José Carlos Oliveira Pereira
Licenciado em Engenharia Informática

Relatório de Atividade Profissional de 1996 a 2014 "Selfcare Convergente Optimus"

Dissertação para obtenção do Grau de Mestre em Engenharia Informática

Orientador: Prof. Doutora Maria Cecília Farias Lorga Gomes, Prof. Auxiliar, Faculdade de Ciências e Tecnologia da UNL – Departamento Informática

Júri:

Presidente: Prof. Doutor Hervé Miguel Cordeiro Paulino
Arguente(s): Prof. Doutor Joaquim Francisco Ferreira da Silva
Vogal(ais): Prof. Doutora Maria Cecília Farias Lorga Gomes

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro 2014

José Carlos Oliveira Pereira
Licenciado em Engenharia Informática

Relatório de Atividade Profissional de 1996 a 2014 "Selfcare Convergente Optimus"

Dissertação para obtenção do Grau de Mestre em Engenharia Informática

Orientador: Prof. Doutora Maria Cecília Farias Lorga Gomes, Prof. Auxiliar, Faculdade de Ciências e Tecnologia da UNL – Departamento Informática

Júri:

Presidente: Prof. Doutor Hervé Miguel Cordeiro Paulino
Arguente(s): Prof. Doutor Joaquim Francisco Ferreira da Silva
Vogal(ais): Prof. Doutora Maria Cecília Farias Lorga Gomes

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Setembro 2014

Indicação de Direitos de Cópia

Agradecimentos

Quero agradecer aos meus pais, esposa e filha pela paciência e compreensão que têm demonstrado ao longo destes anos pelos períodos de longas horas que tenho dedicado à minha atividade profissional. Sempre senti o apoio e confiança e carinho deles.

Tenho também agradecer aos inúmeros colegas com quem tenho tido o privilégio de colaborar, que me têm ajudado a melhorar na busca continua da excelência.

Por fim, agradeço a orientação e apoio da Professora Doutora Maria Cecília Gomes, que me ajudou na elaboração desta tese.

Resumo

Em Maio de 1996, o autor iniciou a sua vida profissional num trabalho em tempo parcial enquanto terminava o projeto de final de curso de Engenharia Informática. Esta experiência foi fundamental para ter aceite integrar a equipa da EUnet Portugal em Dezembro 1996, permitindo lhe perceber que o ramo seguido pela empresa era revolucionário, e que seria o futuro do mundo empresarial da informática nos serviços relacionados com a Internet.

A EUnet Portugal foi comprada pela KPNQwest Internacional em Abril 1999, que foi comprada pela Novis em Abril 2004 que por sua vez foi integrada na Optimus em Setembro 2007. Em 2013, a Optimus fundiu-se com a ZON para formar a empresa NOS.

Com pequenas alterações, a equipa inicial da EUnet Portugal foi-se sempre mantendo, adquirindo mais responsabilidades e notoriedade à medida que as empresas maiores absorviam as mais pequenas.

Ao longo dos anos o autor interligou várias áreas de valência, passando por analista de sistemas, administrador de sistemas, técnico de suporte, instalador, gestor de projeto e arquiteto especialista em desenho de soluções de alta disponibilidade e fiabilidade. O percurso foi efetuado sempre usando soluções código aberto. Muito frequentemente as soluções desenhadas foram implementadas por equipas geridas pelo próprio.

Abstract

In May 1996, the author started his professional life with a part-time job while completing his final course project in Computer Engineering. This experience was fundamental for choosing to work at EUnet Portugal in December, 1996. The experience allowed him to understand that the company's main focus was revolutionary, and that the future of the computer business world in would be related with the Internet.

EUnet Portugal was purchased by KPNQwest International in April, 1999, which was bought by Novis in April 2004 which in turn was integrated into Optimus in September 2007. In 2013 Optimus merged with ZON to form a new telecommunications company, NOS, where the author is currently employed.

The initial EUnet Portugal's team was maintained all through the various companies with little or no changes, gaining more responsibilities and notoriety, as the bigger companies absorbed the smaller ones.

Through all the years, the author merged various areas of expertise, performing roles in system analysis, systems administration, help-desk operation, IT handy-man, project manager and specialist in high availability and reliability system design. The career has been always used open source solutions for his systems. Many of the designed systems have also been developed or implemented by himself.

Índice Geral

1 Introdução....1
1.1 Enquadramento....1
1.2 Descrição genérica da atividade académica e empresarial....1
1.3 Motivação....2
1.4 Objetivos....2
1.5 Organização do relatório....2
2 Percurso Profissional....3
2.1.1 Empresa Infodesporto....3
2.1.2 Empresa EUnet Portugal....4
2.1.3 Empresa KPNQwest Portugal....5
2.1.4 Empresa Novis....6
2.1.5 Empresa Optimus....7
2.1.6 Empresa ZON Optimus / NOS....7
3 Projetos de Relevo....9
3.1 Mass Calling Services (MCS)....10
3.1.1 Enquadramento....10
3.1.2 Definição do Sistema....10
3.1.3 Intervenção....11
3.1.4 Conclusão....12
3.2 Projeto LIBRA....14
3.2.1 Enquadramento....14
3.2.2 Definição do Sistema....14
3.2.3 Intervenção....14
3.2.4 Conclusão....15
3.3 Webselfcare Convergente Optimus....17
3.3.1 Enquadramento....17
3.3.2 Definição de equipa de trabalho....19
3.3.3 Definição do Sistema – Login Único....20
3.3.3.1 Conceitos Básicos - Navegação na Internet....20
3.3.3.2 Conceitos Básico - Autenticação e sessão....21
3.3.3.3 O Sistema de Autenticação....23
3.3.3.4 Conceitos de Portfólio....27
3.3.3.5 Intervenção....28
3.3.4 Definição do Sistema – Webselfcare....31
3.3.4.1 Considerações técnicas....31
3.3.4.2 Migração....32
3.3.4.3 Registo....34
3.3.4.4 Funcionalidades de selfcare....35
3.3.4.5 Backoffice....37
3.3.4.6 Intervenção....38
Gestão de equipa....38
Desafios técnicos....39
3.3.5 Conclusão....43
4 Conclusão....45
5 Bibliografia....47

Índice de Tabelas

Tabela 1: Matriz de decisão de autenticação de login.....33

Índice de Ilustrações

Figura 1: Arquitetura de sistema Mass Calling System (MCS).....10
Figura 2: Fluxo de controlo de Concursos.....12
Figura 3: Sistema de Troca de Informação LIBRA.....15
Figura 4: Fluxo de autenticação.....23
Figura 5: Sequencia de Autenticação bem sucedida.....24
Figura 6: Acesso autenticado através de domínio diferente.....26
Figura 7: Página de login.....32
Figura 8: Página de registo.....34
Figura 9: O selfcare após validação de login.....35
Figura 10: O backoffice do selfcare convergente.....37

Abreviaturas

ISP/PD Internet Services Provider - Project Development
IT Information Technology / Tecnologia de Informação
ITIL Information Technology Infrastructure Library
LIMBO Large Integrated Mass Business Optimus.
LUNO Login Único Na Optimus
PME Pequenas e Médias Empresas
PUUG Portuguese Unix User Group
RFP Request for Proposal
SNMP Simple Network Management Protocol
SOAP Simple Object Access Protocol
UDP User Datagram Protocol
URL Uniform Resource Locator

1 Introdução

1.1 Enquadramento

O presente relatório visa a obtenção do grau de Mestre em Engenharia Informática, de acordo com o estabelecido no ponto 1.b) do Despacho n.º 20/2010 (Obtenção do Grau de Mestre por Licenciados "Pré-Bolonha"), que abrange licenciados com, pelo menos, cinco anos de experiência profissional.
O autor, ao relatar o seu percurso profissional, expondo alguns casos práticos que espelham a sua competência, pretende demonstrar que a carreira tem seguido um percurso sólido na área de engenharia, tendo adquirido valências suficientes para receber o grau de mestre.

1.2 Descrição genérica da atividade académica e empresarial

O autor terminou a Licenciatura em Engenharia Informática na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa em 1996. Seguiu-se uma carreira sempre ligada às tecnologias de Internet, especializando-se em soluções em código aberto (*Open Source* [1]).

Em 1996 integrou a EUnet Portugal, empresa com origem no primeiro ISP Português, PUUG, onde aumentou significativamente os conhecimentos técnicos, passando por cargos de programador, administrador de sistemas, elemento de suporte telefónico e gestor de projetos.

De seguida, a EUnet foi comprada pela KPNQwest Internacional, onde o autor aperfeiçoou as capacidades de gestor de projeto, com a interação com equipas multi-culturais de vários países europeus.

Após a falência da KPNQwest Internacional, a KPNQwest Portugal foi comprada pela Novis, onde o autor voltou um pouco às origens, voltando a programar, desenhar soluções web e fazendo administração de sistemas. Voltou pouco depois novamente a ocupar a posição de gestão de projetos, posição que consolidou quando a Novis foi integrada na Optimus.

Na atual empresa, que resultou da fusão da Optimus com a NOS, o autor continua a efetuar desenho de sistemas e é responsável por efetuar estimativas de esforço de projetos e de alocação de programadores de uma equipa especializada em tecnologia *Open Source*.

Em termos académicos, terminou uma pós-graduação em gestão e avaliação de projetos na Universidade Católica, obtendo certificações em *Information Technology Infrastructure Library (*ITIL [2]) e certificando-se como *Scrum Master*[i] [3].

---

i O *Scrum* é um processo de desenvolvimento iterativo e incremental para a gestão de projetos e desenvolvimento de software; No capítulo 3 será abordado com mais detalhe.

1.3 Motivação

O autor tem feito questão de ciclicamente se colocar desafios dentro e fora do seu âmbito profissional, tais como a pós-graduação e as certificações. Deste modo garante que sai da sua zona de conforto, ganhando novas competências e ao mesmo tempo garantindo uma atitude de não conformidade. O desafio de obter o grau de mestre tornou-se aliciante devido à possibilidade de juntar ao exercício académico, a possibilidade de fazer uma retrospetiva mais profunda sobre a sua carreira.

1.4 Objetivos

Neste documento será relatado o percurso profissional do autor. Serão apresentados alguns projetos que espelham a experiência profissional desenvolvida desde 1996, ano de conclusão da Licenciatura em Engenharia Informática na Faculdade de Ciências e Tecnologia da Universidade Nova de Lisboa. De entre esses projetos apresenta-se, com mais detalhe, o de desenvolvimento do Selfcare Convergente Optimus, sendo este um caso exemplificativo das valências obtidas ao longo do percurso profissional. Deste modo pretende-se demonstrar que o autor possui conhecimentos científicos e técnicos sólidos na sua área, que consegue compreender, gerir, e modificar sistemas informáticos complexos, arranjando soluções inovadoras e criativas, conseguindo colocar essas soluções em prática em equipas heterogéneas. Pretende-se também mostrar que tem a capacidade de motivar e resolver conflitos entre elementos da equipa, motivando-os para a procura da excelência no trabalho.

1.5 Organização do relatório

De acordo com o enquadramento da introdução, o relatório está organizado da seguinte forma:

O capítulo 2 contém um resumo do percurso profissional, descrevendo de modo breve os principais desafios e avanços em termos de conhecimentos.

O capítulo 3 descreve alguns projetos importantes realizados, destacando-se em particular o desenvolvimento do Selfcare Convergente Optimus.

Finalmente, no capítulo 4, resumem-se de forma conclusiva os resultados atingidos ao longo do percurso profissional.

2 Percurso Profissional

2.1.1 Empresa Infodesporto

O autor realizou o projeto final de curso, sob orientação do Professor José Legatheaux Martins com o título "Whiteboard Distribuído sobre o Protocolo Multicast". Este projeto consistiu em desenvolver uma aplicação colaborativa, distribuída, usando o protocolo de comunicação de rede multicast [4]. Desenvolvido em Java [5], o projeto consistia num programa de desenho gráfico, que permitia o desenho de formas geométricas e linhas em formato livre (rabiscos). A aplicação permitia que várias instâncias fossem lançadas através de diversos navegadores de Internet - *browsers*. Qualquer desenho efetuado num dos navegadores era transmitido para as restantes instâncias usando o protocolo multicast.

Nesta altura, surgiu um convite para integrar, em tempo parcial, uma pequena empresa de software especialista em software de recolha de estatísticas em tempo real de jogos de futebol. A empresa estava a alcançar bastante sucesso, e estava a adaptar o software para ser genérico para qualquer desporto, estando o basquetebol no topo das prioridades, devido ao forte interesse do mercado americano. O esforço principal de desenvolvimento estava direcionado na migração de software escrito em C++ para o sistema operativo Macintosh, para *Delphi*[i] a correr em Windows 95, tendo o autor participado quer na correção de problemas na versão em C++, quer na implementação da nova versão em *Delphi*. Durante este período, a pedido da administração, o autor iniciou uma nova linha de desenvolvimento em *Visual Basic*.

Esta primeira experiência foi extremamente dura, tendo sido necessário longas horas para conciliar o trabalho profissional e académico, no entanto, foi uma experiência enriquecedora e fundamental para a primeira grande decisão de vida profissional: o ramo de informática que o autor deveria seguir.

