Mestrado em Engenharia Informática
Estágio
Relatório Final

Visualização de Dados de Missões Aeroespaciais

Ivo Miguel Botelho Amaral
imba@student.dei.uc.pt

Orientador DEI:
Professor Doutor Carlos Fonseca

Orientador VisionSpace Technologies:
Mário Ulisses Costa

Data: 1 de Julho de 2014

FCTUC DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Resumo

Graças às missões realizadas pela ESA (Agência Espacial Europeia) torna-se possível efetuar uma série de observações e medições sobre o planeta Terra. Desta forma, são gerados vários tipos de dados que posteriormente são disponibilizados à comunidade científica permitindo o estudo de várias características passadas, atuais e futuras do nosso planeta.

Durante estas missões são realizadas diversas medições que incidem principalmente sobre a atmosfera terrestre, calotas de gelo, continentes e oceanos, criando assim uma vasta quantidade de material essencial para o desenvolvimento da comunidade científica e sector industrial da Europa e do Mundo.

Depois dos dados serem disponibilizados, torna-se possível o seu estudo e, assim, a elaboração de vários tipos de estatísticas e relatórios sobre áreas importantes, como por exemplo: alterações climáticas, previsão de cheias e tempestades, áreas do planeta que estão a perder vegetação e informações sobre o degelo das calotas polares. Estes relatórios são fundamentais para ajudar na tomada de decisões políticas e, consequentemente, ajudar na preservação do meio ambiente.

A realização deste estágio, na empresa *VisionSpace Technologies*, no âmbito do Mestrado em Engenharia Informática, tem como principal objetivo a visualização de dados recolhidos pelos instrumentos das missões numa plataforma *online* que ofereça aos utilizadores uma interface simplificada, deixando de parte o moroso processo de recolha e processamento de dados pelos pacotes de *software* científico.

Para cumprir o objetivo acima descrito é necessário, em primeiro lugar, realizar um estudo de missões efetuadas pela ESA, assim como dos vários instrumentos de observação terrestre que estas missões levam a bordo. Desta forma, acedendo aos vários tipos de dados disponibilizados pela ESA é possível perceber estas estruturas e também os diferentes tipos de dados existentes.

É essencial para a plataforma de visualização de dados a existência de um sistema que contenha e que disponibilize os diferentes dados. Assim sendo, os dados das missões estão presentes numa base de dados orientada a documentos e é possível acedê-los através de um *web service* que possui uma API pública. A plataforma de visualização *online* utiliza a API do *web service* como forma de obter os dados requisitados pelos utilizadores e, assim, criar vários tipos de representações gráficas. Esta API também vai tornar exequível a criação de novos sistemas que necessitem de aceder aos dados das missões da ESA.

Palavras-chave

Visualização de dados *online*, visualização de dados de missões, exploração de dados, missões ESA.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ASAR – *Advanced Synthetic Aperture Radar*.

AMI – *Active Microwave Instrument*.

ASCII – *American Standart Code for Information Interchange*.

API – *Application Programming Interface*.

ACID – *Atomicty, Cosistency, Isolation, Durability*.

AATSR – *Advanced Along Track Scannig Radiometer*.

AJAX - *Asynchronous JavaScript and* XML.

CSS – *Cascading Style Sheets*.

CUT – *CryoSat User Tool*.

CRUD – *Create, Read, Update, Delete*.

DORIS – *Doppler Orbitography and Radio-Positioning Integrated by Satellite*.

DAO – *Data Access Object*.

ESA – *European Space Agency*.

EnviSat – *Environmental Satellite*.

ELDO – *European Launch Development Organisation*.

ESRO – *European Space Research Organisation*.

ESOC – *European Space Operations Centre*.

ERS1, ERS2 – *European Remote Sensing Satellite* 1/2.

FTP – *File Transfer Protocol*.

GOCE – *Gravity Field and Ocean Circulation Explorer*.

GOMOS – *Global Ozone Monitoring by Occultation of Stars*.

HTML – *Hypertext Markup Language*.

HTTP – *Hypertext Transfer Protocol*.

ISC Kosmotras – *The Internacional Space Company Kosmotras*.

ISS – *Internacional Space Station*.

JSON – *Javascript Object Notation*.

JPG – *Joint Photographic Experts Group*.

LRR – *Laser RetroRflector*.

MERIS – *Medium Resolution Imaging Spectrometer*.

MWR – *MicroWave Radiometer*.

MIPAS – *Michelson Interferometer for Passive Atmospheric Sounding*.

MIRAS – *Microwave Imaging Radiometer using Aperture Synthesis*.

MPH – *Main Product Header*.

MVC – *Model-View-Controller*.

NEST – *Next Esa Sar Toolbox*.

PMG – *Paint Magic image Format*.

PNG – *Portable Network Graphics*.

RA-2 – *Radar Altimeter 2*.

RAM – *Random Access Memory*.

SMOS – *Soil Moisture and Ocean Salinity*.

SAR – *Synthetic Aperture Radar*.

SCIAMACHY – *Scannig Imaging Absorption Spectometer for Atmospheric Chartography*.

SIRAL – *SAR Interferometric Radar Altimeter*.

SQL - *Structured Query Language*.

TIFF- *Tagged Image File Format*.

URI – *Uniform Resource Identifier*.

VST – *VisionSpace Technologies*.

Capítulo 1 - Introdução

A Agência Espacial Europeia é uma organização intergovernamental europeia dedicada à exploração espacial. Foi fundada no ano 1975 através da fusão de duas agências já existentes, *ELDO* (*European Launch Development Organisation*) e *ESRO* (*European Space Research Organisation*), em que a primeira seria responsável pelo sistema de lançamento dos satélites e a segunda responsável pelo desenvolvimento dos próprios satélites [1].

A *ESA* tem a sua sede em Paris, França, e o controlo de missões é feito a partir do *ESOC* (*European Space Operations Centre*) localizado em Darmstadt, na Alemanha [1]. De realçar que o *ESOC* já controlou mais de 50 satélites em toda a sua história. Esta agência possui, como principal base de lançamento de foguetes, o Centro Espacial de *Kourou* situado na Guiana Francesa (América do Sul) [2]. Esta localização foi escolhida devido à sua proximidade da linha do equador, o que permite uma significativa poupança de combustível em relação a outros locais mais afastados do mesmo [3].

Os países que fazem parte da *ESA* são a Alemanha, Áustria, Bélgica, Dinamarca, Espanha, Finlândia, França, Grécia, Holanda, Irlanda, Itália, Luxemburgo, Noruega, Polónia, Portugal, República Checa, Roménia, Reino Unido, Suécia e Suíça. O Canadá faz parte de alguns projetos com um acordo de cooperação. A Hungria, Estónia e Eslovénia fazem parte dos países Europeus de cooperação [4].

Os contributos mais significativos para o orçamento de 2013 da *ESA* recaíram sobre os seguintes países: Alemanha (com 24,8% do orçamento da *ESA* que equivale a 772,7 Milhões de Euros), França (participa em 24% do orçamento o que equivale a 747,5 Milhões de Euros), Itália (com 12,8% do orçamento total que equivale a 400 Milhões de Euros) e Reino Unido (participa em 9,6% do orçamento que é equivalente a 300 Milhões de Euros). A União Europeia tem um peso de 21,3% do orçamento total, o que equivale a 911,1 Milhões de Euros. Portugal contribui com 0,5% do orçamento, ou seja, cerca de 16,1 Milhões de Euros, através do programa Fundação para a Ciência e Tecnologia [5].

O principal objetivo das missões realizadas pela *ESA* é o de realizar pesquisas científicas e tecnológicas com propósitos civis (i.e não militares), como observação terrestre, telecomunicações e navegação. Desta forma, torna-se possível influenciar fortemente o desenvolvimento tecnológico, com vista à melhoria na qualidade de vida de todos os habitantes do planeta Terra [6].

A *ESA* possui uma política, revista no ano de 2010 [7], que consiste em fornecer total acesso aos dados das missões *ERS-1*, *ERS-2*, *GOCE*, *SMOS*, *CryoSat* e a futuras missões de exploração terrestre. Existem missões chamadas "*Third Party Missions*", que correspondem a missões que não pertencem ou não são operadas pela *ESA*, mas cujos dados são recebidos, processados e arquivados por sistemas terrestres da *ESA*. Estes dados estão disponíveis através de acordos com os proprietários dos satélites.

Na maioria dos casos é apenas necessário fazer um registo no *site* da *ESA* para obter livre acesso aos dados de várias missões levadas a cabo pela *ESA* ou com a participação desta. Também existem dados que só ficam disponíveis a pedido. Neste caso pode ser necessário prestar informações sobre o propósito do projeto que está a ser desenvolvido para que os dados sejam disponibilizados.

Os dados resultantes das leituras efetuadas pelos instrumentos dos satélites são posteriormente enviados para bases terrestres da *ESA* que estão responsáveis pela sua receção e processamento. Existem vários níveis de processamento de dados compreendidos entre o nível 0 e o nível 4. O nível 0 corresponde aos dados no seu estado bruto e quanto maior for o nível de processamento maior é o grau de refinamento.

Como todos os dados resultantes das várias missões estão disponíveis ao público em geral existe a necessidade de proceder à sua visualização. Porém, para atingir este objetivo, o utilizador necessita de seguir um processo que pode ser complexo. Desta forma, para visualizar os dados o cliente precisa de encontrar e fazer *download* dos ficheiros que contêm os dados de instalar o *software* compatível com os ficheiros descarregados e, consequentemente, de lidar com a complexidade da utilização destes tipos de *software*.

Este projeto surgiu com a necessidade de reduzir a complexidade associada ao processo de visualização deste tipo de dados, através de uma plataforma interativa de visualização.

1.1 - Objetivos

O principal objetivo deste projeto é a criação de uma plataforma *online* onde seja possível, de um modo interativo, gerar vários tipos de representações gráficas com os dados resultantes das leituras dos vários satélites, facilitando o processo de procura, seleção e visualização desses dados.

Esta plataforma tem o propósito de ser aberta e, assim, poder ser utilizada por todos os utilizadores que estejam interessados em explorar os dados de missões. Mas para que isto seja possível é essencial que os utilizadores possuam um conhecimento básico de informática a nível de utilizador, pois será necessário interagir com a plataforma *online*. Por outro lado, existem os clientes que já estão familiarizados com os dados e sabem exatamente o que pretendem. Neste caso, a plataforma é útil no sentido em que fornece várias ferramentas para a organização e visualização dos dados.

A plataforma de visualização desenvolvida ao longo deste estágio foca-se na exploração dos dados, tema que será debatido no capítulo de visualização de dados. Esta possibilita a visualização das estruturas de dados de uma forma interativa e amiga do utilizador. Foram aplicadas várias técnicas para a visualização dos mesmos e são disponibilizadas várias funcionalidades que passam pela seleção, criação de listas e criação de vários de tipos de gráficos com os dados selecionados.

Para que o objetivo principal seja exequível foi desenvolvido um *web service* com uma API pública. Este serviço permite que sejam realizados vários tipos de pedidos relativos aos dados e informações das missões existentes numa base de dados. Por exemplo, existe um pedido para receber todas as missões existentes ou receber os dados relativos a um instrumento que estejam enquadrados numa janela temporal. Desta forma, esta API pública facilita o acesso aos dados tanto aos utilizadores quanto a outros sistemas informáticos que necessitem de obter dados e informações relativas às missões disponíveis. Este serviço irá ser utilizado pela plataforma de visualização como forma de aquisição dos dados necessários.

Os dados que são utilizados no projeto foram gerados, processados e disponibilizados pelas estações terrestres da *ESA*. Relativamente ao nível de processamento vão ser utilizados os dados com maior nível de processamento disponível para cada instrumento. Apesar da *ESA* realizar várias missões no âmbito deste estágio vão apenas ser estudadas três: *EnviSat*, *CryoSat* e *SMOS*.

