-- WiLOS Window, border for shadow effect-->
    <Border Name="_shadowBorder" Margin="10">
        <Border.Effect>
            <DropShadowEffect Color="Black" Direction="270" BlurRadius="10" ShadowDepth="3" />
        </Border.Effect>
        <!-- Main Grid -->
        <Grid Name="_gridMain" Background="{StaticResource backgroundColor}">
            <Grid.RowDefinitions>
                <RowDefinition Height="1.2*"/>
                <RowDefinition Height="0.4*"/>
                <RowDefinition Height="7*"/>
                <RowDefinition Height="0.9*"/>
            </Grid.RowDefinitions>
            <tb:TaskbarIcon x:Name="MyNotifyIcon" IconSource="/_Images/Error.ico"
                            Visibility="Collapsed" ToolTipText="Balloon Sample Icon" />
            <!-- Header -->
            <Grid Grid.Row="0" MouseLeftButtonDown="_gridHeader_MouseLeftButtonDown">
                <!-- WiLOS Title -->
                <TextBlock VerticalAlignment="Stretch" HorizontalAlignment="Stretch" Margin="0,5,0,0"
                           FontSize="56" Foreground="White" FontFamily="Futura Bk BT"
                           xml:space="preserve">
                           WiLOS
                </TextBlock>
                <TextBlock VerticalAlignment="Center" Margin="45,60,0,5" FontSize="12"
                           Foreground="White" FontFamily="Futura Bk BT">
                           Wi-Fi Location and Occupancy System
                </TextBlock>
                <!-- Window Buttons -->
                <Button Grid.Column="6" Name="_buttonClose" Width="25" Height="25" Margin="0,5,5,0"
                        Style="{StaticResource windowButtonStyle}" Click="_buttonClose_Click">
                    <Button.Content>
                        <Image Source="/_Images/_Icons/c.png" Stretch="Fill" />
                    </Button.Content>
                </Button>
                <Button Grid.Column="6" Name="_buttonMaximize" Width="25" Height="25" Margin="0,5,30,0"
                        Style="{StaticResource windowButtonStyle}" Click="_buttonMaximize_Click"
                        Visibility="Hidden">
                    <Button.Content>
```

```
            <Image Source="/_Images/_Icons/m.png" />
        </Button.Content>
    </Button>
    <Button Grid.Column="6" Name="_buttonMinimize" Width="25" Height="25"
            Margin="0,5,30,0" Style="{StaticResource windowButtonStyle}"
            Click="_buttonMinimize_Click">
        <Button.Content>
            <Image Source="/_Images/_Icons/m.png" Stretch="Fill" />
        </Button.Content>
    </Button>
</Grid>
<!-- Menu -->
<Grid Grid.Row="1" Name="_gridMenu" HorizontalAlignment="Stretch">
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="Auto" />
        <ColumnDefinition Width="*" />
    </Grid.ColumnDefinitions>
    <!-- Menu Buttons -->
    <Button Name="_buttonHouseTopology"  Grid.Column="0" Style="{StaticResource
            menuButtonStyle}" Click="_buttonHouseTopology_Click">
        <Button.Content>
            <TextBlock Style="{StaticResource menuTextBoxStyle}">
                House Topology
            </TextBlock>
        </Button.Content>
    </Button>
    <Button Name="_buttonSignalStrengthMap" Grid.Column="1" Style="{StaticResource
            menuButtonStyle}" BorderThickness="0" BorderBrush="Transparent"
            Click="_buttonSignalStrengthMap_Click">
        <Button.Content>
            <TextBlock Style="{StaticResource menuTextBoxStyle}">
                Signal Strength Map
            </TextBlock>
        </Button.Content>
    </Button>
    <Button Name="_buttonManageUsers" Grid.Column="2" Style="{StaticResource
            menuButtonStyle}" Click="_buttonManageUsers_Click">
        <Button.Content>
            <TextBlock Style="{StaticResource menuTextBoxStyle}">
                Manage Users
            </TextBlock>
        </Button.Content>
    </Button>
    <Button Name="_buttonMonitorService" Grid.Column="3" Style="{StaticResource
            menuButtonStyle}" Click="_buttonMonitorService_Click">
        <Button.Content>
            <TextBlock Style="{StaticResource menuTextBoxStyle}">
               Monitoring Service
            </TextBlock>
        </Button.Content>
    </Button>
    <Button Name="_buttonExternalServices" Grid.Column="4" Style="{StaticResource
            menuButtonStyle}" Click="_buttonExternalServices_Click">
        <Button.Content>
            <TextBlock Style="{StaticResource menuTextBoxStyle}">
                External Services
            </TextBlock>
        </Button.Content>
    </Button>
    <Button Name="_buttonChatService" Grid.Column="5" Style="{StaticResource
            menuButtonStyle}" Click="_buttonChatService_Click">
        <Button.Content>
            <TextBlock Style="{StaticResource menuTextBoxStyle}">
                Chat Service
            </TextBlock>
        </Button.Content>
    </Button>
    <Button Name="_buttonStatistics" Grid.Column="6" Style="{StaticResource
            menuButtonStyle}" Click="_buttonStatistics_Click">
```

```
                    <Button.Content>
                        <TextBlock Style="{StaticResource menuTextBoxStyle}">
                            Statistics
                        </TextBlock>
                    </Button.Content>
                </Button>
            </Grid>
            <!-- Content-->
            <Grid Grid.Row="2" Background="White">
                <m:MyGrid x:Name="_gridContent">
                </m:MyGrid>
            </Grid>
            <!-- Footer -->
            <Grid Grid.Row="3" Name="_gridFooter">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="2.7*" />
                    <ColumnDefinition Width="4*" />
                    <ColumnDefinition Width="2.7*" />
                </Grid.ColumnDefinitions>
                <!-- Login/Logout -->
                <Grid Grid.Column="0" Name="_gridLoginLogout">

                </Grid>
                <!-- WiLOS Current Module -->
                <Grid Grid.Column="1" Name="_gridWiLOSModule">
                    <TextBlock Name="_textblockWiLOSModule" Text="Home" VerticalAlignment="Center"
                               HorizontalAlignment="Center" FontFamily="Futura Md BT" FontSize="18"
                               Foreground="White"/>
                </Grid>
                <!-- Date and Time -->
                <Grid Grid.Column="2" Name="_gridDateTime">
                    <Grid.RowDefinitions>
                        <RowDefinition/>
                        <RowDefinition/>
                    </Grid.RowDefinitions>
                    <TextBlock VerticalAlignment="Bottom" HorizontalAlignment="Right"
                               Name="_textblockDate" Grid.Row="0" Style="{StaticResource
