Mestrado em Engenharia Informática
Estágio
Relatório Final

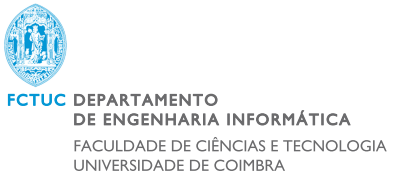
Telecom Knowledge System - Integração Ontológica

Luís Patrício Pereira de Alte da Veiga
lpveiga@student.dei.uc.pt

Orientador do Departamento de Engenharia Informática:
Prof. Dr. Alexandre Miguel Pinto

Orientador da Empresa Maisis - Information Systems, Lda:
Eng. Miguel Grade

1 de Julho de 2013



Agradecimentos

A realização deste projeto, cuja duração foi de dez meses, não teria sido possível sem a contribuição de determinadas pessoas e entidades.

Os meus primeiros agradecimentos são dirigidos às empresas que fizeram a proposta de estágio à Universidade de Coimbra, Inova-Ria PT e Maisis, tornando assim possível a concretização deste estagio. Ainda dirigindo-me a estas entidades, agradeço também pelos recursos fornecidos, assim como a disponibilidade mostrada ao longo do projeto para auxiliar em qualquer problema que surgisse.

De seguida, agradeço aos professores orientadores do projeto, Prof. Dr. Alexandre Miguel Pinto e Engº Miguel Grade, cujo acompanhamento semanal foi muito importante para que o caminho traçado não sofresse quaisquer desvios e atingisse o sucesso final.

Ainda no âmbito académico, dirijo os meus agradecimentos aos meus colegas de curso, que estiveram sempre disponíveis para esclarecer duvidas pontuais e ajudar na preparação das duas defesas, intermédia e final.

A terminar, mas não menos importante, agradeço à família, amigos e companheira, cujo apoio mostrado ao longo dos últimos dez meses foi muito importante para ultrapassar as barreiras mais difíceis, assim como nos festejos das pequenas vitorias conquistadas ao longo do projeto.

Resumo

Palavras-chave: Base de Dados, Mapeamento, Ontologia, *Triple Store*.

A Maisis é uma empresa inserida no mercado das telecomunicações, cujo grande produto, de nome OObian, é uma plataforma que permite consultar informação sob a forma de grafos e efetuar pesquisas personalizadas. A PT Inovação, também uma empresa de telecomunicações, pretende que os seus utilizadores consultem, cruzem e explorem informação de um modo conceptual, pelo que recorreu à Maisis para dar resposta a esta necessidade.

O projeto realizado consistiu no desenvolvimento de um novo módulo, para a plataforma OObian, que respondesse ao pedido da PT Inovação. A abordagem seguida foi a de fazer um mapeamento dos dados presentes numa bases de dados da PT Inovação para o servidor OObian da Maisis, que assenta sobre uma *triple store*. Devido ao diferente tipo de armazenamento de informação entre empresas, foi criada uma ontologia para ser possível guardar esses dados no servidor OObian.

O resultado final do mapeamento foi validado pelo cliente numa demonstração visual da plataforma, cuja apreciação final foi de total satisfação. Ainda durante a demonstração, assim como nos testes realizados, também o requisito de performance foi cumprido. O cliente pretendia obter respostas com um intervalo de tempo de processamento entre cinco e dez segundos, no pior caso, sendo que as pesquisas mais complexas, com um número de relações na casa das centenas, tiveram uma duração média de $180,63$ milissegundos.

Com estes resultados, a PT Inovação pretende reduzir os custos com os seus operadores de *call-center*, e melhorar a capacidade dos seus gestores em analisar e tomar decisões.

Além deste módulo, foi ainda desenvolvida uma metodologia genérica de trabalho que pretende agilizar semelhantes processos futuros da Maisis. Portanto, este estágio veio acrescentar valor tanto à PT Inovação como à Maisis.

Conteúdo

API	*Application Programming Interface*
BSS	*Business Support Systems*
D2R	*Database to RDF*
D2RQ	*Database to RDF Querying*
DBMS	*DataBase Management System*
ETL	*Extract, Transform, Load*
eTOM	*enhanced Telecom Operations Map*
FURPS	*Functionality, Usability, Reliability, Performance, Supportability*
GTD	*Getting Things Done*
HTML	*HyperText Markup Language*
I&D	*Innovation and Development*
ISO/IEC	*International Organization for Standardization / International Electrotechnical Commission*
JDBC	*Java DataBase Connectivity*
KDIS	*Knowledge Data Integration System*
LDIF	*Linked Data Integration Framework*
OBDA	*Ontology-Based Data Access*
OSS	*Operational Support Systems*
OWL	*Web Ontology Language*
OWL-DL	*Web Ontology Language Description Logic*
PT	Portugal Telecom
R2RML	*RDB To RDF Mapping Language*
RAM	*Random Access Memory*
RDB	*Relational Data Base*
RDF	*Resource Description Framework*
RDFS	*Resource Description Framework Schema*
SPARQL	*SPARQL Protocol and RDF Query Language*
SID	*Shared Information and Data*
SQL	*Structured Query Language*
SSH	*Secure Shell*
TAM	*Telecom Application Map*
TELCO	*Telecommunications Company*
TM Forum	*TeleManagement Forum*
TURTLE	*Terse RDF Triple Language*
UC	Universidade de Coimbra
UML	*Unified Modeling Language*
URI	*Uniform Resource Identifier*
VPN	*Virtual Private Network*
W3C	*World Wide Web Consortium*
XML	*eXtensible Markup Language*

Tabela 1: Acrónimos

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

No primeiro capítulo do documento, o leitor poderá contextualizar-se com o problema e, de seguida, ficar a perceber qual o verdadeiro problema que nos propomos a resolver, a motivação para o resolver, e quais os objetivos e características globais que se pretendem alcançar. São ainda apresentadas as metodologias seguidas na realização do projeto, a respetiva análise de riscos, e o capítulo termina com uma apresentação muito geral dos resultados finais.

1.1 Contextualização e Identificação do Problema

No âmbito da conclusão de um Mestrado em Engenharia Informática, foi realizado um estágio curricular em contexto empresarial, mais concretamente, na empresa Maisis[1], em Aveiro.

A Maisis foi fundada em 1994, e a sua principal atividade é no mercado das telecomunicações e nos sistemas de interpretação e criação automática de conhecimento de organizações. Tem como missão, desenvolver soluções tecnológicas, destinadas a servir de forma diferenciada cada cliente, e a sua visão é a de ser uma empresa de vanguarda, capaz de criar soluções inovadoras e de referência.

Com a realização deste estágio, a Maisis pretendeu responder ao pedido de um cliente, a PT Inovação[2], para a concretização de um projeto.

A PT Inovação tem utilizadores, que se podem dividir em gestores (de projeto, de rede, de provisão e de manutenção) e operadores de *call-center*,

[1]Maisis - http://www.maisis.pt/

[2]PT Inovação - http://www.ptinovacao.pt/

que pretendem aceder a informação sobre o cadastro da rede. A informação que pode ser consultada no cadastro da rede refere-se aos clientes da PT Inovação, aos serviços que a própria PT Inovação tem, tanto os que fornece aos seus clientes como os que necessita para o funcionamento interno da empresa, e aos recursos que os serviços precisam para um correto funcionamento.

Os clientes podem ser residenciais ou empresariais. Os clientes residenciais são os típicos utilizadores de serviços em sua casa ou no seu dia-a-dia, como é o caso dos serviços *triple play*[3] e telemóvel. Alguns dos recursos necessários para que estes serviços funcionem podem ir de *routers* e *boxes* de televisão a telemóveis e controlos remotos. Relativamente aos clientes empresariais, os serviços que usufruem vão mais ao encontro de redes internas de Internet ou de trabalho, ou de programas para gestão da própria empresa. Para tais serviços, são necessários equipamentos para fazer de servidores, *routers*, ou até mesmo *switches*, e de estabelecer as devidas ligações entre todos os equipamentos.

Neste momento, toda esta informação encontra-se armazenada numa base de dados relacional, o que significa que os registos encontram-se presentes em tabelas que estão relacionadas entre si. Esta base de dados segue um modelo desenvolvido e apresentado pelo TeleManagement Forum, mais conhecido como TM Forum[4]. TM Forum é uma associação industrial, sem fins lucrativos, focada em fornecer agilidade e inovação na prestação de serviços, que conta com 4 *frameworks* distintas [15]: Business Process Framework (eTOM)[5], Information Framework (SID)[6], Application Framework (TAM)[7] e Integration Framework[8]. O modelo de dados da PT Inovação segue o modelo SID, Shared Information and Data.

O modelo SID pretende ser um modelo de dados genérico que fornece um conjunto de vocabulário comum para toda a informação que diz respeito a processos de negócio em telecomunicações. Foi desenvolvido com o objetivo de permitir que todas as partes envolventes num negócio, desde uma empresa a parceiros de negócio, e até mesmo prestadores de serviços, usem os mesmos termos para descrever os mesmos objetos e relações, facilitando deste modo todo o processo de negócio.

Uma vez que a PT Inovação pretende que os seus utilizadores consultem,

[3]Triple Play = Internet + Telefone + Televisão
[4]TM Forum - http://www.tmforum.org/
[5]eTOM - http://www.tmforum.org/BusinessProcessFramework/1647/home.html
[6]SID - http://www.tmforum.org/InformationFramework/1684/Home.html
[7]TAM - http://www.tmforum.org/ApplicationFramework/2322/Home.html
[8]Integration Framework - http://www.tmforum.org/IntegrationFramework/4866/Home.html

cruzem e explorem a informação de um modo mais conceptual, como apresentado na figura 1.1, tiveram de recorrer à Maisis, parceiro seu que tem *"know-how"* em áreas associadas, de modo a verem o seu problema resolvido.

Para facilitar a visualização da informação desta maneira, a Maisis desenvolveu, e gere, uma plataforma de nome OObian[9], que opera sobre uma *triple store*[10] e tem um interface próprio que representa a informação sob a forma de grafos, pelo que a solução passa por fazer uso desta plataforma para armazenamento, gestão e consulta da informação.

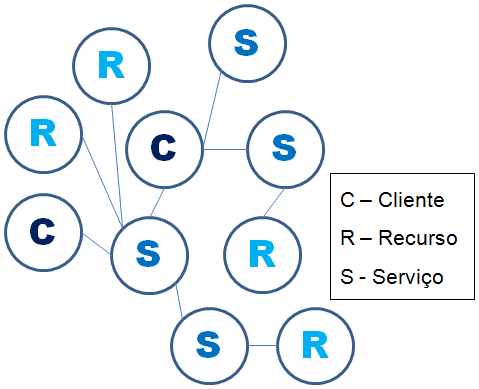


Figura 1.1: Vista Conceptual

Uma vez que a plataforma OObian opera sobre uma *triple store*, significa que se tem de fazer uso de ontologias para representar os conceitos e os relacionamentos entre estes. O recurso a ontologias, para além de facilitar a visualização da informação de um modo mais conceptual, sob a forma de um grafo, permite que o processo de construção de resposta a perguntas evite o cruzamento de tabelas inteiras, como é o caso com bases de dados relacionais.

Web Semântica

A Web Semântica é uma extensão da Web atual que se foca na semântica dos dados e pretende que os computadores sejam capazes de interpretar símbolos, apelidando-os de agentes inteligentes [23]. Sendo a semântica o significado dos dados, o objetivo de um agente inteligente é o de interpretar símbolos (sintaxe), de modo a fazer corresponder esses símbolos a objetos, criar um contexto, e conseguir saber que ação executar.

Uma necessidade destes agentes é a de saber os objetos que existem e quais as ligações entre si. Para tal, foram adotadas conceptualizações comuns denominadas ontologias. De um modo geral, uma ontologia é composta por classes/entidades, relações entre estas, e as respetivas propriedades, sendo que dois dos formalismos mais conhecidos e usados para representar uma ontologia são RDF (Resource Description Framework) e OWL (Web Ontology

[9] OObian - http://www.oobian.com/
[10] *Triple Store* - http://en.wikipedia.org/wiki/Triplestore

Language), cuja representação gráfica pode ser vista como um grafo, nós e arcos a ligar os nós.

Ambos os formalismos fornecem uma linguagem de representação baseada em triplos, sujeito-predicado-objeto. Um triplo significa que um nó (sujeito), tem uma relação com outro nó (predicado), com um determinado valor (objeto). A diferença entre estes formalismos, é que RDF tem o propósito de criar taxonomias e propriedades, enquanto OWL serve para aumentar a expressividade dessas taxonomias, como por exemplo, acrescentando equivalências e restrições.

Por fim, com a evolução da Web Semântica, surgiu a necessidade de haver repositórios para armazenar as ontologias. Estes repositórios são denominados de *triple stores*.

Stakeholders

Existem diferentes stakeholders relacionados com este estágio, sendo que os dois mais importantes já foram apresentados, as empresas Maisis e PT Inovação.

Noutro sentido, o estagiário Luís Veiga é também um stakeholder, pois realizou este estágio com o propósito de terminar o seu percurso curricular. Por fim, um último stakeholder a mencionar é o professor orientador por parte da Universidade de Coimbra, Prof. Dr. Alexandre Miguel Pinto, cujo seu interesse foi o de estar ligado, e apoiar, um estágio na área da Web Semântica que teve sucesso.

1.2 Objetivos

Os objetivos a que nos propusemos cumprir podem ser divididos em dois grupos: objetivos relacionados com o produto final e objetivos relacionados com trabalho futuro.

Este projeto consistiu na construção de uma camada de abstração sobre uma base de dados tradicional, completamente construída e gerida pela PT Inovação, que permite efetuar uma extração, transformação, alinhamento e carregamento de conceitos vitais para o suporte às pesquisas dos utilizadores-tipo, através de operações de mapeamento configuráveis a partir da plataforma OObian. Uma vez que esta plataforma opera sobre uma *triple store*, foi igualmente necessário construir uma ontologia para que a referida camada consiga mapear os dados da base de dados para a ontologia. Por fim, após

terminada a tarefa de mapeamento, os dados são enviados para a plataforma OObian, ficando assim disponíveis para consulta.

Quanto à plataforma OObian, antes do envio dos dados, tem de se carregar a ontologia para a plataforma de modo a que a *triple store* tome conhecimento do *schema ontológico*. É também na própria plataforma que se configura se cada uma das propriedades é filtravél e/ou pesquisável por texto livre.

De referir ainda que a base de dados continuará a ser utilizada para inserção e atualização de dados, sendo que o suporte às pesquisas efetuadas será dado na plataforma OObian, através dos clientes de pesquisa e navegação existentes.

Relativamente ao segundo conjunto de objetivos, aquando do desenvolvimento, a Maisis pretendeu que fossem identificadas oportunidades de melhoria nas suas ferramentas, KDIS e OObian, pois permitirá, num futuro próximo, dotar estas com mais funcionalidades. Estas oportunidades podem ser consultadas no capítulo 9, Trabalho Futuro.

Uma vez que a Maisis poderá vir a ter outros projetos muito semelhantes com o projeto desenvolvido, foi também desenvolvida uma metodologia genérica para agilizar os processos desses futuros projetos. O documento que apresenta essa metodologia pode ser consultado no apêndice D do presente documento.

1.3 Motivação

A motivação para a realização deste estágio centrou-se nos ganhos que a PT Inovação pode retirar do produto final. Tendo um produto que fornece um meio mais eficaz e eficiente, que a via atual, para aceder à informação, a produtividade dos diferentes gestores pode aumentar, pois podem obter informação mais rapidamente e fazer pesquisas que em bases de dados relacionais não são possíveis, pois têm menos expressividade que as ontologias.

Quanto ao aspeto financeiro, através dos cálculos efetuados e apresentados no apêndice B, é possível confirmar que a PT Inovação pode reduzir os seus custos com operadores de *call-center*1 em cerca de 3.340.800€por ano, o que demonstra que um segundo numa chamada tem um grande valor financeiro para a PT Inovação. Relativamente aos cálculos efetuados, é importante salientar que estes foram validados, tanto pela Maisis, como pela PT Inovação.

Existem ainda outros fatores que, não sendo quantificáveis, têm impacto nos lucros da empresa: por um lado, a satisfação dos clientes perante a

resolução de um problema, por outro, a qualidade do atendimento recebido.

1.4 Características Globais

A construção da camada referida em 1.2 envolve determinados aspetos que têm de estar sempre presentes aquando da sua elaboração:

- Escalabilidade - capacidade do sistema em processar milhões de triplos;
- Mutabilidade dos dados - frequência com que os dados são atualizados na base de dados;
- Performance - tempo de resposta às pesquisas dos utilizadores.
- Segurança - mecanismos de segurança para manter integridade da plataforma e dos dados;

Estima-se que o número de utilizadores que irá usar o produto final estará na ordem dos milhares. Contudo, esses utilizadores não irão usar o serviço em simultâneo, pelo que a escalabilidade a este nível não é motivo de preocupação. Porém, a escalabilidade referente à dimensão do número de triplos inseridos no grafo, na ordem dos milhões, pode levantar problemas, uma vez que a própria arquitetura dos grafos (dependências entre nós) pode limitar o particionamento do mesmo e, consequentemente, o processamento de informação levar mais tempo do que o pretendido [24] [13]. O cliente pretende que o tempo de cada resposta, na pior das hipóteses, dure entre cinco e dez segundos. Portanto, uma das características pretendidas é a de ser possível processar uma grande quantidade de informação, na ordem dos milhões de triplos, num curto período de tempo.

Pela informação fornecida pelo cliente, existem atualizações praticamente todos os dias na base de dados. Posto isto, a PT Inovação afirmou que a informação a apresentar pode pertencer a um *"snapshot"* da base de dados, sendo que eles ficam responsáveis por atualizar a plataforma OObian de acordo com as suas necessidades. Ligado a este ponto está outra característica, performance, que é o grande requisito de alto nível que a PT Inovação coloca. Da perspetiva de negócio da PT Inovação, é preferível uma maior eficiência no acesso aos dados relativamente à atualização em tempo real dos mesmos, isto é, é aceitável uma desatualização reduzida dos dados se tal for necessário para garantir a eficiência esperada do serviço.

Para o projeto em questão, a área de segurança não tem um grande peso no desenvolvimento, pois os mecanismos para manter a integridade dos dados e o sistema disponível já se encontram implementados e estão ao cargo

do servidor OObian da Maisis. Aliando o facto do produto final não fazer uso da Internet para o seu correto funcionamento, a PT Inovação mostrou-se satisfeita com as condições de desenvolvimento apresentadas, identificando apenas um requisito de segurança, restrição de acesso. O cliente gostaria que cada utilizador apenas aceda à informação que lhe diz respeito.

1.5 Planeamento Inicial

A presente secção pretende apresentar as metodologias que foram utilizadas, tanto de trabalho, como de desenvolvimento e de mapeamento, assim como uma compilação da análise de riscos de todo o projeto.

1.5.1 Metodologia de Trabalho

De modo a atingir os objetivos propostos, foi adotada a metodologia genérica de trabalho "Getting Things Done"(GTD) [1] como método de organização, documentação e controlo das atividades desenvolvidas.

De referir que o acompanhamento do projeto foi desde início constante e se revelou muito importante para atingir o sucesso. Este acompanhamento foi feito através de reuniões cujo objetivo era o estagiário Luís Veiga apresentar todo o trabalho desenvolvido, quer documentação quer desenvolvimento do produto final, sendo que as reuniões se podem dividir em três tipos:

- Reuniões com o professor orientador da UC (semanais no primeiro semestre e quinzenais no segundo);
- Reuniões com a empresa Maisis e o cliente PT Inovação (mensais);
- Reuniões com o professor orientador da UC e a empresa Maisis (mensais).

De referir ainda que houve algumas reuniões esporádicas entre o estagiário Luís Veiga e a PT Inovação para que o cliente prestasse auxílio na compreensão do modelo de dados de onde se extrai informação.

1.5.2 Metodologia de Desenvolvimento

A abordagem que se pretendia seguir durante a fase de desenvolvimento era uma que nos permitisse dividir todos os requisitos funcionais em pequenos

grupos, e, para cada um desses grupos, implementar, testar, apresentar ao cliente, receber o respetivo *feedback*, fazer as alterações necessárias e voltar a testar. Para tal, o processo de desenvolvimento seguido foi um processo de desenvolvimento ágil [18]. A figura 1.2 demonstra a nossa abordagem para cada um dos grupos de requisitos.

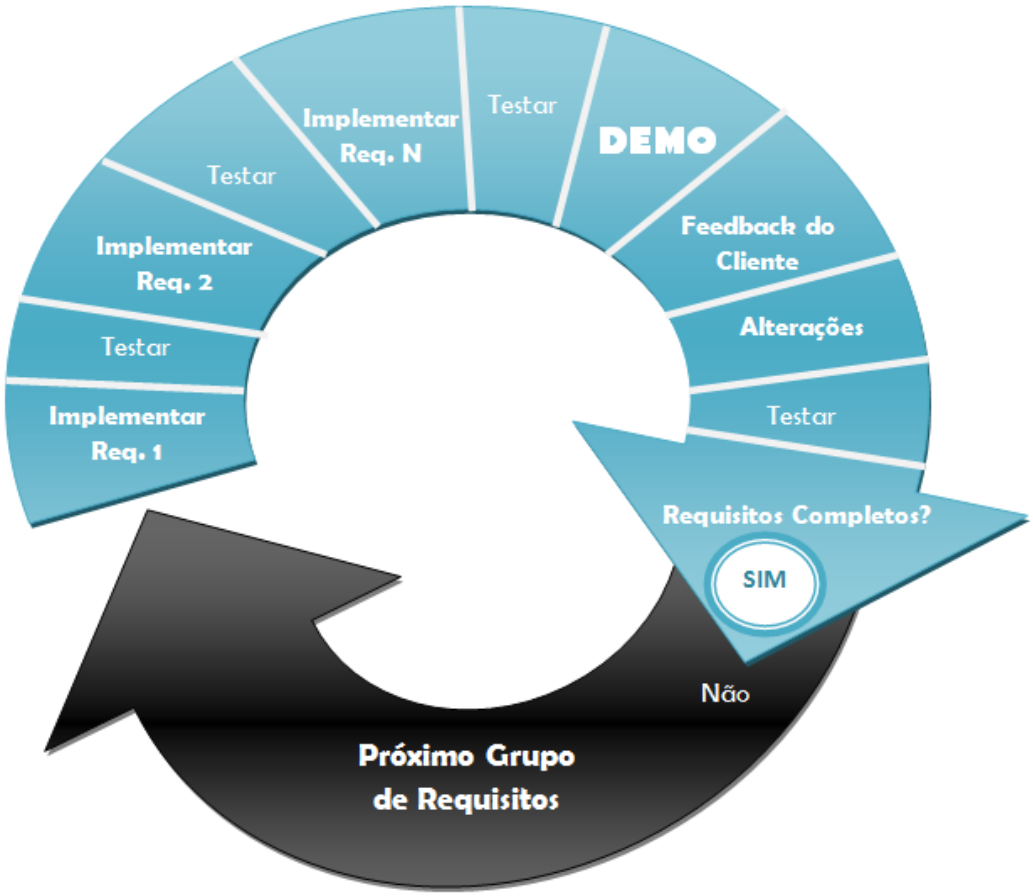


Figura 1.2: Metodologia Ágil de Desenvolvimento.

Após se fazer o levantamento dos requisitos do projeto, dividiram-se estes em pequenos grupos, sendo que o critério seguido foi o de agrupar os requisitos consoante a classe a que pertencessem: cliente, serviço ou recurso.

Para cada um desses grupos, a figura 1.2 ajuda a perceber o método de desenvolvimento. À medida que se implementava um requisito do grupo escolhido, fomos testando se a informação apresentada e relacionada era a pretendida, sendo que, terminada a implementação do grupo de requisitos, preparou-se uma demonstração para apresentar à PT Inovação.

Nestas demonstrações, o cliente forneceu feedback quanto às propriedades a apresentar para cada classe (nome, tipo, descrição, estado operacional, entre outras) e se deveriam surgir novas relações entre classes, o que significava um acréscimo na informação a disponibilizar e, consequentemente, um refinamento da ontologia. De seguida, procedemos às alterações identificadas pelo cliente e voltámos a efetuar testes. Terminada esta tarefa, ou se procedia à implementação do grupo de requisitos seguinte, ou, no caso em que era o último grupo, dava-se por encerrada a fase de desenvolvimento.

1.5.3 Metodologia de Mapeamento Relacional-Ontológico

A tarefa de mapeamento dos dados da base de dados para uma ontologia também pode ser considerada como uma metodologia, pois consiste num conjunto de passos que devem ser seguidos. A figura 1.3 demonstra o conjunto de passos necessários a realizar para completar esta tarefa.

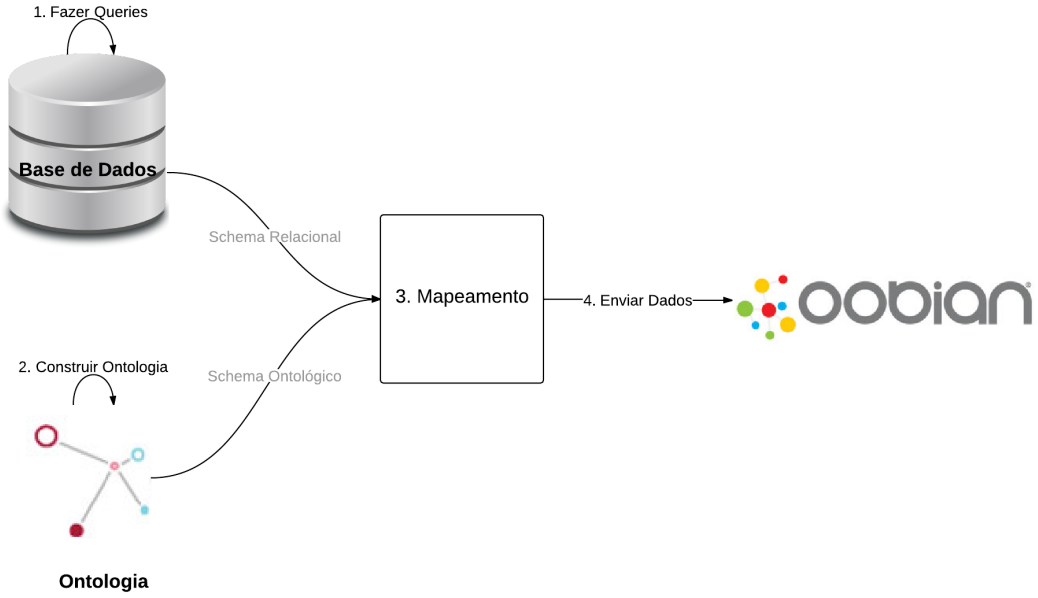


Figura 1.3: Metodologia de Mapeamento.

Inicialmente, efetuam-se *queries* à base de dados para se ficar a conhecer os dados presentes nos diferentes campos e de modo a ser possível construir a ontologia, pois o *schema* ontológico é baseado no *schema* relacional da base de dados. De seguida, pegando nestes dois *schemas*, elabora-se o ficheiro que contém o mapeamento, terminando esta tarefa com o envio do resultado do mapeamento para a plataforma OObian.

1.5.4 Análise de Riscos

Análise de riscos é uma das tarefas mais importantes para um gestor de projeto, pois permite antecipar riscos que podem afetar o projeto, quer a nível temporal quer a nível de qualidade, e tomar decisões para evitar ou minimizar estes riscos.

Em [25], Ian Sommerville apresenta uma análise de riscos que tem em conta os seguintes aspetos para cada risco: categoria que afeta, tipo, probabilidade de ocorrer, efeito no projeto e estratégia para evitar a sua ocorrência ou minimizar os danos caso ocorra. A estes aspetos foram adicionados outros dois, altura de ocorrência e abordagem, de modo a informar se o risco ocorreu, se surgiu um novo risco a partir desse, e qual a abordagem adotada aquando da sua ocorrência. Seguindo esta análise, na tabela 1.1 é apresentada a compilação da análise de riscos que foi feita ao longo do projeto, cujo intervalo de atualização foi de três semanas.

Os riscos identificados na tabela 1.1 são maioritariamente referentes à tarefa de desenvolvimento, uma vez que foi a tarefa que mais tempo ocupou durante a realização do projeto. Devido ao projeto em questão ser de Engenharia, também foram identificados riscos relativos à gestão do projeto, mas em menor quantidade. É do nosso conhecimento que problemas de saúde ou pessoais de um dos membros envolventes no projeto podem ser considerados na análise de riscos, porém, decidimos apenas focarmo-nos na tarefa de desenvolvimento.

A elaboração de planos de contingência para a mitigação da eventual ocorrência dos riscos identificados serviu como ferramenta de minimização do impacto dos referidos riscos na qualidade da solução desenvolvida.

Risco	Categoria	Descrição	Tipo	Probabilidade	Efeito	Estratégia	Altura de Ocorrência
Falta de Informações do Cliente	Projeto e Produto	Necessidade de ter informação para tomar decisões e desenvolver o produto.	Pessoas	Muito Alta	Existência de tarefas pendentes.	Obter o máximo de informação possível em cada reunião.	Primeira Quinzena de Março - levou à identificação do risco **Alteração de Planeamento**.
Dificuldades no Acesso à Base de Dados	Projeto e Produto	Impossibilidade de acesso para obtenção de dados.	Tecnologia	Moderada	Não se tem dados para fazer o mapeamento e assim apresentar aos utilizadores.	Guardar dados localmente.	Última Semana de Abril - levou à identificação do risco **Carregamento Total**.
Limitações de Ferramentas	Produto	Ferramentas terem um mapeamento de dados limitado ou não suportarem pesquisas complexas.	Tecnologia	Moderada	Não ser possível responder a todas as perguntas.	Efetuar um desenvolvimento mais complexo caso essa abordagem responda ao pedido do cliente. Noutro sentido, dotar ferramentas com as funcionalidades necessárias.	Implementação de Requisitos - atributos percentagem de utilização e data de atualização.
Falta de Experiência	Projeto e Produto	Programador ter pouca, ou nenhuma, experiência com as ferramentas a utilizar.	Pessoas	Baixa	Atraso na tarefa de desenvolvimento ou não aproveitamento de todas as funcionalidades disponíveis.	Fazer diversos tutoriais, testes e ler a documentação disponível para uma correta ambientação e aquisição de conhecimentos.	Dia 15 de Fevereiro.
Planeamento Demasiado Desajustado da Realidade	Projeto	Estimativas de duração de tarefas erradas.	Estimativa	Baixa	Alteração da duração de realização das tarefas.	Ter planeamento sempre presente e uma forte organização.	Primeira Semana de Abril - implementação de requisitos *MUST* terminou mais cedo.
Alteração do Planeamento	Projeto	Plano de tarefas pode ter de ser alterado.	Estimativa	Baixa	Tarefas terem uma duração diferente do que a prevista ou terem de ser realizadas numa altura diferente da planeada.	Cumprir plano inicial ao máximo.	
Carregamento Total da Informação Demorado	Projeto e Produto	Carregamento de toda a informação da base de dados para a *triple store* levar mais de uma semana.	Tecnologia	Alta	Impossibilidade de fazer testes após implementação e com utilizadores finais.	Fazer carregamentos parciais para efeitos de teste. Após o primeiro carregamento total, apenas carregar informação mais recente (última semana ou último mês).	Meses de Abril e Maio - levou à identificação do risco **Utilizadores Finais**.
Ausência de Utilizadores Finais para Testes	Projeto e Produto	Impossibilidade de testar produto final com um número alargado e representativo de utilizadores.	Pessoas	Muito Alta	Cenário de testes longe do cenário real.	Abordar cliente atempadamente, final de Maio, para disponibilizar colaboradores para realização de testes.	Última quinzena de Junho.

