Roberto Jorge Melo Veloso

Master of Science

Uma linguagem para modelação de requisitos de interface para WebGIS

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática

Orientador: Prof. Dr. João Araujo, Prof. Auxiliar, Universidade Nova de Lisboa

Co-orientadora: Prof.ª Dra. Armanda Rodrigues, Prof.ª Auxiliar, Universidade Nova de Lisboa

Júri

Presidente: Prof. Dr. Bernardo Toninho
Arguente: Prof.ª Dra. Isabel Brito
Vogal: Prof. Dr. João Araujo

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro, 2019

Uma linguagem para modelação de requisitos de interface para WebGIS

Este documento foi gerado utilizando o processador (pdf)LaTeX, com base no template "novathesis" [1] desenvolvido no Dep. Informática da FCT-NOVA [2].
[1] https://github.com/joaomlourenco/novathesis [2] http://www.di.fct.unl.pt

Aos meus pais.

Agradecimentos

Quero agradecer aos meus orientadores, Armanda Rodrigues e João Araujo. Por todo o apoio, confiança e paciência que me deram durante toda a realização desta dissertação. Um obrigado à Faculdade de Ciências e Tecnologias da Universidade Nova de Lisboa pela formação que me proporcionou. Ao Departamento de Informática, por todos os recursos que providenciou ao longo da minha formação. Um obrigado a todos os que colaboraram nas avaliações sobre o projeto desta dissertação. Aos meus colegas e amigos que de alguma forma convivi ao longo deste percurso. E finalmente aos meus pais e irmão, por tudo, quer na formação quer a nível pessoal.

Resumo

Os Sistemas de Informação Geográfica para a Web (WebGIS) podem pertencer a vários domínios e ter objetivos muito específicos. Utilizar o WebGIS para representar informações sobre uma determinada área geográfica e não apenas a área em si é o panorama real desses sistemas, sendo uma tendência para o futuro. Esta realidade contribui positivamente para o seu crescimento em vários contextos diferentes. Entretanto, há uma falta de abordagens de elicitação de requisitos em WebGIS, limitando a comunicação entre os *stakeholders*, comprometendo a sistematização do desenvolvimento e consequentemente a qualidade desses sistemas.

Portanto, esta dissertação, apresenta uma linguagem para modelação de requisitos de interface para WebGIS suportada por um editor, que foi desenvolvida com os conceitos de desenvolvimento orientado a modelos (MDD), com o objetivo de melhorar o processo de modelação de requisitos. Uma linguagem que engloba os conceitos principais de interfaces de WebGIS (assim tornando-se mais próxima dos stakeholders) tem o potencial de facilitar a comunicação entre os stakeholders e a equipa de desenvolvimento.

Para avaliar a linguagem, um experimento envolvendo 30 participantes (maioritariamente engenheiros informáticos) foi realizado, a fim de medir a facilidade de compreensão dos modelos da linguagem e obter feedback dos participantes sobre a mesma. Em geral os resultados foram bastante positivos, encorajando o uso da linguagem em ambientes de desenvolvimento de WebGIS.

Palavras-chave: WebGIS, requisitos, modelação de requisitos, linguagens de domínio específico.

Abstract

Geographic Information Systems for Web can belong to several domains and have very specific goals. Using WebGIS to represent information about a particular geographic area and not just that area itself is the actual panorama of these systems, being a trend for the future. This reality contributes positively to their growth in several different contexts. Meanwhile, there's a lack of requirements elicitation approaches upon WebGIS applications, restricting the communication between stakeholders, compromising the systematization of development, and the quality of these systems.

Therefore, this dissertation, presents a requirements modeling language of interface for WebGIS supported by an editor, which was developed with Model Driven Development (MDD) concepts, aiming to improve the process of requirements modeling. A language that includes the main concepts of WebGIS interface (becoming closer to the stakeholders) has potential to relieve the communication between stakeholders and the development team.

To evaluate the language, an experiment involving 30 participants (mostly IT Engineers) was performed, in order to measure the ease of understanding of language models and get the feedback of participants about it. In general the results were quite positive, encouraging the use of the language in WebGIS development environments.

Keywords: WebGIS, requirements, requirements modeling, domain specific languages.

Índice

Lista de Figuras

Lista de Tabelas

Acrónimos

APA	Agência Portuguesa do Ambiente
API	Application Programming Interface
ATL	Atlas Transformation Language
DSML	Domain-Specific Modeling Languages
EMF	Eclipse Modeling Framework
EMT	EMF Model Transformation
EOL	Epsilon Object Language
EVL	Epsilon Validation Language
GIS	Geographic Information System
GPS	Global Positioning System
IRML	Interface Requirements Modeling Language
KAOS	Knowledge Acquisition in Automated Specification
MDA	Model Driven Architecture
MDD	Model Driven Development
MDGORE	Model-Driven in Goal-Oriented Requirements Engineering
RDAL	Requirements Definition and Analysis Language
SDLC	Software Development Life Cycle
UML	Unified Modeling Language
WebGIS	Web Geographic Information Systems

WebML Web Modeling Language

Capítulo 1

Introdução

Este capítulo faz uma introdução ao problema, à motivação que nos leva a resolvê-lo e à sua solução. São também destacados os benefícios deste projeto e é apresentada a estrutura de todo o documento.

1.1 Contexto

O crescimento de aplicações WebGIS está relacionado com os avanços tecnológicos visíveis nos dias de hoje, assim também como a utilização efémera de algumas dessas aplicações. Já não se contam os países que têm acesso à internet mas sim aqueles que não têm e face a uma colaboração mundial no desenvolvimento de novas tecnologias, a sociedade cibernética ganhou o benefício nesta nova era de um acesso com custos reduzidos a essas tecnologias.

O aparecimento em massa de interfaces de programação de aplicações (*API*) com mapas, com acessibilidade relativamente fácil e de custos reduzidos tem vindo a estimular uma procura sem precedentes no que toca a estes ambientes (mapas geográficos). A par deste fenómeno veio também a evolução de dispositivos tecnológicos com características de localização global [1] (um exemplo deste tipo de dispositivos é o *GPS*). Hoje em dia, telemóveis, tablets e outro tipo de *gadgets*, contêm tecnologia suficiente para fornecer a sua posição no globo, em tempo real. Estes avanços e ofertas tecnológicas levaram as sociedades a seguir novas tendências e com novas tendências, novos hábitos. O mundo está ligado a todo o momento e todo o momento tem a sua localização. Esta ideia levou a um "*big bang*", na oferta de aplicações de Sistemas de Informação Geográfica para a web (*WebGIS*). Já existem aplicações nos mais diversos domínios: governamentais, ambientais, geográficos, territoriais, etc. Alguns exemplos poderão ser observados no capítulo três, sobre o trabalho relacionado.

1.2 Motivação

A facilidade com que um utilizador consegue mergulhar no conhecimento tecnológico permite ao mesmo tempo, em alguns casos, que ele se envolva na exploração e dê o seu contributo. Aqui nasce o ponto de partida para o problema que se pretende tratar: com o aumento de utilizadores há também um acréscimo de novas formas de desenvolvimento. Isto é, vários *stakeholders* podem não partilhar o mesmo domínio de carreira (Informática, geografia, química, entre outros) levando ao levantamento de requisitos de interface diferentes devido à formação diversa dos utilizadores. Com diferentes tipos de requisitos, há uma falha em capturar os conceitos que são comuns numa aplicação WebGIS. Num domínio específico, a inexistência de uma linguagem comum de modelação de requisitos para interface de aplicações WebGIS dificulta a comunicação entre os diversos *stakeholders*. Isto é, as partes interessadas que possuem formações de domínios distintos que não são o de Informática e o desenvolvimento de software ou neste caso em concreto, o desenvolvimento de aplicações WebGIS.

Eventualmente, poderá existir uma divergência entre o que era pretendido pelos stakeholders e o que foi produzido pelos técnicos de desenvolvimento, os programadores. Deficiências na elicitação de requisitos levam a custos elevados no desenvolvimento de software numa fase mais avançada de projeto. Estes sistemas vão continuar certamente a serem explorados e a crescerem cada vez mais à medida que o tempo avança e vão surgindo novas tecnologias, mas ainda assim há uma escassez de linguagens de modelação de requisitos que detetam os conceitos fundamentais que são comuns aos WebGIS. Portanto, é necessária uma linguagem de modelação de requisitos específica para especificar os requisitos de interface de aplicações WebGIS, não só para as partes interessadas conseguirem exprimir o que realmente pretendem, mas também para facilitar a transmissão da informação à equipa de desenvolvimento. Uma linguagem especializada neste objetivo pode facilitar a vida a um engenheiro de Software. Isto vai permitir um desenvolvimento mais sistematizado e rigoroso, tanto na especificação de requisitos como também das aplicações WebGIS.

A definição de uma linguagem de modelação de requisitos de interface para WebGIS, é uma das razões determinantes que nos movem no desenvolvimento deste projeto, isto é, com um elevado número de partes interessadas na utilização destas tecnologias, surge a necessidade de suavizar o processo de criação dos sistemas desde os requisitos iniciais até à fase de execução técnica.

1.3 Objectivo

A finalidade desta dissertação é conseguir convergir parte da especificação de WebGIS de domínios distintos, em particular a modelação de requisitos. Pretende-se atingir este objetivo, através de uma linguagem de modelação que consiste num conjunto de conceitos, regras e padrões que são comuns a interfaces WebGIS, de forma a suportar o constante

crescimento desta tecnologia que abrange matérias de campos diferentes e investigadores bem como técnicos de várias áreas com propósitos heterogéneos. Conseguir que um *stakeholder* especifique os requisitos que pretende ter na interface WebGIS de forma clara é o objetivo.

Assim, para conseguir este objectivo acreditamos, como já referido, que a solução para este problema passa pela implementação de uma linguagem de modelação de requisitos de domínio específico para interface WebGIS.

Através de conceitos bem definidos e uma sintaxe concreta formada com base no Metamodelo da linguagem (apresentado mais adiante), foi desenvolvido um editor de modelação. Este servirá para realizar a modelação de requisitos, para uma interface WebGIS, que vão ao encontro de um domínio específico.

1.4 Contribuições

Face ao problema definido, criou-se uma linguagem para apoiar a elicitação dos requisitos de interface para aplicações WebGIS. As principais contribuições são:

- A própria linguagem de modelação de requisitos de domínio específico, que simplifica a elicitação dos mesmos, necessários à implementação da interface de WebGIS;
- O editor da linguagem de modelação de requisitos, que permite a criação (por parte de pessoas de todos os domínios) de modelos, que representam interfaces para WebGIS, de acordo com os seus objetivos.

1.5 Organização do documento

O restante do documento está organizado da seguinte forma:

- **Capítulo 2** - Aborda os conceitos fundamentais do problema, WebGIS, Linguagens de modelação, Engenharia de requisitos.
- **Capítulo 3** - Vai ao encontro do trabalho relacionado, tanto na linguagem como na modelação dos requisitos e também sobre os aspetos de desenvolvimentos de WebGIS.
- **Capítulo 4** - Transmite o trabalho que foi desenvolvido para a solução do problema, a linguagem. São também apresentados os conceitos que a constituem. E por fim aborda o editor da linguagem que foi desenvolvido para a solução do problema, a modelação de requisitos. São também apresentadas as tecnologias presentes nele que permitiram o seu levantamento.
- **Capítulo 5** - Reflete a validação realizada para salvaguardar a consistência da linguagem através de um caso de estudo acerca de uma aplicação WebGIS seleccionada.

- **Capítulo 6** - Descreve a avaliação da linguagem mediante a utilização da mesma por entidades empresariais e académicas.
- **Capítulo 7** - Trata a conclusão do relatório desta dissertação, bem como um balanço final e uma direcção de possíveis futuros trabalhos relacionados com esta linguagem.

Capítulo 2

Enquadramento

Este capítulo aborda os temas indispensáveis desta dissertação de modo a aproximar o leitor ao problema que está a ser tratado. Ao desenvolver uma aplicação WebGIS, existem várias características da mesma que definem os seus requisitos. Para uma melhor compreensão do trabalho desenvolvido, são aqui apresentadas algumas delas, como também, sobre Linguagens de Modelação de Domínios Específicos e Engenharia de Requisitos.

2.1 Sistemas de Informação Geográfica

GIS ou em português SIG, são sistemas de informação utilizados para capturar, modelar, armazenar, recuperar, partilhar, manipular, analisar e apresentar dados geograficamente referenciados [2] (capítulo 1, página 2).

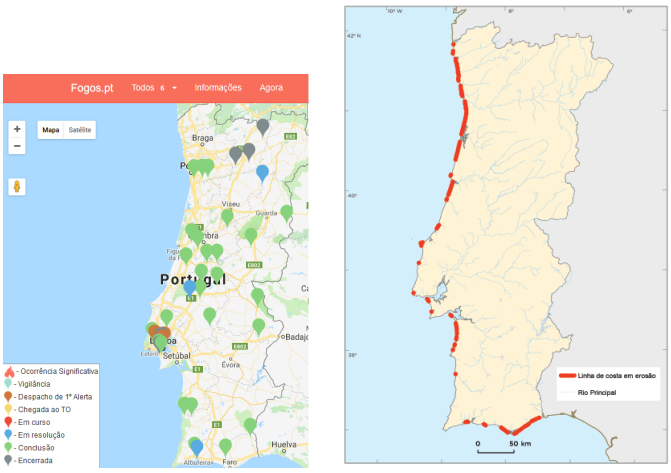
Relacionando com uma definição de Sistemas de Informação, "Os sistemas de informação são componentes inter-relacionados que trabalham em conjunto para colecionar, processar, armazenar e divulgar informações que apoiam na tomada de decisões de uma organização." [3] (capítulo 1, página 45) , podemos então assumir que WebGIS são GIS especializados para apresentar informação georreferenciada na web.

As aplicações GIS permitem que os utilizadores possam executar queries e analisar informação espacial, editar dados em mapas e mostrar o resultado de todas estas operações [4]. Estes sistemas são utilizados por diversas partes, tais como: instituições de investigação, ciências do ambiente, organizações de saúde, planeamento urbano, agências governamentais e algumas empresas [5].

2.1.1 Relevância atual dos SIG

Acontecimentos, fenómenos e eventos podem acontecer a qualquer hora, altura ou lugar. Todos eles têm um nível de importância de acordo com seu impacto na sociedade. Quando as ocorrências têm especial importância no mundo ao seu redor, é importante estar a par do local do seu acontecimento. Os GIS trouxeram a possibilidade de tomar conhecimento sobre determinados locais de forma remota e com esta mesma possibilidade veio também a habilidade para gerir estratégias, políticas e planeamentos[6] (capítulo 1, páginas 4-8). Minas, Cabos subterrâneos, poços petrolíferos, incêndios, voos são alguns dos motivos que destacam a importância da utilização destes sistemas.

Como exemplo, na Figura 2.1a observamos um Sistema de Informação Geográfica português que transmite em tempo real a ocorrência de incêndios e os seus estados no território de Portugal Continental. Na medida em que os incêndios são um dos problemas mais graves do país, este sistema pode ser bastante útil na prevenção, definição de estratégias e combate contra este problema. Por outro lado na Figura 2.1b é apresentado um mapa territorial que esboça as zonas críticas de erosão da costa portuguesa ao longo de 52 anos.



a Fogos.pt

b Mapa de erosão da costa portuguesa 1958 a 2010. APA 2016

Figura 2.1: SIG e mapa portugueses

2.1.2 Conceitos Fundamentais

Para contextualizar o leitor sobre este tema, são abordados nesta secção alguns conceitos fundamentais de SIG.

2.1.2.1 Conjuntos de Dados

Tal como foi referido no início deste capítulo, os SIG trabalham essencialmente com dados georeferenciados. Ainda assim há uma necessidade de conseguir integrar diferentes

conjuntos de dados no mesmo sistema, de forma a conseguir produzir informação útil, devolvendo uma visão integrada da região, no contexto dos tópicos associados aos dados. Para atingir este objetivo é necessário a identificação e colecta dos conjuntos de dados relevantes e necessários para se conseguir produzir os resultados pretendidos.

Na Figura 2.2, observamos um SIG para a web português, conhecido por LXConventos[7][8][1], cujo o objetivo é fornecer informações sobre edifícios religiosos pertencentes a ordens religiosas extintas no Séc. XIX. O objetivo deste projeto é o estudo da evolução da estrutura da cidade de Lisboa e a influência destes edifícios religiosos nessa mesma estrutura. Atualmente a grande maioria destes edifícios cumprem funções de grande importância na cidade, tais como Escolas, Hospitais e até o Parlamento.

Como deverá ser perceptível à primeira vista, para este sistema foram necessárias pelo menos duas bases de dados, uma para a localização dos edifícios que, na Figura 2.2 conseguimos detectar através das áreas coloridas a azul, e outra para a informação associada a cada um deles. Desta forma, a construção deste sistema só foi possível graças à integração destes dois conjuntos de dados, e à colaboração de diferentes serviços da Câmara Municipal de Lisboa.

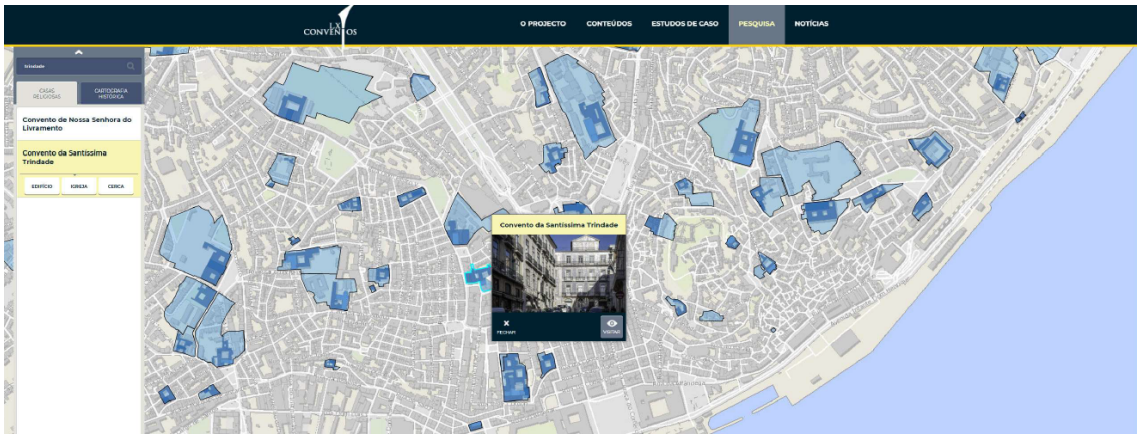


Figura 2.2: LX Conventos

2.1.2.2 Representação de dados geográficos

Para conseguir fazer uma representação geográfica num SIG, é necessária uma transformação do mundo real para o mundo digital. Nesta transformação, é recolhida uma grande quantidade de informação. Para conseguir simplificar este processo e permitir que tal informação possa ser utilizada computacionalmente, existem dois formatos de base para representar dados geográficos, Raster e Vectorial [9].

No método de representação Raster, o espaço é fragmentado numa matriz de células retangulares, onde são depois atribuídas características de acordo com a área geográfica real de cada uma delas. Já na representação Vetorial, as formas dos objetos são definidas através de pontos que estão interligados por linhas retas, quanto mais curvilíneas forem

[1] http://lxconventos.cm-lisboa.pt

as representações mais pontos são necessários para as formar. Esta é uma característica fundamental dos SIG, que determina muitas das opções a tomar em sequência, pelo que tem de ser tida em conta no momento da modelação do sistema. Na Figura 2.3 podemos observar um comparação entre estes dois métodos de representação.

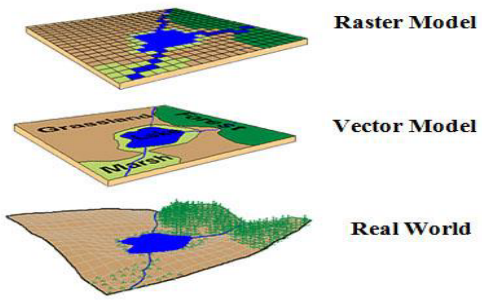


Figura 2.3: Modelos de representação geográficos

2.1.2.3 Recolha de Dados

O processo de recolha e obtenção de dados geográficos para SIG pode ser realizado de várias formas. Primeiramente são classificados por duas maneiras: Primários e Secundários [6]. Os dados primários (ou de base) são aqueles que foram recolhidos para estruturar o espaço, os secundários são os dados que foram obtidos que permitem analisar e estudar o espaço e serem aproveitados posteriormente no sistema. A forma de recolha dos dados secundários diverge. Relativamente aos dados primários, estes devem ser obtidos junto de fontes credenciadas. Na Figura 2.4 é apresentada a classificação destes tipos de dados.

	Raster	Vector
Primary	Digital satellite remote-sensing images	GPS measurements
	Digital aerial photographs	Survey measurements
Secondary	Scanned maps or photographs	Topographic maps
	Digital elevation models from topographic map contours	Toponymy (placename) databases

Figura 2.4: Exemplos de classificação de dados geográficos[6].

2.1.2.4 Sistemas de Coordenadas

Outro conceito fundamental dos SIG é o sistema de coordenadas. O sistema de coordenadas tem como objetivo fornecer uma referência para a representação de localizações geográficas num mapa ou num sistema cartográfico. Existem vários sistemas de coordenadas

que podem ser utilizados, ainda assim para mapas de menor extensão utilizam-se sistemas de coordenadas locais [10] (capítulo 2). Os sistemas de coordenadas locais focam-se nas características físicas do espaço, requerendo, como em qualquer sistema de coordenadas a especificação de uma origem, dois eixos (quando em mapas bi-dimensionais) e uma unidade de medida. Mapas de redes de tranportes, parques públicos e campus geralmente utilizam este tipo de sistemas de coordenadas. Por outro lado, também existem sistemas de coordenadas para espaços maiores, os geodésicos. Sistemas de coordenadas geodésicos, são sistemas criados para aplicação de coordenadas à escala global, porém, podem ter como objetivo a minimização de erro numa determinada região. Este tipo de sistema é concebido à base de modelos matemáticos com a forma do planeta Terra. Esta que não é esfericamente perfeita e possui uma superfície irregular, é então modelada como um elipsóide. Depois há ainda o processo de posicionamento da elipsóide no geóide(modelo físico da Terra mais aproximado da realidade, com uma escala gravitacional)[11] (capítulo 2, página 15). Este processo é realizado através do Datum Geodésico.

2.1.2.5 Datum Geodésico

A utilização de um sistema de coordenadas (conceito falado no tópico anterior) à escala global tem realmente vantagens, mas também algumas preocupações. Uma delas é o facto de a Terra não ser perfeitamente redonda e ser irregular. Face a isto, houve necessidade de se criar uma forma para aproximar ainda mais a representação da superfície terrestre da realidade, e aqui nasce o Datum. Datum geodésico é um sistema que fornece um modelo matemático da Terra baseado numa elipsóide [11], permitindo assim estabelecer a distância entre um local e dois planos utilizando o Datum como referência.

2.1.3 Integração dos dados

Cada implementação de uma aplicação SIG é um modelo da realidade. Para que este modelo seja possível, é necessário um determinado nível de abstração sobre esta realidade. Em SIG, este processo de abstração, é realizado através da integração de várias camadas de conjuntos de dados diferentes [12] que têm de estar todos no mesmo sistema de coordenadas. Isto faz gerar a informação final que o utilizador observa como se fosse o mundo real. Na Figura 2.5 é ilustrado este fenómeno de integração de camadas. Podemos observar a existência de três camadas, a camada das estradas, dos edifícios e da vegetação. Ao integrá-las é formado o mapa da localização que se pretende abstrair e apresentar ao utilizador.

2.1.4 Modelação SIG

Nos SIG, existem dois modelos principais de representação. O modelo baseado em camadas e o modelo baseado em objetos[2] (capítulo 4, página 137). Estes modelos foram concebidos para permitir uma abstração do mundo real dentro destes sistemas. No modelo baseado em camadas, podemos observar que a prioridade é a região espacial que se

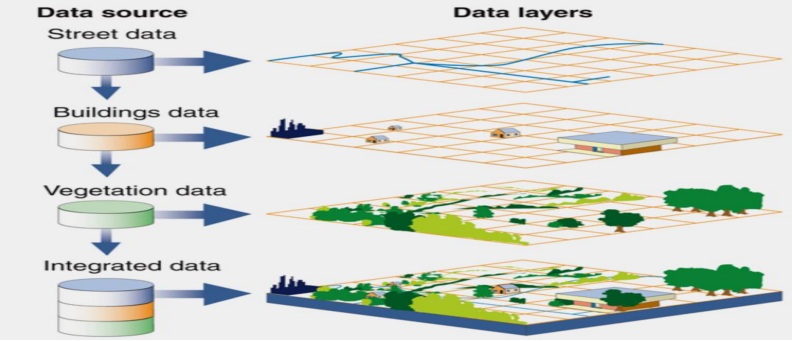


Figura 2.5: Camadas de dados SIG[13]

pretende estudar. Nesta abordagem diversas camadas de informação são integradas de modo a complementar a informação de base já existente. O Sistema Nacional de Informação de Recursos Hídricos, disponibiliza uma aplicação WebGIS sob um modelo baseado em camadas. É possível observar na Figura 2.6a), a aplicação[2]. Observamos que existe uma legenda de etiquetas com vários tipos de dados disponíveis para apresentar no mapa. Como estão seleccionadas duas etiquetas, uma verde que diz respeito à qualidade das águas subterrâneas e outra amarela que representa a qualidade das águas balneares, são ambas representadas no mapa da aplicação, sendo cada uma delas uma camada de dados. Por outro lado, existe o modelo baseado em objetos, que consiste em abstrair o mundo em objetos, que, por sinal, contêm, como atributo, uma forma e localização espacial. Um exemplo comum deste tipo de modelação são os diagramas dos metropolitanos (Figura 2.6b) onde as estações são representadas por pontos que estão ligados por uma linha que representa as linhas férreas. Este tipo de mapas não são fidedignos no que diz respeito à localização, ainda assim, possuem uma abstracção a objetos.

2.1.5 Arquitetura SIG

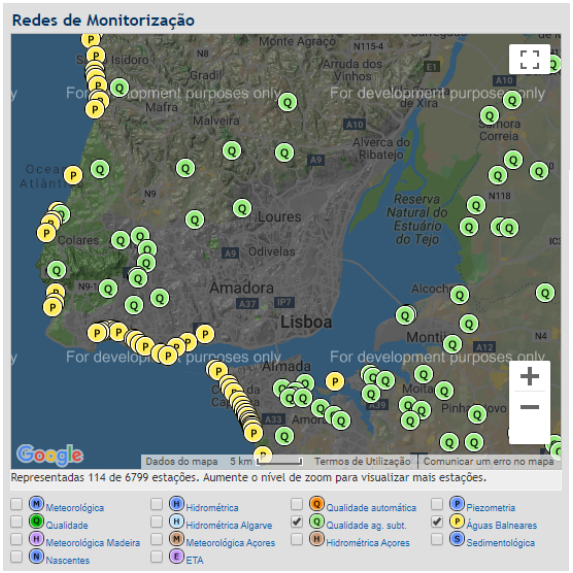
Não apenas em Sistemas de Informação Geográfica, mas essencialmente nos Sistemas de Informação, é necessário existir uma estrutura e uma organização de todas as partes que formam esses mesmos sistemas. A isto chamamos de arquitetura de um sistema [2] (capítulo 7, página 259).

As diversas arquiteturas de plataformas SIG podem ser classificadas segundo duas características de importância, a Modularidade e a Interoperabilidade [14]:

- **A Modularidade** é a qualidade que caracteriza um sistema construído a partir de unidades de software independentes com funções padronizadas ou claramente definidas;

- **A Interoperabilidade** é a capacidade de dois ou mais sistemas de informação partilharem dados, informação ou capacidade de processamento.

[2]https://snirh.apambiente.pt/



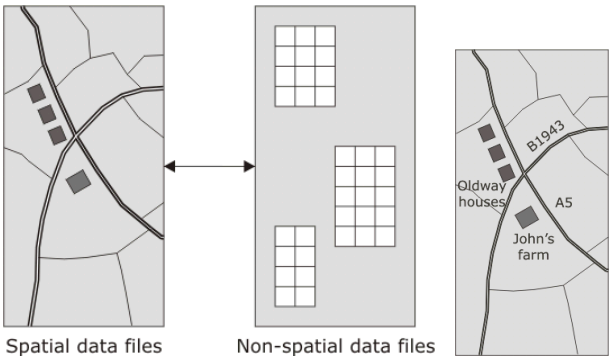
a Camadas

b Objetos

Figura 2.6: Tipos de modelação

Segundo Worboys, historicamente, as arquiteturas SIG são caracterizadas como: Híbridas, Integradas e Compostas [2] (capítulo 7, páginas 260/261). Numa arquitetura híbrida, os dados espaciais (geometria de regiões específicas bem como as relações entre elas) são guardados numa zona à parte dos dados não espaciais (nomes das regiões específicas, moradas, identificadores). Esta era a arquitetura dos SIG originais, ainda se perpetuando na forma de conceptualizar os SIG, em muitos casos. Neste tipo de arquitetura, originalmente, os dados espaciais (gráficos) eram armazenados num sistema de ficheiros enquanto os dados não espaciais residiam numa base de dados relacional. Esta abordagem, que proporciona uma separação dos dados, tem um custo elevado relativamente à segurança e integridade dos mesmos. Desta maneira, mais recentemente, adotou-se uma abordagem integrada, em que o sistema guarda informação sobre objetos com determinadas características, sendo que alguma delas têm uma mapeamento espacial (geográfico). Nas Figuras 2.7 a) e b) podemos observar uma abstração destas arquitecturas.

Por último existe também a abordagem de uma arquitetura composta, que se foca essencialmente em apresentar um sistema baseado em componentes que o estruturam. Os componentes podem ser a interface gráfica, a base de dados, uma componente de comunicação (no caso de ser necessário algum tipo de comunicação específica para o SIG funcionar), entre outros. Nesta abordagem, os componentes podem ser adicionados à medida que o SIG se vai desenvolvendo. Podemos tomar como exemplo um sistema em que se vão adicionando funcionalidades específicas, sendo uma delas, uma ferramenta para análise espacial.



a Híbrida b Integrada

Figura 2.7: Arquitetura Híbrida e Arquitetura Integrada

2.2 WebGIS

WebGIS ou GIS baseado na Web, foi um novo paradigma que nasceu juntamente com os serviços geográficos WEB [12] disponibilizados por tecnologias similares ao Google Maps, Bing Maps e OpenLayers [15]. Estas tecnologias vieram permitir a troca de informação entre o servidor onde os mapas estão armazenados e o cliente que os utiliza no seu terminal.

O desenvolvimento de WebGIS tem crescido muito nas últimas décadas, este fenómeno deve-se em grande parte à vasta propagação de tecnologias que permitem aos utilizadores criar e publicar mapas com informação personalizada [16].

A principal diferença entre GIS e WebGIS é que neste último os dados geográficos estão a ser utilizados no browser a partir da rede, enquanto no primeiro os dados espaciais correm a partir do terminal [12].

Dado o seu desenvolvimento ser ainda relativamente recente, as técnicas de desenvolvimento ainda não são tão bem adequadas. [17]. As propriedades de um WebGIS, nomeadamente o mapa, os dados georeferenciados e ferramentas Web acessórias para o bom funcionamento da aplicação, têm todos eles origens distintas. Isto implica uma interoperabilidade de sistemas que antes não existia num SIG onde todas estas características estavam no mesmo lugar.

2.2.1 Arquitetura WebGIS

Como tínhamos visto anteriormente, existem três tipos de arquiteturas SIG. No que diz respeito a WebGIS a abordagem já não será a mesma. Em WebGIS a arquitetura será sempre distribuída [6]. Isto acontece, porque, como estamos a lidar com uma aplicação para a Web, os dados geoespaciais têm de ser armazenados e disponibilizados na rede (Cliente - Servidor) [2]. Desta forma é assegurada a disponibilidade dos dados, serviços e recursos por parte do servidor para os utilizadores finais (clientes) distribuídos na rede. Estes, a partir de um computador normal ou de um dispositivo movel, enviam pedidos ao

servidor (web-services) a partir de um browser ou outro tipo de aplicação. Ainda assim, as arquiteturas de aplicações WebGIS podem abordar várias estratégias [2] (capítulo 7, páginas 268-271) [18] (um pouco na linhas das arquiteturas web genéricas):

- *Two-Tier* - Arquitetura cliente-servidor básica onde todos os sistemas de informação dentro da arquitetura são clientes ou servidores.
- *Multi-tier* - É semelhante à *Two-Tier* distinguindo-se pelo facto de possuir uma camada intermediária entre o cliente e o servidor. Esta camada funciona exatamente como cliente para o servidor e como servidor para o cliente aplicando um protocolo diferente para cada caso.
- *Thin-Client-Thick-Server*, como os adjetivos thin(fino) e thick(grosso) indicam, nesta estratégia a parte *thick* fica responsável pelo maior processamento do sistema, neste caso o servidor.
- *Thick-Client-Thin-Server*, ao contrário da estratégia anterior, aqui, o cliente é que realiza as operações e cálculos mais pesados.

2.3 Engenharia de Requisitos

No contexto de Engenharia de Software e segundo a norma IEEE 610.12:1990 [19], um requisito diz respeito a:

- Condição ou capacidade necessária por um utilizador para resolver um problema ou atingir um objetivo.
- Uma condição ou capacidade que tem de ser cumprida ou possuída por um sistema ou um componente de sistema para satisfazer um acordo, padrão, especificação ou outros documentos formalmente impostos.
- Uma representação documentada de uma condição ou capacidade como nos pontos 1 e 2.

Já numa abordagem mais profunda, um requisito pode assumir vários tipos[20]:

- **Requisitos Funcionais** - São serviços que o sistema deve fornecer, formas como o sistema deve reagir a entradas específicas e como o sistema se deve comportar em determinadas situações. Nalguns casos, este tipo de requisitos pode também indicar o que o sistema não deve fazer.
- **Requisitos Não Funcionais** - São restrições nos serviços ou funções fornecidas pelo sistema. Incluem restrições de tempo, restrições no processo de desenvolvimento e restrições impostas pelos padrões. Requisitos não funcionais geralmente aplicam-se ao sistema como um todo, em vez de recursos ou serviços individuais.

- **Requisitos de Utilizador** - Condições de alto nível de abstração, definidos a partir de uma linguagem natural com a complementação de diagramas sobre quais os serviços que o sistema deve oferecer aos utilizadores do sistema e as restrições sob as quais ele deve operar.

- **Requisitos de Sistema** - São descrições mais específicas de funções, serviços e restrições operacionais do sistema. Este tipo de requisitos deve definir exatamente o que é para ser implementado. Pode fazer parte do contrato entre o Stakeholder e os responsáveis pelo desenvolvimento de software.

- **Requisitos de Domínio** - São derivados do domínio de aplicação do sistema e não das necessidades específicas dos utilizadores do sistema. Estes podem ser novos requisitos funcionais por si só, restringir requisitos funcionais existentes ou então definir como funções computacionais específicas devem ser executadas.

De acordo com uma definição de Pamela Zave, "Engenharia de Requisitos é o ramo de Engenharia de Software preocupado com os objetivos do mundo real para funções e restrições nos sistemas de software. E é também preocupado com a relação desses fatores para conseguir especificações precisas do comportamento de software e a sua evolução ao longo do tempo e entre famílias de software" [21]. Por outro lado Engenharia de Requisitos pode também ser definida como o processo de investigação, análise, documentação e verificação de serviços e restrições de um sistema[22]. Na Figura 2.8 é apresentado um esboço do processo de Engenharia de Requisitos no modelo em espiral[22].

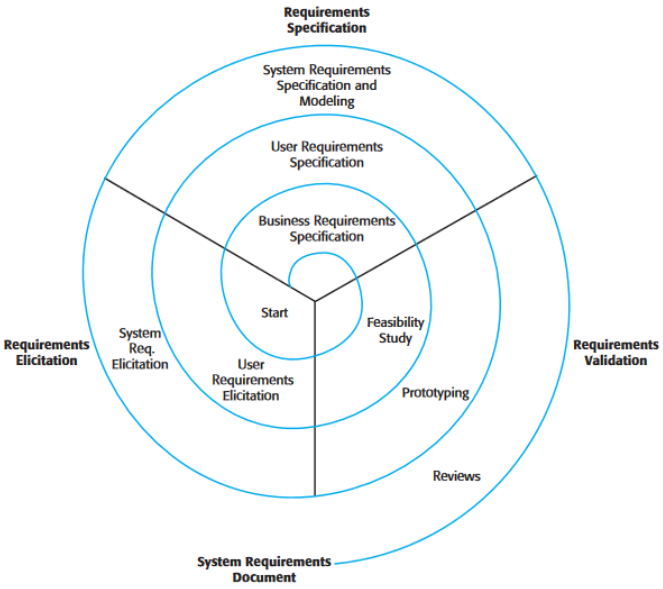


Figura 2.8: Modelo em espiral do processo de Engenharia de Requisitos.

Este processo é caracterizado por quatro etapas fundamentais da Engenharia de Requisitos. A avaliação do sistema face ao modelo de negócio (Feasibility study), a elicitação

de requisitos, a transformação destes numa forma padrão (especificação) e a validação, que verifica se os requisitos estão de acordo com o sistema.

Analisando o custo de reparação de anomalias de software em determinadas fases da sua construção [23] na Figura 2.9, podemos concluir que a Engenharia de Requisitos é um processo de elevado grau de importância, dado que influencia outras etapas seguintes de desenvolvimento do sistema.

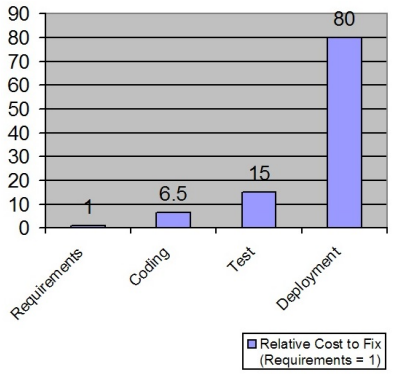


Figura 2.9: Custos de reparação em diferentes fases de desenvolvimento de Software.

2.4 MDD

É um conceito de Desenvolvimento de Software, onde os artefactos principais são modelos e transformações. Estes podem sofrer várias alterações durante o processo de desenvolvimento [24]. Nesta abordagem de MDD, a produção de software é automaticamente gerada a partir dos modelos.

Num contexto de Desenvolvimento Orientado a Modelos, tomando de exemplo programas Java, estes são considerados Modelos. Na Figura 2.10 podemos observar uma relação lógica levantada por Thomas Kühne entre os conceitos fundamentais de DSML (Metamodelo, Modelo e Linguagem) e também o seu papel.

O Metamodelo define as propriedades e operações da linguagem especificando-a e faz a instanciação do Modelo que permite fazer a edição das especificações do sistema.

MDD é importante para o projeto desta tese, pois será a partir desta abordagem que o protótipo do editor da linguagem de modelação de requisitos para WebGIS se vai desenvolver.

De seguida são apresentados alguns conceitos de engenharia, desenvolvimento e arquitetura orientados a modelos [24]:

- MBE – Model-Based Engineering, É o processo onde os modelos têm um peso importante no desenvolvimento mas não são os artefactos principais.

- MDE – Model-Driven Engineering, Não se resume às atividades de desenvolvimento puras e engloba também outras tarefas baseadas em modelos de um processo

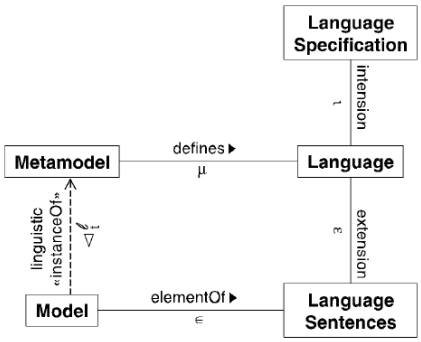


Figura 2.10: Definições de Linguagem [25]

completo de engenharia de software.

- MDD – Model-Driven Development, Como foi possível estudar no tópico anterior, desenvolvimento orientado a modelos utiliza os modelos e transformações como elementos principais do processo de desenvolvimento.
- MDA – Model-Driven Architecture, É uma abordagem MDD particular levantada pelo OMG(Object Management Group), que consiste na execução de MDD com padrões definidos por este grupo particular.