1.2 – Estrutura do Documento

Este relatório é composto por um enquadramento sobre as várias missões da ESA estudadas (Capítulo 2), estudo do estado da arte referente à visualização de dados (Capítulo 3), levantamento de requisitos relativamente ao projeto a ser desenvolvido (Capítulo 4), arquitetura desenvolvida (Capítulo 5), desenvolvimento do projeto (Capítulo 6), metodologia adotada e planeamento (Capítulo 7), testes (Capítulo 8), conclusão (Capítulo 9) e referências bibliográficas (Capítulo 10).

Capítulo 2 – Missões de Satélite da ESA

Neste capítulo são apresentadas algumas missões realizadas pela ESA e são analisados os seus instrumentos, assim como, as suas funcionalidades e características. Foram estudadas três missões, sendo elas a *EnviSat*, *Cyosat* e *SMOS*, todas elas com propósitos civis e relacionadas com a pesquisa ambiental do nosso planeta.

2.1 - EnviSat

O satélite *EnviSat (Environmental Satellite)* que dá o nome à missão, é o maior satélite civil de observação da terra, pesa mais de 8 toneladas e é composto por 10 instrumentos a bordo. É considerado o sucessor do satélite ERS (*European Remote Sensing*). [8] Este satélite possui um avançado sistema de imagens através de radares, um radar altimétrico, um radiómetro que permite medir a temperatura, um espectrómetro de média resolução, sensível à cor do oceano e a características da terra e, por fim, dispõe de outros dois sensores atmosféricos para monitorizar gases.

A missão *EnviSat* teve início a 1 de Março de 2002 e terminou, abruptamente, a 8 de Abril de 2012, devido à falha de contacto com o satélite. Houve intenção de retomar a comunicação com o satélite, mas passado alguns meses a equipa responsável declarou que este é irrecuperável [9]. Este satélite foi lançado do Centro Espacial de *Kourou* situado na Guiana Francesa, América do Sul, muito próximo da linha do equador. Devido às suas dimensões e ao seu peso é fácil de perceber que a escolha deste lugar, para o lançamento, teve um grande impacto na poupança de combustível do foguete utilizado para pôr o satélite em órbita.

Os diversos instrumentos *EnviSat* têm as seguintes designações: ASAR, MERIS, AATSR, RA-2, MWR, GOMOS, MIPAS, SCIAMACHY, DORIS e LRR. Os instrumentos MIPAS, GOMOS e SCIAMACHY dedicam-se à leitura dos diferentes constituintes químicos presentes na atmosfera e, juntos, têm a função de dar continuidade ao estudo da camada do ozono, assim como de gases responsáveis pelo efeito estufa do planeta [10]. Estes instrumentos também enriquecem a capacidade de observação sobre o planeta, maioritariamente na deteção de vários elementos químicos e no desenvolvimento do perfil vertical destes constituintes [10].

Com os dados adquiridos por estes instrumentos, a comunidade científica recebe uma vasta coleção de informações e dados relativos aos constituintes da atmosfera, quantidade de água nos solos e erosão costeira, entre outros. Desta forma proporciona o desenvolvimento de estudos sobre a evolução do clima terrestre [10].

2.1.1 - ASAR

O nome deste instrumento é a abreviatura de *Advanced Synthetic Aperture Radar* e tem como objetivo assegurar a leitura de informação de alta resolução sobre os segmentos de terreno em que as ondas do radar incidem. Este é uma evolução dos instrumentos AMI *(Active Microwave Instrument)* e SAR *(Synthetic Aperture Radar)* das missões *ERS-1* e *ERS-2* [11].

Os resultados das suas medições podem ser utilizados para construir imagens onde é possível ver a estrutura de tornados, tempestades e a altura de ondas. Estas informações podem ser úteis para o planeamento costeiro, possibilitam o estudo de tempestades, e

facilitam a gestão de operações pesqueiras, a monitorização de poluição, a deteção de embarcações e derrames de óleo no oceano, entre outras [11].

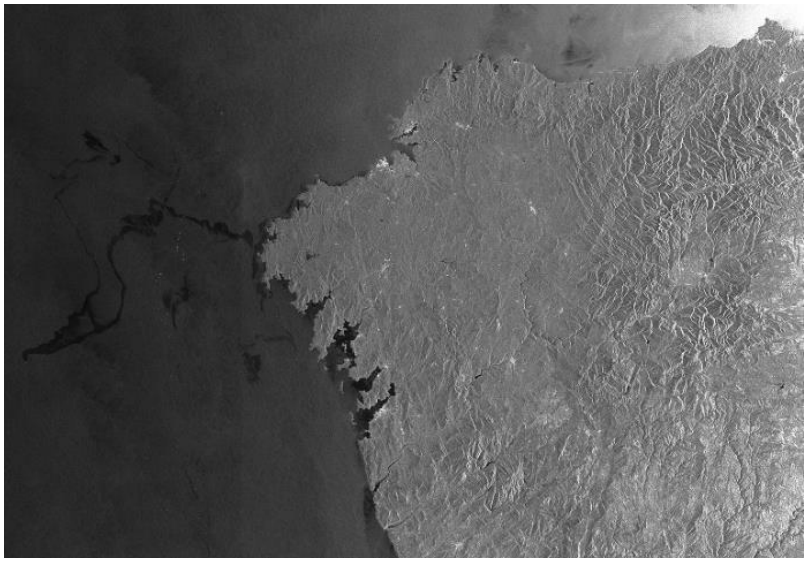


Figura 1 – ASAR, Derramamento de óleo do petroleiro *Prestige* na Galiza em 17 de Novembro de 2002 [12].

2.1.2 - MERIS

Este instrumento, *Medium Resolution Imaging Spectromter*, dedica-se à captura de imagens. Possui 15 bandas de espectro (390nm até 1040nm) que podem ser selecionadas por comandos provenientes de uma base em terra. Permite medir a cor do mar e, assim, realizar perfis de clorofila presentes no oceano. Desta forma, torna-se possível aos cientistas elaborar mapas de clorofila no oceano, assim como perceber a redução ou aumento desta. Também permite medir a vegetação presente em terra, viabilizando a elaboração de mapas onde a desflorestação é mais acentuada e onde é preciso atuar para conseguir reduzir esta desflorestação. Permite ainda perceber o ciclo do carbono no oceano, a dinâmica do oceano, realizar estudos do clima, gestão das zonas costeiras, entre outras [13].

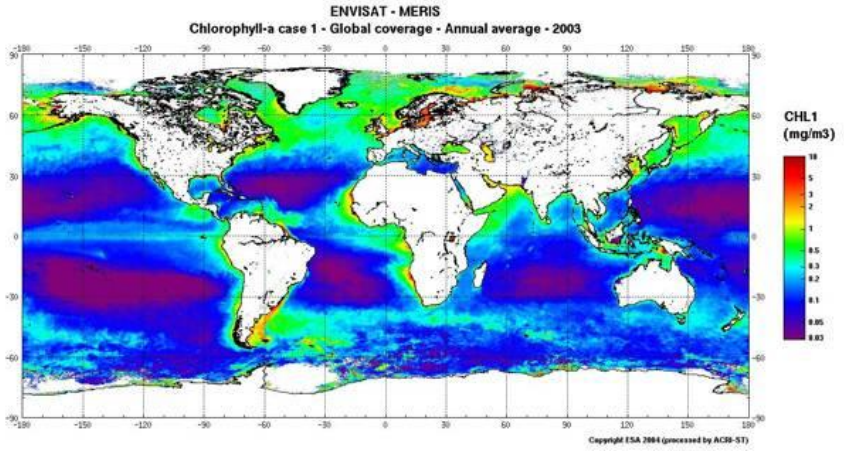


Figura 2 – MERIS, Produto nível 3 – Clorofila média anual 2003 [13].

2.1.3 - AATSR

O principal objetivo do *Advanced Along Track Scanning Radiometer* - AATSR é medir, com precisão, a temperatura da superfície dos mares para pesquisa climática [14].

Os oceanos possuem correntes de água fria, assim como correntes de água quente. Com este instrumento é possível monitorizar a temperatura destas correntes ao longo de vários anos e, assim, com os dados obtidos, realizar estudos sobre o aumento ou diminuição das temperaturas nestas correntes marítimas. A comunidade científica tem grande interesse na ligação entre a temperatura da superfície do mar e alguns fenómenos climáticos. Tal acontece porque existe uma grande quantidade de energia que é transportada pelos oceanos e que pode ser transferida para a atmosfera, dando origem, por exemplo, a tempestades [14].

2.1.4 - RA-2

O *Radar Altimeter 2* é um radar de alta precisão usado para medir a potência das ondas de radar refletidas pela superfície terrestre. Assim sendo, este instrumento mede com precisão o tempo que a onda de radar demora desde que é emitida até ser recebida de novo pelo sensor. Deste modo, consegue medir a forma das superfícies que refletem as ondas que emite (terra, mar ou gelo). Com este instrumento é possível realizar estudos científicos sobre a altura do nível do mar, velocidade do vento e espessura da camada à superfície dos blocos de gelo presentes no mar [15].

2.1.5 - MWR

O principal objetivo do instrumento radiómetro de micro-ondas, MWR, consiste em medir o vapor de água na atmosfera e a quantidade de líquido nas nuvens. Com os dados deste instrumento é possível fazer correções nos dados medidos pelo instrumento RA-2, na medida em que fornece informações sobre a propagação de ondas na atmosfera [15].

2.1.6 - GOMOS

É um instrumento que mede os diferentes constituintes da atmosfera pela análise do espectro. Para realizar estas medidas é utilizada uma técnica designada de "ocultação de estrelas", em que é feita uma análise espectral de várias bandas do espectro eletromagnético provenientes da estrela que está a ser ocultada. Desta forma, são analisadas as bandas do espectro de 250nm a 675nm, 756nm a 773nm e de 926nm a 952nm. Também possui dois fotómetros que operam, respetivamente, no espectro de 470nm a 520nm e 650nm a 700nm. Estes comprimentos de onda possibilitam a medição de O_3 (ozono), NO_2 (Dióxido de nitrogénio), NO_3 (nitrato), O_2 (oxigénio), H_2O (água) [16].

O método de "ocultação das estrelas" consiste na captação de uma vasta gama do espectro que é emitido por uma só estrela durante a órbita do satélite. O perfil de constituintes da atmosfera é obtido através da medição da intensidade do espectro recebido pelo instrumento, uma vez que, os raios de luz, ao atravessarem a atmosfera terrestre, sofrem vários tipos de processos de refração e absorção pelos elementos químicos que estão presentes na atmosfera. Conhecendo as propriedades químicas destes elementos torna-se possível obter os constituintes presentes na atmosfera através da comparação dos diferentes tipos de espectro que foram recebidos pelo instrumento [16].

Quando o satélite está acima dos 120km (termosfera) começa a receber o espectro emitido pela estrela. Nesta fase, o espectro não sofre alterações (i.e processos de absorção ou refração). À medida que o satélite continua na sua órbita descendente, a linha de visão com a estrela vai entrando cada vez mais na atmosfera terrestre até esta estrela estar totalmente oculta. Isto acontece quando, pela órbita do satélite, a estrela que está a emitir a luz é ocultada pelo planeta terra. O brilho das estrelas é um fator muito importante, quanto mais brilhante for a estrela, maior é a qualidade dos dados obtidos pelo instrumento. A *ESA* possui uma lista das 300 estrelas mais brilhantes para o uso deste instrumento. Durante o ano de 2003, 177 estrelas diferentes foram utilizadas para as medições [16].

A cobertura vertical fornecida pelo instrumento ocorre normalmente entre os 120km (termosfera) e os 11km (fim da troposfera). Este limite inferior varia de ocultação para ocultação, dependendo das características da estrela, do estado da atmosfera e do estado da iluminação. Desta forma, é possível realizar o perfil de constituintes químicos da atmosfera nesta janela. Quando a órbita do satélite é feita durante a parte iluminada da terra, as medições são fortemente influenciadas pela luz do sol que é refletida pela atmosfera, o que resulta em amostras de baixa qualidade. Por outro lado, quando as leituras são efetuadas não existindo luz solar, as medições possuem uma melhor qualidade [16].