Este emprego serviu para perceber a situação das empresas pequenas nacionais:

- Não havendo grande margem para contratar pessoal formado em informática por escassez de pessoal qualificado e consequentemente custos mais elevados, não existia um rumo bem definido em termos de objetivos, sendo o software desenvolvido sem grande critério nem exigência.
- Um engenheiro informático de uma destas empresas era suposto ter conhecimento de tudo e saber resolver qualquer tipo de problema tecnológico que surgisse.

Durante os quatro meses em que o autor trabalhou nesta empresa, apareceram desafios em diversas áreas completamente distintas, nomeadamente

- Programação em *C++, Visual Basic, Delphi e Perl*
- Configuração de todo o tipo de hardware, impressoras, telefones, *routers*...

---

i O Delphi era uma linguagem sintaticamente parecida com Pascal que, em relação ao C++, tinha ferramentas mais integradas com o sistema de janelas do Windows 95, transformando assim os desenvolvimentos num processo mais simples e rápido.

- Administração de sistemas em Windows e Linux
- Análise e gestão de projetos
- Elaboração de testes
- Elaboração de documentação

A experiência nesta empresa durou de Maio até Outubro de 1996.

2.1.2 Empresa EUnet Portugal

O convite para integrar a empresa EUnet Portugal surgiu de um modo inesperado e foi causa de uma reflexão profunda sobre o ramo de informática que o autor deveria seguir.

Os serviços de Internet em Portugal na altura estavam ainda na sua infância, existindo apenas quatro empresas nacionais: EUnet Portugal, IP, Estoterica e Telepac. Nessa altura já dava para perceber que a Internet seria algo importante para a sociedade, que iria transformar a maneira que toda a comunicação era feita.

A EUnet Portugal teve origem numa empresa sem fins lucrativos, o *Portuguese Unix User Group* (*PUUG),* que foi o primeiro ISP público em Portugal [6]. A EUnet continuava a contar com vários elementos fundadores da implementação do serviço de Internet neste país. Dificilmente existiria uma outra empresa com um potencial de aprendizagem e margem para crescimento. Este facto foi preponderante para a escolha de aceitação do emprego, experiência que se iniciou em Dezembro de 1996.

A empresa era pequena e focada no fornecimento de serviços, tais como circuitos dedicados de banda larga, a poucas mas grandes empresas. Por este motivo não existiam automatismos de aprovisionamento em grande escala e praticamente todas as ações de criação de serviços eram efetuadas manualmente. A mudança de rumo da empresa, para um foco comercial, implicava a abordagem às Pequenas e Médias Empresas (PME), que aumentava as exigências em termos de criação e manutenção dos diversos serviços.

O primeiro desafio foi implementar um sistema de aprovisionamento técnico, ou seja, um sistema que permitisse a introdução de informação de clientes, informação essa que despoletava, de modo automático, a criação de todos os componentes técnicos necessários para garantir os serviços contratados. O exemplo mais simples da altura era um serviço de ligação à Internet por modem via linha telefónica tradicional. Este serviço incluía:

- O serviço de ligação à Internet.
- O serviço de armazenamento de correio eletrónico.
- O serviço de alojamento de página pessoal.

Após a criação do serviço, o sistema também deveria permitir fazer alterações às componentes de um modo simples, por exemplo, ao alterar a password de um serviço, esta alteração deveria ser propagada para os sistemas de modo automático.

Foi decidido fugir à necessidade de instalação de software em máquinas individuais, e optou-se por seguir um paradigma cliente / servidor [6], em que o cliente seria um *browser* e o servidor um sítio web dinâmico. Esta abordagem foi feliz em inúmeros aspetos, tais como ser independente da plataforma / sistema operativo, ser de fácil manutenção e evolução, já que a complexidade estava concentrada no servidor central e não distribuído pelos computadores pessoais dos colaboradores. Optou-se por evitar sistemas proprietários, escolhendo sempre que possível, sistemas em código aberto.

O autor foi o único recurso responsável pelo desenho e implementação do cliente e do servidor. A evolução do sistema acompanhou o crescimento da empresa ao longo dos anos, começando de um modo muito pouco ambicioso, por gerir apenas caixas de correio sem domino próprio, seguindo quase de imediato para a inclusão dos domínios, e múltiplos endereços de email, integrando depois serviços de conectividade: *dialup*, circuitos dedicados e ADSL, serviços de alojamento web, entre outros. Também foram integrados diversos sistemas externos, tais como o de faturação, de stocks e de controlo de incidentes de clientes. O sistema evoluiu para constituir uma peça central da gestão diária da empresa, permitindo gerar FAXs de clientes, permitindo o acesso, de forma controlada, a informação confidencial de sistemas sensíveis, tais como o de faturação, às diversas equipas que compunham a empresa.

A evolução gradual permitiu ao autor adquirir conhecimento de um modo estruturado e sólido, em todos os ramos de serviços de Internet, permitindo também ganhar experiência na administração de sistemas da família UNIX [8]. Foi durante este tempo que o autor ganhou experiência com duas das linguagens de programação mais populares de programação para a web: Perl [9,10,11] e PHP [12].

2.1.3 Empresa KPNQwest Portugal

Em 1998 a EUnet foi vendida à empresa Qwest, que formou uma nova empresa em conjunto com a Holandesa KPN: a KPNQwest. O objetivo principal da nova empresa era construir uma rede europeia de fibra ótica que estava prevista espalhar-se por mais de treze mil quilómetros. Em cima dessa nova infraestrutura seriam comercializados serviços de Internet. Uma iniciativa desta envergadura implicou a interação estreita entre os vários países que compunham a empresa, tendo havido vários projetos em conjunto, no entanto o modo de trabalho em Portugal não sofreu grandes alterações: continuou-se a investir em sistemas cliente/servidor em sistema de código aberto.

O autor passou a acumular novas funções, tais como ser responsável por um sistema de auditoria de serviços fantasma, ou seja, serviços existentes nos sistemas técnicos sem ligação ao sistema de faturação.

Outro desafio foi ser o gestor de projeto por Portugal, do problema do ano 2000 [13]. Esta função obrigou à recolha de centenas de documentos de certificação de aparelhos e sistemas em uso na infraestrutura da

empresa portuguesa, bem como a auditoria de todas as peças de software usadas nos serviços. O autor integrou uma equipa de auditoria internacional, sediada na Holanda, responsável por analisar software comum às várias empresas da KPNQwest. Foi uma experiência interessante que obrigou a passar algumas semanas na Holanda, que permitiu a interação com equipas multi-culturais, onde foi possível ganhar experiência em negociar abordagens para problemas comuns.

2.1.4 Empresa Novis

Em 2004 a KPNQwest Internacional faliu[ii], sendo a KPNQwest Portugal comprada pela Novis, uma empresa que operava no ramo da Internet fixa, serviços de ligação através de linhas de telefone terrestres - dialup, ADSL e circuitos dedicados.
A empresa tinha uma vasta equipa técnica, acima dos 20 elementos, em comparação com os 2 da KPNQwest Portugal. Existia um leque de conhecimento muito mais extenso, havendo empregados mundialmente reconhecidos como peritos em diversas áreas, tais como a linguagem Perl e o sistema operativo Linux. Pela primeira vez o autor percebeu o isolamento intelectual em que vivia na EUnet/KPNQwest.

As linhas de desenvolvimento eram as mesmas, centradas à volta de serviços de Internet. Devido ao tamanho da equipa, existia um diálogo muito forte na discussão e planeamento de soluções. Esta estratégia tinha como consequência soluções mais robustas e tolerantes a imprevistos.

Foi um período de crescimento profissional muito forte, com aumento de formação profissional, em áreas específicas, como em Oracle e na linguagem Perl. Em 2006, o autor assistiu ao que considera ter sido a melhor formação da sua carreira: *Perl Best Practices* [14], lecionado pelo australiano *Damian Conway*. Esta formação teve efeitos imediatos e impactou o método de trabalho do autor e de todas as equipas por onde passou, já que introduziu os conceitos de normalização de estilos de programação. Uma das principais características do trabalho produzido pelas equipas do autor é a clareza e consistência do código, que facilita a integração de novos programadores.

O autor passou dois anos a planear e a executar, de modo transparente para os clientes, a migração dos serviços da KPNQwest Portugal para a infraestrutura da Novis. Ao longo deste tempo adquiriu um conhecimento profundo das plataformas da Novis. Ao final de dois anos, apesar do processo de migração não estar completamente terminado[iii], o autor passou a integrar ativamente as equipas de desenvolvimento da Novis.

---

ii A falência foi originada pelo colapso generalizado das empresas *dot com,* empresas ligadas ao mundo da Internet, altamente valorizadas em termos comerciais e financeiros, mas sem valor real associado. O colapso retirou mercado aos operadores de telecomunicações de venda de serviços de conectividade a preços inflacionados, premissa na qual os planos de negócios haviam sido elaborados

iii O último serviço foi migrado em Setembro 2013

2.1.5 Empresa Optimus

Em Setembro, 2007, devido a uma reorganização do grupo Soneacom, a Novis foi integrada na Optimus.

A resultado imediato foi a necessidade de encaixar os serviços de Internet fixa num contexto mais abrangente de serviço, para incluir o universo móvel. Isto obrigou a um esforço inicial para adquirir novos conhecimentos técnicos sobre os serviços móveis, conhecimento que foi obtido através de leitura de textos disponíveis na Internet, e complementados com reuniões com equipas especializadas da Optimus.

Outra novidade foi a de ter de lidar com diversas equipas responsáveis pelos produtos comerciais. Nas empresas anteriores, existia uma única unidade de negócio que tomava as decisões sobre o rumo comercial dos produtos oferecidos. Na Optimus, as unidades de negócio estavam fortemente segmentadas, havendo equipas semelhantes, tais como Voz Móvel Residencial e Voz Móvel Empresarial, muitas vezes a defender ideias concorrentes. Muitos dos elementos destas equipas não tinham origem em áreas técnicas, ao contrário do que tinha acontecido nas empresas anteriores. Uma consequência direta foi a necessidade de mudar o modo de diálogo em reuniões, para conseguir transmitir as dificuldades de um modo claro.

Como analista e arquiteto de soluções Internet, o maior desafio foi conseguir arquiteturas eficientes e de simples manutenção. A necessidade de gerir equipas em projetos de longa duração levou ao uso de várias técnicas de gestão de projeto, tendo-se destacado mais a técnica *Scrum* (adaptada), como será descrito no capítulo 3.

2.1.6 Empresa ZON Optimus / NOS

A fusão das empresas Optimus e ZON ocorreu em 2013, dando origem à empresa temporária ZON Optimus. Passados poucos meses, a 16 Maio 2014, foi finalmente criada a nova operadora de telecomunicações nacional: NOS.

Um dos grandes desafios dos próximos anos será a convergência dos sistemas informáticos e respetivas equipas que existiam duplicados em ambas as empresas. Como na maioria das áreas, existia uma equipa equivalente da ZON com funções semelhantes à do autor. Enquanto a equipa oriunda da Optimus usava apenas software livre, de código fonte aberto, o da ZON especializava-se em soluções da Microsoft, usando principalmente soluções assentas na tecnologia Sharepoint. Foi decidido continuar a apostar nos dois paradigmas, colocando os serviços de conteúdos, tais como o site da NOS, e a Intranet corporativa, sob a responsabilidade da equipa de Sharepoint. Os sites de maior complexidade, tais como os *selfcares,* ficarão com a equipa ex-Optimus. Esta decisão foi tomada após uma análise profunda do historial de ambas as equipas, considerando vários pontos, incluindo custo e tempo de desenvolvimento, número e nível (baixo, médio ou alto) de criticidade de incidentes (*bugs*) após entrega e tempo de resolução dos mesmos.

O autor continua a efetuar tarefas de gestão de projeto e de desenho de soluções de serviços para a Internet, no entanto passou também a ser responsável por efetuar estimativas de esforço de projetos e por

gerir uma equipa de programadores especializados em tecnologias *Open Source*.

3 Projetos de Relevo

A escolha de um único projeto no meio de tantos que passaram pelo autor não foi tarefa simples. Todo o projeto complexo bem executado deixa sempre marcas. Alguns deixam um sentimento de saudade, outros de alívio. O autor decidiu apresentar três projetos, dois de um modo muito breve e resumido, onde teve um papel fundamental para um desfecho de sucesso. Todos os exemplos tratam de problemas complexos envolvendo tecnologias completamente dispares.

A descrição dos projetos está divididos em quatro secções, nomeadamente

Enquadramento

Secção onde é dado um esclarecimento alto nível do projeto, incluindo como surgiu a necessidade de ser implementado.

Definição do sistema

Secção onde é dada uma explicação técnica mais pormenorizada.

Intervenção

Secção onde é explicado o papel do autor, das dificuldades que enfrentou e as soluções que encontrou para resolver as dificuldades.

Conclusão

Secção onde é feito um balanço crítico do projeto.

3.1 Mass Calling Services (MCS)

3.1.1 Enquadramento

Em 2005 os serviços de telefonemas de valor acrescentado estavam a surgir em força. Através da televisão, os expetadores eram convidados a ligar para números de valor acrescentado, para participar em vários tipos de concursos de sorte, tais como os que atribuíam prémios cada duzentas chamadas, ou que pediam para adivinhar um código numérico para abrir um cofre virtual que continha o prémio.