2.4.1 Modelo

Segundo Thomas Kuhne, no contexto de Engenharia de Software um modelo é um artefacto formulado numa linguagem de modelação (e.g., UML) representando um sistema através da ajuda de vários tipos de diagramas [26]. Habitualmente, descrições de modelo são orientadas a diagramas.

Modelos podem também ser considerados como a fonte principal na qual os técnicos de desenvolvimento criam, editam e apagam as especificações do sistema [27]. Modelos de domínio específico são utilizados para depois de todas as alterações das especificações do sistema, gerar código.

2.4.2 Metamodelo

Metamodelo é o modelo conceptual que define os conceitos e relações entre conceitos da linguagem de um domínio específico [26], isto é, a partir do Metamodelo define-se o conjunto de operações, propriedades e características distintas que vão formar um sistema. Através do Metamodelo é instanciado um modelo.

2.4.3 Transformação de Modelos

Existem várias definições para transformação de modelos, destaquemos esta:

"Geração automática de um ou múltiplos modelos alvo a partir de um ou múltiplos modelos origem, de acordo com uma descrição de transformação." [28]

Podemos então assumir que a tranformação de modelos diz respeito à gestão automática de modelos de origem/entrada com o objetivo de gerar modelo ou modelos alvo/de saída de acordo com um conjunto de regras e propósitos.

Na Figura 2.11, é ilustrada uma terminologia de transformação de modelos que resume parte do que foi definido inicialmente. Tendo um modelo origem que respeita um metamodelo, é gerado um modelo alvo de acordo com o metamodelo alterado a partir de uma especificação de transformação. Esta especificação obedece a um conjunto de regras de uma linguagem de transformação. Todo este processo é feito totalmente ao mesmo nível de abstração, pois como se pode observar, ambos os metamodelos de origem e destino, bem como a linguagem de transformação respeitam o meta-metamodelo.

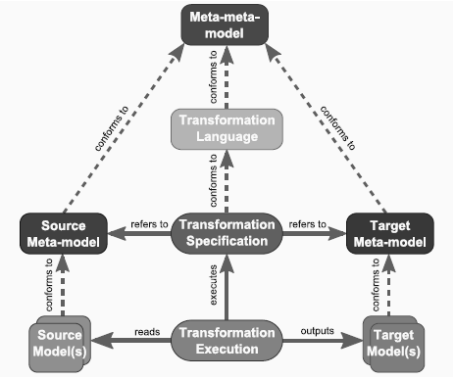


Figura 2.11: Terminologia de transformação de modelos[29]

Nas transformações de modelos, existem duas abordagens de transformação que são muito utilizadas, Modelo para Modelo e Modelo para Texto, sendo que esta última é a mais procurada[30]. Para realizar estas transformações é necessária a utilização de linguagens de transformação. Existem várias, destaquemos a EMT. Esta, é uma framework que fornece uma interface gráfica para descrever regras de transformação num modelo Ecore. Esta framework foi criada para suportar alterações de modelos EMF baseados em regras de transformação gráficas[31]. Na eventualidade desta dissertação vir ser explorada por outras entidades no futuro, se existir interesse para realizar transformações, esta poderá ser a abordagem mais apropriada a tomar.

2.5 DSML

DSML- vai ao encontro de conceitos de desenvolvimento de Software orientado a modelos (Model Driven Software), Metamodelo, Modelo e Linguagem de Domínio Específico.

Linguagens de Modelação de domínio específico concentram-se no desenvolvimento de software de forma automatizada numa determinada área (domínio específico) de interesse [27].

Nesta abordagem, quanto mais restrito for o domínio alvo, maior será a capacidade de suporte para o seu processo de especificação, bem como o acesso à geração de código

automático. Outra grande vantagem de DSML é a habilidade para as tarefas de desenvolvimento, dado que nesta abordagem há um conhecimento maior sobre o domínio do problema ao contrário das linguagens de propósito geral onde é necessário realizar uma abstração maior face ao problema de um domínio específico.

Propriedades importantes de uma linguagem:

- **Sintaxe Abstrata** - A sintaxe abstrata de uma linguagem está diretamente ligada ao metamodelo da mesma, pois a definição dos conceitos de modelação é feita neste nível [24]. Os modelos são abstrações, daí o metamodelo que fornece a habilidade de instanciação de modelos, define a sintaxe abstrata da linguagem.
- **Sintaxe Concreta** - Quando falamos de sintaxe concreta em linguagens de modelação, falamos sobre os artefactos gráficos ou textuais utilizados para processar os elementos dos modelos nos editores de modelação [24].
- **Semântica** - A semântica de uma linguagem de modelação, define o significado de um modelo nessa mesma linguagem e o efeito da sua execução [32]. Vimos que a sintaxe abstrata é definida pelo metamodelo e que os elementos gráficos/textuais dos editores de linguagem definem a sintaxe concreta. A semântica pode ser definida no metamodelo, mas na prática ela é derivada do comportamento do tempo de execução.

2.6 Sumário

Este capítulo aborda as propriedades dos conceitos fundamentais ao desenvolvimento deste projeto. GIS, WebGIS, Engenharia de Requisitos, Linguagens de Modelação de Domínios Específicos e MDD. Sobre GIS e WebGIS são abordados os conceitos fundamentais, tipos de modelação e arquitetura, motivações à implementação destes sistemas e ainda a sua importância e funcionalidades. No tópico da Engenharia de Requisitos, é explicada a sua importância no desenvolvimento de Software. São também referidos os elementos indispensáveis no âmbito de DSML e MDD. No próximo capítulo discutimos os trabalhos relacionados.

Capítulo 3

Trabalho Relacionado

Este capítulo aborda uma investigação sobre outros trabalhos já realizados no âmbito de linguagens para modelação WebGIS, projetos WebGIS e linguagens de modelação de domínio específico para requisitos. O objetivo é verificar o estado de arte de desenvolvimento nos três sectores e relacioná-los com este projeto. Primeiro é dado o destaque acerca de modelação no contexto de WebGIS, dado o objetivo principal ser uma ferramenta para dar suporte a este tipo de aplicações. São também abordados vários projetos WebGIS onde é realizada uma análise sobre as funcionalidades comuns dos mesmos. E por fim é feita uma investigação a linguagens de modelação para requisitos dado que o projeto passa pela criação de uma. Assim é possível visualizar a viabilidade da dissertação de acordo com o alcance geral de desenvolvimento dos três níveis.

3.1 Abordagens de Modelação de WebGIS

Neste tópico vão ser enunciados trabalhos de modelação no contexto de WebGIS. É também realizada a contextualização dos mesmos.

3.1.1 Um Metamodelo para desenho de interface sensível ao contexto

No que diz respeito a trabalhos relacionados com a modelação para o domínio específico de WebGIS, um dos projetos detetados mais relacionados com esta dissertação foi uma proposta de um metamodelo para o desenho de interface de WebGIS sensível ao contexto, a partir de uma visão em MDA (arquiteturas baseadas em modelos).

O estudo [33], levado a cabo por investigadores italianos da Universidade de Roma, crê que em condições de emergência, as aplicações WebGIS sensíveis ao contexto têm de estar preparadas face ao comportamento e desempenho necessários a esse contexto. Há também necessidade de se adaptarem de acordo com a informação e componentes de

sistema ativados por parte dos seus utilizadores. Em caso de emergência a resposta tem de ser dada rapidamente.

Angelaccio et al.[33], através de um caso de estudo sobre as inundações catastróficas de 2002 na Áustria, que resultaram na morte de 9 pessoas e milhares de estragos materiais e económicos, acreditam que um sistema adaptado ao contexto para situações de emergência onde a resposta tem de ser dada rapidamente, através da ajuda das imagens satélite, poderia ter salvo vidas e evitado estragos maiores.

Na Figura 3.1 está ilustrado o metamodelo [34] utilizado pelos investigadores para a realização de transformações de interfaces baseadas ao contexto. Como é possível verificar, o metamodelo que foi concebido para sistemas web, omite as características de interface WebGIS não fazendo qualquer referência às suas propriedades indispensáveis. Deste modo, verifica-se que existem alguns trabalhos à volta do conceito de WebGIS embora superficiais. Este, já foi desenvolvido há mais de dez anos, mas, ainda assim não teve desenvolvimentos posteriores.

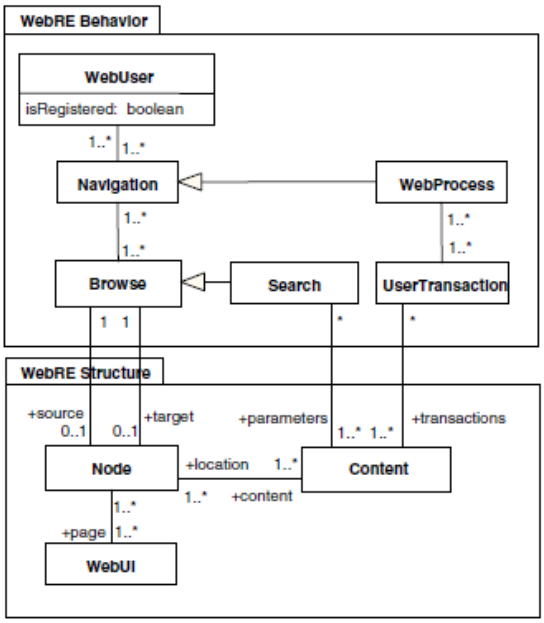


Figura 3.1: Metamodelo de Engenharia de requisitos para sistemas web [34]

3.1.2 Uma abordagem sobre modelação WebGIS orientada a aspectos

Outro trabalho relevante a relacionar com este projeto, foi a análise de observação de uma falha de usabilidade na plataforma WebGIS do Flickr[35]. O projeto levanta a importância da modelação de requisitos em aplicações WebGIS tendo em conta o paradigma dos aspectos.

Inicialmente, o WebGIS nem tinha a funcionalidade de mostrar mais informações acerca de um local selecionado. Acontece que, face à proatividade do crowdsourcing no

Figura 3.2: As informações de um local selecionado atrás do mapa

âmbito de WebGIS, já existiam funcionalidades externas ao Flickr, criadas por outros utilizadores, que permitiam adicionar informações sobre certos locais. O que vemos na Figura 3.2 é uma falta de planeamento na implementação do WebGIS dado que a informação não é pesquisada e apresentada dentro do mapa, mas sim fora. Esta falta obriga a custos elevados de reparação na parte da representação, sendo que esta foi implementada na fase inicial do projeto. A equipa de desenvolvimento envolvida sugere ainda um mapa interno de grandes superfícies como um requisito a ser estudado no futuro dada a complexidade que poderá aumentar num WebGIS a partir desta abordagem (Figura 3.3).

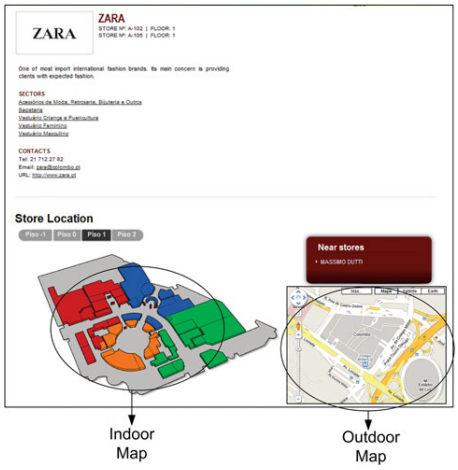


Figura 3.3: Mapas interior e exterior do centro comercial Colombo

Este trabalho de investigação, como os que se apresentam abaixo no domínio do WebGIS, foram os que se encontraram mais próximos do projeto a que nos propomos. Os trabalhos apresentados usam soluções pouco eficazes e desajeitadas para incluir requisitos de interface não disponíveis de raiz nas aplicações. Não foram encontrados projetos especializados na elicitação de requisitos para WebGIS de forma a ajudar e salvaguardar o bom desenvolvimento deste tipo de aplicações. Desta maneira, uma linguagem de modelação de requisitos para WebGIS é necessária, não só para o contributo da união destas

duas áreas, mas também para incentivar um desenvolvimento mais útil nesta relação de dois mundos diferentes.

3.1.3 A abordagem Y-Model

Face às tradicionais metodologias existentes, uma delas até referida anteriormente (Espiral, capítulo 2.3), três investigadores propõem uma nova abordagem no desenvolvimento de WebGIS. A metodologia oferecida designada por Y-Model[36] e à semelhança do ciclo em Cascata, também é composta por nove fases (ver Figura 3.4).

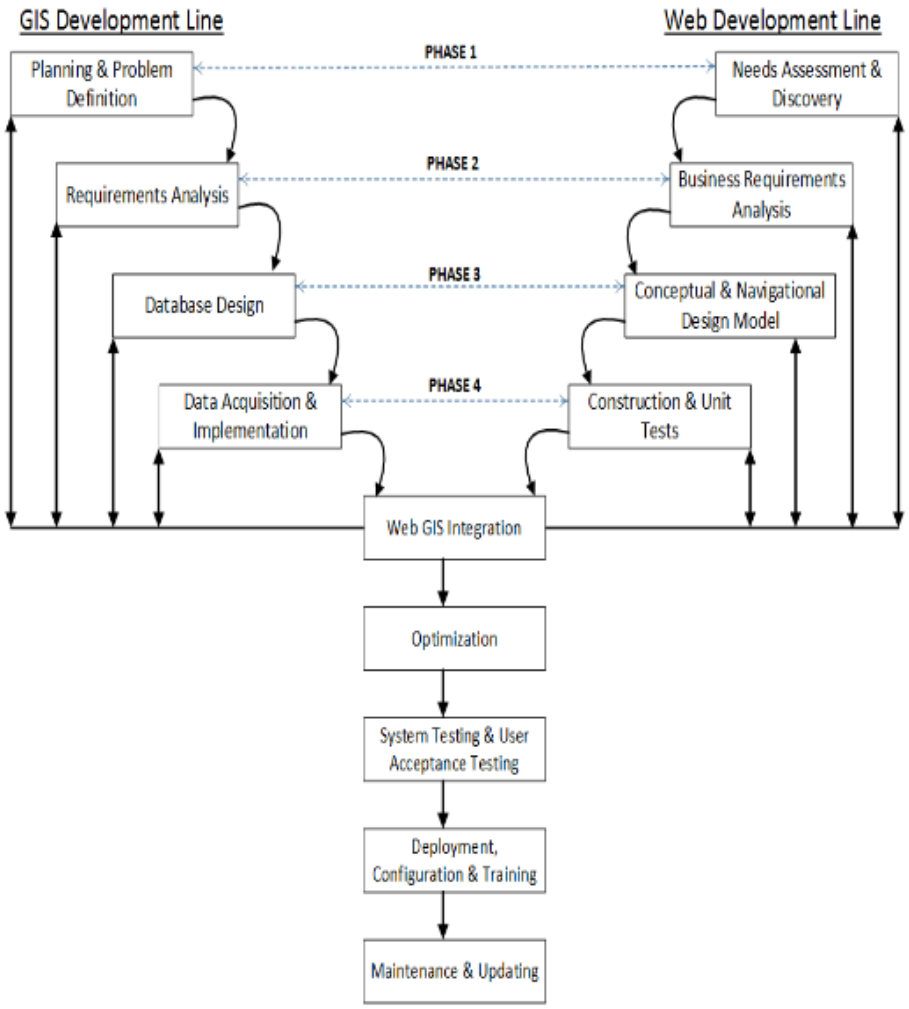


Figura 3.4: Metodologia Y-model

À primeira vista parece não haver diferenças entre as duas metodologias, ainda assim, a proposta Y-Model é mais focada no âmbito comercial, e tende a definir os requisitos do WebGIS de forma mais objetiva, tentando assim acelerar o processo de desenvolvimento.

Em concordância com as metodologias tradicionais de desenvolvimento de software

(Cascata, Espiral, Prototipagem ou design de sistemas estruturados) esta abordagem Y-Model utiliza a abordagem cascata e SDLC. Ambas são aplicadas tanto no desenvolvimento da parte do SIG como na parte da Web. A grande diferença do Y-Model face ao desenvolvimento comum de sistemas de informação, é que esta abordagem inicia o processo de desenvolvimento de uma aplicação WebGIS em separado. Há uma abordagem inicial para a parte SIG em particular e existem também uma abordagem de desenvolvimento para a parte Web, também esta em particular. O Y-Model é composto por quatro fases, sendo que a primeira é o planeamento e definição de objetivos tanto para a parte de SIG como WEB, a segunda fase é para o levantamento de requisitos, a terceira diz respeito ao design e a quarta a implementação. Após a conclusão destas fases, ocorre uma integração da parte do SIG com a WEB procedendo-se à fase de testes. Como é visível na imagem, todo este processo "escoa" na forma da letra Y, daí ter sido atribuído o nome Y-Model.

Este projeto é relevante para esta dissertação, porque levanta a questão dos métodos de desenvolvimento que podem ser aplicados na criação de aplicações WebGIS. Esta dissertação pode muito bem ser útil para a validação desta metodologia. Num caso prático de modelação de requisitos para WebGIS, podem ser definidos todos os requisitos. Se depois se aplicar esta metodologia de desenvolvimento e os requisitos forem todos cumpridos é porque a abordagem é viável.

3.1.4 A abordagem WebML

Sobre linguagens de domínio específico de WebGIS, destaca-se um projeto de origem na Universidade de Salerno em Itália, muito focado nesta área. Sugerindo uma linguagem visual baseada em WebML, para a construção do design de WebGIS, os autores desta proposta[37], acreditam que dada a importância destes sistemas noutros contextos que vão desde o planeamento urbano ao marketing e prevenção de desastres naturais, uma linguagem de modelação web seria uma forte ajuda para entidades não especializadas na implementação destas aplicações.

Esta abordagem, procura modelar interações relevantes e operações de navegação comuns de aplicações WebGIS. É um projeto que se concentra mais na parte interativa da aplicação WebGIS do que no seu sistema como um todo. Sendo que não chega inteiramente para garantir todos os requisitos de uma aplicação WebGIS. É assim mais um trabalho relacionado que nos motiva a levantar esta dissertação.

3.1.5 Discussão

Como foi possível verificar nos projetos investigados acima, todos eles na sua maioria são à volta do desenvolvimento de WebGIS e do desenvolvimento da interface de WebGIS. Porém no que toca a modelação de requisitos de interface para WebGIS, não foram localizados projetos de utilidade para a dissertação. Esta lacuna constitui mais um motivo para o levantamento deste projeto de modelação de requisitos para WebGIS. O tópico 3.2

nasce fruto desta ausência de projetos e vem destacar algumas funcionalidades comuns de aplicações WebGIS.

3.2 Projetos WebGIS

Para identificar funcionalidades e características de interfaces de aplicações WebGIS foram analisados um conjunto de projetos WebGIS, alguns dos quais desenvolvidos no NOVA LINCS. Naturalmente, esta análise não é exaustiva, sendo até bastante baseada em trabalhos desenvolvidos localmente, mas serviu como ponto de partida.

- **Litescape**[38] - É um WebGIS português criado com o propósito de apresentar geografia literária a partir de obras, temas ou nomes de autores.
- **LxConventos**[39] - É outro WebGIS nacional que foi desenvolvido para estudar o impacto de casa religiosas de Lisboa.
- **Estruturas Costeiras**[40] - Foi um projeto realizado no âmbito de uma tese de mestrado e parceria com o LNEC para observação e análise de estruturas costeiras do território Português.
- **Fogos.pt**[41] - É um projeto cujo o seu objetivo é transmitir informação acerca de incêndios e respetivos estados, em tempo real no território nacional.
- **Mapa de cabos submarinos**[42] - É um projeto internacional, que apresenta grande parte dos cabos de fibra ótica submarinos utilizados para telecomunicações.

3.2.1 Discussão de propriedades em comum em projetos WebGIS

Após o estudo e análise de cinco WebGIS com propósitos diferentes apresentados na secção anterior, levantaram-se algumas características que são comuns entre eles.

- **Ferramenta zoom** - Fornece a habilidade ao utilizador de detalhar mais ou menos a visualização do mapa que está a utilizar.
- **Nomes dos locais (etiquetas)** - Servem como uma base de orientação para o utilizador saber onde se encontra no mapa que está a visualizar.
- **Caixa de pesquisa** - Permitem ao utilizador procurar por algum local específico.
- **Marcadores ou áreas delimitadas** - São utilizados para sinalizar locais de alguma relevância no contexto do WebGIS em que estão a ser aplicados.
- **Janelas de informação** - Transmitem informações sobre os locais ou áreas seleccionadas.
- **Integração de dados** - Integração de vários conjuntos de dados no mapa que podem ser geridos como categorias diferentes ou camadas diferentes.

- **Full Extent** - Capacidade de adaptação do nível de detalhe de forma a visualizar todos os conjuntos de dados visiveis.

- **Bases de dados distribuídas** - Integração com bases de dados distribuídas, de forma a poder consultar informação externa (de outro servidor ou base de dados) relativa a um determinado objeto selecionado no mapa.

- **Edição de objetos** - Possibilidade de inserção, alteração ou remoção de novos objectos ou conjuntos de dados.

- **Localização Atual** - Localização do utilizador apresentada no mapa (GPS ou outra forma de localização)

- **Pontos de interesse (POI)** - Identificação de proximidade entre o utilizador e outros pontos de interesse na aplicação.

Na Tabela 3.1 é representada uma comparação feita entre os diferentes WebGIS acima falados e estudados sobre as funcionalidades de cada um deles. Esta avaliação, permitiu definir conceitos da linguagem capazes de modelar as funcionalidades que são comuns em aplicações WebGIS.

Tabela 3.1: Comparação de funcionalidades WebGIS

	Projectos				
PROPRIEDADES	Litescape	Lx Conventos	Estruturas Costeiras	Fogos	Submarine Cable Machine
Zoom	SIM	SIM	SIM	SIM	SIM
Nomes dos Locais	SIM	SIM	NÃO	SIM	NÃO
Marcadores	SIM	NÃO	SIM	SIM	SIM
Áreas	SIM	SIM	SIM	NÃO	NÃO
Clusters	SIM	NÃO	SIM	NÃO	NÃO
Janelas de Informação	SIM	SIM	SIM	SIM	SIM
Caixa Pesquisa	SIM	SIM	SIM	NÃO	SIM
Integração dados	SIM	SIM	SIM	SIM	SIM
Full extent	SIM	SIM	SIM	NÃO	SIM
BDs distribuídas	SIM	SIM	SIM	SIM	SIM
Edição de objetos	NÃO	NÃO	SIM	NÃO	NÃO
Localização atual	SIM	NÃO	SIM	SIM	NÃO
POIs	NÃO	NÃO	SIM	NÃO	NÃO

3.3 Linguagens de Modelação de domínio específico para requisitos

3.3.1 A linguagem VEL

Um dos projetos sob este tema que mais interesse suscitou durante esta investigação foi o trabalho do ex-aluno da Faculdade Ciências e Tecnologias da Universidade Nova de Lisboa, Gonçalo Migueis [43], que realizou uma dissertação acerca de uma linguagem de modelação de requisitos emocionais na indústria de videojogos. O tema em si não está relacionado com WebGIS, mas ainda assim no que diz respeito à criação de uma linguagem de modelação de requisitos está muito próximo do que se pretende nesta dissertação. A linguagem em questão tinha o propósito de reunir os requisitos necessários para implementar um videojogo que despoletasse emoções planeadas pelos seus criadores.

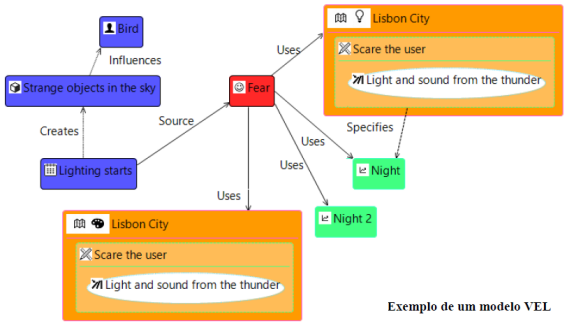


Figura 3.5: Instância do Metamodelo da DSML, adaptado de [43]

No exemplo da Figura 3.5, o objetivo é especificar a emoção medo ao jogador. Para tal, deverá ser criado um cenário onde estejam envolvidos os fatores: noite, luzes e sons de relâmpagos, que para além de fazer agitar os pássaros do cenário do jogo cria um cenário assustador ao espectador que está a visualizar o jogo. Esta linguagem de domínio específico para elicitação de requisitos, num contexto de videojogos foi desenvolvida sob o metamodelo Ecore. O facto dos seus autores estarem relacionados com a nossa instituição, levou a um destaque maior no nosso processo de investigação de trabalhos relacionados. Após a sua observação achámos este projeto útil e interessante e de alguma forma contribuiu para o planeamento do nosso trabalho.

3.3.2 Abordagem MDGore

MDGORE, é uma framework [44] cujo o seu principal objetivo é transformar modelos i* em modelos KAOS utilizando regras definidas em ATL[45].

A aplicação das regras de transformação é feita ao nível da sintaxe abstracta definida nos metamodelos de cada framework, utilizando a técnica de transformação por grafos de MDD. Esta framework permite relacionar as duas linguagens de modelação de forma a facilitar a conversão de modelos quando necessária em diversos ambientes. Os metamodelos criados foram desenvolvidos a partir do Ecore e em conformidade com o seu

metamodelo.

Na Figura 3.6 é possível observar uma visão geral do processo da abordagem MDGore. É de salientar que o processo de transformação apresentado nesta imagem, respeita a terminologia da transformaçao de modelos que foi apresentada na Figura 2.11 da secção 2.4.3 do capítulo anterior.

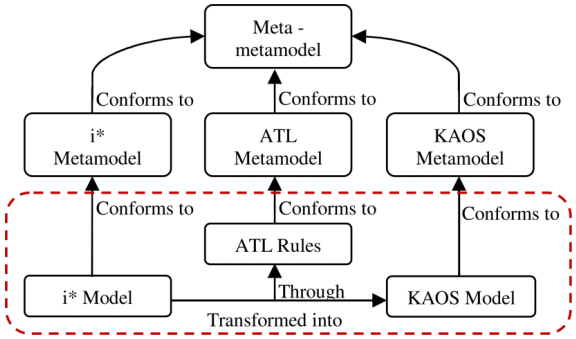


Figura 3.6: Processo MDGore[44]

3.3.3 Geração de modelos orientados a objetivos a partir de requisitos criativos através de MDE

Uma abordagem proposta por Fernando Wanderley e João Araújo propõem a geração de modelos KAOS goal a partir de mind maps [46], utilizando as técnicas de MDE. Os autores levantam um estudo [47] conduzido pela Standish Group onde 66% dos softwares não alcançam as expetativas dos utilizadores em relação às suas funcionalidades e modos de operação. Esta abordagem, fornecerá uma compreensão mais aprimorada por parte dos Stakeholders e um processo de modelação de objectivos sistemático, dando suporte à engenharia de requisitos tendo em conta os elementos do modelo de objetivos do KAOS. Os Metamodelos utilizados neste projeto foram definidos a partir do Ecore e a execução das regras de transformação utilizam ATL.

Na Figura 3.7 é possível observar uma visão do processo desta abordagem e mais uma vez é possível relacionar este processo de geração com a terminologia apresentada no tópico de transformação de modelos do Capítulo 2.

3.3.4 NDT, Abordagem orientada a modelos para requisitos Web

Navigational Development Techniques(NDT) é uma abordagem orientada a modelos para o âmbito de requisitos e análise de fases de desenvolvimento na Web [48]. Com base no paradigma de MDE, a abordagem NDT é composta por três fases: captura, definição e validação de requisitos. Tudo começa na definição dos objetivos, os requisitos são definidos de acordo com a sua natureza (requisitos de informação de armazenamento, agentes, requisitos, requisitos funcionais, requisitos de interação ou requisitos não funcionais). Quando os requisitos estiverem validados, o processo NDT prosegue definindo três modelos. Modelo conteúdo (class diagram), modelo de navegação (que reflete como os utilizadores

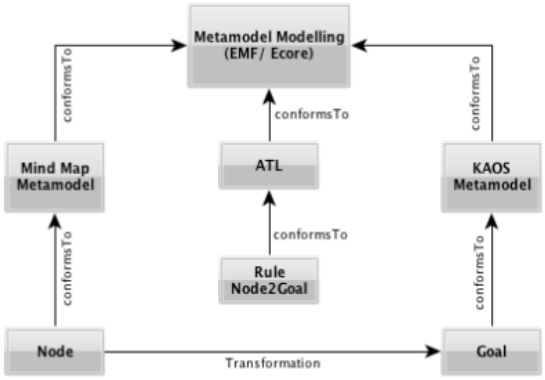


Figura 3.7: Processo de geração de KAOS Goal models a partir de mind maps [46]

podem navegar pelo sistema) e o modelo de interface abstrata (mostra a interface abstrata do sistema). Um dos principais contributos desta abordagem é que ela oferece uma forma sistemática de obter modelos de análise a partir dos requisitos. NDT oferece ainda transformações que indicam como cada modelo deve ser obtido a partir da definição de requisitos. Os modelos, apesar de serem diferentes, estão relacionados uns com os outros, dado que cada um deles representa uma parte diferente do sistema, mas, ainda assim, são independentes devido ao facto de cada um poder ser obtido independentemente dos requisitos. Quando a equipa de desenvolvimento aplica as tranformações, obtêm os modelos básicos. Modelo conteúdo, Navegação e Interface Abstrata.

O principal objetivo da NDT é precisamente oferecer processos sistemáticos para construir modelos que outras metodologias Web possam usar como ponto de partida do seu processo de desenvolvimento e garantir a qualidade destes modelos.

Esta abordagem é útil para este projeto, pois destaca a importância do metamodelo tanto a curto como a longo prazo. Num periodo breve o metamodelo define as propriedades e operações de uma linguagem, no futuro pode ser a chave para transformação de modelos.

3.3.5 Engenharia de Requisitos orientada a modelos para desenvolvimento de sistemas embebidos

O projeto descrito em [49] direccionado para os sistemas embebidos, procura realizar uma análise de objetivos deste tipo de sistemas, com base na sua arquitetura, requisitos não funcionais e suas propriedades e verificação da satisfação de objetivos de acordo com as transformações resultantes de modelo para modelo.

Esta abordagem oferece três contributos. Inicialmente introduz uma extensão da linguagem de definição e análise de requisitos (RDAL) com o objetivo de especificar requisitos não funcionais e a sua relação ao modelo de arquitetura do sistema.

Depois também extende a framework que implementa refinamentos de arquitectura

com transformações modelo para modelo. Esta extensão permite o impacto das tranformações de modelos disponíveis em cada propriedade dos requisitos não funcionais do sistema a ser descrita.

Por fim é proposta a utilização de um método de análise de satisfação de objetivos para verificar a correção do refinamento da arquitetura que foi implementada resultante de uma transformação de modelos.

É uma abordagem muito interessante, pois aplica Engenharia de Requisitos e desenvolvimento de software orientado a modelos em sistemas altamente complexos (no artigo é utilizado o caso de estudo de um pacemaker) e altamente intolerantes à falha, pois qualquer falha pode ser fatal, elevando assim a útilidade destes conceitos em sistemas sensíveis à vida humana.

3.4 Sumário

Ao longo do Capítulo 3, foi apresentado o resultado da investigação sobre Modelação WebGIS, Projetos WebGIS, Linguagens de Modelação de Domínio Específico e Engenharia de Requisitos orientada a modelos, para transmitir o estado de arte do progresso nestas áreas, alimentando ainda o desejo de contribuir com este projeto para uma área que ainda está em expansão. Os trabalhos estudados, contribuíram para o desenvolvimento da linguagem, na medida em que foram definidos alguns conceitos, úteis para modelar. Conceitos como domain concept, builtin, function, event e resource foram definidos para integrar a linguagem a partir da análise feita com base nos trabalhos relacionados com o tema esta dissertação. Por outro lado, os SIG que há muito vieram para ficar, começam também a construir um caminho nas linguagens de domínio específico. No próximo capítulo apresentamos a linguagem para modelação de requisitos de interface WebGIS.

Capítulo 4

Linguagem para modelação de requisitos de interface WebGIS

Neste capítulo é apresentado todo o trabalho desenvolvido para levantar a solução do problema. A linguagem para modelação de requisitos de interface WebGIS e o editor para modelar os requisitos. São destacados todos os conceitos da linguagem e todas as tecnologias utilizadas para a construção do editor. É também explicada a funcionalidade tanto dos conceitos como das tecnologias.

4.1 Conceitos da linguagem

O objetivo da linguagem é definir os requisitos para a implementação de interface de aplicações WebGIS, por pessoas não especializadas. A definição final destas propriedades deverá fazer parte do modelo criado pelo utilizador da linguagem com os requisitos desejados. Estes requisitos vão ser modelados a partir dos conceitos existentes na linguagem que fazem parte do seu metamodelo. De seguida são apresentados os conceitos que serão representados por metaclasses do metamodelo:

- **Terminal —** Esta metaclasse indica o tipo de terminal desejado para a instalação e utilização da aplicação WebGIS. (ex: Desktop, Smartphone, Tablet).

- **Domain —** Esta metaclasse indica qual o domínio no qual se foca a aplicação WebGIS, isto é, a familia dos conceitos identificados e pretendidos na aplicação.

- **Concept —** Esta metaclasse indica quais são os principais dados que se pretende filtrar do domínio escolhido para a aplicação. Define os conceitos subjacentes ao domínio.

- **Map —** Esta metaclasse indica o mapa base que se pretende utilizar na aplicação. (ex: OpenStreetMap ou googleMaps). O mapa base será aquele que servirá de fundo às camadas de dados que vão ser utilizadas no contexto da aplicação.

- **Map Layer —** Esta metaclasse diz respeito às camadas de dados, pretendidas pelo utilizador que vão integrar o mapa. Aqui, é onde será realizada a abstração dos dados inicialmente planeados para a aplicação (Concepts) em dados georeferenciados (map layers) no mapa base.

- **Builtin —** Esta metaclasse representa funcionalidades gerais da aplicação. Estas funcionalidades podem ser relacionadas com o mapa, terminal com outras funcionalidades. (ex: p/Terminal: relógio; p/Mapa: marcador).

- **Event —** Esta metaclasse identifica eventos que dão suporte à utilização da aplicação (ex: click; scroll). Estes eventos permitem também despoletar outras funcionalidades da aplicação WebGIS.

- **Function —** Esta metaclasse, tal como o Builtin, é apontada à aplicação em geral. É objetivamente para apontar comportamentos de funcionamento do sistema. Estes comportamentos podem estar relacionado com builtins, eventos, recursos, mapa e terminal (ex: p/terminal: getSystemTime(); p/mapa: InputData; DeleteData).

- **Resource —** Esta metaclasse diz respeito a repositórios onde se vão guardar os dados do sistema, do mapa ou outros necessários ao correto funcionamento da aplicação (ex: Base de dados dos dados espaciais, ficheiros csv, xml, json).

As relações entre conceitos, que têm a finalidade de interligar as metaclasses apresentadas acima que constituem o modelo total são:

- **Terminal-Domain —** Faz a ligação do tipo de interface da aplicação WebGIS com o domínio da aplicação que se está a modelar (ex: Terminal: Desktop; Domínio de aplicação: Restauração; Quer isto dizer que, a aplicação WebGIS que estamos a modelar será sobre restauração e será uma aplicação concebida para dispositivos Desktops).

- **Terminal-Map —** Relaciona o tipo de interface da aplicação com o mapa base que será utilizado para desenvolver a WebGIS (ex: Terminal: Desktop; Mapa: OpenStreetMap; Quer isto dizer que, a aplicação WebGIS concebida para utilização em Desktops, utilizará o Mapa base: OpenStreet Map).

- **Terminal-Builtin —** Relaciona o tipo de interface com uma funcionalidade geral (ex: Terminal: Desktop; Builtin: Calendário; Significa que, a aplicação WebGIS, utilizará uma funcionalidade específica (um calendário) à parte do Mapa base).

- **Terminal-Function —** Representa uma relação entre o tipo de interface e uma função (ex: Terminal: Desktop; Function: Escurecer a aplicação quando forem dezanove horas; Reflete que, a aplicação WebGIS, utilizará uma função específica para reduzir a sua luminosidade, à parte do Mapa base).

- **Terminal-Event —** Representa uma relação entre o tipo de interface e um evento (ex: Terminal: Desktop; Evento: Alerta despoletado às dezanove horas a informar que a aplicação vai diminuir a luminosidade para operar de forma mais sensível ao ambiente exterior; Reflete que, a aplicação WebGIS, utilizará um evento específico à parte do Mapa base).

- **Domain-Concept —** Relaciona o domínio com o/os conceito/os do domínio. Os conceitos do domínio, são os dados subjacentes ao âmbito da aplicação que se pretende extrair (ex: Domínio: Hotelaria; Conceito de domínio: Hoteis; ou Conceito de domínio: Pousadas; Quer isto dizer que, do Domínio definido, neste caso a Hotelaria, vão ser levantados para apresentar informação, apenas os hotéis).

- **Concept-Map Layer —** Representa a abstração dos conceitos subjacentes ao domínio em dados georeferenciados a partir de camadas no mapa, produzindo informação ao utilizador (ex: Conceito de domínio: Hotéis; Camada do mapa: Marcadores; Os hotéis serão representados no mapa a partir de marcadores).

- **Map-Map Layer —** Demonstra a ligação entre o mapa base da aplicação e a camada do mapa que serve para transformar, em dados georeferenciados, os conceitos do domínio (ex: Mapa base: OpenStreetMap; Camada do mapa: Marcadores; No mapa base vão existir dados (ex: restaurantes) representados em forma de marcadores).

- **Map-Builtin —** Representa a ligação entre o mapa base da aplicação e uma funcionalidade presente no mapa (ex: Mapa base: OpenStreetMap; Funcionalidade: Marcadores; O mapa possui a funcionalidade marcadores, independentemente destes representarem algum tipo de informação).

- **Map-Function —** Representa a ligação entre o mapa base e uma função do mapa ou presente no mapa (ex: Mapa: OpenStreetMap; Function: ApiLicenseKey(); O mapa OpenStreetMap requer a utilização da chave de licença da API da google).

- **Map-Event —** Representa a ligação entre o mapa base e um evento que ocorre ou pode ocorrer dentro do mapa (ex: Mapa: OpenStreetMap; Evento: Scroll; Quer isto dizer que o scroll com o ponteiro do rato em cima do mapa, é um evento que pode depoletar funcionalidades ou funções).

- **Builtin-Builtin —** Representa a ligação de um Builtin com outro (ex: Builtin: Marcador; Builtin: Janela de informação; Um marcador, que representa um dado georeferenciado, pode possuir uma janela de informação específica visível ao utilizador).

- **Builtin-Event —** Representa a ligação entre um Builtin e um Evento (ex: Builtin: Marcador; Evento: Click; O evento Click no marcador produzirá um outro efeito, que pode ser outro Builtin ou uma função). Nesta relação, o evento é despoletado a partir do builtin ou no builtin.
- **Builtin-Function —** Representa a ligação entre um Builtin e uma função (ex: Builtin: Marcador; Function: AlteraCorDoMarcador(); Consoante uma instrução x, o Marcador y altera a sua cor). Nesta relação, a função é despoletada a partir do builtin ou no builtin.
- **Event-Builtin —** Representa a ligação entre um evento e um Builtin (ex: Event: Click; Builtin: Marcador; Indica que ao fazer click será despoletado um marcador).
- **Event-Function —** Representa a ligação entre um evento e uma função (ex: Event: Scroll; Function: RenderZoom(); Indica que mediante o evento scroll in/out, será feito um Zoom onde está o ponteiro do rato). Nesta relação, a função é despoletada a partir do evento ou no evento.
- **Function-Function —** Representa a ligação entre uma função e outra função (ex: Function: AlertarUtilizadorModoNocturno(); Function: ReduzirLuminosidade(); Indica que por exemplo quando forem dezanove horas, é lançado um alerta ao utilizador de que a aplicação irá operar em modo nocturno e de seguida é lançada outra função para reduzir a luminosidade do sistema).
- **Function-Builtin —** Representa a ligação entre uma função e um Builtin (ex: Function: OcultarMarcadores(); Builtin: Marcadores; Indica que uma função vai ocultar os marcadores). Nesta relação, o builtin é despoletado a partir da função ou na função.
- **Function-Event —** Representa a ligação entre uma função e um evento (ex: Function: ManutençãoÀs3daManha(); Event: PopupInformativo; Indica uma função de manutenção de sistema que lança uma janela de informação em forma evento a avisar o utilizador de que se vai realizar manutenção do sistema). Nesta relação, o evento é despoletado a partir da função ou na função.
- **Function-Resource —** Representa a ligação entre uma função e um recurso (ex: Function: InserirDadosNaBaseDados(); Resource: baseDadosX; Indica que se vão inserir um conjunto de dados na base de dados baseDadosX que é representada no modelo em forma de recurso).