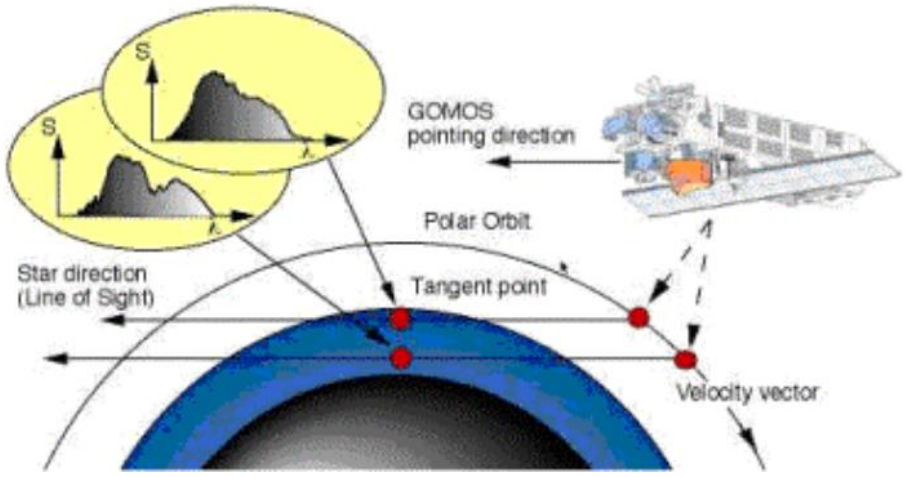


Figura 3 – GOMOS, método "ocultação das estrelas" [16].

2.1.7 - MIPAS

Este instrumento tem o objetivo de medir as emissões de gases que circundam a terra, através de um espectrómetro que opera na região dos infravermelhos onde os principais gases têm emissões significativas. Desta forma, é possível medir os elementos químicos da parte superior da troposfera até à parte inferior da mesosfera (H_2O, CH_4, CO, CO_2, O_3 e NO). Também fornece perfis verticais sobre a temperatura atmosférica [17].

2.1.8 - SCIAMACHY

SCIAMACHY é um espectrómetro que tem como principal missão realizar uma medição global de gases na troposfera, estratosfera e mesosfera. As medições são feitas pela radiação transmitida, defletida e refletida na atmosfera. Este instrumento permite ajudar no estudo dos processos físicos e químicos que ocorrem na atmosfera e, com isto, perceber o seu comportamento. Existem vários componentes químicos na atmosfera que modificam o estado da luz que é recebida pelo espectrómetro do instrumento. Assim, através da sua

assinatura espectral, é possível saber a quantidade e distribuição de um número significativo destes constituintes. Com este instrumento deteta-se vários componentes químicos existentes na atmosfera, sendo estes: O_2, O_3, O_4, NO, NO_2, NO_3, CO, CO_2, HCHO, CH_4, H_2O, N_2O, SO_2, BrO, OClO [18].

2.1.9 - DORIS

O instrumento DORIS, *Doppler Orbitography and Radio-positioning Integrated by Satellit*e, é utilizado para obter a localização precisa do satélite. É composto por um sistema de posicionamento por micro-ondas e funciona em simultâneo com uma rede de emissores terrestres [19].

2.1.10 - LRR

Este é um dispositivo passivo, *Laser RetroReflector*, usado como refletor de laser pelas estações em terra. Desta forma, é possível obter uma medição rigorosa da posição do satélite [20].

2.2 - CryoSat

A missão *CryoSat* tem como objetivo o estudo ambiental e dedica-se à investigação do gelo e da sua espessura nas calotas polares (regiões com maior latitude do planeta). Para realizar a sua tarefa, este satélite utiliza um avançado sistema de radar altimétrico especialmente desenvolvido para monitorizar a criosfera (regiões da superfície terrestre que estão permanentemente cobertas por uma camada de gelo ou neve, incluindo as calotas polares). Para medir a espessura do gelo que está a flutuar no mar é utilizada uma técnica que permite medir a altura do gelo que se encontra acima da superfície do mar através do radar altimétrico. O satélite responsável por esta missão possui o nome de *Cryosat-2* e foi lançado em Abril de 2010 na estação *ISC Kosmotras* (*The International Space Company Kosmotras*). Esta missão tem como duração mínima 3 anos, pelo que a esta data ainda está a decorrer e é composta pelos instrumentos SIRAL e DORIS [21].

2.2.1 - SIRAL

O instrumento SIRAL, *SAR Interferometric Radar Altimeter*, é baseado em ondas de radar, construído especificamente para medir a espessura do gelo que está a flutuar nos mares [21].

2.2.2 - DORIS

O instrumento DORIS, *Doppler Orbitography and Radio-positioning Integrated by Satellite*, é utilizado para obter a localização precisa do satélite. É composto por um sistema de posicionamento por micro-ondas e funciona em conjunto com uma rede de emissores terrestres [21].

2.3 SMOS

A missão SMOS, *Soil Moisture Ocean Salinity,* dedica-se a medir a humidade dos solos e a salinidade dos oceanos. Assim sendo, este satélite utiliza um sistema que capta emissões de micro-ondas provenientes da superfície da terra e faz o mapeamento de humidade do solo e de salinidade dos oceanos. O nível de salinidade dos oceanos é importante para o estudo e compreensão do clima terrestre, sendo um fator essencial para determinar a densidade da água que, por outro lado, influencia as correntes marítimas. Estas correntes são importantes para o clima, na medida em que existe uma transferência de energia (calor) entre o oceano e a atmosfera [22].

A salinidade e a temperatura contribuem para uma maior ou menor densidade da água. Desta forma, quanto mais fria e salgada for a água, maior é a sua densidade. Assim, a densidade da água do mar influencia de uma forma essencial as correntes oceânicas. Quando a salinidade da superfície da água aumenta, esta torna-se mais densa e, por sua vez, afunda-se provocando a circulação de água no mar e, juntamente com o vento e a rotação terrestre, dando origem às correntes marítimas [22].

O estudo da quantidade de água presente nos solos é importante, pois esta está relacionada com clima do planeta. Por isso, é fundamental perceber o ciclo da água (marcado pela evaporação e precipitação) para conseguir determinar eventos como cheias, tempestades, o crescimento da vegetação, entre outros [22].

Este satélite foi lançado a 2 de Novembro de 2009 e a sua missão terá uma duração de pelo menos 3 anos. De momento o satélite ainda está operacional [23].

2.3.1 - MIRAS

Este instrumento é apelidado de MIRAS, *Microwave Imaging Radiometer using Aperture Synthesis*, e tem como objetivo capturar a radiação de micro-ondas emitidas pela superfície da terra em frequências próximas de 1.4GHz. A humidade do solo e a salinidade dos mares diminuem a emissão de micro-ondas. Desta forma, a emissão de micro-ondas da superfície da terra é alterada. Com este instrumento, é possível fazer o mapeamento das zonas com mais humidade e com mais salinidade nos oceanos [24] [22].

2.4 – Ferramentas

São várias as ferramentas utilizadas para visualizar os dados enviados pelos instrumentos a bordo dos satélites. Estes instrumentos fazem diferentes tipos de leituras e, desta forma, geram diferentes tipos de dados. Os dados são recebidos por estações terrestres da *ESA*, onde são processados e armazenados. A informação gerada é guardada em ficheiros com formato e extensão próprios (temos o exemplo do ficheiro ".N1" do satélite *EnviSat*).

Como cada missão possui um formato específico de ficheiro, indicado pela extensão correspondente, existem também pacotes de *software* específicos para abrir e processar estes ficheiros, que normalmente são utilizados pelos cientistas que utilizam os dados. Cada ferramenta é desenvolvida com intuito de abrir os ficheiros de uma missão específica, o que resulta na existência de vários tipos de *software* para as várias missões, não sendo estes compatíveis com os outros tipos de ficheiros gerados por outras missões. Estas ferramentas são desenhadas e desenvolvidas especificamente para uso científico, assim sendo, um utilizador normal irá despender tempo para perceber como o *software* funciona e para a sua ambientação. De notar que estes programas não são fáceis nem amigáveis para o

utilizador normal. Assim, verifica-se a existência de uma curva de aprendizagem pronunciada para a utilização deste tipo de ferramentas.

Todos os pacotes de *software* aqui estudados possuem uma licença gratuita e estão disponíveis para *download* no site da ESA. Desta forma, não é necessário proceder à compra nem ao licenciamento de nenhum tipo de ferramenta.

A discussão que se segue refere-se apenas aos ficheiros com maior nível de processamento disponíveis no *site* da ESA. Como foi referido anteriormente, todos os dados das missões estão disponíveis ao público em geral, sendo necessário um registo no *site*. Porém, existem alguns dados em que é necessário aprovação para a sua utilização. Neste caso, é necessário fazer um registo e submeter a descrição do projeto que está a ser desenvolvido.

2.4.1 - EnviView

O *EnviView* (*Envisat View*) é uma ferramenta de licença livre que permite aos utilizadores ler a maior parte dos dados provenientes dos instrumentos do satélite *EnviSat*. Este *software* permite ler ficheiros com a extensão ".N1" ou ".MPH" (*main product header*), que possuem informação básica sobre algumas características das medições efetuadas, como por exemplo, a data que foi efetuada a medição. Esta ferramenta também consegue exportar dados para o tipo de ficheiro *.HDF* para que sejam processados por outro programa. Apesar de possuir uma interface gráfica relativamente simples, é necessário dedicar e despender algum tempo para conseguir perceber e trabalhar com esta ferramenta. Como foi referido anteriormente, esta é uma ferramenta de uso científico.

As principais funcionalidades desta aplicação são:

- A apresentação das leituras efetuadas pelo instrumento, em *ascii*, assim como a descrição de cada campo;
- Apresentação do ficheiro em hexadecimal;
- Possibilidade de ver a posição do satélite em duas dimensões, no momento em que este realizou a medição;
- Apresentação das imagens obtidas pelos instrumentos do satélite;
- Visualização de dados em vários tipos de gráficos, como por exemplo, gráficos a duas dimensões, representação de *array*, gráficos circulares e gráficos a três dimensões;
- Obtenção de ajuda sobre os tipos de dados existentes, ou seja, de uma descrição de toda a informação dos campos existentes nos ficheiros de todos os instrumentos da missão *EnviSat*.

Com esta ferramenta é possível abrir a informação de todos os instrumentos da missão *EnviSat* com a exceção dos instrumentos RA-2 e MIPAS.

Esta aplicação detém como ponto forte o facto de possuir capacidade de apresentar os dados existentes no ficheiro de forma simples e eficaz. Isto significa que os dados podem ser vistos em formato de texto e, quando os gráficos são apresentados, possibilitam uma fácil compreensão.

Por outro lado, sublinha-se alguns pontos fracos existentes nesta ferramenta. Estes recaem sobre o tempo de aprendizagem necessário para a utilização e ambientação deste *software* e a existência de alguns *bugs* quando estamos a selecionar informação para apresentar em gráficos. Quando acontece este problema, deixa de ser possível a seleção de informação para desenhar novos gráficos e a aplicação tem de ser reiniciada para voltar ao estado normal.

2.4.2 - BEAM

A aplicação BEAM é utilizada para processar, ver e analisar os dados resultantes das medições dos instrumentos do satélite *EnviSat*. Inicialmente foi criada para facilitar a visualização das imagens obtidas pelos instrumentos óticos do satélite, mas o desenvolvimento da aplicação foi continuado e foram criados vários Plug-ins para possibilitar a sua utilização com outro tipo de ficheiros, assim como, outro tipo de missões.

O BEAM possui uma interface gráfica que facilita a sua utilização. No entanto, como é uma aplicação científica, é necessário investimento de tempo para a sua correta utilização. Um ponto a referir sobre esta aplicação é que também é *open-source*. Assim sendo, não é necessário pagar uma licença para a sua utilização.

Com esta ferramenta é possível realizar as seguintes tarefas:

- Ver as faixas de terreno onde foram realizadas as leituras por parte dos instrumentos da missão, recorrendo ao mapa do planeta terra a cores. Possibilidade de interagir com este mapa, fazendo *zoom in* e *zoom out*;
- É disponibilizada a funcionalidade de abrir os ficheiros e visualizar a informação existente em ASCII (texto);
- Abrir as imagens existentes no ficheiro resultantes das medições e realizar algumas operações nas mesmas, como por exemplo, exportar a imagem ou área selecionada em vários tipos de formato (tiff, pmg, jpg, png). Existe a possibilidade de exportar para imagem a legenda da mesma;
- Realizar estatísticas, histogramas, gráficos a duas dimensões e obter informação da geolocalização da imagem.