Tabela 1.1: Análise de Riscos

1.6 Resultados

Os grandes objetivos traçados para este projeto, ao nível do desenvolvimento, foram dois: efetuar o correto mapeamento da informação da base de dados da PT Inovação para a *triple store* do servidor OObian, e ter uma performance cujo tempo de resposta dos pedidos não excedesse os dez segundos.

Após o desenvolvimento, foram feitos os respetivos testes, sendo que a tarefa de mapeamento foi validada numa demonstração ao cliente, cuja apreciação final foi bastante positiva, uma vez que o resultado desta tarefa foi ao encontro do que pretendia. Quanto à performance do produto final, os testes consistiram em ter quatro utilizadores a fazer aproximadamente trinta pesquisas cada um, recolhendo os tempos de processamento de todos os pedidos e, de seguida, fazendo uma análise estatística destes. No caso geral, foram feitas um total de 133 pesquisas, cujo tempo médio de processamento foi de $37,85ms$, com um desvio-padrão de $19,68ms$.

Portanto, os resultados obtidos permitem afirmar que os objetivos traçados foram cumpridos.

O resto deste documento segue a seguinte estrutura: no capítulo 2 apresentamos uma análise do estado da arte em ferramentas de mapeamento de informação de bases de dados para *triple stores*; no capítulo 3 fazemos um levantamento de requisitos, onde incluímos a tipificação dos utilizadores finais; no capítulo 4 detalhamos o plano de tarefas seguido na segunda fase do projeto; no capítulo 5 especificamos a arquitetura das componentes a desenvolver, no capítulo 6 explicamos o seu desenvolvimento, e no capítulo 7 apresentamos os resultados obtidos nos testes ao desenvolvimento efetuado; em modo de conclusão, no capítulo 8 fazemos a nossa análise crítica e retrospetiva de todo o projeto, e no capítulo 9 apresentamos as tarefas identificadas para se realizarem no futuro. Adicionalmente, temos a bibliografia e um conjunto de anexos a completar alguns dos capítulos descritos.

Capítulo 2

Estado da Arte

A realização de um projeto envolve sempre uma análise, entre outros aspetos importantes, do problema em si e das soluções existentes para resolver esse problema, pelo que este capítulo fornece uma vista geral do estado da arte na área de mapeamento de dados de bases de dados tradicionais para ontologias. Inicialmente, será apresentada uma solução existente para efetuar este mapeamento, seguindo-se uma análise de ferramentas que possam ser úteis na realização do projeto. Por fim, este capítulo termina com a apresentação, e respetiva justificação, da solução escolhida para realização do projeto.

2.1 Análise da Solução Existente

A área da Web Semântica é uma área em evolução [22] [26], razão pela qual ainda não existem muitas ferramentas e tecnologias que sirvam os diferentes propósitos de cada empresa. Para cumprir o objetivo de ter informação armazenada numa *triple store*, ao invés de numa base de dados, a empresa Maisis precisava de uma ferramenta que servisse as necessidades de mapear o *schema* relacional de bases de dados com o respetivo *schema* ontológico, instanciasse a respetiva ontologia com os dados vindos desta, e enviasse os resultados para a *triple store*, motivo pelo qual desenvolveu e surgiu o KDIS, *Knowledge Data Integration System*.

2.1.1 KDIS, Knowledge Data Integration System

O KDIS [2] é uma ferramenta de ETL, *Extract Transform Load*, em que o sistema destino é uma *triple store*. O KDIS extrai os dados de bases de dados relacionais, denominados sistemas externos, transforma-os via o módulo KServer, e por fim carrega-os na *triple store* destino. A figura 2.1

ilustra este processo de uma forma simples.

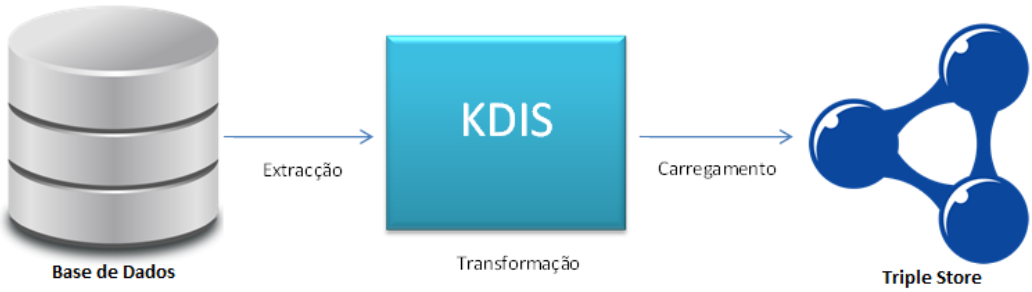


Figura 2.1: Mapeamento do KDIS.

O KServer é uma componente responsável por extrair e analisar documentos e entidades, de os indexar numa base de dados, de manter sobre eles uma *cache* e um sistema de pesquisa, portanto, é de antever que será o responsável por disponibilizar as ontologias ao KDIS, a fim deste poder proceder à integração de dados. Uma das vantagens do KServer é a de criar uma camada aceleradora para reduzir os tempos de resposta dos pedidos feitos à *triple store*.

Por seu lado, para se fazer uso do KDIS, o esqueleto da ontologia encontra-se num ficheiro OWL, sendo que o grau de expressividade pode ser RDF(S), OWL-Lite, OWL-DL (OWL - Description Logic) ou OWL-Full. Este esqueleto contém as classes necessárias, as relações entre classes, e os respetivos atributos, que servirão de base para o modelo ontológico.

Depois de construída a ontologia, procede-se ao desenvolvimento de um ficheiro XML (eXtensible Markup Language), que será usado para a execução das operações de carregamento, através de queries SQL (Search and Query Language), e transformação dos dados vindos da base de dados. Concluído o desenvolvimento deste ficheiro, o KDIS irá então fazer o carregamento dos dados para a triple store, ficando então a informação disponível para visualização na plataforma OObian.

Relativamente aos diferentes graus de expressividade [19], RDF(S) costuma usar-se para criar taxonomias e propriedades; OWL-Lite acrescenta equivalências e restrições a essa taxonomia; OWL-DL fornece o máximo de expressividade, garantindo que todas as conclusões são computáveis e que todas as computações acabam num tempo finito; e por fim, OWL-Full permite um máximo de expressividade mas sem garantias computacionais.

Por seu lado, XML é uma linguagem que define um conjunto de regras para codificação de documentos num formato que seja lido tanto por humanos como por máquinas [8].

2.1.2 Projetos de Apoio

Na subsecção anterior foi apresentada a solução desenvolvida pela Maisis para a realização de projetos semelhantes ao projeto em causa, o KDIS. Sendo esta solução uma das grandes ferramentas da Maisis para integração de sistemas legados (ex: base de dados), significa que já foi utilizada no desenvolvimento de outros projetos e que contém documentação complementar. Posto isto, é possível afirmar que existe material que pode servir de apoio para uma melhor compreensão da arquitetura do KDIS, e para ajudar na tarefa de desenvolvimento através de um conjunto de exemplos.

Voltando à subsecção 1.5.3, conclui-se que a metodologia de mapeamento desenvolvida pela Maisis se adequa a vários projetos e não se restringe a um único projeto, o que facilita em parte o desenvolvimento de projetos futuros.

2.2 Análise de Ferramentas Auxiliares

Tendo em conta o módulo que se pretende desenvolver, as ferramentas que poderão servir de auxilio enquadram-se em ferramentas OBDA, *Ontology-Based Data Access*, uma vez que fazem a tarefa de mapeamento automaticamente.

Num sistema OBDA, o objetivo é o de dar aos utilizadores acesso aos dados fonte, armazenados numa base de dados relacional, ou a uma coleção destes, por meio de uma vista conceptual específica de alto nível, uma ontologia [12]. Esta ontologia é usada como um camada adicional de informação colocada por cima da base de dados, com o propósito de enriquecer esta semanticamente. Normalmente, a ontologia contém a parte terminológica do conhecimento base, enquanto que a *Database Management System* (DBMS) é usada para gerir os dados atuais [10] [17].

Portanto, a ferramenta que se procura tem de ter as seguintes características:

- Ser uma ferramenta OBDA:
  - Acede a uma base de dados para extração de informação;

  - Faz o mapeamento desses dados para uma ontologia;
- Envia o resultado do mapeamento para a plataforma OOBian.

É então necessário procurar outras ferramentas já existentes que tenham, pelo menos parcialmente, alguma ou várias das características mencionadas, ou que possam auxiliar no desenvolvimento do projeto, nomeadamente porque podem facilitar/contribuir para garantir que se cumprem as características e os objetivos pretendidos.

2.2.1 Critérios de Avaliação

Antes de se proceder à análise de ferramentas, é necessário identificar os critérios de avaliação para concluir se a usabilidade de uma determinada ferramenta se enquadra ou não no âmbito do projeto.

Após uma boa compreensão do que se pretende obter com a realização do referido estágio, foram identificados os seguintes critérios de avaliação:

- Fazer *queries* SQL à base de dados relacional para obtenção dos dados;
- Esqueleto da ontologia guardado num ficheiro OWL;
- Aquando do mapeamento, permite modificar o *schema* da base de dados, ou seja, o *schema* ontológico não precisa de corresponder 100% ao *schema* relacional.

Terminada a identificação dos critérios de avaliação, na subsecção seguinte são apresentadas as possíveis soluções a incorporar no projeto.

2.2.2 Análise de Ferramentas

No início da presente secção foram identificadas duas grandes características que devem estar na solução final. O KDIS, solução apresentada na subsecção 2.1.1, encaixa no segundo caso, pois sendo uma ferramenta da Maisis, é normal que já tenha os mecanismos de comunicação com a plataforma OOBian implementados. Quanto a ferramentas de OBDA, serão apresentadas duas ferramentas que têm um grande destaque na atualidade da Web Semântica, R2RML (*RDB To RDF Mapping Language*) e D2RQ (*Database to RDF Querying*).

R2RML - RDB To RDF Mapping Language

Antes demais, mencionar que R2RML é uma W3C *Recommendation*, datada a 27 de Setembro de 2012. Uma ferramenta/tecnologia ser considerada como uma recomendação W3C significa que se encontra na etapa final de um processo de confirmação do W3C[1].

R2RML é uma linguagem de mapeamento personalizado de bases de dados relacionais para grafos RDF, sob sintaxe Turtle (*Terse RDF Triple Language*), que se encontra em desenvolvimento [11].

Em grande maioria dos projetos onde é necessário efetuar um mapeamento, é feito um mapeamento direto da base de dados relacional para RDF [16], ou seja, a estrutura do grafo RDF resultante é igual à estrutura da base de dados, o vocabulário RDF tem de ser igual aos nomes dos elementos do esquema da base de dados, e, por fim, a estrutura e o vocabulário não podem ser alterados. Em R2RML não existem estas restrições, pois o autor pode definir pontos de vista altamente personalizados sobre os dados existentes na base de dados relacional. É esta a grande vantagem do R2RML.

O mapeamento feito em R2RML recorre a tabelas lógicas para aceder aos dados da base de dados, sendo que estas tabelas lógicas são o resultado de *queries* SQL feitas à base de dados que serão mapeadas para triplos RDF (sujeito + predicado + objeto). Tabelas lógicas podem ser divididas em tabelas ou vistas bases de SQL e em vistas R2RML. Tabelas ou vistas bases de SQL não são mais do que tabelas lógicas que contêm dados SQL de tabelas ou vistas da base de dados. Por outro lado, vistas R2RML são as tabelas lógicas que definem a grande vantagem do R2RML, pois são o resultado de *queries* SQL feitas à base de dados que podem sofrer computação para o referido mapeamento personalizado.

D2RQ - Database to RDF Querying

Uma plataforma D2RQ[2] é um sistema que permite aceder a bases de dados relacionais de uma forma virtual, ou seja, permite obter a informação existente nas bases de dados através de grafos RDF [7], sem ser necessário replicar os dados para uma *triple store*.

Esta plataforma faz uso de uma linguagem de mapeamento, que por sua vez também se dá pelo nome de D2RQ, para descrever a relação entre um

[1] W3C - http://www.w3.org/
[2] D2RQ - http://d2rq.org/

schema de uma base de dados relacional e um *schema* ontológico descrito, sendo que o resultado final de um mapeamento D2RQ é um documento RDF escrito sob a sintaxe Turtle.

O mapeamento realizado define um grafo RDF virtual que contém informação da base de dados, onde a plataforma D2RQ fornece acesso a este grafo através de diversas maneiras:

- *Queries* SPARQL;
- Servidor de dados ligados;
- Gerador de RDF;
- Interface HTML.

De realçar que o *schema* ontológico resultante do mapeamento corresponderá na integra ao *schema* relacional, ou seja, ao contrário do R2RML, não é possível modificar o *schema* ontológico e modelá-lo de acordo com as necessidades de cada um.

Ferramentas/Tecnologias Comerciais

Noutro sentido, foram ainda analisadas ferramentas comerciais, ou seja, ferramentas proprietárias de certas entidades. A escolha das ferramentas a analisar foi feita tendo em conta as características apresentadas no início desta secção. Contudo, nenhuma das ferramentas analisadas se enquadra no propósito do projeto, pois não são ferramentas OBDA, pelo que descartámos apresentar cada uma delas.

2.2.3 Avaliação de Características

Terminada a apresentação das duas ferramentas analisadas, de seguida listamos as características identificadas para cada uma delas, assim como da solução da Maisis.

KDIS

Ao longo da análise e da exploração do KDIS foram identificadas determinadas características, nomeadamente:

- O KDIS permite modelar os dados vindos dos sistemas externos, o que pode enriquecer as pesquisas feitas pelos utilizadores finais;
- A sintaxe que usa na construção das *queries* deverá estar de acordo com o sistema externo, o que para o projeto em questão representa o uso de *queries* SQL;
- Apesar de se poder tornar extenso e ter de ser desenvolvido manualmente, o ficheiro de mapeamento não revela um elevado grau de complexidade devido a se encontrar bem estruturado. Inicialmente é necessário indicar todos os parâmetros para estabelecer comunicação com a base de dados e o servidor OObian; de seguida definem-se as *queries* SQL para obtenção de dados; e por fim é feito o mapeamento para cada classe em separado;
- O *schema* ontológico usado pode ser criado a partir de um software, o que agiliza esta tarefa no processo de mapeamento;
- A integração dos dados não é feita em tempo real, mas sim de uma forma assíncrona, ou seja, tem-se sempre que executar o KDIS para atualizar a *triple store* com informação da base de dados uma vez que só o KDIS consegue enviar informação;
- Podem aparecer resultados indesejados aquando de uma pesquisa, pois poderão ser associadas palavras que em nada se refiram ao contexto pretendido, daí advém a importância da relevância das instâncias a fim de se poder efetuar uma triagem dos termos mais importantes;
- A arquitetura do KDIS está construída de modo a que seja possível executar este em separado do servidor OObian. Esta característica pode trazer vantagens, como por exemplo, num cenário em que se pretenda executar o KDIS numa máquina "mais perto"da base de dados para reduzir o tempo de acesso a esta.

De referir que parte destas características estão referenciadas no manual de apresentação e utilização do KDIS, elaborado pela Maisis [2], sendo que foram comprovadas aquando da nossa análise.

R2RML

Relativamente ao R2RML, de seguida são apresentadas as grandes características identificadas para esta ferramenta:

- Mesmo tendo em conta que o mapeamento tem de ser feito manualmente, este mapeamento pode ser personalizado e não tem de seguir o *schema* relacional;
- Linguagem de mapeamento assenta em sintaxe Turtle;
- Permite juntar diferentes mapas de triplos num só, o que significa que se pode juntar informação de mais do que uma base de dados;
- Compatibilidade com vários motores de bases de dados [27];
- Não existe uma plataforma definida que permita usar R2RML de uma forma intuitiva e com uma certa facilidade (*user friendly*);
- Por se encontrar em fase de estudo e evolução, ainda não é considerado como um standard.

D2RQ

O mapeamento em D2RQ é uma tarefa que pode ser feita manualmente, num editor de texto, contudo, não se estaria a aproveitar a sua grande vantagem. Usando D2RQ, podemos utilizar uma ferramenta que gera automaticamente um ficheiro com o esqueleto do mapeamento a partir do *schema* da base de dados.

Além desta vantagem, foram identificadas ainda outras características, nomeadamente:

- O *schema* ontológico segue integralmente o *schema* relacional;
- Não é necessário duplicar os dados da base de dados para uma *triple store*;
- Tal como o R2RML, a linguagem de mapeamento assenta em sintaxe Turtle;
- Não permite operações de criação, eliminação e atualização, ou seja, serve apenas para ler informação [5];

- Apesar de ter uma plataforma para consulta da informação, a sua interface é pouco ilustrativa e bastante limitada, pois apenas permite inserir *queries* SPARQL, visualizar os resultados e pesquisar pelas classes e propriedades do grafo.

2.3 Solução Eleita

Cumprida mais uma fase do processo de análise e avaliação de possíveis soluções para a realização do projeto, chegou a etapa em que se faz a junção das características das diferentes soluções e se define por fim qual a mais indicada.

Na secção 2.2 foram identificadas as duas principais características que a solução deve ter, ser uma ferramenta OBDA e conseguir comunicar com a plataforma OObian para envio do resultado do mapeamento.

Juntando as características de todas as ferramentas, é possível construir uma tabela para visualizar os pontos em comum e as suas diferenças:

Características	**KDIS**	**R2RML**	**D2RQ**
Comunicação com Bases de Dados	x	x	x
Queries SQL na Comunicação com BDs	x	x	x
Mapeamento Ontológico Modelável	x	x	
Mapeamento	Manual	Manual	Automático
Sintaxe da Ontologia	OWL	TURTLE	TURTLE
Comunicação com Plataforma OObian	x		

Tabela 2.1: Características das Ferramentas Analisadas.

Analisando a tabela 2.1 podem-se tirar certas ilações para eleger a solução final. No que ao D2RQ diz respeito, a grande vantagem de realizar o mapeamento automaticamente encaminhava esta ferramenta para ser uma boa aquisição. Contudo, dado que a PT Inovação pretende agrupar e disponibilizar a informação de uma maneira diferente do que faz agora, o D2RQ teve de ser descartado devido ao *schema* ontológico que produz não ser flexível, pois tem de seguir o *schema* relacional.

Estudando agora as outras duas ferramentas, pode-se perceber que é possível utilizar KDIS e R2RML para realizar o mapeamento modelável pretendido pela PT Inovação, sendo que as diferenças encontram-se na sintaxe da ontologia e na comunicação com a plataforma OObian. Relativamente a estes diferenças, a questão da sintaxe seria um problema fácil de resolver, pois existem diversos conversores de Turtle para OWL na Internet, como por

exemplo, *OWL Syntax Converter*, da Universidade de Manchester [3]. Por outro lado, estabelecer comunicação entre R2RML e o servidor OObian já seria uma tarefa mais complicada, pois seria necessário alterar a arquitetura do servidor OObian e essa tarefa poderia entrar em conflito com outros projetos da Maisis.

Perante a análise elaborada, podemos concluir que o KDIS e o R2RML são duas ferramentas com um processo de mapeamento muito semelhante, e que qualquer uma poderia ser usada para esta tarefa. Porém, o KDIS já tem todos os mecanismos de comunicação com a plataforma OObian implementados, pelo que a solução final a utilizar recaiu sobre a ferramenta da Maisis, pois desta maneira podemo-nos focar inteiramente nos requisitos que o cliente pretende ver implementados. Outra diferença entre estas duas ferramentas, é que ao usar o KDIS é-nos garantida a existência de colaboradores com experiência caso seja necessário pedir auxílio.

Em suma, conclui-se que a Web Semântica, à semelhança da grande área da tecnologia, tem vindo a evoluir ao longo dos tempos [22] [26], e assim se deve manter, pelo que não é possível afirmar que determinada ferramenta ou tecnologia é a solução perfeita e por isso não se deva alterar as metodologias de trabalho e de desenvolvimento. Dado a rápida evolução, é recomendável que se elabore um trabalho de engenharia e que de tempos a tempos se investigue a evolução ou aparição de novas soluções.

Num futuro próximo, talvez a empresa Maisis possa incorporar novas ferramentas e tecnologias na plataforma OObian, ou, em sentido contrário, tornar o KDIS numa tecnologia standard.

[3] OWL Syntax Converter - http://owl.cs.manchester.ac.uk/converter/

Capítulo 3

Análise de Requisitos

Neste capítulo apresentamos a análise de requisitos da solução que se pretende construir, dando a possibilidade aos leitores de visualizar alguns mockups gráficos para uma melhor compreensão de cada requisito e, consequentemente, do produto final. Antes de serem apresentados os diferentes requisitos, a secção 3.1 apresenta a metodologia a seguir para uma correta análise.

3.1 Metodologia de Análise

A escolha da metodologia a seguir recaiu sobre FURPS+, uma metodologia muito usada devido à sua estrutura estar bem definida e organizada, e por os grupos de características serem os mais adequados para uma correta avaliação final deste projeto.

FURPS/FURPS+

FURPS [14] é um acrónimo que representa um modelo para classificar a qualidade dos atributos de um sistema, sendo que estes atributos não são mais do que os requisitos, quer funcionais quer não-funcionais.

- **F**uncionalidade - requisitos funcionais, que descrevem os comportamentos e capacidades funcionais que o cliente espera da aplicação;
- **U**sabilidade - requisitos do ponto de vista do utilizador, ou seja, aspetos como os graus de eficácia e de eficiência do produto, se a interface tem uma estética aceitável e é consistente, e precisão e completude da documentação;

- Confiabilidade (***R**eliability*) - requisitos referentes a disponibilidade do sistema (tempo que está disponível), exatidão nos cálculos e capacidade para recuperar de falhas;
- **P**erformance - características como rendimento e diferentes tempos, de resposta, de recuperação, de iniciação e de encerramento;
- **S**uporte - aspetos sobre possibilidade de fazer testes ou não, adaptação, manutenção, compatibilidade, configuração, instalação, escalabilidade e localização.

Quanto à metodologia FURPS+, não é mais do que uma extensão da FURPS que ajuda a ter em conta mais algumas necessidades que os clientes possam ter:

- Requisitos de Design - especifica as opções para desenhar um sistema, como por exemplo, a especificação de uso de uma base de dados relacional;
- Requisitos de Implementação - obrigação ou não do uso de standards no desenvolvimento, uso de determinadas linguagens de programação, e existência de limitação nos recursos disponíveis;
- Requisitos de Interface - aspetos relacionados com interação: restrições de formatos ou itens externos que o sistema tem de usar;
- Requisitos Físicos - que tipo de hardware é necessário para suportar o produto final.

3.2 Levantamento de Requisitos

O desenvolvimento de um produto de software está muitas vezes associado aos utilizadores finais, pelo que é necessário tipificar este tipo de utilizadores de modo a saber o que cada um pretende com o uso do produto final. Esta tipificação resulta num conjunto de utilizadores denominado de *Personas*.

A presente secção será divida em duas subsecções, uma onde se identificam as *personas* e quais os padrões de interação de cada uma com o sistema, e quais as suas exigências relativas ao sistema, e outra onde são apresentados os requisitos segundo a metodologia escolhida anteriormente.

3.2.1 *Personas*

Para tipificação dos utilizadores, a PT Inovação teve um papel preponderante uma vez que já tinha identificado a priori as *personas* e quais as perguntas que cada uma quer ver respondidas. A tabela 3.1 agrega esta informação.

		Personas				
		Gestor de Projeto	Gestor de Rede	Gestor de Provisão	Gestor de Manutenção	Operador de *Call Center*
Principais Perguntas	Encaminhamento de Rede, por Serviço		X		X	
	Equipamentos de Suporte a VPNs			X	X	X
	Pontos de Acesso a VPNs		X	X	X	
	Dependências de Serviço		X	X	X	
	Serviços por Cliente				X	X
	Distribuição de Ocupação por Equipamento	X				

Tabela 3.1: *Personas* e Principais Perguntas

Gestor de Projeto

Persona cujo objetivo é o de projetar diferentes redes, redes de serviços para clientes (telefone, *triple play*, etc.) ou redes internas da empresa (Internet, VPNs, etc.), e ainda identificar as variações bruscas de ocupação dessas redes.

Para tais tarefas, necessita de consultar a informação dos equipamentos existentes para saber a conectividade e o estado dos respetivos portos, e ainda de saber que equipamentos estão associados a quais. Os dados devolvidos podem ser úteis para alarmística, pois podem ajudar a evitar ou a corrigir eventuais problemas na rede.

Gestor da Rede

Tem como objetivo construir a rede "projetada", pelo Gestor de Projeto, para satisfazer os clientes.

De modo a atingir os seus objetivos, necessitam de identificar os recursos (equipamentos, portos e ligações) que podem utilizar para criar a rede, assim como saber quantos pontos de acesso existem para aceder à rede.

Gestor de Provisão

Pessoas responsáveis por configurar/ativar serviços para clientes empresariais e residenciais, quer a componente física, como equipamentos e cabos de cobre ou fibra ótica, quer a componente lógica, como é o caso de portos e ligações.

Tal como os gestores de rede, necessitam principalmente de identificar recursos e a quantidade de pontos de acesso.

Gestor de Manutenção

Conjunto de colaboradores responsáveis por assegurar o correto funcionamento dos equipamentos e garantir a total e correta disponibilidade dos diferentes serviços, como por exemplo, o serviço MEO dos seus clientes ou a rede de Internet da própria empresa.

Sendo uma área com grande responsabilidade, necessitam de praticamente toda a informação existente relacionada com serviços: todos os tipos de serviços que existem, que serviços funcionam em conjunto, quais os clientes que usam cada um dos serviços, e por fim, os recursos que cada serviço necessita para um correto funcionamento.

Operador de *Call Center*

Colaboradores que se encontram numa estrutura física, com capacidade para receber, em média, seis milhões de ligações telefónicas por mês, com o propósito de fornecer atendimento aos clientes que ligarem a pedir auxílio.

Para ajudar os clientes, estes colaboradores precisam de ter acesso a todos os serviços de cada cliente e aos respetivos equipamentos de suporte.

3.2.2 Requisitos

Na realização de um projeto de software, muitas vezes é necessário priorizar os diferentes requisitos para ajudar a organizar o desenvolvimento e orientar o produto para o resultado pretendido. A tabela 3.2 apresenta os tipos de prioridade pelos quais os requisitos serão listados.

Prioridade	**Descrição**
MUST	Requisito que tem de ser totalmente desenvolvido para o produto final poder ser considerado como um sucesso.
SHOULD	Requisito com prioridade alta que, se possível, deve estar na solução final; não impossibilita o sucesso do produto final caso não seja implementado.
NICE	Requisito considerado desejável mas sem ser necessário implementar, sendo apenas desenvolvido caso haja tempo.

Tabela 3.2: Prioridades dos Requisitos

Na presente subsecção optou-se por apresentar duas listas de requisitos, e respetivas prioridades, com o objetivo de apresentar ao leitor uma vista geral dos requisitos do projeto. Em anexo, no apêndice C, está presente uma descrição mais detalhada de cada um dos requisitos.

A primeira lista de requisitos, tabela 3.3, contém os requisitos funcionais, requisitos que descrevem um serviço ou uma função que o sistema deve realizar.

ID	**Requisito**	**Prioridade**
1	Efetuar pesquisas	MUST
2	Distribuição de ocupação por equipamento	MUST
3	Serviços por cliente	MUST
4	Dependências de serviço	MUST
5	Encaminhamento de rede, por serviço	MUST
6	Pontos de acesso a VPNs	MUST
7	Equipamentos de suporte a VPNs	MUST
8	Lista de clientes	SHOULD
9	Lista de serviços	SHOULD
10	Lista de equipamentos	SHOULD
11	Lista de portos	SHOULD
12	Lista de ligações	SHOULD
13	Nome dos registos na visualização	SHOULD
14	Clientes associados	SHOULD
15	Serviços associados	SHOULD
16	Estado dos serviços	SHOULD
17	Estado dos equipamentos	SHOULD
18	Estado dos portos	SHOULD
19	Equipamentos ligados entre si	SHOULD
20	Localização geográfica de equipamentos	SHOULD
21	Diferentes níveis de localização geográfica de equipamentos	NICE
22	Páginas de Internet	NICE
23	Restrição de Acesso	SHOULD

Tabela 3.3: Lista de Requisitos Funcionais

Por fim, a tabela 3.4 contém os requisitos não funcionais, considerados como restrições impostas tanto no sistema como no seu desenvolvimento.

Classificação	**ID**	**Requisito**	**Prioridade**
Usabilidade	24	Eficácia	MUST
	25	Eficiência	MUST
Performance	26	Tempo de Resposta	MUST
Suporte	27	Integração com Sistemas BSS	NICE
	28	Processamento de Informação	MUST
Design	29	Base de Dados Relacional Oracle	MUST
	30	Plataforma OObian	MUST
Implementação	31	Linguagem SQL	MUST
	32	Linguagem OWL	MUST
	33	Duplicação de Dados	MUST
Interface	34	Cliente OObian	MUST
Físicos	35	Dispositivo de Computação	MUST

Tabela 3.4: Lista de Requisitos Não Funcionais

Relativamente aos requisitos de confiabilidade, estes fogem ao âmbito do projeto, pois questões relacionadas com disponibilidade do sistema, ou com a capacidade para recuperar de falhas, são aspetos controlados pela Maisis uma vez que o servidor foi desenvolvido e é gerido pelos seus colaboradores.

Quanto à base de dados da PT Inovação, uma vez que esta é gerida pelos mesmos, o cliente indicou que são eles os responsáveis por controlar os acessos à base de dados e por atualizar os dados na *triple store*.

Capítulo 4

Planeamento Detalhado

Aquando da etapa final da primeira fase, foi delineado um plano de tarefas para a segunda fase (presente no final do apêndice A), que sofreu certas alterações relativamente ao plano seguido. Portanto, o capítulo 4 consiste na descrição do plano de tarefas efetivamente seguido na segunda fase. De seguida apresentamos o respetivo diagrama de Gantt.

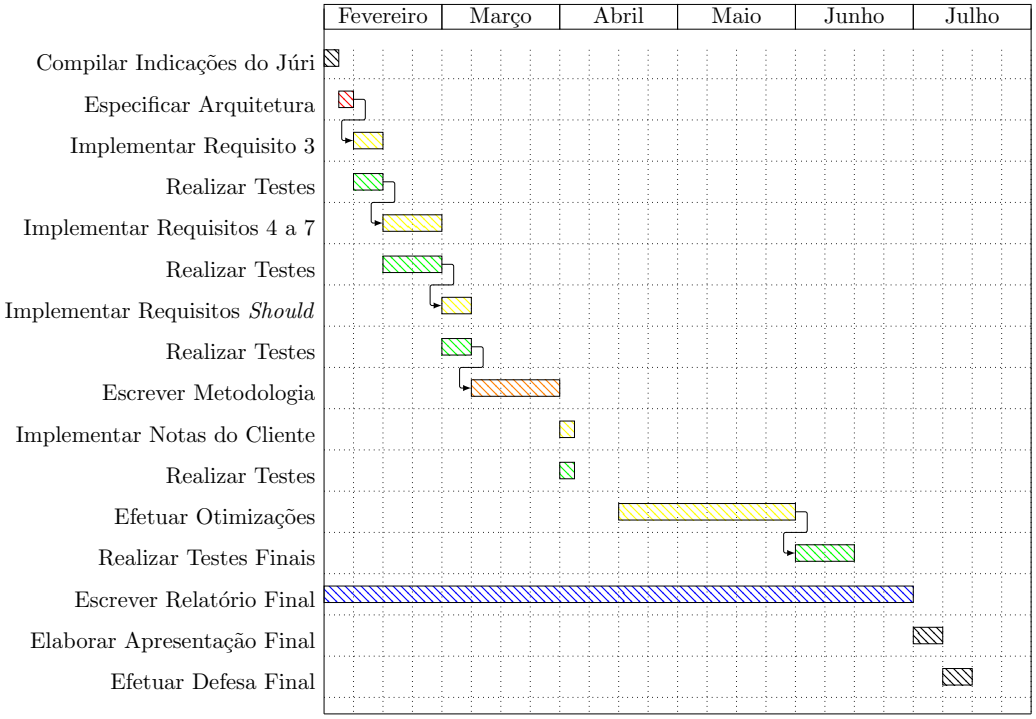


Figura 4.1: Diagrama de Gantt da Segunda Fase

Ao analisar a lista de tarefas e respetiva duração, a tarefa de escrever o relatório final salta à vista devido a ter um tempo de duração equivalente ao segundo semestre. Pretende-se assim mostrar que esta tarefa foi feita ao longo do semestre, em paralelo com as restantes tarefas, para não haver perda de informação e para ter sempre um registo atualizado de todo o trabalho desenvolvido.