4.2 Descrição da DSML

A DSML WebGIS IRML, foi desenvolvida a partir do Metamodelo e do Editor.
Nesta secção vamos explorar estas duas partes da linguagem, começando pelo Metamodelo, responsável pelas características e operações da mesma, que depois são refletidas a

partir do funcionamento do editor na modelação de interfaces de aplicações WebGIS.

4.2.1 Metamodelo

Tendo em conta os conceitos capturados pelas metaclasses descritas na secção anterior, compomos o metamodelo da nossa linguagem.

Infelizmente devido à sua grande dimensão foi necessário "dividi-lo"para aumentar o seu detalhe, transmitindo uma imagem mais nítida ao leitor. Aqui estão inseridas todas as classes e metaclasses de forma interligada.

Na Figura 4.1, podemos observar uma parte central do diagrama. Foquemo-nos na classe **interfaceMap**. Esta, é estendida a partir da classe **Interface**, classe essa que abstrai toda a interface WebGIS e a partir da qual se definem os conceitos da interface da aplicação que estamos a modelar. Através de uma ligação (link) denominada por **hasMap** verificamos que para cada Interface existe apenas um mapa. É também visível que interfaceMap possui um único atributo do tipo String.
Nas classes e relações apresentadas no tópico anterior é verificável que existem quatro relações com o Mapa, porém nesta imagem só estão visíveis duas relações. Mapa com Evento através de uma metaclasse **interfaceMap_interfaceEvent** e Mapa com Função, com a metaclasse **interfaceMap_interfaceFunction**. Podemos observar que, a ligação entre a interfaceMap com a metaclasse interfaceMap_interfaceEvent é realizada através do link **map_meta_line** e a conexão entre a metaclasse e a classe interfaceEvent é feita a partir do link **meta_event_line**.
Além das classes, todas as metaclasses também são estendidas pela classe principal Interface. No caso desta metaclasse interfaceMap_interfaceEvent, podemos notar que, ao ser estendida pela classe principal Interface, é indicado no link **map_event** que para cada Interface podem existir várias metaclasses interfaceMap_interfaceEvent. Quer isto dizer que, o Mapa pode ter vários Eventos, mas só pode existir uma única ligação entre cada relação Mapa com Evento, pois os links destas classes com a respetiva metaclasse que as relaciona, traduzem um limite de 0 para 1.

Mudando o foco para a classe Terminal, na Figura 4.2 observamos a classe **interfaceTerminal** com várias ligações, entre elas uma com a classe **Terminal**, esta, diz respeito ao tipo ao qual pertence a interfaceTerminal. As classes tipo, são classes abstratas, servindo apenas para projectar um modelo a herdar pelas classes operacionais. Mais à frente serão apresentadas outras classes desta categoria.
É também visível que a interfaceMap está também ela conectada ao tipo Map (o que não era possível na imagem anterior).
Terminal diz respeito ao tipo de interface da aplicação que estamos a modelar (PC,

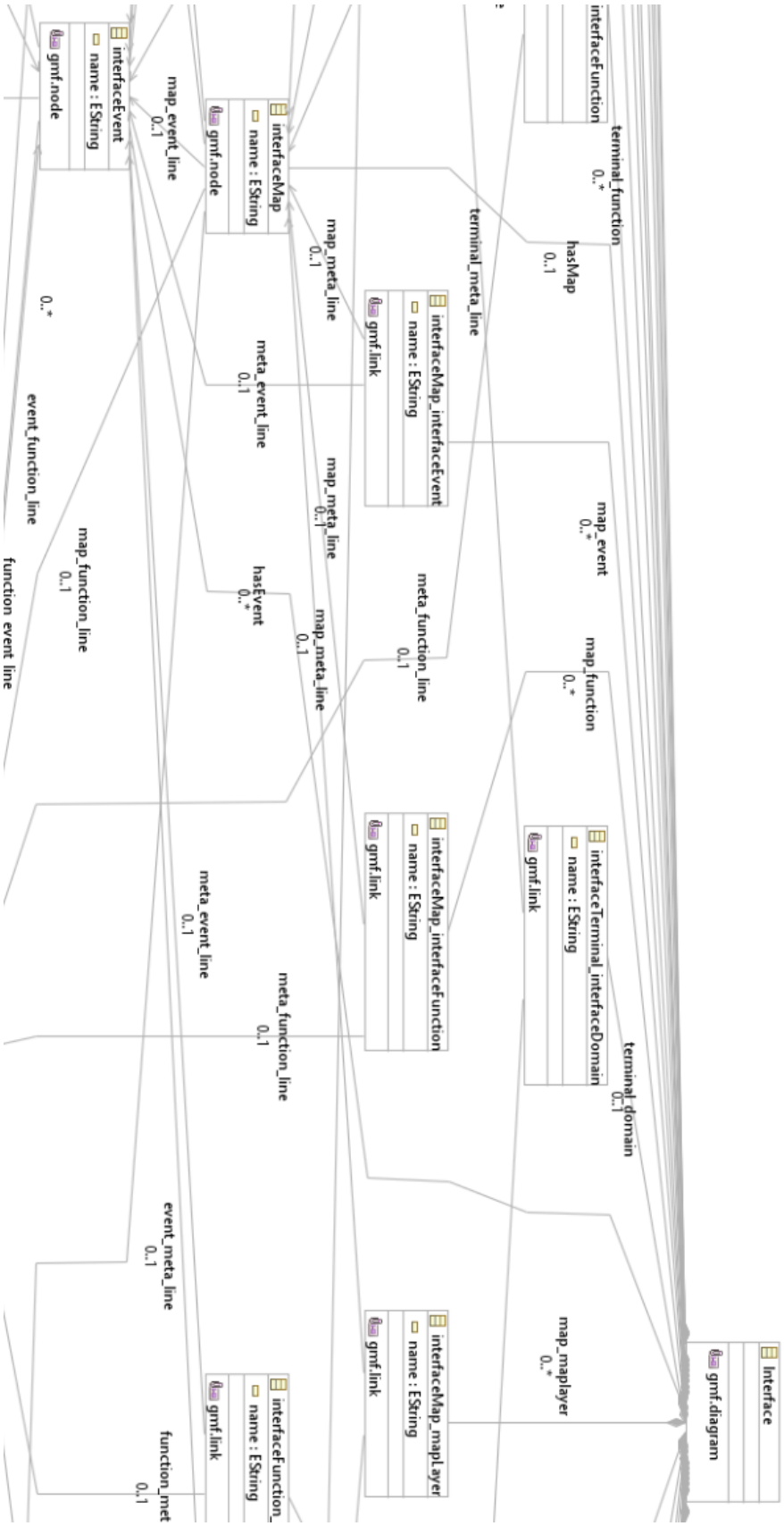


Figura 4.1: Uma parte do metamodelo, foco na classe Map

Smartphone, Tablet) daí possuir várias relações, sendo que algumas delas não estão visíveis nesta imagem. Ainda assim, podemos verificar a relação entre interfaceTerminal e interfaceMapa. Esta relação nasce a partir da metaclasse **interfaceTerminal_interfaceMap**. A classe **interfaceMap**, à semelhança da classe **interfaceTerminal** que se relaciona com **Terminal**, está também ligada a uma classe tipo, neste caso a classe **Map**.
É também possível detetar, na mesma figura, que o Terminal se relaciona com **Eventos** e **Funções**. A primeira através da metaclasse **interfaceTerminal_interfaceEvent** que se encontra à esquerda da **interfaceTerminal**, e a segunda a partir da **interfaceTerminal_interfaceFunction**, à direita da **interfaceTerminal**. A classe **interfaceEvent** possuí o tipo **Event** como também se pode ver na imagem.
Apesar de não ser visível a classe principal Interface, existe uma ligação na parte superior, **hasTerminal**, que é proveniente de lá. Nesta ligação é verificado que a Interface só pode ter um Terminal, isto indica que, no caso do utilizador querer modelar a sua aplicação para computador e para smartphone, ele terá de instanciar dois modelos, um para cada tipo de Terminal.

Regressando à zona central do Metamodelo, olhamos agora a partir da Figura 4.3, para a classe Function ou tecnicamente **interfaceFunction**.

Conseguimos encontrar, pelo menos quatro relações. Uma com **interfaceMap**, duas relações com **interfaceEvent**, e outra com a própria **interfaceFunction**. As duas relações com a classe Evento são propositadas, porque na verdade, o que está a ser definido é que uma função pode originar um evento ou um evento pode resultar numa função. Na relação com o Mapa, a função é dependente do Mapa e não o contrário. Por último, temos a relação Function-Function, onde esta vai permitir ao utilizador modelar uma função que é despoletada por outra função.

Todas estas relações são visíveis de forma práctica na modelação de interfaces de aplicações WebGIS. No próximo capítulo, é ilustrado de uma forma mais clara o que o metamodelo realmente permite relacionar de modo a conseguir uma modelação completa de uma interface WebGIS.

4.3 Editor —WebGIS IRML: WebGIS Interface Requirements Modeling Language

A implementação do editor passou pela utilização de algumas tecnologias e criação de vários ficheiros para suporte das mesmas.

Nesta secção, serão abordadas as tecnologias principais, que foram indispensáveis ao levantamento e bom funcionamento do editor, de forma a conseguir atingir o objetivo que foi definido inicialmente.

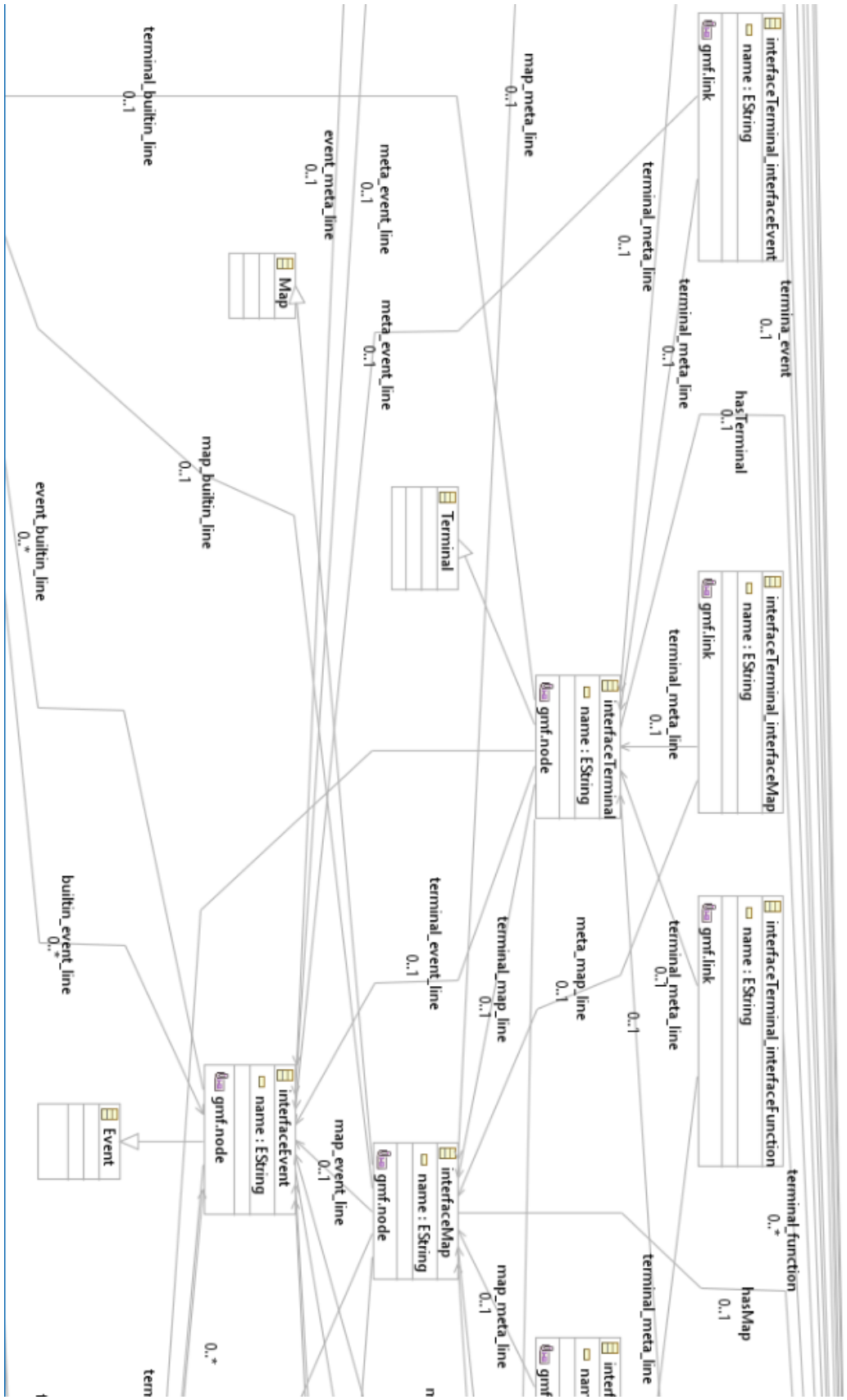


Figura 4.2: Uma parte do metamodelo, foco na classe Terminal

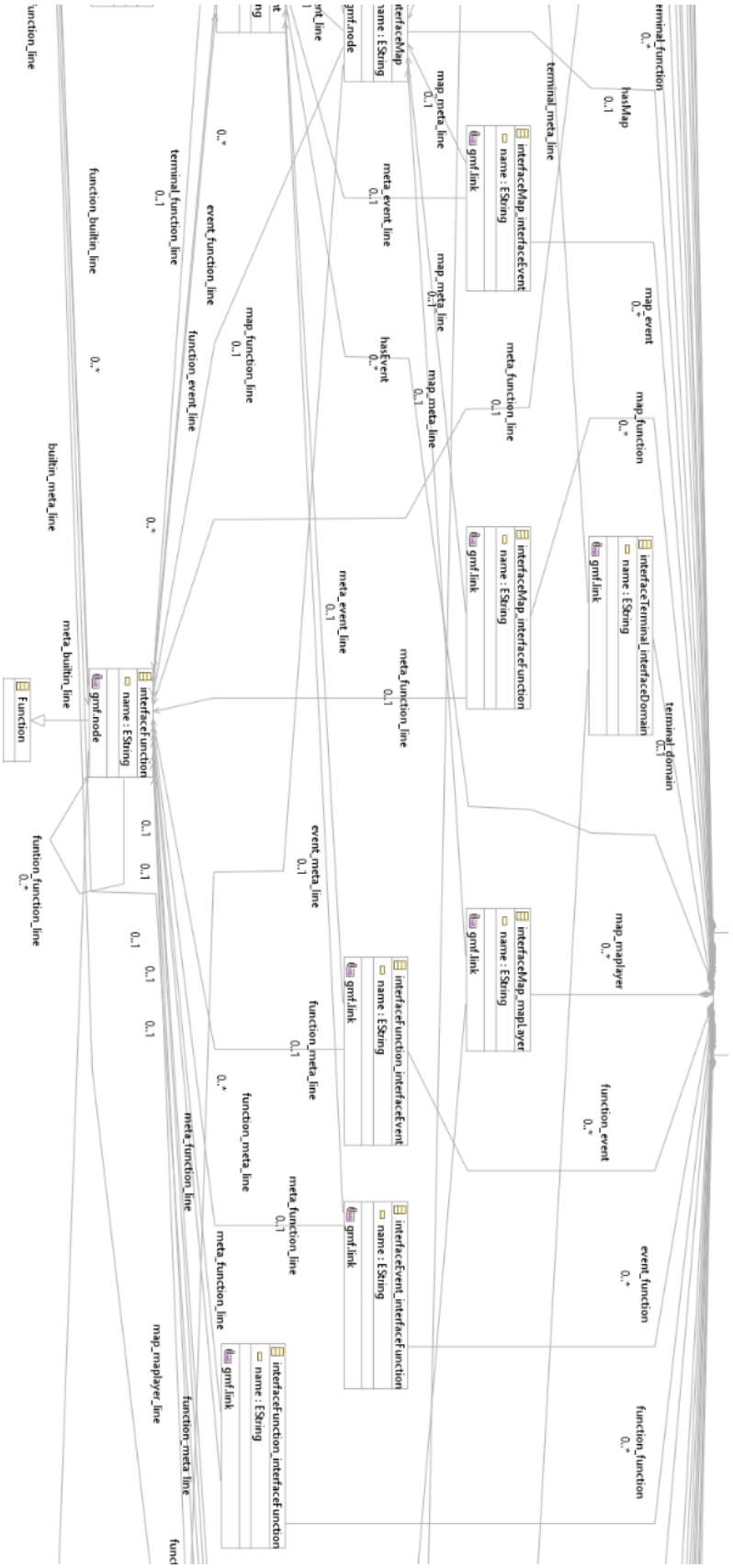


Figura 4.3: Uma parte do metamodelo, foco na classe Function

4.3.1 EMF

Para a criação e combinação de todas as classes e metaclasses destacadas na secção anterior recorreu-se à criação de um ficheiro EMF[1] para o levantamento de todas elas. Este processo podia ter sido realizado através de um ficheiro Ecore, sendo depois feita a geração do ficheiro emf. Ainda assim fez-se do modo inverso que permite à mesma a geração de um outro ficheiro na direção contrária (ecore). Na Figura 4.4 podemos observar um excerto do código do emf onde são declaradas as classes e metaclasses do metamodelo.

```
@gmf
@namespace(uri="http://myTese/1.0", prefix="myTese")
package myTese;
@gmf.diagram
class Interface {
  val interfaceTerminal hasTerminal;
  val interfaceDomain hasDomain;
  val interfaceMap hasMap;
  val interfaceConcept[*] hasConcept;
  val mapLayer[*] hasLayer;
  val interfaceBuiltIn[*] hasBuiltIn;
  val interfaceFunction[*] hasFunction;
  val interfaceEvent[*] hasEvent;
  val interfaceResource[*] hasResource;
  //metaclasses
  val interfaceTerminal_interfaceDomain terminal_domain;
  val interfaceTerminal_interfaceMap terminal_map;
  val interfaceTerminal_interfaceBuiltIn terminal_builtin;
  val interfaceTerminal_interfaceFunction terminal_function;
  val interfaceTerminal_interfaceEvent  termina_event;
  val interfaceDomain_interfaceConcept[*] domain_concept;
  val interfaceConcept_mapLayer[*] concept_maplayer;
  val interfaceMap_mapLayer[*] map_maplayer;
  val interfaceMap_interfaceBuiltIn[*] map_builtin;
  val interfaceMap_interfaceFunction[*] map_function;
  val interfaceMap_interfaceEvent[*] map_event;
  val interfaceBuiltIn_interfaceBuiltIn[*] builtin_builtin;
  val interfaceBuiltIn_interfaceFunction[*] builtin_function;
  val interfaceBuiltIn_interfaceEvent[*] builtin_event;
  val interfaceEvent_interfaceBuiltIn[*] event_builtin;
  val interfaceEvent_interfaceFunction[*] event_function;
  val interfaceFunction_interfaceFunction[*] function_function;
  val interfaceFunction_interfaceBuiltIn[*] function_builtin;
  val interfaceFunction_interfaceEvent[*] function_event;
  val interfaceFunction_interfaceResource[*] function_resource;
}
```

Figura 4.4: Declaração de metaclasses no Emf

Há que referir que todas as metaclasses possuem um género de um prefixo "interface"que antecede o seu nome. Na altura da criação destas ferramentas, foram pensadas

[1] https://www.eclipse.org/modeling/emf/

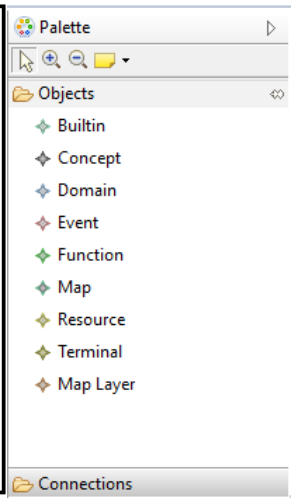
como pertencentes a um WebGIS. Não existindo um nome específico de aplicação WebGIS, dado que o objetivo é possibilitar a criação de vários modelos para vários WebGIS tentou criar-se um nome genérico para a aplicação em construção. Daí, interface.

4.3.2 EOL

Para complementar o editor, seguiu-se a organização dos seus componentes. Para estruturá-los houve necessidade de utilizar a tecnologia EOL[2] . O EOL permite ao utilizador alterar a forma de apresentação da palette (nome dado à caixa onde estão os elementos da linguagem). Isto ajuda o criador a facilitar a apresentação da sua linguagem ao utilizador final, simplificando ao máximo complexidades de percepção. Na Figura 4.5, podemos observar um bom exemplo do funcionamento desta tecnologia. Do lado esquerdo é apresentado um excerto de código EOL e do lado direito o resultado da sua implantação. Neste caso o que está a ser feito pelo EOL, é o renaming dos objetos, que conhecemos por metaclasses. É subtraído o prefixo interface ficando apenas o nome do tipo da metaclasse que o editor possui.

```
var palette := GmfTool!Palette.all.first();
var toolGroupObjects = GmfTool!ToolGroup.all.selectOne(r|r.title = 'Objects');
var toolGroupConnections = GmfTool!ToolGroup.all.selectOne(r|r.title = 'Connections');
var toolEntry;

--renaming objects:

toolEntry := toolGroupObjects.tools.selectOne(r|r.title = 'interfaceTerminal');
toolEntry.title := 'Terminal';
toolGroupObjects.tools.add(toolEntry);

toolEntry := toolGroupObjects.tools.selectOne(r|r.title = 'interfaceConcept');
toolEntry.title := 'Concept';
toolGroupObjects.tools.add(toolEntry);

toolEntry := toolGroupObjects.tools.selectOne(r|r.title = 'interfaceDomain');
toolEntry.title := 'Domain';
toolGroupObjects.tools.add(toolEntry);

toolEntry := toolGroupObjects.tools.selectOne(r|r.title = 'interfaceEvent');
toolEntry.title := 'Event';
toolGroupObjects.tools.add(toolEntry);

toolEntry := toolGroupObjects.tools.selectOne(r|r.title = 'interfaceResource');
toolEntry.title := 'Resource';
toolGroupObjects.tools.add(toolEntry);
```

b Código EOL



b Palette do editor alterada pelo EOL

Figura 4.5: Funcionamento do EOL

4.3.3 EVL

Para conseguir aumentar a consistência do editor, no que toca à construção de modelos, recorreu-se à utilização do EVL[3]. Esta linguagem de validação, permite-nos alertar o utilizador para a eventualidade de erros na produção do modelo da aplicação WebGIS

[2]https://www.eclipse.org/epsilon/doc/eol/
[3]https://www.eclipse.org/epsilon/doc/evl/

desejada. O EVL atua diretamente no modelo mediante a validação executada pelo utilizador, isto é, o utilizador desenha o seu modelo e pode à medida que o seu protótipo vai ficando mais complexo, verificar se existem erros na interligação de metaclasses ou nas propriedades das mesmas.

Na Figura 4.6 podemos observar três verificações feitas a três relações diferentes no documento EVL. As verificações, atuam de forma a prevenir que existam conceitos modelados, que não estejam interligados.

```
231  constraint domain_concept{
232  check:(self.domain_concept.size() == self.hasConcept.size())
233  message: 'All domain Concepts must have a relation with Domain.'
234  }
235
236  constraint concept_maplayer{
237  check:(self.concept_maplayer.size() == self.hasLayer.size())
238  message: 'A map Layer must be related with a Concept of Domain.'
239  }
240
241  constraint map_maplayer{
242  check:(self.map_maplayer.size() == self.hasLayer.size())
243  message: 'The Map must be related with all map Layers.'
244  }
```

Figura 4.6: Excerto de código EVL

De seguida na Figura 4.7 observamos outro excerto de código no documento EVL. Há uma prevenção relativamente ao nome da relação entre os conceitos **Concept** e **Map Layer**. É também definido um conjunto de possíveis soluções para a resolução do problema.

```
328 context interfaceConcept mapLayer{
329 constraint concept_maplayer{
330 check : self.Name.isDefined()
331 message :'This relation between Concept and Map Layer should be defined according its purpose'
332             fix {
333                 title: 'represented as'
334                 do{ self.Name:= 'represented as';   }
335             }
336             fix {
337                 title: 'abstracted as'
338                 do{ self.Name:= 'abstracted as';    }
339             }
340 }
341 }
```

Figura 4.7: Outro excerto de código EVL

Na Figura 4.8 observamos a validação de um modelo concebido no editor, que lança

dois erros de acordo com os excertos de código EVL apresentados anteriormente. O primeiro problema diz respeito à relação entre os conceitos Map Layer e Concept. O segundo, surge devido ao nome da relação entre o domínio e o conceito do domínio não estar definido.

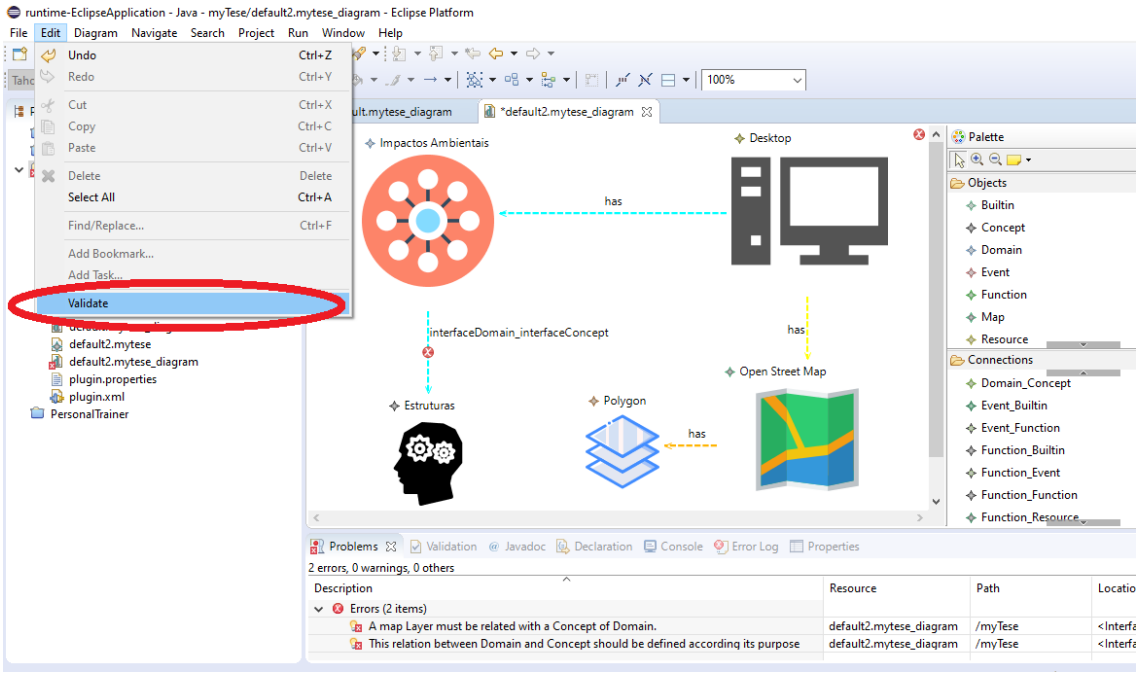


Figura 4.8: Validação EVL de acordo com o código apresentado anteriormente

Por fim, na Figura 4.9, observamos a correção, de um dos problemas que foram lançados no alerta da validação do EVL. A partir de um clique, sobre o problema do nome da relação entre dois conceitos é disponibilizado um mecanismo de resolução rápida "Quick fix". Ao ser accionado, é levantada uma janela, onde são apresentadas as soluções disponíveis. Estas, são "filter"e "focus". Existe também uma opção para ignorar, mas esta não vai resolver o problema.

4.4 Guião

Já foram aprofundadas na secção anterior, as tecnologias fundamentais ao protótipo desenvolvido, agora é transmitido o conjunto de passos para a boa utilização da linguagem. Na Figura 4.10, é apresentada a palette do editor. Como já foi falado na secção do EOL,

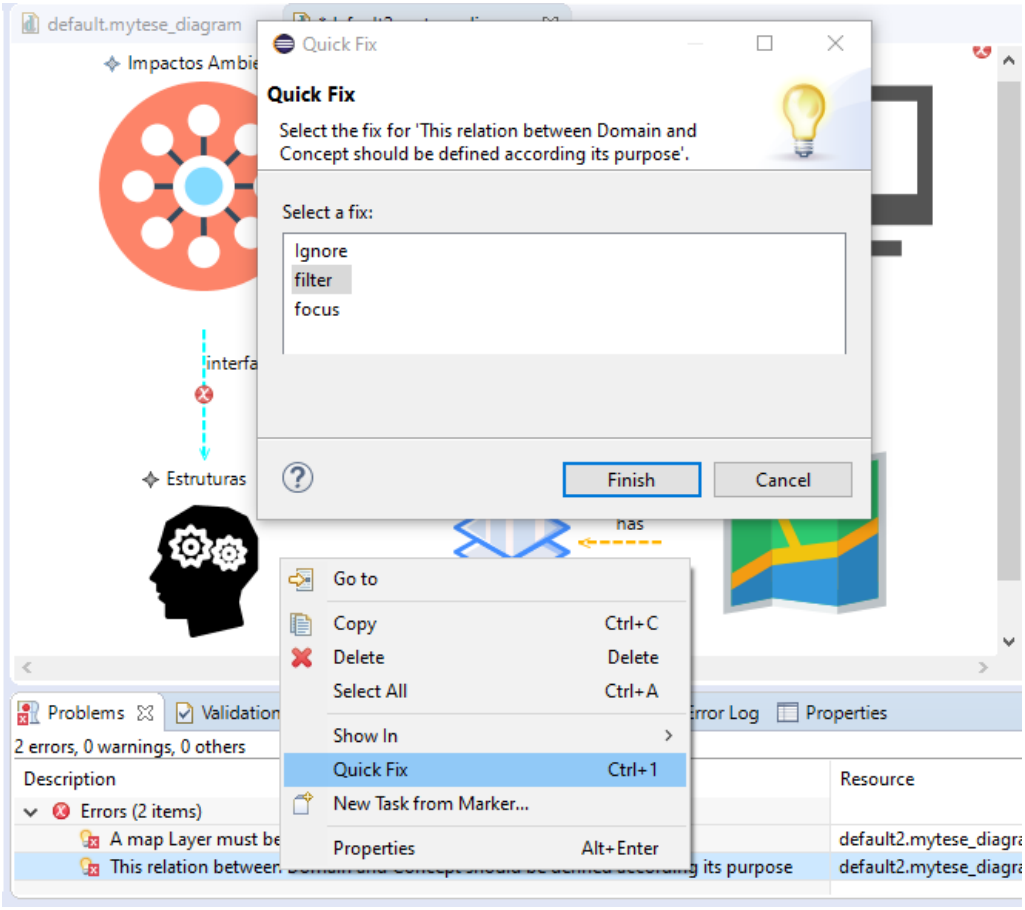


Figura 4.9: Resolução de conflitos no modelo a partir do EVL

a palette é a caixa onde estão as metaclasses que vão ser utilizadas para criar o modelo de interface. Podemos observar que a palette está dividida em duas partes, Objects e Connections. Na divisão Objects, vão estar as funcionalidades e propriedades que formam a interface WebGIS desejada. Na divisão Connections onde temos as ligações para interligar os componentes, isto é, os Objects da divisão acima. A partir daqui o utilizador deverá criar os modelos que desejar, combinando os componentes (Objects) como entender, respeitando a essência de cada um, através das ligações (Connections).

De seguida, é produzido um exemplo de modelação de interface para um aplicação WebGIS de localização de hotéis. Na mesma Figura 4.10, observamos, do lado esquerdo da palette do editor, o resultado da instanciação da metaclasse Terminal. Esta, está bem visível na palette do editor sendo a penúltima na lista da divisão Objects. Esta metaclasse, diz respeito ao terminal para a qual a interface está a ser criada. O terminal pode ser um computador *Desktop*, um *Smartphone*, ou um *Tablet*. Para este exemplo utilizaremos o **Desktop**. Sendo que o tipo de interface deverá ser especificado no nome da figura que será carregada no editor após a escolha da metaclasse a partir da Palette (neste caso, Terminal).

Outra propriedade importante na construção de uma aplicação WebGIS é o domínio,

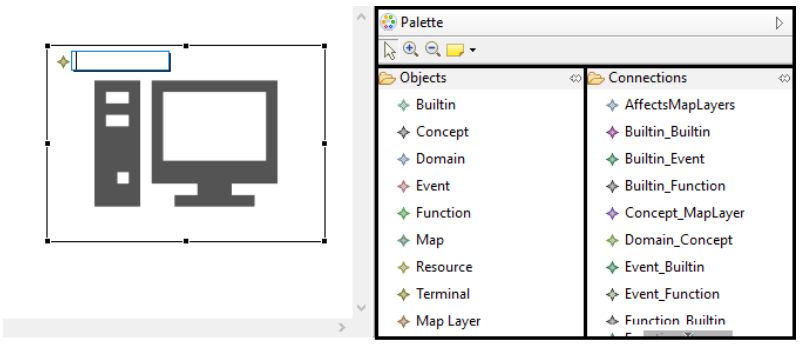


Figura 4.10: Editor

pois é o âmbito pelo o qual se está a desenvolver a aplicação. Na palette, os elementos do modelo estão em inglês pelo que o domínio é apresentado como Domain. Neste exemplo o domínio será **Hotelaria**, ver Figura 4.11.

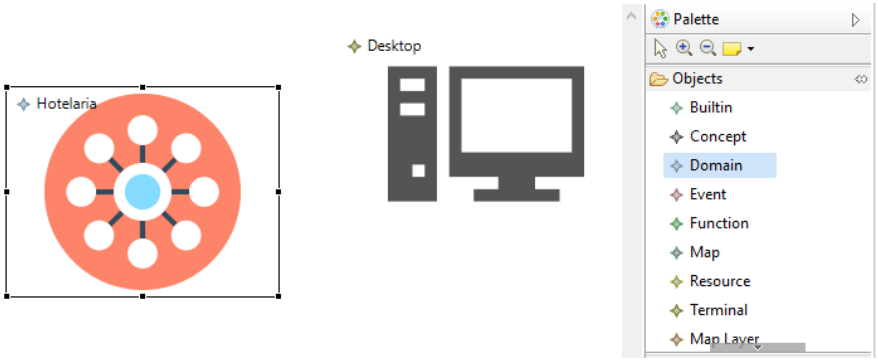


Figura 4.11: Metaclasse Domínio

Seleccionado o domínio da aplicação com sucesso há depois a necessidade de relacionar o mesmo com o sistema, neste caso o terminal. Como tal, recorre-se a uma ligação do separador Connections, sendo a mais adequada **Terminal_Domain** (Figura 4.12).

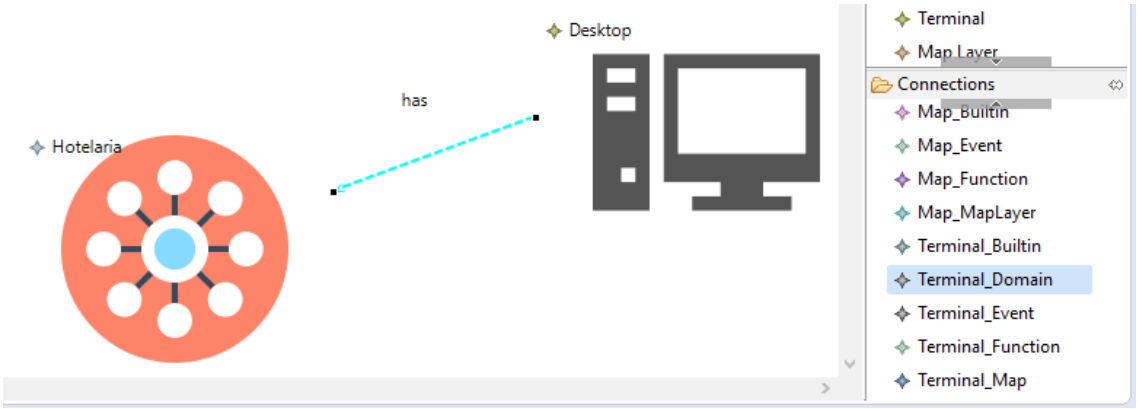


Figura 4.12: Terminal has Domain

Seleccionada a ligação Terminal_Domain, há que homologar a respetiva relação, sendo que o utilizador deverá clicar na metaclasse Terminal e arrastar o clique até ao domínio

tal como o nome da ligação o indica sendo depois criada uma linha tracejada (azul clara nesta relação, noutras ligações a cor pode variar). Há que referir que as cores utilizadas nas relações, bem como o tipo de linha (tracejada ou não) servem apenas para orientar o utilizador no processo de modelação, não existindo qualquer critério ou significado extra para a sua utilização. Isto também é aplicado aos nomes das relações, que são definidas pelo utilizador (ex: has; filter;). De modo a conseguir um modelo mais apróximado à aplicação pretendida, o utilizador deverá nesta fase definir de forma explicita o que pretende do Domínio. Neste exemplo queremos localizar hotéis, como tal serão filtrados os: hotéis do domínio hotelaria. Sendo que o que é filtrado é um **Concept** na secção de Objects da Palette, e a ligação para esta filtragem será **Domain_Concept**, como podemos visualizar na Figura 4.13.

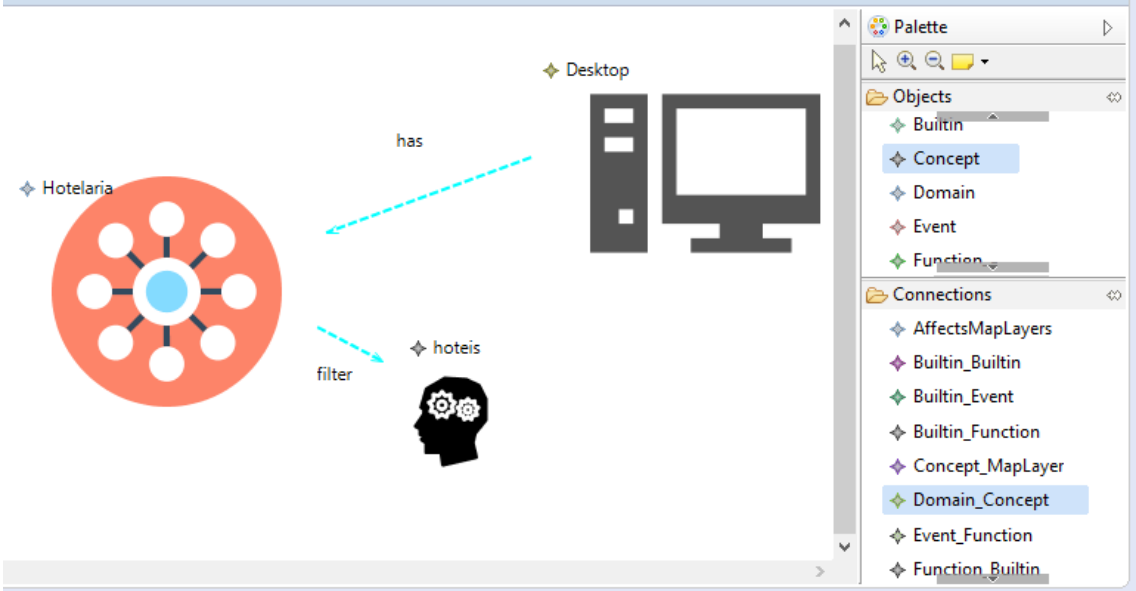


Figura 4.13: Concept of Domain

Há que salientar que neste exemplo o objetivo é apenas localizar hotéis, mas se por exemplo, o utilizador pretendesse apresentar mais alguma informação seria adequado acrescentar mais um conceito de acordo com o que seria pretendido. Outra propriedade muito importante que deve ser modelada é o mapa. O mapa da aplicação WebGIS é onde será apresentada a informação, e como tal tem de ser representada e definida qual a escolhida para a aplicação do utilizador. Como é observável na Figura 4.14, **Map** foi instanciada com o nome "Open Street Map", sendo que será este o mapa pretendido para a aplicação. Há também a ligação do Terminal ao mapa refletindo a sua relação.