Esta ferramenta dirige-se à visualização e processamento de imagens. Assim sendo, não é compatível com todos os ficheiros resultantes de todas as medições dos instrumentos do satélite *EnviSat*. Apenas os instrumentos ASAR, AATSR e MERIS são indicados para esta ferramenta. Quando são abertos os dados do instrumento DORIS, somente é possível aceder às leituras, como por exemplo, das coordenadas da posição do satélite. Desta forma, não é permitido realizar qualquer tipo de gráfico ou estatística sobre estes dados.

Relativamente aos pontos fortes, é de salientar o seu vasto leque de opções de representação em gráficos de vários tipos, assim como, a boa apresentação gráfica de todos os gráficos. Por outro lado, um ponto fraco desta ferramenta consiste em não oferecer de nenhuma ajuda sobre os campos existentes nos ficheiros, nem informações sobre os tipos de ficheiro dos instrumentos.

Em comparação com o *Enviview* existem neste *software* mais funcionalidades para trabalhar com imagens e também possui uma melhor componente gráfica.

2.4.3 - NEST

Next ESA SAR Toolbox - NEST é uma ferramenta *open-source* utilizada para ler, processar, analisar e visualizar os dados dos instrumentos de leitura de radar como, por exemplo, o ASAR. Esta aplicação foi desenvolvida utilizando a ferramenta BEAM.

Com este *software* é possível realizar uma série de tarefas:

- Visualizar num globo a três dimensões, ou a duas dimensões, a trajetória do satélite. Desta forma, torna-se possível verificar a zona onde o instrumento fez as medições;
- Utilizar a notação de grafo para fazer uma sequência de operações sobre os dados do ficheiro;
- Processar os dados de forma a identificar objetos, derramamento de petróleo ou direção do vento;
- Visualizar os cabeçalhos do ficheiro de dados em ASCII (texto);
- Dispor os dados em gráficos, como por exemplo, histogramas. Também existe a possibilidade de realizar estatísticas.

Este *software* tem como foco a área científica, pelo que é necessário despender tempo para aprender como funciona e para a sua familiarização. É uma ferramenta que se dedica apenas às leituras de radar, não sendo compatível com os dados dos outros instrumentos da missão *EnviSat*. Possui uma boa interface gráfica, onde a visualização dos dados, como por exemplo, a órbita do satélite num mapa em três dimensões, está bem conseguida. Relativamente aos restantes gráficos são iguais aos do BEAM.

2.4.4 - CUT

O *CryoSat User Tool* (CUT) é uma ferramenta FTP (*File Transfer Protocol*) utilizada para fazer *download* dos ficheiros da missão *CryoSat* processados pela ESA. Esta ferramenta é compatível com o formato ".HDR" (*XML Header File*) e através deste ficheiro é possível fazer *download* do respetivo ficheiro de informação ".DBL" (*Product File*) que corresponde ao ficheiro final processado pela ESA contendo as medições efetuadas pelo satélite.

Este *software* fornece a representação globo onde é possível ver a órbita do satélite e a data correspondente a esta órbita. Podemos ainda verificar a representação gráfica no globo terrestre da zona de leitura do satélite (faixas de terreno que foram mapeadas). É também possível desenhar áreas no globo terrestre com o propósito de obter a informação sobre a mesma área. Desta forma, torna-se possível aceder aos ficheiros disponíveis para *download* das faixas de terreno selecionadas pelo utilizador.

Quando conectada ao servidor de FTP a ferramenta fica muito lenta. Possivelmente tal acontece devido à quantidade de informação existente no servidor de FTP que está a ser transferida para o computador do utilizador. Juntando isto à informação representada no globo terrestre existe uma grande latência ao utilizar o *software*.

Ao recorrer a alguns ficheiros previamente descarregados do *site* da ESA, a ferramenta mostra-se mais fluída. No entanto, apenas é possível ver a órbita do satélite, a data da leitura e as áreas onde a leitura foi efetuada. Todas estas informações estão representadas no globo terrestre. A representação do globo terrestre nesta ferramenta está muito bem conseguida, existindo também a possibilidade de alteração do modo de visualização do

globo. Da mesma forma estão bem representadas as órbitas do satélite e o terreno que foi mapeado por este.

2.4.5 - Cryoview

Com esta ferramenta dedicada à leitura dos dados da missão *CryoSat* é possível abrir os ficheiros de dados do tipo ".DBL", desenhar gráficos, mostrar o conteúdo em tabelas ou em imagens. Este *software* não é fácil de usar nem é amigável. Existe uma interferência na sua utilização, uma vez que, constantemente mostra a informação que ocorreu um erro e, por vezes, é necessário fechar e voltar a ligar a aplicação. Quando esta é reiniciada é necessário procurar de novo os ficheiros que queremos utilizar, pois a árvore de ficheiros que é mostrada inicialmente não corresponde ao local onde os ficheiros estão armazenados.

2.4.6 - BEAM - SMOS Plugin

Para ler os ficheiros processados pela ESA das medições relativas à missão SMOS existe um *plugin* disponível para a ferramenta BEAM (utilizada para abrir os ficheiros do *EnviSat*). Este *plugin* tem o nome de "SMOS *Plugin*" e encontra-se disponível para *download* de forma gratuita. Depois de instalado, este *plugin* torna possível a utilização da mesma ferramenta (BEAM) mas com utensílios extra, que possuem a característica de abrir os ficheiros com extensão ".DBL" (ficheiros que contêm os dados das leituras) ou ".HDR" (ficheiro *header*) desta missão.

É uma ferramenta científica que incorpora uma interface gráfica, que permite ver o terreno onde foram efetuadas as leituras do satélite, visualizar informação relativa a cabeçalhos, exportar as imagens para outros formatos (tiff, pmg, jpg, png), realizar estatísticas, histogramas, gráficos a duas dimensões, entre outros.

Com esta extensão à ferramenta é possível visualizar, em hexágonos coloridos no mapa do planeta terra (através do *SMOS Snapshot*), a informação captada de vários ângulos pelo instrumento deste satélite (MERIS). Também é possível realizar gráficos e tabelas específicas com a informação de cada hexágono representado. Com estes é permitido ver dados dos diferentes ângulos incidentes responsáveis pela medição, assim como, informação de polarização.

2.4.7 – Ferramentas de visualização de dados *online*

Existem alguns *websites* que representam informação de vários satélites:

www.n2yo.com – Neste *website* é possível ver em tempo real a órbita de satélites onde é fornecida a informação relativa à latitude, longitude, altitude, velocidade, entre outras. Quando abrimos o *website* verificamos que estão a seguir a órbita da Estação Espacial Internacional (*ISS – Internacional Space Station*). Podem ser pesquisados vários satélites, como por exemplo o *EnviSat* e o *CryoSat*. Embora a representação da órbita do satélite esteja bem conseguida e apresente animações utilizando como ferramenta a API do *GoogleMaps,* o *website* limita-se a representar apenas a órbita do satélite e a fornecer previsões desta mesma órbita, não estando disponíveis mais nenhuns dados.

www.heavens-above.com – Neste *website* também é possível ver as órbitas de alguns satélites, em imagens, sendo necessário atualizar o *site* para conseguir ver o progresso do

satélite na sua órbita. Neste há a possibilidade de obter informações sobre astronomia, que incluem cometas, asteroides, constelações, entre outras.

www.rammb.cira.colostate.edu – Fornece algumas imagens de satélite em tempo quase real. Neste *website* é possível ver uma sequência de imagens que cria uma animação.

2.4.8 – Comentários finais

Nesta secção foram analisadas várias ferramentas de caráter científico utilizadas para a visualização dos dados gerados por diversos instrumentos de várias missões. Relativamente às ferramentas utilizadas para visualizar os dados do satélite *EnviSat*, a que possui melhor desempenho é a aplicação *EnviView*. Esta demonstra uma maior compatibilidade com todos os dados, embora exista a ocorrência de alguns *bugs* e não contém uma interface gráfica superior quando comparado com outros pacotes de *software (*BEAM e NEST*)*. Quanto às missões *CryoSat* e SMOS apenas possuem uma ferramenta onde é possível a visualização dos seus dados.

Os *web sites* analisados baseiam-se na visualização da órbita de vários satélites podendo o utilizador selecioná-los conforme o seu interesse. Alguns destes *sites* disponibilizam imagens sobre a terra em tempo quase real.

Em suma, podemos verificar a existência de uma lacuna na *Internet* no que toca à visualização de dados de missões. Desta forma, o objetivo deste estágio consiste em colmatar a inexistência deste tipo de plataformas criando um *site* onde é possível ver e explorar os dados resultantes das medições de diversos instrumentos, sob várias formas (p.ex. gráficos).

Capítulo 3 – Visualização de Dados

Os computadores são uma poderosa ferramenta de análise de dados, permitindo processar e apresentar uma vasta gama de informações graficamente. Esta combinação de poder de processamento e visualização gráfica torna-os a ferramenta ideal para pesquisa científica através da análise de dados. A visualização de dados pode ser considerada como a combinação de pontos, linhas, sistemas de coordenadas, números, símbolos, palavras, sombras e cores, de forma a representar informação tornando visível padrões e tendências existentes nas relações entre as variáveis que estão sendo relacionadas [25].

A quantidade de dados gerados e guardados em ambientes organizacionais e em pesquisas científicas está constantemente a crescer. Os dados que são captados têm origem em diferentes fontes, como por exemplo os dados resultantes das vendas ou de sistemas de monitorização baseados em sensores (sensores estes que podem estar presentes nos instrumentos das missões realizadas pela ESA ou por outras agências espaciais) resultando, assim, num conjunto de dados multidimensionais ou em estruturas não numéricas, como é o caso de documentos *web* [26].

3.1 - História

Ao contrário do que normalmente pensamos, a representação gráfica de dados e estatísticas não é um assunto moderno. A história da representação gráfica remonta aos primórdios onde foram desenvolvidos e apresentados de forma visual os primeiros mapas. Posteriormente ocorreu o surgimento da cartografia temática, o aparecimento das estatísticas ligadas às ciências, como por exemplo, a área das ciências médicas. Esta beneficiou com o crescimento do pensamento estatístico devido ao aumento da quantidade de dados provenientes do comércio e do planeamento que ocorreu no século XIX. Estes fatores juntamente com avanços na matemática e com o surgimento de técnicas de representação gráfica contribuíram significativamente para a visualização de dados moderna [27].

As primeiras representações gráficas consistiram em diagramas geométricos, em tabelas com as posições de corpos celestes e mapas que ajudavam a navegação e exploração. Na figura 4, datada do século X, é possível visualizar um diagrama, em séries temporais, onde são mostrados a posição de sete corpos celestes em relação ao tempo [27].

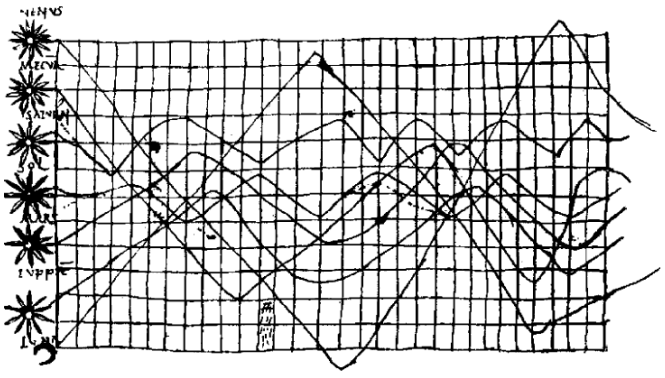


Figura 4 – Representação de corpos celestes datada do século 10 [27].

Durante o século XVIII, na área da cartografia, os cartógrafos começaram a inovar e a procurar novas formas de representar mais informações nos mapas. Assim, surgiu a cartografia temática, ramo que trata dos aspetos matemáticos relacionados com a conceção dos mapas. Na figura 5 é apresentado um exemplo de um mapa onde são representadas as linhas isógonas (linhas do mesmo sentido do vento) [27].