Um dos capítulos do relatório que foi escrito ao longo do tempo foi o capítulo 9 - Trabalho Futuro, pois muito deste trabalho consiste em melhorias, identificadas ao longo do desenvolvimento, a fazer nas ferramentas de desenvolvimento da Maisis. Por seu lado, o capítulo 8 - Análise Crítica e Retrospetiva, foi elaborado somente no fim de todo o desenvolvimento e fase de testes, uma vez que é um capítulo que conclui todo o processo do projeto.

Inicialmente, foi necessário compilar as indicações provenientes do júri durante a defesa intermédia num documento para as usar como guia extra durante o segundo semestre. Esta tarefa não ocupou mais do que dois dias, pelo que nos restantes dias dessa semana completou-se a especificação da arquitetura. De seguida, entrou-se na grande tarefa da segunda fase, desenvolvimento.

O desenvolvimento encontra-se dividido em duas fases: segunda semana de Fevereiro à primeira semana de Março, inclusive, e os dois meses de Abril e Maio. Durante a primeira fase foram implementados os requisitos funcionais de maior prioridade, requisitos 3 a 7, e os de prioridade *Should*, requisitos 8 a 20. Devido à metodologia de desenvolvimento seguida, sub-secção 1.5.2, à medida que se desenvolvia um grupo de requisitos, estes eram testados e corrigiam-se os erros que fossem detetados, e apenas depois de corretamente implementados se passava à implementação do grupo de requisitos seguinte.

Terminada a primeira fase de implementação, pretendia-se ter uma reunião com o cliente de modo a apresentar o trabalho implementado, receber o respetivo *feedback*, e proceder às alterações identificadas pelo cliente. Porém, a disponibilidade da PT Inovação era reduzida e apenas conseguimos ter esta reunião no fim do mês de Março, ou seja, cerca de três semanas após a conclusão da primeira fase de implementação. De modo a avançar na lista de tarefas a realizar, durante este intervalo de tempo foi elaborado o documento com a metodologia genérica de trabalho, pretendido pela Maisis, que se encontra em anexo no apêndice D.

Após a reunião com a PT Inovação, demos início à segunda fase de desenvolvimento. Inicialmente procedemos às alterações identificadas pelo cliente, testando simultaneamente o respetivo resultado, e de seguida demos início ao carregamento de toda a informação da base de dados para a *triple store*,

cerca de dois milhões de registos. Enquanto a informação era carregada, aproveitou-se o tempo para avançar na escrita do relatório, como por exemplo, do capítulo 6 - Desenvolvimento.

Ao fim de semana e meia de carregamento, apenas um terço da informação tinha sido carregada, pelo que nos vimos na necessidade de criar um segundo cenário de carregamento, secção 6.4, de modo a se fazerem otimizações, com o objetivo de reduzir o tempo de carregamento de toda a informação. Esta tarefa levou cerca de mês e meio, mas os resultados obtidos foram positivos e podem ser consultados no capítulo 7 - Testes e Resultados.

Para terminar esta segunda fase de implementação, foram feitos testes com cerca de três/quatro utilizadores em simultâneo de modo a testar alguns requisitos não funcionais, nomeadamente, requisitos de usabilidade, performance e suporte.

Por fim, após a entrega do documento final, será elaborada a apresentação de todo o trabalho, a defender na segunda semana do mês de Julho, terminando deste modo todo o processo do projeto de estágio.

Capítulo 5

Arquitetura da Solução

Neste capítulo será apresentada a arquitetura geral da solução, sendo que, posteriormente, é apresentado um outro diagrama que apenas diz respeito ao que foi desenvolvido.

5.1 Arquitetura Geral

Uma vez que a arquitetura do projeto pode ser apresentada através dos vários componentes do sistema, e do modo como estes comunicam entre si, a sua apresentação segue o diagrama de componentes da linguagem UML [4], apresentado na figura 5.1.

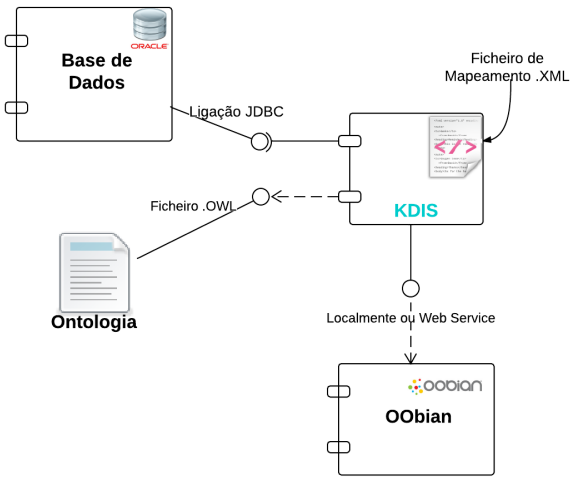


Figura 5.1: Arquitetura Geral da Solução.

Como se pode observar na figura 5.1, é possível identificar três componentes: Base de Dados, KDIS e OObian. Para obter os dados a apresentar, o KDIS estabelece uma ligação JDBC com a base de dados relacional Oracle de onde vai extrair informação através de *queries* SQL. JDBC, Java Database Connectivity, é uma API cujo propósito é mesmo o de estabelecer comunicações com bases de dados relacionais através de instruções SQL. Relativamente à ontologia, será criada manualmente no software Protégé e exportada para um ficheiro a que o KDIS vai aceder de modo a ter conhecimento do *schema* ontológico.

Por forma a garantir a correção da ontologia face aos objetivos pretendidos, esta será regularmente apresentada à PT Inovação para validação e pedido do respetivo *feedback*.

Tendo a informação vinda da base de dados e a ontologia, constrói-se um ficheiro em linguagem XML que conterá todo o mapeamento do *schema* relacional para o *schema* ontológico. Terminada a tarefa de mapeamento, o KDIS comunica com a plataforma OObian para lhe enviar o resultado final. Esta comunicação pode ser feita de duas maneiras: diretamente, caso o servidor de conhecimento se encontre local, ou através de um Web Service, caso a ligação tenha de ser feita remotamente.

5.2 Arquitetura do Desenvolvimento

Com o objetivo de apresentar a arquitetura das partes que foram desenvolvidas, transformou-se a figura 5.1 na figura 5.2.

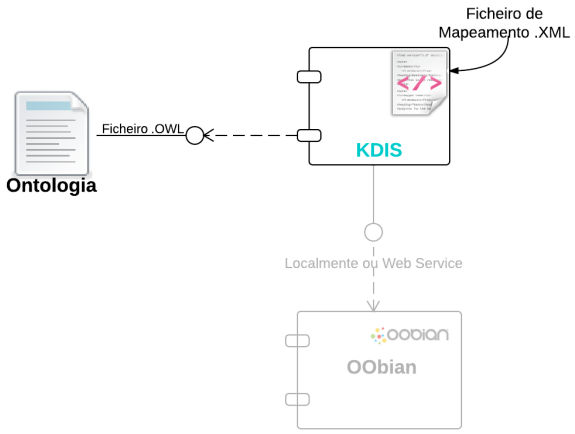


Figura 5.2: Arquitetura do Trabalho Desenvolvido.

A criação da ontologia, no software Protégé, envolve três fases:

- Criação das diferentes classes;
- Definição das propriedades que ligam essas classes;
- Especificação dos atributos de cada uma das classes.

Relativamente ao KDIS, o que se pretende é configurar um ficheiro .XML que contenha o mapeamento do *schema* relacional para o *schema* ontológico. A estrutura do ficheiro é a seguinte:

- Comunicação com o KServer;
- Definição da ontologia a utilizar;
- Comunicação com a base de dados;
- *Queries* SQL para recolha de informação;
- Mapeamento do *schema* relacional para o *schema* ontológico.

Como se pode observar na figura 5.2, a componente da plataforma OObian encontra-se ligeiramente opaca. Isto deve-se ao facto de que na plataforma OObian não se vai fazer qualquer desenvolvimento, pois já se encontra desenvolvida, mas serão feitas algumas tarefas para gestão do produto final, nomeadamente:

- Carregamento do *schema* ontológico;
- Configuração de quais propriedades são filtráveis e quais são pesquisáveis por texto livre.

No próximo capítulo, Desenvolvimento, serão apresentados excertos destas três componentes, de modo a apresentar o trabalho desenvolvido, mas também para auxiliar numa melhor compreensão da arquitetura de cada uma.

Capítulo 6

Desenvolvimento

O presente capítulo tem o objetivo de descrever os principais métodos, escolhas e resultados do processo de desenvolvimento da ferramenta pretendida de acordo com a arquitetura previamente definida. Para facilitar a compreensão do que foi desenvolvido, ilustramos este capítulo com diversas imagens.

Inicialmente temos a ontologia que foi criada no software Protégé, detalhando depois a configuração do ficheiro .XML, que consiste no mapeamento do *schema* relacional para o ontológico, seguida de uma secção que contém as tarefas realizadas na plataforma OObian, assim como um conjunto pequeno de imagens a ilustrar a visualização da informação neste mesma plataforma. Por fim, apresentamos e explicamos os dois cenários desenvolvidos para carregamento da informação para a *triple store*.

6.1 Ontologia

Na secção 5.2 foram mencionados os três passos que dizem respeito à criação da ontologia, sendo que na presente secção é apresentada uma imagem, por cada um desses passos, com o resultado obtido.

Por fim, é apresentada uma imagem a ilustrar o caminho a seguir para se obter o ficheiro .OWL, que contém o *schema* ontológico.

A criação da ontologia foi feita no software Protégé, versão 3.4.8, mas também é possível construir ontologias noutras versões do mesmo software, ou até com outro editor de ontologias, como por exemplo o TopBraid Composer[1].

[1]TopBraid Composer - http://www.topquadrant.com/products/TB_Composer.html

O primeiro passo foi o de criar as diferentes classes. A figura 6.1 ilustra a hierarquia de classes final.

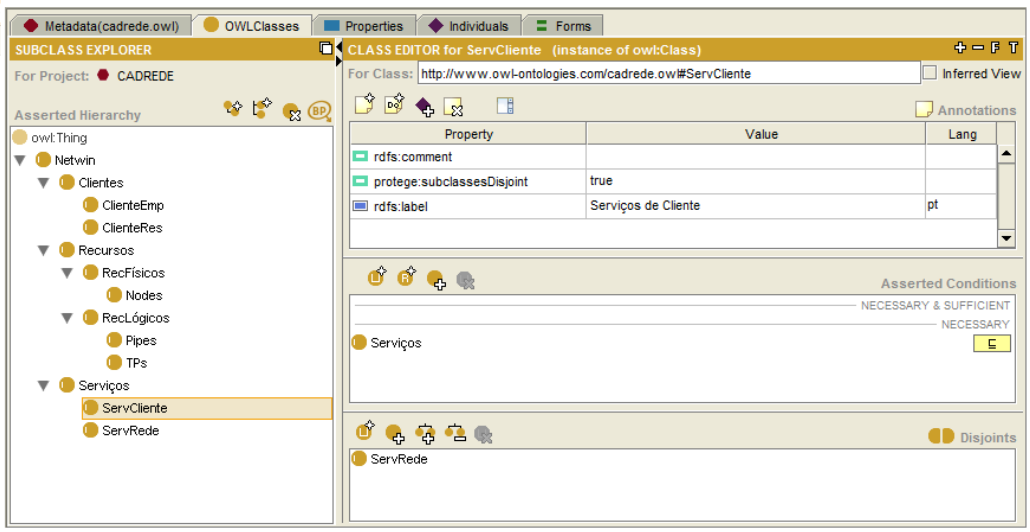


Figura 6.1: Ontologia - Classes.

Como se pode observar no lado esquerdo na figura 6.1, existe a classe principal **Netwin**, que é o nome escolhido pela PT Inovação para designar o projeto, que por sua vez tem três sub-classes, **Clientes**, **Recursos** e **Serviços**.

Dentro da classe de clientes existem dois tipos, **ClienteEmp**, empresariais, e **ClienteRes**, residenciais, cujos serviços que lhes dizem respeito pertencem à sub-classe **ServCliente** da classe serviços. Os outros serviços, de rede, são os serviços internos da PT Inovação, pelo que pertencem à sub-classe **ServRede**.

Relativamente aos recursos que os serviços necessitam para um correto funcionamento, existem dois tipos: recursos físicos, **RecFísicos**, e recursos lógicos, **RecLógicos**. Os físicos são os equipamentos, cujo nome técnico é **Nodes**, e os lógicos são os portos e as ligações, que se designam, respetivamente, por **TPs** e **Pipes**.

Por último, de referir a propriedade **label** que se encontra ao centro da figura 6.1. O valor que é definido em cada label, é o texto que irá aparecer na interface da plataforma OObian.

Esta propriedade é também definida para cada uma das *Object* e *Datatype Properties*, como se pode confirmar nas figuras 6.2 e 6.3, respetivamente.

De seguida, na figura 6.2, estão presentes as propriedades que ligam as classes anteriores entre si.

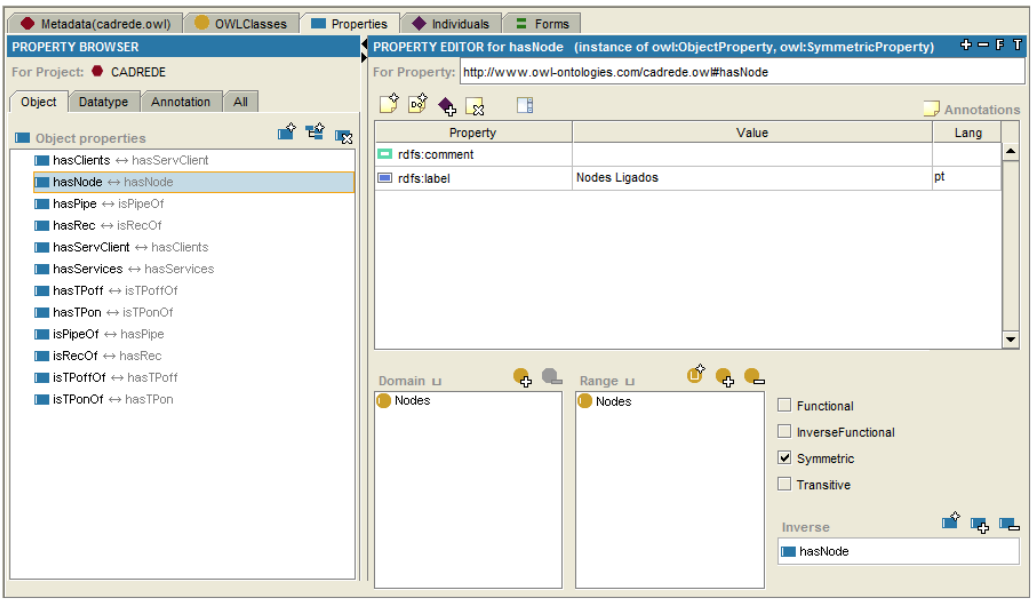


Figura 6.2: Ontologia - *Object Properties*.

Na figura 6.2 encontram-se as doze propriedades que ligam as diferentes classes, sendo que todas elas têm uma propriedade inversa, que é a propriedade que se encontra à frente de cada uma, de cor cinzenta. Agrupando estas propriedades, temos que:

- **hasClients** liga os serviços com os clientes que usufruem destes, e **hasServClient** liga os clientes com os serviços que lhes dizem respeito;
- **hasPipe** permite relacionar os equipamentos e os portos com as suas ligações, e **isPipeOf** faz com que seja possível ver que equipamentos e portos uma ligação liga;
- **hasRec** é a ponte de ligação entre os serviços e os recursos que necessitam para funcionar corretamente, enquanto **isRecOf** faz a relação contrária;
- **hasTPoff** relaciona os equipamentos com os seus portos que se encontram desligados, e **isTPoffOf** corresponde à ligação inversa;

- **hasTPon** e **isTPonOf** relacionam as mesmas classes que as propriedades anteriores, com a diferença que os portos são os que se encontram ligados.

Quanto às outras duas propriedades, a sua inversa é ela própria, o que significa que são propriedades simétricas. Analisando cada uma em separado:

- **hasNode** liga equipamentos que comunicam entre si;
- **hasServices** estabelece ligação entre serviços que funcionam em dependência.

O terceiro e último passo da construção da ontologia foi o de especificar os atributos de todas as classes. A figura 6.3 apresenta a lista desses atributos.

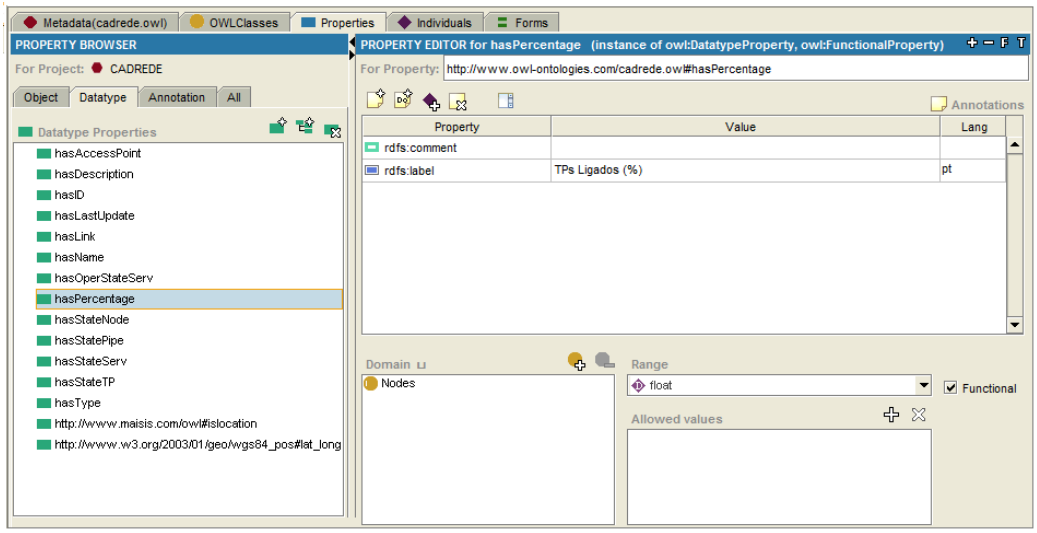


Figura 6.3: Ontologia - *Datatype Properties*.

Passando a apresentar os atributos da figura 6.3:

- **hasAccessPoint** - número de pontos de acesso que existem a um serviço VPN;
- **hasDescription** - descrição textual;
- **hasID** - identificador;
- **hasLastUpdate** - data da última atualização da informação na base de dados;

- **hasName** - nome;
- **hasOperStateServ** - estado operacional de um serviço:
- **hasPercentage** - percentagem de portos ligados que um equipamento tem;
- **hasStateNode** - estado operacional de um equipamento;
- **hasStatePipe** - estado operacional de uma ligação;
- **hasStateServ** - estado de vida de um serviço;
- **hasStateTP** - estado operacional de um porto;
- **hasType** - tipo de classe.

Na lista de atributos apresentada anteriormente, é possível observar que existem dois tipos de estado: operacional e de vida, sendo que este último apenas pertence a serviços. Os valores que estes atributos podem ter foram definidos pela PT Inovação aquando do início dos seus projetos. Portanto, os valores de um estado operacional podem ser ENABLE, DISABLE ou DISABLE.MAINTENANCE, enquanto o estado de vida de um serviço informa se esse serviço se encontra Ativo, Abortado, Em Provisão ou Terminado.

No fim da figura 6.3 é ainda possível visualizar mais duas propriedades, **islocation** e **lat_long**, que têm uma taxonomia diferente. Esta taxonomia deve-se ao facto de serem propriedades pré-definidas pela Maisis para que seja possível visualizar informação geograficamente, na funcionalidade de mapas, e por essa razão provêm de *namespaces* diferentes dos da nossa ontologia.

Cumpridos estes passos, a ontologia estava criada e pronta a ser exportada para o ficheiro .OWL a que o KDIS acede. A figura 6.4 ilustra o caminho para fazer esta exportação.

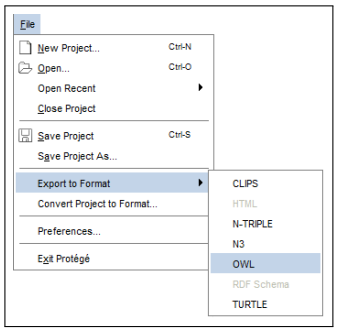


Figura 6.4: Ontologia - Ficheiro .OWL.

Neste momento já se tinha, quer o *schema* relacional quer o *schema* ontológico, pelo que estavam reunidas as condições para se avançar para a tarefa de mapeamento, tarefa essa que é da responsabilidade do agente de integração de dados, KDIS.

6.2 Agente de Integração de Dados

Relativamente ao KDIS, o pretendido era configurar um ficheiro .XML que contivesse o mapeamento do *schema* relacional para o *schema* ontológico. Nesta secção descrevemos o processo de mapeamento, ilustrando-o com imagens de excertos deste ficheiro correspondentes às várias partes que o constituem.

Tendo como base os pontos mencionados na secção 5.2 sobre a estrutura do ficheiro .XML, de seguida damos início à apresentação e explicação do referido excerto.

1. Comunicação com o KServer e definição da ontologia a utilizar.

```
<kdismapping:KServerConnect isRemoteConnection="true"
    username="stepahead" password="PaSSw0rd!"
    serverURI="http://localhost:8280/services/ws/KServerWS?wsdl">

    <kdismapping:TargetOntologyID>
        http://www.owl-ontologies.com/cadrede.owl#
    </kdismapping:TargetOntologyID>
</kdismapping:KServerConnect>
```

Figura 6.5: KDIS - KServer e Ontologia.

A figura 6.5 apresenta o preenchimento dos campos para comunicação com o KServer, sendo que o primeiro campo, **isRemoteConnection**, deve ter valor **"true"** quando a comunicação com o servidor OObian tem de ser feita remotamente. Caso contrário, se for uma ligação local, o valor deve ser **"false"**. De seguida são inseridos os dados de utilizador, dados estes que servem de acesso ao *back office* da plataforma OObian. No campo **serverURI** define-se o porto do endereço de acesso, que neste caso é 8280.

Por seu lado, para definir a ontologia, deve inserir-se o URI desta (*Uniform Resource Identifier*), definido aquando da sua construção no software Protégé, no bloco **TargetOntologyID**.

2. Comunicação com a base de dados.

```
<kdismapping:Connection
    uri="jdbc:oracle:thin:@10.112.80.23:1521:devcad"
    connDriver="oracle.jdbc.driver.OracleDriver"
    id="myDB" dbUser="cadrede" dbPassword="admdev"
    xsi:type="kdismapping:DBConnectType"/>
```

Figura 6.6: KDIS - Base de Dados.

O bloco de código XML apresentado na figura 6.6 diz respeito à configuração dos parâmetros para comunicação com a base dados onde se encontra a informação a extrair. Os campos **uri** e **connDriver** servem para definir qual o tipo de base de dados a que se vai aceder, que para o efeito é uma base de dados Oracle.

Quanto aos dados de utilizador, aqui têm de se colocar os dados de utilizador da base de dados, pois é necessário ter permissões de acesso para consulta da informação.

3. *Queries* SQL para recolha de informação da base de dados.

A fase seguinte na configuração do ficheiro .XML passa por definir as *queries* SQL que serão executadas na base de dados para obtenção da informação. Esta fase pode ser dividida em três conjuntos, onde o primeiro grupo serve somente para ir buscar os atributos para cada uma das classes da ontologia, como se pode constar na figura 6.7.

```
<!-- Queries -->
<!-- Nodes -->
<kdismapping:Query dbID="myDB" id="queryNode1"
    value="SELECT ino.id_bd_entity node_id, cn.type node_type, ino.operational_state node_state
                    FROM ns_res_ins_node ino, ns_res_cat_node cn
                    WHERE (ino.id_bd_cat_node = cn.id_bd_cat_node)
                        and (ino.operational_state IN ('ENABLED', 'DISABLED.DAMAGED', 'DISABLED.MAINTENANCE'))
                        and (ino.id_bd_entity between 22 and 5200000)"
    xsi:type="kdismapping:DBQueryType"/>

<kdismapping:Query dbID="myDB" id="queryNode2"
    value="SELECT ino.id_bd_entity node_id, cn.type node_type, ino.operational_state node_state
                    FROM ns_res_ins_node ino, ns_res_cat_node cn
                    WHERE (ino.id_bd_cat_node = cn.id_bd_cat_node)
                        and (ino.operational_state IN ('ENABLED', 'DISABLED.DAMAGED', 'DISABLED.MAINTENANCE'))
                        and (ino.id_bd_entity between 5200001 and 6300000)"
    xsi:type="kdismapping:DBQueryType"/>
```

Figura 6.7: KDIS - *Queries* SQL para Informação.

A figura 6.7 apresenta a configuração de uma *query*, onde o primeiro campo a preencher, **dbID**, tem o valor definido no campo **id** da figura 6.6, e o segundo é o nome da *query*, campo **id**. O código SQL que constitui a *query* deve ser inserido no campo **value**.

Como se pode observar na figura 6.7, temos duas *queries* que devolvem os mesmos atributos, mas cuja única diferença é o intervalo de registos. Esta configuração tem o objetivo de dividir o volume de registos a carregar da base de dados em conjuntos mais pequenos.

Uma vez que para cada registo se fazem *queries* para saber os seus relacionamentos, e no caso dos serviços e equipamentos mais uma e duas, respetivamente, para atributos, os restantes registos da *query* em execução ficam guardados em memória, pelo que tiveram de ser criados diferentes blocos para que o carregamento da informação não fosse interrompido por falta de memória. De modo a saber qual a quantidade possível de registos a carregar de cada vez, foram feitos testes, seguindo a abordagem "tentativa e erro", para concluir qual a quantidade suportada em memória. Após estes testes, chegámos à conclusão que a quantidade máxima deve ser aproximadamente de 100.000 registos.

O segundo conjunto de *queries* tem o propósito de recolher os atributos, **ObjectProperties**, que relacionam as diferentes classes entre si. Na figura 6.8 encontram-se duas destas *queries*.

```
<!-- ObjectProperties Queries -->
<!-- Nodes e TPs -->
<kdismapping:Query dbID="myDB" id="queryHasTPon"
    value="SELECT id_bd_entity_tp FROM ns_res_ins_tp_node
        WHERE id_bd_entity_node = ? AND id_bd_entity_tp in (SELECT id_bd_entity_tp FROM ns_res_ins_pipe_tp)"
    xsi:type="kdismapping:DBQueryType"/>

<kdismapping:Query dbID="myDB" id="queryHasTPoff"
    value="SELECT id_bd_entity_tp FROM ns_res_ins_tp_node
        WHERE id_bd_entity_node = ? AND id_bd_entity_tp not in (SELECT id_bd_entity_tp FROM ns_res_ins_pipe_tp)"
    xsi:type="kdismapping:DBQueryType"/>
```

Figura 6.8: KDIS - *Queries* SQL para *Object Properties*.

Ao observar a figura 6.8 é possível confirmar que a estrutura do código XML é semelhante à da figura 6.7. Contudo, a *query* SQL é diferente na cláusula **WHERE**, pois esta tem de receber um parâmetro, símbolo **?**, para que sejam devolvidos todos os registos que lhe estão associados. A apresentação, e respetiva justificação, da figura 6.13 pode ajudar a compreender esta *query*.

Por fim, existe um último grupo, apresentado na figura 6.9, cujo propósito

é o de obter dados para criar atributos, **Datatype Properties**, que não estão na base de dados mas que se pretendem no produto final.

```
<!-- DataProperties Queries -->
<!-- Nodes -->
<!-- Percentagem de TPs Ligados -->
<kdismapping:Query dbID="myDB" id="queryPercentage"
    value="SELECT round(((tpON.lig*100)/allTP.tot) , 2) resultado
            FROM (SELECT count(itn.id_bd_entity_tp) lig FROM ns_res_ins_tp_node itn, ns_res_ins_pipe_tp ipt
                WHERE itn.id_bd_entity_node = ? AND itn.id_bd_entity_tp = ipt.id_bd_entity_tp ) tpON,
               (SELECT count(id_bd_entity_tp) tot FROM ns_res_ins_tp_node WHERE id_bd_entity_node = ?) allTP
            WHERE allTP.tot > 0"
    xsi:type="kdismapping:DBQueryType"/>
```

Figura 6.9: KDIS - *Queries* SQL para *Datatype Properties* Específicas.

A *query* apresentada na figura 6.9 diz respeito ao requisito "Distribuição de ocupação por equipamento", pelo que pretendia-se que o seu resultado fosse um valor percentual. Uma vez que este valor não existe na base de dados, foi necessário efetuar operações no próprio código SQL, como se pode confirmar na figura 6.9. As operações efetuadas são: contagem (**count()**), multiplicação e divisão (**(tpON.lig*100)/allTP.tot**). Relativamente à função **round()**, esta serve para arredondar e apresentar um valor com duas casas decimais.

4. Mapeamento do *schema* relacional para o *schema* ontológico.

Após definidas as *queries* para obtenção de informação, procede-se ao bloco que contém o mapeamento do *schema* relacional para o ontológico. A figura 6.10 contém a estrutura do mapeamento que deve ser feito para cada classe.

```
<!-- Recursos -->
<!-- Node -->
<kdismapping:Map confidence="1.0" isIndexable="true" queryID="queryNode1" source="myDB"
        targNamespace="http://www.owl-ontologies.com/cadrede.owl#">

    <!-- Identificador na Triple Store -->
    <kdismapping:ClassID id="http://www.owl-ontologies.com/cadrede.owl#Nodes"
        namespace="http://www.owl-ontologies.com/cadrede.owl#"/>

    <kdismapping:ID>
        <kdismapping:Field language="en" type="xsd:String">
            <kdismapping:Data value= "NODE_" />
            <kdismapping:Data queryColumn="NODE_ID" />
        </kdismapping:Field>
    </kdismapping:ID>

    <kdismapping:Properties>
        <!-- DataProperties -->

        <!-- ObjectProperties -->
    </kdismapping:Properties>

</kdismapping:Map>
```

Figura 6.10: KDIS - Mapeamento: Estrutura e Configurações.

As primeiras configurações servem para especificar qual a *query* de onde vêm os dados, campo **queryID**, definida anteriormente no bloco da figura 6.7, e definir um ID para a classe em questão na *triple store*, secções **ClassID** e **ID**. Para tal, na primeira secção, especifica-se qual o URI da classe a atribuir ao ID, definido anteriormente no Protégé, no campo **id**. De seguida, na segunda secção, indica-se qual a coluna da *query*, campo **queryColumn**, que contém o valor pretendido para o ID.

O campo **value** da segunda secção é opcional, sendo que a sua utilização apenas vai influenciar o identificador que será guardado na *triple store*. Neste caso, os equipamentos ficaram com um identificador do tipo 'NODE_123456789'. Caso se pretenda especificar este campo, também se tem que o especificar, e ter o mesmo valor, quando se relacionar esta classe com outra, através de uma *Object Property*, de modo a fazer a correta correspondência.