A Novis era a empresa líder de mercado de receção e tratamento de deste tipo de chamadas telefónicas.
Todo o sistema de lógica dos concursos, em conjunto com o *backoffice* de configuração e visualização de detalhes dos mesmos, era feito por um parceiro externo. Este não estava a conseguir reagir em tempo útil a pedidos de alteração, e foi decido implementar um sistema alternativo na Novis.
O desenvolvimento do projeto derrapou e quando a pressão da administração já se começava a sentir, o principal recurso e líder do projeto abandonou o projeto, deixando-o em risco de colapso total. Foi pedido ao autor para assumir o desenvolvimento e recuperação do projeto.

3.1.2 Definição do Sistema

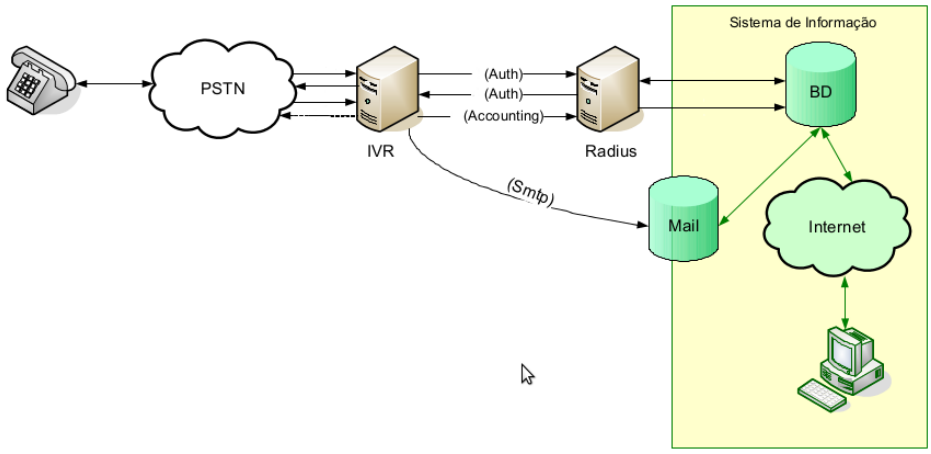

Figura 1: Arquitetura de sistema Mass Calling System (MCS)

A figura 1 descreve a arquitetura geral e os fluxos que decorrem durante a participação num concurso telefónico. O sistema que estava a ser reescrito pela equipa do autor era o bloco “Sistema de Informação” e o de *Radius* que se pode ver na figura.

Detalhando um pouco melhor o sistema, este era composto por:

- Um elemento de Rede telefónica que recebia as chamadas dos clientes que:
    - os encaminhava para um sistema de autenticação e de cadastro: *Radius* [15,16].
    - devolvia uma mensagem ao cliente (prémio/não prémio) consoante a resposta do *Radius.*
- *Radius*.
    - Responsável por implementar a lógica do concurso.
    - Por requisito teria de poder dar resposta a mil pedidos por segundo.
    - Por questões de performance tinha sido escolhido o software de código aberto, *FreeRadius* [16], um servidor escrito em C [17].
- *Backend* de controlo de concursos escrito em Perl.
- *Backoffice* em Java.

3.1.3 Intervenção

Além da complexidade do sistema, o maior desafio para o autor foi lidar com o pivô da área de negócio responsável pelo produto, que não aceitava o atraso nos desenvolvimentos e estava com uma atitude extremamente agressiva. A solução passou por aumentar o diálogo com o pivô, mostrando os avanços diários, e envolvendo-o nas dificuldades do dia-a-dia de desenvolvimento.
Em termos técnicos, o autor optou por primeiro documentar detalhadamente o comportamento do sistema e de seguida corrigir as falhas do código que lhe foi entregue.
A intervenção foi necessária quer no *Radius*, quer na "*BD*". Esta última, após análise cuidada levada a cabo pelo autor, levou ao esquema apresentado na figura 2, que é apresentada apenas para passar a ideia da complexidade de interações entre os vários componentes. O sistema BD foi escrita em Perl, e o de *Radius* em C.

O processo de recuperação demorou três meses.

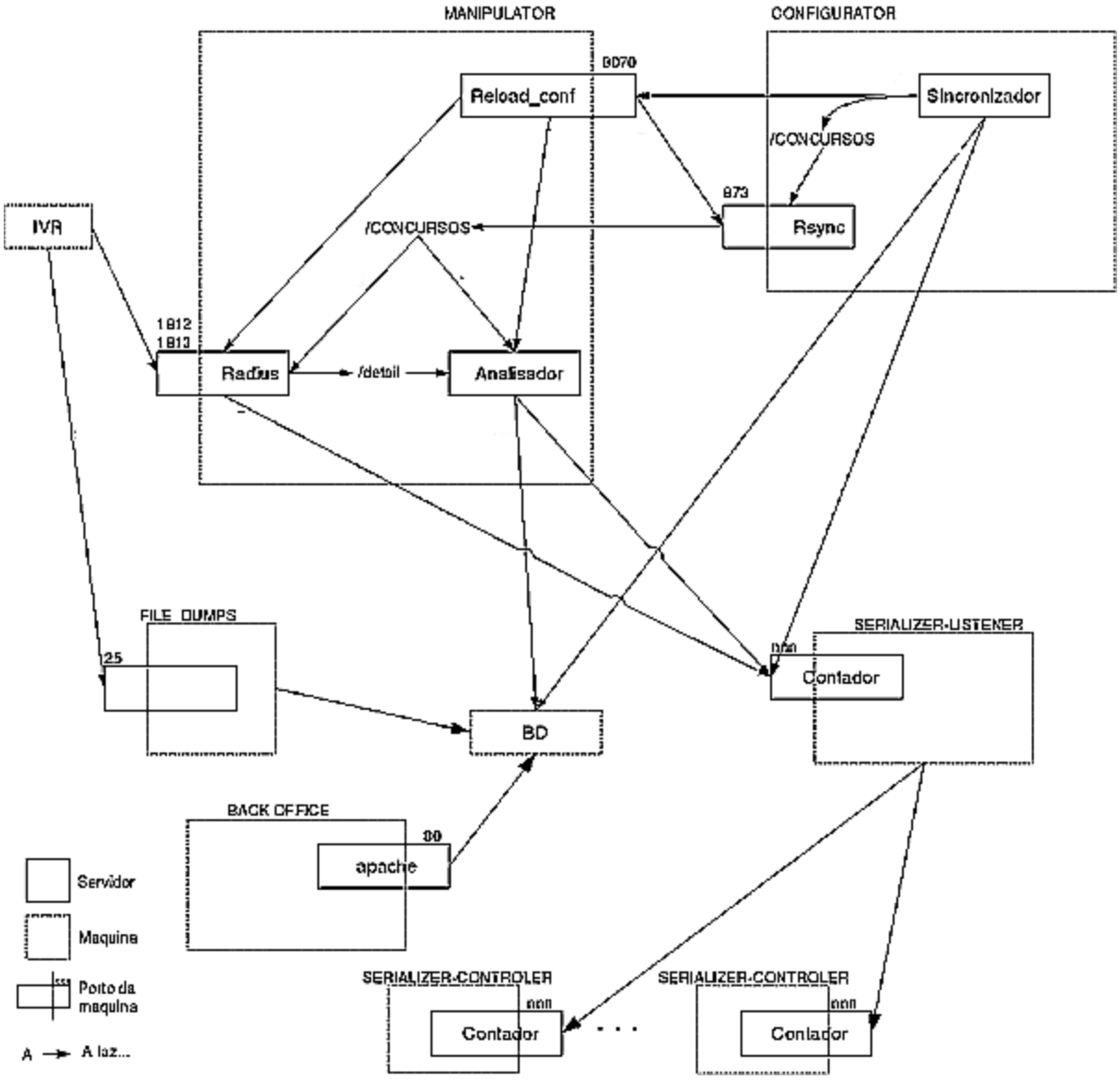

Figura 2: Fluxo de controlo de Concursos

3.1.4 Conclusão

Em termos de *softskills* foi uma experiência positiva, tendo sido possível conquistar um elemento abertamente hostil, envolvendo-o no projeto e fazendo sentir que fazia parte da equipa. Na entrega final, ficou a sensação de uma vitória de todos.

Em termos técnicos é a opinião do autor que teria sido mais eficiente a reescrita do sistema em vez de correção do código existente à altura da entrada no projeto, já que tinham sido feitos alguns erros de arquitetura, tais como não guardar o estado de processamento dos telefonemas de um modo persistente, considerando que os serviços nunca morriam. Estes erros foram remediados mas resultaram num sistema mais complexo do que se tivesse sido desenhado de raiz com esta premissa. Não é simples tomar uma decisão tão radical a meio de um projeto a decorrer, e a inexperiência do autor em processos negociais não o permitiu impor esta opção.

A solução implementada ainda hoje é usada, embora tenha sofrido melhoramentos. Trata-se de um sistema altamente estável, sendo muito raro a necessidade de intervenções no motor do sistema.
A documentação inicial ainda hoje é usada como referência de suporte.
Durante o ano de 2014, numa altura de pico, foram registadas acima das mil chamadas por segundo.

3.2 Projeto LIBRA

3.2.1 Enquadramento

No início de 2011 a Optimus e a Vodafone celebraram um acordo de partilha de infraestrutura de Fibra. Deste modo, a Optimus ao angariar clientes em zonas de fibra Vodafone, poderia contar com o aprovisionamento técnico do serviço pela Vodafone, ficando a Optimus responsável apenas pela faturação, e *vice-versa*.

Este acordo trouxe alguns desafios técnicos, um dos quais a troca de informação de avarias de equipamentos da rede partilhada entre a Optimus e a Vodafone, para efeitos de suporte telefónico.

Os avisos de avaria na rede da Optimus eram dados por avisos *SNMP* [18] - avisos muito baixo nível, praticamente ao nível de hardware, usando o protocolo de rede UDP [19]. A Vodafone não tinha capacidade de receber este tipo de avisos, nem era de interesse da Optimus enviar todos os avisos de todos os seus equipamentos para a Vodafone. No sentido inverso, a Vodafone apenas conseguia enviar pedidos HTTP, que a Rede Optimus não sabia lidar (apenas tratavam avisos *SNMP*).

3.2.2 Definição do Sistema

O projeto LIBRA foi elaborado para que fosse possível haver uma troca de informação entre as duas empresas, de modo fiável e sem falhas. Do lado da Optimus, foram envolvidas diversas equipas, nomeadamente:

- RT - Rede técnica Optimus.
  - Responsável pelos equipamentos de emissão dos avisos *SNMP*.
- ISP - *Internet Services Provider* (equipa do autor).
- RC - Rede corporativa Optimus.
  - Responsável pelo controlo de tráfego entre a RT e ISP.
- Tibco – *Middleware*- sistema de interligação que permite a comunicação entre vários sistemas, usando sempre o mesmo protocolo [21].

3.2.3 Intervenção

Coube ao autor desenhar e coordenar a equipa de desenvolvimento do sistema.

O maior desafio deste projeto foi a componente técnica:

- Desdobrar os avisos para serem enviados a múltiplos sistemas – para a Vodafone e para a equipa interna de qualidade de dados (para poderem auditar os dados enviados).
- Tratar milhares de avisos vindos de todos os equipamentos de rede Optimus.
- Comunicar com vários sistemas em diversos dialetos – SOAP e *SNMP.*

- Garantir a comunicação entre várias redes existentes na empresa (ACLs[i]).
- Garantir a entrega dos avisos *SNMP* nos diversos sistemas.

A peça chave do sistema foi uma componente nova de processamento de pedidos em paralelo chamada *Catapult.*
Foi montado um servidor para estar atento a todos os avisos *SNMP* enviadas pela RT. Esses avisos eram inseridos diretamente no *Catapult*, que os desdobrava em tantas ações quanto quantas as necessárias. O *Catapult* garantia a comunicação e entrega correta dos avisos.
A figura 3 apresenta os fluxos entre estes vários sistemas.

Todas as componentes foram escritas em Perl, tendo o autor escrito a *Catapult* e o servidor de receção de avisos *SNMP*.