                               menuTextBoxStyle}" TextAlignment="Right" Margin="0,0,10,2" />
                    <TextBlock VerticalAlignment="Top" HorizontalAlignment="Right"
                               Name="_textblockTime" Grid.Row="1" Style="{StaticResource
                               menuTextBoxStyle}" TextAlignment="Right" Margin="0,2,10,0"/>
                </Grid>
            </Grid>
        </Grid>
    </Border>
</Window>
```

_HT_AddHouse3.xaml, WiLOS

```
<UserControl x:Class="WiLOS._HT_AddHouse3"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             xmlns:s="clr-namespace:WiLOS"
             mc:Ignorable="d" d:DesignWidth="1024" Height="495.982" Unloaded="UserControl_Unloaded">
    <UserControl.Resources>
        <ResourceDictionary>
            <ResourceDictionary.MergedDictionaries>
                <ResourceDictionary Source="/_Dictionaries/WiLOSstyles.xaml"/>
            </ResourceDictionary.MergedDictionaries>
        </ResourceDictionary>
    </UserControl.Resources>
    <!-- Main Grid  -->
    <Grid Background="White" Name="_gridHouseTopology" Loaded="_gridHouseTopology_Loaded"
          KeyDown="_gridHouseTopology_KeyDown">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="283*"/>
            <ColumnDefinition Width="16*" />
            <ColumnDefinition Width="725*"/>
        </Grid.ColumnDefinitions>
        <!-- Menu Grid -->
```

```
<Grid Grid.Column="0" Name="_gridHouseTopologyMenu">
    <Grid.RowDefinitions>
        <RowDefinition Height="0.3*" />
        <RowDefinition Height="0.3*" />
    </Grid.RowDefinitions>
    <!-- Menu Top -->
    <Grid Name="_gridMenuTop" Grid.Row="0" VerticalAlignment="center"
          HorizontalAlignment="Center" Width="260">
        <Grid.RowDefinitions>
            <RowDefinition />
            <RowDefinition Height="10"/>
            <RowDefinition />
            <RowDefinition Height="10"/>
            <RowDefinition />
            <RowDefinition Height="10"/>
            <RowDefinition />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="45" />
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <TextBlock Text="House Options" Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="2"
                   VerticalAlignment="Center" HorizontalAlignment="Left" Margin="20,0,0,0"
                   TextWrapping="Wrap" TextDecorations="Underline" FontFamily="Futura Bk BT"
                   FontSize="16" FontWeight="Bold"/>
        <Button Name="_buttonAddDivision" BorderThickness="0" Background="Transparent"
                Grid.Row="2" Grid.Column="1" VerticalAlignment="Center"
                HorizontalAlignment="Left" Click="_buttonAddDivision_Click">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto" />
                    <ColumnDefinition Width="Auto" />
                </Grid.ColumnDefinitions>
                <Image Grid.Column="0" Source="/_Images/_Icons/add_division.png" Height="25"
                       Width="35" Stretch="Uniform" HorizontalAlignment="Center"
                       VerticalAlignment="Center" />
                <TextBlock Margin="6,0,0,0" Grid.Column="1" HorizontalAlignment="Left"
                           VerticalAlignment="Center" FontFamily="Futura Bk BT"
                           FontSize="16">Add Division</TextBlock>
            </Grid>
        </Button>
        <Button Name="_buttonAddDoor" BorderThickness="0" Background="Transparent" Grid.Row="4"
                Grid.Column="1" VerticalAlignment="Center" HorizontalAlignment="Left"
                Click="_buttonAddDoor_Click">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto" />
                    <ColumnDefinition Width="Auto" />
                </Grid.ColumnDefinitions>
                <Image Grid.Column="0" Source="/_Images/_Icons/add_door.png" Height="25"
                       Width="35" Stretch="Uniform" HorizontalAlignment="Center"
                       VerticalAlignment="Center" />
                <TextBlock Margin="6,0,0,0" Grid.Column="1" HorizontalAlignment="Left"
                           VerticalAlignment="Center" FontFamily="Futura Bk BT"
                           FontSize="16">Add Door</TextBlock>
            </Grid>
        </Button>
        <Button Name="_buttonDefineScale" BorderThickness="0" Background="Transparent"
                Grid.Row="6" Grid.Column="1" VerticalAlignment="Center"
                HorizontalAlignment="Left" Click="_buttonDefineScale_Click">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto" />
                    <ColumnDefinition Width="Auto" />
                </Grid.ColumnDefinitions>
                <Image Grid.Column="0" Source="/_Images/_Icons/define_scale.png" Height="25"
                       Width="35" Stretch="Uniform" HorizontalAlignment="Center"
                       VerticalAlignment="Center" />
                <TextBlock Margin="6,0,0,0" Grid.Column="1" HorizontalAlignment="Left"
                           VerticalAlignment="Center" FontFamily="Futura Bk BT"
                           FontSize="16">Define Scale</TextBlock>
            </Grid>
        </Button>
    </Grid>
```

```
<!-- Menu Other Options -->
<Grid Name="_gridOtherOptions" Grid.Row="2" VerticalAlignment="center"
      HorizontalAlignment="Center" Width="260">
    <Grid.RowDefinitions>
        <RowDefinition />
        <RowDefinition Height="10"/>
        <RowDefinition />
        <RowDefinition Height="10"/>
        <RowDefinition />
        <RowDefinition Height="10"/>
        <RowDefinition />
        <RowDefinition Height="10"/>
        <RowDefinition />
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="45" />
        <ColumnDefinition />
    </Grid.ColumnDefinitions>
    <TextBlock Name="_textblockOtherOptions" Text="Other Options" Grid.Row="0"
               Grid.Column="0" Grid.ColumnSpan="2" VerticalAlignment="Center"
               HorizontalAlignment="Left" Margin="20,0,0,0"  TextWrapping="Wrap"
               FontFamily="Futura Bk BT" FontSize="16" TextDecorations="Underline"
               FontWeight="Bold"/>
    <Button Name="_buttonRemoveAll" BorderThickness="0" Background="Transparent"
            Grid.Row="2" Grid.Column="1" VerticalAlignment="Center"
            HorizontalAlignment="Left" Click="_buttonRemoveAll_Click">
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto" />
                <ColumnDefinition Width="Auto"  />
            </Grid.ColumnDefinitions>
            <Image Grid.Column="0" Source="/_Images/_Icons/remove_all.png" Height="25"
                   Width="35" Stretch="Uniform" HorizontalAlignment="Center"
                   VerticalAlignment="Center" />
            <TextBlock Margin="6,0,0,0" Grid.Column="1" HorizontalAlignment="Left"
                       VerticalAlignment="Center" FontFamily="Futura Bk BT"