O Mapa por si só não faz uma aplicação WebGIS, há a necessidade de apresentar dados no mapa. Aqui nasce **Map_Layer**. Esta tem como função abstrair os dados do domínio da aplicação WebGIS que o utilizador pretende representar num tipo de funcionalidade do Mapa. Na Figura 4.15, a map layer serão marcadores, quer isto dizer que há um tipo de dados que serão apresentados no mapa mediante a utilização de marcadores.

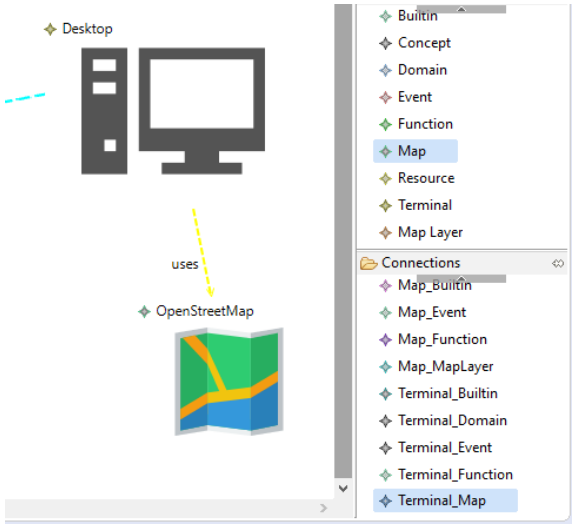


Figura 4.14: Map

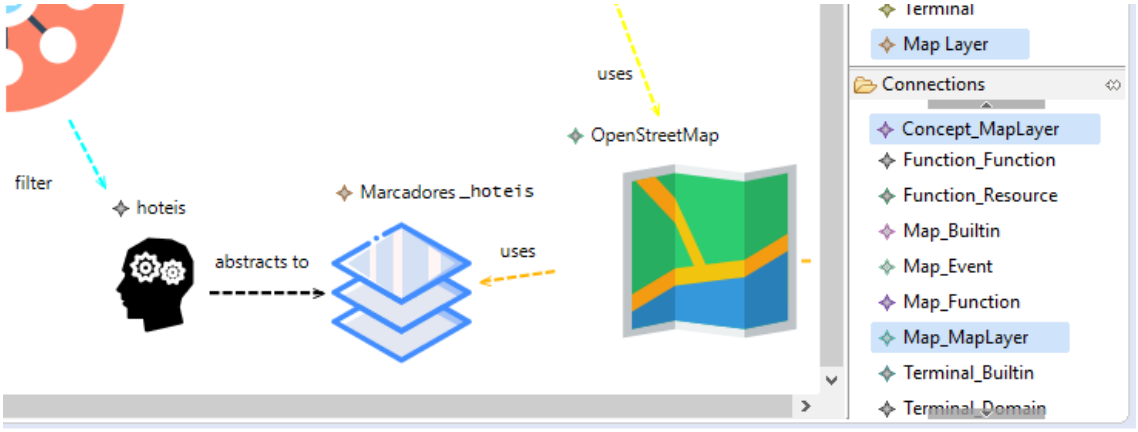


Figura 4.15: Map layer

Como também é possível ver na mesma imagem, há uma ligação denominada "abstracts to" entre o conceito hotéis e a camada do mapa (Map Layer) marcadores. Esta ligação em paralelo à ligação da camada denominada "Marcadores" com a metaclasse Mapa que é denominada "Open Street Map", representa a abstração dos dados filtrados do domínio hotelaria, os hotéis, que são representados no mapa "Open Street Map" sob forma de uma camada de dados "Marcadores". Estes Marcadores, que são dados absorvidos do domínio hotelaria são funcionalidades (Builtins) do Mapa. Assim é necessário representá-los também como Builtins no editor. Na Figura 4.16 podemos visualizar essa confirmação, onde o **Builtin** "Marcadores" é representado por uma roda de engrenagem verde e a sua ligação ao Mapa é feita através da conexão **Map_Bultin** com a designação "has".

Nesta fase, em que o utilizador já definiu o "core" da aplicação WebGIS, pode começar a personalizá-la à sua maneira. Imaginemos que o utilizador pretende que quando se clique num marcador (hotel) presente no mapa, seja informado acerca do preço pela

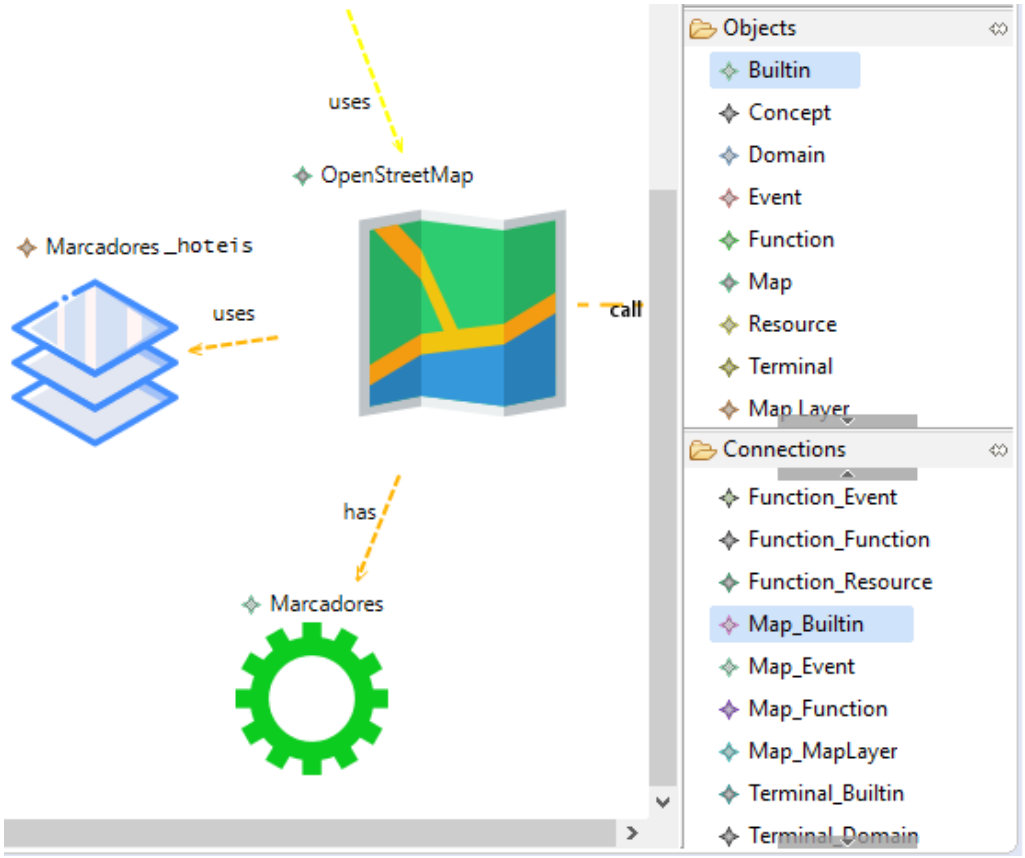


Figura 4.16: Built in: Marcadores

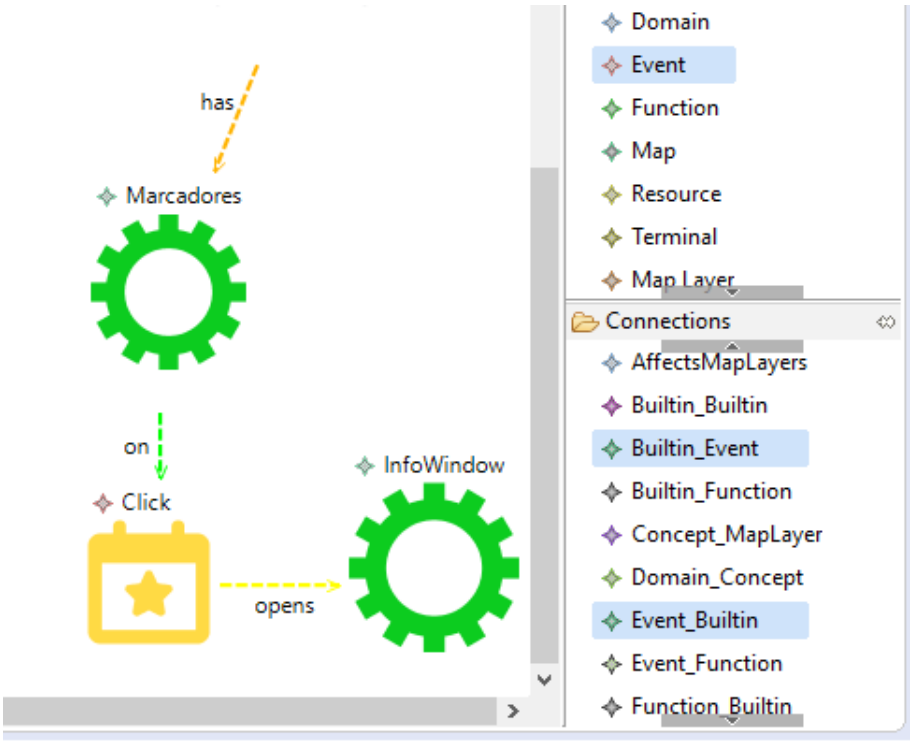


Figura 4.17: Evento: On Click

estadia de uma noite. Nesse caso, pode ser definido um Evento "Click" nos "Marcadores" que faz abrir uma janela de informação (Info Window) com os dados acerca do hotel (marcador) onde clicou (ver Figura 4.17).

Agora imagine-se que o utilizador pretendia que este tipo de janelas de informação acerca dos preços de estadia, fornecesse a habilidade para alterar o preço quando ele quisesse. Para modelar este cenário, o utilizador adicionava uma metaclasse **Function** relacionando-a com o Builtin da janela de informação (InfoWindow) sob a designação "provide" (fornece) com o objetivo de editar a informação do hotel, "Edit Info", esta modelação pode ser acompanhada na Figura 4.18.

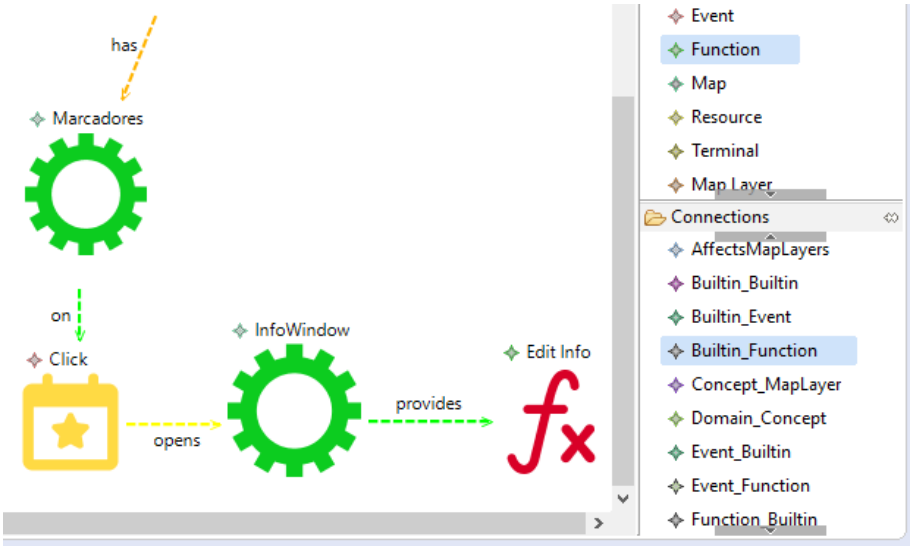


Figura 4.18: Function: Edit Info

Sabendo que, atualmente, quase todas as aplicações utilizam bases de dados, avançamos neste exemplo que os dados dos hotéis, tais como nomes, preços e moradas se encontram num repositório de dados. Desta forma, a edição dos dados por parte da função exemplificada na Figura anterior (4.18) é realizada através de queries (perguntas) à base de dados. Então, a modelação deste processo poderá ser feito através de uma ligação da função "Edit Info" a um **Resource** cujo o nome é "Database Hoteis" sendo então o nosso recurso (resource) uma base de dados. Esta modelação pode ser observada na Figura 4.19.

Tendo terminado toda a modelação de requisitos de interface, o utilizador deverá fazer a validação do modelo através do suporte desenvolvido com a tecnologia EVL (ver Figura 4.8) e no caso de não haver problemas lançados pelo editor, o modelo está pronto a ser entregue às entidades de desenvolvimento. Esta poderia ser uma instanciação de modelação de requisitos de interface da linguagem desenvolvida (Figura 4.20), cabendo ao utilizador desenvolver uma modelação de acordo com os seus interesses. A complexidade da modelação de requisitos vai depender do nível de profundidade que a entidade responsável adoptar, por isso, é necessário ter em conta que os nomes e combinações definidas, devem ser as mais claras possíveis.

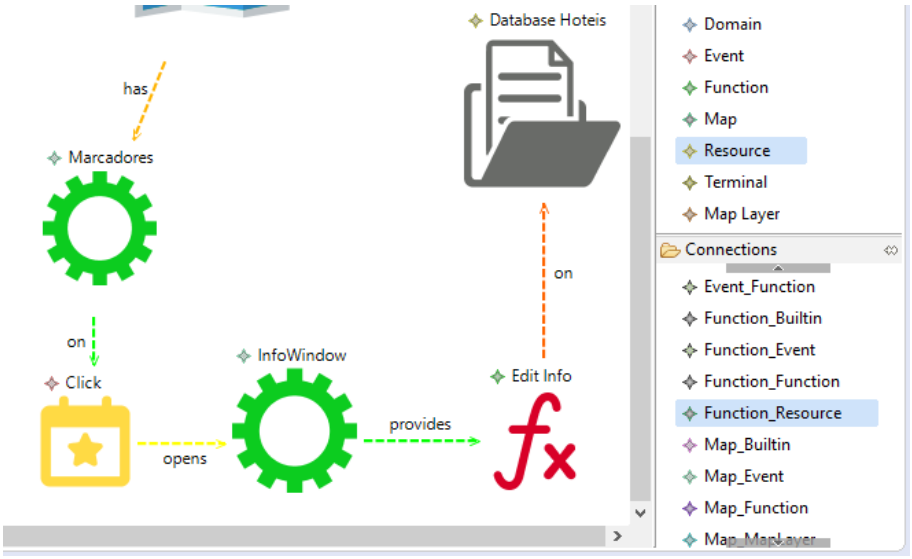


Figura 4.19: Resource: Database

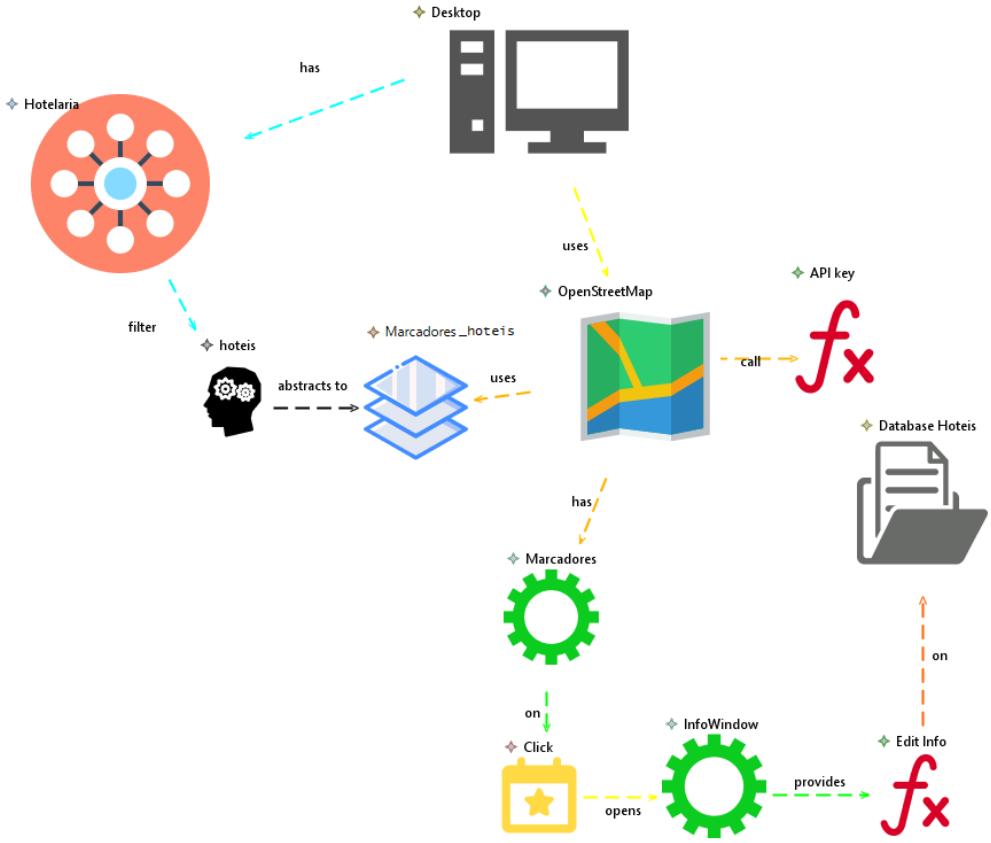


Figura 4.20: Modelo final

4.5 Sumário

Neste Capítulo 4, verificou-se a apresentação dos conceitos do metamodelo, isto é, as metaclasses, bem como as suas relações entre si. É explicada a funcionalidade de todas as metaclasses. Neste capítulo é também refletido todo o trabalho desenvolvido à volta do editor. São apresentadas as tecnologias utilizadas na sua produção (EMF, EOL e EVL), como também é explicado o propósito de cada uma delas. Mais adiante foi apresentado um guião sobre a utilização práctica do editor. É mostrado como se deve utilizar o software para realizar a modelação, a partir das metaclasses definidas, de interfaces de aplicações WebGIS. No próximo capítulo é apresentada a validação da linguagem a partir de um caso de estudo.

Capítulo 5

Validação da Linguagem

Neste capítulo é descrita a aplicação do projeto desenvolvido num caso de estudo. Esta execução foi realizada com o intuito de detetar possíveis metaclasses em falta ou outro tipo de falhas que não tenham sido pensadas durante a fase de implementação.

5.1 Caso aplicado

De forma a verificar se a implementação do protótipo foi bem desenvolvida, aplicou-se a modelação de requisitos de uma aplicação WebGIS, já desenvolvido no departamento de informática da FCT/UNL. Esta aplicação, foi desenvolvida no âmbito de uma dissertação de Mestrado, sobre estruturas costeiras [50]. O sistema geográfico utilizado, foi desenvolvido para analisar estragos em plataformas de estruturas costeiras (pontões). Estas estruturas sofrem forte erosão ao longo do tempo, devido aos avanços do Mar, surgindo a necessidade de se desenvolver uma aplicação WebGIS, para ajudar, a partir da recolha e armazenamento de informação diretamente na base de dados à sua consulta. Isto, veio permitir subtrair o passo intermédio, de recolha dos dados em papel, que podia adicionar erros ao processo.

Na Figura 5.1, podemos observar uma parte da aplicação, onde o **pontão** é representado por um polígono azul. Visualizamos também alguns marcadores amarelos, que representam **pontos de visibilidade**. Pontos de visibilidade são lugares específicos no pontão onde foram tiradas uma ou várias fotografias num determinado intervalo de tempo, para estudar a evolução do estado da estrutura. Observamos também um marcador encarnado que representa a **Location Awareness** (localização atual do utilizador). Neste caso, o utilizador estaria naquele lugar da estrutura quando recolheu os dados associados à **InfoWindow** (janela de informação). Uma InfoWindow é ativada sempre que o utilizador faz um clique na área azul, apresentando o nome do lugar onde se situa o

pontão e disponibilizando **três funções** (Ver Info; Ver Fotos; Ver Vídeos;). Todas elas, são funções de consulta de dados sobre o pontão, sendo que estes dados estão armazenados na **base de dados** da aplicação. Observamos também, à parte do mapa, na zona inferior da imagem, um **menu** em forma de três botões, "Carregar Shapefile", "Carregar Fotos"e "Logout".

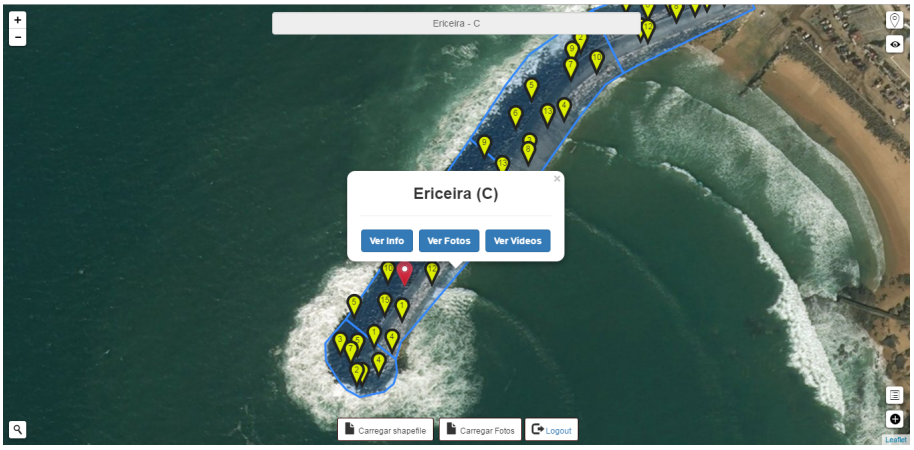


Figura 5.1: Aplicação WebGIS acerca de Estruturas Costeiras.

De seguida, na Figura 5.2, no mesmo lugar, visualizamos uma imagem menos detalhada. Aqui, a visualização do mapa é feita com menor detalhe, assumimos que houve um evento **Zoom Out**. Ao ocorrer este evento, os marcadores amarelos, apresentados na figura anterior como pontos de visibilidade, são agrupados em círculos verdes com um número. Estes círculos, representam **Clusters**, que consistem neste contexto, num agrupamento de marcadores. Quer isto dizer que num círculo verde com o número sete, existem sete pontos de visibilidade escondidos, que serão estendidos caso se clique no mesmo, isto é, sete lugares naquela área do pontão onde foram tiradas fotografias para avaliar a sua evolução ao longo do tempo.

Por fim, visualizamos, na Figura 5.3, o cenário da aplicação quando o utilizador faz clique num ponto de visibilidade, onde são depois visualizadas todas as fotografias tiradas naquele lugar e a direcção de captura de imagem selecionada no momento.

5.2 Modelação WebGIS IRML

A modelação de requisitos para uma aplicação WebGIS que se realizou com base no caso de estudo apresentado na secção anterior é apresentada a partir da Figura 5.4. Há que relembrar que este processo de modelação deverá ser feito antes da implementação do sistema geográfico e não depois. Como o objetivo desta etapa era verificar falhas, optou-se por realizar desta forma, numa tentativa de chegar ao modelo mais aproximado possível do WebGIS escolhido. Começando pelo inicio do modelo, o tipo de terminal para o

Figura 5.2: Clusters utilizados na aplicação WebGIS.

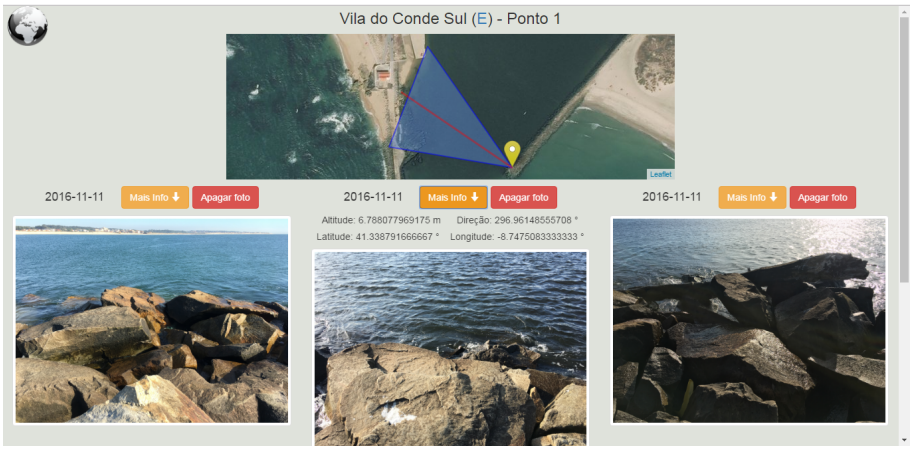


Figura 5.3: Pontos de visibilidade na aplicação.

qual se desenvolveu o sistema foi inicialmente o computador de escritório, conhecido por **Desktop**, sendo depois possível a sua utilização em dispositivos móveis Tablets. Posto isto, o WebGIS possui um **Builtin** (funcionalidade) de sensibilidade ao contexto, **Location Awareness**. Também é possível observar que possui a funcionalidade **Menu**. Esta utilidade, permite trabalhar com alguns dados dos pontões (estruturas costeiras) visíveis no mapa que estão armazenados na base de dados (**Resource**). Quanto a estes últimos dois conceitos será feita a sua explicação mais adiante. Esta parte inicial modelada, pode ser observada na Figura 5.4.

O WebGIS foca-se no domínio das Estruturas Costeiras como já foi referido. Sendo também visível a relação entre este domínio e o Terminal Desktop. A partir deste foco, são destacados os **Concepts** (conceitos) essenciais que são extraídos deste domínio, os **Pontões** e os **Pontos de Visibilidade**. Estes dois conceitos, são desejados no mapa. Para tal, é realizada uma abstração desses mesmos conceitos em **Camadas de Dados**. Os Pontões em **Polygons** (Polígonos) e os Pontos de visibilidade em **Markers** (Marcadores). As camadas

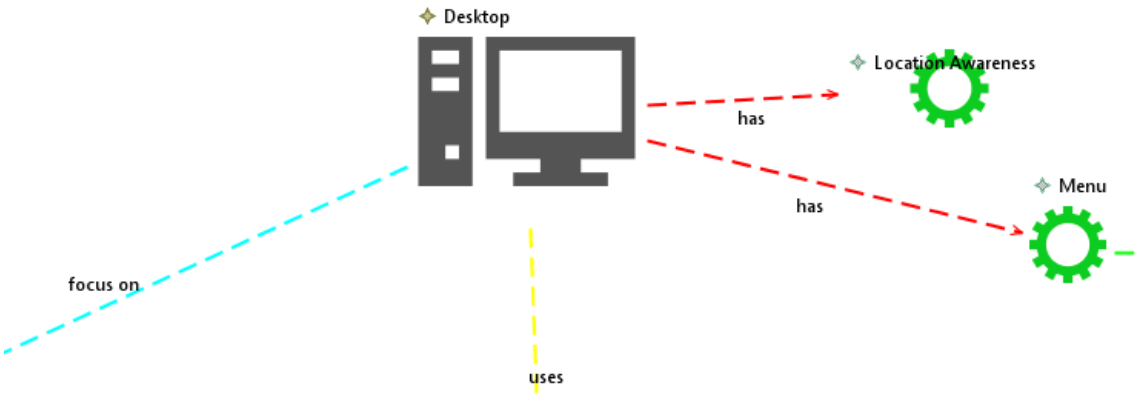


Figura 5.4: Terminal da aplicação WebGIS para Estruturas Costeiras.

são propriedades do **Mapa**. Para esta aplicação, foi utilizado o Mapa **Open Street Map** [1], que é geralmente utilizado em contexto de aplicações de código aberto. Esta modelação que vai desde o Domínio até ao Mapa, pode ser observada na Figura 5.5.

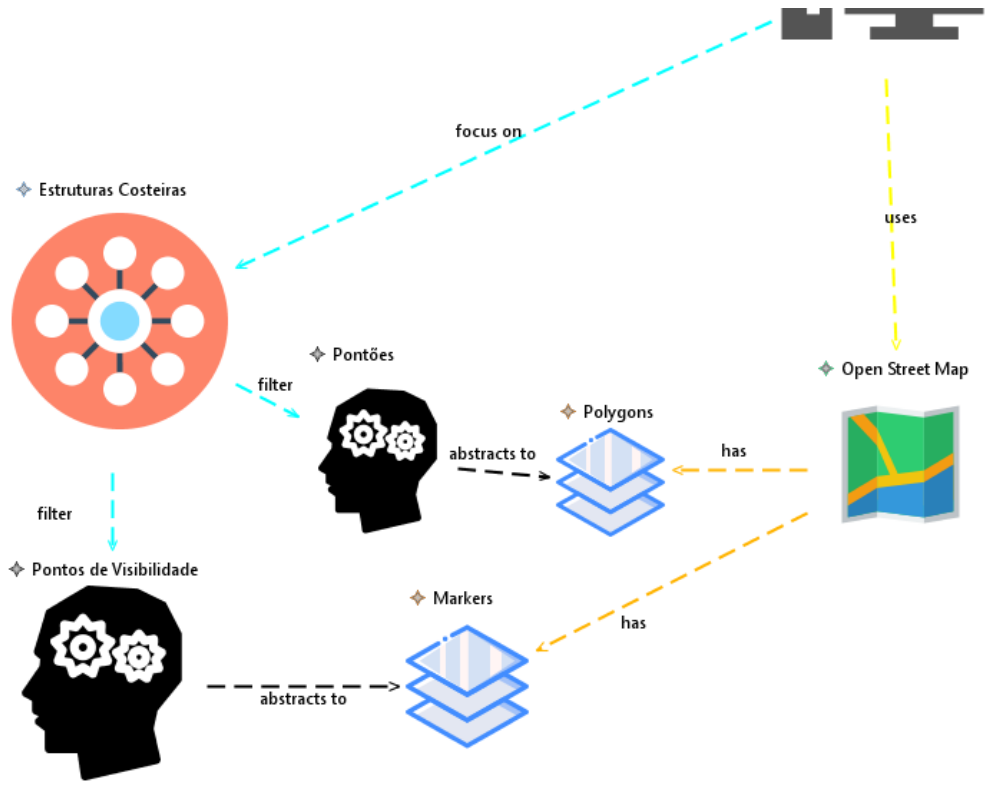


Figura 5.5: Abstracção dos conceitos do domínio no mapa da aplicação.

Assumindo que esta aplicação foi desenvolvida sob uma API da Google Maps que fornece várias funcionalidades para aplicar sobre o mapa, a sua utilização requer uma licença da Google. Esta concessão é realizada através da atribuição de uma chave de API (**API key**) que é feita através da chamada de uma função.

[1] https://www.openstreetmap.org/

O Mapa, possui várias funcionalidades. Algumas das utilizadas são: **Polygons**, **Markers**, **Clusters** (agrupamento de marcadores) que são feitos mediante o evento **Zoom out**. Existem também as funcionalidades **Zoom**, **Pan Crop** (navegação pelo mapa) e **InfoWindow** (janela de informação), esta última está incoporada nos Polygons (estruturas) e é ativada a partir do **Event** Click. Esta parte do modelo, acerca do Mapa e respetivas funcionalidades à excepção da InfoWindow e do Event (Figura 5.7), pode ser observada na Figura 5.6.

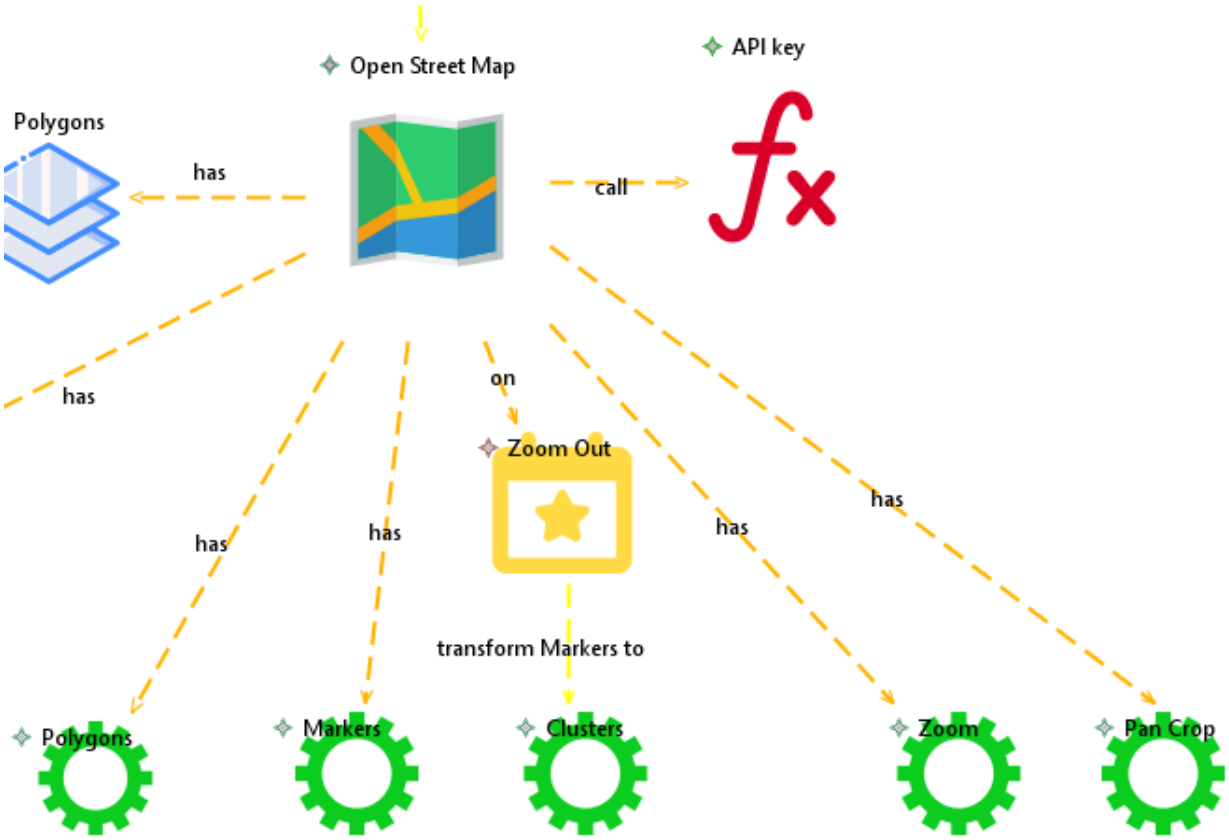


Figura 5.6: Funcionalidades do mapa.

A partir das InfoWindows, isto é, as janelas de informação, são fornecidas ao utilizador funcionalidades de **Ver**, **Editar** e **apagar** informações, fotos e videos sobre as estruturas costeiras. Estas, como foram alvo de abstração para um polígono, permitem assim, a ativação das InfoWindows, mediante um click na estrutura ou polígono, como for preferível. Podemos observar esta interacção modelada na Figura 5.7.

Todas as informações das estruturas estão armazenadas num **Resource**, que no contexto da aplicação deste caso de estudo é uma base de dados de estruturas costeiras. Este processo de armazenamento de dados está modelado e visível na Figura 5.8.

Relembrando uma das funcionalidades mencionadas no inicio desta secção, o Menu

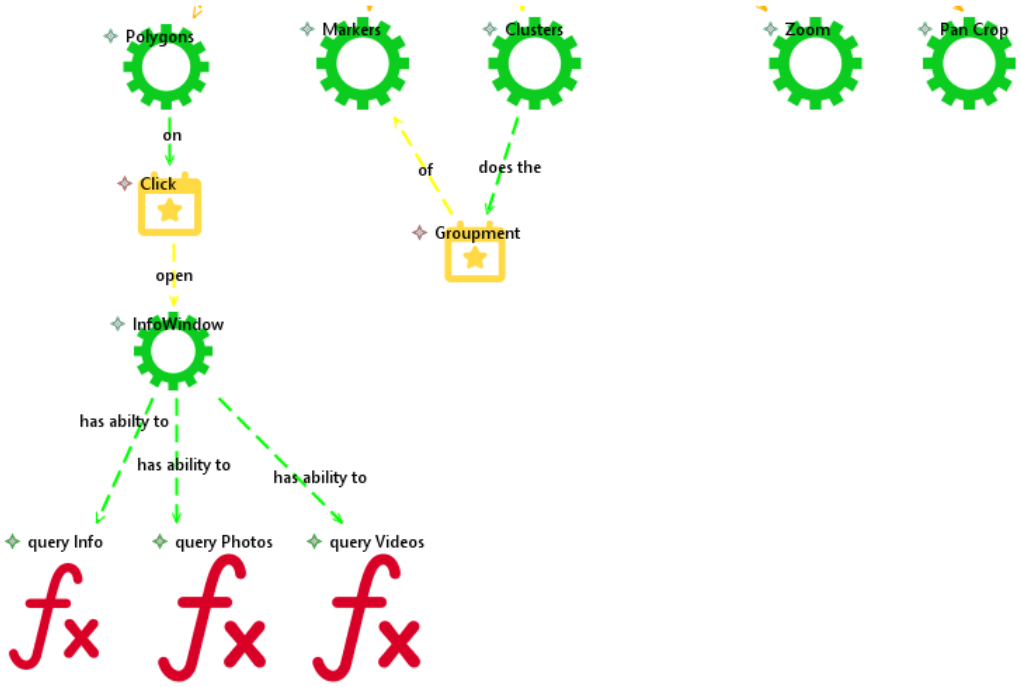


Figura 5.7: Janela de informação para tratamento de dados.

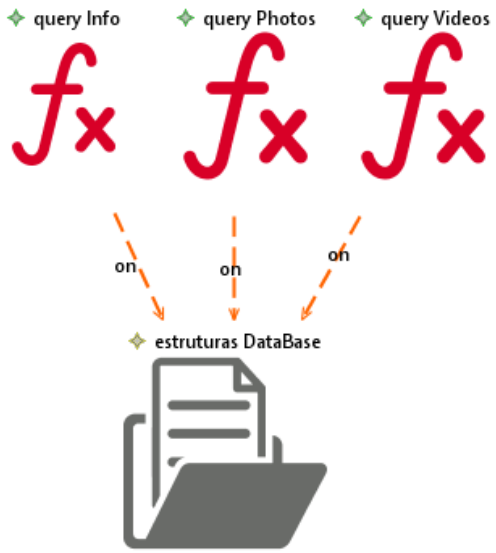


Figura 5.8: Armazenamento de dados.

do terminal (Desktop), permite também editar informações armazenadas no Resource definido, ainda que de maneira menos clara. No caso do utilizador optar por realizar alguma alteração nos dados a partir deste "caminho" terá necessariamente que saber qual o pontão que está a editar, pois caso contrário poderá corromper os dados. Esta ligação da base de dados também às funcionalidades do Terminal, bem como o modelo final, estão representadas na Figura 5.9.

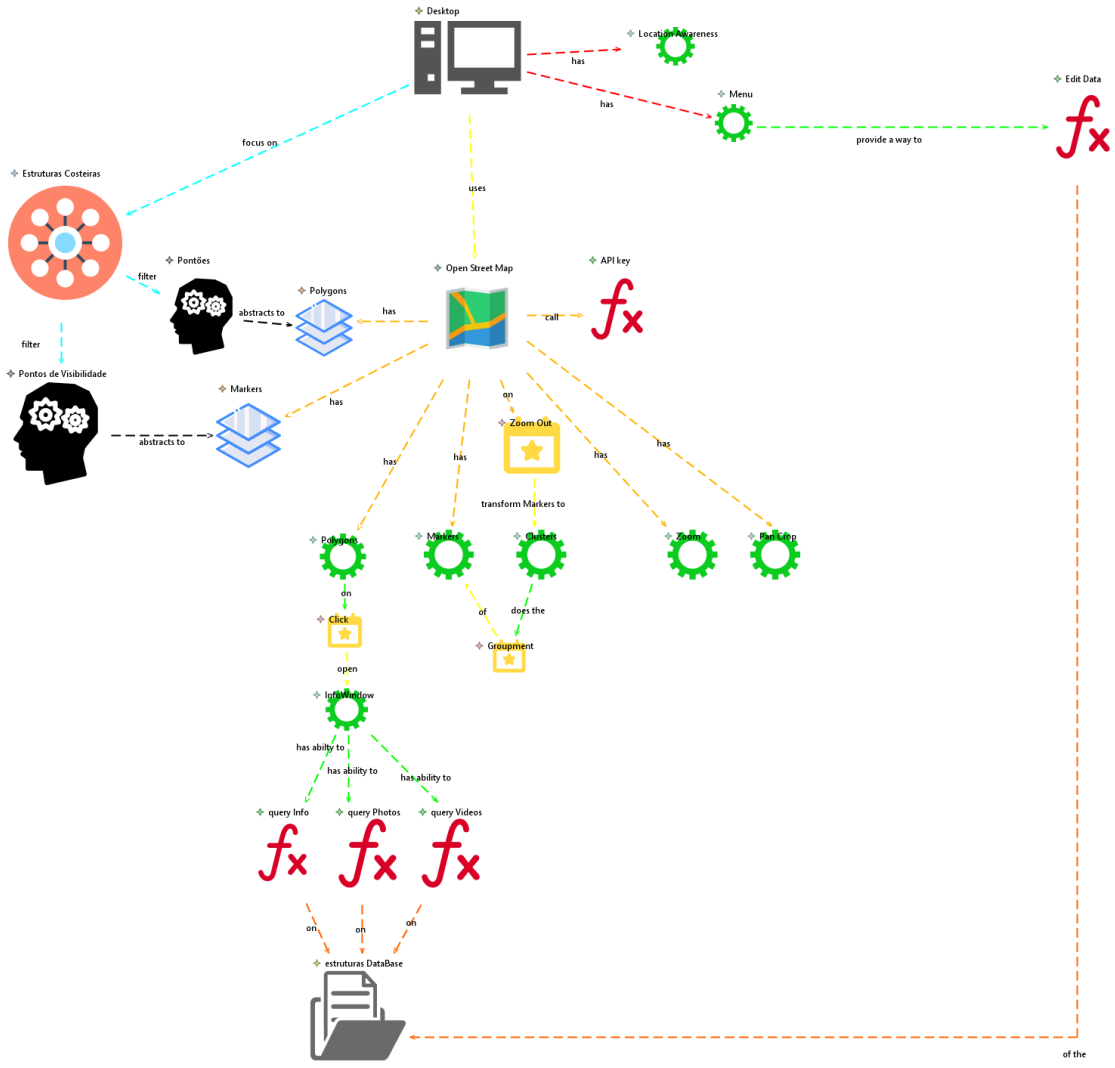


Figura 5.9: Modelação de requisitos para WebGIS de Estruturas Costeiras.