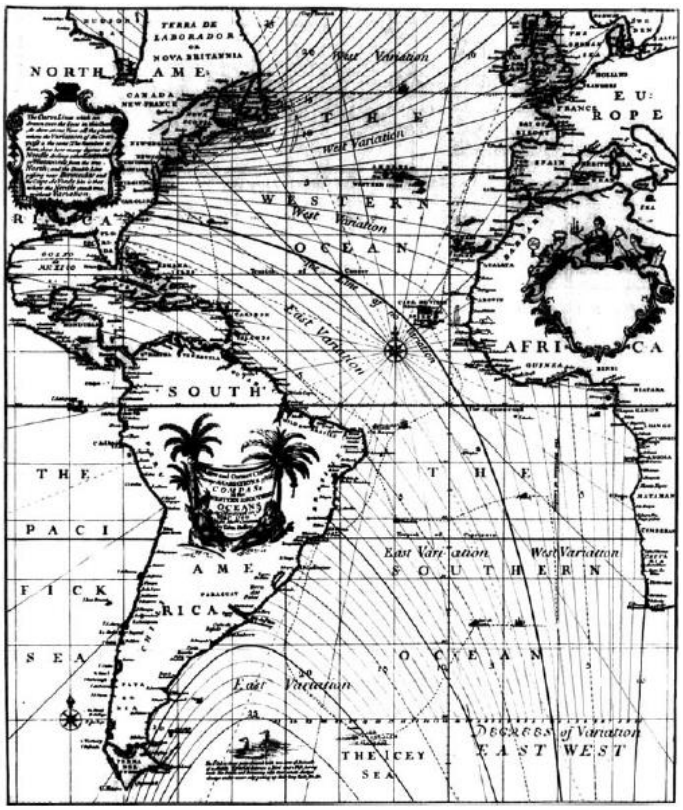


Figura 5 – Representação no mapa de linhas isógonas [27].

A partir de 1975, último quarto do século XX, a visualização de dados atingiu uma grande dimensão transformando-se numa área de pesquisa multidisciplinar. Neste período e até ao presente surgiram várias ferramentas de *software* com capacidade de representar uma grande variedade de técnicas de visualização de dados e, assim, estes colocam todo o seu potencial à disposição do público em geral através da utilização do computador pessoal. Embora seja difícil sublinhar os maiores progressos e desenvolvimentos no que toca à visualização de dados, é possível verificar alguns destaques [27]:

- O desenvolvimento de linguagens de programação para estatística, como por exemplo, as linguagens S e APL;
- A criação de novos métodos de visualização de dados, nomeadamente métodos de visualização multidimensional como por exemplo, a técnica de coordenadas paralelas ou SPLOM (matrizes de gráficos de dispersão);
- A reinvenção de técnicas de representação de variáveis discretas e categóricas;
- Aumento da importância dada às questões cognitivas e percetuais no que toca à visualização de dados.

Desta forma, é possível verificar que a evolução desta área surgiu com a necessidade de visualizar fenómenos e relações de acordo com novas técnicas ou sob diferentes perspetivas. A visualização de dados teve um grande impulso devido ao enorme desenvolvimento na área da tecnologia, aparecimento de novos métodos de coleta de dados e com o desenvolvimento da estatística.

3.2 - Importância da Visualização de Dados

A visualização de dados tem um papel fundamental na interpretação teórica dos dados e é útil em auxiliar pesquisadores na busca de modelos matemáticos apropriados. Também possibilita a exploração e análise de extensas quantidades de dados, viabilizando a descoberta de relações de todos os tipos presentes nos dados. Esta análise é conseguida através da geração de uma ou mais janelas de visualização [25].

Através da visualização de dados é possível transmitir rapidamente uma enorme quantidade de informação utilizando o poder do cérebro humano para identificar padrões, relações e perceber o seu significado. Também ajuda a levantar novas questões ou identificar outros problemas [28].

Os principais objetivos para a visualização de dados estão relacionados com a monitorização de sistemas, verificação de soluções que se adaptam a uma determinada situação, verificar dados (pontos dos dados), encontrar dados isolados (*outliers*), mostrar tendências, suportar argumentos ou simplesmente obter uma visão geral dos dados [28].

3.3 - Propósitos da Visualização de Dados

Nesta secção irão ser debatidos os principais motivos para a visualização de dados que consistem na exploração ou na apresentação dos mesmos.

Quando o intuito é a apresentação dos dados existe a necessidade de decidir sobre qual a informação a ser exibida e a criação de uma janela (gráfico) apropriada para o conteúdo e, consequentemente, ter em conta o público-alvo da apresentação. É necessário considerar se o gráfico vai transmitir a informação que é pretendida, isto é, se a audiência vai entender o gráfico do mesmo ponto de vista que o apresentador quer transmitir. Existem tipos de representações gráficas pouco comuns e difíceis de interpretar exigindo uma experiência prévia para que a informação seja corretamente assimilada [29]. Desta forma, sistemas de visualização eficazes possuem a característica de serem fáceis de ler e perceber [25]. Por exemplo, verifica-se que um leitor que está familiarizado com um tipo de gráfico, não vai ter nenhum problema a interpretar um exemplo diferente do mesmo gráfico. Por outro lado, quando lhe é apresentado um novo tipo de gráfico, podem surgir várias dificuldades na sua interpretação [29].

A exploração de dados surge com a necessidade de analisar, visualmente, vários tipos de dados ou informações. Para que isto seja possível é necessário, em primeiro lugar, procurar os dados que podem estar inseridos em diferentes estruturas (por vezes complexas). Desta forma, a exploração de dados trata-se de um assunto subjetivo, onde podem ser aplicados vários tipos de gráficos e efetuar várias operações sobre estes. Assim, é possível modificá-los, descartá-los e criar novas versões, a fim de procurar informações relevantes (como por exemplo, tendências e padrões) ou encontrar novas ideias. Por consequência destes fatores, nenhum gráfico em particular é importante e, normalmente, o seu tempo de vida é curto [29].

Para que seja possível a extração dos dados contidos em estruturas de dados (em alguns casos, são estruturas de dados complexas), numa primeira fase, é necessário representá-las graficamente. Este conceito de exploração de dados é importante pois o ser humano possui a facilidade de, visualmente, compreender a estrutura e o relacionamento dos dados [26].

3.4 - Planificação e escolha do tipo de gráficos

Existem inúmeros tipos de gráficos que podem ser utilizados para a visualização, contudo o tipo de dados tem um peso fundamental nesta escolha. Desta forma, para variáveis contínuas o gráfico do tipo histograma é uma boa escolha e para variáveis categóricas podem ser utilizados os gráficos de barras ou gráficos circulares. Por outro lado, os dados contínuos univariados não podem ser representadas em gráficos circulares e dados categóricos bivaridados (de duas variáveis) não podem ser demonstradas utilizando diagramas de caixa. Relativamente à transformação ou agregação dos dados vai depender da distribuição dos dados e do objetivo do gráfico. Embora não exista uma escolha ótima no que toca à forma como os dados irão ser representados, este aspeto é de extrema importância. No entanto, existem diversos modos de representação que enfatizam diferentes características podendo ser igualmente adequados [29].

A existência de informações adicionais é considerada útil para enfatizar alguns pormenores nos gráficos, como é o caso das legendas e anotações. Por exemplo, a legenda explica a cor ou tipo de linha que representa uma informação específica. Estas devem estar inseridas no próprio gráfico, sendo necessário prestar atenção à sobreposição de informação, pois podem levar a representações confusas e impercetíveis. A presença de um título é importante pois tem a função de explicar o gráfico e de fornecer a origem dos dados. Porém o título não deverá ser extenso porque pode levar a que o leitor não o leia. Quando existe um texto explicativo de um gráfico, este deve estar junto do mesmo, pois torna-se inconveniente para o leitor tentar perceber o gráfico e estar a ler o texto que se encontra noutra página ou longe da imagem [29].

O tamanho das representações gráficas deve ser o suficiente para que a informação seja percetível. Este fator é uma aproximação e depende muito da informação circundante à janela onde é apresentado o gráfico [29].

3.5 – Cores

A escolha de um tipo de cor para a representação de dados é uma das formas mais eficazes de visualização. Porém, é a mais difícil de concretizar com sucesso. Quando uma cor é escolhida para representar algum tipo de dados, é preciso ter em consideração a existência de cores associadas a outros fatores, como por exemplo, a cor vermelha está associada ao calor ou a coisas perigosas. Outro espectro a ter em consideração reside no facto de existirem pessoas daltónicas, incapazes de visualizar a cor que está representada [29].

O tipo de cor também é uma característica muito utilizada no sentido de acrescentar uma nova dimensão na representação gráfica. Por exemplo, se existe um gráfico que realiza o mapeamento de três dimensões, quando são adicionadas novas cores para fornecer outras informações nos pontos, é adicionada uma quarta dimensão ao gráfico [30].

3.6 – Glifos

Glifo ou ícone é uma figura que associa uma característica a um símbolo, podendo possuir vários atributos, como por exemplo, uma forma, tamanho ou cor. A utilização de glifos na visualização de dados foca-se nas capacidades de perceção humanas. Assim sendo, podem ser utilizados diferentes tipos de glifos para representar informações diferenciadas nos gráficos e, com isto, pode ser acrescentada uma nova dimensão à janela de visualização de dados [30].

A utilização de glifos é uma das técnicas utilizadas para a representação de dados em contexto multidimensional. Contudo, existe o problema relativo ao número de variáveis que podem ser codificadas utilizando esta técnica, pois, ao utilizar várias fontes de dados pode resultar numa diminuição na capacidade de perceção devido à sobreposição dos glifos [31].

3.7 - Grafos

A visualização utilizando grafos é útil uma vez que permite representar as relações entre diferentes objetos, tornando possível a construção gráfica das relações dos diferentes tipos de dados existentes, como por exemplo, os dados presentes numa base de dados ou até numa rede social. [26]

Um grafo é um conjunto de vértices, ou nós, ligados através de arestas. É possível representar um grafo através de $G = (V,E)$ onde E é um conjunto de arestas e V um conjunto de vértices. Cada aresta é representada por $e=(u,v)$, onde e pertence a E e (u,v) são pares de vértices, u e v, pertencentes a V. Dois vértices estão conectados se existir uma aresta de um vértice para o outro [32].

Uma árvore é um grafo em que qualquer par de vértices está ligado por um e apenas um caminho. Desta forma, existe uma hierarquia onde cada vértice (nó) tem apenas um pai, exceto a raiz, e este possui um ou mais filhos. Um simples exemplo de um grafo deste tipo é um sistema de ficheiros hierárquico [32].

Uma das técnicas mais utilizadas para a representação gráfica de grafos é designada de *force-directed technique* que consiste em utilizar conceitos da física e adaptá-los à visualização de grafos. Desta forma, os vértices são considerados como corpos que possuem uma massa, atraindo-se e repelindo-se uns aos outros. Deste modo, têm de existir forças que desempenhem o papel de repelir e atrair os diferentes vértices. Quando todas as forças envolvidas resultam num equilíbrio resulta, assim, a representação ótima do grafo [26].

Na figura 6 é possível visualizar um exemplo da representação em grafo relativo a um pequeno *web site* utilizando uma *force directed technique*. Desta forma, cada nó representa uma página e cada aresta representa o *link* entre as diferentes páginas (nós) [32].