                       FontSize="16">Delete All</TextBlock>
        </Grid>
    </Button>
    <Button Name="_buttonSaveDoorChanges" BorderThickness="0" Background="Transparent"
            Grid.Row="4" Grid.Column="1" VerticalAlignment="Center"
            HorizontalAlignment="Left" Click="_buttonSaveDoor_Click" >
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto" />
                <ColumnDefinition Width="Auto" />
            </Grid.ColumnDefinitions>
            <Image Grid.Column="0" Source="/_Images/_Icons/save_deep_sky_blue.png"
                   Height="25" Width="35" Stretch="Uniform" HorizontalAlignment="Center"
                   VerticalAlignment="Center" />
            <TextBlock Margin="6,0,0,0" Grid.Column="1" HorizontalAlignment="Left"
                       VerticalAlignment="Center" FontFamily="Futura Bk BT"
                       FontSize="16">Save Changes</TextBlock>
        </Grid>
    </Button>
    <Button Name="_buttonHouseSetupDone" BorderThickness="0" Background="Transparent"
            Grid.Row="6" Grid.Column="1" VerticalAlignment="Center"
            HorizontalAlignment="Left" Click="_buttonComputeHousePoints_Click">
        <Grid>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="Auto" />
                <ColumnDefinition Width="Auto" />
            </Grid.ColumnDefinitions>
            <Image Grid.Column="0" Source="/_Images/_Icons/show_house_points_crimson.png"
                   Height="25" Width="35" Stretch="Uniform" HorizontalAlignment="Center"
                   VerticalAlignment="Center" />
            <TextBlock Margin="6,0,0,0" Grid.Column="1" HorizontalAlignment="Left"
                       VerticalAlignment="Center" FontFamily="Futura Bk BT"
                       FontSize="16">Get Calibration Map</TextBlock>
        </Grid>
    </Button>
    <Button Name="_buttonNextStep" BorderThickness="0" Background="Transparent"
            Grid.Row="8" Grid.Column="1" VerticalAlignment="Center"
            HorizontalAlignment="Left" Click="_buttonNextStep_Click" >
```

```
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto" />
                    <ColumnDefinition Width="Auto" />
                </Grid.ColumnDefinitions>
                <Image Grid.Column="0" Source="/_Images/_Icons/finish_house_setup.png"
                       Height="25" Width="35" Stretch="Uniform" HorizontalAlignment="Center"
                       VerticalAlignment="Center" />
                <TextBlock Margin="6,0,0,0" Grid.Column="1" HorizontalAlignment="Left"
                           VerticalAlignment="Center" FontFamily="Futura Bk BT"
                           FontSize="16">Finish House Setup</TextBlock>
            </Grid>
        </Button>
    </Grid>
    <!-- Menu Door Selected -->
    <Grid Name="_gridDoorSelected" Grid.Row="2" VerticalAlignment="center"
          HorizontalAlignment="Center" Width="260">
        <Grid.RowDefinitions>
            <RowDefinition />
            <RowDefinition Height="10"/>
            <RowDefinition />
            <RowDefinition Height="10"/>
            <RowDefinition />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="45" />
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <TextBlock Name="_textblockDoorSelected" Text="Selected Door Options" Grid.Row="0"
                   Grid.Column="0" Grid.ColumnSpan="2" VerticalAlignment="Center"
                   HorizontalAlignment="Left" Margin="20,0,0,0"  TextWrapping="Wrap"
                   FontFamily="Futura Bk BT" FontSize="16" TextDecorations="Underline"
                   FontWeight="Bold"/>
        <Button Name="_buttonSaveDoor" BorderThickness="0" Background="Transparent"
                Grid.Row="2" Grid.Column="1" VerticalAlignment="Center"
                HorizontalAlignment="Left" Click="_buttonSaveDoor_Click" >
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto" />
                    <ColumnDefinition Width="Auto"  />
                </Grid.ColumnDefinitions>
                <Image Grid.Column="0" Source="/_Images/_Icons/save.png" Height="25" Width="35"
                       Stretch="Uniform" HorizontalAlignment="Center"
                       VerticalAlignment="Center" />
                <TextBlock Margin="6,0,0,0" Grid.Column="1" HorizontalAlignment="Left"
                           VerticalAlignment="Center" FontFamily="Futura Bk BT"
                           FontSize="16">Save Changes</TextBlock>
            </Grid>
        </Button>
        <Button Name="_buttonRemoveDoor" BorderThickness="0" Background="Transparent"
                Grid.Row="4" Grid.Column="1" VerticalAlignment="Center"
                HorizontalAlignment="Left" Click="_buttonRemoveDoor_Click">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto" />
                    <ColumnDefinition Width="Auto" />
                </Grid.ColumnDefinitions>
                <Image Grid.Column="0" Source="/_Images/_Icons/remove_door.png" Height="25"
                       Width="35" Stretch="Uniform" HorizontalAlignment="Center"
                       VerticalAlignment="Center" />
                <TextBlock Margin="6,0,0,0" Grid.Column="1" HorizontalAlignment="Left"
                           VerticalAlignment="Center" FontFamily="Futura Bk BT"
                           FontSize="16">Remove</TextBlock>
            </Grid>
        </Button>
    </Grid>
    <!-- Menu Division Selected -->
    <Grid Name="_gridDivisionSelected" Grid.Row="2" VerticalAlignment="center"
          HorizontalAlignment="Center" Width="260">
        <Grid.RowDefinitions>
            <RowDefinition />
            <RowDefinition Height="10"/>
            <RowDefinition />
            <RowDefinition Height="10"/>
```

```
            <RowDefinition />
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="45" />
            <ColumnDefinition />
        </Grid.ColumnDefinitions>
        <TextBlock Name="_textblockDivisionSelected" Text="Selected Division Options"
                   Grid.Row="0" Grid.Column="0" Grid.ColumnSpan="2" VerticalAlignment="Center"
                   HorizontalAlignment="Left" Margin="20,0,0,0"  TextWrapping="Wrap"
                   FontFamily="Futura Bk BT" FontSize="16" TextDecorations="Underline"
                   FontWeight="Bold"/>
        <Button Name="_buttonSaveDivision" BorderThickness="0" Background="Transparent"
                Grid.Row="2" Grid.Column="1" VerticalAlignment="Center"
                HorizontalAlignment="Left" Click="_buttonSaveDivision_Click" >
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto" />
                    <ColumnDefinition Width="Auto" />
                </Grid.ColumnDefinitions>
                <Image Grid.Column="0" Source="/_Images/_Icons/save.png" Height="25" Width="35"
                       Stretch="Uniform" HorizontalAlignment="Center"