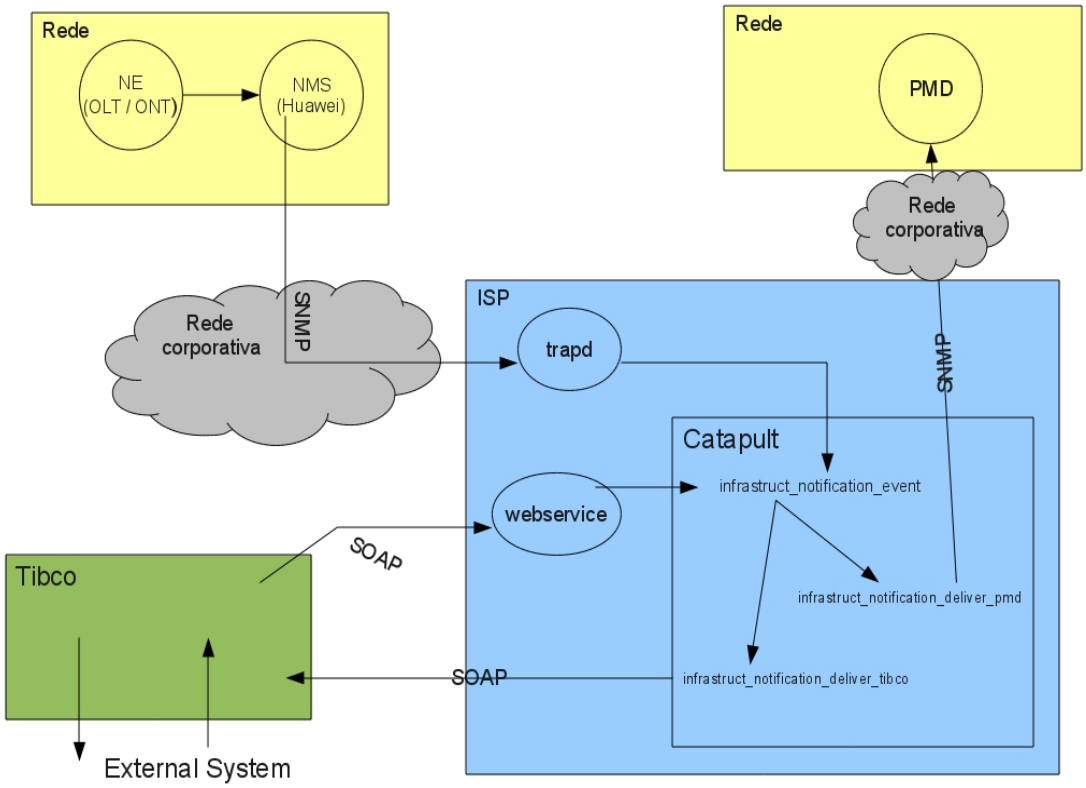

Figura 3: Sistema de Troca de Informação LIBRA

3.2.4 Conclusão

Este projeto foi bastante exigente devido às diversas tecnologias envolvidas, desde as de baixo nível com avisos *SNMP*, até à comunicação com uma *framework* de *middleware* TIBCO, usando SOAP.

---

i Access control list – definição de permissões de acesso

A eficiência do sistema teve de ser muito trabalhada para garantir que não fossem perdidos avisos. Um exemplo foi ter sido necessário substituir o servidor de escuta de avisos *SNMP*, já que o que inicialmente foi utilizado não aguentava o volume de avisos - cerca de trezentos mil diários. Para otimizar ao máximo a eficiência, o autor implementou o servidor substituto, que apenas lidava com os avisos relevantes, ignorando todo o processamento desnecessário para a função.

O projeto foi um sucesso, estando, à data de escrita deste texto, ainda em funcionamento.

3.3 Webselfcare Convergente Optimus

Este projeto foi escolhido para ser descrito devido aos múltiplos desafios que apresentou. Foi o projeto de maior complexidade técnica, tática e de interação humana dos então doze anos de experiência profissional do autor. Apesar de todos os problemas, foi o projeto que mais satisfação pessoal e profissional gerou, permitindo o uso pela primeira vez de várias tecnologias e metodologias.

3.3.1 Enquadramento

Um dos componentes mais importantes de uma empresa de telecomunicações é o serviço de suporte. Quanto mais pedidos de apoio recebidos telefonicamente, maior o tempo médio de espera dos clientes até serem atendidos. Uma maneira de evitar as chamadas, é de colocar serviços online onde os clientes podem efetuar alterações, configurações, compras, pedidos sem ajuda de um operador. Estes serviços agrupados chamam-se *selfcare* (auto ajuda). Os serviços desta natureza, de acesso via *browser* de Internet, chamam-se *web selfcare.*

Em 2010, a Optimus encontrava-se na fase final de um longo processo de convergência de marca. Este processo visava eliminar a segmentação da sua oferta comercial entre os serviços moveis (voz móvel – telemóveis, serviço Home, dados móveis – serviço kanguru) e fixo (voz fixa – voip e serviço voz tradicional, dados fixos ADSL, Fibra, circuitos dedicados, *hostings*,...).

No campo técnico, a Optimus tinha vários *selfcares:*

- *Selfcare* Optimus Particulares.
    - Destinava-se a serviços de telefone móvel para clientes particulares.
    - URL: www.optimus.pt/Particulares
- *Selfcare* Optimus Empresariais.
    - Destinava-se a serviços de telefone móvel para clientes empresariais.
    - URL: www.optimus.pt/Empresas
- *Selfcare* Soluções Fixas.
    - Destinava-se a serviços de Internet sobre linhas fixas para clientes empresariais.
    - URL: www.novis.pt
- *Selfcare* Kanguru
    - Destinava-se a serviços de Internet móvel, apenas para acesso de dados, para clientes particulares e empresariais.
    - URL: www.kanguru.pt
- *Selfcare* Home
    - Destinava-se a serviços de telefone fixos sobre tecnologia móvel.
    - URL: www.optimus.pt/Home

Esta situação era caótica. Um cliente Optimus com vários serviços, teria de saber explicitamente quais dos *selfcares* deveria usar para cada serviço.

Foi tomada a decisão de criar um *selfcare* convergente.

A Optimus contratou uma empresa externa independente e imparcial, a *log* [22], para efetuar um levantamento dos diversos *selfcares,* com intuito de efetuar

- Levantamento de Requisitos & Use Cases.
- Design Centrado no Utilizador.
- Codificação HTML + CSS + JavaScript.
- Desenho de Arquitetura Aplicacional.

Com base neste trabalho, a Optimus convidou alguns fornecedores de confiança, entre os quais a Outsystems [23], Wedo [24] e a própria *log*, a fornecerem propostas de implementação do novo *selfcare*.
Na altura do pedido de propostas, o autor pertencia a uma equipa denominada por *Internet Services Provider - Project Development*, (ISP/PD), que decidiu pedir autorização para também apresentar uma proposta de desenvolvimento. Estava convencida que reunia a experiência e conhecimento para fornecer a melhor solução para a Optimus. O pedido foi prontamente autorizado pela administração.

Os primeiros parágrafos do pedido de proposta resumiam muito bem o que se pretendia:

“ (…) O presente RFP visa a consulta a algumas empresas, por parte da Sonaecom, para a implementação do WebSelfCare – Massbusiness (WSC-MB), integrando as várias áreas de negócio da Optimus, nomeadamente: Comunicações móveis MB/Corporate; Kanguru MB/Corporate e Soluções Fixas MB/Corporate.

No âmbito desta consulta pretende-se também receber propostas para a implementação do sistema de login único, que irá suportar a identificação/autenticação de utilizadores/clientes da Optimus nos vários sites web. (…)”

O autor, em conjunto com outro colega de trabalho, elaborou um plano de desenvolvimento pormenorizado, das duas componentes, tendo ficado estimado 8 meses de desenvolvimento para o *selfcare* e 7 meses para o sistema de login único.

Esta proposta foi aceite e deu origem a nove meses de trabalho intenso, que em termos de equipa de desenvolvimento envolveu sete programadores júnior, três recursos sénior e uma gestora de projeto.

3.3.2 Definição de equipa de trabalho

Devido à longa duração do projeto, a equipa do autor decidiu separar o esforço de desenvolvimento em duas equipas, organizadas numa estrutura hierárquica, consistida por:

Responsável de Área

Figura de autoridade, para garantir o desbloqueio de imprevistos estruturais.

Gestor de Projeto

Responsável

- pela gestão de projeto em todas as suas atividades e itens relacionados.
- por garantir o cumprimento dos prazos.
- por alertar o Responsável de Área de eventuais problemas.

Líder Técnico de Projeto

Responsável

- pela análise técnica e de arquitetura de projeto.
- pelas atividades de desenvolvimento técnico de projeto.
- pela gestão da equipa técnica de programadores.
- pela integridade do código produzido.
- pela identificação de todas as necessidades técnicas em tais como
  - a arquitetura de sistemas.
  - a gestão de Base de Dados.
  - questões de segurança.
  - realização de testes.
  - gestão da configuração.
  - interligação de ambientes de desenvolvimento, testes e produção.
- por providenciar o suporte técnico necessário ao Gestor de Projeto e Programadores, sempre que solicitado.

Programador

Responsável pela implementação de código de acordo com o plano técnico definido pelo **Líder Técnico de Projeto**, de acordo com as melhores práticas e políticas de desenvolvimento de software.

O autor ficou com o cargo de Líder Técnico de Projeto pela implementação da componente de login único, ficando alocado dois programadores para o efeito. A implementação da componente do *selfcare* ficou a cargo de outro colega, ficando alocado três programadores. A equipa do autor não dispunha de tantos programadores livres, havendo na altura apenas dois que tinham de continuar em projetos já adjudicados. Desse modo, foi necessário efetuar contratações de um modo acelerado.

O autor, ao prever a falta de tempo que se avizinhava, escreveu um sistema de formação modular. Os programadores novatos passariam duas semanas a resolver desafios na linguagem de programação Perl. Deste modo, de modo autónomo, passavam pelos conceitos mais importantes e aprendiam as regras de programação da equipa. O sistema foi tão bem sucedido que ainda hoje é utilizado no processo de formação de novos elementos. Deste modo, mesmo antes do início do projeto, a equipa já tinha conquistado um sucesso.

3.3.3 Definição do Sistema – Login Único

Um sistema de login serve para autenticar uma pessoa, para garantir, de algum modo, que a pessoa que se está a autenticar é quem diz que é. Quando um sistema de login é partilhado por vários sistemas, ou seja ao efetuar uma autenticação num, fica-se automaticamente autenticado nos outros, costuma-se usar o tempo login único.
Construir um sistema destes para uso na Internet tem alguns desafios específicos, tais como a segurança, a propagação e validação da autenticação,...
De seguida será apresentado os problemas e as soluções escolhidas para implementar o sistema proposto, ao qual foi dado o nome **LUNO**: Login Único Na Optimus

3.3.3.1 Conceitos Básicos - Navegação na Internet

A interação com a Internet na nossa sociedade é algo que é banal. Aceder a um sítio através do computador pessoal, de um *tablet* ou de um telemóvel é algo tão natural como ligar a televisão ou rádio. Para garantir a clareza da explicação, convém visitar alguns conceitos básicos.

Página web

Página de conteúdo existente na Internet, mais concretamente na *World Wide Web.*
Pode ser uma página estática, escrita numa linguagem própria – HTML, ou uma página gerada dinamicamente por diversas linguagens de programação, tipicamente para aceder a informação alojada em base de dados.

Servidor de alojamento de páginas:

Local onde são colocadas as páginas web.

Sítio de Internet

Um conjunto de páginas web acessíveis usando um navegador / *browser* de Internet.
É caracterizado por ter um endereço de acesso, denominado por URL.

URL

Endereço de um recurso da Internet, com o formato

```
protocolo://domínio:porto/caminho/recurso?querystring
```

- protocolo, poderá ser HTTP, HTTPS, FTP, entre outros.
- domínio designa o servidor de alojamento de páginas.
- porto designa o porto onde o servidor está à escuta.
- caminho especifica o local do recursos dentro do servidor.
- *querystring* é um conjunto de parâmetros a ser enviado ao servidor.

O protocolo indica *como* ligar ao servidor, o domínio especifica *onde* ligar, e o resto especifica *o que* está a ser pedido.

Cookies

Trata-se de um pequeno pedaço de informação enviado por um sítio que é guardado no *browser* do utilizador, numa diretoria do sistema de ficheiros onde o browser está instalado.

Foram desenhados para servirem como um sistema fiável de manutenção de estado.
Cada vez que o utilizador acede ao mesmo sítio, o *browser* entrega sempre os dados contidos no *cookie* desse sítio.
Por razões de segurança, os *cookies* estão trancados ao domínio, sendo apenas entregues ao domínio/sub-domínio que os criou.

Ex: O www.optimus.pt NUNCA vai receber os *cookies* do www.zon.pt
O www.optimus.pt e areacliente.optimus.pt recebem os *cookies* de optimus.pt

3.3.3.2 Conceitos Básico - Autenticação e sessão

A navegação por sites usando um *browser* tipicamente não contempla estado, ou seja, se um utilizador aceder duas vezes de seguida ao mesmo URL, o servidor não terá noção que se trata da mesma pessoa. Os *cookies* foram criados para que pudesse haver manutenção de estado entre pedidos. Este mecanismo é fundamental para o sistema de autenticação.

A autenticação é feita tipicamente usando um par de credenciais: login / password, que é fornecido pelo utilizador quando tenta aceder a conteúdo restrito.
Quando as credenciais chegam ao servidor, são validadas usando dados guardados normalmente em base de dados controladas pelo servidor. Se as credenciais forem válidas, é criado uma sessão autenticada, ou seja:

- é gerado um identificador enorme, difícil de adivinhar.
- esse identificador é guardado em base de dados, juntamente com um tempo de validade.
- o identificador enorme é colocado no *cookie*, no *browser* do utilizador.

A sessão considera-se válida enquanto não passar o tempo de validade desse identificador.

Se o utilizador voltar ao mesmo sítio, o seu *browser* entrega o *cookie* do domínio do sítio ao servidor. Este lê o identificador de sessão que foi anteriormente guardado e verifica na sua base de dados se existe e se

continua válido. Se ainda estiver válido, permite o acesso ao conteúdo protegido, caso contrário pede novamente as credenciais.