Posteriormente, temos presente a secção de mapeamento das propriedades, **Properties**. Uma vez que as propriedades se dividem em *Object* e *Datatype*, a apresentação dos seus mapeamentos foi divida em duas partes. A figura 6.12 contém o código XML referente às *Datatype Properties*, enquanto a figura 6.13 diz respeito às *Object Properties*.

Contudo, imediatamente antes do bloco *Datatype Properties*, pode existir

o bloco de código apresentado na figura 6.11.

```
<kdismapping:Labels>
    <kdismapping:Field language="en" type="xsd:String">
        <kdismapping:Data queryColumn="NODE_NAME" />
    </kdismapping:Field>
</kdismapping:Labels>
```

Figura 6.11: KDIS - Mapeamento: *Label* da Classe.

O bloco de código da figura 6.11 diz respeito ao nome da classe que será apresentado na interface de visualização, secção **Labels**. Para esta configuração a PT Inovação escolheu a coluna da *query* que contem o nome do registo, campo **queryColumn**, com exceção nos portos, pois pediu que a sua apresentação tivesse o formato 'NODE_NAME - TP_NAME', onde 'NODE_NAME' é o nome do equipamento a que o porto pertence.

Continuando na secção das propriedades, passamos à apresentação das *Datatype Properties* na figura 6.12.

```
<!-- DataProperties -->
<!-- Tipo -->
<kdismapping:DataProperty id="http://www.owl-ontologies.com/cadrede.owl#hasType">
    <kdismapping:Field language="en" type="xsd:String">
        <kdismapping:Data queryColumn="NODE_TYPE" />
    </kdismapping:Field>
</kdismapping:DataProperty>

<!-- Percentagem -->
<kdismapping:DataProperty id="http://www.owl-ontologies.com/cadrede.owl#hasPercentage" queryID="queryPercentage">
    <kdismapping:QueryParameters>NODE_ID</kdismapping:QueryParameters>
    <kdismapping:QueryParameters>NODE_ID</kdismapping:QueryParameters>
    <kdismapping:Field language="en" type="xsd:float">
        <kdismapping:Data queryColumn="RESULTADO" />
    </kdismapping:Field>
</kdismapping:DataProperty>
```

Figura 6.12: KDIS - Mapeamento: *Datatype Properties*.

O mapeamento das *Datatype Properties* pode ser feito de duas maneiras, como demonstra a figura 6.12. No primeiro caso, simplesmente se faz corresponder o valor da coluna que vem na *query* com o URI da propriedade, especificados nos campos **queryColumn** e **id**, respetivamente.

O segundo caso de mapeamento está relacionado com o mapeamento feito na figura 6.9. Para se fazer uso do **resultado** devolvido pela *query* da figura 6.9, é necessário adicionar um campo na primeira linha, **queryID**, que é a referência à *query* que contém o valor da propriedade, **queryPercentage**, e de seguida acrescentar a secção **QueryParameters** para se especificar qual

será o parâmetro a enviar para essa *query*.

Ao observar a figura 6.9 com certo detalhe, é possível confirmar que existem dois caracteres **?**, caracteres esses que não fazem parte da sintaxe SQL. O seu significado é o de indicar que é necessário enviar parâmetros para completar a *query*, sendo esse envio feito no mapeamento da *Datatype Property* correspondente, figura 6.12. Neste caso específico, existem dois sinais **?**, pelo que é necessário enviar dois parâmetros, apesar desse parâmetro ser o mesmo, **NODE_ID**, razão pela qual existem duas secções **QueryParameters**.

Por fim, é apresentada na figura 6.13 a estrutura geral de um mapeamento de uma *Object Property*.

```
<!-- ObjectProperties -->
<!-- TPs Ligados Associados -->
<kdismapping:ObjectProperty confidence="1.0" dbSource="myDB"
        id="http://www.owl-ontologies.com/cadrede.owl#hasTPon"
        queryID="queryHasTPon" notCheckInverse="true">

    <kdismapping:QueryParameters>NODE_ID</kdismapping:QueryParameters>
    <kdismapping:ConnectedObjectID>
        <kdismapping:Field language="en" type="xsd:String">
            <kdismapping:Data value= "TP_" />
            <kdismapping:Data queryColumn="ID_BD_ENTITY_TP" />
        </kdismapping:Field>
    </kdismapping:ConnectedObjectID>
</kdismapping:ObjectProperty>
```

Figura 6.13: KDIS - Mapeamento: *Object Properties*.

Observando a figura 6.13 é possível concluir que o mapeamento de uma *Object Property* é muito parecido com o segundo caso de mapeamento das *Datatype Properties*. No início especifica-se qual o URI da propriedade no campo **id**, depois a *query* onde se encontra o seu valor, no campo **queryID**, de seguida indica-se o parâmetro a enviar para a *query* na secção **QueryParameters**, e termina-se indicando no campo **queryColumn** qual a coluna da *query* que se pretende.

A terminar, referir que o campo **notCheckInverse** tem de ser definido com valor **true**, uma vez que as propriedades inversas são configuradas neste ficheiro e não se pretende portanto que o KServer calcule a inversa de cada uma das propriedades. A definição das propriedades inversas é feita uma

vez que para cada classe se configura todas as suas propriedades, *Object e Datatype Properties*, pelo que acaba-se sempre por definir as propriedades inversas.

Comunicação com a Plataforma OObian.

Assim que o ficheiro de mapeamento se encontrar configurado, corre-se o KDIS para que este envie o resultado do mapeamento para o servidor OObian. Assim que o KDIS terminar o envio da informação, basta aceder à versão cliente do OObian e consultar a informação.

Para mais informação de como correr o KDIS e aceder ao cliente OObian, consultar o final da secção 4.2 do documento Metodologia Genérica de Trabalho, em anexo no apêndice D.

6.3 Plataforma OObian

Anteriormente, na secção 5.2, foi mencionado que na plataforma OObian apenas serão feitas duas tarefas para gestão do produto final.

Para tal, tendo em conta a configuração efetuada para comunicação com o KServer, aquando do mapeamento, acede-se ao endereço `http://localhost:8280/oobian-admin/`, com os dados de utilizador especificados na figura 6.5, e procede-se então à gestão da ontologia.

A primeiro tarefa é a de carregar o ficheiro .OWL para a plataforma OObian de modo a que esta tenha conhecimento do *schema* ontológico. A figura 6.14 apresenta o local onde se pode fazer este carregamento.

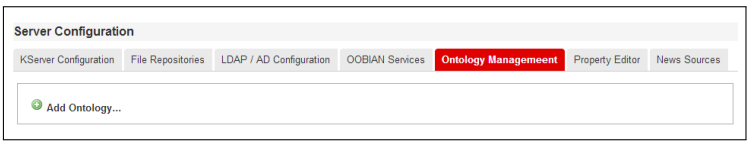


Figura 6.14: Plataforma OObian - Gestão de Ontologias.

Na área **Server Configuration** da figura 6.14, tab **Ontology Management**, é possível adicionar a ontologia. Ao primir o botão **Add Ontology** o utilizador é redirecionado para a página apresentada na figura 6.15.

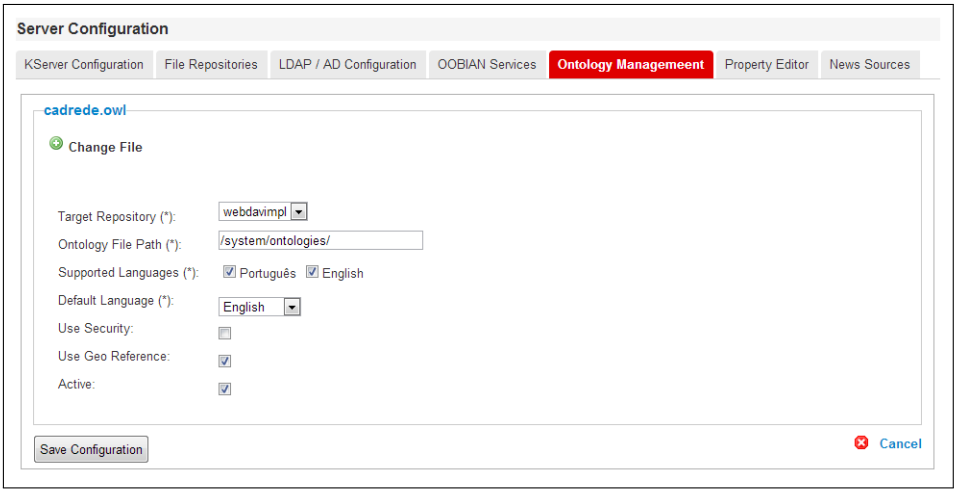


Figura 6.15: Plataforma OObian - Adicionar uma Ontologia.

A figura 6.15 mostra as definições que se têm de especificar para se adicionar uma ontologia. As mais importantes de definir são as línguas suportadas, se se faz uso de referências geográficas, e se a ontologia a carregar fica logo ativa ou não.

Por seu lado, a realização da segunda tarefa, configuração de quais propriedades são filtráveis e quais são pesquisáveis por texto livre, é feita na mesma área, **Server Configuration**, mas na tab **Property Editor**, tal como ilustra a figura 6.16.

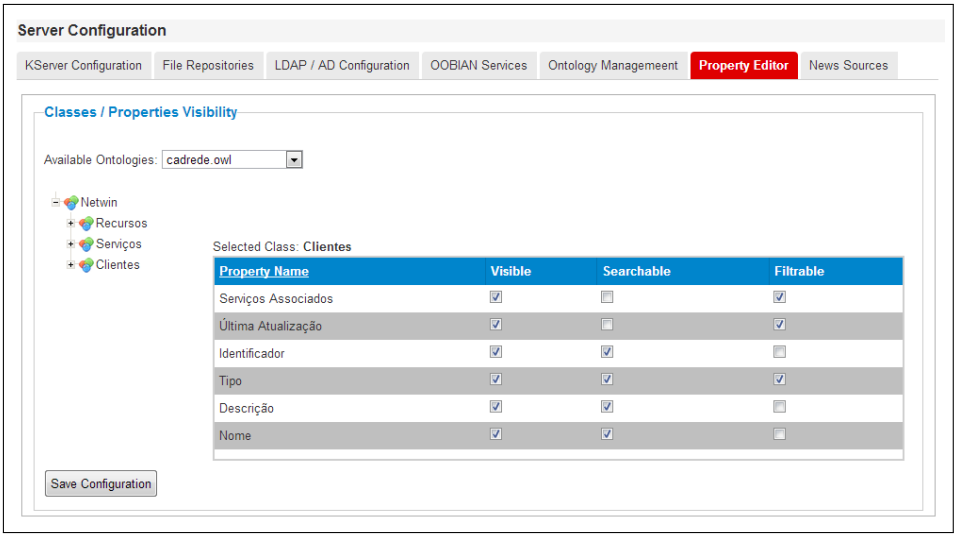


Figura 6.16: Plataforma OObian - Configurar Propriedades.

Para fazer esta configuração, basta selecionar sequencialmente cada classe da ontologia, no lado esquerdo da figura 6.16, e para cada propriedade da classe selecionada definir se é pesquisável e/ou filtrável.

Interface de Visualização

Ao longo do documento é referido que a interface da plataforma OObian já se encontra desenvolvida, não sendo por isso uma das componentes a desenvolver. Contudo, com o objetivo de completar a descrição das componentes desenvolvidas, de seguida temos três imagens que ilustram o modo como a informação mapeada é apresentada.

A figura 6.17 ilustra parte da hierarquia de classes criada no software Protégé, e apresentada na figura 6.1.

Figura 6.17: Plataforma OObian - Hierarquia de Classes.

Como se pode observar, derivada da classe Serviços temos as suas duas sub-classes, Serviços de Cliente e Serviços de Rede, fazendo tudo parte do projeto Netwin. Anteriormente, na figura 6.1, estas sub-classes têm os nomes ServCliente e ServRede, respetivamente, contudo, por motivos de usabilidade, a sua visualização na plataforma OObian recorre às labels criadas e apresentadas na figura 6.1, que são preenchidas com os valores "Serviços de Cliente"e "Serviços de Rede", respetivamente.

Após o utilizador navegar para a janela da figura 6.17, uma das suas possibilidades é navegar para a classe Serviços de Cliente, cujo resultado se encontra na figura seguinte.

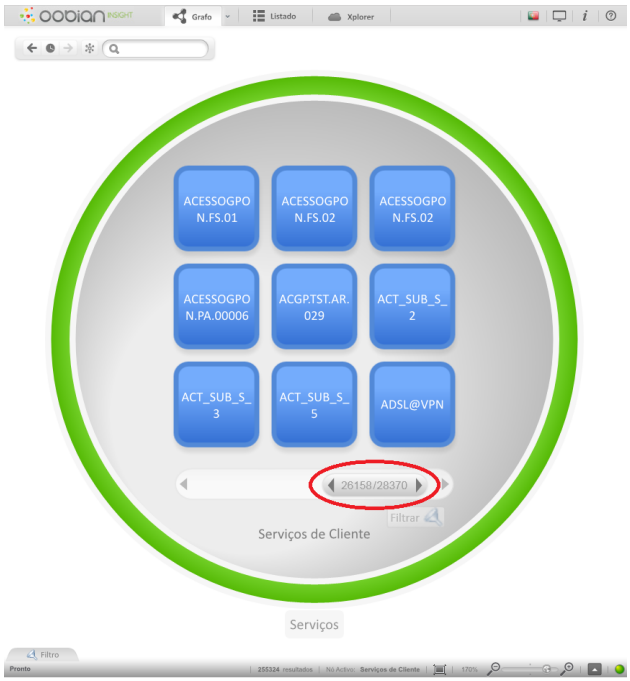


Figura 6.18: Plataforma OObian - Instâncias de uma Classe.

A figura 6.18 apresenta todas as instâncias da classe Serviços de Rede, que se encontram distribuídas de forma igual por 28.370 páginas.

Estas instâncias foram carregadas da base de dados em *queries* semelhantes às da figura 6.7, e o seu mapeamento foi configurado seguindo o exemplo apresentado na figura 6.10.

De seguida terminamos a presente secção com a figura 6.19, que permite visualizar um equipamento, as suas relações e os seus atributos.

Figura 6.19: Plataforma OObian - Instância e suas Propriedades.

A figura 6.19 pode ser dividida em três partes: do lado esquerdo os campos de pesquisa e filtragem, no lado direito os atributos da instância em visualização, e no centro a instância e respetivas relações.

As relações de uma instância têm três fases de criação: definição na ontologia como uma *Object Property* (figura 6.2), carregamento dos dados da base de dados (figura 6.8), e configuração do mapeamento (figura 6.13).

Relativamente aos atributos, estes têm uma criação muito semelhante à das relações, sendo a sua composta pelas mesmas três fases: definição na ontologia, mas como *Datatype Properties* (figura 6.3), carregamento dos dados da base de dados (figuras 6.7 e 6.9), e configuração do mapeamento (figura 6.12). O carregamento dos dados pode ser feito por duas vias, uma vez que é possível criar atributos que não se encontram na base de dados mas que se pretendem no produto final, como o caso do atributo TPs Ligados (%), onde é necessário efetuar cálculos para se obter o valor pretendido (figura 6.9). Por último, a outra via é carregar os dados existentes na base de dados sem fazer qualquer manipulação, figura 6.7.

A terminar, apresentamos na figura 6.20 uma funcionalidade da própria

plataforma OObian.

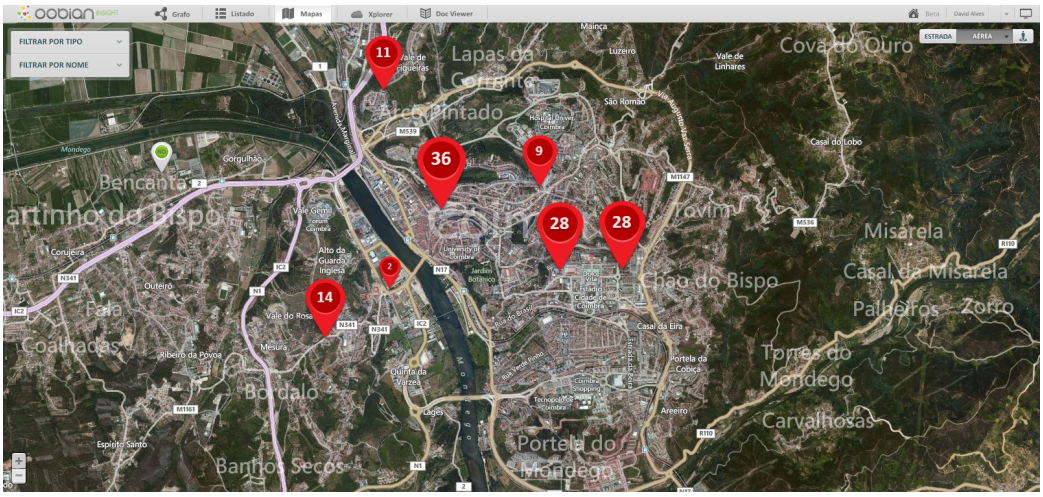


Figura 6.20: Plataforma OObian - Funcionalidade de Mapas.

A funcionalidade apresentada na figura 6.20 é a de mapas. Portanto, é possível visualizar geograficamente as instâncias que se encontrem georreferenciadas, sendo que nesta imagem estamos a visualizar grupos de equipamentos que se situam na área de Coimbra.

6.4 Carregamento da Informação

O carregamento de toda a informação presente na base de dados para a *triple store*, aquando da tentativa de o fazer pela primeira vez na sua totalidade, foi uma tarefa cujo tempo de duração foi aproximadamente de quatro semanas, o que nos levou a uma fase de desenvolvimento que se centrava em reduzir o tempo de carregamento. Para tal, foi criado um novo cenário de carregamento e feitas otimizações nas *queries*.

Esta secção ilustra os dois cenários de carregamento desenvolvidos, ficando a apresentação e discussão dos resultados obtidos para os capítulos seguintes. A primeira imagem, figura 6.21, representa o primeiro cenário.

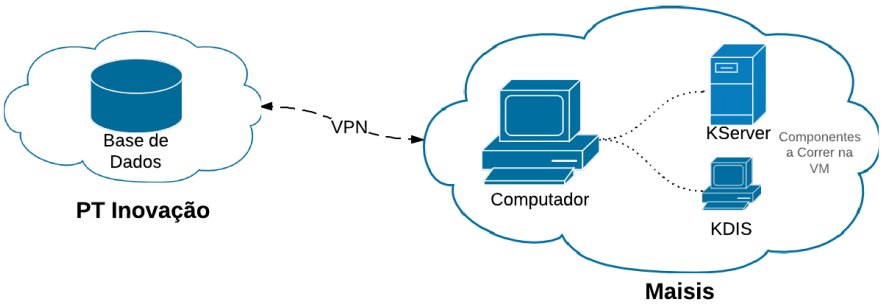


Figura 6.21: Cenário 1 de Carregamento.

O primeiro ambiente encontrava-se num computador com processador Intel®Core™i5, memória RAM de 6.00GB, sistema operativo Windows 7 de 64-bit, e com um disco rígido de 500GB. Dentro desta máquina foi instalada uma máquina virtual com sistema operativo Windows Server 2008, onde foram reservados 4.00GB de memória RAM e 50GB de espaço livre.

Todo o desenvolvimento foi efetuado dentro da máquina virtual, o que significa que esta continha tanto o servidor OObian como o KDIS em execução. De modo a ser possível aceder à base de dados, foi estabelecida uma comunicação VPN com a PT Inovação.

Relativamente ao segundo cenário, a figura 6.22 contém as alterações efetuadas.

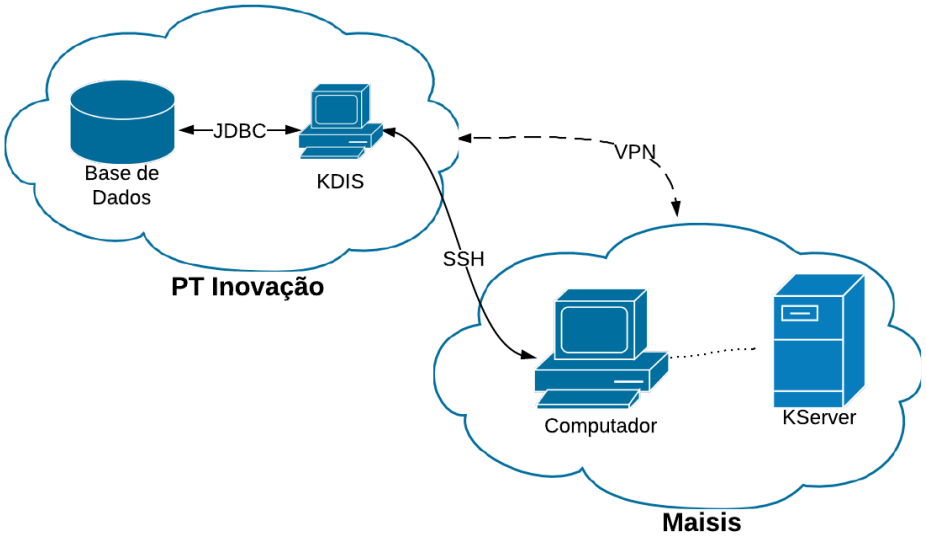


Figura 6.22: Cenário 2 de Carregamento.

O segundo ambiente, por seu lado, foi criado num computador que continha as mesmas características a nível de processador, sistema operativo e disco rígido, mas cuja memória RAM era apenas de 4.00GB. Contudo, não foi instalada nenhuma máquina virtual e o servidor OObian encontrava-se em execução na raiz da máquina, pelo que a memória RAM disponível era a mesma em relação ao primeiro ambiente.

A grande diferença neste cenário centra-se na comunicação com a base de dados e no local onde se executou o KDIS. Foi novamente estabelecida comunicação VPN com a PT Inovação, mas de seguida foi ainda feita uma ligação SSH (*Secure Shell*) a uma máquina em ambiente Linux que se encontra na PT Inovação, e assim ”mais perto”da base de dados. Por fim, foi colocado o KDIS em execução nesta nova máquina e a enviar os resultados para o servidor em execução na máquina da Maisis.

Este cenário é possível de acontecer devido à arquitetura do KDIS. Uma vez que se encontra separado do servidor, comunica primeiro com a base de dados e depois envia o resultado do mapeamento para o servidor.

Uma ligação SSH é um protocolo de rede que permite ligar um computador a outro e executar comandos neste segundo computador a partir do primeiro. Uma vantagem deste tipo de ligação é que já oferece algum nível de segurança, pois faz uso de um protocolo de rede encriptado para garantir confidencialidade e integridade dos dados, o que não se verifica na Internet.

Capítulo 7

Testes e Resultados

O presente capítulo apresenta o conjunto de testes realizados, e respetivos resultados, validando todo o desenvolvimento efetuado. Podemos dividir a fase de testes em duas frentes: carregamento da informação e validação dos requisitos.

A análise crítica dos resultados obtidos e respetivas conclusões podem ser consultadas no próximo capítulo, capítulo 8. Análise Crítica e Retrospetiva.

7.1 Carregamento da Informação

Na última secção do capítulo anterior, foi mencionado que o carregamento de toda a informação da base de dados para a *triple store* teve uma duração aproximada de quatro semanas, pelo que identificámos a necessidade de desenvolver um novo cenário de carregamento e fazer otimizações nas *queries* SQL.

Otimização de *Queries* SQL

Um dos pontos considerados logo de início como fator para o elevado tempo de carregamento foram as *queries* SQL efetuadas para obtenção dos dados da base de dados, razão pela qual foram feitos testes para concluir se o tempo total de carregamento podia ser reduzido através da otimização destas. Uma vez que o número total de clientes não chega a dois mil (2.000) registos, apenas foram feitos *scripts* para saber os tempos de *queries* relacionadas com serviços e recursos.

A tarefa de mapeamento consiste, para cada instância de uma classe, em

mapear os seus atributos e as suas relações, pelo que, para cada um dos registos carregados na primeira *query* (figura 6.7 do capítulo 6), ainda é executada mais uma *query* por cada relacionamento e, no caso de serviços e equipamentos, ainda são executadas mais uma e duas *queries*, respetivamente, para obter outros atributos.

Cada um dos *scripts* criados contém o conjunto de todas as *queries* que são executadas para cada instância, sendo que foram escolhidas cem instâncias de serviços, equipamentos, portos e ligações para conhecer os seus tempos de execução. Num universo de dois milhões de registos, cem instâncias de cada classe pode ser uma amostra pequena, no entanto, este teste apenas tinha o objetivo de identificar potenciais *queries* cujo tempo de execução fosse entre cinco e dez segundos, devido ao requisito de performance, pelo que não achámos necessário testar mais instâncias. A tabela 7.1 apresenta a média dos valores obtidos.

Classe	**Tempo Médio**
Serviços	$0.42seg$
Equipamentos	$0.86seg$
Portos	$0.198seg$
Ligações	$0.182seg$

Tabela 7.1: Tempos de Execução dos *Scripts*

Analisando a tabela 7.1 é possível constar que os tempos de execução não chegam a um segundo no pior caso, equipamentos. No entanto, estes tempos dizem respeito a instâncias que têm relações com apenas uma instância de cada uma das outras classes, o que significa que o número de *queries* executadas para cada uma delas é no máximo quatro.

Durante o carregamento, foi-nos possível consultar os ficheiros de *logs*, tanto do servidor OObian como do KDIS, e podemos concluir que o mapeamento dos equipamentos estava a ter um tempo de carregamento superior ao das outras classes. Tendo esta premissa, analisámos de seguida o ficheiro de mapeamento para estudar o número de relacionamentos e as *queries* executadas para cada equipamento.

A análise efetuada permitiu-nos concluir que um equipamento é sem dúvida a classe com maior número de relacionamentos, pois tem relações com os serviços onde é usado, os equipamentos que tem ligado a si, todos os seus portos, separados por ligados e desligados, e ainda as suas ligações. Quanto aos seus atributos, dois deles são resultados de novas *queries* feitas à base de dados, nomeadamente, a percentagem de portos ligados e georreferenciação.

Devido às conclusões tiradas, foi necessário correr o *script* de tempos de *queries* para um equipamento que tivesse um grande número de relações, na ordem das centenas, pelo que escolhemos o equipamento apresentado na figura 6.19. O tempo de execução deste *script* foi de $8,89seg$, muito superior ao tempo obtido para os equipamentos da tabela 7.1.

Perante o valor obtido, vimo-nos na necessidade de estudar individualmente cada uma das *queries* executadas por equipamento. Ao executar cada uma dessas *queries* em separado, os valores obtidos foram de $0,35seg$, em média, exceto numa *query*, cujo tempo de execução foi de $6,80seg$. Esta *query* dizia respeito ao relacionamento de um equipamento com outros equipamentos, e facilmente se conclui que era esta a *query* que estava a perturbar o tempo de carregamento.

Estudando o corpo da *query*, foi possível concluir que o seu tempo de execução era aproximadamente de $10seg$ devido a agregar um grande conjunto de tabelas, nove, e de ter quatro sub-consultas em cadeia, pelo que era estritamente necessário fazer uma otimização. De referir que uma sub-consulta é uma *query* dentro de outra *query*, ou seja, eram feitas quatro *queries*, cada uma dentro da anterior, na *query* principal.

A otimização efetuada consistiu em estudar melhor o modelo de dados de modo a reduzir o número de tabelas agregadas, o que permitiu reduzir o conjunto de tabelas agregadas para cinco e ter apenas uma sub-consulta. Após esta otimização, o tempo de execução da *query* passou de $6,80seg$ para $0,43seg$, o que representa uma redução de $93,68\%$.

No geral, o tempo de execução do *script* para o equipamento selecionado passou de $8,89seg$ para $2.52seg$, o que significa uma redução de $71,65\%$.

Por fim, e devido a termos concluído que as otimizações em *queries* podem surtir um grande efeito no tempo de redução do carregamento, todas as restantes *queries* foram analisadas. Contudo, nenhuma delas se revelou passível de otimização, pois encontravam-se o mais simples possível, pelo que a tarefa de otimização de *queries* estava concluída.

Cenários de Carregamento

De modo a poder comparar os dois cenários criados, foram feitos diferentes carregamentos de informação tendo em conta as seguintes variáveis:

- Memória alocada ao KDIS para execução;

- Número máximo de registos que um bloco suporta (*Page Size*) na navegação base de dados - *triple store*;
- Número de registos a mapear para cada classe.

De referir que o número de registos de clientes na base de dados, quer empresariais quer residenciais, não atinge os intervalos usados nos testes, pelo que o total de registos mapeados para estas classes é sempre constante, respetivamente, 1.847 e 6 registos.

O nosso primeiro teste foi o de recolher os tempos de carregamento entre um cenário e outro, fazendo apenas variar o número de registos a mapear. A tabela 7.2 contém os valores obtidos.

Mem=800MB & *Page Size*=200				
	2.500 Registos		5.000 Registos	
Classe	**Cenário 1**	**Cenário 2**	**Cenário 1**	**Cenário 2**
ClientEmp	$1min : 6seg$	$2min : 59seg$	$2min : 2seg$	$3min : 10seg$
ClientRes	$1seg$	$1seg$	$5seg$	$2seg$
ServCliente	$9min : 23seg$	$5min : 38seg$	$17min : 18seg$	$12min : 15seg$
ServRede	$9min : 57seg$	$5min : 58seg$	$17min : 25seg$	$12min : 44seg$
Nodes	$19min : 39seg$	$16min : 16seg$	$40min : 25seg$	$34min : 46seg$
TPs	$4min : 28seg$	$4min : 28seg$	$10min : 36seg$	$11min : 31seg$
Pipes	$3min : 38seg$	$4min : 34seg$	$8min : 00seg$	$10min : 47seg$
Total	$48min : 12seg$	$39min : 54seg$	$1h : 35min : 51seg$	$1h : 25min : 15seg$

Tabela 7.2: Tempos de Carregamento entre Cenários.

Os resultados recolhidos com este teste permitiram-nos concluir que o segundo cenário reduz o tempo de carregamento relativamente ao primeiro. Contudo, duplicando a quantidade de registos a mapear, constou-se que a diferença de tempos foi próxima. Para 2.500 registos, a variação de percentagem entre o cenário um e o cenários dois é de $17,22\%$. Por seu lado, para 5.000 registos, esta variação é de apenas $11,06\%$, o que significa que não existe uma grande redução temporal.

De seguida optámos por efetuar testes somente no segundo cenário, por se ter revelado mais rápido no sentido em que o tempo de processamento por registo é menor, fazendo variar a memória alocada e o *page size*, juntamente com o número de registos a mapear. Na tabela 7.3 apresentamos os resultados dos testes feitos com $5GB$ de memória e *page size* de 2.500 e 20.000, para o carregamento de 2.500 e 5.000 registos.

Memória de 5GB				
	Page Size=2.500		*Page Size*=20.000	
Classe	**2.500 Registos**	**5.000 Registos**	**2.500 Registos**	**5.000 Registos**
ClientEmp	$2min : 50seg$	$2min : 58seg$	$3min : 3seg$	$2min : 56seg$
ClientRes	$3seg$	$1seg$	$1seg$	$1seg$
ServCliente	$5min : 50seg$	$11min : 53seg$	$6min : 8seg$	$12min : 25seg$
ServRede	$6min : 16seg$	$12min : 14seg$	$5min : 55seg$	$13min : 39seg$
Nodes	$14min : 42seg$	$33min : 27seg$	$15min : 13seg$	$33min : 58seg$
TPs	$4min : 32seg$	$10min : 18seg$	$4min : 31seg$	$11min : 35seg$
Pipes	$4min : 36seg$	$9min : 59seg$	$4min : 24seg$	$10min : 10seg$
Total	$38min : 49seg$	$1h : 20min : 50seg$	$39min : 15seg$	$1h : 24min : 44seg$

Tabela 7.3: Tempos de Carregamento no Segundo Cenário.