5.3 Sumário

O Capítulo 5, destaca a importância da validação da linguagem. É aplicada a modelação de uma interface, de uma aplicação WebGIS já existente. O principal objetivo é transmitir que os conceitos da linguagem desenvolvidos, são suficientes para modelar uma aplicação WebGIS e garantir que não existe nenhuma metaclasse em falta. No geral, os conceitos

criados, conseguem modelar qualquer tipo de propriedade de aplicações WebGIS, não tendo sido detectada a ausência de um conceito necessário para modelar algo específico e comum ao universo WebGIS. Acreditamos que a validação foi concluída com sucesso. No próximo capítulo é abordado o processo de avaliação de todo o trabalho desenvolvido.

Capítulo 6

Avaliação

Neste capítulo é apresentado o processo de avaliação dos conceitos da linguagem e do editor da linguagem. O objetivo desta avaliação foi detetar possíveis inconsistências e falhas não detetadas anteriormente na fase de desenvolvimento da linguagem e respetivo editor. A avaliação, que envolveu trinta participantes, dividiu-se em várias fases. Sendo elas, primeiro a apresentação do glossário de conceitos da linguagem, apresentação de modelos onde se aplicaram estes conceitos e um questionário sob cronometragem para verificar se o participante percebeu os conceitos e a forma como são utilizados para modelar requisitos de interface de WebGIS. Estas fases vão ser explicadas detalhadamente ao longo do capítulo.

6.1 Materiais de Avaliação

No sentido de conseguir uma avaliação rigorosa, houve necessidade de desenvolver e utilizar alguns materiais para podermos proceder à realização de experimentos para avaliação do editor da linguagem. Estes materiais incluem uma apresentação em slideshow com um glossário de conceitos da linguagem, um modelo exemplo de uma aplicação WebGIS onde é apresentada a aplicação dos conceitos e um modelo de outra aplicação WebGIS aplicado a um caso de estudo. Estes conteúdos podem ser observados no **Anexo I** deste relatório. Foi também desenvolvido um questionário, disponível no **Anexo II**, para avaliar a percepção e a dificuldade de aprendizagem dos conceitos da linguagem e da sua aplicação na modelação, por parte do utilizador. E para cada utilizador/avaliador, criou-se também uma declaração de consentimento que foi apresentada a todos os que participaram na avaliação antes do seu inicio. Esta declaração está disponível em **Anexo III**.

6.2 Descrição dos materiais de avaliação

Nesta secção, vão ser apresentados os materiais utilizados para realizar o experimento bem como o objetivo da sua utilização.

6.2.1 Glossário de Conceitos e Modelos

O glossário de conceitos do editor (disponível no Anexo I, Figura I.3 e I.4), traduz o significado de cada um deles e também a forma como se relacionam entre si. O modelo exemplo, de uma interface de aplicação WebGIS, tem como objetivo clarificar o funcionamento dos conceitos do editor, através da sua instanciação. A avaliação foi realizada de acordo com a aprendizagem e reconhecimento dos elementos do editor e a sua aplicação, num modelo de requisitos de interface de uma aplicação WebGIS, desenvolvida de acordo com um caso de estudo.

6.2.2 Questionário

O objetivo do questionário era verificar se os conceitos da linguagem utilizados no editor eram reconhecidos. O reconhecimento dos conceitos da linguagem, que se refletem em elementos do editor, foi avaliado com base na resposta de cada avaliador ao questionário (grupo: Avaliação do modelo). Apesar de ser possível verificar a partir das respostas ao questionário se o avaliador percebeu ou não os conceitos da linguagem que são utilizados no editor para modelar requisitos, recorreu-se também ao cronómetro das respostas que eram dadas acerca do caso de estudo. De seguida são apresentadas as principais perguntas do questionário:

- **Questão 1 —** Sem contar com as funcionalidades, o Terminal tinha outras duas relações. Uma era com o Domínio e a outra era?
- **Questão 2 —** Qual o domínio do modelo observado?
- **Questão 3 —** Qual é o elemento do modelo utilizado para efectuar a ligação entre um conceito do domínio e o mapa?
- **Questão 4 —** Liste cinco Builtins encontrados no modelo.
- **Questão 5 —** Qual o elemento do modelo utilizado para representar a chave de licença da API sobre o mapa escolhido?
- **Questão 6 —** Qual o elemento do modelo que permite abrir uma janela de informação (InfoWindow)?
- **Questão 7 —** A partir da InfoWindow é possível visualzar informação, fotos e videos. Estes dados estão armazenados numa base de dados, que no modelo é representada por um elemento, qual?
- **Questão 8 —** No modelo observado, o Terminal tinha duas funcionalidades, quais?

6.2.3 Cronómetro

O cronómetro permitiu analisar de forma indirecta a dificuldade de aprendizagem e reconhecimento dos elementos do editor. Uma resposta dada no questionário, independentemente de estar certa ou errada, leva o avaliador a demorar algum tempo no seu raciocínio. A duração deste tempo, por si só, pode não querer dizer nada mas se for verificado um padrão de duração ao longo dos trinta participantes poderá existir algum detalhe importante (por exemplo uma dificuldade de compreensão) acerca de um conceito.

6.2.4 Métricas

O processo de avaliação teve ainda o suporte de um conjunto de métricas[51] de avaliação de desempenho de tempo e sucesso, sendo esta métrica de sucesso, utilizada apenas numa pergunta de enumeração de conceitos existentes no modelo.

6.2.4.1 Métricas de avaliação de desempenho de sucesso

A avaliação de desempenho de sucesso tem como objetivo verificar a dificuldade de compreensão da notação dos conceitos do editor. A sua execução será realizada a partir de quatro métricas:

Precision: Esta métrica foi utilizada na questão quatro do questionário (Liste cinco Builtins encontrados no modelo) e tem como objetivo revelar a precisão do utilizador na sua listagem de acordo com o número de elementos respondidos sob o número de elementos corretos.

Fórmula: $\frac{Nr.ElementosQueCorrespondemAoQueEPedido}{Nr.TotalElementosRespondidos}$

Recall: Esta métrica foi também utilizada na questão quatro do questionário (Liste cinco Builtins encontrados no modelo) e tem como objetivo revelar a memorização do utilizador face a este tipo de elemento. Isto é realizado através do número de elementos corretos sobre a quantidade total deste tipo de elementos presentes no modelo. (Nota: Existem ao todo oito elementos do tipo Builtin no modelo apresentado, porém esta questão apenas solicitava cinco. O cálculo é efetuado com base nos oito elementos).

Fórmula: $\frac{Nr.ElementosQueCorrespondemAoQueEPedido}{Nr.TotalDeElementosExistentes}$

Medida F (F-Measure): É a métrica que relaciona a métrica Precision e a métrica Recall com o objetivo de transmitir a média destas duas medidas quando os seus resultados são similares.

Fórmula: $2 * \frac{Precision*Recall}{Precision+Recall}$

Taxa de Sucesso (Hit Rate): $\frac{Nr.RespostasCorretas}{Nr.TotalRepostas}$

6.2.4.2 Métricas de avaliação de desempenho de tempo

A avaliação de desempenho de tempo aponta para a dificuldade da utilização dos conceitos da linguagem. Esta execução será realizada a partir de duas métricas [51]:

-Duração (Duration): Serve esta métrica para contar o tempo que o avaliador demora a responder ao questionário (grupo do modelo) na integra. TF diz respeito ao tempo final, TI diz respeito ao tempo inicial.

Fórmula: TF(Tarefa) - TI(Tarefa)

-Tempo de detecção (Detection time): É a métrica que cronometra o tempo de resposta a cada pergunta do questionário. É aplicada a cada uma pergunta das oito totais sobre o modelo.

Fórmula: T(Detecção resposta) - TI(Tarefa)

6.3 Participantes

A avaliação da linguagem resultou da participação de trinta pessoas, duas mulheres e vinte e oito homens. A idade dos avaliadores vai desde os vinte e um anos até aos trinta e três (Figura 6.1).

As nacionalidades, variam entre a portuguesa, romena e são-tomense(Figura 6.2). Todos os participantes possuíam pelo menos o grau académico de Licenciatura. Destas trinta pessoas, apenas duas só tinham o grau de Licenciatura enquanto as outras vinte e oito possuiam licenciatura e mestrado.

Para além destes dados, foram também recolhidos indicadores, numa escala de um a cinco, onde um representa pouco e cinco representa muita, quanto à experiência dos avaliadores em relação ao desenvolvimento de aplicações WebGIS (Figura 6.3).

Recolheu-se também, numa escala diferente, o tempo de experiência dos avaliadores de desenvolvimento de aplicações WebGIS (Figura 6.4).

Também, numa escala de um a cinco, onde um representa pouca experiência e cinco representa muita, recolheu-se o indicador de acordo com a experiência de utilização de aplicações WebGIS por parte dos utilizadores (Figura 6.5).

Num contexto tecnológico diferente, utilizando a mesma escala de avaliação de um a cinco, onde um representa pouco e cinco representa muita, foram também recolhidos dados em relação à experiência no desenvolvimento de linguagens de domínio específico por parte dos avaliadores (Figura 6.6).

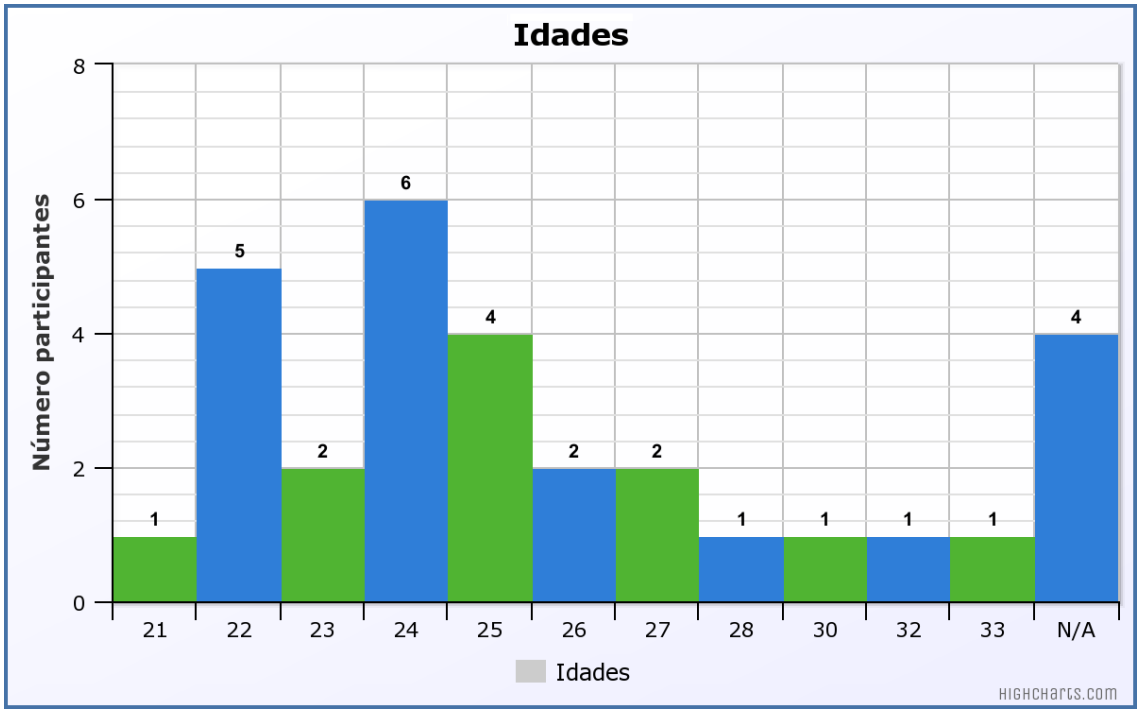


Figura 6.1: Idades dos avaliadores

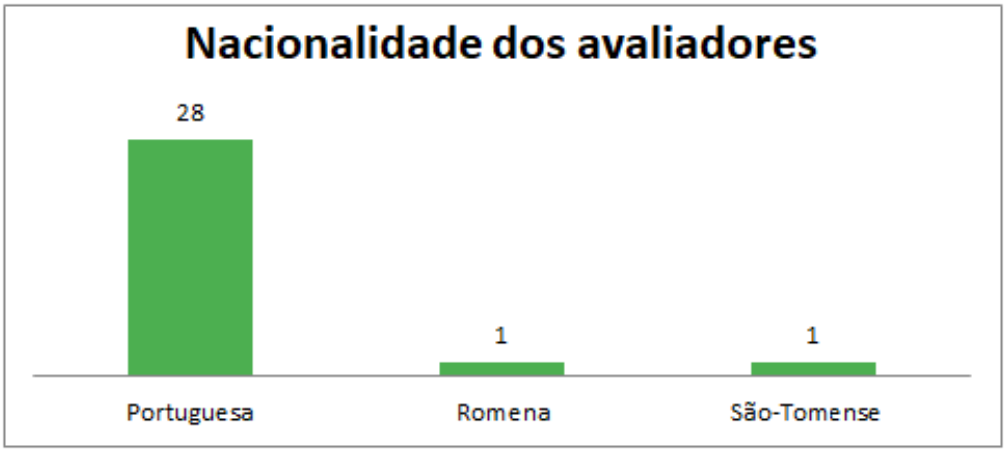


Figura 6.2: Nacionalidade dos avaliadores

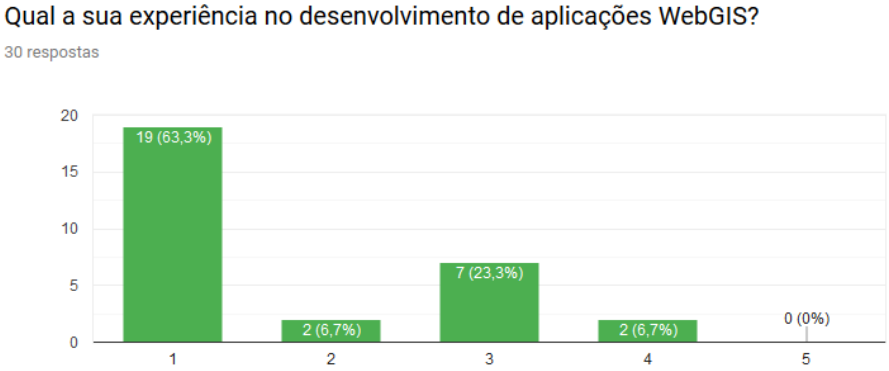


Figura 6.3: Experiência dos avaliadores no desenvolvimento de WebGIS

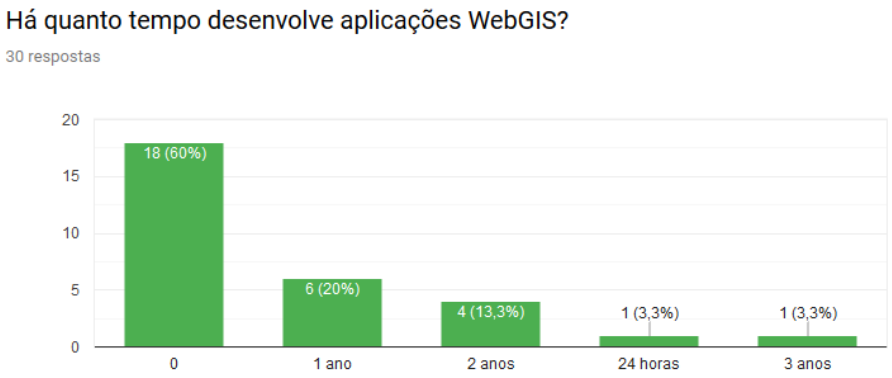


Figura 6.4: Tempo de desenvolvimento de WebGIS dos avaliadores

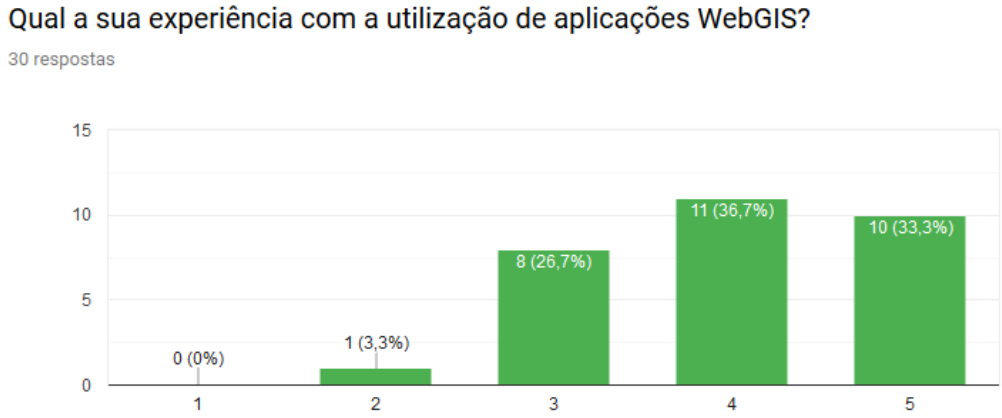


Figura 6.5: Experiência de utilização de WebGIS por parte dos avaliadores

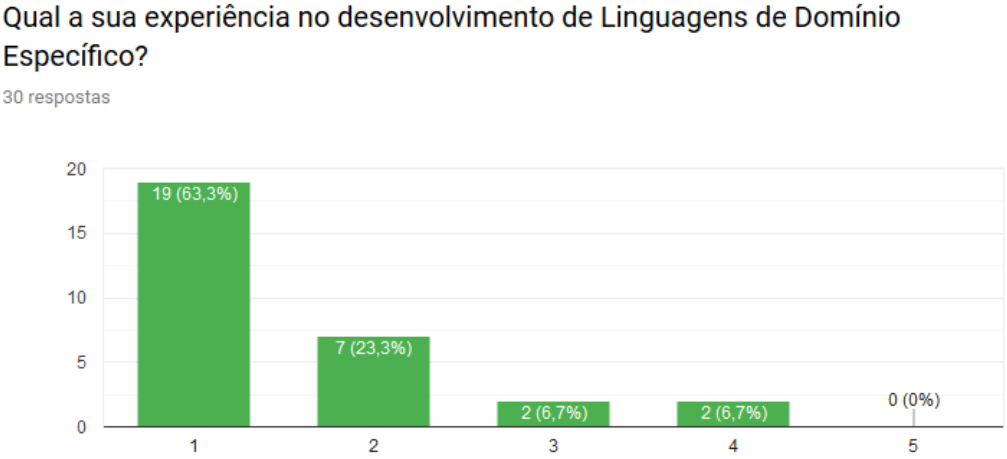


Figura 6.6: Experiência dos avaliadores no desenvolvimento de Linguagens de Domínio Específico

Por fim, antes de iniciar a avaliação do modelo, recolheram-se também, numa escala semelhante, onde um representa pouco e cinco representa muita, dados acerca da utilização de linguagens de domínio específico por parte dos avaliadores (Figura 6.7).

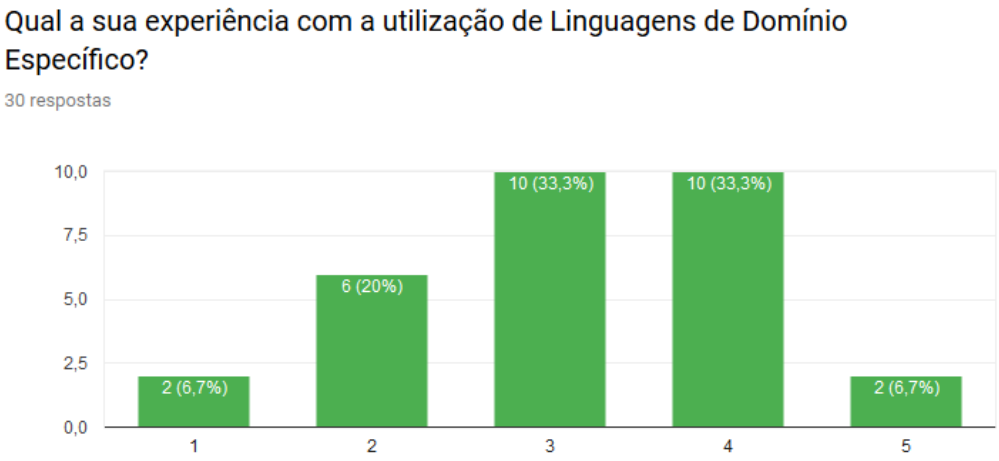


Figura 6.7: Experiência dos avaliadores na utilização de Linguagens de Domínio Específico

6.4 Avaliação

Face à avaliação do modelo concebido acerca de um caso de estudo (capítulo 5) onde se verifica a percepção que o avaliador tem sobre os conceitos da linguagem desenvolvida relativamente aos elementos utilizados pelo editor, observa-se o seguinte:

Na primeira pergunta do modelo (*Sem contar com as funcionalidades, o Terminal tinha outras duas relações. Uma era com o Domínio e a outra era?*), que pedia uma resposta de escolha múltipla (Figura 6.8), vinte-cinco responderam corretamente à pergunta, e 5 responderam erradamente (quatro, Domain Concepts e um, Resource).

Na pergunta seguinte, de resposta livre (*Qual o domínio do modelo observado?*) (Figura 6.9), vinte e dois, responderam corretamente, Estruturas Costeiras. Cinco pessoas, deram três respostas aproximadas à certa, sendo elas "Zonas Costeiras" com três avaliadores, uma resposta dada como "Orla Costeira" e outra por "Erosão das rochas". Por outro lado houve duas respostas erradas e uma abstenção.

Na terceira questão (*Qual é o elemento do modelo utilizado para efectuar a ligação entre um conceito do domínio e o mapa?*), também de resposta livre (Figura 6.10), revelou-se uma maior ambiguidade na percepção do conceito Map Layer. Como é possível observar, houve doze respostas certas e duas respostas aproximadas à resposta certa (Layer). Também aproximado, mas não tanto, registaram-se quatro respostas, duas delas Marcadores e as outras duas Polígonos. Estas eram as camadas do mapa, ainda assim a resposta correta era Map Layer e não os objetos utilizadores para definir as camadas.

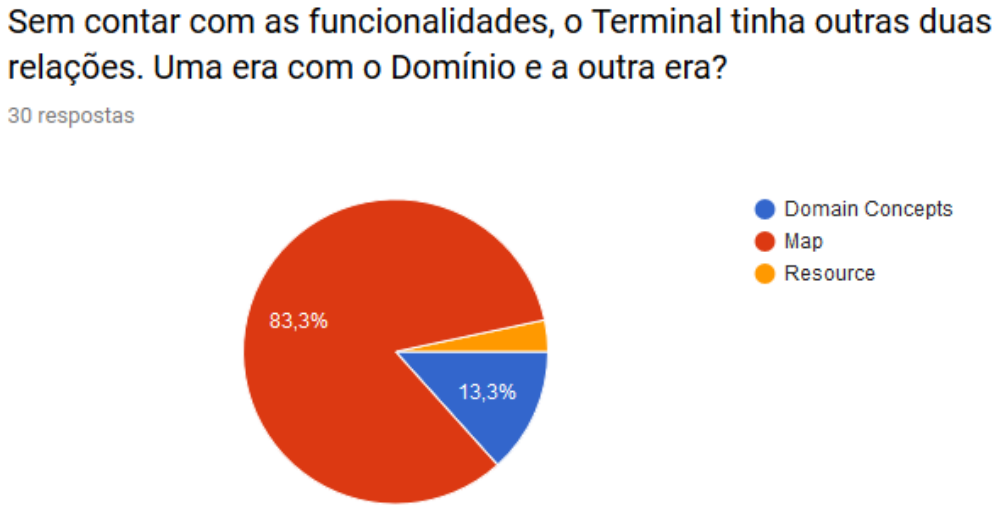


Figura 6.8: Primeira pergunta do questionário

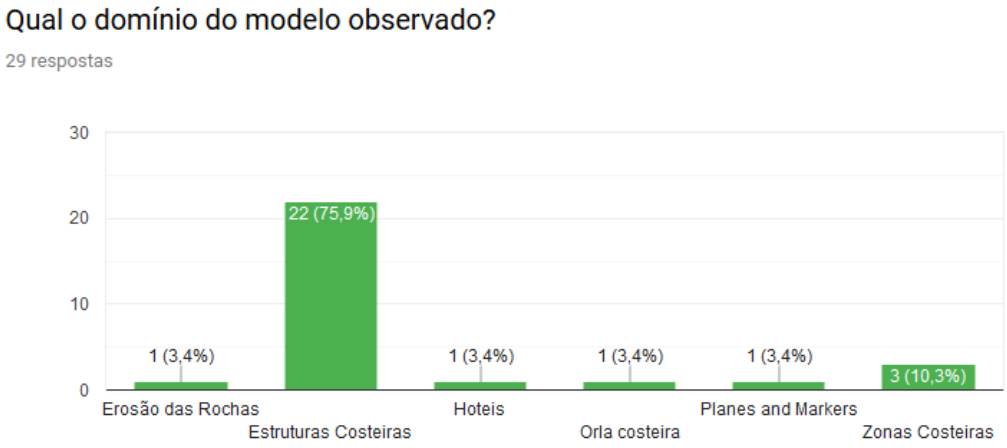


Figura 6.9: Segunda pergunta do questionário

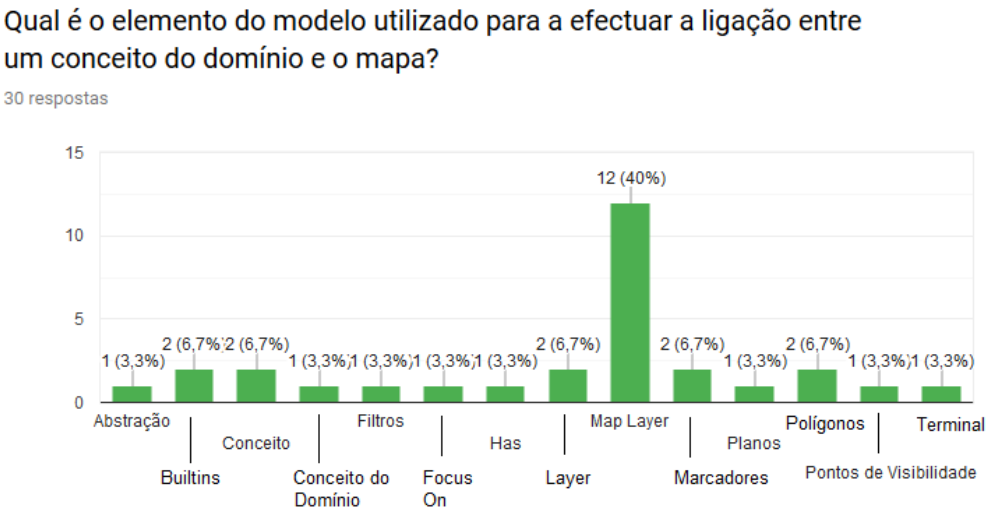


Figura 6.10: Terceira pergunta do questionário

Na quarta questão (*Liste cinco Builtins encontrados no modelo.*), de resposta livre, onde o objetivo era identificar cinco funcionalidades (Builtins) houve apenas uma resposta totalmente errada, isto é, nenhum Builtin identificado corretamente. É possível observar que a Precision (Figura 6.11), o Recall (Figura 6.12) e a Medida F (Figura 6.13), foram na sua grande maioria elevados e bastante semelhantes. Apenas quatro avaliadores não acertaram totalmente ainda que tenham acertado alguns Builtins.

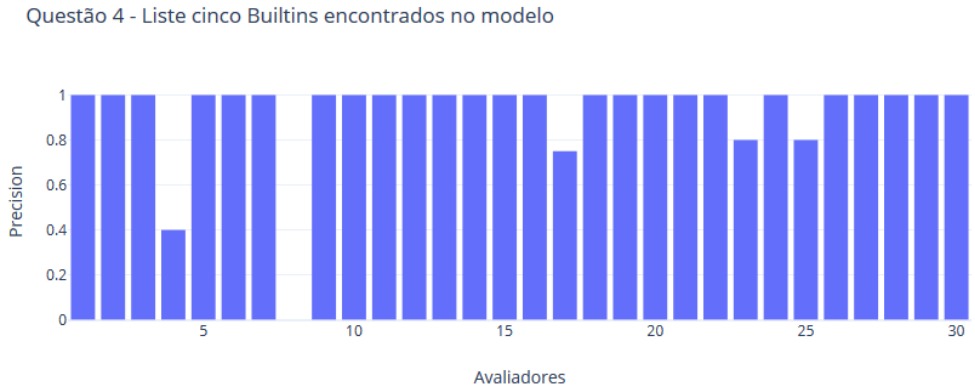


Figura 6.11: Precision, Quarta pergunta do questionário

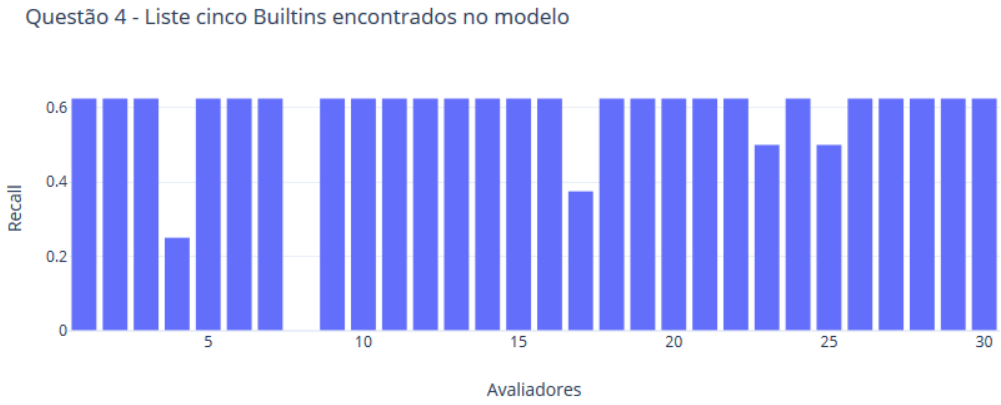


Figura 6.12: Recall, Quarta pergunta do questionário

Na quinta questão (*Qual o elemento do modelo utilizado para representar a chave de licença da API sobre o mapa escolhido?*), última de resposta livre (Figura 6.14), houve vinte e quatro avaliadores que acertaram e um que teve uma resposta aproximada (Api Function). Duas pessoas responderam o que o conceito representava (Api Key) e não o próprio nome do conceito, o que é encarado como resposta não certa. Houve também outras duas respostas erradas e uma abstenção.

Na sexta pergunta (*Qual o elemento do modelo que permite abrir uma janela de informação (InfoWindow)?*), a segunda de resposta de escolha múltipla (Figura 6.15), vinte e oito avaliadores responderam corretamente. Em contraste, outros dois, não acertaram.

Na sétima questão (*A partir da InfoWindow é possível visualzar informação, fotos e videos.*

Figura 6.13: F-Measure, Quarta pergunta do questionário

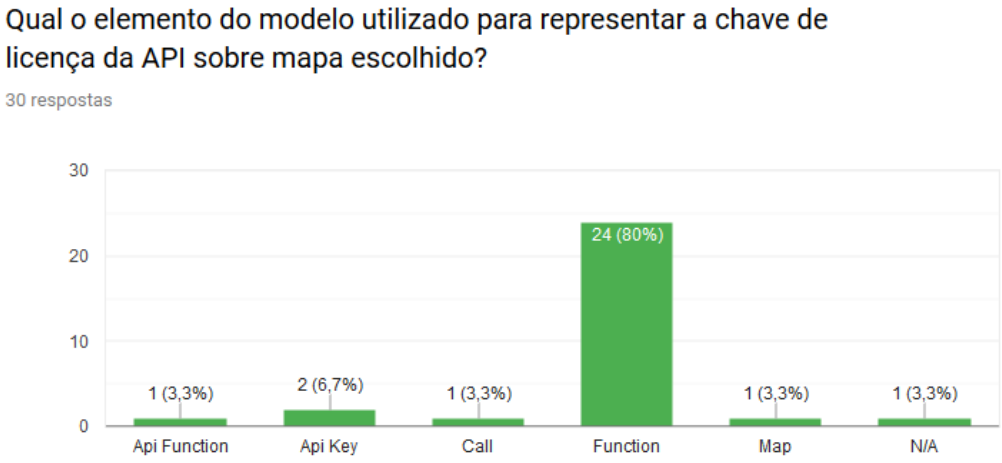


Figura 6.14: Quinta pergunta do questionário

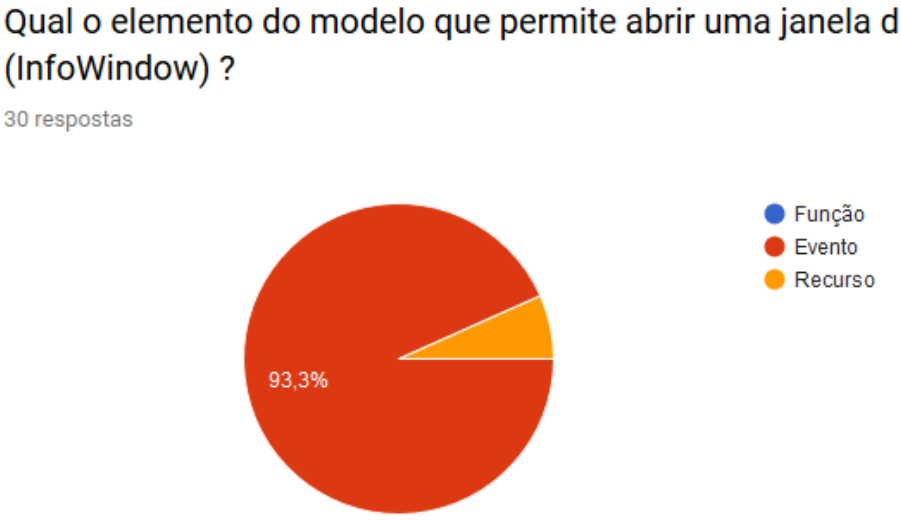


Figura 6.15: Sexta pergunta do questionário

Estes dados estão armazenados numa base de dados, que no modelo é representada por um elemento, qual?), de resposta de escolha múltipla (Figura 6.16), houve uma aprovação de noventa e sete por cento, o que equivale a vinte e nove respostas corretas e uma não certa.

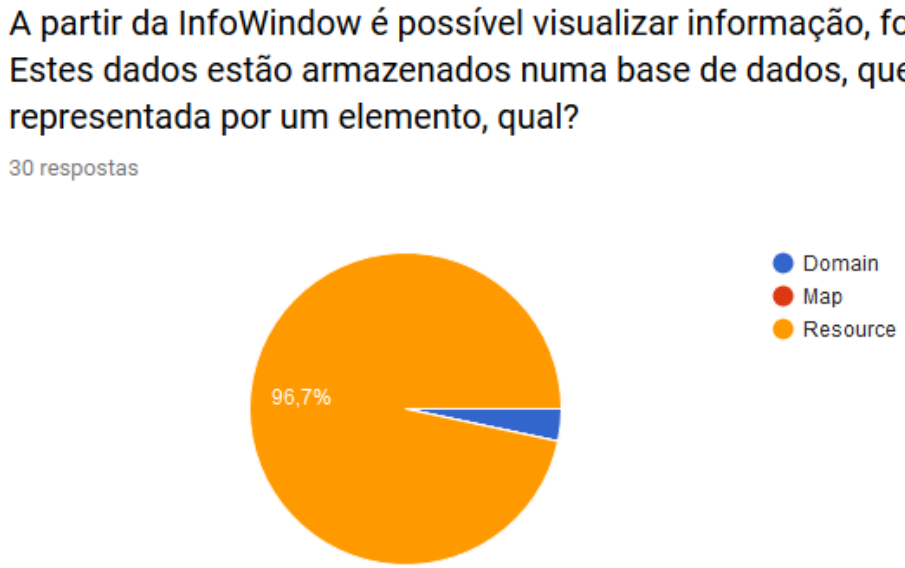


Figura 6.16: Sétima pergunta do questionário

Com vinte e sete respostas certas, foi concluída a avaliação do modelo, na última questão (*No modelo observado, o Terminal tinha duas funcionalidades, quais?*), também de resposta de escolha múltipla (Figura 6.17). Havendo também, três respostas erradas.

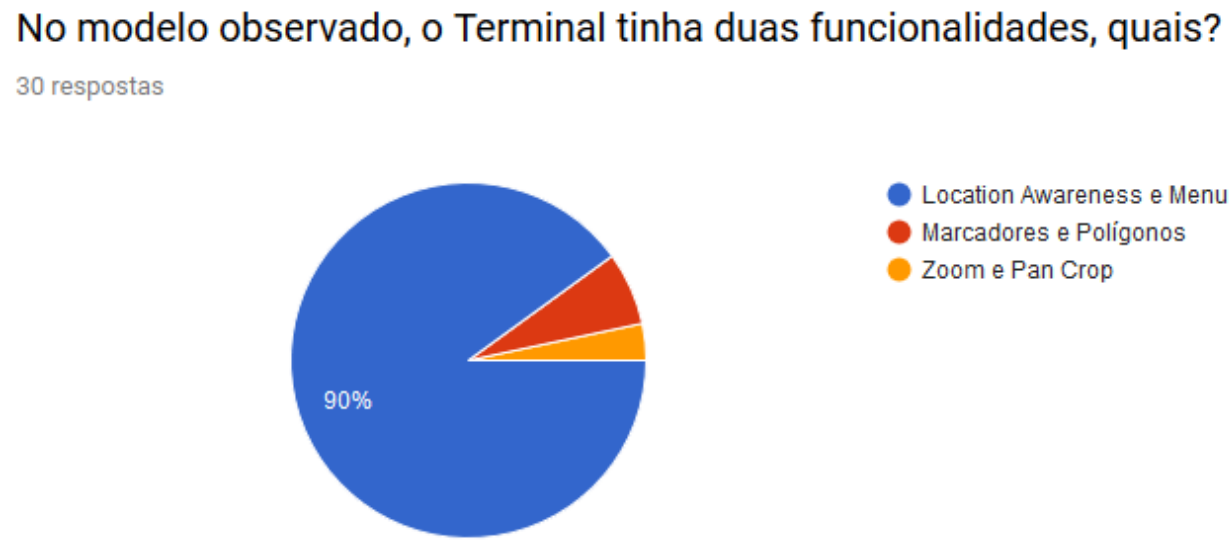


Figura 6.17: Oitava pergunta do questionário

Para terminar esta parte da avaliação com base no modelo acerca de um caso de estudo, são apresentados (Figura 6.18) os cronómetros de cada resposta por cada avaliador,

salvaguardando a sua identidade. Os tempos registados estão em segundos no caso em que aparecem com duas casas decimais e em minutos quando apresentados em três casas decimais.