                       VerticalAlignment="Center" />
                <TextBlock Margin="6,0,0,0" Grid.Column="1" HorizontalAlignment="Left"
                           VerticalAlignment="Center" FontFamily="Futura Bk BT"
                           FontSize="16">Save Changes</TextBlock>
            </Grid>
        </Button>
        <Button Name="_buttonRemoveDivision" BorderThickness="0" Background="Transparent"
                Grid.Row="4" Grid.Column="1" VerticalAlignment="Center"
                HorizontalAlignment="Left" Click="_buttonRemoveDivision_Click">
            <Grid>
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="Auto" />
                    <ColumnDefinition Width="Auto" />
                </Grid.ColumnDefinitions>
                <Image Grid.Column="0" Source="/_Images/_Icons/remove_division.png" Height="25"
                       Width="35" Stretch="Uniform" HorizontalAlignment="Center"
                       VerticalAlignment="Center" />
                <TextBlock Margin="6,0,0,0" Grid.Column="1" HorizontalAlignment="Left"
                           VerticalAlignment="Center" FontFamily="Futura Bk BT"
                           FontSize="16">Remove</TextBlock>
            </Grid>
        </Button>
    </Grid>
</Grid>
<!-- Content Grid -->
<Grid Grid.Column="1" Name="_gridHouseTopologyHome" Grid.ColumnSpan="2">
    <Grid.RowDefinitions>
        <RowDefinition Height="0.02*"/>
        <RowDefinition Height="0.78*"/>
        <RowDefinition Height="0.2*"/>
    </Grid.RowDefinitions>
    <Grid.ColumnDefinitions>
        <ColumnDefinition Width="0.2*"/>
        <ColumnDefinition Width="3.5*"/>
        <ColumnDefinition Width="0.2*"/>
    </Grid.ColumnDefinitions>

    <!-- First Row -->
    <Canvas Name="_canvasDivisions" Grid.Row="1" Grid.Column="1"
            MouseRightButtonDown="_canvasDivisions_MouseRightButtonDown"
            MouseLeftButtonDown="_start_Click" Opacity="0.6">
    </Canvas>
    <Canvas Name="_canvasDoors" Grid.Row="1" Grid.Column="1">
        <Canvas.Resources>
            <ControlTemplate x:Key="ResizeDecoratorTemplate" TargetType="{x:Type Control}">
                <Grid>
                    <s:ResizeThumb Cursor="SizeNS" VerticalAlignment="Top"
                                   HorizontalAlignment="Stretch" Opacity="0" />
                    <s:ResizeThumb Cursor="SizeWE" VerticalAlignment="Stretch"
                                   HorizontalAlignment="Left" Opacity="0" />
                    <s:ResizeThumb Cursor="SizeWE" VerticalAlignment="Stretch"
                                   HorizontalAlignment="Right" Opacity="0" />
                    <s:ResizeThumb Cursor="SizeNS" VerticalAlignment="Bottom"
```

```
                                HorizontalAlignment="Stretch" Opacity="0" />
                        <s:ResizeThumb Cursor="SizeNWSE" VerticalAlignment="Top"
                                HorizontalAlignment="Left" Opacity="0" />
                        <s:ResizeThumb Cursor="SizeNESW" VerticalAlignment="Top"
                                HorizontalAlignment="Right" Opacity="0" />
                        <s:ResizeThumb Cursor="SizeNESW" VerticalAlignment="Bottom"
                                HorizontalAlignment="Left" Opacity="0" />
                        <s:ResizeThumb Cursor="SizeNWSE" VerticalAlignment="Bottom"
                                HorizontalAlignment="Right" Opacity="0"/>
                    </Grid>
                </ControlTemplate>
                <ControlTemplate x:Key="MoveThumbTemplate" TargetType="{x:Type s:MoveThumb}">
                    <Rectangle Name="templateRectangle" Fill="Transparent"
                               MouseLeftButtonDown="_doorSelected_Click"/>
                </ControlTemplate>
                <ControlTemplate x:Key="DesignerItemTemplate" TargetType="ContentControl">
                    <Grid DataContext="{Binding RelativeSource={RelativeSource TemplatedParent}}">
                        <s:MoveThumb Name="moveT" Template="{StaticResource MoveThumbTemplate}"
                                     Cursor="SizeAll"/>
                        <Control x:Name="ResizeDecorator" Template="{StaticResource
                                 ResizeDecoratorTemplate}"/>
                        <ContentPresenter x:Name="oi" Content="{TemplateBinding
                                          ContentControl.Content}"/>
                    </Grid>
                </ControlTemplate>
            </Canvas.Resources>
        </Canvas>
        <Canvas Name="_canvasScale" Grid.Row="1" Grid.Column="1">
        </Canvas>
        <!-- Second Row -->
        <Grid Name="_gridDivisionInformation" Grid.Row="2" Grid.Column="1">
            <Grid.RowDefinitions>
                <RowDefinition Height="0.5*"/>
                <RowDefinition Height="0.7*"/>
                <RowDefinition Height="1*"/>
                <RowDefinition Height="1*"/>
                <RowDefinition Height="0.5*"/>
            </Grid.RowDefinitions>
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="0.15*"/>
                <ColumnDefinition Width="0.30*"/>
                <ColumnDefinition Width="0.15*"/>
                <ColumnDefinition Width="0.40*"/>
            </Grid.ColumnDefinitions>
            <TextBlock Name="_textblockDivInfo" Grid.Column="0" Grid.Row="1" Grid.ColumnSpan="2"
                       Text="Division Information"  Style="{StaticResource fontTextBlockStyle}"
                       HorizontalAlignment="Left" FontWeight="Bold" FontSize="15"/>
            <TextBlock Name="_textblockDivName" Grid.Column="0" Grid.Row="2" Text="Name"
                       Grid.RowSpan="2" Style="{StaticResource fontTextBlockStyle}"
                       HorizontalAlignment="Right" Margin="0,0,20,0" TextDecorations="Underline"/>
            <TextBox Name="_textboxName" Grid.Column="1" Grid.Row="2" Grid.RowSpan="2"
                     Text="Kitchen" Style="{StaticResource fontTextBoxStyle}" Margin="0,0,20,0"
                     TextAlignment="Center" VerticalContentAlignment="Center"
                     VerticalAlignment="Center" Height="25"></TextBox>
            <TextBlock Name="_textblockDescription" Grid.Column="2" Grid.Row="2" Text="Description"
                       Grid.RowSpan="2" Style="{StaticResource fontTextBlockStyle}"
                       HorizontalAlignment="Right" Margin="0,0,20,0" TextDecorations="Underline"/>
            <TextBox Name="_textboxDescription" Grid.Column="3" Grid.Row="1" Grid.RowSpan="3"
                     Text="Just a normal white house with yellow details in the suburbs wtih a
                     little garden and a swimming pool..." Style="{StaticResource
                     fontTextBoxStyle}" Margin="0,10,20,1" TextAlignment="Center"