3.3.3.3 O Sistema de Autenticação

Como referido anteriormente, os *cookies* não são partilhados entre domínios, portanto um site não consegue ler o identificador de sessão colocado por um domínio diferente.
Por exemplo, se um cliente Optimus, que se autentica no site www.optimus.pt (identificador de sessão em .optimus.pt), não será considerado autenticado no site www.wow.pt, já que este nunca receberá os *cookies* do optimus.pt, apenas do .wow.pt.

A solução adotada na implementação do LUNO baseou-se em delegação ao servidor LUNO, por DNS, de sub-domínios dos diversos sítios, para que ele pudesse manipular os *cookies* adequadamente. Deste modo o sistema contemplava três peças chave que podem ser vistos na figura seguinte.

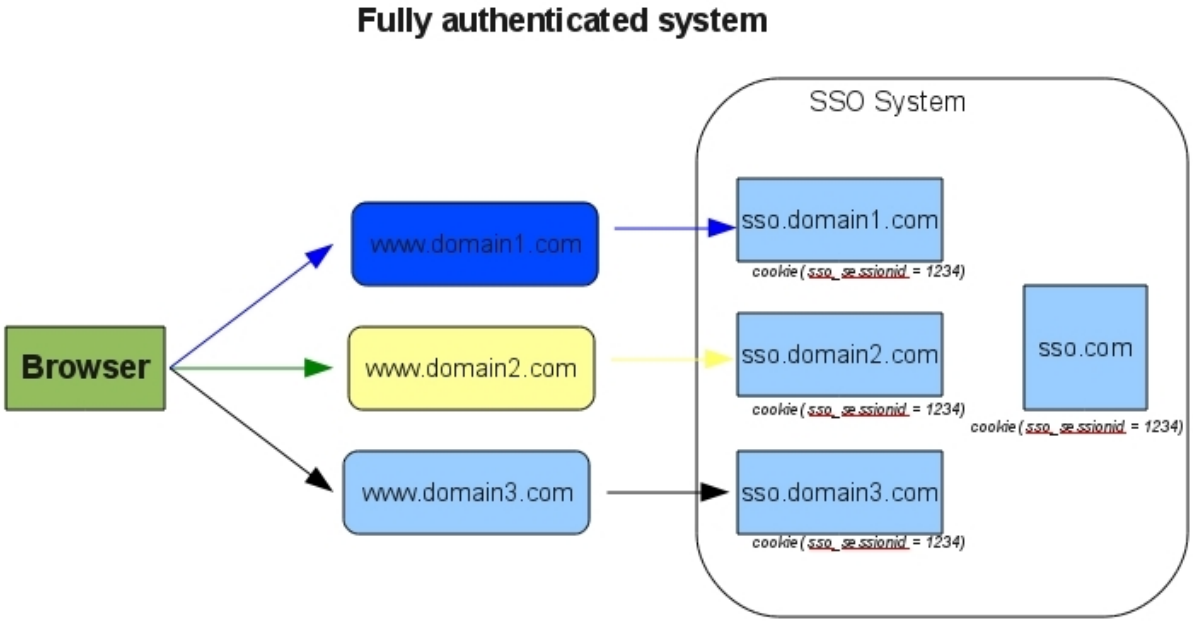

Figura 4: Fluxo de autenticação

As três componentes são:

- O sistema de autenticação.
    - sso.com
- Os endereços dos sítios de conteúdo.
    - www.domain1.com, www.domain2.com e www.domain3.com
- Os endereços dos sub-domínios.
    - sso.domain1.com, sso.domain2.com, sso.domain3.com
    - Todos apontam para o endereço físico (IP) do servidor sso.com[ii].

ii Efetuado através de configuração de DNS

- Estes sub-domínios servem apenas para que o sso.com possa fazer reencaminhamentos para colocar a sessão num domínio diferente.

O fluxo completo e em maior detalhe pode ser visto na figura seguinte.

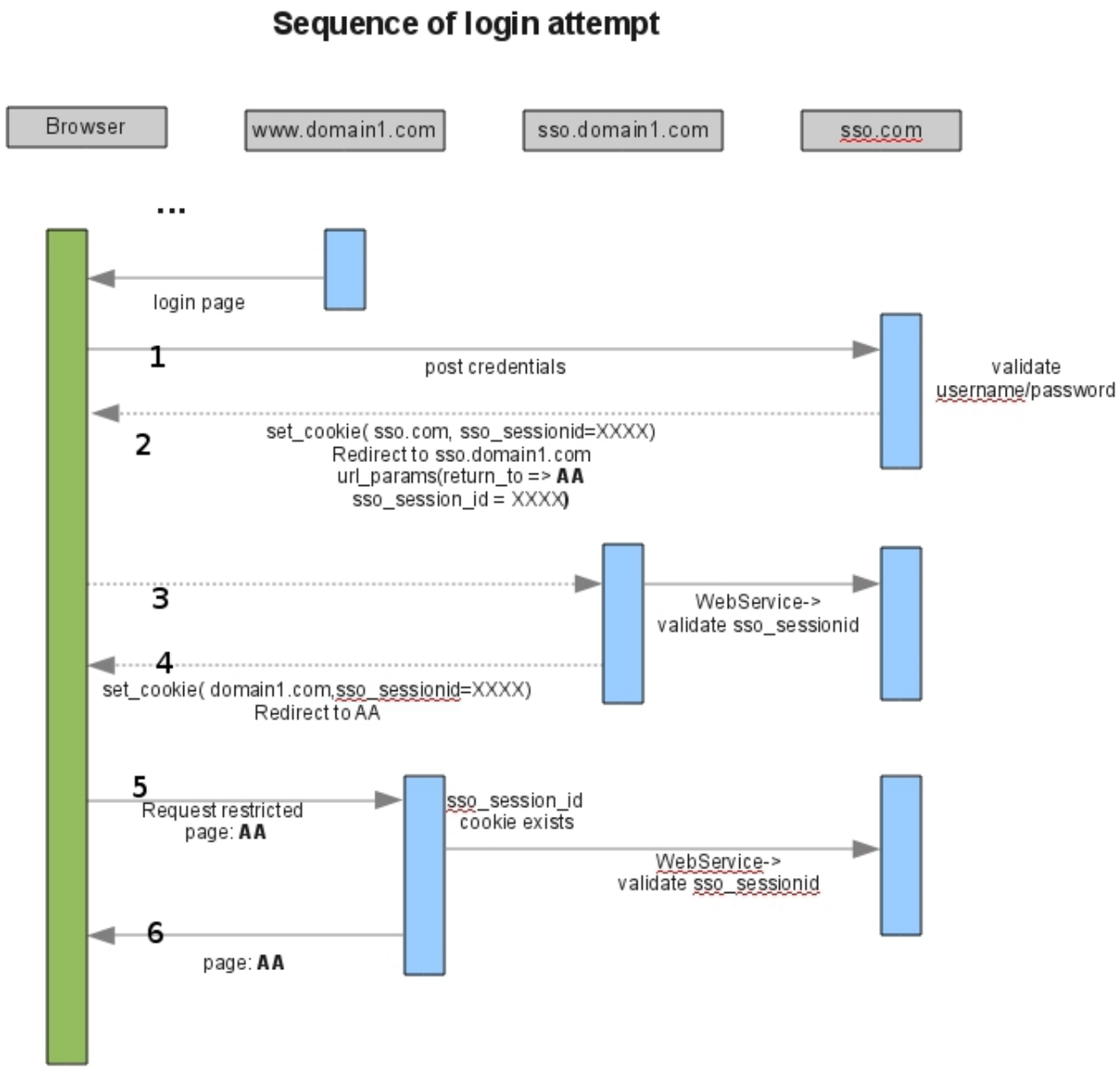

Figura 5: Sequencia de Autenticação bem sucedida

Neste caso, o utilizador tentou entrar na página **AA** do domínio www.domain1.com, que necessita de autenticação e é lhe pedido as suas credenciais. O fluxo apresentado é resultado da ação de envio das credenciais, e embora resulte em várias reencaminhamentos, não necessita de qualquer intervenção do utilizador, acontecendo de um modo transparente.

1 – São enviados para o servidor de autenticação, *sso.com* :

- as credenciais.
- a indicação que se quer aceder à página **AA** (página de retorno).

(O servidor valida as credenciais).

2 – O servidor manda o *browser*:

- colocar um *cookie* no domínio *sso.com*, com a chave de sessão.
- redirecionar-se para o domínio *sso.domain1.com*, com os parâmetros (*querystring)* chave de sessão e página de retorno **AA.**

3 – O sítio *sso.domain1.com* valida, com sucesso, a chave de sessão junto ao servidor sso.com.

4 – O sso.domain1.com manda o *browser*:

- colocar um *cookie* no domínio domain1.com com a chave de sessão.
- redirecionar-se para a página **AA.**

5 – O browser chega ao site *www.domain1.com*/**AA**, que necessita de autenticação. O site deteta a existência de uma chave de sessão no cookie *domain1.com,* e valida-a, com sucesso, junto ao servidor sso.com.

6 – O site *www.domain1.com* devolve a página **AA.**

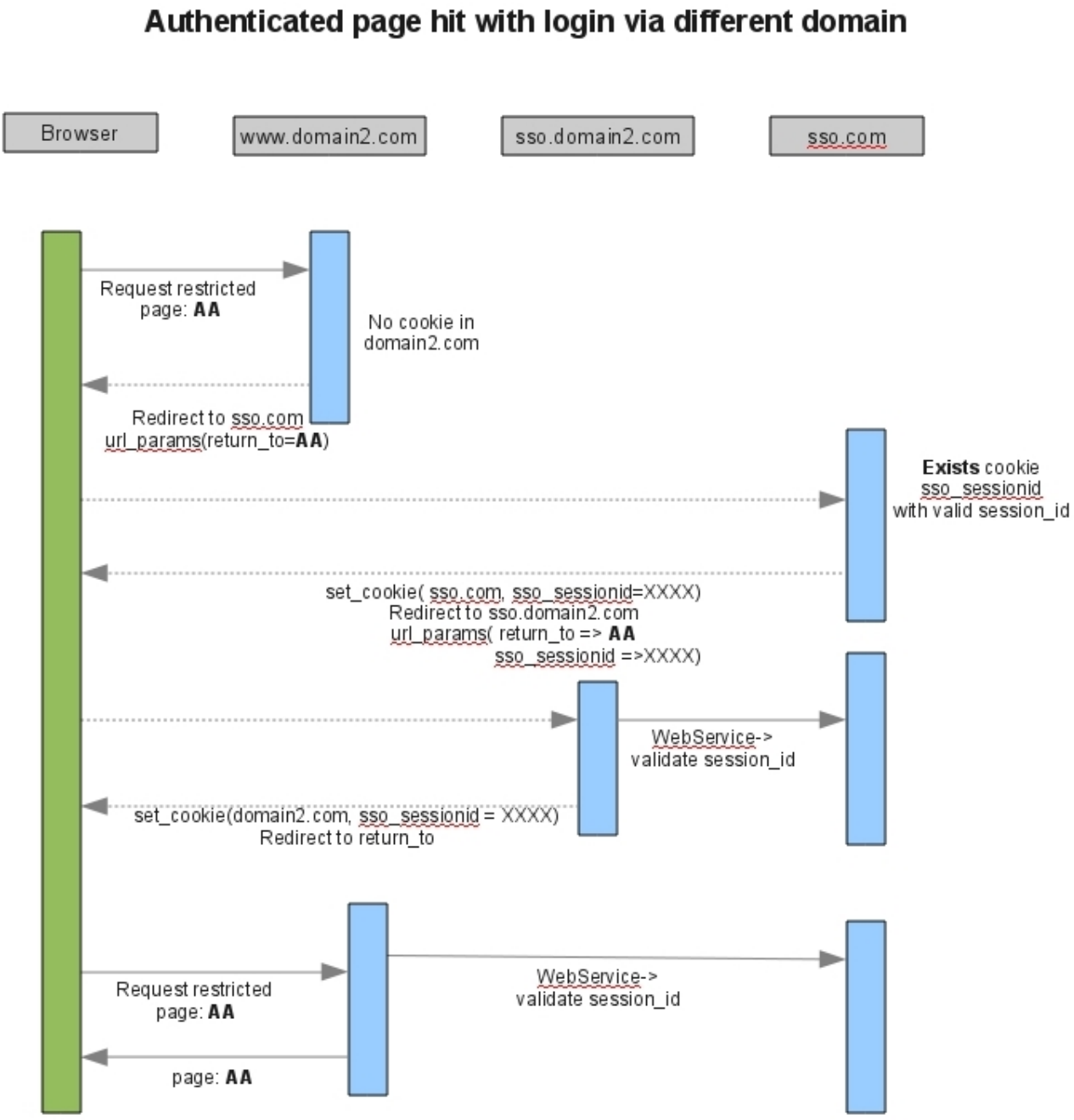

Figura 6: Acesso autenticado através de domínio diferente

Esta figura demonstra o que acontece quando o mesmo utilizador visita um domínio diferente do que onde se tinha autenticado. O fluxo é semelhante, não havendo o pedido de credenciais já que, quando se chega pela primeira vez ao servidor de autenticação (sso.com), já existe um *cookie* de sessão no domínio *sso.com* Deste modo, apenas é feito o fluxo para colocar a sessão no novo domínio, *domain2.com.*

Neste fluxo, não foi necessário pedir novamente as credenciais ao utilizador.