Analisando a tabela 7.3, e tendo também presentes os tempos da tabela 7.2 no que ao cenário 2 diz respeito, é possível concluir que as variáveis memória e *page size* não têm influência no tempo de carregamento de informação.

Carregamento Total

Terminadas as tarefas de otimização, quer nas *queries* SQL quer no cenário de carregamento, foi efetuado um novo carregamento de toda a informação para se concluir se as otimizações efetuadas tiveram sucesso ou não.

A nossa abordagem foi a de dividir o total de registos a mapear em diferentes grupos para que fosse possível ter uma melhor noção dos tempos de carregamento, assim como para se ir validando o resultado do mapeamento na própria plataforma OObian. Esta divisão teve como critério ter aproximadamente 100.000 registos de cada classe, como apresentado na secção 6.2 do capítulo Desenvolvimento, e dado que a classe *TPs* tem mais de 600.000 registos, o número de grupos criado foi de seis. A quantidade de registos a mapear em cada um destes grupos é apresentada de seguida na tabela 7.4.

Classe	Grupo 1	Grupo 2	Grupo 3	Grupo 4	Grupo 5	Grupo 6	Total
ClientEmp	1.849	–	–	–	–	–	1.849
ClientRes	6	–	–	–	–	–	6
ServCliente	101.937	71.431	82.253	–	–	–	255.621
ServRede	100.215	71.435	79.476	–	–	–	251.126
Nodes	105.295	104.698	114.262	95.232	63.116	–	482.603
TPs	100.100	111.428	100.558	114.268	104.757	154.153	685.264
Pipes	102.793	95.256	104.764	104.769	95.786	–	503.368
Total	512.195	454.248	481.313	314.269	263.659	154.153	2.179.837

Tabela 7.4: Quantidade de Registos por Grupo.

Após configurarmos o ficheiro de mapeamento para ser feito em seis fases sequenciais, uma para cada grupo da tabela 7.4, demos início ao novo carregamento total. Relativamente ao cenário em que foi feito este carregamento, foi escolhido o segundo cenário, e a memória alocada foi de 5GB e o *page size* de 200. O tempo total deste carregamento, assim como de cada grupo, encontra-se na tabela 7.5.

Grupos	Tempo
Grupo 1	$29h : 37min : 16seg$
Grupo 2	$26h : 00min : 53seg$
Grupo 3	$48h : 36min : 19seg$
Grupo 4	$20h : 51min : 17seg$
Grupo 5	$15h : 34min : 40seg$
Grupo 6	$05h : 08min : 27seg$
Total	$6d : 1h : 48min : 52seg$

Tabela 7.5: Tempos de Carregamento Total.

Observando a última linha da tabela 7.5, facilmente se conclui que as otimizações efetuadas surtiram efeito, pois reduziu-se o tempo de carregamento total de quatro semanas para aproximadamente seis dias, o que significa uma redução de $78,57\%$.

Nota: um aspeto importante a revelar para a redução do tempo de carregamento total é que foram retirados alguns atributos, criados através de *queries* SQL, uma vez que não enriqueciam o resultado final e não eram requisitos do cliente, pelo que era de esperar que o tempo de carregamento descesse apenas por ser necessário efetuar menos *queries* à base de dados.

Por fim, foram feitos testes, também de carregamentos totais e com cenário igual ao do último carregamento total, tendo por base intervalos tem-

porais. O nosso objetivo foi o de apresentar ao cliente uma solução para quando tivesse a necessidade de atualizar a *triple store*. Os intervalos escolhidos, tendo como referência a data de 8 de Maio de 2013, foram: última semana, último mês e últimos três meses. Os tempos de carregamento obtidos foram os seguintes.

Intervalo Temporal	**Tempo**
Última Semana	$1min : 49seg$
Último Mês	$3min : 13seg$
Últimos 3Meses	$11min : 26seg$

Tabela 7.6: Tempos de Carregamento Total em Intervalos Temporais.

Os resultados da tabela 7.6 permitem assim ao cliente saber que a atualização da *triple store* será uma tarefa de curta duração, não se prevendo portanto que tenham de interromper os seus serviços e projetos para que seja feita a atualização.

7.2 Validação dos Requisitos

Os testes aos requisitos foram uma das tarefas mais importantes de todo o projeto, pois o sucesso final do projeto dependia dos resultados obtidos nesta tarefa.

No levantamento de requisitos, secção 3.2, fez-se a distinção entre requisitos funcionais e não-funcionais, pelo que os testes realizados também têm esta divisão.

7.2.1 Requisitos Funcionais

Os requisitos funcionais do projeto centram-se essencialmente na apresentação da informação na plataforma OObian, pelo que os testes realizados, ao longo do desenvolvimento, consistiram em ir visualizando o resultado do mapeamento na plataforma OObian e efetuar diferentes pesquisas.

O grande teste a estes requisitos foi feito no final da fase de desenvolvimento, quando se preparou uma demonstração para se fazer ao cliente. Esta demonstração consistiu numa história onde os atores eram as *personas*, identificadas na sub-secção 3.2.1, e os seus pedidos os diferentes requisitos.

Inicialmente foram demonstrados os requisitos de prioridades *MUST* e *SHOULD*, com o objetivo de validar de entrada os principais requisitos levantados. De seguida, de modo a potencializar as funcionalidades da plataforma OObian e do próprio produto, foram efetuadas pesquisas personalizadas e mostrada a distribuição de todos os equipamentos geograficamente, na funcionalidade de mapas.

Por último, foi o próprio cliente a navegar pela interface de visualização. O seu objetivo foi o de consultar instâncias que sabe serem fulcrais no cadastro da rede, como serviços e equipamentos específicos.

Requisitos Não Implementados

Em relação à lista de requisitos funcionais, apresentada na sub-secção 3.2.2, existem três que não foram implementados: diferentes níveis de localização geográfica de equipamentos, páginas de Internet e restrição de acesso.

O requisito de ter diferentes níveis de localização geográfica não pôde ser implementado pelo facto de a plataforma OObian ainda não se encontrar preparada para ter esta funcionalidade. Por seu lado, o requisito de páginas de Internet não foi implementado devido à equipa da PT Inovação responsável por fazer estas páginas ainda não as ter terminado. Por fim, não se restringiu acesso por não ser possível distinguir diferentes utilizadores para uma mesma ontologia. Uma possível solução seria criar uma ontologia para cada *persona*, mas essa abordagem implicaria criar cinco ontologias diferentes e efetuar cinco carregamentos totais. Ou seja, seria necessário um mês só para se fazer todos os carregamentos totais, e como não conseguimos ter este tempo disponível, optámos por apenas construir uma ontologia mas ser possível apresentar ao cliente todos os requisitos de prioridade máxima a funcionar corretamente. Ao apresentar esta justificação ao cliente, aquando da demonstração, este não revelou qualquer preocupação, sendo da opinião que o importante para ele, nesta fase, é a de ter um produto que apresente a informação de um modo concetual e, consequentemente, que ajude e melhore os seus colaboradores nas suas tarefas. É do nosso conhecimento, e do cliente, que mais tarde há questões de segurança a endereçar, mas devido a essas questões terem de seguir certas políticas da PT Inovação, às quais não temos acesso, para o projeto em questão não se torna relevante ter o requisito "restrição de acesso" implementado.

Uma vez que nenhum dos requisitos tem prioridade máxima, dois primeiros têm prioridade *NICE* e o terceiro prioridade *SHOULD*, e como o facto de não terem sido implementados não influenciou o produto final corresponder

ao pretendido pelo cliente, este não mostrou qualquer descontentamento nem entrave perante a ausência destes requisitos.

7.2.2 Requisitos Não-Funcionais

Dos diferentes grupos de requisitos não funcionais apresentados na sub-secção 3.2.2, apenas os grupos de usabilidade, performance e suporte são passíveis de testes, pois os restantes quatro grupos, Design, Implementação, Interface e Físicos, tiveram de ser cumpridos para o correto desenvolvimento do produto final. Por exemplo, o requisito de design "base de dados relacional Oracle" foi cumprido uma vez que era este o motor de base de dados da PT Inovação; noutro sentido, sem um "cliente OObian", requisito de interface, não seria possível visualizar a informação.

Requisitos de Suporte

Os testes efetuados ao requisito "processamento de informação" podem-se dizer que foram feitos ao longo do tempo, pois em todas as pesquisas realizadas, quer nas diferentes fases de testes quer na demonstração ao cliente, foi sempre necessário testar que a informação pretendida era processada.

Quanto ao requisito "integração com sistemas BSS", este não pôde ser implementado por dois motivos: o modelo de negócio da PT Inovação, além de complexo, encontra-se em constante atualização, e existe informação confidencial que não nos pode ser revelada. Uma vez que a própria PT Inovação já tinha conhecimento destes dois pontos, a prioridade do requisito foi classificada de *NICE*, prioridade mais baixa, pelo que não é impeditivo de o projeto atingir o sucesso pretendido.

Requisito de Performance

Na secção 1.4 afirmámos que uma das características mais importantes para o cliente é a performance. O requisito pretendido é que o "tempo de resposta" de todos os requisitos funcionais, de prioridade *MUST* e *SHOULD*, se situe entre cinco e dez segundos, no pior caso. Ao verificar-se esta condição, poderemos considerar a performance de positiva.

O cenário ideal para testar a performance seria um que fosse o mais próximo possível do cenário real, ou seja, haver aproximadamente cinquenta utilizadores a fazer várias sessões de testes, todos eles com máquinas com os

mesmos recursos, memória RAM, processador, etc. Cada uma destas sessões consistiria em que todos os utilizadores fizessem cerca de vinte a trinta pedidos/pesquisas em simultâneo, e de seguida eram recolhidos os tempos de processamento de cada pedido e calculados os valores de médias e desvios-padrões. Devido aos mecanismos de *cache* feitos pelo hardware onde se encontra o servidor, os pedidos feitos a partir da segunda vez seriam excluídos, por desestabilizarem os valores reais, e no final de cada sessão seria necessário limpar a *cache*. O número de sessões a realizar não pode ser definido, pois teriam de ser feitas tanto quanto necessárias. O critério para decidir quando parar seria a partir do momento em que os valores das médias e dos desvios-padrões começassem a estabilizar.

Tendo já o nosso cliente uma disponibilidade muito reduzida, ainda mais difícil seria, ou mesmo impossível, termos cinquenta utilizadores disponíveis, pelo que a abordagem seguida foi a de criar um cenário com um conjunto mais reduzido de utilizadores que pudessem testar o produto em simultâneo. Infelizmente, a disponibilidade destes utilizadores também não é a ideal, tendo disponibilizado aproximadamente trinta minutos do seu tempo, pelo que apenas conseguimos fazer uma sessão de testes.

O nosso cenário consistiu em ter quatro utilizadores, cada um na sua máquina, a navegar pela plataforma e a fazer diferentes pedidos. Os valores obtidos foram tratados e são apresentados de seguida, estando a sua análise no próximo capítulo. De referir que foram excluídos os valores dos tempos de processamento de pedidos feitos por mais do que uma vez, devido ao fator de *cache* mencionado anteriormente, e que os resultados apresentados dizem respeito a instâncias com algumas dezenas de relações.

Inicialmente, na tabela 7.7, apresentamos os valores dos tempos de processamento, em milissegundos, para cada uma das classes de mapeamento.

	Clientes Empresariais	**Clientes Residenciais**	**Serviços de Cliente**	**Serviços de Rede**	**Nodes**	**TPs**	**Pipes**
Nº Pesquisas	17	3	23	21	25	24	20
Média	$42,71ms$	$28,33ms$	$51,61ms$	$38,05ms$	$31,72ms$	$37,46ms$	$27,25ms$
Desvio-Padrão	$22,91ms$	$3,06ms$	$18,52ms$	$22,84ms$	$20,75ms$	$16,60ms$	$9,26ms$

Tabela 7.7: Tempos de Processamento por Classe de Mapeamento.

Tendo em conta que na base de dados apenas existem seis clientes residenciais, o número de pesquisas feitas nesta classe é muito reduzido, comparativamente às outras classes. Observando a tabela 7.7, podemos constar que o tempo médio de processamento mais alto é de $51,61ms$, na classe Serviços de Cliente, e que o maior desvio-padrão pertence à classe Clientes Empresariais,

sendo de $22,91ms$.

Com o objetivo de visualizarmos os resultados graficamente, a figura 7.1 ilustra os valores da tabela 7.7 num gráfico.

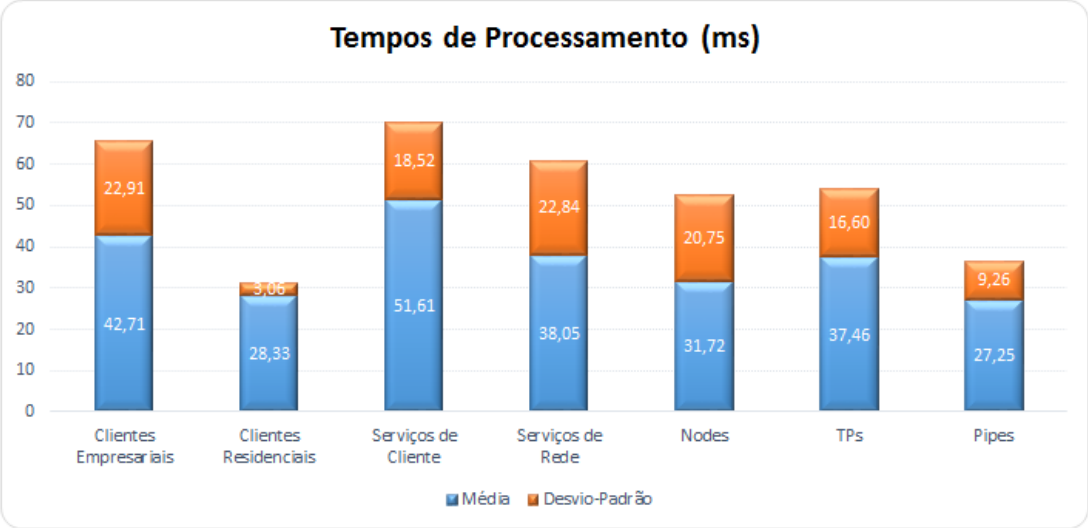


Figura 7.1: Tempos de Processamento por Classe de Mapeamento.

A figura 7.1 permite-nos observar a junção da média e do desvio-padrão para cada classe, sendo possível concluir que a sua soma, em cada classe, não chega a $0,80ms$.

De seguida agrupámos os resultados por grupos de classes e na sua totalidade, estando o resultado deste agrupamento na tabela 7.8.

	Clientes	**Serviços**	**Recursos**	**Total**
Nº Pesquisas	20	44	69	133
Média	$40,55ms$	$45,14ms$	$32,42ms$	$37,85ms$
Desvio-Padrão	$19,40ms$	$21,57ms$	$16,92ms$	$19,68ms$

Tabela 7.8: Tempos de Processamento por Grupos de Classes.

Ao agrupar as classes da tabela 7.7 pela sua classe "mãe" na hierarquia, consequentemente, o número de pedidos das amostras aumentou. No entanto, observando a tabela 7.8, é possível verificar que os valores das médias se mantêm inferiores a $50,00ms$ e que os desvios-padrões rondam os $20ms$.

Relativamente à última coluna da tabela 7.8, esta indica que juntando todos os pedidos feitos durante a sessão de testes, também os valores da média e do desvio-padrão não sofrem grandes alterações.

À semelhança da tabela 7.7, apresentamos agora o gráfico para os valores da tabela 7.8.

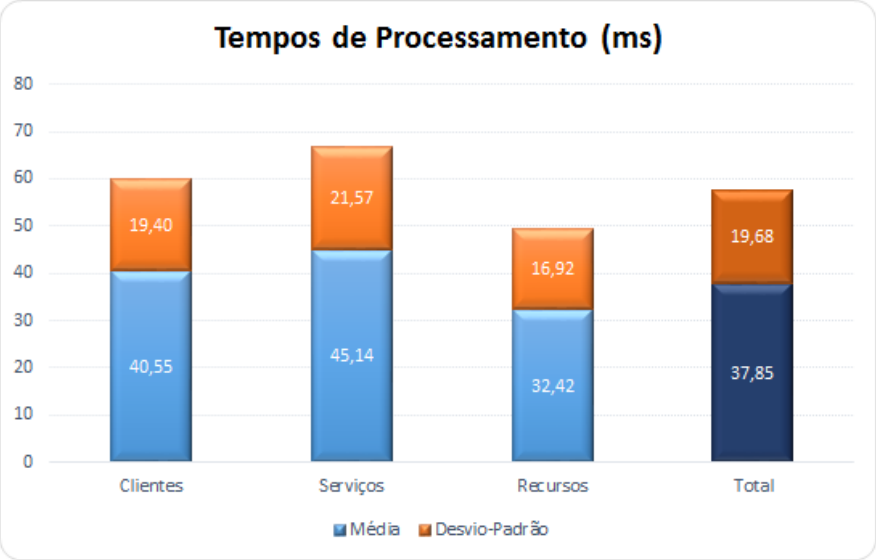


Figura 7.2: Tempos de Processamento por Grupo de Classes.

A figura 7.2 é interessante de analisar comparando-a com a figura 7.1, pois é possível observar que, no pior caso, a soma da média com o desvio-padrão fica ligeiramente abaixo dos $70ms$, o que não se verifica na figura 7.1, onde o valor desta soma é ligeiramente superior a $70ms$. Outra ilação que se pode tirar deste gráfico, é que de um modo geral, coluna Total, esta soma chega mesmo a ser inferior a $60ms$.

Por fim, de modo a não desestabilizar os valores apresentados anteriormente, fizemos um estudo à parte para pesquisas que incidam sobre instâncias com relações na ordem das centenas. A tabela 7.9 contém esses resultados.

	Serviços de Cliente	**Nodes**
Nº Pesquisas	9	8
Média	$125,00ms$	$180,63ms$
Desvio-Padrão	$23,80ms$	$46,05ms$

Tabela 7.9: Tempos de Processamento em Instâncias com Centenas de Relações.

Os resultados da tabela 7.9, assim como o gráfico da figura 7.3, demonstram que instâncias com relações na casa das centenas têm um tempo médio superior a instâncias com relações na casa das dezenas. Comparando estes valores com os do caso geral, coluna Total da tabela 7.8, temos que o tempo médio nos serviços de cliente é aproximadamente três vezes superior, e que no caso dos nodes é quase cinco vezes maior.

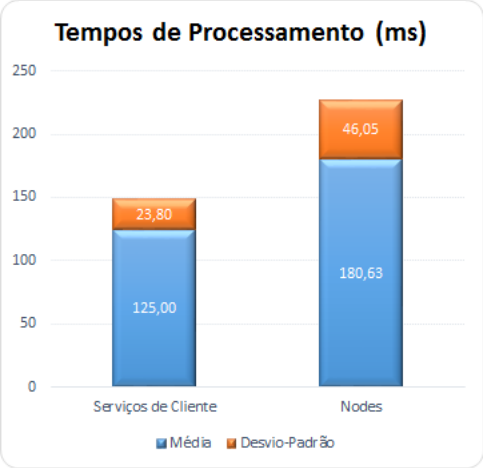


Figura 7.3: Tempos de Processamento em Instâncias com Centenas de Relações.

Quanto ao desvio-padrão, o gráfico da figura 7.3 ajuda a perceber que este não sofre grandes alterações relativamente ao caso geral de processamento.

Os resultados recolhidos, e que acabámos de apresentar, dizem respeito ao tempo de processamento de cada pedido, sendo estes valores que podemos controlar. Uma vez que no produto nenhum resultado é apresentado com uma rapidez de, por exemplo, $50,00ms$, recolhemos também valores para o tempo que os resultados demoram a aparecer na plataforma. Contudo, é importante ter em mente que estes valores não estão no nosso controlo, pois a máquina

de um utilizador, assim como a quantidade de processos em execução, são variáveis que podem alterar o tempo de apresentação da informação.

Portanto, na sessão de testes realizada, em média, uma pesquisa demora cerca de dois a três segundos a ser apresentada ao utilizador. Por outro lado, devido à grande quantidade de registos mapeados (ligeiramente mais que dois milhões), o campo de filtragens demora entre três e quatro segundos a ficar disponível.

Requisitos de Usabilidade

Por último, temos o par de requisitos de usabilidade eficácia-eficiência. De modo a se classificar o produto de eficaz, é necessário que os requisitos funcionais de prioridade *MUST* e *SHOULD*, e que sejam perguntas de *personas*, se encontrem implementados na sua totalidade. Relativamente ao segundo requisito de usabilidade, eficiência, este encontra-se diretamente ligado ao requisito "tempo de resposta". Ou seja, se este requisito de performance se encontrar cumprido para todos os requisitos funcionais, poderemos igualmente classificar o produto como eficiente.

Portanto, os testes feitos a este par de requisitos foram feitos ao mesmo tempo que se testaram os requisitos funcionais e o requisito de performance, respetivamente.

Capítulo 8

Análise Crítica e Retrospetiva

A nossa análise crítica e retrospetiva incide sobre os resultados obtidos durante as fases de teste apresentadas anteriormente, pelo que a estrutura deste capítulo é idêntica ao do capítulo anterior no que às secções diz respeito.

Carregamento da Informação

O nosso primeiro teste, com vista a diminuir o tempo de carregamento total, centrou-se em tentar otimizar as *queries* SQL realizadas para recolha dos dados da base de dados. Como demonstrado na secção 7.1, conseguimos otimizar com sucesso a *query* que mais tempo demorava, tendo a sua otimização resultado numa redução de $6,80seg$ para $0,43seg$. Devido ao grande volume de registos que faz uso desta *query*, os 482.603 equipamentos, o resultado obtido com esta otimização permitiu-nos desde logo concluir que o tempo do carregamento total podia ser substancialmente reduzido.

De seguida, os nossos testes focaram-se em melhorar o cenário em que o carregamento é feito. Os valores presentes na tabela 7.2 indicam que o segundo cenário é ligeiramente mais rápido do que o primeiro, aproximadamente $10min$, mas sem apresentar melhorias substanciais, como por exemplo, reduzir o tempo de carregamento para metade do tempo.

Uma vez que a mudança não proporcionou grandes melhorias, a nossa abordagem foi a de aumentar os recursos disponíveis, memória RAM e *page size*. Perante os resultados obtidos na tabela 7.3, é possível constatar que o KDIS, responsável por aceder à base de dados, fazer o mapeamento dos dados e enviar o resultado para o servidor OObian, mantém a performance quer tenha pouca ou muita memória disponível para executar; por seu lado, o servidor consegue processar de igual modo blocos de 200, 2.500 e 20.000 registos.

Terminados os testes nas *queries* SQL e nos cenários de carregamento, procedemos a um novo carregamento total. Desta vez, o carregamento total demorou seis dias, o que representa uma redução de $78,77\%$ em relação ao tempo do primeiro carregamento. Tendo em conta os resultados obtidos durante as otimizações, apontamos a otimização das *queries* como o principal fator para esta redução do tempo de carregamento.

Portanto, chegámos à conclusão que o principal *bottleneck* no tempo de carregamento encontra-se na base de dados da PT Inovação e não nas ferramentas da Maisis, ou seja, a própria estrutura da base de dados, assim como o grande volume de dados a carregar (aproximadamente dois milhões de registos), resultam num carregamento total com uma duração de aproximadamente seis dias.

Para reduzir este tempo de carregamento, existem duas possíveis soluções. A primeira passa por analisar o modelo de dados da base de dados e estudar uma possível otimização, como por exemplo, normalização de tabelas, pois permite eliminar atributos que sejam de áreas diferentes e se encontrem numa mesma tabela, assim como eliminar repetições desnecessárias de dados. Quanto à segunda solução, melhorar os recursos da máquina onde se encontre a base de dados, como memória RAM, processador e disco rígido, também pode ajudar a reduzir o tempo de acesso à base de dados.

Requisitos

Começando pelos requisitos não-funcionais, nomeadamente, por analisar o requisito de suporte, podemos afirmar que o "processamento de informação" é um requisito cumprido com sucesso, pois em todas as pesquisas efetuadas, quer nas diferentes fases de testes quer na demonstração ao cliente, a informação é processada e apresentada de acordo com o esperado.

Quanto ao requisito de performance, "tempo de resposta", as tabelas e gráficos apresentados na secção 7.2 indicam que os diferentes tempos de processamento não chegam a um décimo de segundo em instâncias com relacionamentos na casa das dezenas, e que para instâncias com centenas de relações o seu tempo de processamento é inferior a um quinto de segundo. Portanto, estes tempos de resposta ficam assim bastante abaixo do intervalo definido pelo cliente neste requisito, entre cinco e dez segundos no pior caso.

Outra abordagem que podemos analisar, é a de somar os tempos de processamento com os tempos de apresentação dos resultados de uma pesquisa. Em média, uma pesquisa leva dois a três segundos a ser apresentada, ou seja, na pior das situações, a soma destes tempos é de $3,20$ segundos, o que significa que mesmo assim não se chega a entrar no intervalo definido. Contudo,

caso se pretenda ainda assim reduzir este tempo, melhorando os recursos quer da máquina onde se encontra o servidor quer da máquina onde se acede à plataforma OObian, este poderá ser igualmente reduzido.

Perante estes resultados, afirmamos que cumprimos este requisito na sua totalidade, e com bastante sucesso.

Concluindo a análise crítica dos resultados obtidos para os requisitos não-funcionais, uma vez que o único requisito funcional não implementado foi o de "restrição de acesso", que não se enquadra numa pergunta das *personas*, podemos afirmar que o produto final é eficaz por responder a todas as perguntas das *personas*. Relativamente ao segundo requisito de usabilidade, eficiência, este encontra-se diretamente ligado ao requisito "tempo de resposta". Como este requisito de performance se encontra cumprido, podemos igualmente classificar o produto como eficiente .

Os nossos grandes objetivos eram dois: efetuar o mapeamento dos dados da base de dados para a plataforma OObian, e garantir performance no tempo de resposta.

Ao longo da demonstração, o cliente mostrou-se sempre satisfeito com o resultado final, sendo que o ponto alto da sua validação dos requisitos funcionais foi quando o próprio navegou pela plataforma e afirmou que o produto correspondeu às suas expectativas, pelo que o primeiro objetivo estava cumprido. Após os testes de performance, o tempo médio para se responder a um pedido nunca entra no intervalo estipulado pelo cliente, ou seja, também o segundo objetivo foi alcançado.

Estando os dois grandes objetivos cumpridos, estamos em condições de afirmar que o projeto realizado atingiu o sucesso pretendido.

Capítulo 9

Trabalho Futuro

A identificação do trabalho que pode ser realizado no futuro foi uma tarefa que foi feita ao longo do desenvolvimento, uma vez que grande parte deste trabalho incide sobre as ferramentas da Maisis que foram utilizadas: o KDIS e a plataforma OObian.

Como referido no final da secção 1.2 - Objetivos, o pretendido era identificar lacunas e oportunidades de melhoria nestas ferramentas, para que projetos futuros consigam evitar os problemas apresentados de seguida.

Começando pela ferramenta de mapeamento, o **KDIS**, foram identificados os seguintes aspetos para implementar no futuro:

- Permitir computação matemática no mapeamento, através da aplicação de operadores matemáticos aos resultados vindos nas *queries*, de modo a ser possível fazer cálculos numéricos;
- Aumentar a quantidade de formatos de data e/ou hora aceites, pois apenas aceita três: "yyyy-MM-dd", "yyyy-MM-dd'T'HH:mm:ss"e "HH:mm:ss". Há pelo menos duas formas de atingir este objetivo: ou adicionando programaticamente novos formatos, ou fazendo transformações sintáticas à entrada para converter os dados num formato processável;
- Devido à grande quantidade de registos que tiveram de ser carregados, pouco mais de dois milhões, o mapeamento teve de ser divido em vários blocos de registos, tal como apresentado e justificado na secção 6.2. De modo a evitar esta divisão, e todo o trabalho adjacente, a solução ideal será o KServer contar os registos das várias *queries*, para saber quanta informação vai carregar, dividir em blocos iguais, consoante

a memória da máquina onde esteja a ser processado o mapeamento, e fazer o mapeamento desses blocos. Com esta funcionalidade, a equipa de desenvolvimento apenas precisa de configurar um bloco de mapeamento para cada *query*, o que simplifica e acelera a criação do ficheiro .XML

Relativamente à **plataforma OObian**, a oportunidade de melhoria identificada vai ao encontro da apresentação da informação:

- Neste momento, quando se tem várias instâncias do mesmo tipo, como por exemplo portos, estas são todas agrupadas num único "bolo", sendo que este pode ter diferentes "fatias" para distinguir portos entre si. A figura 9.1 ilustra este caso.

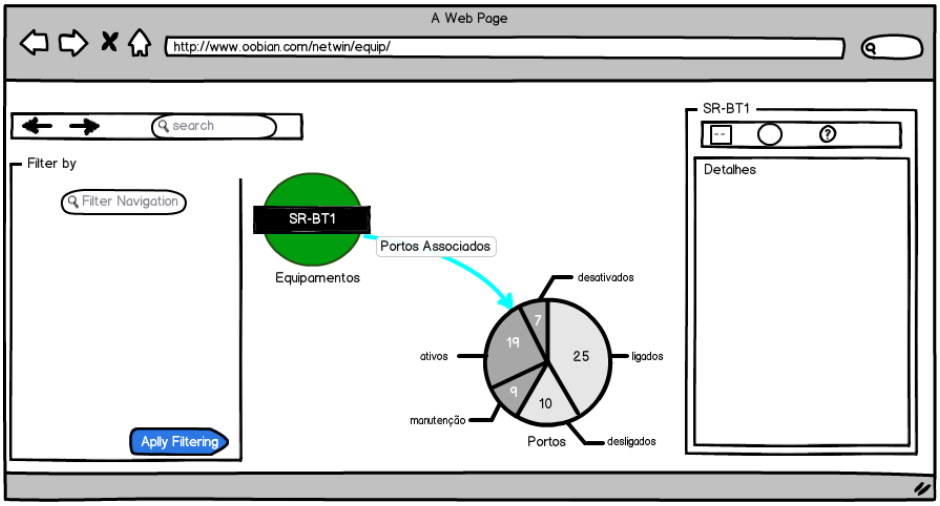


Figura 9.1: Plataforma OObian Atualmente.

Como se pode observar na figura 9.1, cada uma das fatias simboliza um determinado conjunto de portos: portos com ligação estabelecida, portos ativos, portos em manutenção, entre outros.

No futuro, idealmente, deveria ser permitido dividir este bolo em diferentes conjuntos, como por exemplo, ter um bolo que representa os portos agrupados pelo seu estado e ter outro que agrupa segundo a sua conectividade. Para perceber melhor o que se pretende, de seguida é apresentada a figura 9.2.

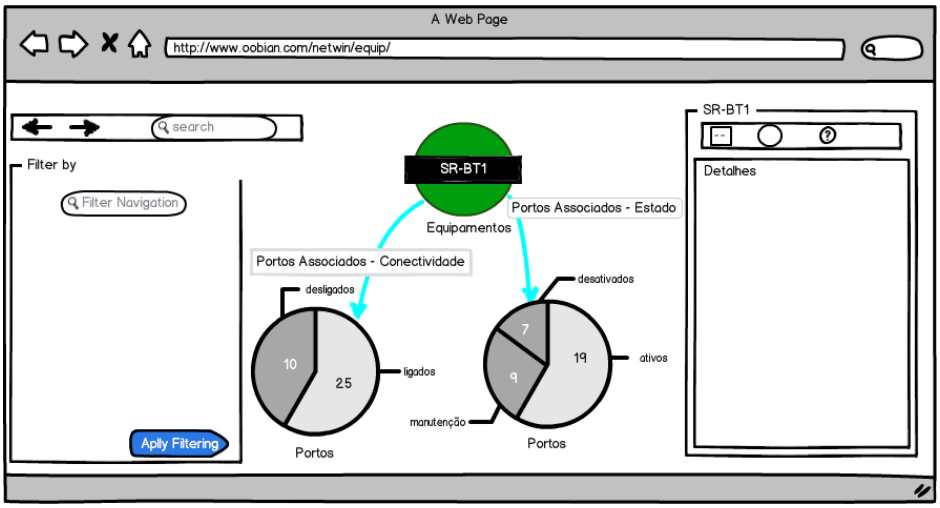


Figura 9.2: Plataforma OObian Futuramente.