Avaliador	Resposta	Tempos									Total
		1	2	3	4	5	6	7	8		
1		14.66	1.00	45.32	53.21	3.61	43.75	16.61	12.19		8.24.69
2		20.27	7.42	25.76	47.46	44.00	4.66	7.42	6.46		4.30.66
3		4.54	5.85	8.01	18.54	27.71	4.49	6.57	3.54		2.26.12
4		12.26	54.19	46.89	1.27.19	6.56	2.70	3.72	7.03		4.52.94
5		3.92	15.32	12.38	31.02	4.83	15.94	13.92	40.72		3.57.59
6		1.11.23	10.96	54.25	45.40	5.63	6.38	1.91	9.82		5.31.78
7		3.94	14.62	12.86	1.11.57	21.12	2.03	1.66	22.93		3.43.15
8		8.32	21.23	39.49	1.56.17	20.22	16.35	16.40	1.48.45		7.34.07
9		31.55	6.24	1.51.83	2.10.39	1.33.43	29.42	1.95	27.10		8.46.92
10		53.71	1.15.33	38.79	1.13.57	2.16.87	24.68	16.77	21.49		8.58.47
11		14.75	8.17	17.83	40.78	17.65	6.51	1.92	12.46		3.01.70
12		11.81	13.92	21.44	28.96	22.74	18.98	1.49	7.97		3.34.41
13		20.72	6.87	1.28.40	50.03	17.07	1.00	14.72	15.91		6.32.27
14		10.08	8.80	9.31	27.14	6.85	13.98	1.25	6.63		2.50.99
15		4.14	41.08	21.67	52.36	8.66	02.06	1.24	4.11		3.33.64
16		22.55	38.76	56.45	2.11.89	1.25.41	2.67	26.02	14.31		8.30.26
17		10.74	20.66	13.74	28.05	14.05	13.98	12.90	5.85		3.37.48
18		10.04	2.08	1.06.79	24.69	16.00	29.11	1.00	24.80		5.05.21
19		17.32	5.53	1.39.57	19.64	10.42	1.87	1.00	9.15		3.49.93
20		33.19	16.07	1.16.69	31.78	19.90	25.09	9.56	11.13		5.31.18
21		27.26	10.09	15.99	33.57	7.31	16.64	3.60	25.26		4.21.72
22		8.57	8.72	23.22	22.75	8.54	9.69	10.22	10.49		2.48.79
23		12.20	15.73	59.72	1.01.69	3.04	10.49	2.90	8.51		3.53.34
24		6.23	9.19	14.14	46.88	2.93	4.58	1.74	12.09		2.36.85
25		6.56	19.30	15.76	30.56	4.43	6.47	1.00	3.27		2.31.82
26		6.75	35.12	1.02.90	45.74	16.85	5.36	22.45	20.80		5.18.46
27		1.07.65	59.75	1.18.76	51.11	7.30	8.86	2.67	10.79		7.32.90
28		32.20	1.09.88	3.20.74	1.49.69	3.40.44	8.12	4.18	2.64		14.23.54
29		7.83	5.07	1.28.22	49.32	2.63	11.84	1.00	5.99		4.57.14
30		24.08	35.62	12.06	2.44.22	15.74	5.67	51.84	3.03		7.11.79

Figura 6.18: Cronómetro de respostas ao questionário de cada avaliador

É apresentado também (Figura 6.19), para uma melhor apreciação, os tempos médios de cada resposta a cada pergunta, referenciando também, o tipo de reposta pedida (livre ou múltipla). É observável, que as respostas que foram dadas mais rápidamente e ao mesmo tempo assertivas, foram as que eram questionadas sob forma de resposta em escolha múltipla.

			Tempos Gerais		
Nr Pergunta	**Tipo de Resposta**	**Pergunta**	**Mais rápido**	**Mais lento**	**Média**
1	Múltipla	Sem contar com as funcionalidades, o Terminal tinha outras duas relações. Uma era com o Domínio e a outra era?	3.92 seg	1.11.23 min	19.3 seg
2	Livre	Qual o domínio do modelo observado?	1.00 seg	1.15.33 min	21.42 seg
3	Livre	Qual é o elemento do modelo utilizado para a efectuar a ligação entre um conceito do domínio e o mapa?	8.01 seg	3.20.74 min	47.97 seg
4	Livre	Liste cinco Builtins encontrados no modelo	18.54 seg	2.44.22 min	57.51 seg
5	Livre	Qual o elemento do modelo utilizado para representar a chave de licença da API sobre mapa escolhido?	2.63 seg	3.40.44 min	29.06 seg
6	Múltipla	Qual o elemento do modelo que permite abrir uma janela de informação (InfoWindow) ?	1.00 seg	43.75 seg	11.78 seg
7	Múltipla	A partir da InfoWindow é possível visualizar informação, fotos e videos. Estes dados estão armazenados numa base de dados, que no modelo é representada por um elemento, qual?	1.00 seg	51.84 seg	8.65 seg
8	Múltipla	No modelo observado, o Terminal tinha duas funcionalidades, quais?	2.64 seg	1.48.45 min	15.83 seg
		Questionário completo:	2.26.12 min	14.23.54 min	5.17.79 min

Figura 6.19: Tempos médios de resposta ao questionário

De forma a conseguir representar estes tempos para transmitir uma ideia mais sensível à percepção que os avaliadores tiveram, foi desenhado um gráfico boxplot (Figura 6.20). É possível observar que, no geral, grande parte das respostas foram lançadas em menos de 60 segundos, havendo a questão 3 e 4 cujas respostas foram mais demoradas.

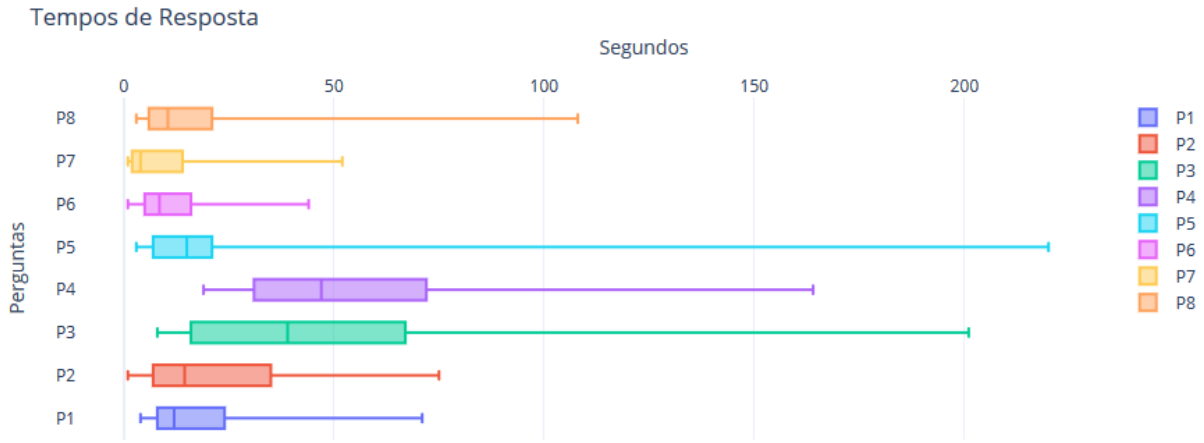


Figura 6.20: Boxplot - Tempos de resposta

Numa apreciação final à linguagem desenvolvida, requisitou-se aos avaliadores um feedback acerca da notação dos elementos da linguagem, dos ícones[1,2,3,4,5] utilizados para representar os elementos da linguagem e à capacidade de modelação dos elementos. Isto é, se os elementos presentes no editor conseguem colmatar todos os desafios que possam surgir.

A notação dos elementos da linguagem (Figura 6.21) obteve, de uma escala de um a cinco onde um equivale a "Muito Insuficiente" e cinco equivale a "Excelente", uma nota três, dezassete notas quatro e doze notas cinco.

A qualidade dos ícones da linguagem utilizados para representar os elementos do modelo (Figura 6.22) obteve, também numa escala de um a cinco onde um equivale a "Muito Fracos" e cinco equivale a "Excelente", sete notas três, dez notas quatro, doze notas cinco e uma abstenção.

Já na avaliação da suficiência dos elementos do editor (Figura 6.23), obteve-se, numa escala de um a cinco onde um equivale a "Muito Insuficientes" e cinco equivale a "Excelente", doze notas quatro e dezoito notas cinco.

[1] Ícones (Desktop; BuitIn) desenvolvidos por SimpleIcon de www.flaticon.com
[2] Ícones (Domain) desenvolvidos por Vectors Market de www.flaticon.com
[3] Ícones (Concept; Function; Event; Resource) desenvolvidos por Freepik de www.flaticon.com
[4] Ícones (Map Layer) desenvolvidos por Smashicons de www.flaticon.com
[5] Ícones (Map) desenvolvidos por Paomedia de www.brandeps.com

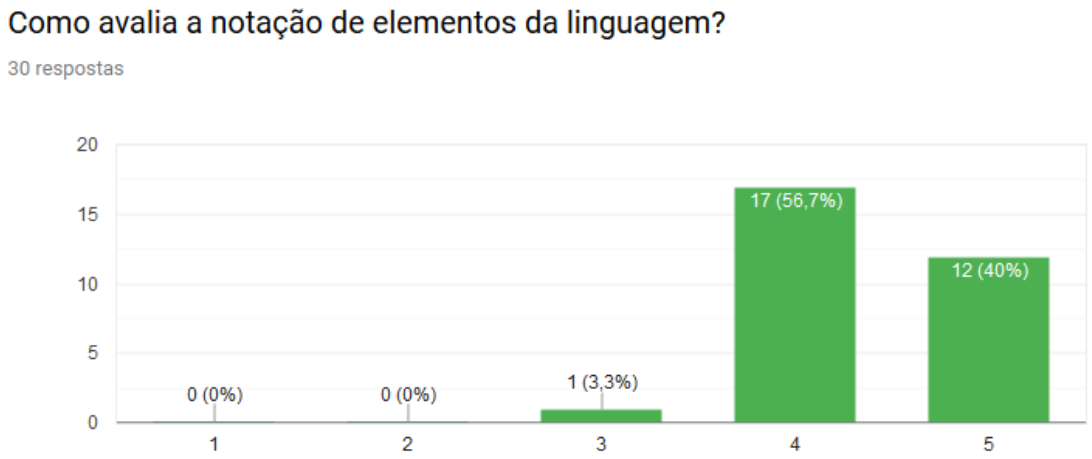


Figura 6.21: Feedback final sobre a notação dos elementos do editor

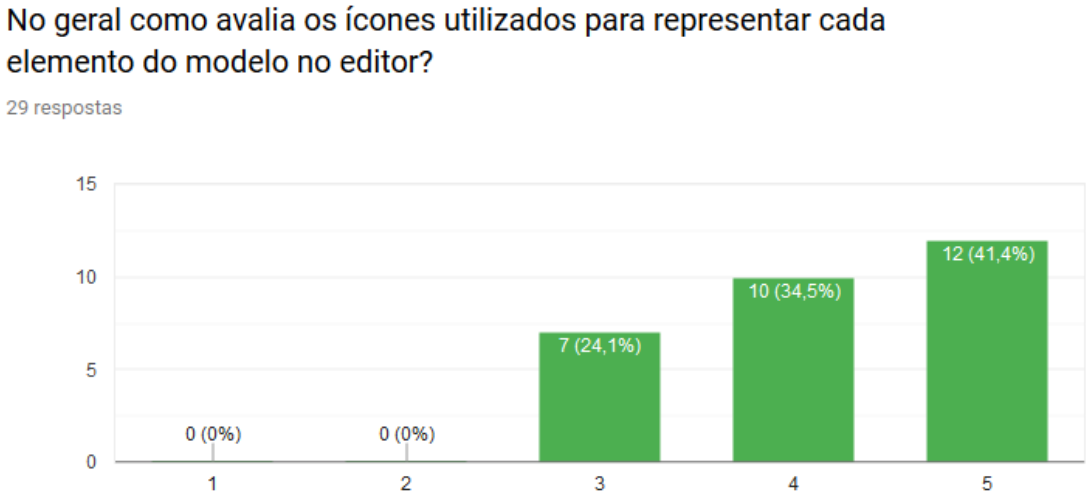


Figura 6.22: Feedback final sobre ícones utilizados para representar os elementos do editor

Por fim, os avaliadores foram também questionados acerca da avaliação da dificuldade de aprendizagem da linguagem e na avaliação da linguagem de uma forma geral. Acerca da questão sobre a dificuldade de aprendizagem, dos elementos da linguagem (Figura 6.24), obteve-se numa escala de um a cinco onde um equivale a "Muito Difícil" e cinco equivale a "Muito Fácil", uma nota um, duas notas três, dezassete notas quatro e dez notas cinco.

Quanto à avaliação da linguagem de uma forma geral (Figura 6.25), numa escala de um a cinco onde um equivale a "Muito Fraca" e cinco equivale a "Excelente", obteve-se quinze notas quatro e quinze notas cinco.

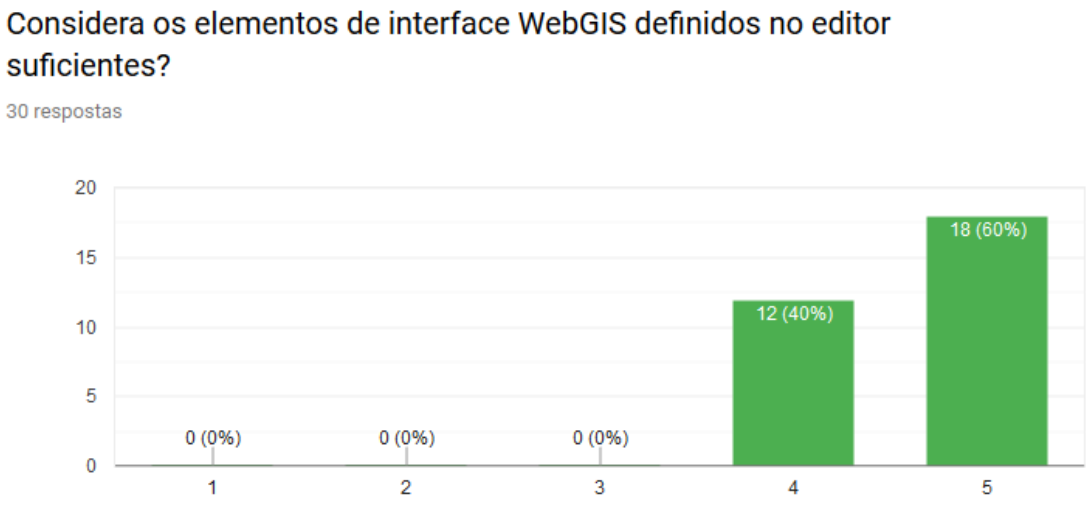


Figura 6.23: Feedback final sobre a suficiência dos elementos do editor

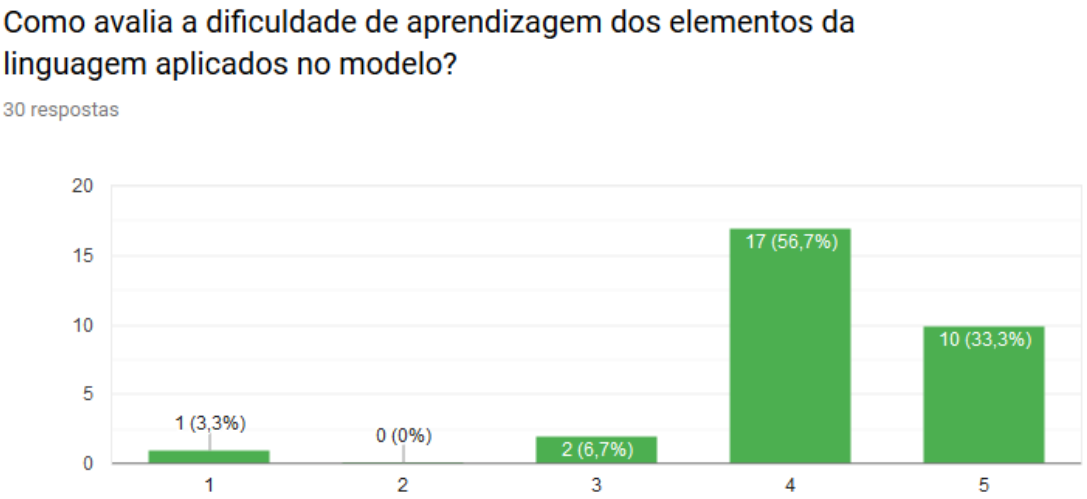


Figura 6.24: Feedback final sobre a dificuldade de aprendizagem da linguagem

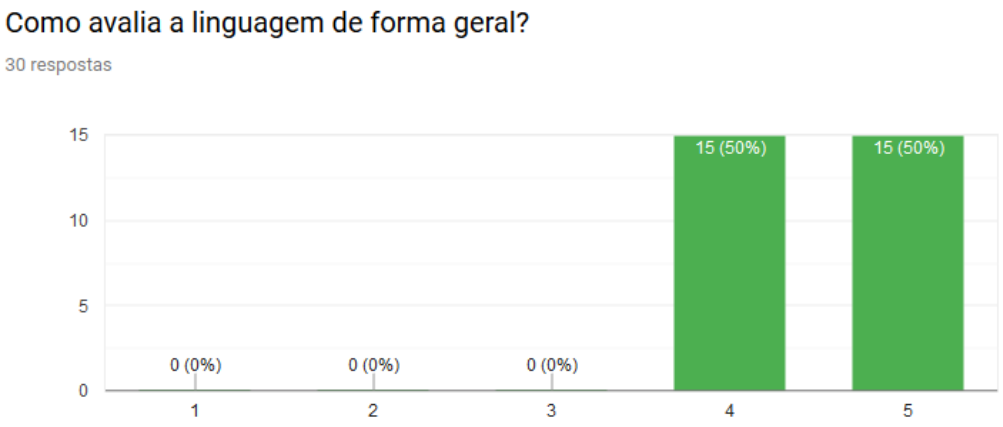


Figura 6.25: Feedback final sobre a linguagem de uma forma geral

6.5 Discussão

Com base nos resultados obtidos perante esta avaliação e tendo em conta também os cronómetros que complementam a avaliação das respostas dadas, acreditamos que o trabalho desenvolvido é muito positivo. Os participantes revelaram na sua grande maioria, antes de realizarem a avaliação ter tido contacto quer no desenvolvimento quer na utilização de aplicações WebGIS. Houve também uma clara evidência, de contacto na utilização de Linguagens de Dominío Específico e algum no seu desenvolvimento. Face à avaliação do modelo, os resultados também foram muito positivos. Todas as questões que solicitavam resposta de escolha múltipla (Questões 1, 6, 7 e 8), tiveram uma taxa de aprovação superior a 80%. As questões de resposta livre, variaram um pouco mais. Ainda assim, a taxa de aprovação de três destas quatro perguntas foi superior a 75%. Já uma destas quatro perguntas revelou na sua resposta, existir alguma confusão na percepção do avaliador. Nomeadamente na terceira pergunta (*Qual o elemento do modelo utilizado para efectuar a ligação entre um conceito do domínio e o mapa?*), houve uma disparidade nas respostas dadas. Apenas doze pessoas acertaram corretamente, seis pessoas lançaram respostas próximas da correta, ainda que não corresponde-se completamente ao que era pretendido e as outras doze responderam erradamente. No que diz respeito ao cronómetro das respostas de escolha múltipla, a média do tempo de resposta em todas elas é inferior a 20 segundos. Já nas perguntas de resposta livre, que naturalmente demoram um pouco mais tempo, obteve-se uma média de resposta em todas elas, inferior a 60 segundos. Observamos aprovação e rapidez na resposta, na Avaliação ao Modelo. Nas perguntas finais, não cronometradas, acerca da linguagem no geral, há uma aceitação muito forte por parte dos avaliadores. Todos consideraram a notação dos elementos da linguagem e os ícones utilizados, positivos. Houve uma aprovação muito forte sobre a suficiência dos conceitos da linguagem. No contexto da aprendizagem destes conceitos, apenas 3 pessoas não consideraram uma aprendizagem fácil. E de uma forma geral todos os participantes consideraram a linguagem, "Muito Boa" ou "Excelente". Houve uma percepção geral positiva, o que revela uma aplicabilidade elevada do protótipo levantado. Há uma convicção de que este trabalho vai, de alguma forma, melhorar a interacção entre o utilizador e o desenvolvimento de aplicações WebGIS.

6.6 Ameaças à validade

Embora exista satisfação da nossa parte pelos resultados obtidos, há um sentido de responsabilidade face a alguns aspectos sobre a avaliação que levantam algumas questões face à sua validade.

No que diz respeito aos participantes da avaliação, apenas foram convidados engenheiros informáticos. Inicialmente houve uma tentativa de reunir apenas pessoal com alguma experiência e contacto com aplicações WebGIS, depois, com pessoal com alguma experiência e contacto em linguagens de domínio específico e por fim, pessoal com experiência

e contacto em engenharia de requisitos. No que diz respeito à experiência de desenvolvimento, tanto de aplicações WebGIS como linguagens de domínio específico, verificou-se que a amostra recolheu dezanove pessoas sem experiência alguma em desenvolvimento de aplicações WebGIS e também dezanove pessoas sem experiência de desenvolvimento de linguagens de domínio específico. Estes indicadores foram disponibilizados na secção 6.3. Sessenta e três por cento dos participantes não tinham experiência de desenvolvimento quer WebGIS quer DSL. Reconhecemos assim que esta pode ser uma ameaça à validade da avaliação. Há que salientar também que a experiência dos participantes que já tinham desenvolvido este tipo de aplicações e linguagens era maioritariamente académica e não profissional. Este facto pode também ser considerado como uma ameaça à validade da avaliação.

À parte dos indicadores da experiência, houve uma atenção quanto ao número de participantes. Para obter uma amostra minimamente significativa e resultados mais consistentes, definimos que este número deveria ser pelo menos de trinta pessoas. Sabendo que foram convidados entre noventa a cem pessoas da área de informática e que apenas trinta acederam ao convite, sentimos alguma frustração em não conseguir reunir uma amostra ainda maior. Esta poderá ser outra ameaça à validação. Dado que a amostra requer que todas as avaliações fossem sobre o mesmo modelo, não houve uma experiência de modelação com um elevado número de aplicações WebGIS de domínios distintos. Face a isto, não descartamos a possibilidade de existir algum domínio, ou aplicação WebGIS não explorada, onde os conceitos do domínio que são utilizados no editor para modelar a aplicação não sejam suficientes. Consideramos assim, que esta possibilidade é uma ameaça à validação da avaliação.

Por último, o facto da avaliação ser direccionada mais para a parte da codificação do editor da linguagem e não tanto para a codificação da linguagem em geral. Este foco na codificação do editor, pode trazer problemas de compreensão da linguagem que não são propriamente da mesma. Consideramos que esta pode ser uma ameaça à validação da avaliação da linguagem.

6.7 Sumário

Este capítulo aborda todo o processo de avaliação do trabalho desenvolvido. São apresentados os materiais de avaliação necessários que foram utilizados e a sua importância. Sem revelar a identidade, é referido o grupo de participantes que foi escolhido face às suas áreas de formação, idades e nacionalidades. São referidas algumas métricas usadas no processo de análise das respostas obtidas por parte dos participantes. É apresentado um conjunto de dados estatísticos obtidos com base nas respostas dos participantes e levantada uma discussão acerca desses mesmos dados que mostram a consistência do trabalho desenvolvido. Por fim são levantadas alguns factos acerca da avaliação que podem ser considerados uma ameaça à sua validade.

Capítulo 7

Conclusão

Este capítulo transmite um sumário sobre todo o trabalho desenvolvido neste projeto de dissertação.
São abordados, o ponto de partida com a definição do problema e o seu ambiente.
A solução produzida, os seus frutos e as suas limitações.
Por fim é também apresentada uma breve reflexão sobre as possíveis direcções que este projeto pode tomar no futuro.

O objetivo principal deste projeto era diminuir a distância cognitiva entre os utilizadores e os produtores de sistemas WebGIS. Aproximar as partes interessadas ao redor do desenvolvimento de aplicações WebGIS, implicitamente diminuirá falhas de comunicação entre um comprador de WebGIS e o seu produtor. Isto leva a que não sejam levantados problemas técnicos nas fases iniciais do desenvolvimento, que dão origem a custos dispendiosos aos interessados no sistema a desenvolver. O método para alcançar este objetivo, passou pelo desenvolvimento de uma Linguagem de Modelação de Requisitos direccionada para entidades sem habilidades de desenvolvimento de aplicações WebGIS. Antes de ser iniciado o processo de desenvolvimento da linguagem, realizou-se um estudo em torno de linguagens de modelação de domínio específico e aplicações WebGIS. Este estudo, tinha a finalidade de reunir propriedades comuns a linguagens de modelação e aplicações WebGIS. Após a recolha e identificação de características similares em várias linguagens e aplicações geográficas, procedeu-se ao desenvolvimento da Linguagem de Modelação de Requisitos para Interface WebGIS. Esta, foi desenvolvida a partir da construção do Metamodelo que serve de alicerce da linguagem. Neste Metamodelo definiram-se também, os conceitos necessários para efetuar modelação de interfaces WebGIS consistentes. A partir daqui, foi desenvolvida a interface da linguagem e os seus orgãos complementares, que permitem ao utilizador fazer a modelação e verificar a consistência do modelo criado. Terminado o desenvolvimento da linguagem, aplicou-se um

caso de estudo para modelar uma aplicação WebGIS. Após esta abordagem efectuaram-se avaliações acerca do modelo definido, com várias pessoas ligadas às áreas de desenvolvimento e utilização de software WebGIS. O balanço final do processo de avaliações da linguagem foi extremamente positivo e encorajou as entidades ligadas a esta dissertação a chegar até aqui.

7.1 Contribuições

Foram identificadas como contribuições, as seguintes:

- Uma nova forma de elicitar requisitos para interface WebGIS, uma solução para aproximar partes especializadas e partes não especializadas no desenvolvimento de aplicações WebGIS.
- Espera-se que o editor, facilite a modelação de requisitos de interfaces de aplicações WebGIS e acelere seu o desenvolvimento.
- A suavização no desenvolvimento inicial destas aplicações, quando existem várias partes interessadas envolvidas é a principal contribuição desta dissertação.

7.2 Limitações

Foram identificadas algumas limitações, nomeadamente:

- O ambiente específico (Epsilon) para inicializar esta ferramenta e a incapacidade de exportar os materiais desenvolvidos e redefini-los noutro ambiente é uma exclusividade da linguagem restritiva.
- A inexistência de uma habilidade do ambiente em que foi criado o Metamodelo da linguagem para focar uma determinada relação neste último. Para metamodelos de grandes dimensões, se houver necessidade de se efetuar alguma alteração, a tarefa pode ser dolorosa.
- A limitação dos ícones das metaclasses(conceitos do editor) necessitarem de ser externas à linguagem e não existir possibilidade de criar imagens diretamente no ambiente. Isto é, as imagens que representam os conceitos, já existiam antes da linguagem começar a ser desenvolvida, a impossibilidade de criar imagens dentro da linguagem levou a que se recorre-se a fontes de ícones externas.
- O motor de verificação e correção de conflitos que é utilizado para verificar a consistência do modelo, é em parte limitado. A verificação, consegue detectar incorreções nos atributos das metaclasses e tem capacidade para oferecer soluções rápidas para este tipo de problemas. Já nas relações entre metaclasses, a verificação consegue detetar ausência de relações entre metaclasses, porém não consegue oferecer soluções rápidas nesta parte.

7.3 Desenvolvimentos futuros

A ampliação de novos conceitos na linguagem, não só direccionados para a interface de aplicações WebGIS mas também para o seu background, poderá ser um dos caminhos/objetivos futuros deste projeto. A distribuição da linguagem para habilitar a sua utilização noutras plataformas diferentes em contraste com a limitação do ambiente em que foi desenvolvida, poderá ser também um dos rumos do horizonte. Outro caminho a seguir, poderá ser o desenvolvimento da habilidade para gerar código específico da aplicação WebGIS modelada. Isto, permitirá uma aceleração no processo de desenvolvimento de aplicações WebGIS.

Bibliografia

[1] S. Lammes e C. Wilmott. "Mapping the city, playing the city: Location-based apps as navigational interfaces." Em: *Convergence: The International Journal of Research into New Media Technologies* (2016). ISSN: 1354-8565.

[2] M. Worboys e M. Duckham. *GIS: A Computing Perspective, 2Nd Edition*. Boca Raton, FL, USA: CRC Press, Inc., 2004. ISBN: 0415283752.

[3] K. Laudon e J. Laudon. *Management Information Systems: Managing the Digital Firm*. Always Learning. Pearson Education Limited, 2013. ISBN: 9780273789970. URL: https://books.google.pt/books?id=uSuMlwEACAAJ.

[4] K. C. Clarke. "Advances in geographic information systems". Em: *Computers, environment and urban systems* 10.3-4 (1986), pp. 175–184.

[5] C. Reviews. *Experiencing MIS: Computer science, Information technology*. Cram101, 2016. ISBN: 9781490289427. URL: https://books.google.pt/books?id=Qw_MAwAAQBAJ.

[6] P. Longley, M. Goodchild, D. Maguire e D. Rhind. *Geographic Information Systems and Science*. Wiley, 2005. ISBN: 9780470870020. URL: https://books.google.pt/books?id=-w_FYTfaxRUC.

[7] J. Gouveia, F. Branco, A. Rodrigues e N. Correia. "Travelling Through Space and Time in Lisbon ' s Religious Buildings". Em: *Proceedings of Digital Heritage 2015 - DH'15*. Granada, Spain: IEEE, 2015.

[8] NOVA-LINCS. *LxConventos*. 2015. URL: lxconventos.cm-lisboa.pt (acedido em 01/07/2018).

[9] P. Longley, M. Goodchild, D. Maguire e D. Rhind. *Geographic Information Science and Systems*. Wiley, 2015. ISBN: 9781118676950. URL: https://books.google.pt/books?id=C_EwBgAAQBAJ.

[10] J. A. Gaspar. *Cartas e projecções cartográficas*. Lidel, 2005.

[11] J. de Matos. *Fundamentos de informação geográfica, 4º Edição*. Lidel, 2001. ISBN: 9789727571857.

[12] P. Fu e J. Sun. *Web GIS: Principles and Applications*. Esri Press, 2010. ISBN: 158948245X, 9781589482456.

[13] A. Ibraheem e A. Daham. "Development and Use of Large-Scale Land Information System (LIS) by Using Geographic Information System (GIS) and Field Surveying". Em: 8 (2011), pp. 29–43.

[14] P. Wegner. "Interoperability". Em: *ACM Comput. Surv.* 28.1 (mar. de 1996), pp. 285–287. ISSN: 0360-0300. DOI: 10.1145/234313.234424. URL: http://doi.acm.org/10.1145/234313.234424.

[15] B. P. Walawender e M. W. Davis. "6A. 1 Development of Web-Based GIS Applications for decision support and situational awareness." Em: (2010).

[16] M Adnan, A Singleton e P Longley. "Developing efficient web-based GIS applications". Em: (2010).

[17] C.-y. Qu, H. Ye e Z. Liu. "Application of WebGIS in seismological study". Em: *Acta Seismologica Sinica* 15.1 (jan. de 2002), pp. 99–106. DOI: 10.1007/s11589-002-0052-8. URL: https://doi.org/10.1007/s11589-002-0052-8.

[18] G. Held[1], A. Rahman e S. Zlatanova. "Web 3D GIS for Urban Environments". Em: (set. de 2004).

[19] A. Geraci, F. Katki, L. McMonegal, B. Meyer e H. Porteous. *IEEE Standard Computer Dictionary. A Compilation of IEEE Standard Computer Glossaries*. 1991. DOI: 10.1109/IEEESTD.1991.106963.

[20] I. Sommerville. *Software Engineering*. 10th. Pearson, 2015. ISBN: 0133943038, 9780133943030.

[21] P. Zave. "Classification of Research Efforts in Requirements Engineering". Em: *ACM Comput. Surv.* 29.4 (dez. de 1997), pp. 315–321. ISSN: 0360-0300. DOI: 10.1145/267580.267581. URL: http://doi.acm.org/10.1145/267580.267581.

[22] I. Sommerville. *Software Engineering*. 9th. USA: Addison-Wesley Publishing Company, 2010. ISBN: 0137035152, 9780137035151.

[23] J. Marasco. *What Is the Cost of a Requirement Error?* 2007. URL: https://www.stickyminds.com/article/what-cost-requirement-error.

[24] M. Brambilla, J. Cabot, M. Wimmer e L. Baresi. *Model-Driven Software Engineering in Practice: Second Edition*. Synthesis Lectures on Software Engineering. Morgan & Claypool Publishers, 2017. ISBN: 9781627056953. URL: https://books.google.pt/books?id=iq2zDgAAQBAJ.

[25] T. Kühne. "Matters of (Meta-) Modeling". Em: *Software Systems Modeling* 5 (dez. de 2006), pp. 369–385. DOI: 10.1007/s10270-006-0017-9.

[26] T. Kühne. "What is a Model?" Em: *Language Engineering for Model-Driven Software Development*. Ed. por J. Bezivin e R. Heckel. Dagstuhl Seminar Proceedings 04101. Dagstuhl, Germany: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2005. URL: http://drops.dagstuhl.de/opus/volltexte/2005/23.

[27] S. Kelly e J.-P. Tolvanen. *Domain-specific modeling: enabling full code generation*. John Wiley & Sons, 2008.

[28] T. Mens e P. Van Gorp. "A Taxonomy of Model Transformation". Em: *Electron. Notes Theor. Comput. Sci.* 152 (mar. de 2006), pp. 125–142. ISSN: 1571-0661. DOI: 10.1016/j.entcs.2005.10.021. URL: http://dx.doi.org/10.1016/j.entcs.2005.10.021.

[29] E. Syriani, J. Gray e H. Vangheluwe. "Modeling a Model Transformation Language". Em: *Domain Engineering: Product Lines, Languages, and Conceptual Models*. Ed. por I. Reinhartz-Berger, A. Sturm, T. Clark, S. Cohen e J. Bettin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 211–237. ISBN: 978-3-642-36654-3. DOI: 10.1007/978-3-642-36654-3_9. URL: https://doi.org/10.1007/978-3-642-36654-3_9.

[30] M. Stephan e A. Stevenson. "A comparative look at model transformation languages". Em: *Software Technology Laboratory at Queens University* (2009).

[31] E. Biermann, K. Ehrig, C. Ermel, C. Köhler e G. Taentzer. "The EMF model transformation framework". Em: *International Symposium on Applications of Graph Transformations with Industrial Relevance*. Springer. 2007, pp. 566–567.

[32] J. den Haan. *MDE – Model Driven Engineering – reference guide*. 2018. URL: http://www.theenterprisearchitect.eu/blog/2009/01/15/mde-model-driven-engineering-reference-guide/.

[33] M. Angelaccio, A. Krek e A. D'Ambrogio. "A Model-driven Approach for Designing Adaptive Web GIS Interfaces". Em: *Information Fusion and Geographic Information Systems*. Ed. por V. V. Popovich, C. Claramunt, M. Schrenk e K. V. Korolenko. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 137–148. ISBN: 978-3-642-00304-2.

[34] M. Escalona e N. Koch. "Metamodeling the Requirements of Web Systems". Em: 1 (2007), pp. 267–280.

[35] M. Urbieta, A. Oliveira, J. Araújo, A. Rodrigues, A. Moreira, S. E. Gordillo e G. Rossi. "Web-GIS models: accomplishing modularity with aspects". Em: *ISSE* 10.1 (2014), pp. 59–75. DOI: 10.1007/s11334-013-0206-y. URL: https://doi.org/10.1007/s11334-013-0206-y.

[36] F. Ananda, D. N. Kuria e M. M. Ngigi. "Towards a new Methodology For Web GIS Development". Em: (2016).

[37] S. D. Martino, F. Ferrucci, L. Paolino, M. Sebillo, G. Vitiello e G. Avagliano. "A WebML-based Visual Language for the Development of Web GIS Applications". Em: *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2007)*. IEEE Computer Society. Set. de 2007, pp. 209–214. DOI: 10.1109/VLHCC.2007.53.

[38] M. Almas, A. Rodrigues, N. Correia, A. Queiroz e D. Alves. "LITESCAPE.PT - A PORTUGUESE LITERARY ATLAS". English. Em: *Congresso de Humanidades Digitais em Portugal*. Faculdade de Ciencias Sociais e Humanas, UNL, Lisboa, Portugal. Out. de 2015.

[39] J. Gouveia, F. Branco, A. Rodrigues e N. Correia. "Travelling through space and time in Lisbon's religious buildings". Em: *2015 Digital Heritage*. Vol. 1. Set. de 2015, pp. 407–408. DOI: 10.1109/DigitalHeritage.2015.7413916.

[40] A. Maia, A. Rodrigues, R. Lemos, R. Capitão e C. Fortes. Em: *GISTAM 2017 - Proceedings of the 3rd International Conference on Geographical Information Systems Theory, Applications and Management* (2017), pp. 1–23.

[41] Tomahock. *Fogos.pt*. 2018. URL: https://fogos.pt/.

[42] TeleGeography. *Submarine Cable Map*. 2017. URL: https://www.submarinecablemap.com/.

[43] G. Migueis. "Dissertação de Mestrado: Linguagem para Modelação de Requisitos Emocionais na Indústria de Videojogos". Em: Departamento de Informática, Universidade Nova de Lisboa. Caparica, 2017.

[44] R. Monteiro, J. Araujo, V. Amaral e P. Patricio. "Mdgore: Towards Model-Driven and Goal-Oriented Requirements Engineering". Em: *2010 18th IEEE International Requirements Engineering Conference*. Set. de 2010, pp. 405–406. DOI: 10.1109/RE.2010.60.

[45] T. E. Foundation. *ATL - a model transformation technology*. 2018. URL: http://www.eclipse.org/atl/.

[46] F. Wanderley e J. Araujo. "Generating goal-oriented models from creative requirements using model driven engineering". Em: *2013 3rd International Workshop on Model-Driven Requirements Engineering (MoDRE)*. Jul. de 2013, pp. 1–9. DOI: 10.1109/MoDRE.2013.6597258.

[47] S. G. International. "The chaos report". Em: *United States of America* (2015).

[48] M. J. Escalona e G. Aragón. "NDT. A model-driven approach for web requirements". Em: *IEEE Transactions on software engineering* 34.3 (2008), pp. 377–390.

[49] G. Loniewski, E. Borde, D. Blouin e E. Insfran. "Model-driven requirements engineering for embedded systems development". Em: *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on*. IEEE. 2013, pp. 236–243.

[50] A. Maia, A. Rodrigues, R. Lemos, R. Capitão e C. J. Fortes. "Developing a Responsive Web Platform for the Systematic Monitoring of Coastal Structures". Em: *Geographical Information Systems Theory, Applications and Management*. Ed. por L. Ragia, R. Laurini e J. G. Rocha. Cham: Springer International Publishing, 2019, pp. 176–197. ISBN: 978-3-030-06010-7.

[51] M. Santos, C. Gralha, M. Goulao, J. Araújo, A. Moreira e J. Cambeiro. "What is the impact of bad layout in the understandability of social goal models?" Em: *Requirements Engineering Conference (RE), 2016 IEEE 24th International*. IEEE Computer Society. 2016, pp. 206–215.