                     TextWrapping="Wrap" VerticalContentAlignment="Center"
                     VerticalAlignment="Center" Height="62"/>
        </Grid>
    </Grid>
</Grid>
</UserControl>
```

Locating and monitoring tenants in PV based buildings

Duarte Arrobe, João Martins
Electrical Engineering Department
Universidade Nova de Lisboa
Faculty of Sciences and Technology
Monte da Caparica, Portugal
d.arrobe@campus.fct.unl.pt, jf.martins@fct.unl.pt

Celson Lima
Institute of Engineering and Geoscience
Computer Science program
Federal University of Western Pará
Brasil
celson.lima@ufopa.edu.br

***Abstract*—This paper describes a wireless location and occupancy system used to create house and user energetic profile systems. Both procedures will be used to raise consumers' energy awareness in PV efficient building systems. Home energy consumption must be improved in order to evolve towards Net Zero Energy Buildings. Only the combine use of home PV grid-connected power systems with the active and responsible citizen's engagement in addressing the carbon and energy reduction challenge will allow homes and buildings to become Net Zero Energy Buildings. The developed system is composed by two subsystems. The first one (designated as WiLOS) deals with the problem of users' location inside a building, while the second (designated as HUEPS) uses the energy consumption of the building and the location information to create the users' energetic profiles. Both subsystems are described and experimental data is presented in order to show the effectiveness of the proposed methodology.**

Keywords - PV Energy Efficient Buildings, Location Based Services, Wireless Location Systems

I. INTRODUCTION

With the constant increase of population and level of life around the world, it is clear that the demand for energy will follow the same path [1,2]. Nowadays the most versatile and cheapest way to produce energy is through fossil fuels [1,3]. However, the environmental impact is too great and the use of this kind of energy source prevents using Earth's resources in a sustainable way [3]. One way to overcome this problem is to switch to other "clean" energy sources, like solar or wind power [1].

Photovoltaic (PV) source is becoming one of the significant renewable energy sources in the world's energy portfolio, and it will have a major contribution to electricity generation. Among several advantages, we can consider pollution-free, low maintenance, high reliability, continuous cost reduction and fast technological progress [4]. Additionally, home PV grid-connected power systems are becoming a fast growing segment in several continents [5].

Unfortunately society cannot only depend on renewable energy sources to mitigate the energy consumption impact on nature. It is also necessary to take into consideration consumers' environmental awareness, thus resulting in an behavioural change regarding the use of energy [6-8]. I this context, technology is vital in order to reach an "ideal" solution. Technology evolution also reduces the energy demand by increasing user's end equipment efficiency [3,8]. This leads to the insurgence of energy efficient buildings which can contribute for the energy production has micro-suppliers or simply provide its users with the equipment, mechanisms and information that allow reducing energy consumption [6,9-11].

This article describes the concept and preliminary studies for raising consumers' energy awareness in PV efficient building systems. The proposed system seeks to engage people more actively and responsibly to address the energy and carbon reduction challenge. To achieve that, the system will monitor the building's energy consumption and its users, creating energetic profiles for the building, each division and its users. These profiles can be used by the users to determine in which areas they can intervene to improve energy efficiency, mainly by making them change their energy consumption behaviours or replacing outdated equipment. Eventually the information generated may be used by home automation systems to improve the building energy efficiency without direct users' action. Section II gives a brief description of the system concept and the drive behind its development. Section III describes implementation-related aspects of WiLOS. Section IV presents the WiLOS evaluation. Finally, section V concludes this preliminary study and refers the future work to be done.

II. SYSTEM CONCEPT AND MOTIVATION

The basic question driving the development of this work was: how to create energy profiles for several users moving around the building without having to make drastic changes on the building infrastructure? Looking to the modern world it is ease to realize that much has changed on the last decade, especially on the telecommunication world. Smartphones with greater processing capabilities and other interfaces allow users to surf the web, check emails or play games, just to name a few new possibilities. Besides 3G and 4G connectivity, 802.11 is commonly used by mobile devices (laptops and tablets

978-1-4799-1464-7/13/$31.00 ©2013 IEEE

included) to access the Internet or the internal company network. To support this trend, office and commercial buildings have expanded their network capabilities, especially using 802.11 access points to support communications with these devices. In Portugal (and likely in other parts of the globe), residential buildings also make part of this movement since the ISP providers supply an access point (AP) during the installation process as part of the service.

The mobile industry "boom" created a new business trend in the market that is growing each day, the Location Based Services [12]. In USA it started as an emergency mechanism to help locate people in case of 911 calls [13,14]. After that, other entities realized new applications for using the current user position. For instance, military strategies (both attack and defense), navigation and tracking systems, virtual tourist guides are some examples of location based services. The most common technology used for location is the Global Position System (GPS), which possesses great accuracy on outdoor environments but lacks reliability on dense urban zones and it hardly works on indoors environments [13,14].

In order to solve this problem many location systems were developed and studied using different technologies. Wi-Fi (802.11), Radio Frequency Identification (RFID), Ultrasound, Zig bee, Infra-Red (IF), (Very High Frequency) VHF, (Ultra High Frequency) UHF, Bluetooth, Ultra Wide Band (UWB), Vision Systems and Radio Interferometry (RI) are just some of the attempts to create a system capable of achieving a similar or better performance than GPS, considering indoor environments [15-17].