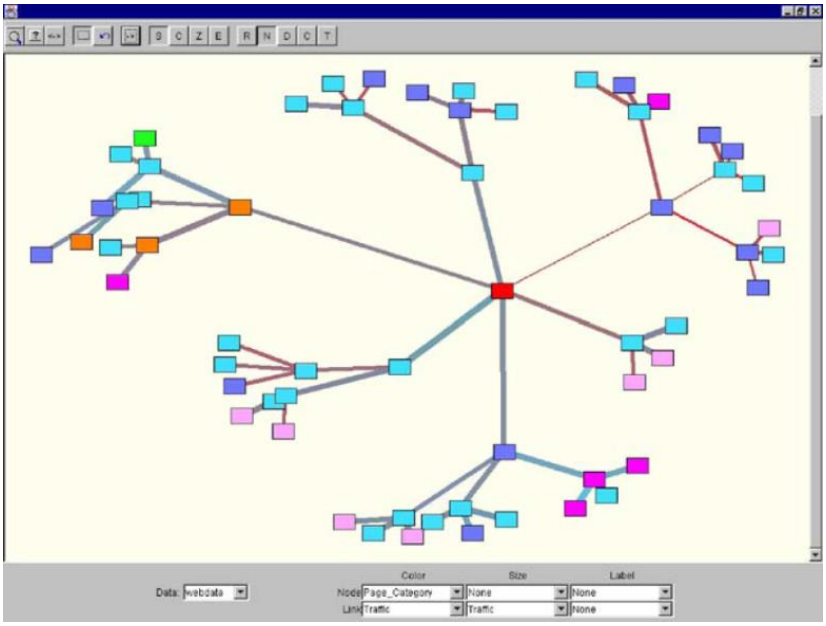


Figura 6 - Grafo [32].

Como já foi referido anteriormente é possível fazer o mapeamento de diferentes tipos de estruturas sob a forma de um grafo. Temos o exemplo de um sistema de ficheiros ou a forma de como está organizado um ficheiro com a extensão ".N1" (que resulta da medição dos instrumentos da missão *Envisat*). Este raciocínio pode ser empregue para a exploração visual e interativa dos dados existentes neste projeto. Assim sendo, este método resulta numa forma diferente de mostrar ao cliente a organização dos dados e informações disponíveis para consulta e posterior representação gráfica.

A pesquisa de dados é facilitada no sentido em que se torna possível ao utilizador navegar através das diferentes missões, instrumentos, domínios e parâmetros, resultando numa nova perceção de como é que a informação está organizada e, também, possibilitando a seleção dos parâmetros existentes em diferentes instrumentos ou até de diferentes missões. Também existe a possibilidade de organizar as informações representadas através da expansão ou retração dos diferentes componentes apresentados.

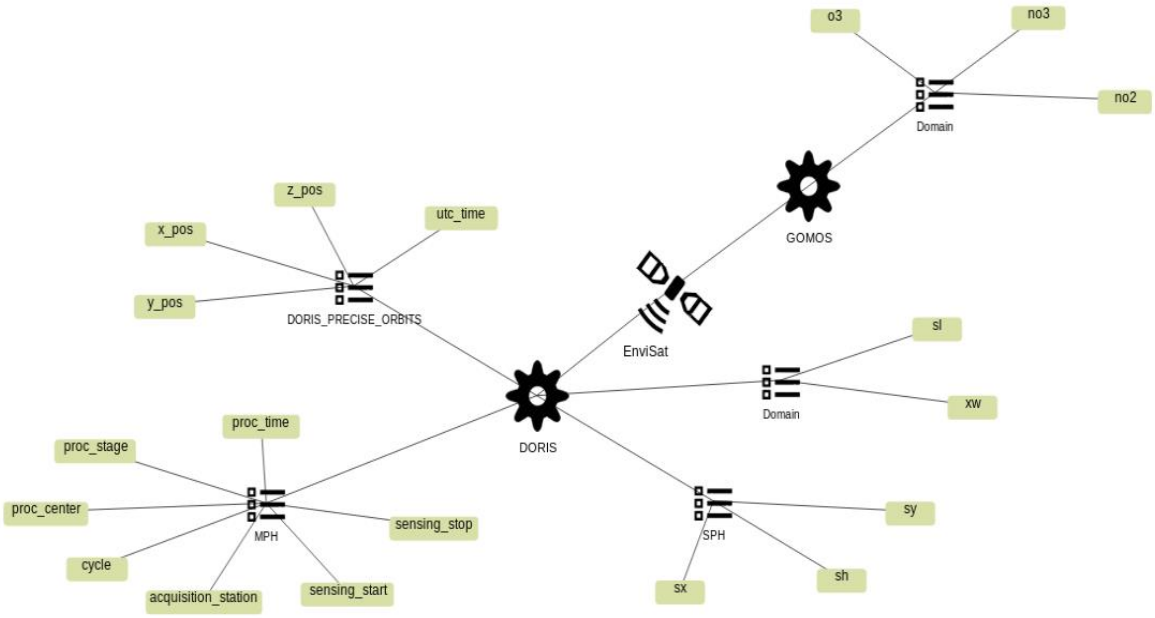


Figura 7 - Grafo adaptado à visualização de missões.

Embora a missão *EnviSat* possua dez instrumentos, na Figura 7, apenas estão representados dois deles, o GOMOS e o instrumento DORIS, uma vez que este exemplo é meramente figurativo. Deste modo, estando todos os nós expandidos, é possível visualizar toda a organização e estrutura da informação relativa à missão *EnviSat* (que se encontra no centro e possui um ícone de um satélite). Seguidamente verifica-se os dois instrumentos expandidos (DORIS e GOMOS), sendo também apresentados os seus domínios, onde cada um destes domínios contém diferentes parâmetros.

A forma de interação do utilizador com este tipo de grafo é baseada em cliques do rato, onde são expandidos ou fechados os diferentes nós que representam os diferentes elementos. Quando um nó é expandido são mostrados todos os seus filhos. Por outro lado, quando um nó é fechado, todos os filhos são fechados. O utilizador tem a possibilidade de reposicionar os nós da árvore, ação que pode resultar num desequilíbrio da árvore e, seguidamente, é possível verificar que a técnica de representação do grafo (*force directed technique)* volta a tentar equilibrar os nós existentes.

3.8 - Séries temporais

As séries temporais são um caso particular devido ao facto de existir uma ordem estrita dos dados. Estas são consideradas importantes na representação gráfica, uma vez que os bons gráficos respeitam o seguimento temporal dos dados. Assim, torna-se possível diferenciar os valores que ocorrem em diferentes tempos (como por exemplo o valor de uma empresa ao longo do tempo) e realizar médias que ocorrem durantes certos períodos (temos o exemplo da quantidade de ações vendidas no ultimo mês ou no último ano) [29].

A janela temporal tem de ser escolhida com atenção devido à possibilidade de ocorrer um problema relativo à inexistência de valores correspondentes ao tempo selecionado. Assim sendo, caso este problema aconteça, vão existir no gráfico secções onde se verifica a ocorrência de "buracos" por falta de dados no espaço temporal escolhido (como por

exemplo a inexistência de dados resultantes das medições do instrumento em algumas secções da janela temporal ou a inexistência de dados de ações nos dias em que os mercados estiveram fechados). Esta janela temporal também tem um forte impacto na interpretação dos dados apresentados graficamente [29].

Quando várias séries temporais são representadas no mesmo gráfico, é necessário assegurar que estas estão alinhadas temporalmente e também é necessário alinhar os valores do eixo vertical (como por exemplo a visualização de duas ou mais janelas temporais relativas à mesma variável resultante de várias medições do instrumento) [29].

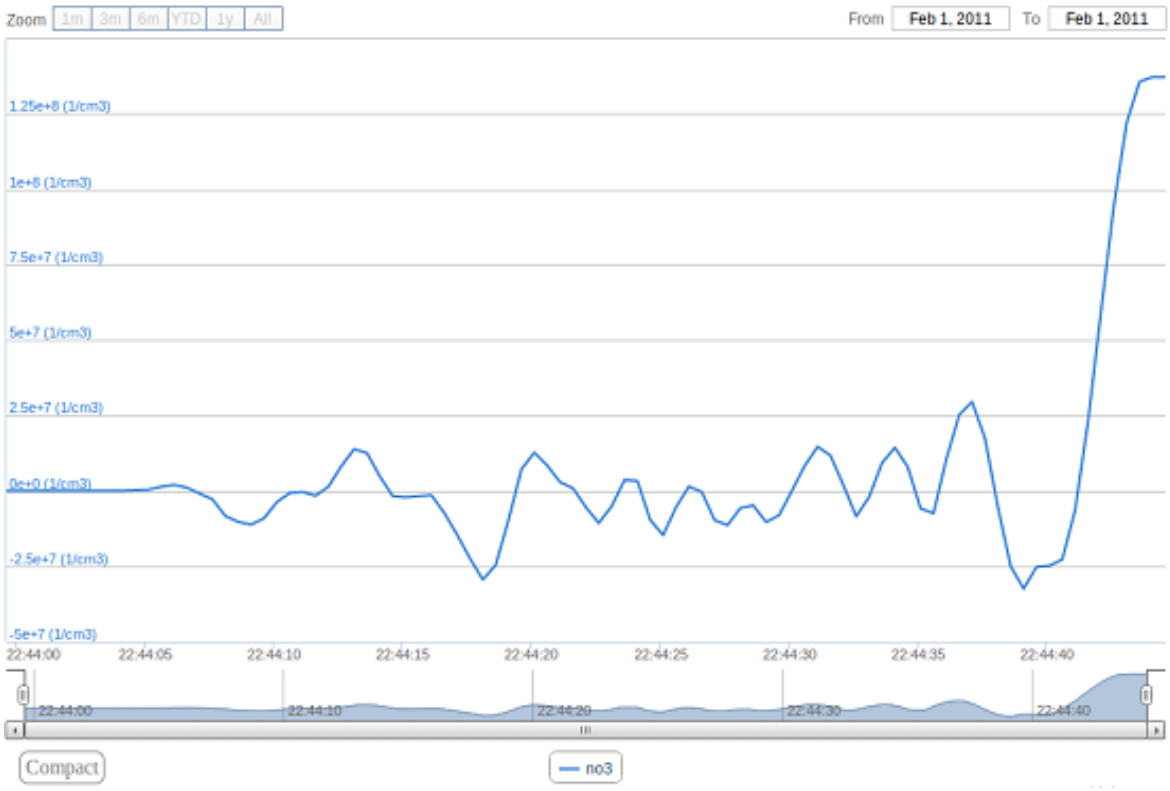


Figura 8 – Gráfico do tipo *time series*.

Uma característica importante a realçar consiste no facto de todos os parâmetros existentes relativos às medições dos instrumentos estarem relacionados com o tempo, ou seja, para cada leitura efetuada pelo satélite existe uma relação com o tempo. Assim sendo, os dados existentes relativos ao projeto respeitam o conceito de séries temporais e podem ser representados como estas.

Na Figura 8, apresenta-se a representação dos dados de uma variável que corresponde a uma série temporal. A linha está codificada com a cor azul e o eixo dos y possui a mesma cor para realçar a identificação dos valores relativos à série temporal. De notar que no eixo dos x é representada a variável do tempo e no eixo dos y estão representados os valores da variável NO3.

3.9 - Gráficos de Barras

Este tipo de gráficos é um dos mais utilizados e úteis, podendo as colunas estar orientadas na vertical ou na horizontal. Assim, é utilizado para comparar dados categóricos, onde as suas barras podem conter múltiplos valores associados à categoria e são bons para representar variáveis discretas. Na figura 9 podemos ver um exemplo deste tipo de gráfico [28].

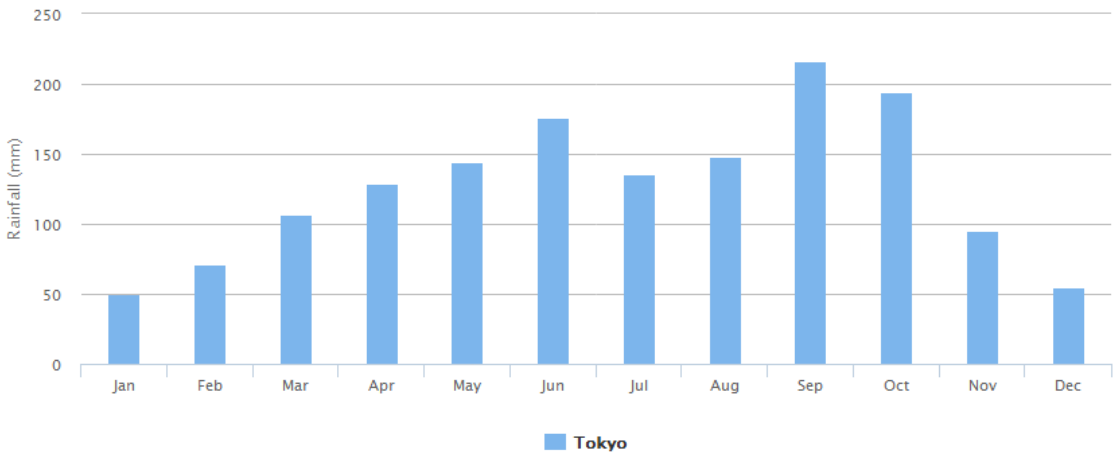


Figura 9 – Gráfico de barras [33].