ANEXO I

Apresentação de avaliação

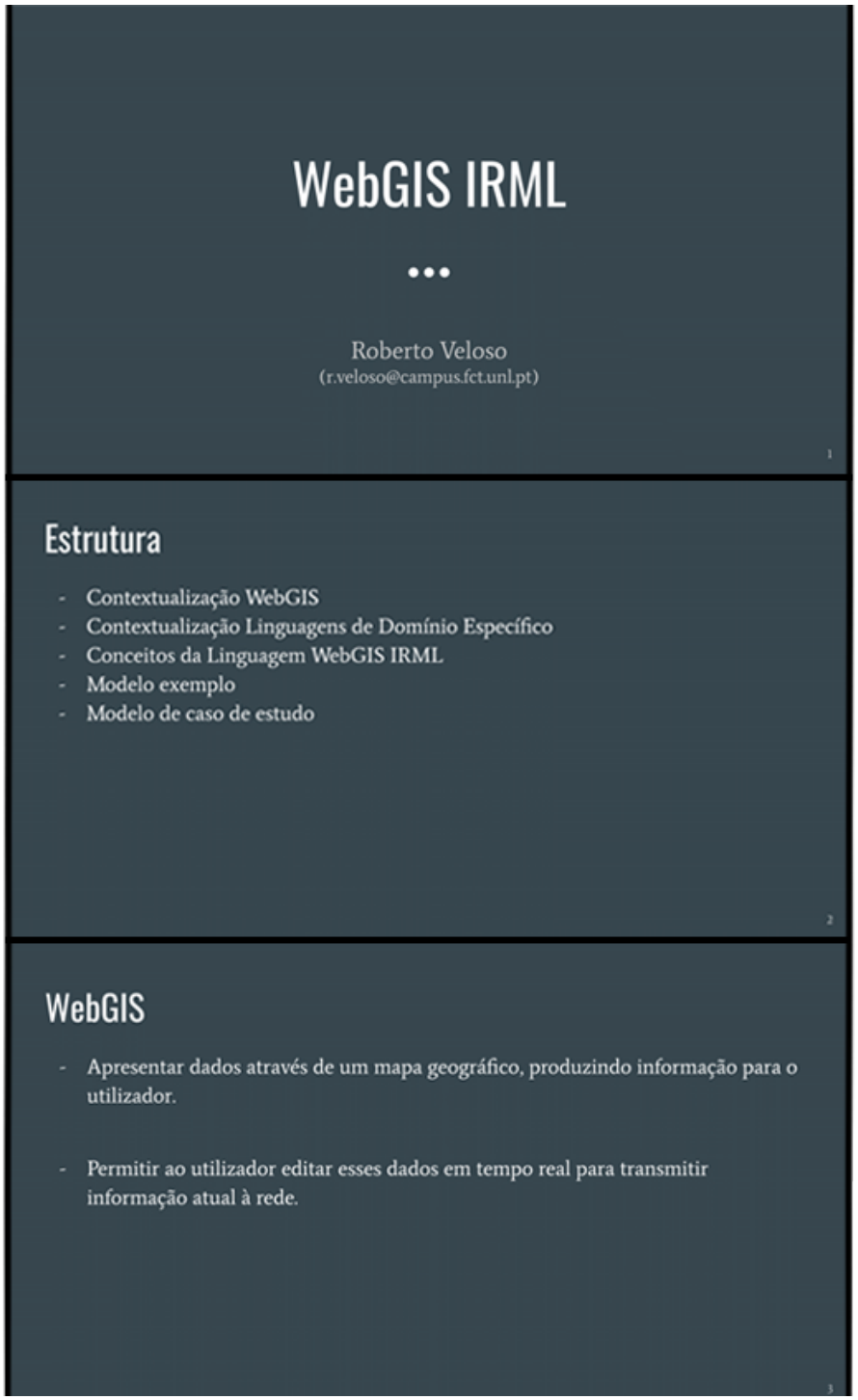


Figura I.1: Apresentação realizada aos avaliadores, slides 1, 2 e 3

Figura I.2: Apresentação realizada aos avaliadores, slides 4, 5 e 6

Figura I.3: Apresentação realizada aos avaliadores, slides 7, 8 e 9

Elemento	Notação	Definição
Resource		Tipos de repositórios onde se vão guardar dados do sistema, do mapa ou outros necessários ao correto funcionamento da aplicação (ex: Base de dados).
MAP		O mapa base que a aplicação vai utilizar (ex: OpenStreetMap (google-Maps)).
MAP LAYER		Camada de informação espacial a integrar com o mapa base. (ex: Marcadores, Polígonos).

Elemento	Notação	Definição
BUILTIN		Funcionalidades disponíveis, para reutilização em qualquer aplicação. (ex: p/Terminal: relógio; p/Mapa: marcador).
FUNCTION		Funções a desenvolver à aplicação (ex: p/terminal: getSystemTime(); p/mapa: InputData(); DeleteData();).
EVENT		Eventos que despoletam a aplicação das funcionalidades. (ex: click; scroll).