3.3.3.4 Conceitos de Portfólio

As secções anteriores referentes à autenticação, descrevem o conceito de validação de uma pessoa usando credenciais. No final do processo, é possível identificar a pessoa autenticada, no entanto não é possível determinar os serviços comerciais que essa pessoa tem com a Optimus. Para conseguir essa ligação, é necessário manter informação que sirva de elo entre entre a pessoa com credenciais e a pessoa com ligação contratual.

Optou-se por implementar um mecanismo genérico de ligação com sistemas externos, ao qual chamou-se portfólio, que podia ser composto por vários elementos chamados associações. Cada associação constituía um elo de ligação com um único sistema externo.

Uma associação era caracterizada por

- Um nível de permissão de acesso ao sistema externo, administrador ou utilizador – role.
- O identificador do sistema externo – system.
- O nome da chave de ligação com o sistema externo – key_name.
- O valor da chave de ligação – key_value.

Com esta definição, um exemplo completo de uma pessoa-cliente no LUNO teria o seguinte aspeto:

Utilizador

```
Nome:     José Carlos
Estado:   Ativo
Password: nomNomNom
```

Login

```
Email: jose.cpereira@optimus.pt
Móvel: 931003540
```

Pessoa

Associações

Role	system	key_name	key_value
Admin	billing	client_number	1.234567
User	billing	mobile	931013540
User	isp	username	nn1234567
Admin	wholesale	client_id	7897

Portfólio / Cliente

3.3.3.5 Intervenção

O autor desenhou a solução, coordenou e participou ativamente no processo de desenvolvimento. Algumas das opções de implementação estavam condicionadas por razões de eficiência e rapidez de execução, já que existia maior experiência de utilização de algumas tecnologias, nomeadamente:

- Perl: A linguagem de programação usada.
- Linux: O sistema operativo, gratuito e em formato código aberto.
- Oracle: A base de dados oficial de soluções estruturantes da Optimus.
- Apache: O servidor de páginas, em formato código aberto, o líder de mercado em servidores.
- Twiki: Um servidor de páginas dinâmicas para partilha de informação e documentação, gratuito e de código aberto [25].
- CVS: Um sistema de controlo de versões - um sistema para ajudar a registar as alterações efetuadas no código [26].
- *Framework de Webservices* SOAP:
    - *Webservice* [27] - Tecnologia que permite a interação entre máquinas.
    - SOAP – um protocolo específico de *webservice.*

Os desenvolvimentos do LUNO iniciaram-se a 14 de Janeiro de 2011, com o autor e um programador júnior. As primeiras duas semanas foram dedicados a elaborar documentação detalhada do sistema, com especial enfoque no sistema de autenticação, sendo as figuras 5 e 6 exemplos do resultado final. Em paralelo, o programador efetuava a formação descrito na secção 3.3.2.

O LUNO foi desenhado para ser consultado principalmente por aplicações e não por humanos. O processo de autenticação, como se pode ver pelas figuras 5 e 6, têm uma participação mínima de pessoas. A partir do momento que as credenciais são envidas para validação, toda a interação é feita:

- Entre o navegador de páginas e os servidores de autenticação - figura 5 pontos 1, 3 e 5.
- Entre os vários servidores de autenticação – figura 5, ponto 3.
- Entre o sítio visitado e o sistema de autenticação – figura 5, ponto 5.

A interação entre os servidores (chamada interação *machine to machine*) é garantida usando webservices. Os *webservices* não são mais do que métodos que se podem invocar através da rede, passando parâmetros e obtendo uma resposta de sucesso ou erro. O LUNO disponibilizou 46 destes *webservices*, que permitiam efetuar ações como as de validação de credenciais, criação de logins, validação de chaves de sessão, criação e remoção de associações,...

Por norma interna da equipa de desenvolvimento, quando é implementado qualquer *webservice* também é obrigatório:

- Escrever um ou mais testes unitários de validação de casos de utilização.
    - Os testes mais básicos validam pelo menos se é devolvido um código de erro na ausência

de parâmetros obrigatórios, se o número de parâmetros de entrada enviados é verificado, se são retornados códigos de sucesso e erro.

- Escrever documentação indicando os parâmetros de entrada, os códigos de sucesso e erro retornados, exemplos de invocação e descrições úteis.

Devido ao número elevado de *webservices* a implementar, o autor optou por seguir um paradigma de *test-driven-development* (TDD [28]), que consiste em escrever os testes unitários ANTES de desenvolver as funcionalidades. Esta aproximação traz várias vantagens, já que fica-se logo com exemplos funcionais de uso e ao mesmo tempo obriga a pensar de como o sistema será usado em código, apanhando assim eventuais erros conceptuais logo de início.

O programador que iniciou os desenvolvimentos chamou logo à atenção que perdia imenso tempo a copiar blocos de código repetidos entre os diversos *webservices,* nomeadamente*:*

- No *webservice* em si, com a validação do número dos parâmetros de entrada, a validação sintática dos parâmetros, a execução de alguma ação específica e no retorno de erros e sucesso
- Nos testes unitários ao *webservice*, na validação de códigos de sucesso e de erro

O autor ouviu as queixas e concordou com a observação: efetuar 46 pedaços de código praticamente iguais seria, além de uma tarefa aborrecida, um risco já que aumentaria a probabilidade de ocorrência de erros devido a descuido. Decidiu então escrever um gerador de código. Todos os *webservices* seriam documentados em formato texto[iii]. Com base nessa documentação, todas as peças de código repetitivas seriam geradas.

A experiência correu tão bem, que o autor aproveitou para acrescentar às funcionalidades do gerador as de geração da documentação dos *webservices* e exemplos de invocação dos mesmos. Deste modo, mal se criava um novo *webservice*, ficava-se logo com testes e documentação para partilhar.

Passado cerca de 5 semanas de trabalho, um segundo programador entrou na equipa, efetuou a formação e começou a contribuir ativamente para o projeto. Este facto causou um problema já que, por motivos de segurança, os programadores não tinham acesso de escrita ao sistema de controlo de versões, CVS. Assim, cada programador desenvolvia uma funcionalidade e pedia ao autor para a integrar no CVS, para que o colega ao lado a pudesse receber. Esta tarefa nem sempre era simples já que era frequente existir conflitos, quando ambos os programadores mudavam código no mesmo ficheiro, que obrigava o autor a tentar perceber o que se pretendia.

Este esforço começou a prejudicar a eficiência da equipa, que levou o autor a decidir mudar de sistema de controlo de versões, passando para o *git* [30] – um sistema de controlo de versões escrito pelo criador do Linux, que além de funcionar de modo distribuído (não tendo necessidade de um servidor centralizado para manter estado), supostamente tinha mecanismos de resolução de conflitos muito avançados. O tempo

iii Foi escolhido um formato simples de ler - YAML [29]

perdido com a adaptação desta nova peça foi rapidamente recuperado com o tempo poupado em integrações. Os programadores ganharam a capacidade de registar as suas alterações e receberam um voto de confiança, já que passaram a ter um papel ativo em resolução de conflitos.

Os desenvolvimentos do LUNO foram dados por terminados a 13 de Maio, 2011.
A equipa passou a estar alocada aos desenvolvimentos do selfcare.

3.3.4 Definição do Sistema – Webselfcare

O objetivo deste projeto foi de eliminar os vários *selfcares* existentes associados aos vários produtos/linhas comerciais. O trabalho efetuado pela empresa *log* resultou num documento que serviu como base para o desenho do novo *selfcare* da Optimus, ao qual foi dado o nome **LIMBO**: *Large Integrated Mass Business Optimus*.

O plano de implementação deste sistema contemplou quatro blocos funcionais:

- **Migração**
  Os clientes registados nos diversos *selfcares* antigos deveriam poder ser migrados para o novo sistema de um modo simples.

- **Registo**
  Os clientes que ainda não estavam registados no *selfcare* deviam poder efetuar o registo de modo autónomo, ficando registados com o nível correto de permissões.

- **Funcionalidades de selfcare**
  Deviam ser disponibilizadas todas as funcionalidades existentes nos diversos *selfcares*.

- **Backoffice**
  Devia ser fornecido uma ferramenta de apoio às equipas de suporte, que permitisse ajudar a despistar problemas dos clientes.

3.3.4.1 Considerações técnicas

A análise da *log* tinha identificado cerca de sessenta e oito funcionalidades distintas a serem implementadas no novo *selfcare*, que obrigaria a integrar com sete sistemas distintos da Optimus, tais como o sistema de faturação, de envio de cartas, de envio de emails, de envio de SMS,...
A interação com estes sistemas seria extremamente penosa se o *selfcare* tivesse que interagir diretamente com cada um, já que todos sistemas tinham protocolos de comunicação distintos.
Para eliminar esta dificuldade, toda a comunicação seria efetuada usando um *middleware*, Tibco, para a comunicação, que disponibilizava os *webservices* usando sempre o mesmo protocolo de comunicação: SOAP. A equipa do autor já usava ativamente o Tibco, usando nove *webservices* seus.

Logo após a adjudicação do projeto à equipa do autor, foi comunicado pela Optimus que não seriam criados novos *webservices* Tibco no âmbito do projeto do *selfcare* convergente, sendo necessário aproveitar os existentes, já que o projeto não contemplava nenhuma nova funcionalidade, apenas juntava funcionalidades existentes. Este constrangimento gerou alguns problemas, obrigando a equipa encontrar algumas soluções criativas de engenharia, que são relatadas de seguida.

3.3.4.2 Migração

O levantamento efetuado pela *log* indicava que existiam três sistemas distintos usado pelos *selfcares* antigos para guardar as credenciais dos seus *logins*:

- Siebel para *logins* do tipo telemóvel.
- ISP para contas de serviços de Internet fixa.
- SGU para serviços de dados móvel (serviço Kanguru).

A hipótese de migrar todas as credenciais destes sistemas em *batch*[iv] para o LUNO foi colocado de lado por vários motivos, nomeadamente:

- Estes sistemas não tinham o conceito de perfil de permissão (administrador/utilizador do serviço).
- Os sistemas guardavam as passwords cifradas, cada um com um algoritmo diferente, não sendo possível a obter a as passwords originais para as transpor para o LUNO.
- No sistema ISP não havia um mapeamento direto com o sistema de faturação da Optimus, não sendo assim possível definir o elo comercial para construir as associações LUNO.
- O sistema Siebel tinha milhões de *logins* registados, muitos dos quais não eram usados no *selfcare*. Não fazia sentido exportar *logins* que depois não iriam ser usados.

Todos estes sistemas tinham *webservices* de validação de credenciais, daí decidiu-se que o LIMBO validaria as credenciais em todos os sistemas no ato de autenticação, sendo despoletado o processo de registo caso fossem validadas num dos sistemas antigos.

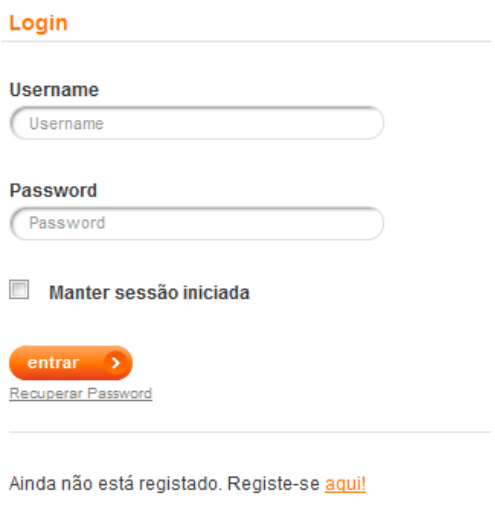

Figura 7: Página de login

iv Tarefa automatizadas sem que seja necessário intervenção humana

O processo de autenticação foi otimizado, para tentar minimizar o tempo de espera de um cliente no ato de validação. Deste modo, as credenciais eram sempre validadas primeiro no LUNO, para servir logo os clientes já registados. Ao falhar a validação, usava-se a análise do formato do *login* para tentar efetuar o mínimo de consultas aos sistemas, seguindo a seguinte lógica:

Formato do login	**Sequência de validação**	**Razão**
Contêm algum carácter não numérico	LUNO, ISP	Apenas o ISP tinha *logins* com caracteres não numéricos
Número de telemóvel	LUNO, Siebel, SGU, ISP.	O Siebel tinha principalmente registo de telemóveis
Apenas dígitos	LUNO, SGU, Siebel e ISP	Os identificadores de serviços Internet móvel eram numéricos

Tabela 1: Matriz de decisão de autenticação de login

3.3.4.3 Registo

O processo de registo foi desenhado para ser o mais flexível possível, garantindo ao mesmo tempo a máxima segurança. Um dos requisitos principais foi a necessidade de diferenciar os clientes particulares dos clientes empresariais, obrigando estes últimos ao registo por carta, enviada para a morada de faturação. Esta limitação foi para garantir que o acesso às contas de clientes de maior notoriedade, tais como tribunais, bancos, institutos públicos,... seria feita de modo mais controlado.