3.10 - Gráfico de Linhas

Os gráficos de linhas são ideais para representar dados contínuos. São indicados para mostrar tendências onde a variável dependente (por exemplo a altura em metros de uma pessoa) é representada no eixo dos y e a variável independente (por exemplo a idade da pessoa ou o tempo) é colocada no eixo dos x. Podendo existir casos em que esta disposição não faça sentido devido à relação dos dados [28].

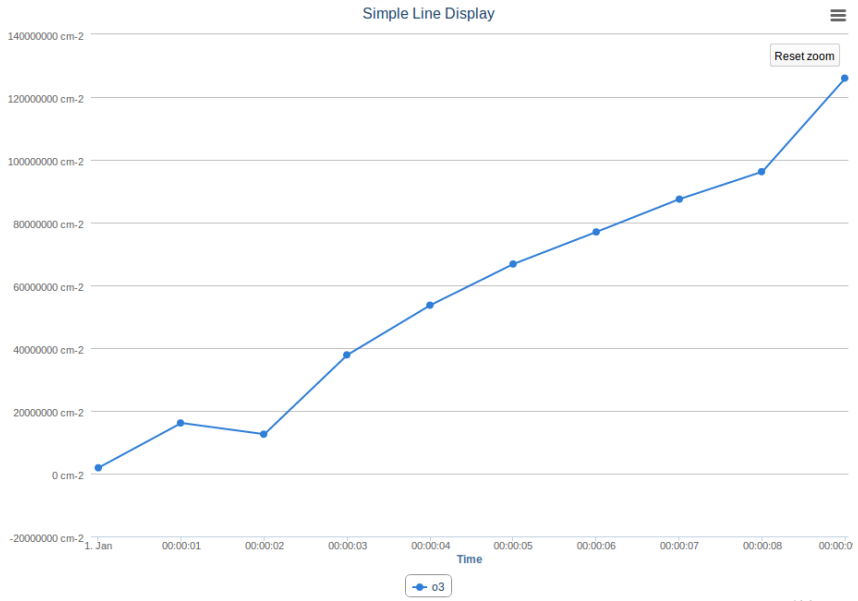


Figura 10 – Gráfico simples de linha com as medições O3.

A Figura 10 permite visualizar um gráfico de linha simples referente à medição de O3 do instrumento GOMOS que está inserido na missão *EnviSat*. É possível ainda referir que o tempo é a variável independente e está representada no eixo dos x. Por outro lado, a variável relativa aos valores de O3 é dependente e está representada no eixo dos y. Este exemplo é parecido ao gráfico da série temporal, mas podemos representar aqui outros tipos de variáveis contínuas ou a relação entre estas. Pode ainda ser identificado visualmente que a linha que representa a variável o3 está codificada com a cor azul e possui um glifo redondo para identificar os pontos (x, y) das duas variáveis.

3.11 - Gráficos de dispersão

Os gráficos de dispersão representam a relação entre duas variáveis utilizando o sistema de coordenadas no plano cartesiano e são úteis para procurar correlações entre essas duas variáveis. Também são utilizados para visualizar informações de três ou quadro dimensões. Nesses casos, cada fonte de informação é codificada para a visualização de uma forma individual, isto é, cada variável possui um único tipo de codificação [28].

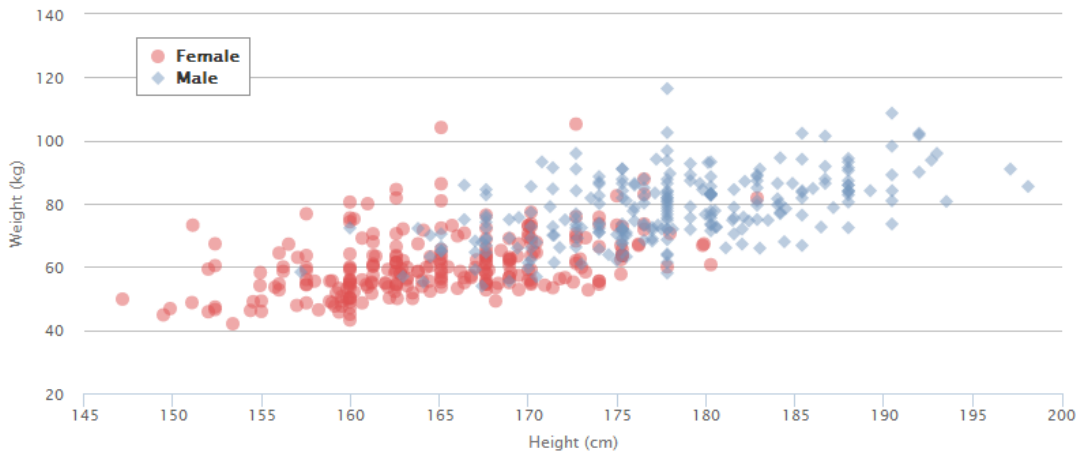


Figura 11 – Gráfico de dispersão relativo à altura e ao peso de elementos do sexo feminino e masculino [33].

Na Figura 11 é possível visualizar um gráfico de dispersão que apresenta os valores relativos ao peso e a altura de elementos do sexo feminino e masculino. É de salientar que conseguimos diferenciar estes elementos pela sua codificação, ou seja, os elementos do sexo masculino estão representados a azul e possuem o símbolo de diamante, por outro lado, os elementos do sexo feminino estão representados a cor-de-rosa e com um símbolo esférico. Neste gráfico é possível verificar a correlação entre os dados e detetar casos isolados.

3.12 - Tabelas

Tabelas com os diferentes dados inseridos nas células são um tipo de visualização que é utilizado quando a apresentação da precisão dos dados é necessária. Este tipo de representação pode ser eficaz em dados onde seja possível observar tendências. De notar que as células não precisam de conter apenas os dados em formato de texto, estas podem ser formatadas para ter cores ou outro tipo de codificação (como por exemplo uma tabela periódica) [28].

Parameter	Domain	Instrument	Mission
z_pos	DORIS_PRECISE_ORBITS	DORIS	EnviSat

Value	Unit	Timestamp
-16892085.607	m	1167613234
-26892085.607	m	1167613235
-36892085.607	m	1167613236
-46892085.607	m	1167613237
-56892085.607	m	1167613238
-66892085.607	m	1167613239
-76892085.607	m	1167613240

Figura 12 – Representação do parâmetro z_pos em tabela.

A representação de dados em tabelas, como por exemplo as tabelas da Figura 12, é útil para o utilizador no sentido em que permite visualizar todos dados sob a sua forma bruta. Também é útil quando é necessário apresentar campos isolados ou *flags*. Por exemplo, existem informações no cabeçalho dos ficheiros "*.N1*" da missão *EnviSat* que são *flags* e esta torna-se uma maneira simples de as representar visualmente. As tabelas da imagem 12 são a apresentação dos dados das leituras relativas ao parâmetro "*z_pos*" medido pelo instrumento DORIS. Verifica-se que as unidades são o metro (m) e que existe a informação do tempo em que a medição foi efetuada (*Timestamp* em *posix time*).

3.13 – Mapas

Algumas práticas relativas à representação de dados em mapas são baseadas na coloração de regiões e na representação de quantidades através da utilização de bolhas na região geográfica pretendida. No entanto, existem práticas que distorcem a representação física do espaço para melhor representar o significado dos dados. Por exemplo, quando ocorrem eleições, se existe uma grande região geográfica com pouca demografia, esta deve ser transformada e reduzida quando for comparada com uma região geográfica menor mas com muita população. Isto acontece porque, se uma grande região for colorida com as cores de um partido e uma pequena região for colorida com as cores de outro partido, visualmente o partido que perdeu pode possuir a maior parte do gráfico colorido com a sua cor, mas, não significa que tenha ganho as eleições, pois as zonas com mais habitantes e mais pequenas representam a maioria dos votos, resultando assim, num gráfico que transmite de forma errada os dados [28].

Contudo, neste projeto estas questões são postas de parte e apenas é necessário reproduzir num mapa terrestre as diferentes posições que correspondem à órbita dos diferentes satélites que estão a realizar missões. Desta forma, a posição do satélite é representado com um ícone no mapa.

3.14 - *Linked Views*

O paradigma das *Linked Views* leva a que dois ou mais gráficos partilhem e troquem informações entre si. Para que isto seja possível é necessário fazer uma ligação entre todos os tipos de visualização pretendidos. Assim, quando o utilizador está a explorar os dados pode destacar alguns dados num tipo de gráfico que estas ações são traduzidas nos outros gráficos. Esta técnica resulta numa maior capacidade de interação entre o utilizador e os tipos de gráficos que estão a ser apresentados [34].

3.15 - Representação multidimensional

Os conjuntos de dados multivariável, também chamados de multidimensional ou *n*-dimensional, são grupos de *n* canais de dados [28] [31]. Por exemplo, um gráfico financeiro pode ter três dimensões, sendo estas a data, o preço e a respetiva empresa. Caso seja adicionado o volume de negócios o gráfico passa a ter quatro dimensões. Já se forem adicionadas mais empresas, isto pode levar a um estado de sobreposição no gráfico que o torna mais difícil de perceber, mas isto não aumenta a sua dimensão. Aumentando o volume de dados na representação gráfica também não altera a sua dimensão, pois o facto de mostrar mais anos de dados não é mais complexo do que mostrar uma semana, é apenas mais volumoso [28].

O total de dimensões existentes num gráfico pode ser descrito como a complexidade deste gráfico. Quando as visualizações tornam-se mais complexas, isto é, quando a dimensão se torna elevada, estas tornam-se mais difíceis de entender e até de desenhar. Isto verifica-se devido ao facto de que quando a complexidade é aumentada existe a necessidade de codificar no gráfico mais propriedades únicas a nível individual (características diferenciadoras). Por este motivo, as representações gráficas de três ou quatro dimensões são as mais comuns [28].

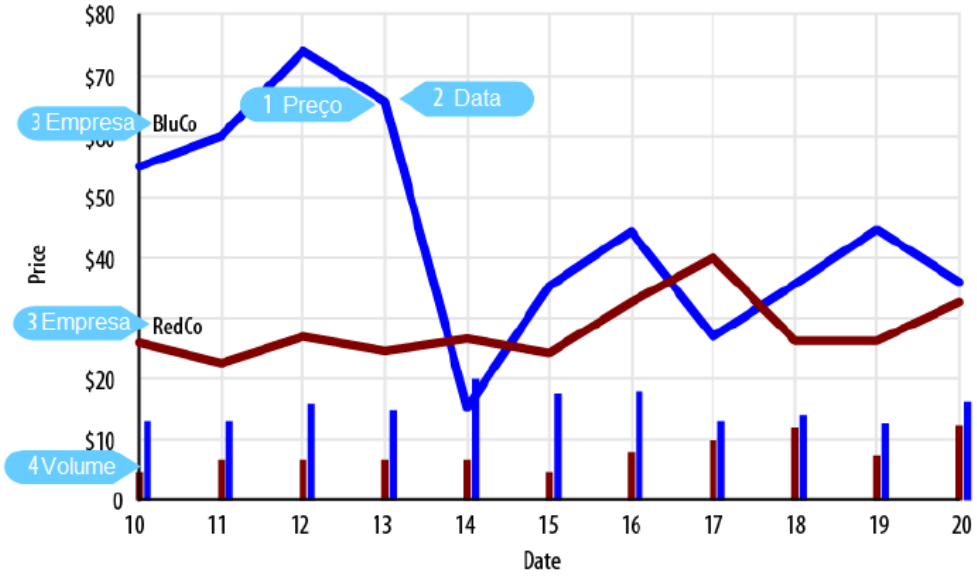


Figura 13 – Gráfico que representa quatro dimensões [28].