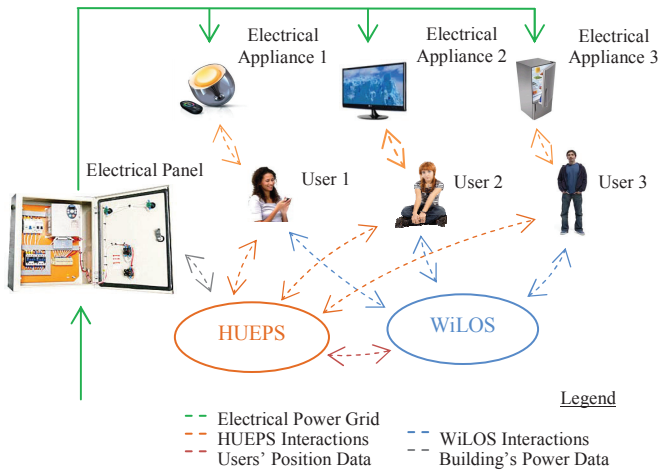
Using this trend of location based services, especially for indoor positioning, an opportunity for developing the proposed system occurred. Is it possible to use the users' locations inside the building to perform some correlation with the current consumed energy of the building? Using Wi-Fi technology for the location system it is possible to reuse the existing access point network and the users' smartphones, reducing the implementation and equipment costs. The energy consumption of the building can be easily obtained at the electrical panel using a smart meter or other kind of data acquisition device.

With this concept in mind, a pair of tools was conceived, namely WiLOS (Wireless Location and Occupancy System) and HUEPS (House and User Energetic Profile System). WiLOS deals with the problem of users' location inside a building, while HUEPS uses the energy consumption of the building and the location information from WiLOS to create energy profiles for building users. Fig. 1(a) illustrates the global system concept, where the users interact with the whole environment, (the building itself, WiLOS and HUEPS).

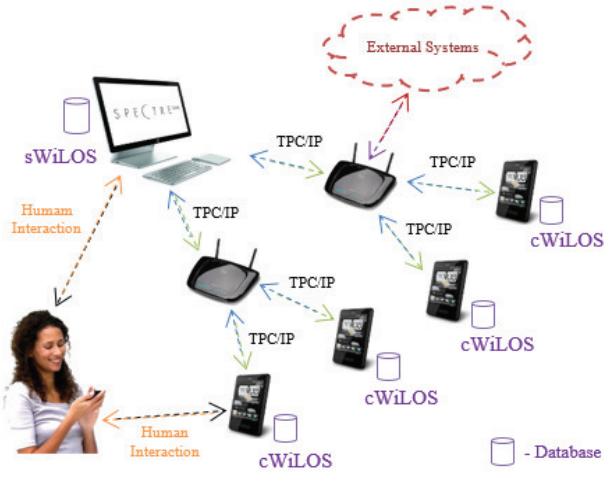
III. System Description

WiLOS provides the location of every single user inside the wireless network. WiLOS adopted client-server architecture. As such, sWiLOS, the server, acts as the system manger, coordinating and requesting information to determine mobile devices' location. cWiLOS, the client, is applied on the mobile device to access its 802.11 interface and respond sWiLOS pending requests. All the communication between sWiLOS and cWiLOS is supported by TCP/IP mechanisms. Database technology is used on both to store all the information exchanged and generated by WiLOS. Fig. 1(b) illustrates the implementation diagram of WiLOS and its actors.

WiLOS falls into the "fingerprint" location based systems, requiring a two-step process to locate the mobile devices within the network [14,16,17]. The first step is the offline calibration stage, where the signal strength of the APs is collected on several points of the building. I produces a signal strength map that is used to identify the different zones of the implementation site. This assumption is true if both number and distribution of existing APs inside the wireless infrastructure guarantee the identity of each zone. The calibration process occurs with the device (and the user)



(a)



(b)

Figure 1. (a): Global system concept. Definition of system elements' interactions and information's flow.
(b): WiLOS implementation diagram, existing actors and interactions between them.

oriented according to 4 directions: North, East, South and West. This additional information is used to decrease errors originated by the user's body when blocking the mobile device and the AP "line of sight" [16].

On step two, sWiLOS requests cWiLOS the current signal strength values detected by the mobile device. These values are compared to those previously acquired in the map in order to find the best match. This phase is also called monitoring phase, where WiLOS monitors and locates the users in real time. The algorithm used by WiLOS to find the best match between the current read value and the signal strength map values stored on the database is the k-Nearest Neighbour Search (k-NNS) with Euclidean Distance.

The approach adopted here (i.e., a fingerprint location system with 4 orientation maps and k-NNS) has been used before. "RADAR" [16] and works [17,18] are just few examples of systems that adopted same approach. However, those systems had the objective to determine the "point to point" precision and analyse the users tracking at that level. WiLOS main goal is to locate users within a division, therefore it does not require the same level of precision. With this work, WiLOS performance will be tested to confirm if this approach is enough to provide "division to division" precision and support HUEPS requirements.

IV. Experimental Tests/Trials

WiLOS simulation scenario was implemented at the 1st floor of the Electrical Engineering Department, Faculty of Sciences and Technology from Universidade Nova de Lisboa. The whole area used (133m^2) allowed simulating 5 divisions. An initial study was required to check the APs' network coverage and quality of service. The APs' distribution, the simulation scenario, and the calibration points used to get the signal strength maps for WiLOS are represented in Fig. 2(a). In order to acquire the calibration map, 10 signal strength values were retrieved from each direction at each point and from the 3 APs. A total of 672 calibration points defines the signal strength map for the simulation scenario.

In order to test the real time performance of WiLOS, the route on Fig. 2(b) was designed. The user walks through the route and stops at every checkpoint for 10 seconds. Using a 2 seconds per sample sampling rate, this assures 5 signal strength values per checkpoint. Data acquisition during users' transition among checkpoints is also used to compute WiLOS precision. The route is repeated 5 times per trial to guarantee that results obtained are consistent and not a result of mere chance. On each trial the 'k' value of the k-NNS was changed in order to check the algorithm performance with different numbers of neighbors during the matching process. A total of 7478 points resulted from the trials. Result is presented on Table I and illustrated on Fig. 3.