> Portanto, observando a figura 9.2 e comparando-a com a figura 9.1, agora temos dois bolos em vez de um para representar os portos que estão ligados a um determinado equipamento. Por um lado, estão os portos agrupados consoante o seu estado (ativo, manutenção, etc.), e por outro, temos os portos agrupados pela sua conectividade (os que estão ligados e os que não estão).
>
> Com esta melhoria, pretende-se dividir a informação de modo a não ter conjuntos sobrecarregados, e a torná-la mais percetível para o utilizador.

De mencionar que o caso prático em que este ponto foi identificado não diz respeito ao desenvolvimento feito neste projeto, mas sim a outros projetos da Maisis. Contudo, esta oportunidade de melhoria surgiu aquando de uma reunião com a Maisis e a PT Inovação, numa conversa sobre a melhor maneira de apresentar os portos que estão ligados a um equipamento.

Voltando à secção 6.1 do capítulo Desenvolvimento, mais concretamente, à descrição da figura 6.3, foi mencionado que existem propriedades, pré-definidas pela Maisis, que têm de seguir uma determinada taxonomia caso se queira integrá-las numa **ontologia**.

Juntamente com as propriedades apresentadas, existem outras, também pré-definidas pela Maisis, para representar imagens, vídeos, endereços Web,

entre outras, pelo que, futuramente, o ideal será criar uma ontologia com todas essas propriedades, e integrá-la com a ontologia que for criada em cada projeto. Com a criação desta ontologia, pretende-se evitar realizar trabalho igual em todos os projetos, assim como simplificar a criação da nova ontologia.

Por fim, podem vir a implementar-se os três requisitos funcionais que ficaram em falta: diferentes níveis de localização geográfica de equipamentos, páginas de Internet e restrição de acesso. No capítulo 7 encontra-se a justificação pela qual não se implementaram estes requisitos, sendo que em anexo, no apêndice C, está a sua apresentação detalhada.

Ferramenta para Mapeamento

Noutro sentido, a Maisis pode proceder à criação de uma ferramenta, com um interface gráfico próprio, para ajudar na tarefa de mapeamento. Esta ferramenta tem de ser capaz de comunicar com bases de dados e instanciar automaticamente os dados, dados esses que serão introduzidos pelo utilizador na interface da ferramenta. O utilizador tem de ter igualmente a possibilidade de definir os relacionamentos, *Object Properties*, e os atributos, *Datatype Properties*, desses dados. O resultado final pretende-se que seja o ficheiro .XML configurado por completo e pronto a ser executado pelo KDIS.

De seguida, nas figuras 9.3 e 9.4, é apresentada uma proposta de interface gráfico para essa ferramenta.

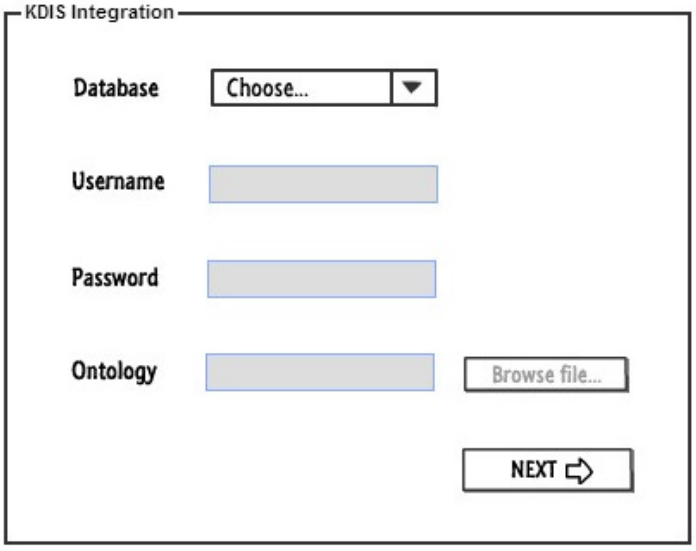


Figura 9.3: Ferramenta de Mapeamento - Início.

A figura 9.3 é um protótipo da página inicial da ferramenta, onde se pede ao utilizador para indicar o tipo de base de dados a usar (Oracle, MySQL, etc.), os dados de acesso a essa base de dados, e para carregar o ficheiro com a ontologia. Assim que os campos estiverem preenchidos, avança-se na interface para a página de mapeamento, figura 9.4.

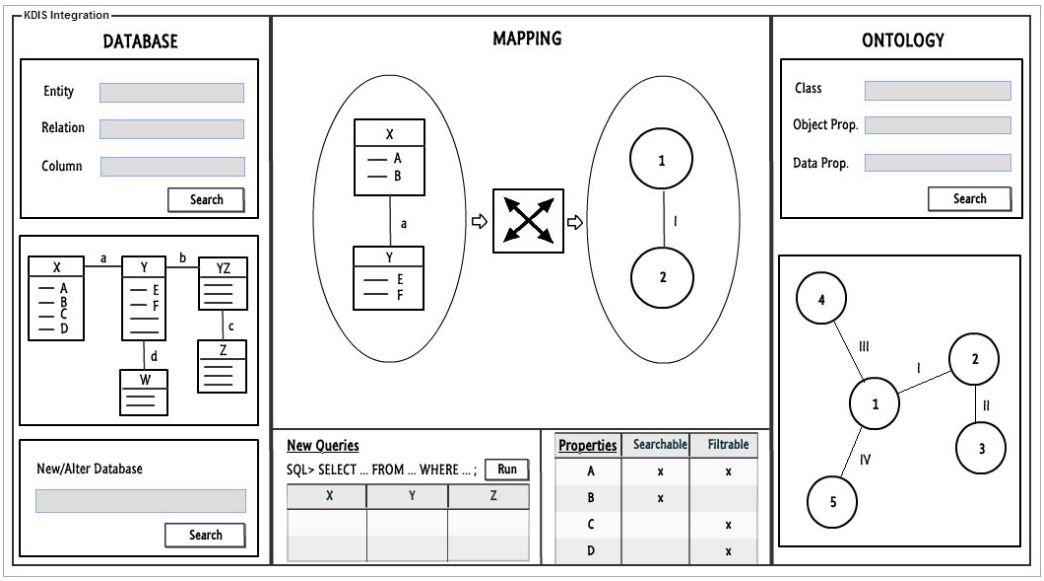


Figura 9.4: Ferramenta de Mapeamento - Mapeamento.

O interface de mapeamento apresentado na figura 9.4 pode ser divido em três importantes zonas: base de dados, mapeamento e ontologia. No lado esquerdo será possível definir as tabelas, as relações e os atributos da base de dados que se pretendem mapear; no lado esquerdo especificam-se as classes e as propriedades da ontologia para onde os dados serão mapeados; e no centro da figura será feita a configuração do mapeamento.

Outras possíveis funcionalidades a constar no interface podem ser:

- Selecionar uma base de dados diferente de modo a ser possível mapear informação de diferentes bases de dados;
- Criar atributos através de novas *queries*;
- Definir se as propriedades da ontologia são pesquisáveis e/ou filtráveis.

O documento presente no apêndice D - Metodologia Genérica de Trabalho, além da metodologia de trabalho apresentada, pode servir de auxílio no

desenvolvimento da referida ferramenta, no sentido em que o programador pode consultar a estrutura do ficheiro .XML que a ferramenta deve gerar, após inserção dos dados por parte do utilizador.

Novos Cálculos de Custos

Uma tarefa importante de se fazer, agora por parte da PT Inovação, é a de dar um período de adaptação para que os seus operadores de call-center se familiarizem com a plataforma OObian, com o objetivo de fazer uma efetiva comparação de custos entre a fase "Pré-OObian" e a fase "Pós-OObian".

Uma sugestão é a de todas as semanas recolher médias do número de chamadas atendidas, assim como da duração de cada uma delas. Nas primeiras semanas, estes valores podem ter oscilações significativas, mas é de esperar que ao fim de algumas semanas estes valores já não tenham uma grande diferença entre si. A partir desta altura, pode-se considerar que se atingiu a fase "Pós-OObian".

Após terminar este período de adaptação, deve-se voltar a efetuar os cálculos do apêndice B para se apurar ao certo quanto é que a PT Inovação estará a lucrar ao fazer uso da plataforma OObian para consulta da informação.

Bibliografia

[1] Allen, D. *Getting Things Done. The Art of Stress-Free Productivity.* Penguin, 2003. ISBN-10: 0142000280 ISBN-13: 978-0142000281.

[2] Alves, D., and Macedo, R. *Agente de Integração de Dados - Concepção/Especificação.* Maisis, November 2009.

[3] APCC. Estudo Diagnóstico e Benchmarking. `http://www.apcontactcenters.com/estudo/estudo%202012.pdf`, 2012.

[4] Bell, D. UML basics: The component diagram. `http://www.ibm.com/developerworks/rational/library/dec04/bell/`, December 2004. IBM Corporation. [Ativo em 28/01/2013].

[5] Bizer, C., and Cyganiak, R. D2RQ - Lessons Learned. *W3C Workshop on RDF Access to Relational Databases* (September 2007).

[6] Bizer, C., and Schultz, A. The berlin sparql benchmark. *International Journal On Semantic Web and Information Systems* (2009).

[7] Bizer, C., and Seaborne, A. D2RQ - treating non-RDF databases as virtual RDF graphs, 2004.

[8] Bray, T., Paoli, J., Sperberg-Mcqueen, C. M., Eve, and Yergeau, F., Eds. *Extensible Markup Language (XML) 1.0*, fourth ed. W3C Recommendation. W3C, 2003.

[9] Chebotko, A., and Lu, S. *Querying the Semantic Web: An Efficient Approach Using Relational Databases.* Lambert Academic Publishing, 2009.

[10] Civili, C. Research Summary: Datalog-Based Data Access. In *RR* (2012), pp. 272–277.

[11] Das, S., Sundara, S., and Cyganiak, R. R2RML: RDB to RDF Mapping Language. `http://www.w3.org/TR/r2rml`, September 2012.

[12] De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rosati, R., Ruzzi, M., and Savo, D. F. *MASTRO: A Reasoner for Effective Ontology-Based Data Access*. Rome, Italy.

[13] Dehmer, M. Information processing in complex networks: Graph entropy and information functionals. *Applied Mathematics and Computation* (2008).

[14] Eeles, P. Capturing Architectural Requirements. `http://www.ibm.com/developerworks/rational/library/4706.html`, 2005. [Ativo em 21/01/2013].

[15] Forum, T. Frameworx Metamodels - Concepts and Principles . http://www.tmforum.org/TechnicalReports/TR177FrameworxMetamodels/48552/artic 2012. [Ativo em 22/01/2013].

[16] Ghawi, R., and Cullot, N. *Database-to-Ontology Mapping Generation for Semantic Interoperability*. Vienna, Austria, 2007.

[17] Krötzsch, M., and Straccia, U., Eds. *Web Reasoning and Rule Systems - 6th International Conference, RR 2012, Vienna, Austria, September 10-12, 2012. Proceedings* (2012), vol. 7497 of *Lecture Notes in Computer Science*, Springer.

[18] Lines, M., and Ambler, S. *Disciplined Agile Delivery: A Practioner's Guide to Agile Software Delivery in the Enterprise*. IBM Press, 2012.

[19] McGuinness, D. L., and van Harmelen, F. OWL Web Ontology Language Overview. `http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.3`, 2004.

[20] Nielsen, J. Response Times: The 3 Important Limits. Technical report, Nielsen Norman Group, January 1993.

[21] Nielsen, J. 10 Usability Heuristics. Technical report, Nielsen Norman Group, January 1995.

[22] Scheir, P., Prettenhofer, P., Lindstaedt, S. N., and Ghidini, C. *Progressive Concepts for Semantic Web Evolution: Applications and Developments*. IGI Global, 2010.

[23] Shadbolt, N., Berners-Lee, T., and Hall, W. The Semantic Web Revisited. *IEEE Intelligent Systems 21*, 3 (May 2006), 96–101.

[24] Simkin, D., and Hastie, R. An Information-Processing Analysis of Graph Perception. *Journal of the American Statistical Association* (1987).

[25] Sommerville, I. *Software Engineering*, 9. ed. Addison-Wesley, Harlow, England, 2010, ch. 22.

[26] Spivack, N. Web 3.0: The Third Generation Web is Coming . http://lifeboat.com/ex/web.3.0. [Ativo em 22/01/2013].

[27] Villazón-Terrazas, B., and Hausenblas, M. RDB2RDF Implementation Report. `http://www.w3.org/2001/sw/rdb2rdf/implementation-report/`, August 2012.

Apêndice A

Planeamento

Planeamento da Primeira Fase

Este projeto de engenharia dividiu-se em duas fases coincidentes com os semestres letivos. A primeira fase (Setembro 2012 a Janeiro 2013) envolveu principalmente tarefas de análise, como estado da arte e análise de requisitos, e a segunda fase (Fevereiro a Junho 2013) focou-se no desenvolvimento do produto final, tendo sempre presente as decisões tomadas na primeira fase.

O diagrama de Gantt da primeira fase do projeto é apresentado na figura A.1, sendo que o planeamento da segunda fase do trabalho foi apresentado no capítulo 4.

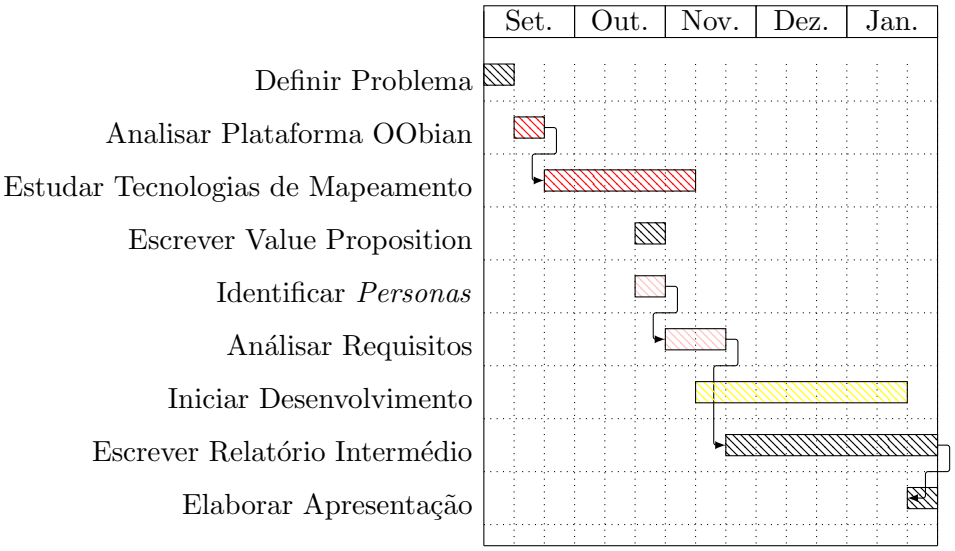


Figura A.1: Diagrama de Gantt da Primeira Fase

Analisando a figura A.1, facilmente se observa que determinadas tarefas foram coloridas, pois são grupos de tarefas que podem ser considerados como os mais importantes. A vermelho temos o estado da arte, a cor-de-rosa a análise de requisitos, e por fim, o desenvolvimento a amarelo.

A tarefa de desenvolvimento, não sendo uma tarefa esperada de se realizar na primeira fase, encontra-se presente uma vez que foi necessário efetuar um ligeiro desenvolvimento para uma correta ambientação com as soluções finais a usar, e também para ajudar no levantamento de alguns requisitos e na elaboração do planeamento de trabalho da segunda fase.

Planeamento da Segunda Fase

No início do presente apêndice foi mencionado que o planeamento da segunda fase do trabalho encontra-se detalhado no capítulo 4. Contudo, no final da primeira fase, foi projetado um plano de tarefas para a segunda fase, mas que na prática sofreu algumas alterações. Portanto, no capítulo 4 é possível consultar o plano de tarefas que foi cumprido e a explicação de cada tarefa, enquanto de seguida apresentamos somente o plano projetado.

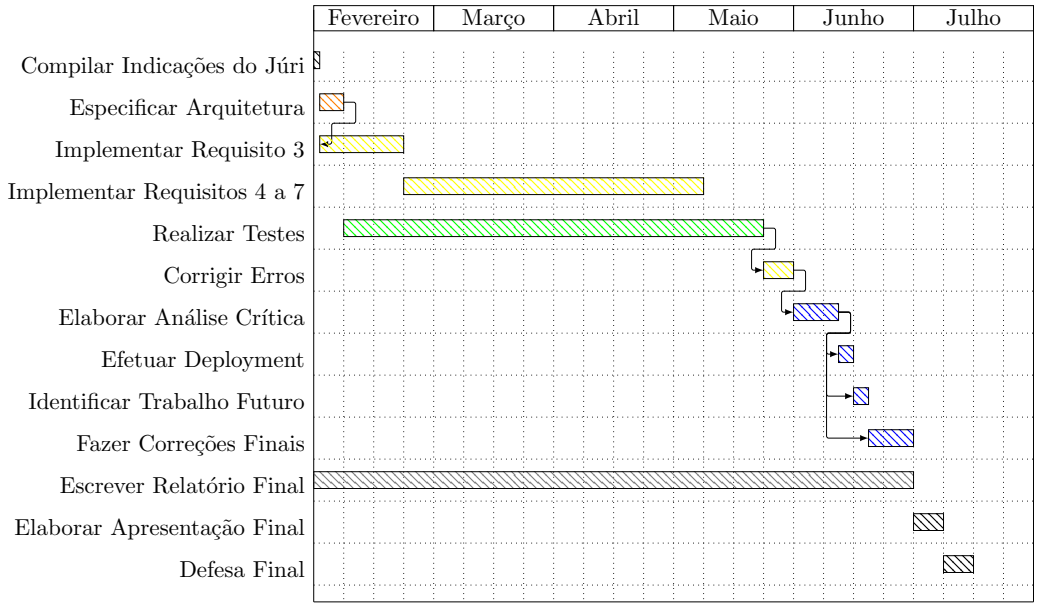


Figura A.2: Diagrama de Gantt da Segunda Fase (Projeção)

Apêndice B

Cálculos para Redução de Custos

Na secção 1.3 - Motivação, mencionou-se que uma das motivações da PT Inovação para este projeto se centra no aspeto financeiro, nomeadamente, redução de custos com operadores de call-center.

Os seguintes cálculos fornecem uma ideia de quais os valores que se podem reduzir com esses operadores:

- Duração média de uma chamada numa TELCO $\simeq 6:43 minutos$ [3];
- PT Contact [1], empresa responsável pelo call-center da PT Inovação, recebe em média 6 milhões de chamadas por mês;
- $6M chamadas/mes \simeq 200m/dia \simeq 8.333/hora$;
- Se uma chamada tem a duração média de $6:43 minutos$, então um operador atende em média $60 minutos/6:43 minutos \simeq 8 chamadas/hora$;
- Para responder a $8.333 chamadas/hora$ são então necessários $8.333/8 \simeq 1042 operadores/hora$;
- Havendo 3 turnos por dia, são necessários: $3*1042 = 3126 operadores/dia$;
- Tendo por base que um operador recebe $5€/hora$ e trabalha $8h/dia$, então ele recebe $5*8 = 40€/dia = 200€/semana = 800€/mes$;
- Portanto, os gastos com operadores são: $3126*800€ = 2.500.800€/mes = 30.009.600€/ano$.

[1] PT Contact - http://www.ptcontact.pt/ptcontact.htm

Caso se pretenda reduzir a duração média de uma chamada em $13 segundos$, para $6:30 minutos$, um número considerado "redondo", os gastos com operadores serão:

- Se $1 chamada$ durar em média $6:30 minutos$, um operador passará a responder a $60 minutos/6:30 minutos \simeq 9 chamadas/hora$;
- Para responder às $8.333 chamadas/hora$ já só serão necessários $8.333/9 \simeq 926 operadores/hora$, o que resulta num total de $2778 operadores/dia$;
- Ou seja, os gastos com operadores passarão para $2778 * 800€ = 2.222.400€/mes = 26.668.800€/ano$.

Calculando a diferença de gastos, a redução de custos pode chegar aos:

- $30.009.600€ - 26.668.800€ = 3.340.800€/ano$;
- O que significa que $1 segundo$ numa chamada tem um valor de $3.340.800€/13 segundos \simeq 256.984€$.

Apêndice C

Análise de Requisitos

De acordo com a metodologia FURPS+, de seguida são apresentados em maior detalhe os requisitos do projeto. Apresentamos para cada requisito funcional o respetivo mockup gráfico, para uma compreensão completa do produto pretendido, sendo que nalguns casos é apresentado um único mockup para um determinado conjunto de requisitos. Isto deve-se ao facto da interface da plataforma OObian permitir agrupar diversa informação sob um mesmo aspeto.

Relativamente aos mockups apresentados, de salientar que todos eles foram apresentados e validados pela PT Inovação, pelo que já é possível ter uma perceção de como a informação é apresentada no produto final.

A apresentação de cada requisito segue um template, criado pelo estagiário Luís Veiga, com o objetivo de ajudar na compreensão de cada requisito. O template construído é o seguinte:

<Nome do Requisito>

<ID> - número identificador do requisito.

<Prioridade> - classificação da importância do requisito segundo a tabela 3.2, presente no capítulo 3.

<Descrição> - texto a descrever o requisito, que no caso dos requisitos funcionais será acompanhado de mockups gráficos.

<Fonte> - stakeholder que identificou o requisito.

Requisitos de Funcionalidade

Uma vez que o produto final tem o propósito de ser um meio de consulta de informação por parte de colaboradores da PT Inovação, os requisitos funcionais incidem sobre o sucesso na consulta e na apresentação de determinada informação.

Efetuar pesquisas

ID - 1.
Prioridade - MUST.

Descrição - o grande propósito do produto final é o de fornecer informação aos utilizadores, pelo que um meio destes efetuarem pesquisas é de extrema importância para que tenham um acesso mais rápido à informação que pretendem obter. Está previsto existirem duas vias diferentes para se efetuar pesquisas, como se pode observar na figura C.2. Estas duas vias encontram-se ambas no lado esquerdo da página, sendo que na parte superior será dada a possibilidade aos utilizadores de introduzirem palavras-chave na caixa **search**, e na coluna imediatamente abaixo encontram-se as várias propriedades pela qual é possível filtrar de modo a somente consultar a informação pretendida.

Fonte - Maisis.

Sendo este requisito composto por ações que precedem e têm influência na visualização de informação, a figura C.1 pretende mostrar os casos de uso inerentes ao requisito de efetuar pesquisas.

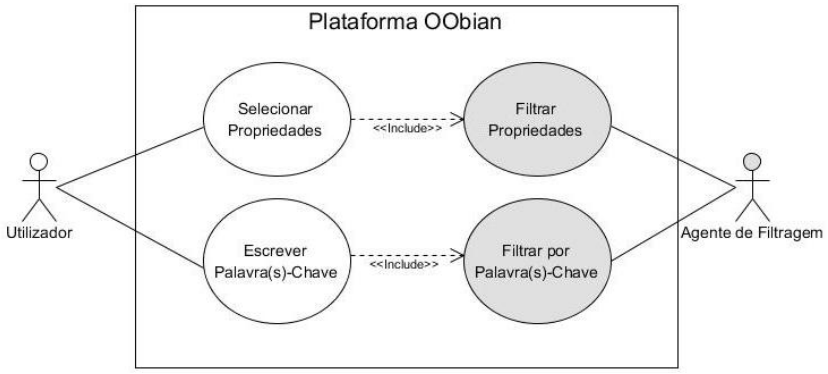


Figura C.1: Diagrama de Casos de Uso do Requisito 1.

Tal como mencionado na descrição, é possível fazer pesquisas filtrando propriedades ou pesquisando por palavras-chave, ações estas presentes do lado esquerdo da figura C.1. Do lado direito da imagem está presente o agente de filtragem, plataforma OObian, que é responsável por apresentar a informação pretendida e foi desenvolvida pela Maisis.

Distribuição de ocupação por equipamento

ID - 2.
Prioridade - MUST.

Descrição - como cada equipamento tem vários portos para se poder ligar a outros equipamentos, uma informação importante a apresentar é a quantidade desses portos que estão ligados e desligados.

Fonte - PT Inovação.

O mockup ilustrativo do presente requisito pode ser visualizado na figura C.2.

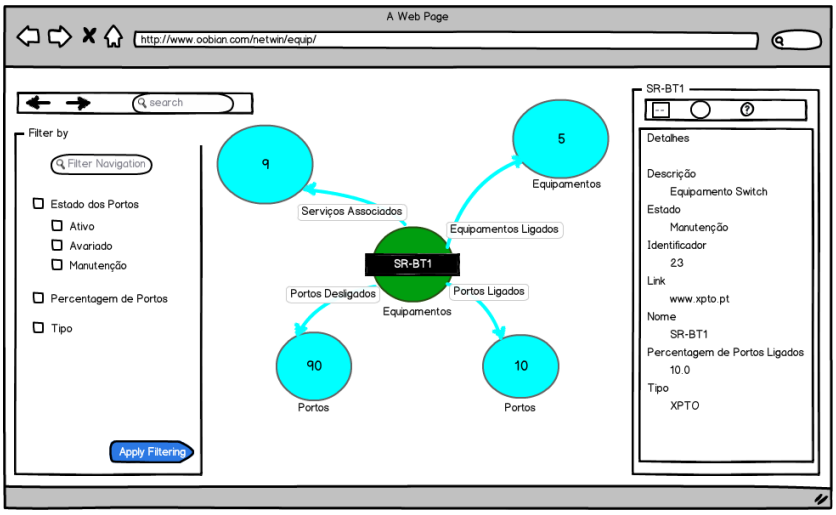


Figura C.2: Requisitos 1, 2, 13, 15, 17 a 19 e 22.

É importante salientar que a figura C.2 representa graficamente um determinado conjunto de requisitos, sendo eles os dois primeiros requisitos apresentados, cinco requisitos com prioridade SHOULD, requisitos 13, 15 e 17 a 19, e o requisito NICE número 22.

Ao dividir a imagem em três partes verticais, temos que no lado direito se encontram descritos textualmente os detalhes do equipamento em apresentação, do lado esquerdo os campos de filtragem e pesquisa (requisito 1), e por fim, no centro da figura, o grafo com todas as relações do equipamento. Nos detalhes do equipamento é possível consultar diversa informação, entre ela, estado do equipamento (requisito 17) e um link para a página Web do equipamento (requisito 22).

Relativamente ao grafo, na parte inferior é possível ver a distribuição de ocupação dos portos (requisito 2), e na parte superior estão presentes os serviços e os equipamentos associados (requisitos 15 e 19, respetivamente). A referência ao equipamento em análise é feita através do seu nome (requisito 13).

Por fim, nos campos de filtragem, é possível filtrar os portos pelo seu estado (requisito 18), sendo que a aplicação desse filtro mudará o número de portos apresentados no grafo.

Serviços por cliente

ID - 3.

Prioridade - MUST.

Descrição - ao consultar os dados de um cliente, juntamente com a informação de cada cliente, todos os tipos de serviço de que o cliente usufrui têm de ser visíveis.

Fonte - PT Inovação.

A figura C.3 ilustra os serviços que podem pertencer a um determinado cliente.

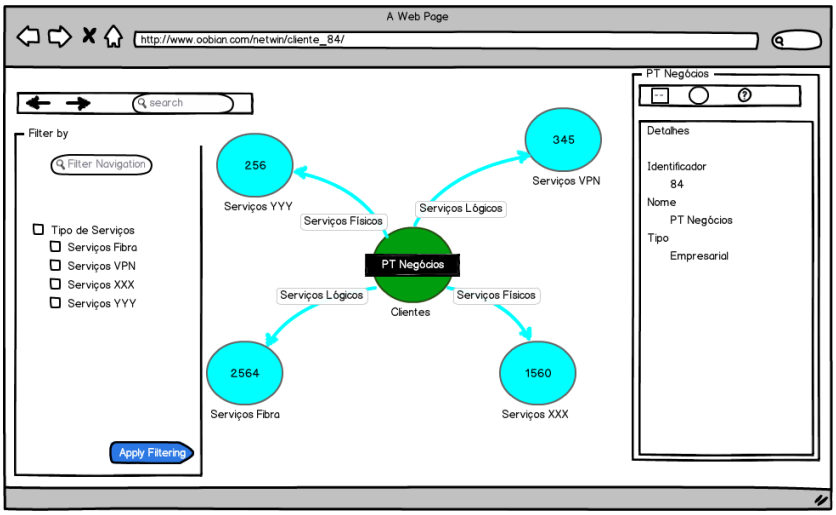


Figura C.3: Requisito 3, Serviços por cliente.

Observando a figura C.3, constata-se que estes serviços podem ser de dois tipos, físicos e lógicos, sendo que dentro destes tipos ainda existe outra classificação. No campo de filtragem é possível configurar quais destes tipos podem ser apresentados.

Neste ponto, e devido à existência de requisitos referentes a VPNs, é muito importante referir que uma VPN é considerada como um tipo de serviço.

Dependências de serviço

ID - 4.
Prioridade - MUST.

Descrição - a informação relativa a um determinado serviço, além da própria informação, deve incorporar aspetos adjacentes a este, como por exemplo: encaminhamento da rede, clientes que usam esse serviço, e serviços que tem associado.

Fonte - PT Inovação.

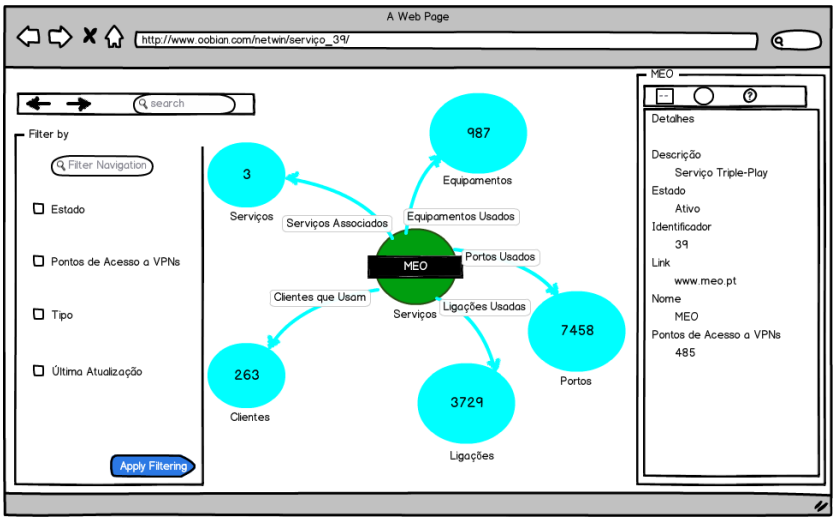


Figura C.4: Requisitos 4 a 7, 14 e 16.

O mockup anterior contém o resultado pretendido para apresentar as dependências de um determinado serviço, sendo que no centro temos o serviço em questão e a rodeá-lo estão as suas dependências.

Encaminhamento de rede, por serviço

ID - 5.
Prioridade - MUST.

Descrição - a informação a apresentar sobre o encaminhamento de rede de um determinado serviço centra-se nos diferentes recursos que um determinado serviço utiliza: equipamentos, portos e ligações.