Aplicação dos conceitos num modelo exemplo

Figura I.4: Apresentação realizada aos avaliadores, slides 10, 11 e 12

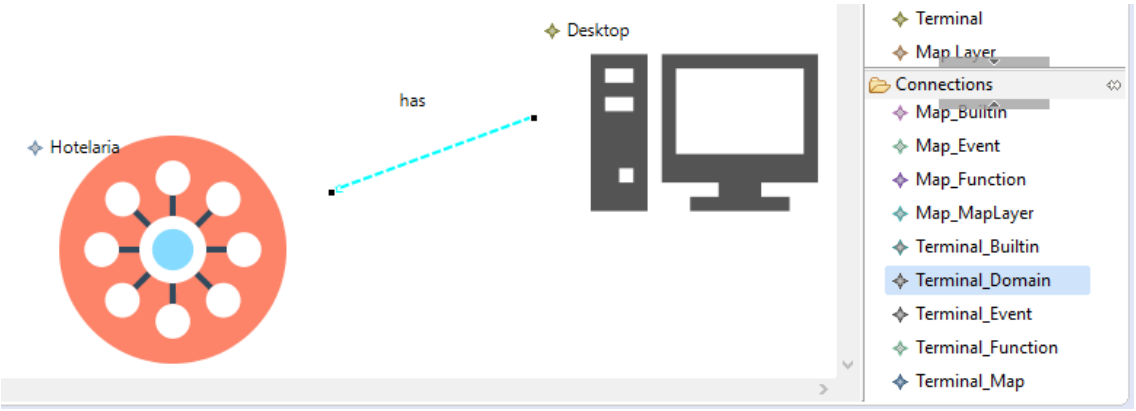


Figura I.5: Apresentação realizada aos avaliadores, slide 13

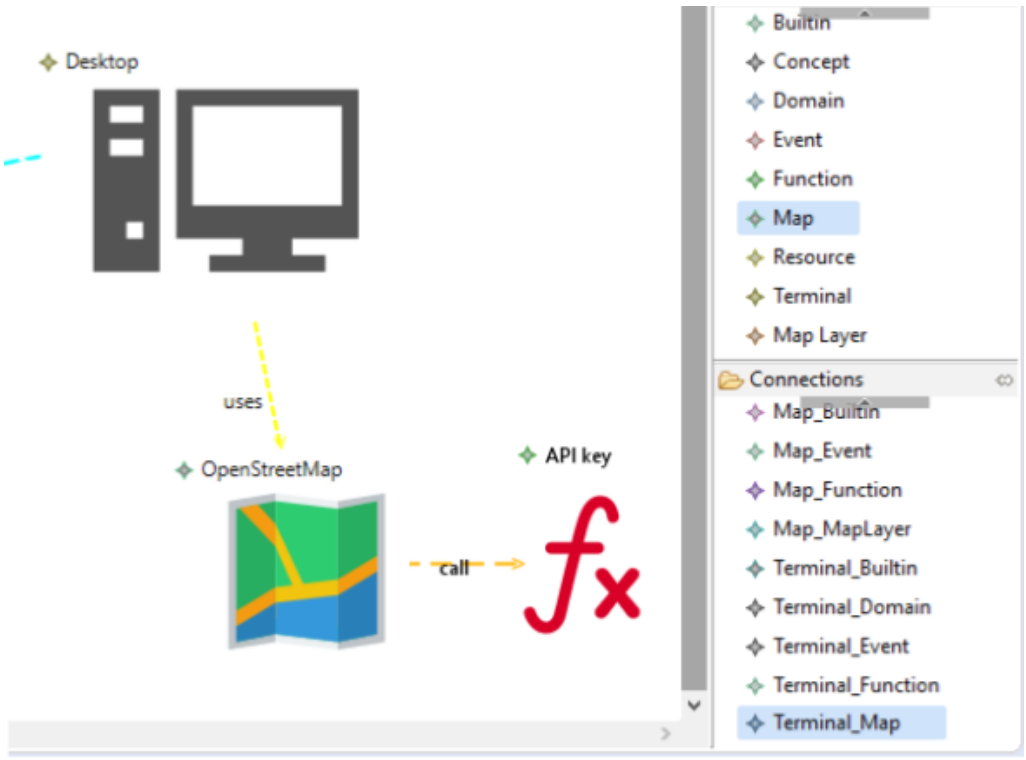


Figura I.6: Apresentação realizada aos avaliadores, slide 14

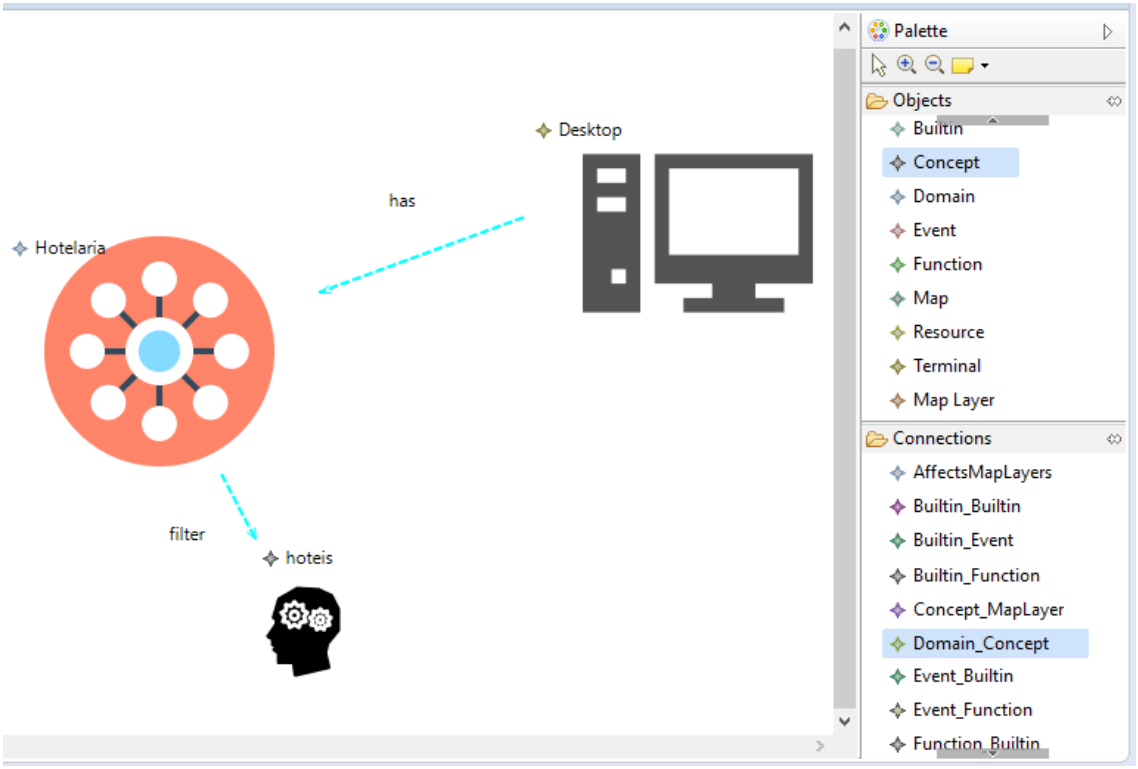


Figura I.7: Apresentação realizada aos avaliadores, slide 15

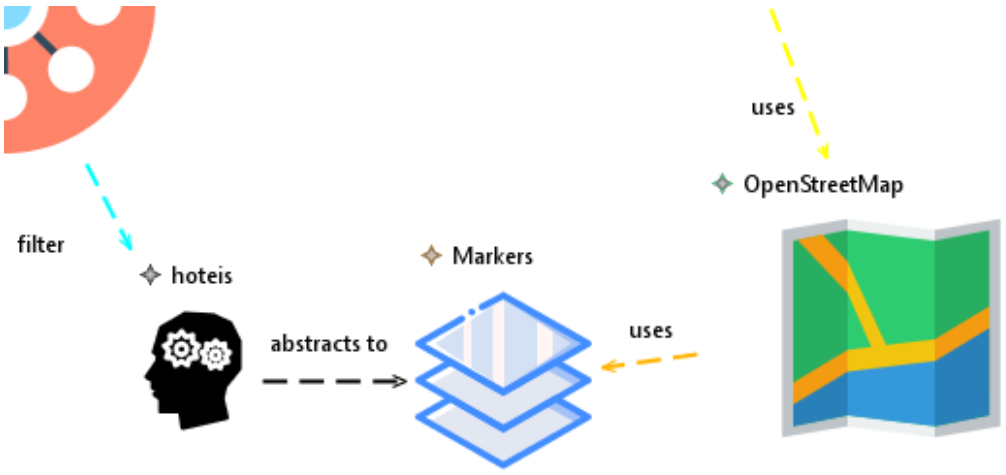


Figura I.8: Apresentação realizada aos avaliadores, slide 16

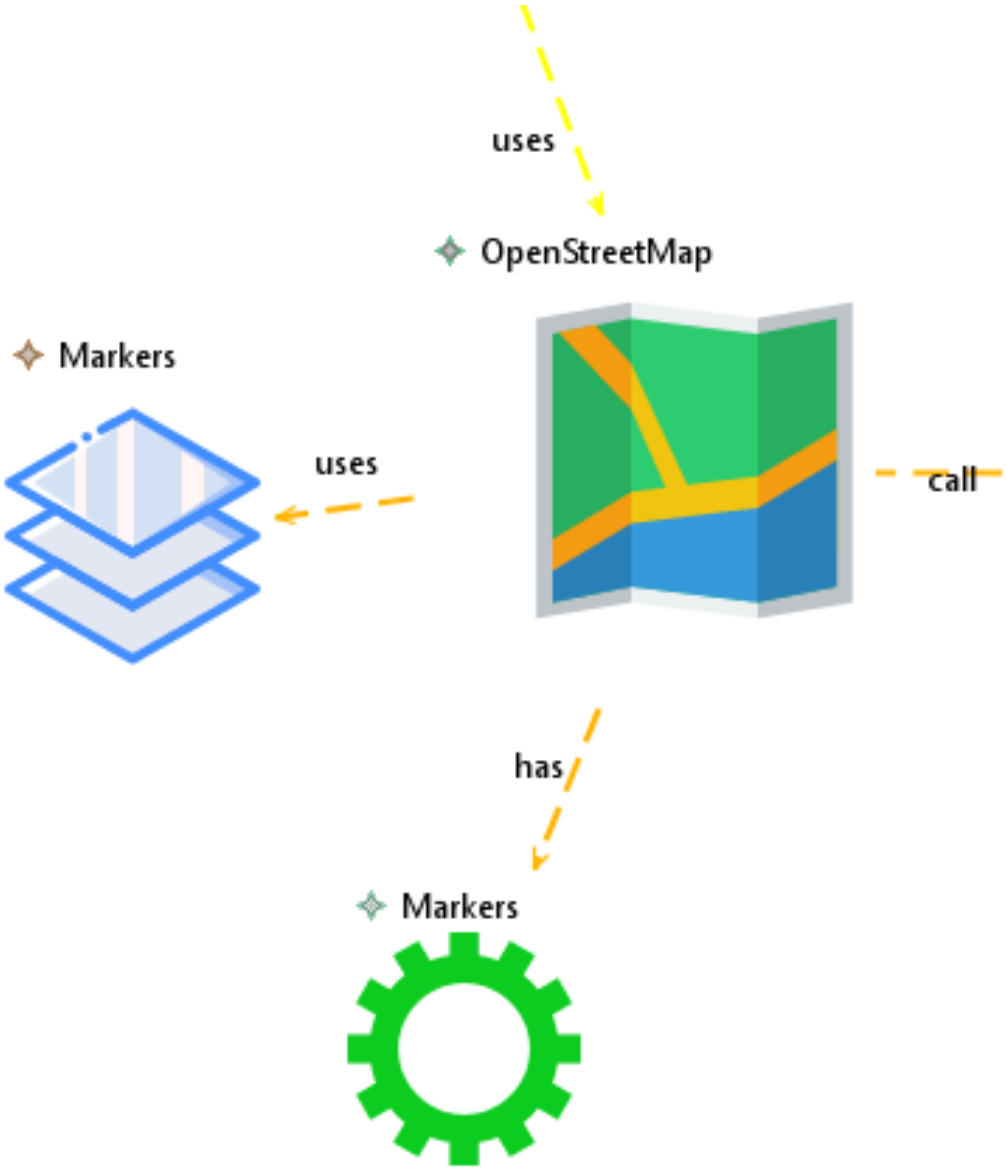


Figura I.9: Apresentação realizada aos avaliadores, slide 17

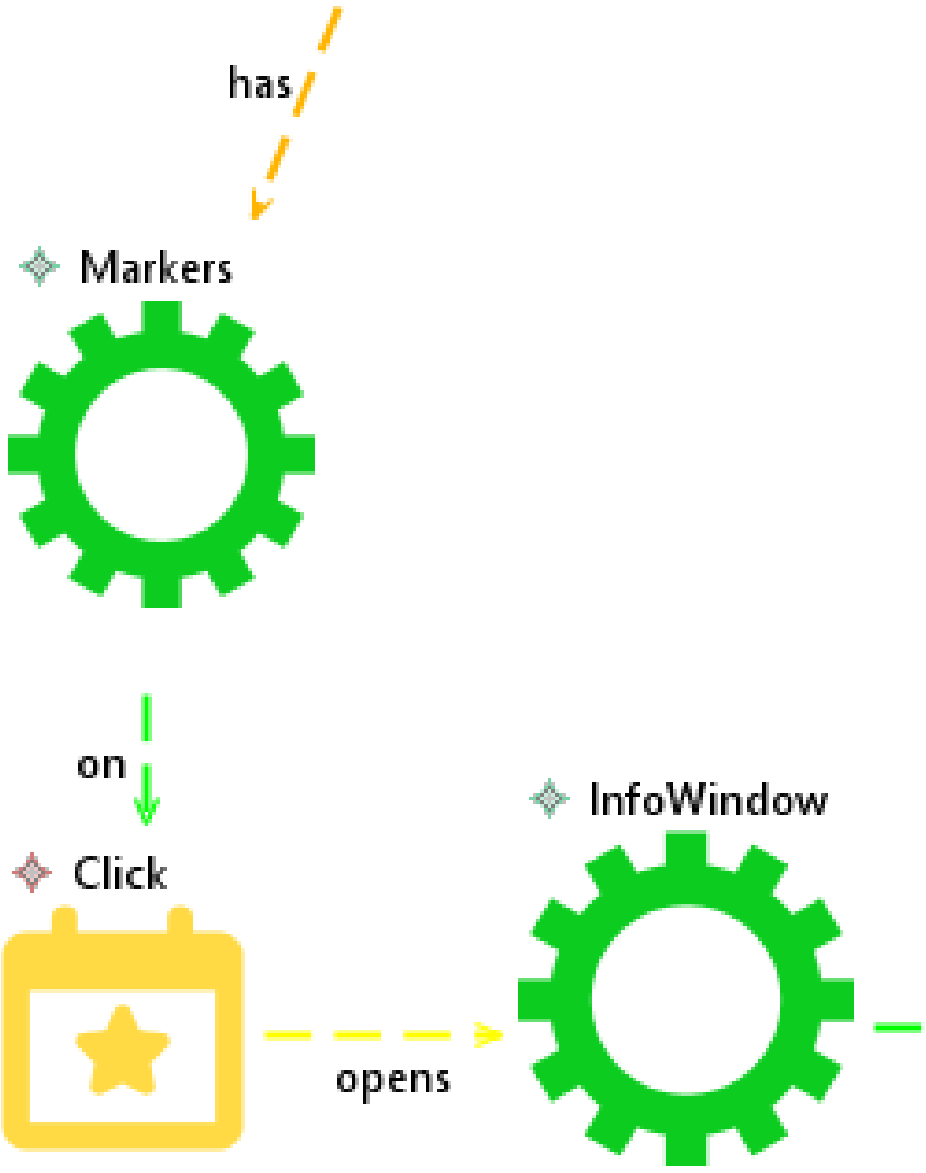


Figura I.10: Apresentação realizada aos avaliadores, slide 18

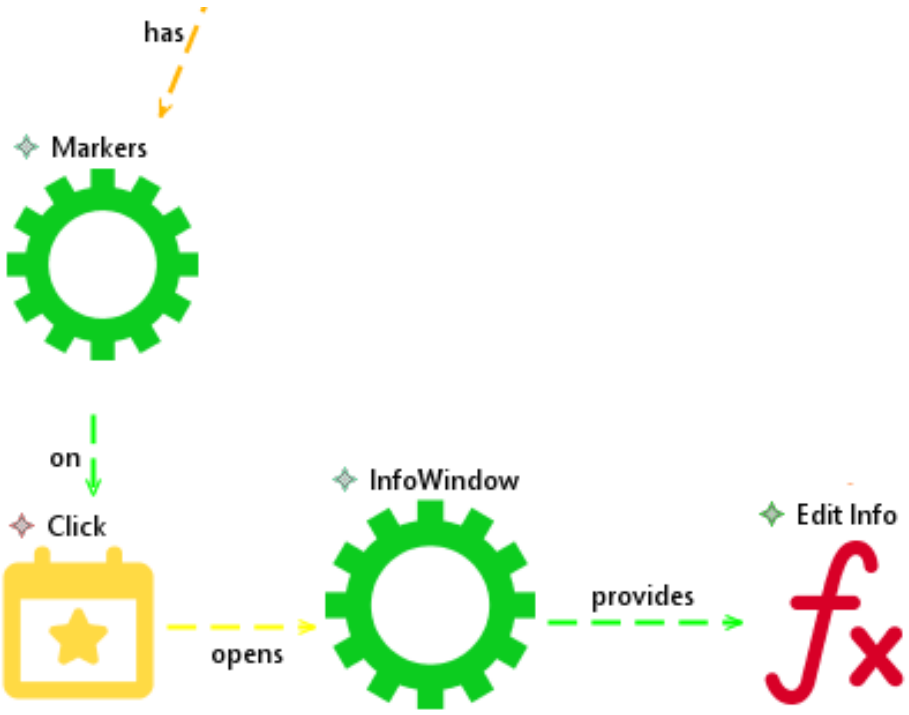


Figura I.11: Apresentação realizada aos avaliadores, slide 19

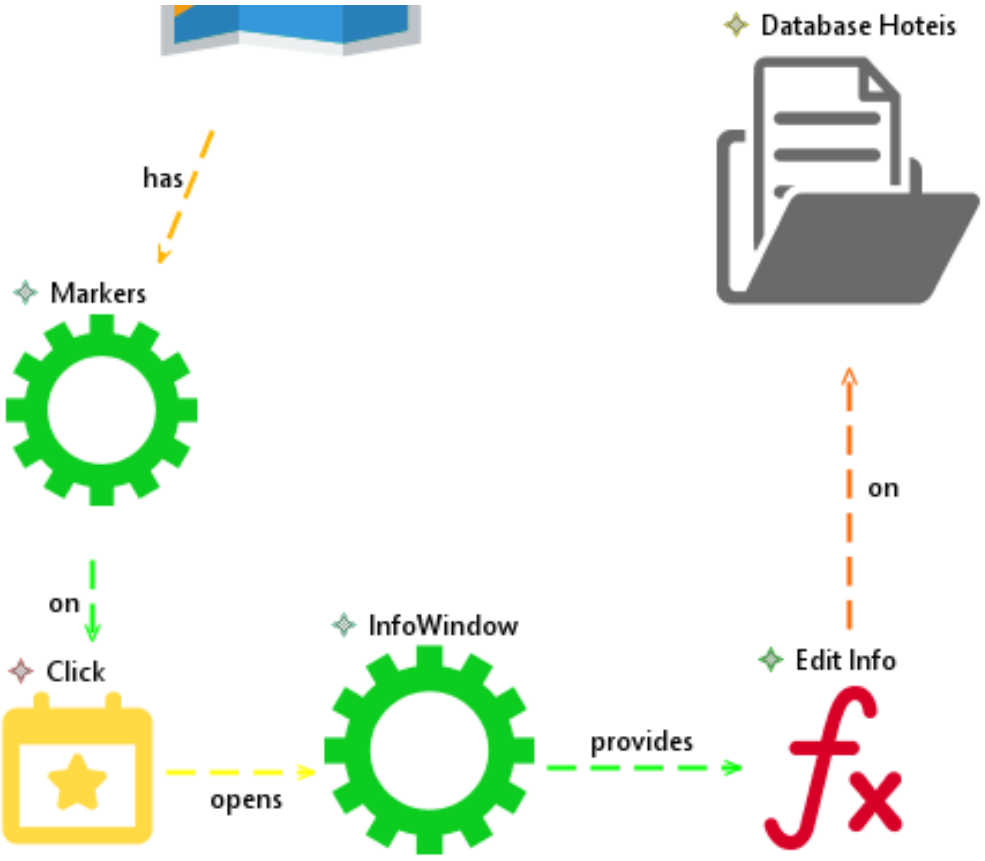


Figura I.12: Apresentação realizada aos avaliadores, slide 20

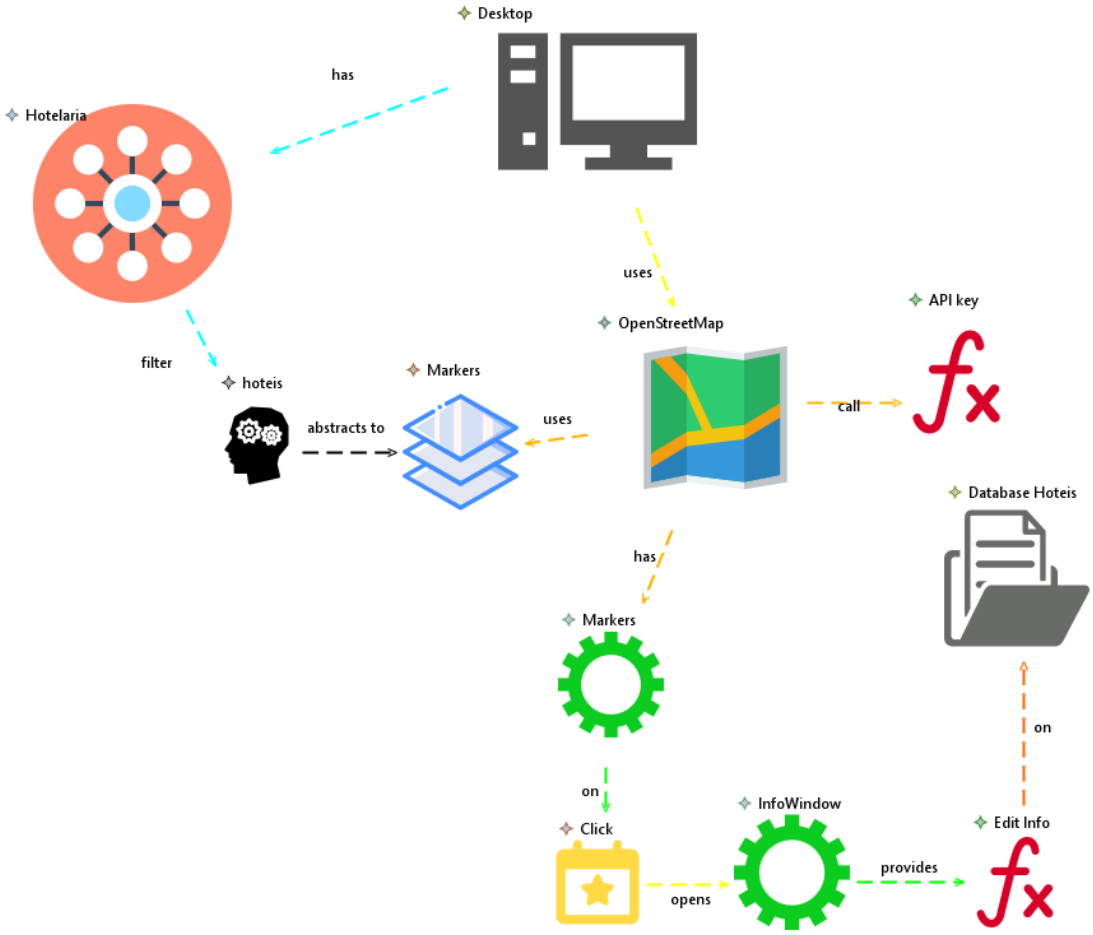


Figura I.13: Apresentação realizada aos avaliadores, slide 21

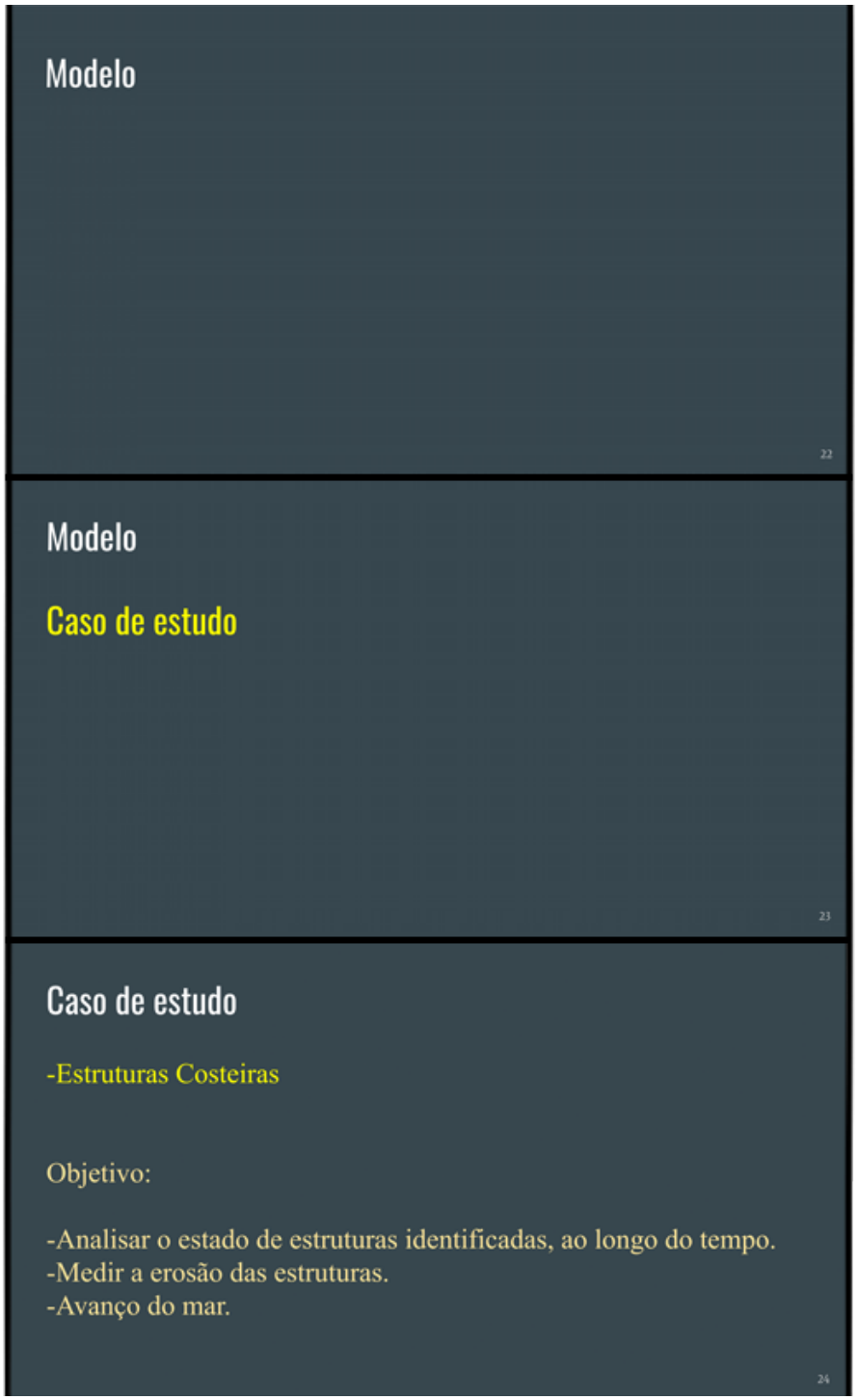


Figura I.14: Apresentação realizada aos avaliadores, slides 22, 23 e 24

Figura I.15: Apresentação realizada aos avaliadores, slides 25, 26 e 27

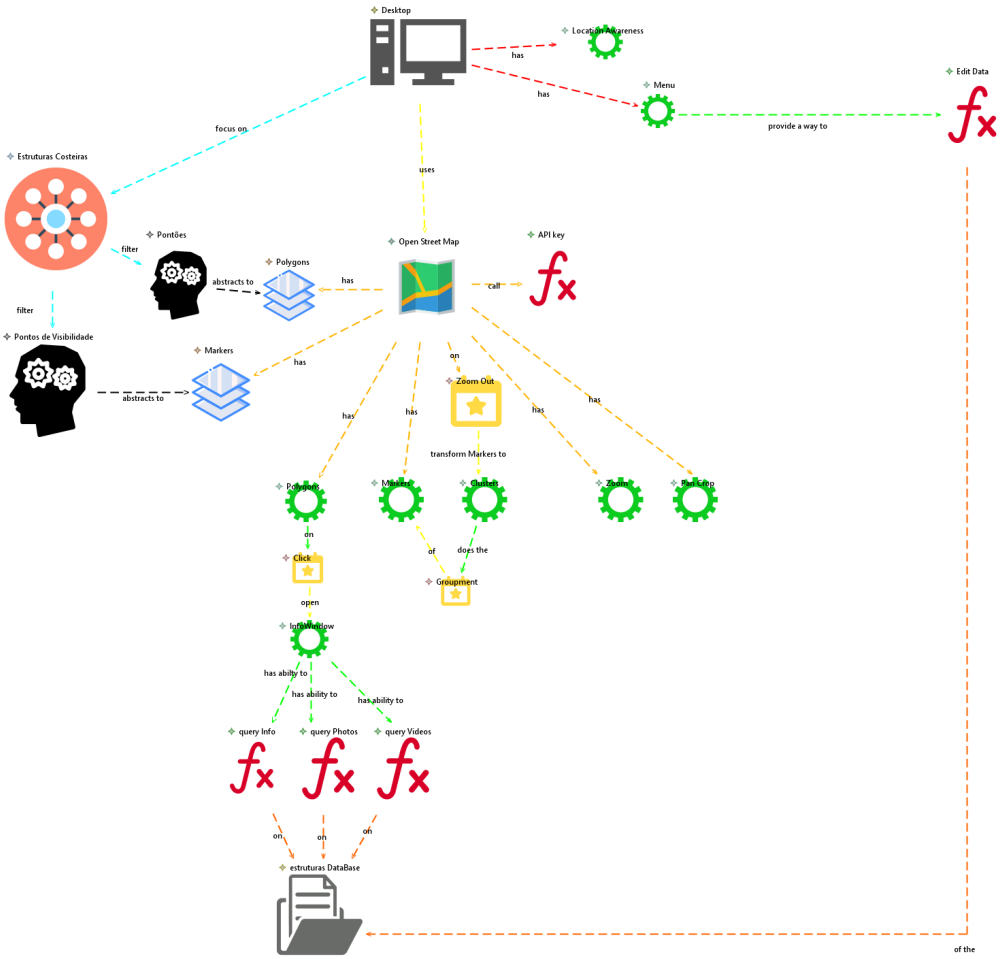


Figura I.16: Apresentação realizada aos avaliadores, slide 28

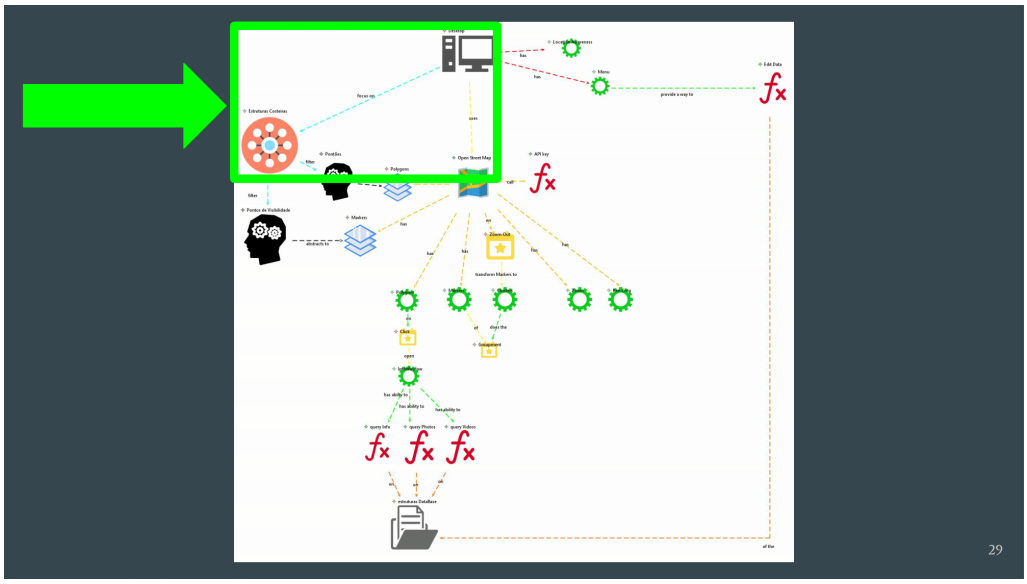


Figura I.17: Apresentação realizada aos avaliadores, slide 29

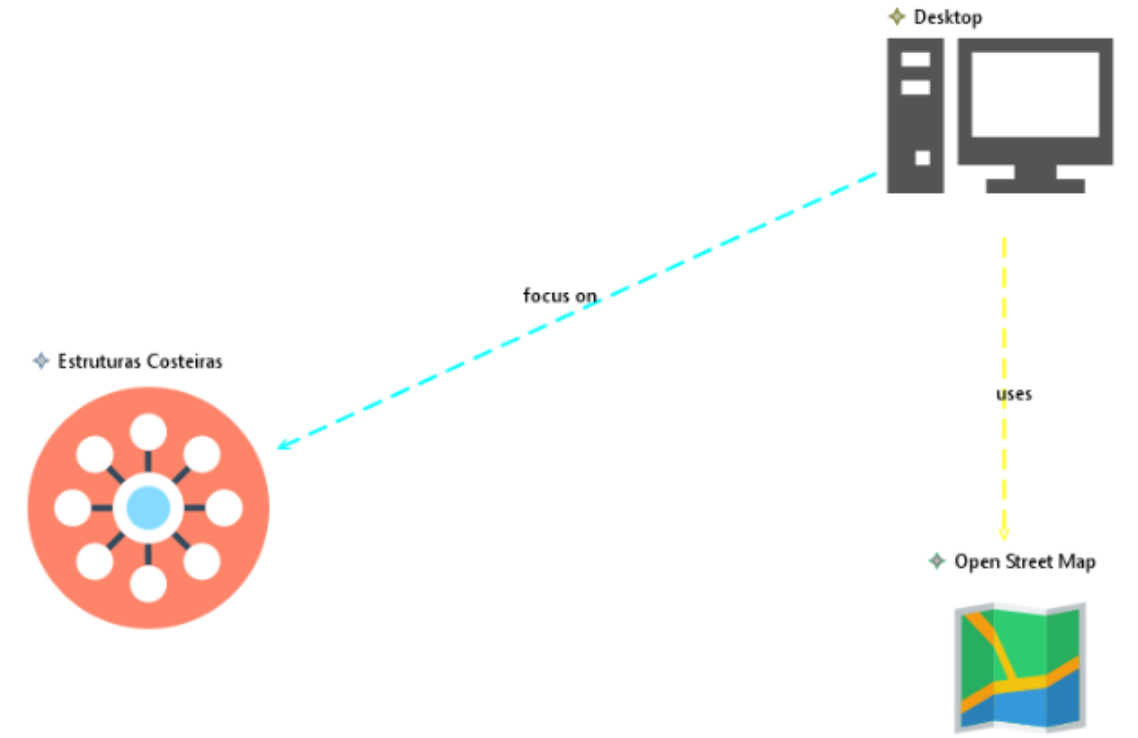


Figura I.18: Apresentação realizada aos avaliadores, slide 30

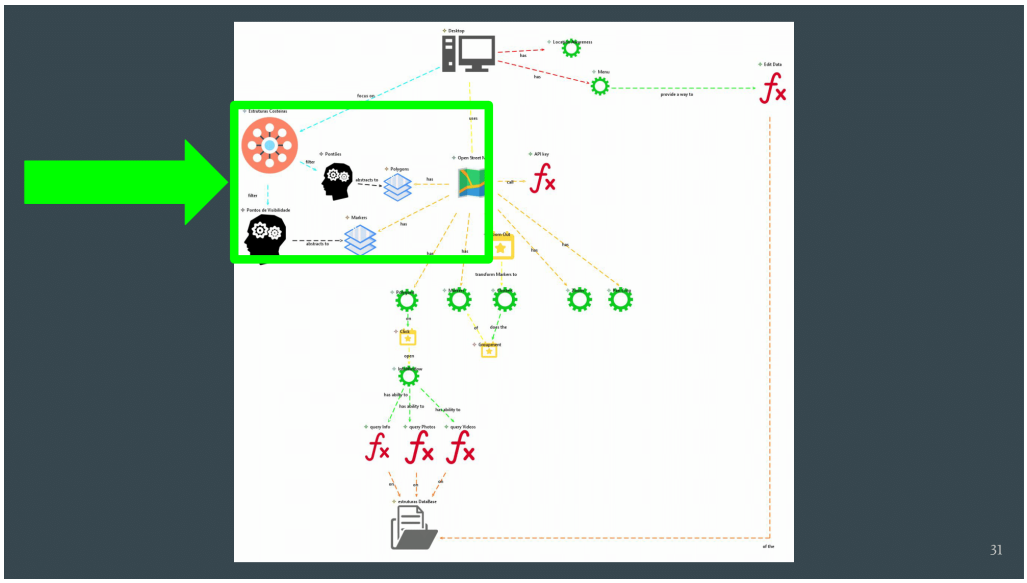


Figura I.19: Apresentação realizada aos avaliadores, slide 31

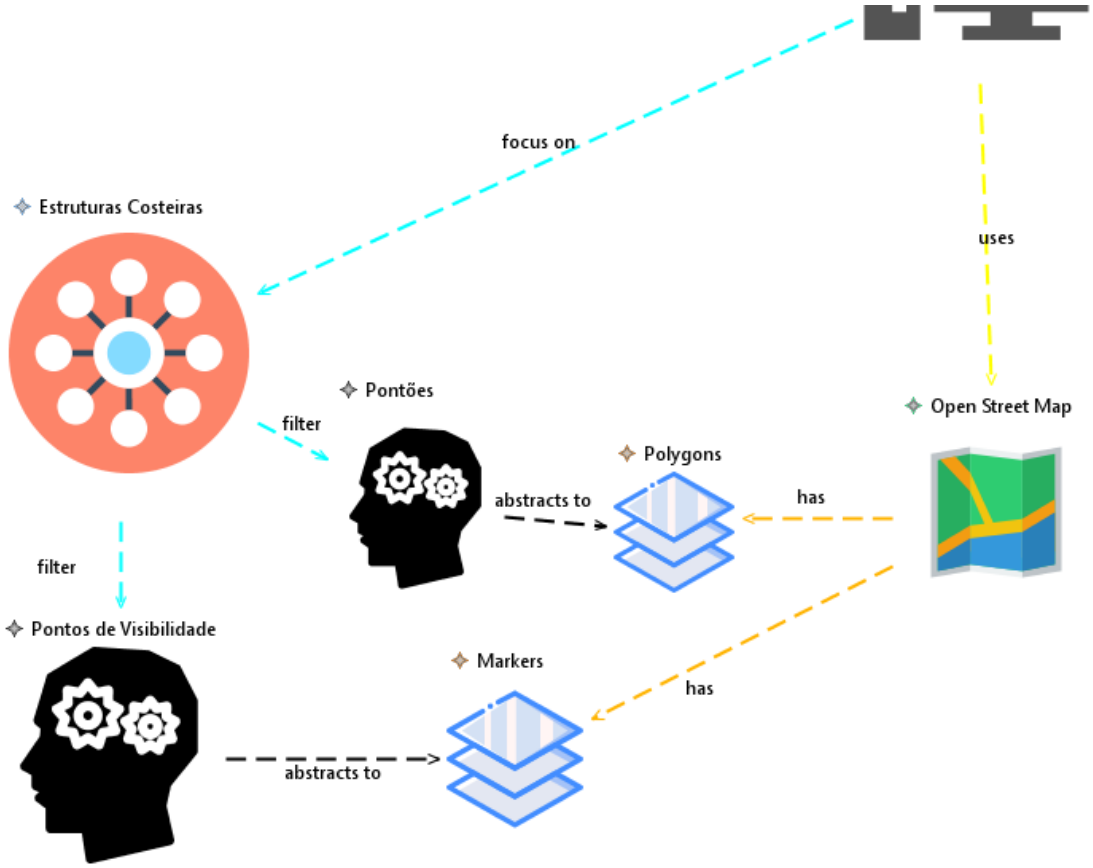


Figura I.20: Apresentação realizada aos avaliadores, slide 32

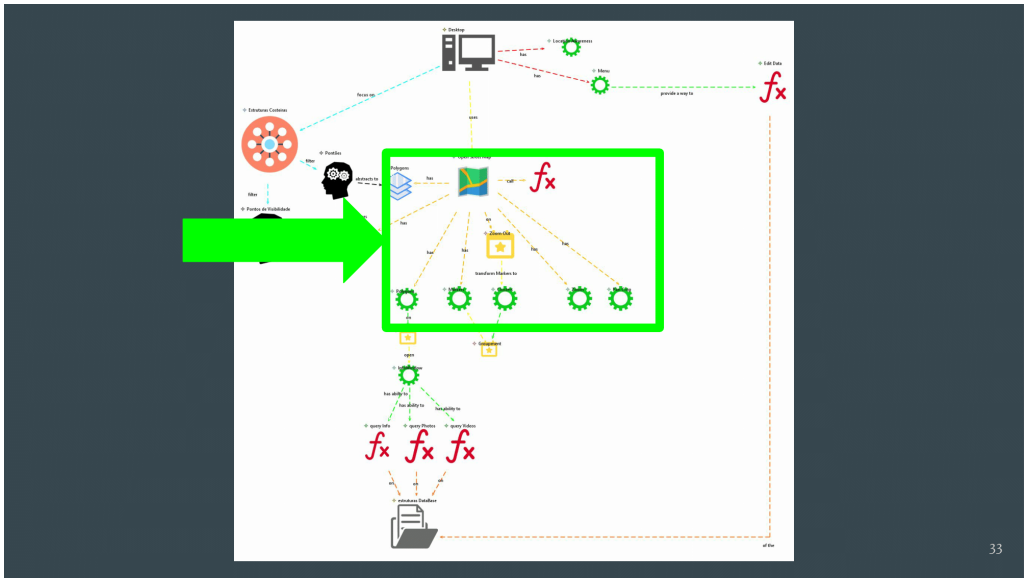


Figura I.21: Apresentação realizada aos avaliadores, slide 33

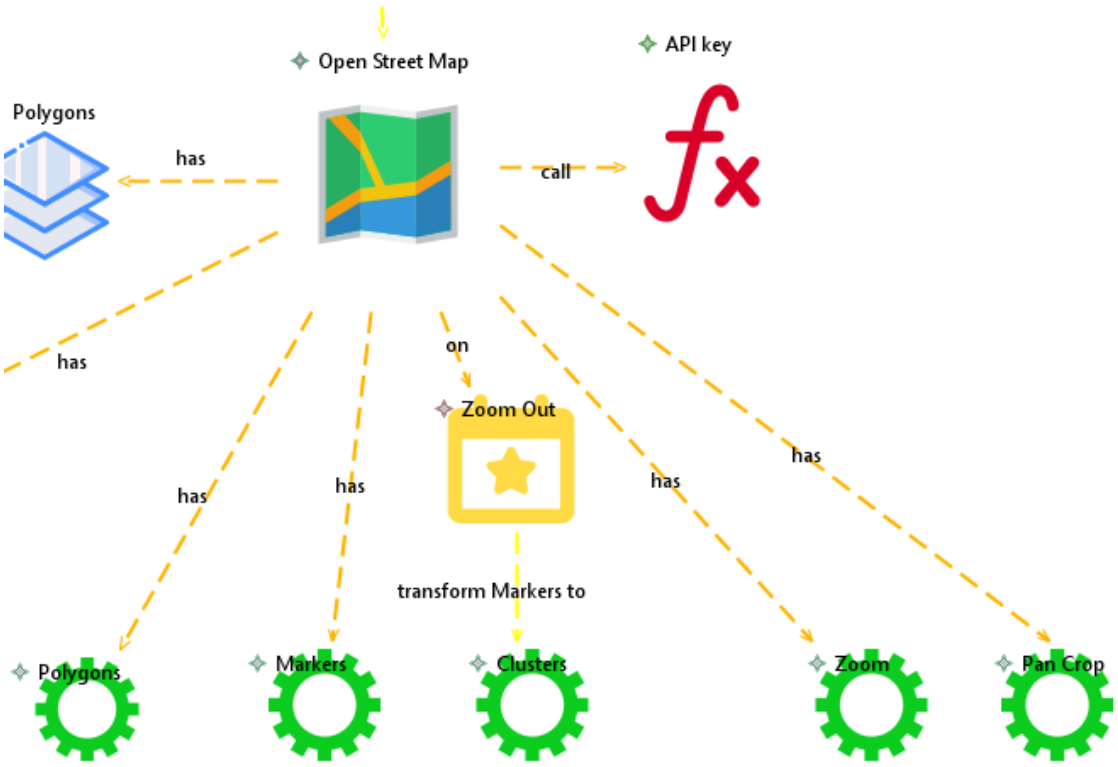


Figura I.22: Apresentação realizada aos avaliadores, slide 34

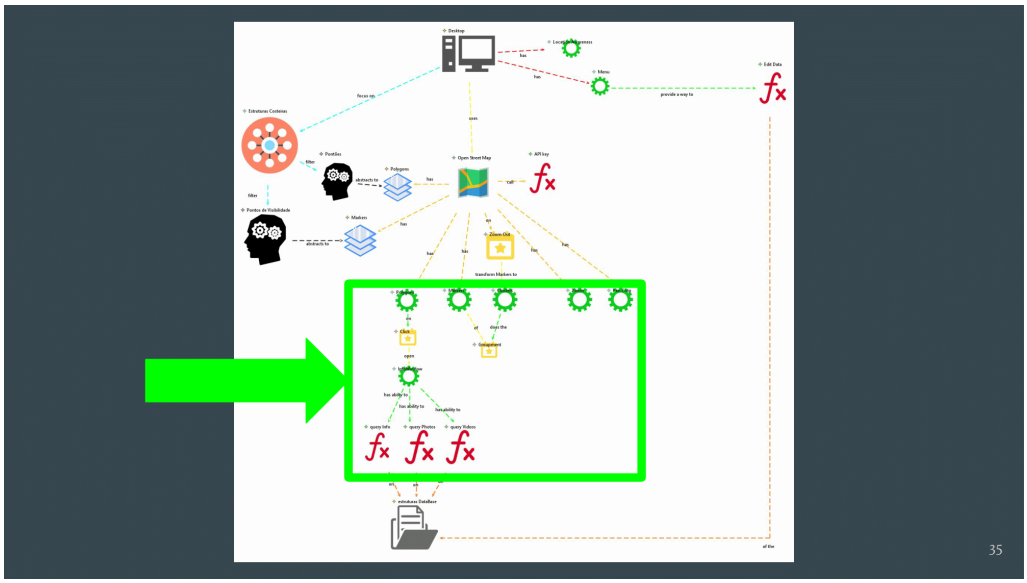


Figura I.23: Apresentação realizada aos avaliadores, slide 35

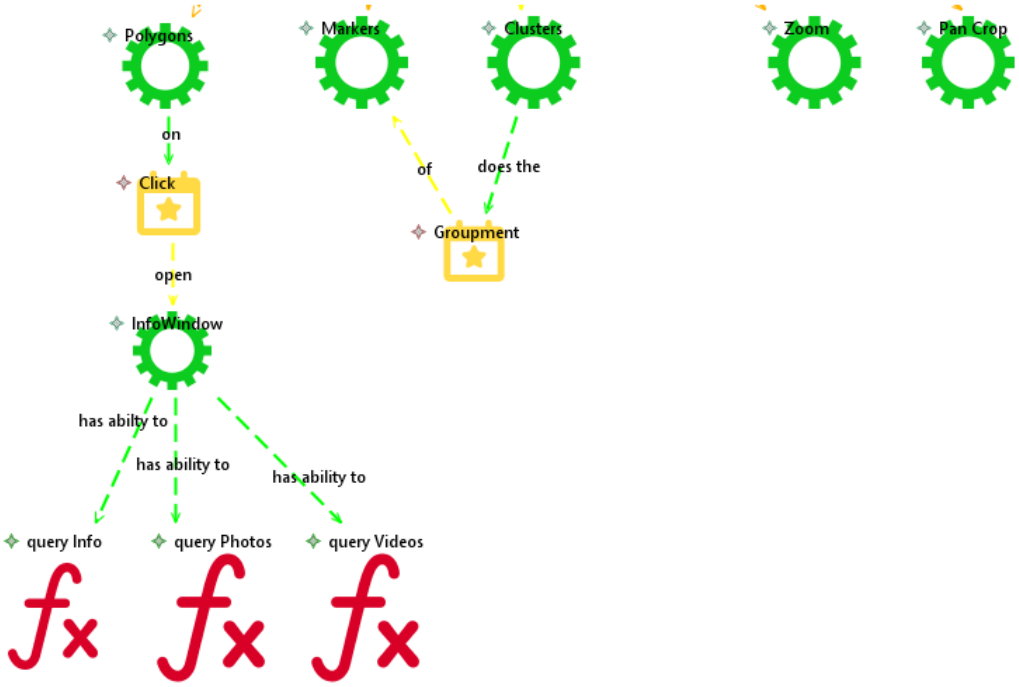


Figura I.24: Apresentação realizada aos avaliadores, slide 36

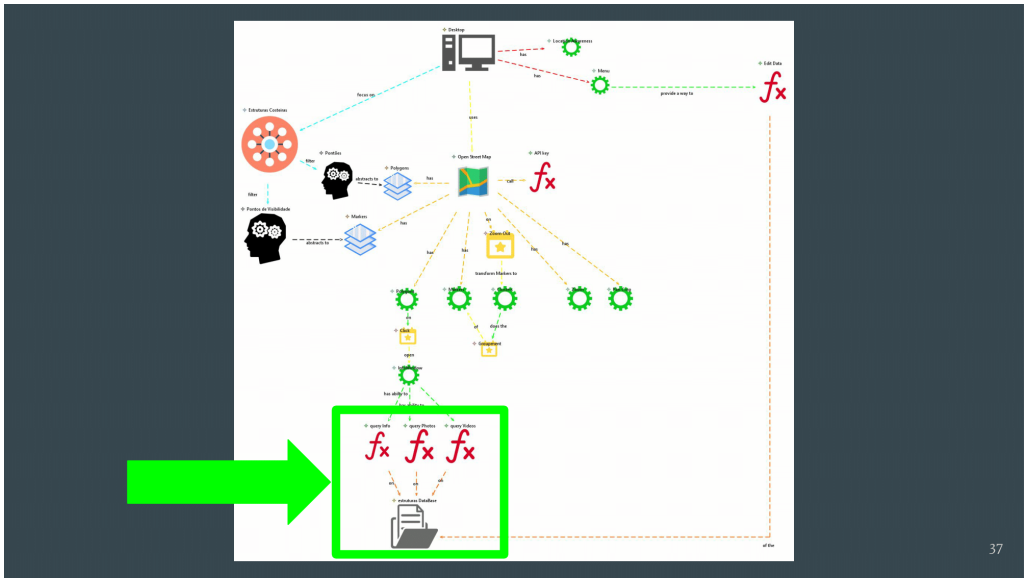


Figura I.25: Apresentação realizada aos avaliadores, slide 37

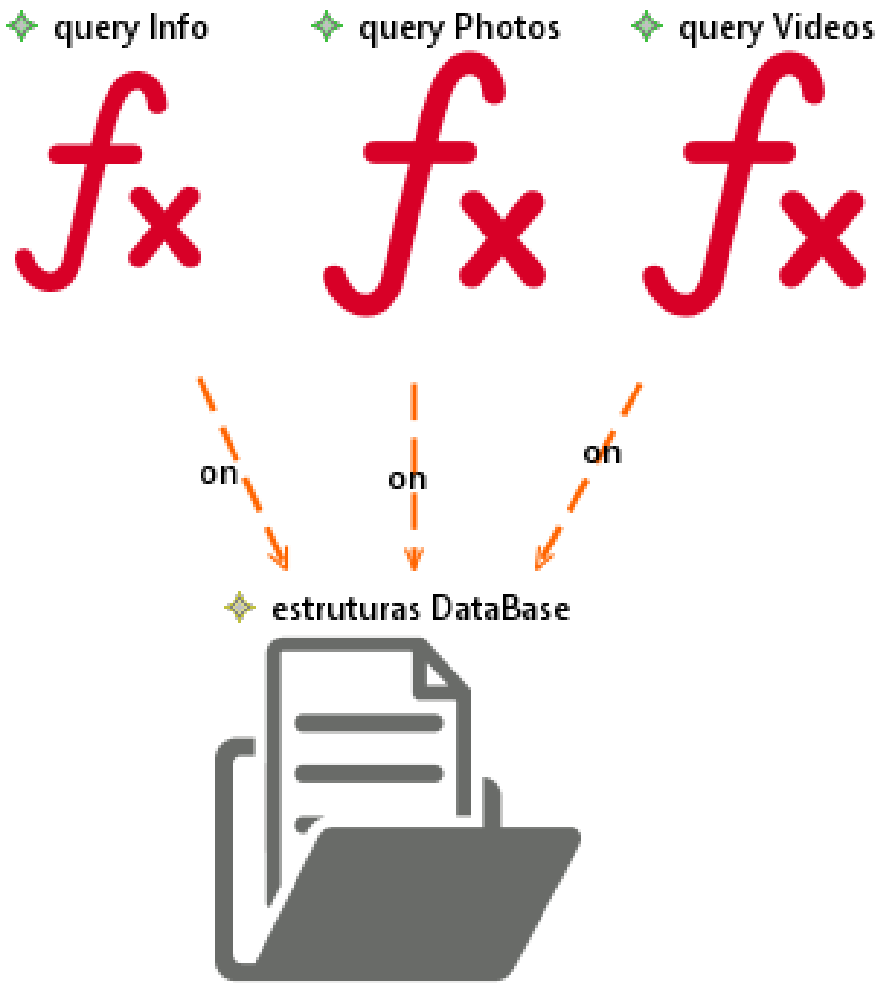


Figura I.26: Apresentação realizada aos avaliadores, slide 38

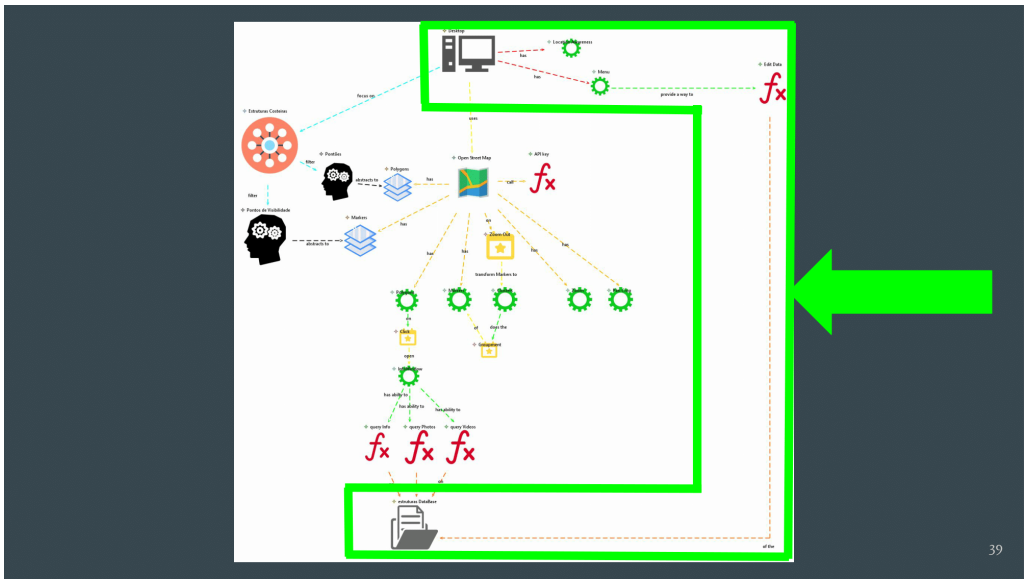


Figura I.27: Apresentação realizada aos avaliadores, slide 39

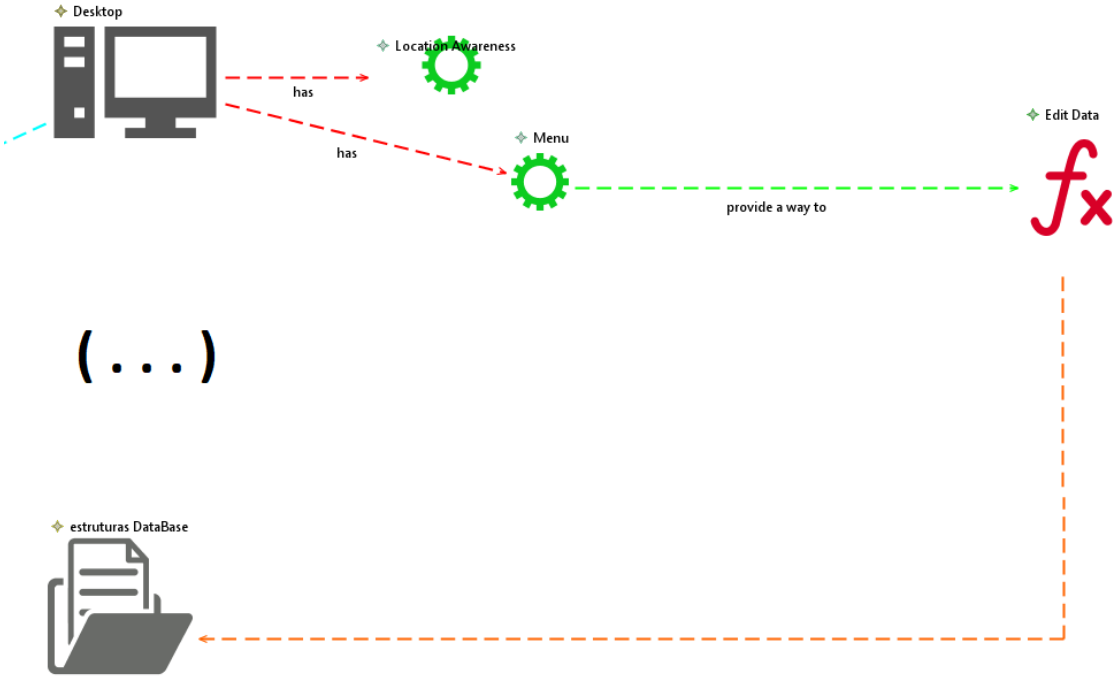


Figura I.28: Apresentação realizada aos avaliadores, slide 40

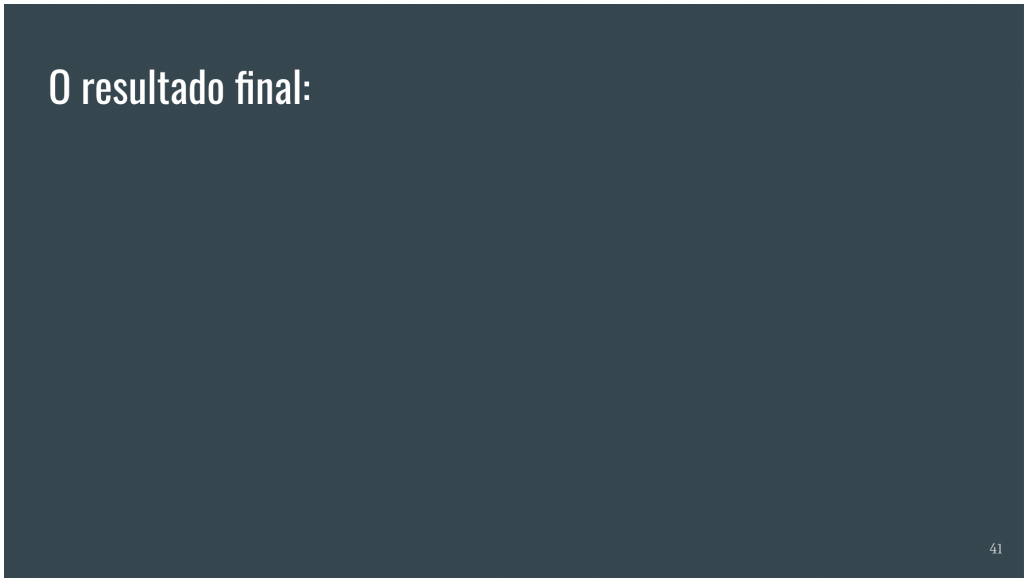


Figura I.29: Apresentação realizada aos avaliadores, slide 41

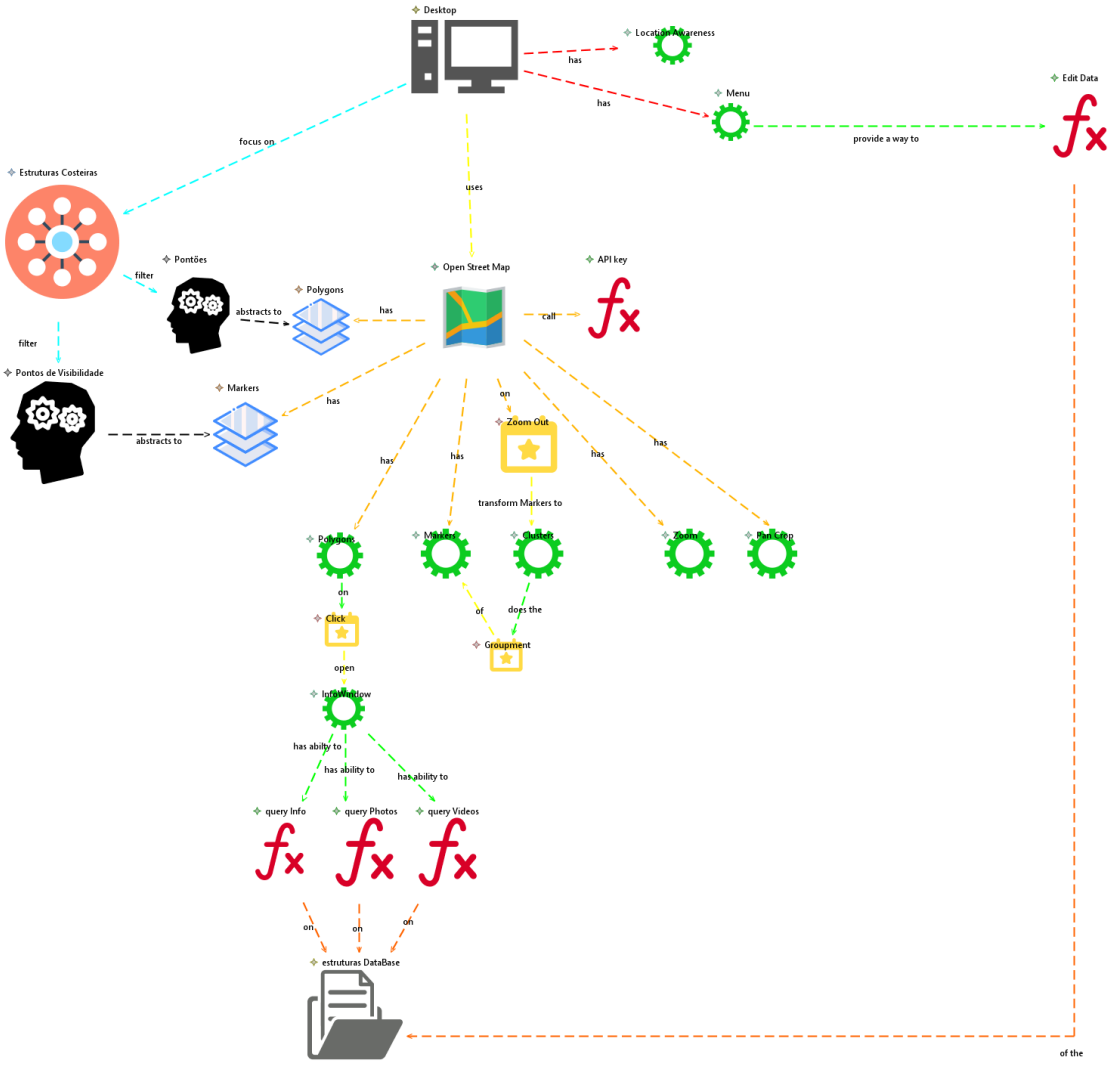


Figura I.30: Apresentação realizada aos avaliadores, slide 42

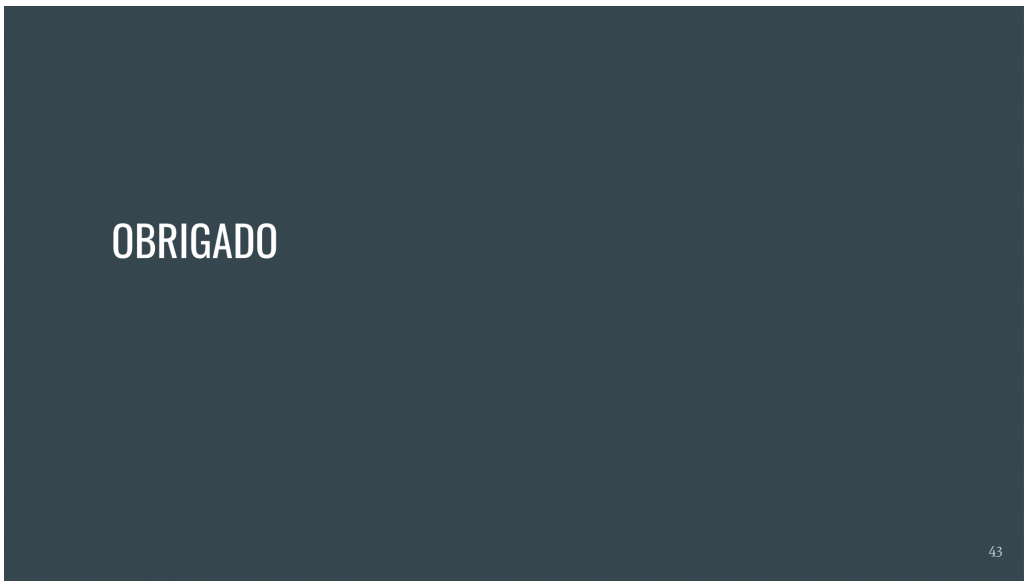
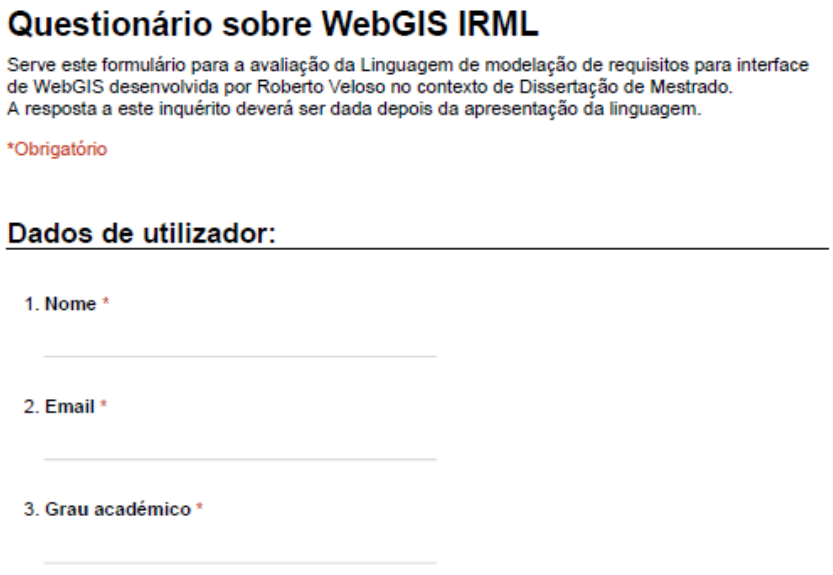


Figura I.31: Apresentação realizada aos avaliadores, slide 43

ANEXO II

Questionário sobre a WebGIS IRML



Questionário sobre WebGIS IRML

Serve este formulário para a avaliação da Linguagem de modelação de requisitos para interface de WebGIS desenvolvida por Roberto Veloso no contexto de Dissertação de Mestrado.
A resposta a este inquérito deverá ser dada depois da apresentação da linguagem.

*Obrigatório

Dados de utilizador:

1. Nome *

2. Email *

3. Grau académico *

Figura II.1: Parte inicial do questionário, identificação do avaliador

Avaliação inicial

Esta secção serve para ter um diagnóstico acerca do contacto que o avaliador tem ou não com o contexto do projeto.

4. **Qual a sua experiência no desenvolvimento de aplicações WebGIS?**
   Marcar apenas uma oval.

	1	2	3	4	5	
Nenhuma	◯	◯	◯	◯	◯	Muita

5. **Há quanto tempo desenvolve aplicações WebGIS?**

_____

6. **Qual a sua experiência com a utilização de aplicações WebGIS?**
   Marcar apenas uma oval.

	1	2	3	4	5	
Nenhuma	◯	◯	◯	◯	◯	Muita

7. **Qual a sua experiência no desenvolvimento de Linguagens de Domínio Específico?**
   Marcar apenas uma oval.

	1	2	3	4	5	
Nenhuma	◯	◯	◯	◯	◯	Muita

8. **Qual a sua experiência com a utilização de Linguagens de Domínio Específico?**
   Marcar apenas uma oval.

	1	2	3	4	5	
Nenhuma	◯	◯	◯	◯	◯	Muita

Figura II.2: Segunda parte do questionário, contacto do avaliador com os conceitos fundamentais da avaliação

Avaliação do modelo

Avaliação acerca dos conceitos utilizados no modelo. Esta avaliação só deverá ser feita depois da apresentação.

9. **Sem contar com as funcionalidades, o Terminal tinha outras duas relações. Uma era com o Domínio e a outra era?**
   Marcar apenas uma oval.
   - ◯ Domain Concepts
   - ◯ Map
   - ◯ Resource

10. **Qual o domínio do modelo observado?**

11. **Qual é o elemento do modelo utilizado para a efectuar a ligação entre um conceito do domínio e o mapa?**

12. **Liste cinco Builtins encontrados no modelo**

13. **Qual o elemento do modelo utilizado para representar a chave de licença da API sobre mapa escolhido?**

14. **Qual o elemento do modelo que permite abrir uma janela de informação (InfoWindow) ?**
    Marcar apenas uma oval.
    - ◯ Função
    - ◯ Evento
    - ◯ Recurso

15. **A partir da InfoWindow é possível visualizar informação, fotos e videos. Estes dados estão armazenados numa base de dados, que no modelo é representada por um elemento, qual?**
    Marcar apenas uma oval.
    - ◯ Domain
    - ◯ Map
    - ◯ Resource

16. **No modelo observado, o Terminal tinha duas funcionalidades, quais?**
    Marcar apenas uma oval.
    - ◯ Location Awareness e Menu
    - ◯ Marcadores e Polígonos
    - ◯ Zoom e Pan Crop

Figura II.3: Terceira parte do questionário, avaliação do modelo apresentado

Avaliação final

17. **Como avalia a notação de elementos da linguagem?**
Marcar apenas uma oval.

	1	2	3	4	5	
Muito Insuficiente	◯	◯	◯	◯	◯	Excelente

18. **Considera os elementos de interface WebGIS definidos no editor suficientes?**
Marcar apenas uma oval.

	1	2	3	4	5	
Muito Insuficientes	◯	◯	◯	◯	◯	Excelente

19. **No geral como avalia os ícones utilizados para representar cada elemento do modelo no editor?**
Marcar apenas uma oval.

	1	2	3	4	5	
Muito fracos	◯	◯	◯	◯	◯	Excelente

20. **Como avalia a dificuldade de aprendizagem dos elementos da linguagem aplicados no modelo?**
Marcar apenas uma oval.

	1	2	3	4	5	
Muito difícil	◯	◯	◯	◯	◯	Muito fácil

21. **Como avalia a linguagem de forma geral?**
Marcar apenas uma oval.

	1	2	3	4	5	
Muito fraca	◯	◯	◯	◯	◯	Excelente

22. **Considera algum aspecto em falta na linguagem? Justifique**

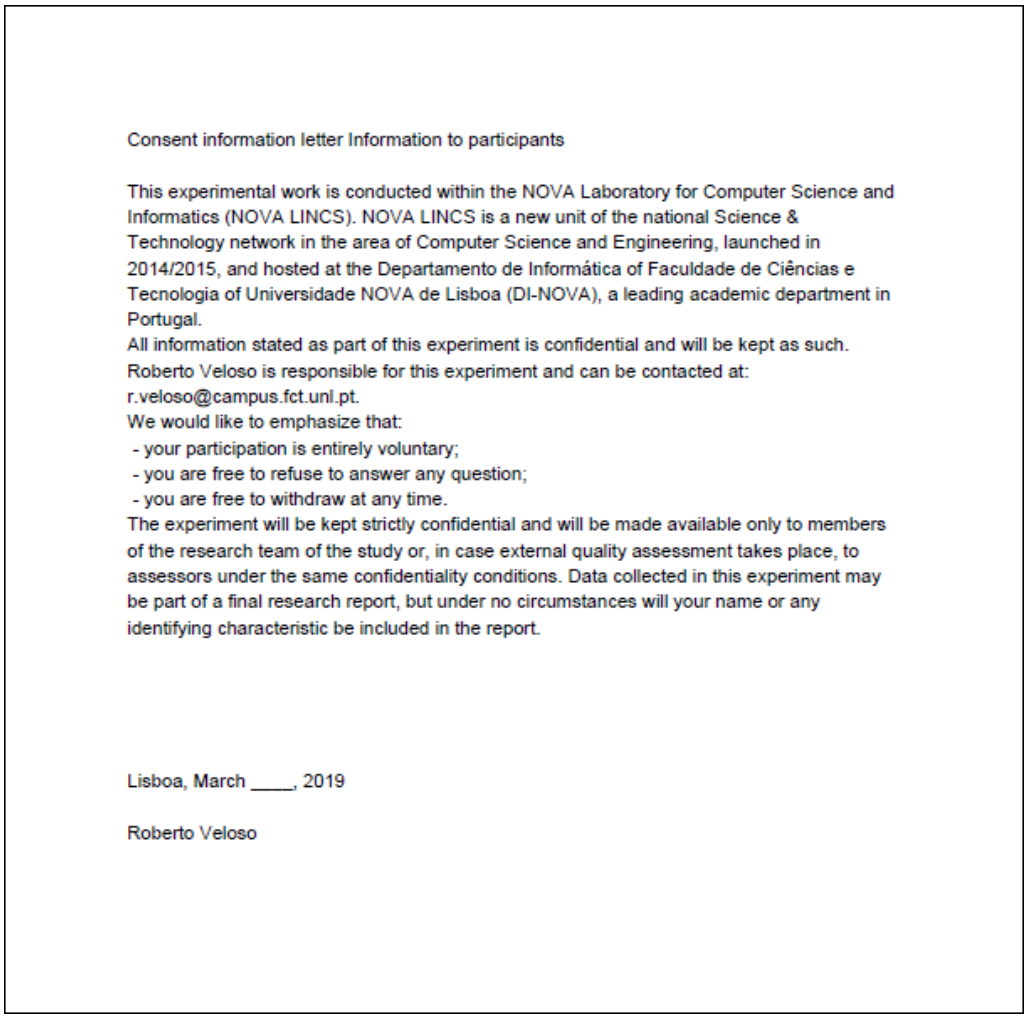
23. **Sugestões?**

Com tecnologia
Google Forms

Figura II.4: Quarta e última parte do questionário, avaliação final

ANEXO III

Carta de Consentimento



Consent information letter Information to participants

This experimental work is conducted within the NOVA Laboratory for Computer Science and Informatics (NOVA LINCS). NOVA LINCS is a new unit of the national Science & Technology network in the area of Computer Science and Engineering, launched in 2014/2015, and hosted at the Departamento de Informática of Faculdade de Ciências e Tecnologia of Universidade NOVA de Lisboa (DI-NOVA), a leading academic department in Portugal.
All information stated as part of this experiment is confidential and will be kept as such.
Roberto Veloso is responsible for this experiment and can be contacted at:
r.veloso@campus.fct.unl.pt.
We would like to emphasize that:
- your participation is entirely voluntary;
- you are free to refuse to answer any question;
- you are free to withdraw at any time.
The experiment will be kept strictly confidential and will be made available only to members of the research team of the study or, in case external quality assessment takes place, to assessors under the same confidentiality conditions. Data collected in this experiment may be part of a final research report, but under no circumstances will your name or any identifying characteristic be included in the report.

Lisboa, March ____, 2019

Roberto Veloso

Figura III.1: Carta de consentimento apresentada aos avaliadores antes de iniciar o questionário