The k-NNS always implies some kind of commitment. When the number of neighbors is too high more computation is required and errors related with "wrong neighbors" start to occur, decreasing the system performance. On the other hand, too few neighbors may not be enough to determine the correct position. According to Table I, a slightest better performance is achieved when the 'k' value is 3 (91.01%). However, similar performance can be reached with 4, 5 and 6 neighbors (≈ 90%). Precision reduction occurs with 1, 2 and bigger 'k' values, such as 9 (≈ 87%). The data collected reinforces the k-NNS commitment between number or neighbors and accuracy. Results achieved here are similar to those found in [16,19].

Few trials showed a quite high standard deviation (≈ 4%). The trials were done in real time, with the same conditions and same user orientation at each checkpoint. The mobile device was held by the user at a horizontal position, the same position used to calibrate WiLOS. These options were made to minimize external factors that could compromise the analysis of the results. Therefore, the elements causing this deviation must coexist with WiLOS in the same environment. This error might be justified with the APs emitted signal strength variation. The signal sent suffers reflection, diffraction and propagation loss, depending on the buildings infrastructure, introducing on WiLOS a random error [14,18]. Tests showed that

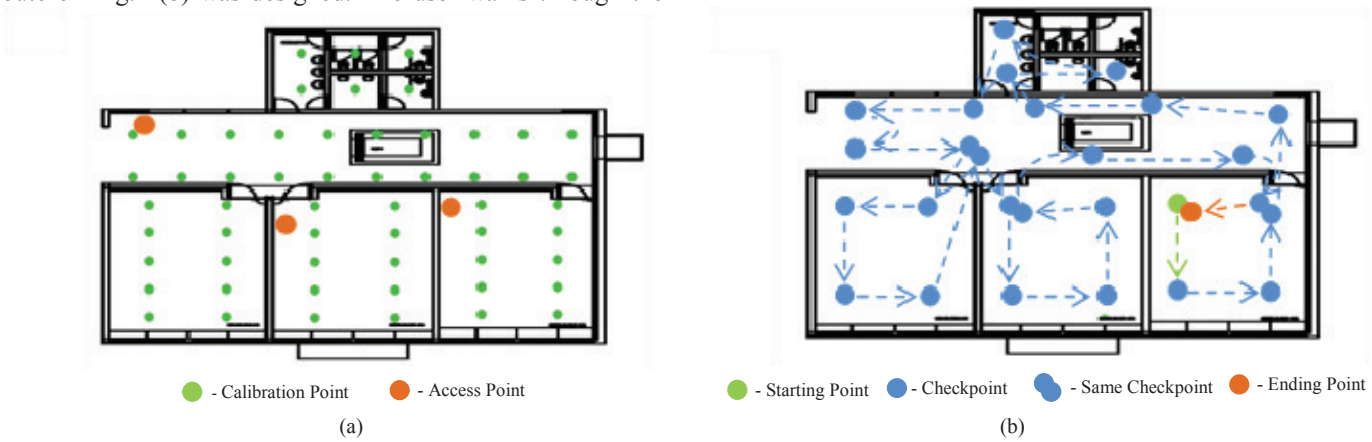


Figure 2. (a): WiLOS implementation scenario and defined calibration points. (b): WiLOS experimental test course.

TABLE I. WILOS PRECISION ON TESTING SCENARIO

NNS 'k' Value	WiLOS Precision by Division		
	Collected Points	*Average (%)*	*Standard Deviation (%)*
1	1116	86.61	3.01
2	1073	87.65	2.74
3	1060	91.01	4.24
4	1073	89.76	3.93
5	1062	89.70	2.11
6	1111	90.41	4.03
9	983	87.07	1.89

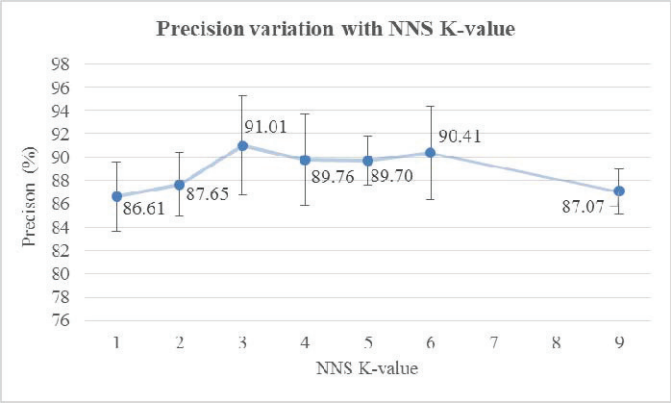


Figure 3. WiLOS precision variation with k-NNS 'k' value. Standard Deviation as measurement error.

most cases results were very good but not always

Other reason for this deviation is linked to the natural moving of users in the experimental environment. Human body is composed by water, among other things, which attenuates the signal during its propagation. This effect is stronger when the person crosses the mobile device and the AP's line of sight [18]. Unfortunately, WiLOS is not the only wireless system on the building. Other networks with APs, including other devices that share the same bandwidth as the 802.11 standard (e.g., wireless keyboards and Bluetooth devices) can cause fluctuations and interferences on WiLOS, which also reinforces the performance deviation noticed on the trials.

V. CONCLUSIONS AND FUTURE WORK

Location based services have numerous applications, either on outdoor environment as in indoor situations. For them to work, the location system must provide the users' location with a certain level of precision, depending on the application. On this case is desired a room precision in order to detect the user inside the house. Perfect tracking is not required feature. This preliminary study shows that WiLOS has potential to be the location system supporting HUEPS. Using only a simple search algorithm (k-NNS) a level of precision near 90% was achieved. The 'k' value analysis allows to set the number of neighbours to 3 or 4, granting good results without extra computation costs.

To deploy WiLOS, an initial study for APs' distribution was performed. This study was inspired on works [12,20-22] which referred APs location to be crucial for location systems. Normally the APs are disposed only to achieve only maximum coverage and service quality [12]. The APs distribution in Fig. 2(a) allows to maintain the mentioned network characteristics while taking location based services in account. An interesting approach would be to confirm WiLOS performance on different scenarios with 2, 3 or 4 APs with different distributions, and changing the 'k' value.

Still related with the APs' distribution, some studies also deal with the "bad APs neighbors" problem. In other words, using scenarios with lots of APs, some of them do not add new location information and can even introduce error in the system. In these cases, some mechanisms must be implemented to determine which AP is "good" or "bad" neighbor [23]. Therefore, APs' distribution is a vital part of the system and the development of an algorithm capable of best APs' distribution determination must be considered, without compromising the network coverage and quality of service.