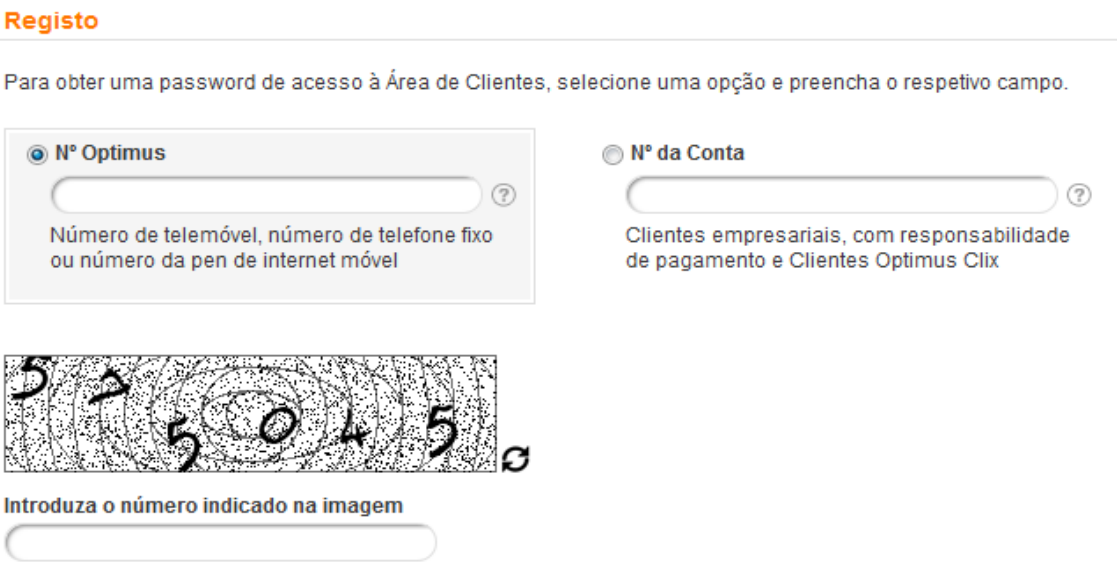

Figura 8: Página de registo

Pela figura 8, podemos ver os dois modos possíveis de registo, por número de telefone ou por número de cliente.

Registo por número de telemóvel:

> Um código de registo era enviado para o número indicado por SMS. Este código permitia o registo imediato na área de cliente. O LIMBO, consultando o sistema de faturação para obter a informação necessária, atribuía os níveis de permissões adequados ao novo *login*: perfil de utilizador sempre e de administrador se estivesse indicado como responsável de pagamento.

Registo por número de cliente:

> Este processo estava limitado aos clientes empresariais, sendo pedido o NIF e um número de fatura como dados de validação. O LIMBO validava os dados consultando o sistema de faturação. Se os dados estivessem corretos, era despoletando o envio de uma carta para a morada de faturação do cliente com um código de validação, juntamente com as instruções de como terminar o registo. O cliente, ao terminar o registo, receberia apenas um perfil de administrador.

3.3.4.4 *Funcionalidades de selfcare*

O objetivo principal do projeto do novo *selfcare era c*riar um conjunto de conteúdos e funcionalidades consolidadas.

Figura 9: O selfcare após validação de login

Tal como referido anteriormente, tinham sido identificado sessenta e oito funcionalidades distintas para implementar, que dependiam de sete sistemas distintos da Optimus. Durante a implementação foram detetados mais dois sistemas que tinham passado despercebidos à análise. A falha no levantamento das funcionalidades foi mais surpreendente, já que no final dos desenvolvimentos tinham sido implementadas duzentas!

Um problema resultante da convergência de vários produtos num único sítio, foi a dificuldade em saber as características das contas associadas a um *login*, logo após à sua autenticação. Isto era necessário para que se pudesse apresentar um menu lateral adequado ao cliente, que refletisse os seus serviços. Por exemplo, não faria sentido apresentar o serviço de configuração de endereços de email a um cliente com apenas um serviço de telemóvel.

Este problema não se punha com os vários *selfcares* anteriores, já que cada um apenas servia um determinado produto comercial, estando assim a classificação explicitamente feita.

Dado que não foi autorizado a implementação de novos *webservice*s específicos para o LIMBO, a classificação requeria em média invocações a seis *webservices*, alguns dos quais não eram muito eficientes, demorando largos segundos. A solução para este problema está descrita na secção 3.3.4.6 – Intervenção.

3.3.4.5 Backoffice

O backoffice é uma ferramenta indispensável para as equipas de suporte ao cliente. É através deste tipo de ferramenta que é possível ajudar os cliente a resolverem os seus problemas.

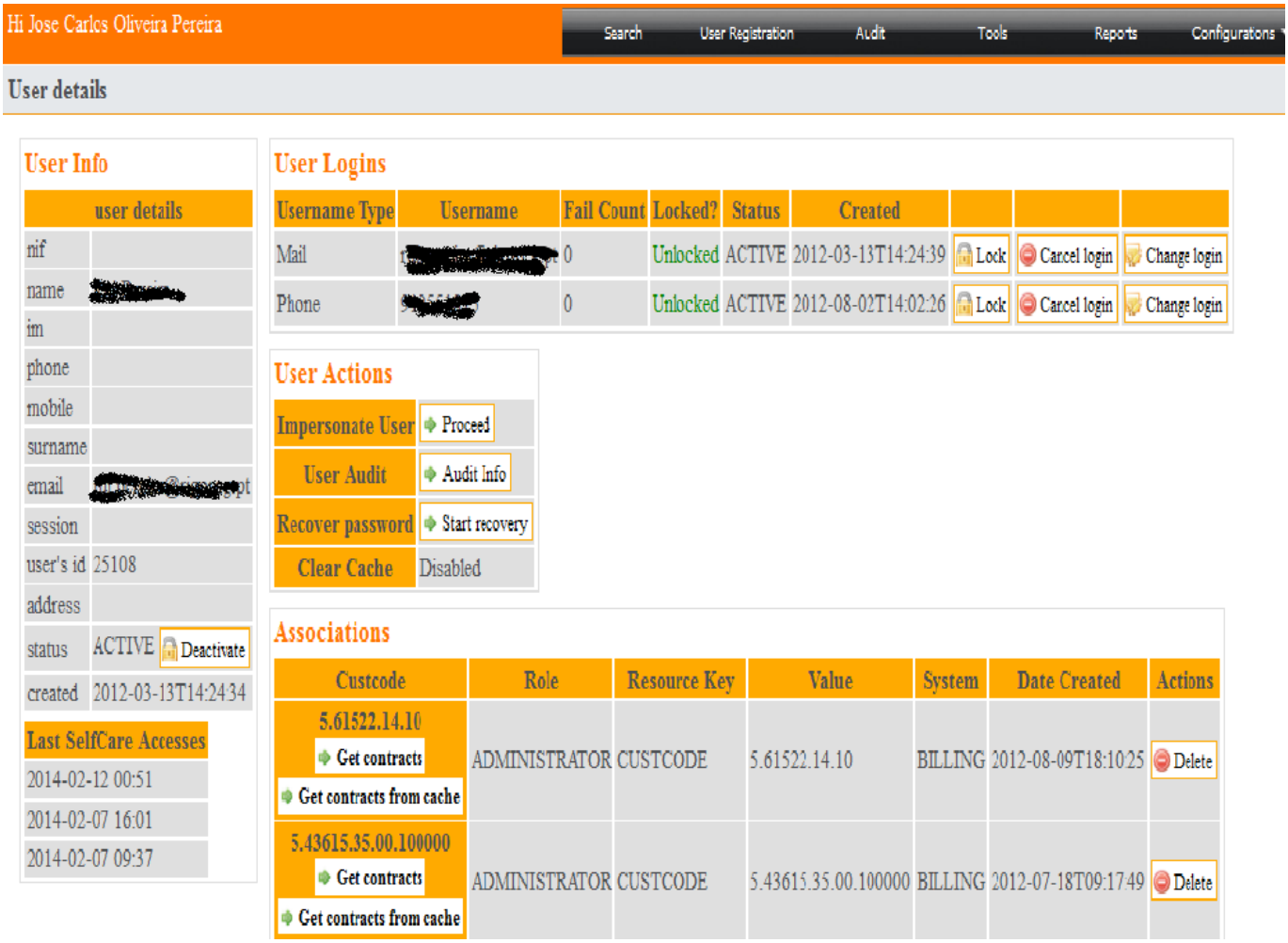

Figura 10: O backoffice do selfcare convergente

O backoffice do *selfcare* convergente é bastante simples, permitindo a pesquisa de clientes por número de telefone ou número de cliente, mostrando os dados numa visão consolidada do LUNO e LIMBO, como se pode observar na figura 10. A funcionalidade mais utilizada pelo suporte é o mecanismo que permite entrar no *selfcare* com o login de um cliente sem que seja necessário conhecer a password – *impersonate*. Deste modo é possível ver exatamente o que o cliente vê, simplificando o processo de apoio. É de referir que acessos efetuados deste modo não permitem visualizar os dados pessoais do cliente, nem efetuar alterações.

3.3.4.6 Intervenção

Os desenvolvimentos do LIMBO iniciaram-se no início de Fevereiro, 2011, altura em que o autor estava envolvido no projeto do LUNO, não tendo assim participado ativamente na fase inicial. O processo de contratação de programadores, foi efetuado em apenas duas semanas, onde foram feitas quinze entrevistas técnicas, das quais resultaram na contratação de três elementos. O autor participou em todas as entrevistas, tendo liderado a maior parte delas.

A participação ativa iniciou-se em Maio, altura em que o LUNO havia sido terminado. Deste modo, a equipa do LIMBO foi reforçada por dois programadores juniores e um líder técnico, ficando 2 líderes e 5 programadores. Nesta altura, os processos de registo e migração estavam na fase final, e foi iniciado o desenvolvimento das sessenta e oito funcionalidades do *selfcare*. Os mesmos problemas enfrentados no LUNO, o de conflitos entre código comum surgiu logo, mas o autor rapidamente migrou o sistema de controlo de versões de CVS para *git*, dando formação à equipa inicial do LIMBO.

Gestão de equipa

Durante algumas semanas o trabalho foi sendo feito de modo tranquilo, mas começou-se a notar que existia alguns problemas pessoais entre dois dos elementos da equipa. Os feitios não combinavam e apesar do autor ter tentado ajudar na resolução do conflito quer individualmente, quer em conjunto, a realidade era que não se gostavam e tinham deixado de se falar. Esta situação chegou ao ponto de ambos terem feito a mesma tarefa, sem darem por ela. As chefias sugeriram o afastamento de um dos elementos, mas o autor, além de gostar do trabalho de ambos, não queria lidar com o impacto negativo que a redução de capacidade causaria.
A solução surgiu após ter lido sobre um método de gestão de projetos que ajudava equipas a funcionar como equipas e ao mesmo tempo melhorava a eficiência do ciclo de desenvolvimento. Este método chamava-se *Scrum*. O autor sabia que não podia implementar a metodologia na sua plenitude, por ser demasiado diferente dos processos que estavam a ser seguidos, mas decidiu aproveitar alguns dos conceitos, nomeadamente:

- Dividir o projeto numa lista de tarefas que ainda faltava fazer.
- Cada recurso ficava responsável por uma tarefa de cada vez.
- Todas as manhãs, havia uma breve reunião de equipa, não mais de quinze minutos no total, com todos os elementos, onde cada um indicava o que tinha estado a fazer no dia anterior, e o que iria fazer nesse dia.
- Cada elemento podia indicar dúvidas ou pedir sugestões de resolução, embora se o tema fosse demasiado específico, ficava logo marcado uma reunião entre os interessados, fora da reunião matinal.
- Se um recurso descobrisse uma nova tarefa a ser executada, em vez de pegar logo nela, acrescentava à lista global de tarefas. Seria distribuída apenas no dia seguinte, caso se justificasse.

Esta mudança resultou muito bem, As conversas num fórum alargado ajudava a dissipar os problemas pessoais, e todos ficavam com uma visão geral do estado de desenvolvimento.
Outro efeito desta mudança foi que a lista de tarefas, em vez de diminuir, começou a crescer, à medida que se iam analisando melhor as funcionalidades. Este facto também foi positivo já que permitiu ganhar a perceção que não seria possível cumprir o calendário inicialmente proposto numa fase ainda embrionária. A lista foi passada para a direção que tutelava o *selfcare*, que a usou para indicar prioridades às tarefas, definindo assim o rumo dos desenvolvimentos.

Desafios técnicos

Um projeto desta envergadura apresentará sempre múltiplos desafios, nem que seja conseguir implementar os requisitos de uma maneira consistente, interligando todas as componentes. De seguida são apresentados alguns desafios que ocorreram neste projeto, nos quais o autor teve um papel fundamental na resolução.

Classificação de contas

Como explicado anteriormente, a classificação do cliente e suas contas no ato de autenticação era fundamental para poder ser apresentado um menu contextualizado aos seus serviços. Na primeira implementação, o processo de autenticação demorava cerca de um minuto para clientes simples (por ex, pré-pagos móveis) e até oito minutos em clientes empresariais com vários serviços (por ex, circuitos dedicados com serviços de vpns, fax, *hosting*,...). Estes tempos não são aceitáveis para serviços web.
O processo de classificação era efetuado em dois passos: a classificação do cliente e a classificação dos seus serviços, sendo o segundo passo o mais lento dos dois.
O autor decidiu alterar o processo de classificação, classificando apenas o cliente no ato de autenticação. Esta classificação dava apenas informação acerca da família de serviços que o cliente tinha, tais como cliente residencial pré ou pós pago móvel; empresarial móvel; empresarial fixo; kanguru (móvel de dados),... Com esta informação já era possível apresentar um menu com todas as funcionalidades que fizessem sentido para a família, falhando apenas em alguns casos em que apareciam opções não disponíveis para o portfólio de serviços contratados pelo cliente. Por exemplo, um cliente da família de serviços kanguru recebia um menu com opções de ativar o serviço de roaming, no entanto existem algumas ofertas kanguru que não contemplam esta opção. A classificação ao nível dos serviços do cliente era efetuada apenas quando o cliente entrava na operação específica do menu. No caso em que os serviços contratados do cliente não suportavam a opção escolhida, era apresentada uma mensagem de conforto, a indicar que a opção não estava disponível para os seus serviços.