Fonte - PT Inovação.

Pontos de acesso a VPNs

ID - 6.
Prioridade - MUST.

Descrição - um dos atributos a apresentar nos detalhes de um serviço, apenas se for um serviço do tipo VPN, é a quantidade de pontos de acesso que existem para se ligar ao serviço em questão.

Fonte - PT Inovação.

Equipamentos de suporte a VPNs

ID - 7.
Prioridade - MUST.

Descrição - sendo uma VPN um serviço, os equipamentos que lhe dão suporte têm de ser apresentados, tal como ilustra a figura C.4.

Fonte - PT Inovação.

Lista de clientes / serviços / equipamentos / portos / ligações

ID - 8 / 9 / 10 / 11 / 12 .
Prioridade - SHOULD.

Descrição - apresentar a lista de todos os clientes / serviços / equipamentos / portos / ligações existentes, de modo a ser possível selecionar um e consultar a sua informação.

Fonte - Luís Veiga.

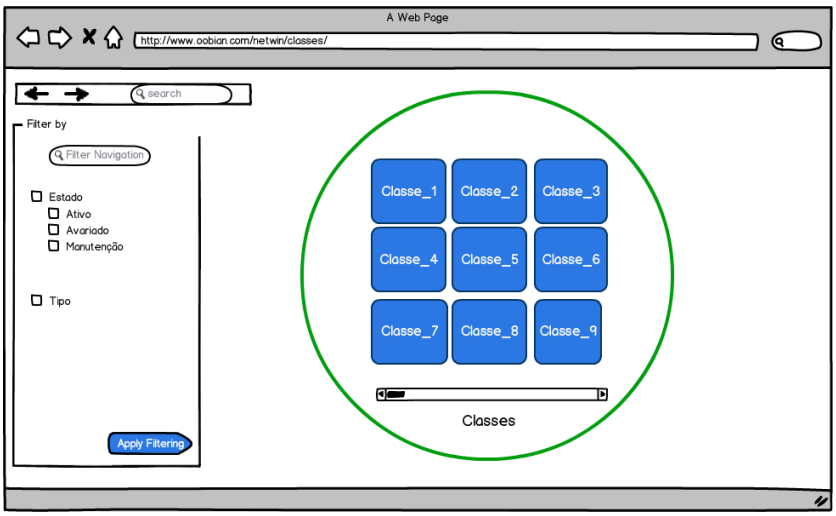


Figura C.5: Requisitos 8 a 12.

A figura C.5 é um mockup modelo que se adapta aos requisitos 8 a 12, uma vez que as diferentes listas apresentadas apenas diferem no nome, pois a apresentação é igual para todas. Mais uma vez, do lado esquerdo estão os campos de filtragem, sendo que variam de classe para classe. Quanto ao centro da imagem, desta vez não aparece um grafo mas sim um conjunto de elementos, sendo que este conjunto pode ser de clientes, serviços, equipamentos, portos ou ligações, formando assim a lista que se pretende consultar.

Nome dos registos na visualização

ID - 13.
Prioridade - SHOULD.

Descrição - aquando da apresentação das diferentes listas de instâncias (clientes, serviços e recursos), assim como de cada uma delas em específico, a identificação deve ser feita pelo seu nome. No caso dos portos existe uma exceção, pois a PT Inovação referiu que o nome a apresentar deve seguir o template 'Nome do Equipamento - Nome do Porto', ou seja, antes do nome do porto deve aparecer o nome do equipamento a que pertence.

Fonte - PT Inovação.

Nota: de lembrar que o mockup gráfico do presente requisito, assim como dos requisitos 15 e 17 a 19, foi apresentado anteriormente, na figura C.2.

Clientes associados

ID - 14.
Prioridade - SHOULD.

Descrição - apresentar os clientes que usam um determinado serviço. A implementação da funcionalidade a que corresponde este requisito pode ser importante para aferir da adesão dos clientes a esse serviço, ou, caso seja preciso suspendê-lo ou cancelá-lo, se afeta um grande número de clientes.

Fonte - Luís Veiga.

Nota: de lembrar que o mockup gráfico do presente requisito e do requisito 16 foi apresentado anteriormente, na figura C.4.

Serviços associados

ID - 15.
Prioridade - SHOULD.

Descrição - à semelhança do requisito anterior, é igualmente importante saber os serviços associados a um determinado serviço, bem como a determinado equipamento, sempre com o objetivo de estudar as dependências entre serviços e a carga de utilização dos equipamentos.

Fonte - Luís Veiga.

Estado dos serviços

ID - 16.
Prioridade - SHOULD.

Descrição - para cada serviço, têm de ser apresentados na lista de atributos os seus dois tipos de estado: operacional e de vida. Os valores de um estado operacional são ENABLE, DISABLE ou DISABLE.MAINTENANCE, enquanto o estado de vida de um serviço informa se esse serviço se encontra Ativo, Abortado, Em Provisão ou Terminado.

Fonte - PT Inovação.

Estado dos equipamentos

ID - 17.
Prioridade - SHOULD.

Descrição - uma das informações de cada equipamento deve ser o estado operacional em que se encontra: ENABLE, DISABLE ou DISABLE.MAINTENANCE. A aplicação deve permitir visualizar esta informação para cada equipamento.

Fonte - Luís Veiga.

Estado dos portos

ID - 18.
Prioridade - SHOULD.

Descrição - semelhante ao requisito anterior, pois tem de ser possível filtrar mas também apresentar no grafo os portos pelo seu estado operacional: ENABLE, DISABLE ou DISABLE.MAINTENANCE.

Fonte - Maisis.

Equipamentos ligados entre si

ID - 19.
Prioridade - SHOULD.

Descrição - juntamente com a distribuição de ocupação por equipamento e dos respetivos serviços associados, será muito interessante visualizar também os equipamentos a que um equipamento se encontra ligado. Este requisito permitirá, a um primeiro nível, mostrar se alterações a determinado equipamento têm uma grande influência noutros.

Fonte - Maisis e Luís Veiga.

Localização geográfica de equipamentos

ID - 20.
Prioridade - SHOULD.

Descrição - fazer uso da funcionalidade **Maps** da plataforma OObian para apresentar a localização geográfica de um determinado equipamento no mapa.

Fonte - Maisis.

Diferentes níveis de localização geográfica de equipamentos

ID - 21.
Prioridade - NICE.

Descrição - poder ver diferentes níveis de localização geográfica, ou seja, ver a localização dos equipamentos por diferentes áreas: concelho, distrito e região.

Fonte - Maisis.

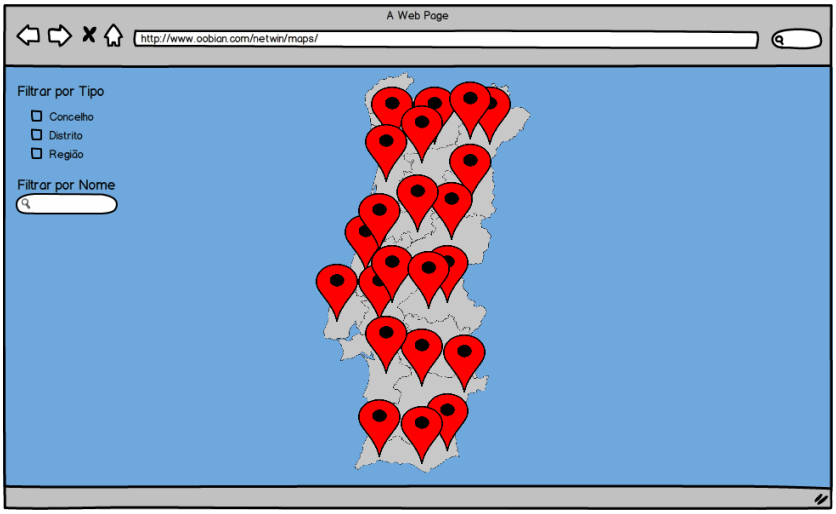


Figura C.6: Requisitos 20 e 21.

Ao analisar a figura C.6 pode-se observar que o interface difere das figuras anteriores, pois o objetivo agora passa por visualizar informação geograficamente. No canto superior esquerdo estão os campos de filtragem onde será possível definir qual o nível de localização pretendido, e no centro da figura aparece o mapa de Portugal Continental com os pontos onde existem equipamentos.

Páginas de Internet

ID - 22.
Prioridade - NICE.

Descrição - com o objetivo de fornecer mais informação aos utilizadores, um possível atributo de serviços e de equipamentos pode ser uma hiperligação (atributo **Link** das figuras C.2 e C.4) para uma página da Internet com uma maior especificação.

Fonte - Maisis.

Restrição de Acesso

ID - 23.
Prioridade - SHOULD.

Descrição - a informação apresentada a cada *persona* é limitada à estritamente necessária para dar resposta às suas perguntas, isto é, o acesso à *triple store* é restringido por *persona*. A sua prioridade não é máxima devido à limitação, apresentada pela Maisis, de não ser possível definir diferentes tipos de utilizadores aquando da criação da ontologia. Devido a esta contrapartida, a PT Inovação optou por baixar a prioridade deste requisito.

Fonte - PT Inovação e Maisis.

Requisitos de Usabilidade

Dois requisitos importantes de usabilidade são o par eficácia/eficiência, pois são sempre pontos importantes no sucesso de um projeto e que deixa os utilizadores satisfeitos. No que à interface de utilização diz respeito, não existem quaisquer requisitos uma vez que a plataforma OObian tem a sua própria interface.

Eficácia

ID - 24.
Prioridade - MUST.

Descrição - o produto final tem de ser capaz de responder a todas as perguntas pertencentes às *personas*, identificadas no capítulo 3.

Fonte - PT Inovação.

Eficiência

ID - 25.
Prioridade - MUST.

Descrição - recorrendo à sétima heurística de usabilidade de Nielsen [21], para um produto ser eficiente tem de ter um meio fácil e rápido para aceder à informação. No caso da plataforma OObian, este meio é a caixa de pesquisa ou os campos de filtragem que se encontram sempre presentes no interface. Relativamente à rapidez com que se obtém uma resposta, o tempo limite que um utilizador aceita esperar é de 10 segundos [20], o que vai ao encontro do valor que se pretende alcançar e indicado pelo cliente para obtenção de uma resposta (cinco a dez segundos).

Fonte - PT Inovação.

Requisitos de Confiabilidade

Os requisitos que se enquadram neste conjunto são requisitos que fogem ao âmbito do projeto, pois questões relacionadas com disponibilidade do sistema, ou com a capacidade para recuperar de falhas, são aspetos controlados pela Maisis uma vez que o servidor foi desenvolvido e é gerido pelos seus colaboradores.

Quanto à base de dados da PT Inovação, uma vez que esta é gerida pelos mesmos, o cliente indicou que são eles os responsáveis por controlar os acessos à base de dados e por atualizar os dados na *triple store*.

Requisitos de Performance

Tempo de Resposta

ID - 26.
Prioridade - MUST.

Descrição - recuando à secção 1.4, é possível confirmar que este requisito é um dos requisitos mais importantes para que o projeto atinja o sucesso. De momento, o acesso à informação é feito através de pesquisas na base de dados relacional, sendo que, passando a fazer pesquisas numa *triple store*, o cliente deseja ver o tempo de resposta destas pesquisas reduzido. O cliente indicou

que o tempo ideal que se pretende obter para uma resposta situa-se entre cinco e dez segundos. Portanto, este requisito está diretamente ligado com a eficiência do produto.

Fonte - PT Inovação.

Requisitos de Suporte

O resultado que se pretende atingir pode ser considerado como um novo módulo a incorporar na plataforma OObian, pelo que existem certos pontos que já se encontram desenvolvidos e tratados pela Maisis: possibilidade de fazer testes, manutenção, configuração e instalação.

Contudo, existem dois requisitos de suporte, um de adaptação e outro de escalabilidade.

Integração com Sistemas BSS

ID - 27.
Prioridade - NICE.

Descrição - caso seja possível, pretende-se que o produto final, que tem informação de um sistema OSS (*Operational Support System*), seja desenvolvido de modo a ser possível, no futuro, integrá-lo com sistemas BSS, (*Business Support Systems*).

Sistemas OSS são sistemas vocacionados para o auxílio à gestão dos serviços fornecidos por uma empresa, enquanto sistemas BSS se centram em auxiliar os processos de negócio relacionados com os clientes.

Preparar o produto final para mais tarde ser integrado com sistemas BSS significa construir uma ontologia cuja hierarquia de classes e grau de detalhe se aproximem o mais possível do conhecimento que os clientes têm sobre o negócio e os serviços de uma empresa, assim como deixar em aberto a possibilidade de se adicionar novas classes mais tarde.

Fonte - Maisis.

Processamento de Informação

ID - 28.
Prioridade - MUST.

Descrição - como mencionado na secção 1.4, existe uma característica relacionada com escalabilidade. Portanto, o produto tem de ser capaz de processar e apresentar uma grande quantidade de informação ao utilizador, na ordem dos milhões de triplos (factos).

Fonte - PT Inovação e Maisis.

Requisitos de Design

Base de Dados Relacional Oracle

ID - 29.
Prioridade - MUST.

Descrição - a base de dados para consulta da informação a extrair e a mapear é uma base de dados relacional Oracle Database 11g que pertence ao cliente, PT Inovação.

Fonte - PT Inovação.

Plataforma OObian

ID - 30.
Prioridade - MUST.

Descrição - a plataforma de gestão de conhecimento desenvolvida pela Maisis é a ferramenta que contem o resultado do mapeamento de modo a fazer uso da sua interface para apresentar a informação aos utilizadores.

Fonte - Maisis.

Requisitos de Implementação

Linguagem SQL

ID - 31.
Prioridade - MUST.

Descrição - o acesso à base de dados da PT Inovação tem de ser feito através de queries SQL devido a ser uma base de dados relacional Oracle.

Fonte - PT Inovação.

Linguagem OWL

ID - 32.
Prioridade - MUST.

Descrição - o *schema* ontológico tem de estar descrito em linguagem OWL-DL, pois é a linguagem compreendida pela plataforma de gestão de conhecimento.

Fonte - Maisis.

Duplicação de Dados

ID - 33.
Prioridade - MUST.

Descrição - devido às políticas de gestão e de trabalho da PT Inovação, a duplicação dos dados da base de dados na *triple store* é um requisito obrigatório. As suas políticas defendem a duplicação de dados devido aos seguintes fatores:

- Garantia de performance na obtenção de respostas;
- Base de dados pode não se encontrar otimizada, o que pode levar a tempos de resposta longos, ou mesmo à não obtenção de respostas;
- Não sobrecarregar a base de dados com *queries*;
- Segurança dos dados existentes na base de dados, pois os utilizadores nunca lhes vão conseguir aceder diretamente.

O ponto sobre a performance é um dos pontos mais importantes para justificar o porquê da duplicação de dados.

Uma vez que o servidor de conhecimento opera sobre uma *triple store*, consegue ter um grau de expressividade maior e melhor performance [6] [9], do que se usasse uma base de dados relacional. Devido a estas vantagens, o servidor consegue suportar operações de paginação, indexação e pesquisa livre, operações complexas devido ao volume de dados a instanciar, mas que são uma mais-valia na usabilidade do sistema.

A Maisis pretendia ter um sistema que tivesse um tempo de resposta muito reduzido e que conseguisse instanciar uma grande quantidade de informação, na ordem dos milhões de registos, pelo que optaram por construir o próprio modelo de dados. Por terem conhecimento de quais as *queries* que pretendiam responder, estruturam a informação de acordo com os objetivos pretendidos, o que lhes permitiu passar a ter o modelo de dados em disco em vez de memória. Com esta abordagem, foi-lhes possível criar um índice de apoio à *triple store* que lhes traz duas vantagens: maior performance na obtenção de informação e possibilidade de pesquisa livre. Como referido anteriormente, a PT Inovação não pretende sobrecarregar a base de dados, e quer fazer uso destas mais-valias, razão pela qual se vai efetuar uma duplicação de dados, pois evita-se acrescentar o *overhead* de fazer *quering* na base de dados às operações feitas pelo servidor.

Por fim, quando se fala em duplicação de dados, o problema que salta logo à vista é o de espaço de armazenamento necessário, para além das inconsistências que podem surgir da dessincronização de dados entre a base de dados e a *triple store*, e todos os problemas que daí vêm. Porém, a PT Inovação não se mostrou preocupada com este problema uma vez que o teve em consideração aquando da tomada de decisão de quais as suas políticas.

Fonte - PT Inovação e Maisis.

Requisitos de Interface

Cliente OObian

ID - 34.
Prioridade - MUST.

Descrição - para ser possível aos utilizadores usufruírem do produto final, é necessário ter um cliente OObian para se poder aceder ao servidor da

Maisis.

Fonte - Maisis.

Requisitos Físicos

Dispositivo de Computação

ID - 35.
Prioridade - MUST.

Descrição - de modo a suportar um cliente OObian, é necessário ter uma máquina com os seguintes requisitos mínimos:

- Sistema operativo Mac OS 9 ou Windows 2008;
- Microsoft Silverlight;
- Processador Intel Dual Core 1.7GHz;
- 2 Gigabyte (GB) de memória (RAM);
- 200 Megabyte de espaço em disco;
- Monitor;
- Dispositivo apontador compatível.

Fonte - Maisis.

Apêndice D

Metodologia Genérica de Trabalho

Metodologia de Integração de Dados

Especificação

30 de Junho de 2013

v1.0

Luís Alte da Veiga

Índice

Lista de Acrónimos

BD	Base de Dados
KDIS	*Knowledge Data Integration System*
OWL	*Web Ontology Language*
SQL	*Structured Query Language*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
XML	*eXtensible Markup Language*

Tabela 1: Acrónimos

Lista de Tabelas

Lista de Figuras

1 Introdução

1.1 Contextualização

A Maisis é uma empresa cuja principal atividade tem como alvo o mercado das telecomunicações, desenvolvendo sistemas de interpretação e criação automática de conhecimento de organizações. Tem como missão, desenvolver soluções tecnológicas, destinadas a servir de forma diferenciada cada cliente, e a sua visão é a de ser uma empresa de vanguarda, capaz de criar soluções inovadoras e de referência.

Grande parte dos seus clientes recorre à Maisis devido a terem utilizadores que pretendem consultar, cruzar e explorar informação de um modo conceptual. Esta informação, normalmente, encontra-se armazenada em bases de dados relacionais, e contém dados sobre as próprias empresas, os seus projetos, os seus clientes, as suas ferramentas, os seus recursos, entre outro tipo de informação que varia de empresa para empresa.

Para facilitar a visualização da informação desta maneira, a Maisis desenvolveu, e gere, uma plataforma de nome OObian, que opera sobre uma triple store e tem um interface próprio que representa a informação sob a forma de grafos, pelo que a solução passa por fazer uso desta plataforma para armazenamento, gestão e consulta da informação.

Uma vez que a plataforma OObian opera sobre uma triple store, significa que se tem de fazer uso de ontologias para representar os conceitos e os relacionamentos entre estes. O recurso a ontologias, para além de facilitar a visualização da informação de um modo mais conceptual, sob a forma de um grafo, permite que o processo de construção de resposta a perguntas evite o cruzamento de tabelas inteiras, como é o caso com bases de dados relacionais.

Normalmente os dados de uma empresa estão armazenado em bases de dados relacionais, pelo que, para permitir o acesso a estes, e sua visualização, via a plataforma OObian, é necessário fazer um mapeamento da informação

que se encontra em bases de dados relacionais para ontologias. Isto significa que é necessário fazer um mapeamento de um *schema* relacional para um *schema* ontológico.

Para realizar esta tarefa de mapeamento, à semelhança da plataforma OObian, a Maisis desenvolveu e gere um agente de integração de dados de nome KDIS. As suas funções são a de recolher o *schema* relacional, recolher o *schema* ontológico, fazer o mapeamento do primeiro *schema* para o segundo, e enviar o resultado para a plataforma OObian.

1.2 Objetivo do Documento

O objetivo primordial do presente documento é o de apresentar uma metodologia de trabalho para a integração de bases de dados relacionais com ontologias, mais concretamente, com a plataforma OObian, proprietária da Maisis.

Juntamente com a metodologia apresentada, é ainda apresentado um guia para a fase de desenvolvimento.

1.3 Documentos Relacionados

Documentos de Entrada	**Breve Descrição**
Agente de Integração de Dados	Este documento tem como objetivo a especificação do Agente de Integração de Dados, baseada no documento de requisitos criado e aprovado anteriormente.
Manual de Formação do OObian	O documento consiste num manual de apresentação da plataforma OObian: descrição, objetivos e usabilidade.

Tabela 1.1: Documentos Relacionados

2 Arquitetura do Sistema

Neste capítulo é apresentada a arquitetura geral dos diferentes projetos de integração de dados. A figura 2.1 apresenta a arquitetura através de um diagrama com as diferentes componentes de um sistema e do modo como estas comunicam entre si.

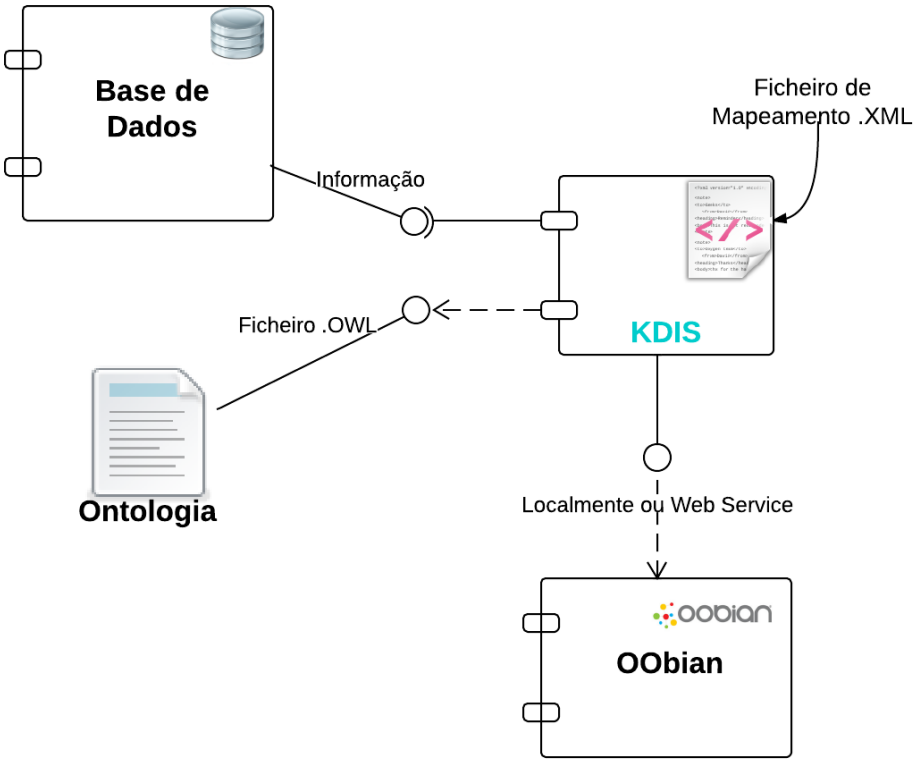


Figura 2.1: Arquitetura.

Como se pode observar na figura 2.1, é possível identificar três componentes: Base de Dados, KDIS e OObian. Para obter os dados a apresentar, o KDIS estabelece ligação com a base de dados de onde vai extrair informa-

ção. Relativamente à ontologia, é criada manualmente no software *Protégé* e exportada para um ficheiro .OWL a que o KDIS vai aceder de modo a ter conhecimento do *schema* ontológico.

Tendo a informação vinda da base de dados e a ontologia, constrói-se um ficheiro em linguagem XML que conterá todo o mapeamento do *schema* relacional para o *schema* ontológico. Terminada a tarefa de mapeamento, o KDIS comunica com a plataforma OObian para lhe enviar o resultado final. Esta comunicação pode ser feita de duas maneiras: diretamente, caso o servidor se encontre local, ou através de um *Web Service*, caso a ligação tenha de ser feita remotamente.

3 Metodologia de Trabalho

3.1 *Schemas* Relacional e Ontológico

Os *schemas* relacional e ontológico revelam-se como duas das ferramentas mais importantes para a integração de dados, razão pela qual é de extrema importância conseguir ter ambos antes de se começar a fase de mapeamento.

Uma vez que o *schema* ontológico se baseia no *schema* da base de dados, a primeira tarefa é a de recolher, junto do cliente, todo o *schema* relacional, e ao mesmo tempo perceber quais as entidades, relações e principais atributos que o cliente pretende que estejam no produto final.

Deste modo, reduz-se o tempo de familiarização com o modelo de dados, e torna-se possível a construção de uma ontologia o mais perto possível da ontologia final, pois, devido à abordagem de desenvolvimento apresentada na secção 3.3, a ontologia terá de ser refinada ao longo do tempo.

3.2 Programa de Reuniões

Os projetos empresariais têm a si associadas reuniões entre o cliente e a entidade que desenvolve o produto que o cliente pretende. Contudo, o cliente tem tipicamente uma disponibilidade muito reduzida para a realização destas reuniões, pelo que é de extrema importância ter o trabalho organizado e as reuniões bem preparadas de modo a se tirar um maior proveito destas.

Um possível programa de reuniões a seguir é apresentado na tabela 3.1.

Reunião	**Assuntos**	**Resultados**	**Tarefas a Realizar**
1	Estrutura Geral da BD Entidades, Relações e Atributos	*Schema* da BD	*Querying* Ontologia Mapeamento (parcial)
2	Esclarecer Dúvidas da BD Validar Ontologia e Mapeamento	Ontologia Novas Funcionalidades	Grupos de Requisitos Implementar um Grupo Testar
3-N	Validar Implementação	*Feedback* do Cliente	Alterações Implementar Novo Grupo

Tabela 3.1: Programa de Reuniões

Analisando a tabela 3.1, é possível confirmar que existem três tipos de reuniões. Na primeira reunião deve-se perceber a estrutura geral da base de dados, entidades, relações e atributos, e saber quais destas componentes o cliente pretende que sejam mapeadas e estejam no produto final. No final desta reunião pretende-se que o cliente disponibilize o *schema* da BD para que seja possível realizar as seguintes tarefas:

- Fazer querying para se consolidar o conhecimento sobre a estrutura da BD e a informação a mapear;
- Ser possível construir a ontologia;
- Ter o material necessário para fazer uma primeira versão parcial do mapeamento.

A segunda reunião tem como principais objetivos esclarecer quaisquer dúvidas que se tenham sobre a estrutura da BD e/ou sobre alguns atributos, assim como validar a ontologia que foi construída e o mapeamento parcial que foi feito. Esta validação deve ser feita através da apresentação do trabalho já na plataforma OObian, pois permitirá ao cliente ter desde cedo uma perceção visual de um protótipo do que será o produto final, o que fomentará a sua correção e completude desde o início. Esta validação irá permitir, por um

lado, refinar e aproximar a ontologia da sua estrutura final, e por outro, levar a um debate sobre a possibilidade de haver ou não novas funcionalidades.

Após a segunda reunião, já será possível haver um maior foco na tarefa de mapeamento. A abordagem a seguir é apresentada em maior detalhe na secção 3.3, mas de um modo geral, consiste em dividir os requisitos em diferentes grupos, implementar um desses grupos e fazer os respetivos testes.

Por fim, as seguintes reuniões consistem em apresentar ao cliente a implementação que vai sendo feita e o cliente fornece o seu *feedback*. Após cada uma destas reuniões, fazem-se as alterações que o cliente indicou e passa-se à implementação do grupo seguinte de requisitos, caso não se tenha atingido o final do projeto.

Tal como mencionado no início da presente secção, o cliente nem sempre se encontra disponível para lhe ser apresentado o trabalho desenvolvido. Nesta situação, uma possível solução passa por alocar o projeto num servidor e fornecer ao cliente um endereço URL para que este possa consultar e acompanhar o desenvolvimento do produto final. Relativamente a tarefas, de modo a que a equipa de desenvolvimento não fique parada à espera de novas indicações, é sempre possível passar ao desenvolvimento de um novo grupo de requisitos, ou até mesmo de implementar novas funcionalidades que o cliente não mencionou mas que podem ser do seu agrado, sendo necessário obter o acordo explícito do cliente relativamente à implementação de tais funcionalidades extra.

3.3 Desenvolvimento

A abordagem que se pretende seguir durante a fase de desenvolvimento é uma que nos permita dividir todos os requisitos funcionais em pequenos grupos, e, para cada um desses grupos, implementar, testar, apresentar ao cliente, receber o respetivo *feedback*, fazer as alterações necessárias e voltar a testar. Para tal, deve-se seguir um processo de desenvolvimento ágil. A figura 3.1 demonstra a abordagem para cada um dos grupos de requisitos.

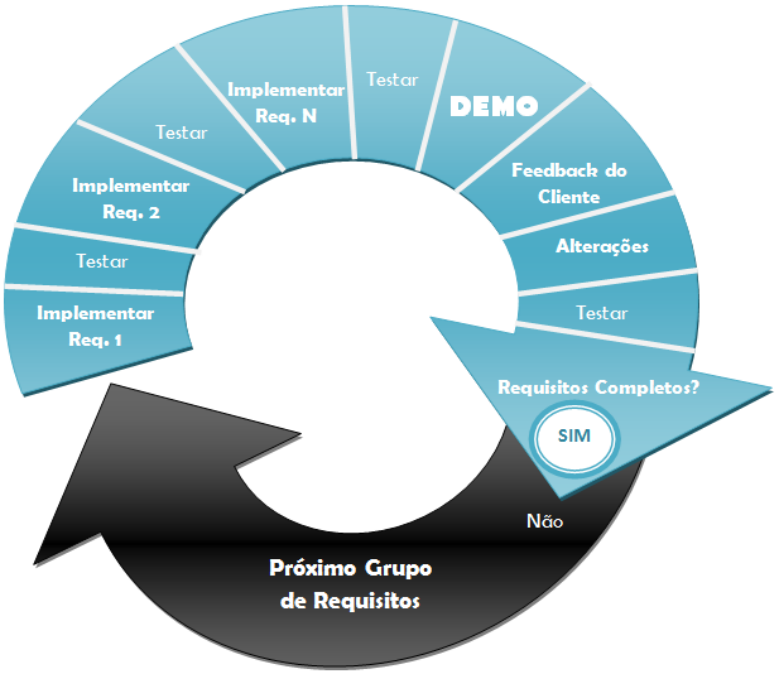


Figura 3.1: Processo de Trabalho.

Após se fazer o levantamento dos requisitos do projeto, dividem-se estes em pequenos grupos, sendo o critério o de agrupar os requisitos consoante a classe a que pertencem na ontologia.

Para cada um desses grupos, a figura 3.1 ajuda a perceber o método de desenvolvimento. À medida que se implementa um requisito do grupo escolhido, testa-se se a informação apresentada e relacionada é a pretendida, sendo que, terminada a implementação do grupo de requisitos, prepara-se uma demonstração para apresentar ao cliente.

Nestas demonstrações, o cliente fornece *feedback* quanto às propriedades a apresentar para cada classe (nome, tipo, descrição, entre outras) e se devem surgir novas relações entre classes, o que significa um acréscimo na informação a disponibilizar, assim com o referido refinamento da ontologia. De seguida, procede-se às alterações identificadas pelo cliente e volta-se a efetuar testes. Terminada esta tarefa, ou se procede à implementação do grupo de requisitos seguinte, ou, no caso de ser o último grupo, dá-se por encerrada a fase de desenvolvimento.

4 Desenvolvimento

4.1 Ontologia

A criação da ontologia para o exemplo apresentado de seguida foi feita no software *Protégé*, versão 3.4.8, mas também é possível construir ontologias noutras versões do mesmo software, ou até com outro editor de ontologias, como por exemplo o *TopBraid Composer*[1].