Other algorithms or additional information will be introduced to evaluate if WiLOS performance will improve, without affecting the system computation and latency times. For instance, WiLOS was calibrated with the mobile device held at a horizontal position. The trials were also done following the same path. How would WiLOS perform if the mobile device orientation were not optimal? In other words, what if the device were inside the user's pocket (vertical orientation)? This is a quite common situation because normally the user carries the device inside his/her pocket, only taking it out during its usage, like texting or calling. If the device orientation can reduce the system reliability, this problem must also be taken into account on future WiLOS work. HUEPS development starts after completion of WiLOS assessment with the required quality and precision.

REFERENCES

[1] Morrison, A.J., "Global Demand Projections for Renewable Energy Resources," *Electrical Power Conference, 2007. EPC 2007. IEEE Canada* , Pages 537-542, October 2007, in press.

[2] Mattick, C.S.; Williams, E.; Allenby, B.R., "Historical Trends in Global Energy Consumption," *Technology and Society Magazine, IEEE*, vol.29, no.3, Pages 22-30, Fall 2010, in press.

[3] Bose, B.K., "Global Warming: Energy, Environmental Pollution, and the Impact of Power Electronics," *Industrial Electronics Magazine, IEEE*, vol.4, no.1, Pages 6-17, March 2010, in press.

[4] J. Carrasco, L. Franquelo, et. al., "Power-Electronic Systems for the Grid Integration of Renewable Energy Sources: A Survey," *IEEE Trans. on Ind. Electron*., vol. 53, no. 4, pp. 1002–1016, Aug. 2006, in press.

[5] J. Selvaraj and N. A. Rahim, "Multilevel inverter for grid-connected PV system employing digital PI controller*," IEEE Trans. Ind. Electron*., vol. 56, no. 1, pp. 149–158, January 2009, in press.

[6] Alahmad, M.; Zulfiqar, M.F.; Hasna, H.; Sharif, H.; Sordiashie, E.; Aljuhaishi, N.A., "Green and Sustainable Technologies for the Built Environment" *Developments in E-systems Engineering (DeSE), 2011* , Pages 521-526, December 2011, in press.

[7] Stragier, J. ; Hauttekeete, L. ; De Marez, L. ; Derboven, J. ;Laporte, L., "Household Energy Use and Creating Awareness: Opportunities for ICT", *Emerging Intelligent Data and Web Technologies (EIDWT), 2012 Third International Conference on*, Pages 276-280, September 2012, in press.

[8] Siweya, N.A., "Innovative energy efficiency communication and awareness campaign," *Industrial and Commercial Use of Energy (ICUE), 2011 Proceedings of the 8th Conference on the*, Pages 181-183, August 2011, in press.

[9] Spagnolli A.; Corradi N.; Gamberini L.; Hoggan E.; Jacucci G.; Katzeff C.; et al., "Eco-Feedback on the Go: Motivating Energy Awareness", *Computer (Volume 44, Issue 5),* Pages 38-45, May 2011, in press.

[10] GALP Energia, "Projeto Smart GALP", *Sustentabilidade e Eficiência Energética*, Portugal, 2012. Available at: http://www.mysmartgalp.com.

[11] Energias de Porugal EDP Distribuição, "InnovGrid, Next Generation Grid", *Seminar about a Inovação para a Competividade em Serviços de água*, Porto, Portugal, 9 February 2012.

[12] Beale, R., "Improving the accuracy and reliability of wireless location Systems: a case study", *7th Mexican International Conference on Artificial Intelligence (MICAI)*, Pages 383-387, October 2008, in press.

[13] Fuller, R; Christie, J; Nichols, J.; et al, "A Highly Flexible and Scalable System for Location Determination of Wireless Devices*", IEEE Position and Location National Symposium*, Pages 233-239, 2002, in press.

[14] Joy, V.; Laxman, P.V., "Smart Spaces; Indoor Wireless Location Management System", *The 2007 International Conference o Next Generation Mobile Applications, Services and Techonologies*, Pages 261-266, September 2007, in press.

[15] Shang, J.; Yu, S.; Zhu, L.; "Location-Aware Systems for Short-range Wireless Networks", *Computer Network and Multimedia Technology, 2009. CNMT 2009. International Symposium on*, Pages 1-5, January 2009, in press.

[16] Bahl P.; Padmanabhan, V. N., "RADAR: An In-Building RF-based User Location and Tracking System", *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communication Societies. Proceedings. IEEE*, Pages 775-784, March 2000, in press.

[17] Hong I.; Song S.; Park J.; Koo K., "Location-Aware Real Time Positioning with JinifMap", *2010 2nd International Conference on Computer Enginnering and Technology (ICCET)*, Pages 309-312, April 2010, in press.

[18] Heredia B.; Ocanã M.; Bergasa L. M.; Sotelo M. A.; Revenga P.; Flores R.; et al., "People Location System based on WiFi Signal Measure", *IEEE International Symposium on Intelligent Signal Processing*, Pages 1-6, October 2007, in press.

[19] Farivar, R.; Wiczer, D.; Gutierrez A.; Campbell R. H., "A Statistical Study on the Impact of Wireless Signals' Behavior on Location Estimation Accuracy in 802.11 Fingerprinting Systems", *IEEE International Symposium on Computing & Processing (Hardware/Software)*, Pages 1-8, May 2009, in press.

[20] Zhao, M.; Yang, H.; Liu, J.; Chen, Y.; Zhou J., "Directional Wi-Fi Based Indoor Location System for Emergency", *7th International Conference on Ubiquitous Intelligence & Computing and 7th International Conference on Autonomic & Trusted Computing (UIC/ATC)*, Pages 501-502, October 2010, in press.

[21] Zhao, Y.; Zhou, H.; Li, M.; Kong, R., "Implementation of Indoor Positioning System based on Location Fingerprinting in Wireless Networks", *4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, Pages 1-4, October 2008, in press.

[22] Wang, C.; Gao, M.; Wang, X., "An 802.11 Based Location-Aware Computing: Intelligent Guide System", *First International Conference on Communications and Networking in China*, Pages 1-5, October 2006, in press.

[23] Fang S.; Lin T., "Projection-Based Location System via Multiple Discriminant Analysis in Wireless Local Area Networks", *IEEE Transactions on Vehicular Technology*, Pages 5009-5019, November 2009, in press.