Deste modo, o processo de autenticação passou a demorar cerca de três segundos, ficando o cliente logo dentro do *selfcare*. Se o cliente utilizasse uma funcionalidade que visasse os seus serviços teria de esperar um pouco mais, uns três segundos no caso de serviços simples até uns vinte segundos para casos mais complexos, enquanto se classificavam os seus contratos.

Lentidão de navegação

À medida que o *selfcare* foi sendo completado, notou-se que existiam alguns *webservices* do Tibco muito lentos. Se um cliente acedesse duas vezes seguidas à mesma funcionalidade, esperava sempre o mesmo tempo para receber a informação. Se acedesse a uma funcionalidade que visava os serviços contratados, ficava largos segundos à espera, enquanto o *selfcare* classificava os seus serviços.
Tornou-se óbvio que seria necessário implementar um mecanismo de *cache*, ou seja, um mecanismo de gravar a informação obtida dos *webservices* Tibco. Deste modo, um segundo acesso à mesma informação resultaria na entrega da resposta da *cache*, evitando a necessidade de ir novamente ao Tibco.

A primeira solução implementada foi gravar a informação em ficheiros em disco, solução que apesar de melhorar os tempos de acesso, mostrou-se pouco eficiente devido ao volume de informação guardado e os tempos de leitura de disco relativamente altos (0.3s de leitura por registo guardado).
O autor investigou soluções possíveis na Internet, encontrando boas referências para um produto chamado *redis* – ***RE****mote* ***Di****ctionary* [31], um sistema de acesso de leitura e escrita veloz que permitia guardar informação indexada por chave (*key-value store*). Os testes mostraram tempos médios de 0.014s de leitura por registo, que eram claramente melhores do que a solução baseada em ficheiros.
Um pormenor interessante deste serviço é que permitia associar tempos de validade para os dados inseridos. Passado esse tempo de validade, os registos eram automaticamente removidos pelo *redis*. Com esta funcionalidade, não seria necessário tratar da limpeza de dados.

O autor implementou uma solução genérica de *cache* de métodos, simples de configurar e completamente transparente para quem invocava os métodos visados.

A configuração passava por indicar:

- O nome do método.
- O tempo de validade dos dados.

A solução garantia:

- Guardar apenas respostas com resultados de sucesso.
    - Isto garantia que um erro temporário inesperado, tal como indisponibilidade do sistema de faturação, não negava acesso à informação durante o tempo de validade dos dados.
- Devolver sempre a mesma informação para os mesmos parâmetros, durante o tempo de validade
    - Tal foi garantido usando uma função hash[v] para gerar um identificador único baseado nos parâmetros de entrada.
- Que os dados de um cliente não seriam entregues a pedidos de outro cliente.
    - O identificador do cliente fazia parte da chave de acesso à informação.

Um exemplo prático da configuração no código com o resultado no *redis* pode ser visto nos blocos

v Algoritmo que transforma uma sequencia variável de texto num texto de tamanho constante

apresentados a seguir.

```
#@ list of methods with cache and the TTL (sec's) for cache
my @_methods = (

      'classify_contract_by_msisdn'      => 3600,
      'classify_account'                 => 3600,
      'get_billing_details'              => 240,
      'get_customer_details'             => 600,
      #
```

Código 1: Exemplo de configuração de cache

Alguns métodos foram configurados com tempos de validade muito baixos, para obrigar a frequente renovação da informação, como é o caso do “get_billing_details”, com um tempo de validade de quatro minutos. Por outro lado, métodos que devolvessem informação mais estável, podiam ter tempos de validade superiores, como se pode ver no “classify_account” com sessenta minutos de validade.

```
#
#Sistema:IdCliente:api:método:hash
#
"limbowsc:177334:api:classify_contract_by_msisdn:pQvQRcrJUAngkTTcFSuRfw"
"limbowsc:74229:api:classify_contract_by_msisdn:4UC8xvOeORmT5pwiNrNLng"
"limbowsc:322753:api:get_customer_details:+eqQZ5vJZTkU4BU5RYa45w"
```

Código 2: Exemplo de chaves da cache redis

Não é possível apresentar um exemplo dos dados guardados no *redis*, já que os mesmos são comprimidos, ficando em formato binário.

Os ganhos em termos de velocidade de acesso ao selfcare foi logo evidente, passando de minutos por página para meros segundos.

Qualidade de código

A equipa de desenvolvimento era composta principalmente por programadores juniores, já que a disponibilidade do mercado de programadores seniores em Perl tem sido sempre limitado. Uma das estratégias que a equipa do autor tem seguido é recrutar programadores de PHP, e, através de formação interna, transformá-los em programadores Perl. Este método tem funcionado com algum sucesso, no entanto necessita de algum tempo para que os recursos fiquem realmente eficientes e autónomos.
Devido aos prazos muito agressivos do projeto do *selfcare*, não foi possível formar uma equipa adequada a tempo do início dos desenvolvimentos. O resultado foi que muito do código inicial entregue, apesar de

funcionar, não tinha a qualidade desejada pelos líderes técnicos.

A estratégia sugerida pelo autor, e adotada pelos restantes elementos, foi de fazer revisão de código diário de todos os programadores juniores. Com o correr do tempo, esta estratégia evoluiu para um formato em que os seniores recebiam o código dos juniores, reescreviam / reestruturavam de acordo com a sua experiência e sabedoria e devolviam para que os juniores pudessem aprender com os erros. Apesar de ser extenuante, a estratégia resultou em pleno, já que passados poucos meses os juniores já estavam a entregar trabalho de altíssima qualidade. O resultado foi um sistema que desde que foi lançado, tem se mostrado extremamente estável, fiável e com um número de incidentes semanais negligenciável.

3.3.5 Conclusão

O autor nunca participou num projeto tão exigente e recompensador quanto o do *selfcare* convergente.
Tem a convicção que a sua participação foi fundamental para o sucesso, tendo estado envolvido em todas as suas vertentes, desde do desenho à implementação. Foi preponderante para a introdução de novas metodologias, como o *Scrum* e *test-driven-development* e tecnologias, tais como *redis* e *git*, que aumentaram a eficiência de trabalho de toda a equipa. Em termos de sistemas, desenvolveu um gerador de código que automatizou a geração de código de testes, de *webservices* e de documentação, e um sistema de formação modular para ajudar novos programadores a aprenderem as regras básicas da equipa de trabalho.

Implementou, com a ajuda de programadores juniores, um sistema de *login* único usado por vários sítios[vi] tais como:

- o *selfcare* convergente da Optimus – areadeclientes.optimus.pt / areadecliente2.nos.pt.
- o site Optimus – www.optimus.pt.
- o *selfcare* do continente mobile – www.continentemobile.pt/.
- o *selfcare* de produto wholesale Optimus – wholesaleportal.optimus.pt / wholesaleportal.nos.pt.
- as apps móveis da Optimus e kanguru.
- os sites de parceiros de *calling rings* e alertas.

Ajudou a planear e implementar o *selfcare* convergente, que disponibilizou cerca de duzentas funcionalidades aos clientes, interagindo para tal com nove sistemas internos, usando cento e quarenta e dois *webservices* Tibco. O sistema tem-se mostrado tolerante a falhas, e extremamente fiável, sendo o número de incidentes semanais tão baixo, que o autor acumula a função de suporte de terceira linha sem que isso lhe cause transtornos.

vi Alguns destes sítios não funcionam devido à fusão da Optimus com a ZON

4 Conclusão

No presente relatório foram enunciados, não de forma exaustiva mas representativa, as atividades profissional do autor desde que terminou a Licenciatura em Engenharia Informática em 1996.

O autor foi ganhando conhecimentos lentamente e de modo exaustivo no ramo dos serviços de Internet. Passou por diversas funções, desde as mais básicas, como suporte a clientes por telefone e email, passando pelo mundo tecnicamente exigente de administração de sistemas, passando por experiências de interação humana através de gestão de projetos e de equipas.

Atualmente efetua um misto de funções, incluindo gestão de projetos, desenho e arquitetura de soluções para serviços de Internet, gestão de equipas de desenvolvimento e de suporte de terceira linha. Em situações extremas de emergência, é frequente ser envolvido na análise e resolução de incidentes.

O percurso profissional passou por uma equipa pequena, que por sucessivas compras / reorganizações / fusões foi aumentando de tamanho e de responsabilidade, interagindo cada vez mais com mais intervenientes, muitos com parcos conhecimentos técnicos, obrigando também o desenvolvimento de capacidade de comunicação, de discussão e persuasão.

A carreira que foi sendo construída foi possível devido aos fortes alicerces fornecidos pela Licenciatura de Engenharia Informática. Na opinião do autor, a licenciatura não só valeu pelos conhecimentos basilares fornecidos em disciplinas como as várias de Programação, Estrutura e Funcionamento de Computadores, Redes de Computadores e Análise de Projetos, mas principalmente pela capacidade de trabalho e de reação a situações inesperadas que incutiu nos seus alunos.

Uma questão frequente é de como foi possível não mudar de empresa, especialmente nas alturas mais difíceis, tais como a falência da KPNQwest Internacional. A resposta é bastante simples: a viagem tem sido muito boa, cheia de colegas fabulosos, brilhantes e amigos, com desafios algumas vezes aparentemente impossíveis, em tecnologias dispares. No final do dia tem sido possível olhar para a Obra com satisfação de dever cumprido, e no início do próximo dia, não tem sido difícil sair de casa para ir trabalhar. Enquanto a viagem continuar a ter estas características, será difícil pensar em mudar!

O autor considera que a formação académica tem sido complementada pelo conhecimento adquirido ao longo da progressão profissional, de tal forma que considera que cumpre os requisitos necessários à obtenção do grau de Mestre em Engenharia Informática, de acordo com o estabelecido no ponto 1.b) do Despacho n.º 20/2010 (Obtenção do Grau de Mestre por Licenciados "Pré-Bolonha").

5 Bibliografia

[1] http://pt.wikipedia.org/wiki/C%C3%B3digo_aberto

[2] http://www.itil.org. ITIL, 2007-10-01

[3] Schwaber,Ken; Beedle,Mike. Agile Software Development with Scrum, 2001

[4] http://www.cisco.com/c/en/us/td/docs/ios/solutions_docs/ip_multicast/White_papers/mcst_ovr.html. IP Multicast Technology Overview, 2000-09

[5] https://www.java.com/en/. Java, 2014

[6] https://sites.google.com/site/historiadainternetpt/o-puug. "A montagem do serviço do PUUG 1991 / 1995", 2014-08

[7] http://en.wikipedia.org/wiki/Client%E2%80%93server_model. "Client–server model", 2014/08

[8] http://www.unix.org/. "The UNIX® System", 1997-03-16

[9] https://www.perl.org/. The Perl Programming Language, 1996-12

[10] http://www.cpan.org/. Comprehensive Perl Archive Network, 1998-01

[11] Conway, Damien. Object Oriented Perl: A Comprehensive Guide to Concepts and Programming Techniques, 2000-01

[12] Castagnetto, Jesus M; Rawat, Harish; Veliath, Deepak T; "Professional PHP Programming", 1999-12-15

[13] http://en.wikipedia.org/wiki/Year_2000_problem. Year 2000 problem, 2014-08

[14] Conway,Damian. PBP - Perl Best Practices - Standards and Styles for Developing Maintainable Code, 2005-07

[15] https://www.gnu.org/software/radius/. Introduction to Radius, 2005-04

[16] http://freeradius.org/ FreeRadius, 2005-04

[17] Kernighan, Brian W; Ritchie, Dennis M. The C Programming Language, 1994

[18] http://www.net-snmp.org/. Net-SNMP, 2011-03

[19] Comer, Douglas. Internetworking With TCP/IP, 2000

[20] http://www.w3schools.com/webservices/ws_soap_intro.asp. SOAP Introduction, 2008-01

[21] http://www.tibco.com/. TIBCO, 2008

[22] http://log.pt/projeto/web-self-care/

[23] http://www.outsystems.com/

[24] http://www.wedotechnologies.com

[25] http://twiki.org/. TWiki - the Open Source Enterprise Wiki and Web Application Platform, 2007

[26] http://www.nongnu.org/cvs/. CVS - Concurrent Versions System, 2004-04

[27] http://www.w3schools.com/webservices/. Web Services Tutorial, 2008

[28] http://c2.com/cgi/wiki?TestDrivenDevelopment. Test Driven Development, 2010

[29] http://www.yaml.org/. "%YAML 1.2", 2010

[30] http://git-scm.com/. "git - --distributed-is-the-new-centralized ", 2010

[31] http://redis.io/. Redis, 2010