Passando então a apresentar os passos de criação da ontologia, estes são três:

1. Criação das diferentes classes - na tab OWL Classes é criada a hierarquia de classes da ontologia (figura 4.1).

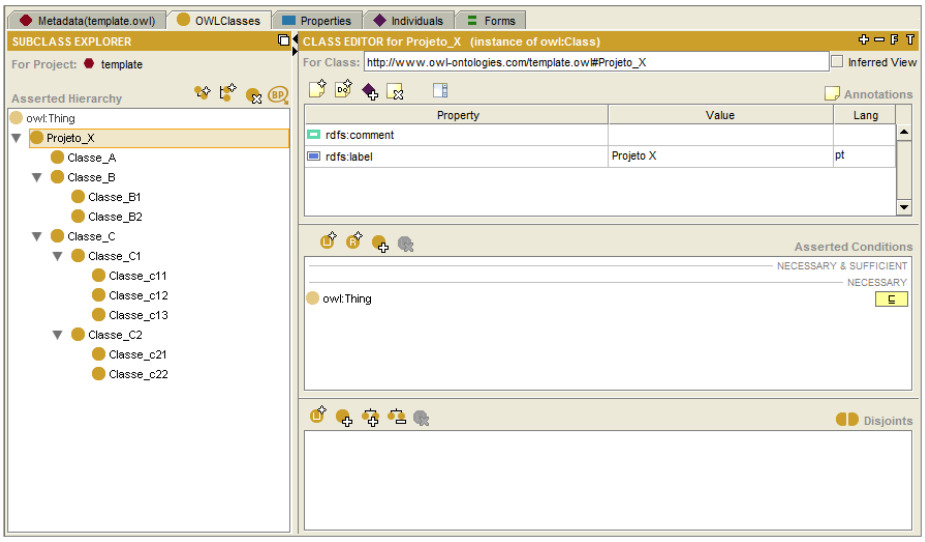


Figura 4.1: *Protégé* - Criação de Classes.

[1] http://www.topquadrant.com/products/TB_Composer.html

2. Definição das propriedades que ligam essas classes - na tab seguinte, Properties, estas propriedades são definidas na sub-tab Object (figura 4.2).

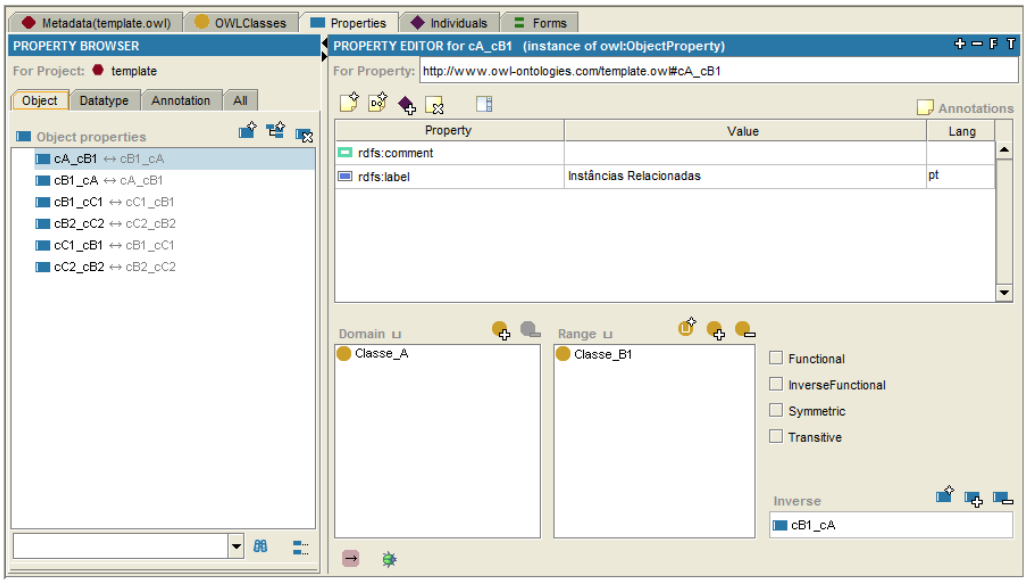


Figura 4.2: *Protégé* - Definição de Propriedades de Ligação.

3. Especificação dos atributos de cada uma das classes - por fim, mantendo a mesma tab do passo 2, os atributos são definidos na sub-tab Datatype (figura 4.3).

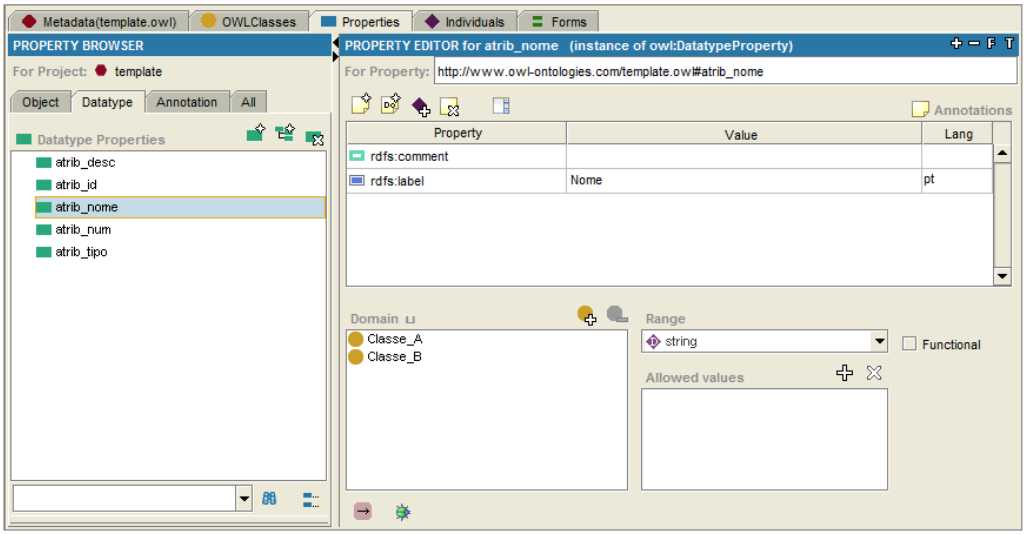


Figura 4.3: *Protégé* - Especificação de Atributos.

Nestes três passos é importante a definição de uma *label* para cada classe e cada propriedade, pois ajudará os utilizadores na compreensão da informação apresentada na plataforma OObian.

Após cumpridos estes passos, a ontologia está criada e pode ser exportada para o ficheiro .OWL a que o KDIS vai aceder. A figura 4.4 ilustra o caminho para fazer esta exportação.

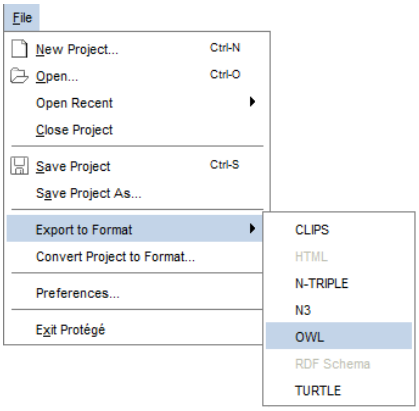


Figura 4.4: *Protégé* - Ficheiro .OWL.

Nota: este documento não pretende ser um manual completo de como criar uma ontologia, pelo que, para um aprofundamento do conhecimento de como construir uma ontologia, é aconselhado consultar o link em rodapé [2].

Neste momento já se tem quer o *schema* relacional quer o *schema* ontológico, pelo que estão reunidas as condições para se avançar para a tarefa de mapeamento, tarefa essa que é da responsabilidade do agente de integração de dados, KDIS.

4.2 Agente de Integração de Dados

Relativamente ao KDIS, o que se pretende é configurar um ficheiro .XML que contenha o mapeamento do *schema* relacional para o *schema* ontológico. No conjunto de imagens que se segue, é possível observar a estrutura deste ficheiro através da apresentação dos diferentes blocos de código XML.

[2] http://owl.cs.manchester.ac.uk/tutorials/protegeowltutorial/resources/ProtegeOWLTutorialP4_v1_3.pdf

1. Comunicação com o KServer e definição da ontologia a utilizar.

```
<kdismapping:KServerConnect
    isRemoteConnection="true"
    username="admin" password="admin"
    serverURI="http://localhost:8280/services/ws/KServerWS?wsdl">

    <kdismapping:TargetOntologyID>
        http://www.owl-ontologies.com/template.owl#
    </kdismapping:TargetOntologyID>

</kdismapping:KServerConnect>
```

Figura 4.5: KDIS - KServer e Ontologia.

Inicialmente preenchem-se os campos para comunicação com o KServer, servidor da plataforma OObian, sendo que os dados de utilizador inseridos são os dados de acesso ao *back office* da plataforma OObian. No campo serverURI define-se o porto do endereço de acesso, que no exemplo é 8280.

De seguida, procede-se à definição da ontologia. Neste campo deve-se inserir o URI da ontologia que foi definido aquando da construção da ontologia no software *Protégé*.

2. Comunicação com a base de dados.

```
<kdismapping:Connection
    uri="jdbc:oracle:thin:@10.112.80.23:1521:devcad"
    connDriver="oracle.jdbc.driver.OracleDriver"
    id="myDB" dbUser="admin" dbPassword="admin"
    xsi:type="kdismapping:DBConnectType"/>
```

Figura 4.6: KDIS - Base de Dados.

O bloco de código XML apresentado na figura 4.6 diz respeito à configuração dos parâmetros para comunicação com a base dados onde se encontra a informação a extrair. Os atributos uri e connDriver servem para definir qual o tipo de base de dados a que se vai aceder.

Quanto aos dados de utilizador, aqui têm de se colocar os dados de utilizador da base de dados, pois é necessário ter permissões de acesso para consulta da informação.

No exemplo apresentado, temos uma base de dados *Oracle*, no entanto, a Maisis teve, tem e terá projetos onde o motor de base de dados é diferente. De seguida apresenta-se um exemplo de configuração para alguns desses motores:

- MySQL
  - uri = ”jdbc:mysql://c051:3306/docebo”
  - connDriver = ”com.mysql.jdbc.Driver”
- PostgreSQL
  - uri = ”jdbc:postgresql://localhost/facebook”
  - connDriver = ”org.postgresql.Driver”
- SQLServer
  - uri = ”jdbc:sqlserver://oobiansrv:1433;databaseName=TICE;”
  - connDriver = ”com.microsoft.sqlserver.jdbc.SQLServerDriver”

Nota: consoante o projeto, o campo a alterar é o uri, mais concretamente, o endereço de acesso e , no caso de o motor ser SQLServer, o nome da base de dados.

3. *Queries* SQL para recolha de informação da base de dados.

```
<!-- Queries -->
<!-- Classes -->
<kdismapping:Query dbID="myDB" id="queryA"
    value="SELECT id_a, name_a, desc_a FROM classea" xsi:type="kdismapping:DBQueryType"/>
<kdismapping:Query dbID="myDB" id="queryB1"
    value="SELECT id_b, name_b, type_b FROM classeb WHERE type_b='B1' " xsi:type="kdismapping:DBQueryType"/>

<!-- ObjectProperties -->
<kdismapping:Query dbID="myDB" id="queryAB1"
    value="SELECT id_b1 FROM ab WHERE id_a = ?" xsi:type="kdismapping:DBQueryType"/>
<kdismapping:Query dbID="myDB" id="queryB1A"
    value="SELECT id_a FROM ab WHERE id_b1 = ?" xsi:type="kdismapping:DBQueryType"/>

<!-- DatatypeProperties -->
<kdismapping:Query dbID="myDB" id="queryNumber"
    value="SELECT count(id_b1) FROM ab WHERE id_a = ?" xsi:type="kdismapping:DBQueryType"/>
```

Figura 4.7: KDIS - *Queries* SQL.

A figura 4.7 apresenta três conjuntos de *queries*. O primeiro grupo serve somente para ir buscar os atributos para cada uma das classes da ontologia. O segundo, ObjectProperties, serve para ir buscar o atributo que relaciona uma determinada classe, que tem de ser passada como argumento na cláusula

'WHERE', com uma outra classe (a descrição da figura 4.11 ajuda a perceber este *query*). Por fim, existe um último grupo cujo propósito é o de obter dados para criar atributos, *Datatype Properties*, que não estão na base de dados mas que se pretendem no produto final, como por exemplo, a contagem do número de ocorrências de um certo campo.

4. Mapeamento do *schema* relacional para o *schema* ontológico.

```
<!-- Mapeamento -->
<!-- Classe A -->
<kdismapping:Map confidence="1.0" isIndexable="true" queryID="queryA" source="myDB"
        targNamespace="http://www.owl-ontologies.com/template.owl#">

    <kdismapping:ClassID id="http://www.owl-ontologies.com/template.owl#Classe_A"
    namespace="http://www.owl-ontologies.com/template.owl#"/>

    <!-- Identificador na Triple Store -->
    <kdismapping:ID>
        <kdismapping:Field language="pt" type="xsd:String">
            <kdismapping:Data value= "Classe_" />
            <kdismapping:Data queryColumn="id_a" />
        </kdismapping:Field>
    </kdismapping:ID>

    <kdismapping:Properties>
        <!-- DataProperties -->

        <!-- ObjectProperties -->
    </kdismapping:Properties>

</kdismapping:Map>
```

Figura 4.8: KDIS - Mapeamento: Estrutura e Configurações.

Após definidas as *queries* para obtenção de informação, procede-se ao bloco que contém o mapeamento do *schema* relacional para o ontológico. A figura 4.8 contém a estrutura do mapeamento que deve ser feito para cada classe. As primeiras configurações servem para especificar qual a *query* de onde vêm os dados, definida no bloco da figura 4.7, e definir um ID para a classe em questão na *triple store*. Para tal, especifica-se qual o URI da classe, definido anteriormente no *Protégé*, a atribuir para o ID, e indica-se qual o campo da *query*, 'id_a', que contém o valor pretendido para o ID.

Nota: caso se especifique o campo value, também se tem que o especificar, e ter o mesmo valor, quando se relacionar esta classe com outra através

de uma *Object Property*. Para o exemplo apresentado, isto teria de ser feito no mapeamento da Classe B1.

Posteriormente, temos presente o grupo de mapeamento das propriedades. Uma vez que as propriedades se dividem em *Object* e *Datatype*, a apresentação dos seus mapeamentos foi divida em duas partes. A figura 4.10 contém o código XML referente às *Datatype Properties*, enquanto a figura 4.11 diz respeito às *Object Properties*.

Contudo, imediatamente antes do bloco *Datatype Properties*, existe o bloco de código apresentado na figura 4.9.

```
<kdismapping:Labels>
    <kdismapping:Field language="pt" type="xsd:String">
        <kdismapping:Data queryColumn="name_a" />
    </kdismapping:Field>
</kdismapping:Labels>
```

Figura 4.9: KDIS - Mapeamento: *Label* da Classe.

Este bloco de código diz respeito ao nome que será apresentado na interface de visualização, quer nas classes, nas instâncias, nas relações e nos atributos, como se pode confirmar nas figuras 4.14 a 4.16. Neste caso, escolheu-se o campo da *query* que contem o nome da classe.

```
<!-- DatatypeProperties -->
<!-- Atributo Nome -->
<kdismapping:DataProperty id="http://www.owl-ontologies.com/template.owl#atrib_nome">
    <kdismapping:Field language="pt" type="xsd:String">
        <kdismapping:Data queryColumn="name_a" />
    </kdismapping:Field>
</kdismapping:DataProperty>

<!-- Atributo Número -->
<kdismapping:DataProperty id="http://www.owl-ontologies.com/template.owl#atrib_num" queryID="queryNumber">
    <kdismapping:QueryParameters>id_a</kdismapping:QueryParameters>
    <kdismapping:Field language="pt" type="xsd:int">
        <kdismapping:Data queryColumn="count(id_b1)" />
    </kdismapping:Field>
</kdismapping:DataProperty>
```

Figura 4.10: KDIS - Mapeamento: *Datatype Properties*.

O mapeamento das *Datatype Properties* pode ser feito de duas maneiras, como demonstra a figura 4.10. No primeiro caso, simplesmente se faz corresponder o valor do campo que vem na *query* com o URI da propriedade.

Voltando à figura 4.7, no final desta é apresentada uma *query* que diz respeito a *Datatype Properties* e que necessita de receber um parâmetro para devolver um valor. Portanto, o segundo caso de mapeamento da figura 4.10 contém mais um campo na primeira linha, que é a referência à *query* que contém o valor da propriedade, e de seguida é acrescentada uma linha para se especificar qual será o parâmetro a enviar para essa *query*.

Nota: nalguns ficheiros de mapeamento, é possível encontrar o campo isSearchable no mapeamento das *Datatype Properties*. Contudo, a sua funcionalidade encontra-se desativada e a sua especificação torna-se dispensável, pois a definição de se uma propriedade é pesquisável ou não por texto livre é feita na plataforma OObian, como se pode observar na figura 4.13.

```
<!-- ObjectProperties -->
<!-- Classes B1 Associadas -->
<kdismapping:ObjectProperty confidence="1.0" dbSource="myDB"
        id="http://www.owl-ontologies.com/template.owl#cA_cB1"
        queryID="queryAB1" notCheckInverse="true">

    <kdismapping:QueryParameters>id_a</kdismapping:QueryParameters>
    <kdismapping:ConnectedObjectID>
        <kdismapping:Field language="pt" type="xsd:String">
            <kdismapping:Data value= "Classe_" />
            <kdismapping:Data queryColumn="id_b1" />
        </kdismapping:Field>
    </kdismapping:ConnectedObjectID>
</kdismapping:ObjectProperty>
```

Figura 4.11: KDIS - Mapeamento: *Object Properties*.

Quanto às *Object Properties*, o seu mapeamento é muito parecido com o segundo caso de mapeamento das *Datatype Properties*. Observando a figura 4.11, conclui-se que no início se especifica qual o URI da propriedade e a *query* onde se encontra o seu valor, de seguida indica-se o parâmetro a enviar para a *query*, e termina-se indicando qual o campo da *query* que se pretende.

Por fim, referir que o campo notCheckInverse tem de ser definido com valor true, uma vez que as propriedades inversas são configuradas neste ficheiro e não se pretende portanto que o KServer calcule a inversa de cada uma das propriedades. A definição das propriedades inversas é feita uma vez que para

cada classe se configura todas as suas propriedades, pelo que acaba-se sempre por definir as propriedades inversas.

Nota: nas imagens apresentadas ao longo da presente secção, é possível observar que a referência aos campos das *queries* é feita sempre em minúsculas. Contudo, estas referências podem ter de ser feitas em maiúsculas para os casos em que os motores de base de dados devolvem a informação com os caracteres em maiúsculas, como acontece com o motor *Oracle*. Portanto, é necessário ter em atenção o modo como cada motor de bases de dados devolve a informação.

Comunicação com a Plataforma OObian.

Assim que o ficheiro de mapeamento se encontrar configurado, é altura de correr o KDIS e enviar o resultado para o servidor OObian. O procedimento é o de iniciar e configurar o servidor OObian (ver secção 4.3), e de seguida correr o ficheiro **start-kdis-total.bat**. Neste ficheiro é necessário especificar qual o ficheiro .XML que contém o mapeamento.

Assim que o KDIS terminar o envio da informação, basta aceder à versão cliente do OObian através do endereço `http://localhost:8280/insight` e consultar a informação.

4.3 Plataforma OObian

Na plataforma OObian não se vai fazer qualquer desenvolvimento, pois esta já se encontra desenvolvida, mas serão feitas algumas tarefas para gestão do produto final. Para tal, é necessário aceder à maquina onde se encontra o servidor OObian e correr o ficheiro **oobian-run.bat** para dar início ao servidor.

De seguida, e tendo em conta a configuração efetuada para comunicação com o KServer, a quando do mapeamento, acede-se ao endereço `http://localhost:8280/oobian-admin/`, com os dados de utilizar especificados, e procede-se então à gestão da ontologia:

1. Carregamento do *schema* ontológico - o primeiro passo é o de carregar o ficheiro .OWL para a plataforma OObian de modo a que esta tenha conhecimento do *schema* ontológico. Na secção Configuração do Servidor, tab Gestão de Ontologias, é possível carregar o ficheiro com a ontologia e especificar as suas definições (figura 4.12).

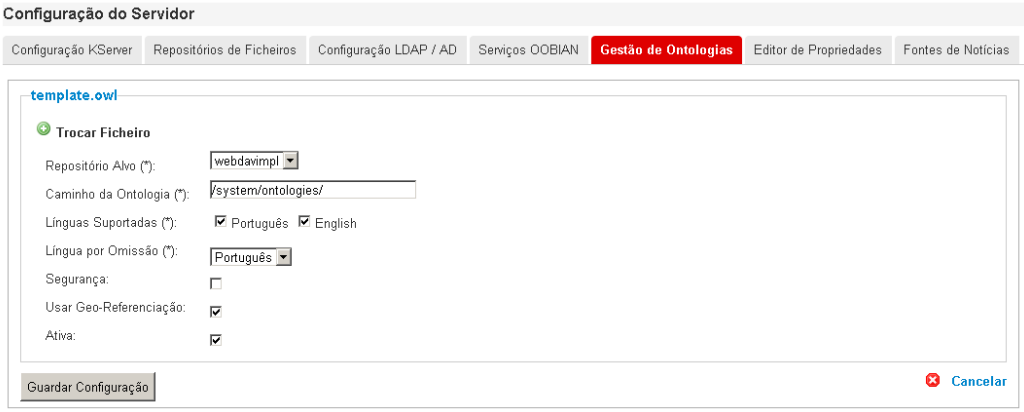


Figura 4.12: OObian - Carregamento do *Schema* Ontológico.

2. Configuração das propriedades - de seguida, na mesma secção mas numa tab diferente, Editor de Propriedades, definem-se quais propriedades são filtráveis e quais são pesquisáveis por texto livre (figura 4.13).

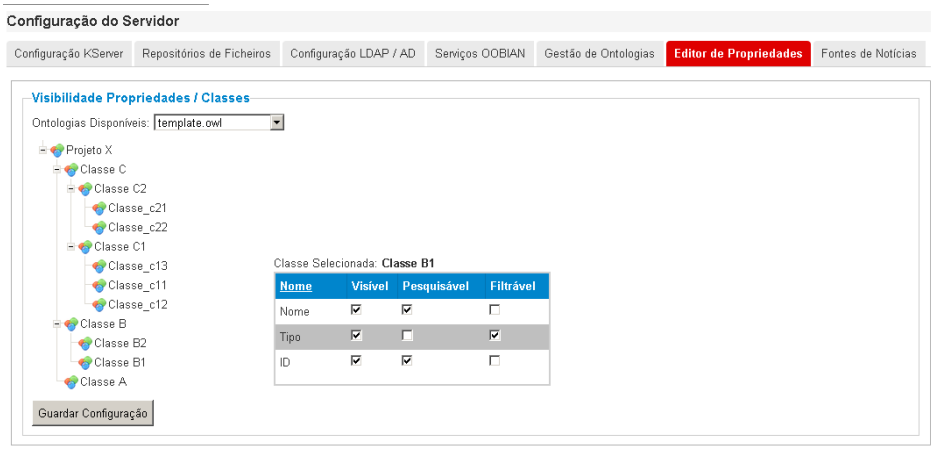


Figura 4.13: OObian - Configuração de Propriedades.

Normalmente, as propriedades definidas como pesquisáveis são aquelas cujos valores são mais específicos, como por exemplo, textos descritivos ou números identificadores. Por seu lado, as propriedades filtráveis são mais centradas em atributos relacionados com quantidade, ou seja, propriedades em que os filtros a aplicar podem ser, por exemplo, intervalos temporais ou filtrar por determinados tipos de instâncias. Na figura 4.16 é possível consultar os campos onde se podem fazer pesquisas e filtragens.

De modo a validar as alterações efetuadas, deve-se reiniciar o servidor OObian e a partir desse momento as propriedades já se encontram configuradas como pretendido.

Interface de Visualização

Com o objetivo de completar a descrição das componentes de desenvolvimento, de seguida existem três imagens a ilustrar o resultado do mapeamento.

A figura 4.14 pretende ilustrar parte da hierarquia de classes criada no software *Protégé*, e apresentada na figura 4.1.

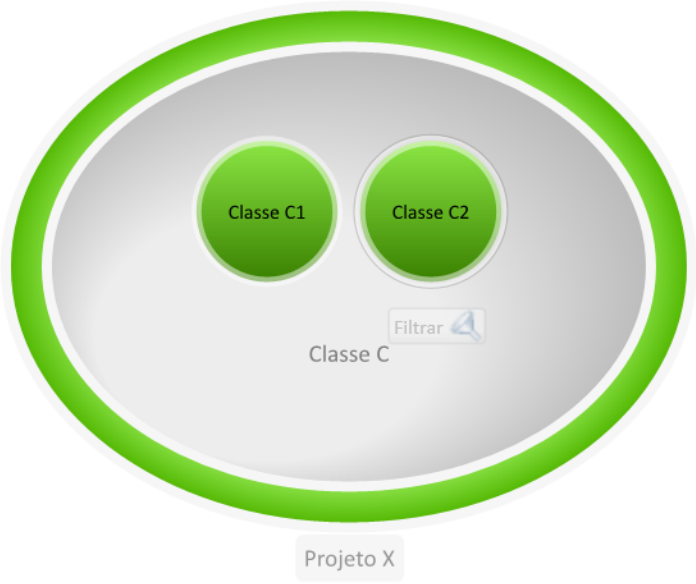


Figura 4.14: Interface OObian - Hierarquia de Classes.

Como se pode observar, derivada da classe Classe C temos as suas duas sub-classes, Classe C1 e Classe C2, fazendo tudo parte do Projeto X. Anteriormente, na figura 4.1, estas entidades têm os nomes Projeto_X, Classe_C, Classe_C1 e Classe_C2, o que não se confirma ao visualizá-las na plataforma OObian. A razão para tal deve-se à propriedade label que foi definida para cada classe, tal como se pode confirmar na figura 4.1.

Ao observar a ontologia na figura 4.1, facilmente se conclui que a apresentação das sub-classes da Classe B serão apresentadas de forma semelhante à figura 4.14. Uma vez numa destas janelas, uma das possibilidades do utilizador é continuar a navegar nas sub-classes. Caso se encontre numa sub-classe que não tenha descendentes, como por exemplo a Classe B1, o resultado a aparecer será o ilustrado na figura 4.15.

Figura 4.15: Interface OObian - Instâncias de uma Classe.

A figura 4.15 apresenta todas as instâncias da classe B1. Estas instâncias foram carregadas da base de dados em *queries* semelhantes às duas primeiras da figura 4.7, e o seu mapeamento foi configurado seguindo o exemplo apresentado na figura 4.8. Quanto mais dados estas *queries* devolverem, maior será o número de instâncias apresentadas.

Por fim, ao navegar-se para uma instância em específico, o resultado será algo parecido com o da figura 4.16, ou seja, será possível visualizar essa instância em específico, as suas relações e os seus atributos.

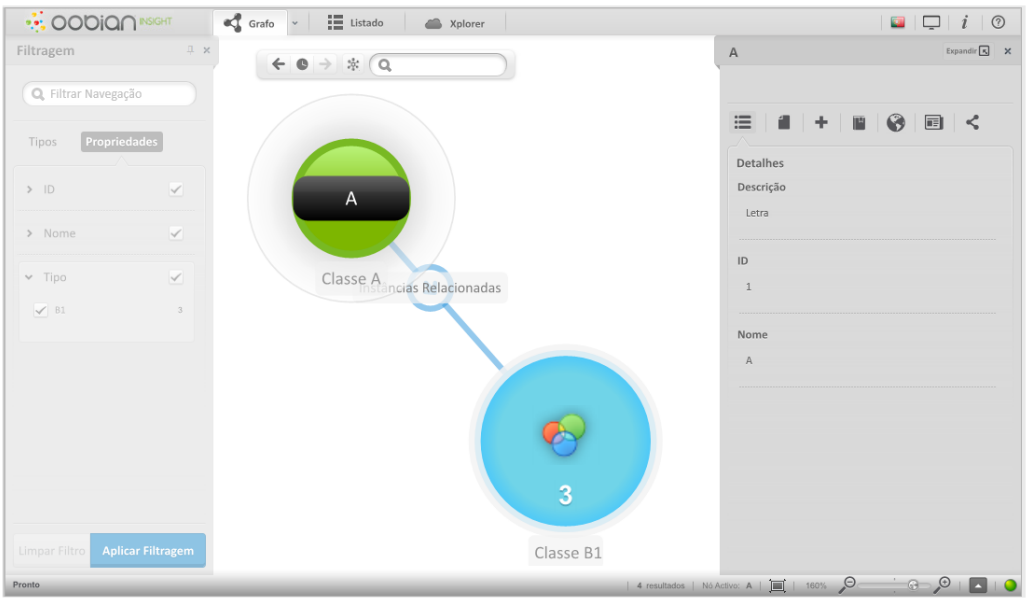


Figura 4.16: Interface OObian - Instância e suas Propriedades.

A figura 4.16 pode ser dividida em três partes: do lado esquerdo os campos de pesquisa e filtragem, no lado direito os atributos da instância em visualização, e no centro a instância e respetivas relações.

As relações de uma instância têm três fases de criação: definição na ontologia como uma *Object Property* (figura 4.2), carregamento dos dados da base de dados (figura 4.7), e configuração do mapeamento (figura 4.11). Caso se pretenda aumentar o número de relações de determinada instância, a solução passa por criar mais *Object Properties* entre essa instância e diferentes classes.

Relativamente aos atributos, estes têm uma criação muito semelhante à das relações, sendo a sua composta pelas mesmas três fases: definição na ontologia, mas como *Datatype Properties* (figura 4.3), carregamento dos dados da base de dados (figura 4.7), e configuração do mapeamento (figura 4.10). Por fim, para se aumentar o número de atributos apresentados, devem-se definir mais *Datatype Properties* na ontologia, e carregar mais campos da base de dados ou criar novos atributos a partir de *queries*.

5 Considerações Finais

A metodologia apresentada neste documento tem como objetivo principal agilizar os processos de negócio e o desenvolvimento dos projetos inerentes à empresa Maisis, contribuir para uma standardização dos seus processos internos relativos a este tipo de projetos, e assim melhorar a qualidade dos produtos desenvolvidos.

Aliado a esta metodologia está o guia apresentado para a fase de desenvolvimento. Com este guia, pretende-se facilitar a ambientação de novos colaboradores com as ferramentas da Maisis através da apresentação do conjunto de passos a seguir, de imagens ilustrativas do que se pretende (*screenshots*), e da identificação de características importantes.

Contudo, nenhum sistema é perfeito. Por isso mesmo a Maisis tem em mente certas melhorias que se podem fazer, sendo uma das mais importantes a criação de uma ferramenta, com um interface gráfico próprio, para ajudar na tarefa de mapeamento.

Esta ferramenta tem de ser capaz de comunicar com bases de dados e instanciar automaticamente os dados, dados esses que serão introduzidos pelo utilizador na interface da ferramenta. O utilizador tem de ter igualmente a possibilidade de definir os relacionamentos, *Object Properties*, e os atributos, *Datatype Properties*, desses dados. O resultado final pretende.se que seja o ficheiro .XML configurado por completo e pronto a ser executado pelo KDIS.

Analisando os blocos de código XML apresentados na secção 4.2, é possível constatar que a configuração do mapeamento de cada classe envolve quatro partes idênticas entre todas as classes: *id*, *label*, *datatype properties* e *object properties*. A configuração geral de cada uma destas componentes apenas difere na coluna da base de dados onde se encontra o atributo pretendido, e no URI correspondente na ontologia. Portanto, este documento pode servir de auxílio no desenvolvimento da referida ferramenta, no sentido em que o programador pode consultar a estrutura do ficheiro .XML que a ferramenta deve gerar, após inserção dos dados por parte do utilizador.