3.15.1 - Matrizes de Dispersão (SPLOMS)

As matrizes de dispersão ou SPLOMS são compostas por vários gráficos de dispersão. Podemos definir este tipo de visualização como a representação de *p* variáveis, mostradas na diagonal e todos os gráficos de dispersão que não estão na diagonal são a relação entre duas dessas *p* varáveis [32]. Ou seja, as matrizes de dispersão são utilizadas para mapear uma variável em relação a todas as outras variáveis que o utilizador selecionou para a visualização. Este tipo de representação é útil para encontrar correlações nos dados das várias dimensões apresentadas [30].

Porém, pode ocorrer um problema com este tipo de representação que se baseia no facto deste tornar-se ilegível quando existe um número elevado de variáveis, pois vão estar dispostas no gráfico um número elevado de gráficos de dispersão. Desta forma, existe uma limitação visual visto que é necessário mais espaço para representar muitas variáveis. Para mais de 25 variáveis este tipo de visualização torna-se impraticável [32].

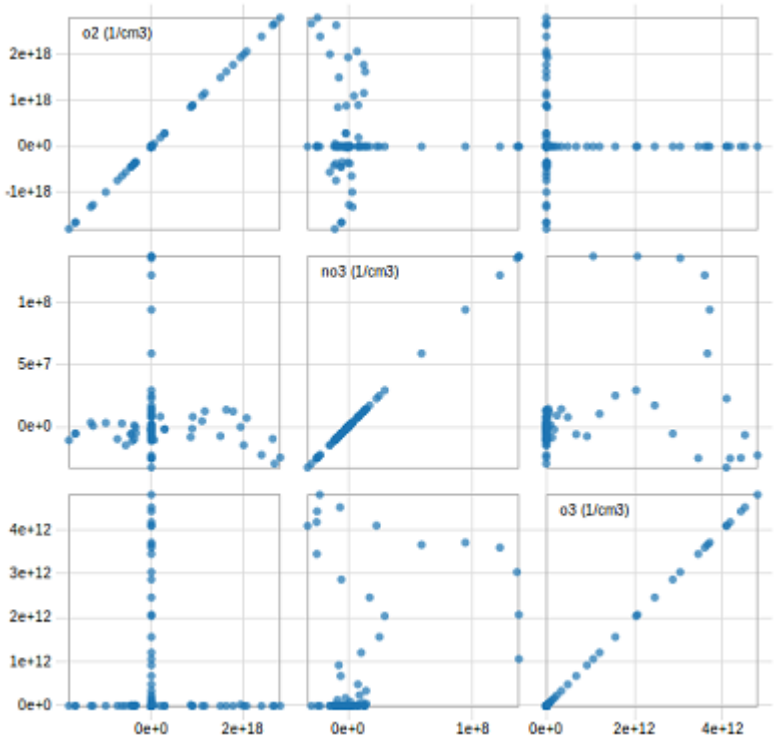


Figura 14 – Gráfico do tipo sploms.

Na Figura 14 conseguimos visualizar a técnica de representação gráfica correspondente às matrizes de dispersão (SPLOMS). Verifica-se que neste exemplo existem três variáveis que estão representadas na diagonal do gráfico, os valores desta diagonal são os valores da própria variável (valores de x e y iguais). Os outros gráficos de dispersão são o resultado da relação de uma variável em relação à outra, ou seja, na primeira linha e segunda coluna podemos ver a relação, num gráfico de dispersão, da variável NO3 (eixo dos x) em relação à variável O2 (eixo dos y) e que a relação entre a variável O2 (eixo dos x) e NO3 (eixo dos y) está representada no gráfico presente na primeira coluna e segunda linha.

Neste exemplo da Figura 14 verifica-se que as variáveis estão todas representas utilizando a cor azul e que existe um glifo redondo para identificar o ponto (x, y) da relação entre as variáveis.

3.15.2 – Gráficos de Linha - Múltiplas Linhas

Os gráficos de linha são normalmente utilizados para mostrar os dados provenientes de apenas uma fonte de informação e de um modo geral, criam uma visualização em duas dimensões, onde os dados são mapeados para um gráfico de coordenadas cartesianas (x, y). Ao adicionar múltiplas linhas ao gráfico, este pode mostrar mais de duas dimensões (x, y1, y2, y3…), onde a variável independente (x) é única. Este tipo de gráfico é apropriado para analisar funções contínuas em detalhe [30].

Como forma de distinguir cada dimensão adicionada têm de ser utilizadas técnicas de coloração da linha ou a utilização de vários glifos. Cada dimensão também pode ter a sua própria escala, que pode ser diferente das outras dimensões já existentes no gráfico [30].

Contudo, para um número de dimensões superior a três este tipo de visualização pode tornar-se confuso, mas isto vai depender da escala e da forma utilizada para mapear as várias dimensões [30].

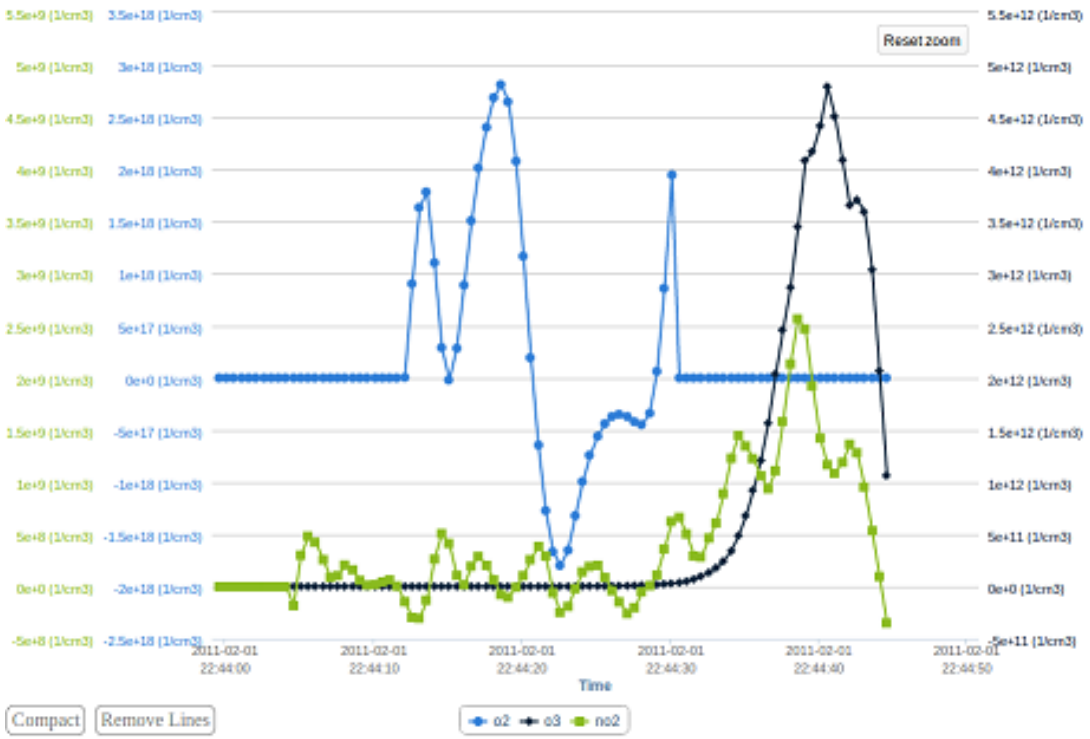


Figura 15 – Gráfico múltiplas linhas e múltiplos eixos.

Na Figura 15 podemos verificar que existem três dimensões representadas no gráfico de múltiplas linhas. Cada variável é codificada no gráfico utilizando uma cor única, sendo utilizadas as cores azul, verde e preto para representar as variáveis O2, NO2 e O3. Também são utilizados diferentes tipos de glifos para cada uma, sendo estes quadrados, círculos e diamantes. Verifica-se também que cada eixo (dos y) possui a cor da variável que representa. Assim, torna-se possível identificar os valores correspondentes à variável no eixo dos y. O eixo do x representa o tempo e os eixos dos y representam o valor de cada variável e possuem as mesmas unidades (1/cm^3).

3.15.3 - Coordenadas Paralelas

O tipo de gráfico designado por "coordenadas paralelas" é apropriado para representar, em simultâneo, um grande número de variáveis contínuas. Assim sendo, esta forma de representação torna-se muito atrativa para a investigação de grupos de inúmeras variáveis num único gráfico. Não existe outro tipo de gráfico onde seja possível a representação da quantidade de informação que é possível neste [29].

A visualização deste tipo de gráfico consiste em representar dados multidimensionais utilizando linhas que cruzam os eixos que estão dispostos na vertical e que representam cada dimensão [30]. Desta forma, torna-se na ferramenta ideal para obter uma primeira análise sobre todos os dados que estão a ser estudados. Na Figura 16 é possível visualizar um gráfico de coordenadas paralelas onde se encontra a representação de 10 variáveis relativas a mais de 400 veículos automóveis [35].

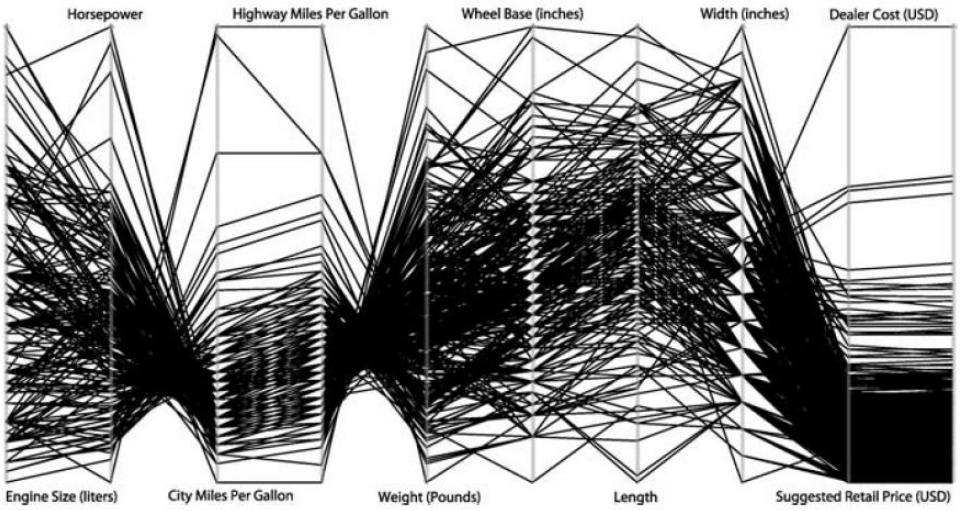


Figura 16 – Coordenadas paralelas [35].

Apesar da sobreposição das linhas que representam cada automóvel, o sistema de visualização baseado em coordenadas paralelas pode possuir uma funcionalidade de seleção e, assim sendo, permite a visualização de um único item (i.e, neste caso, cada carro) o que possibilita a comparação do item selecionado com os restantes presentes na janela de visualização. Este tipo de gráfico torna-se eficiente quando os eixos das coordenadas apresentam uma ordenação, como por exemplo, ordenação temporal [35].

Porém, mostrar muita informação em simultâneo e num único gráfico acarreta fatores negativos, como por exemplo, a sobreposição das linhas torna o gráfico praticamente ilegível. Com esta limitação, o diagrama apenas consegue representar poucas centenas de variáveis. Outra limitação corresponde à fraca possibilidade de detetar caraterísticas que não são visíveis nos gráficos de uma ou duas dimensões [35].

De forma a solucionar as falhas apresentadas é possível aplicar várias técnicas para reduzir a sobreposição das linhas e tornar a representação mais explícita. É de salientar a técnica de *alpha-blending* ("α-*Blending*") e uma técnica baseada na alteração da escala do gráfico. A primeira consiste em aplicar a cada linha uma transparência de (1-α)%. Com valores de α baixos, as áreas do gráfico com maior densidade são mais visíveis do que áreas com menor densidade. A Figura 17 apresenta os resultados da aplicação desta técnica [35].

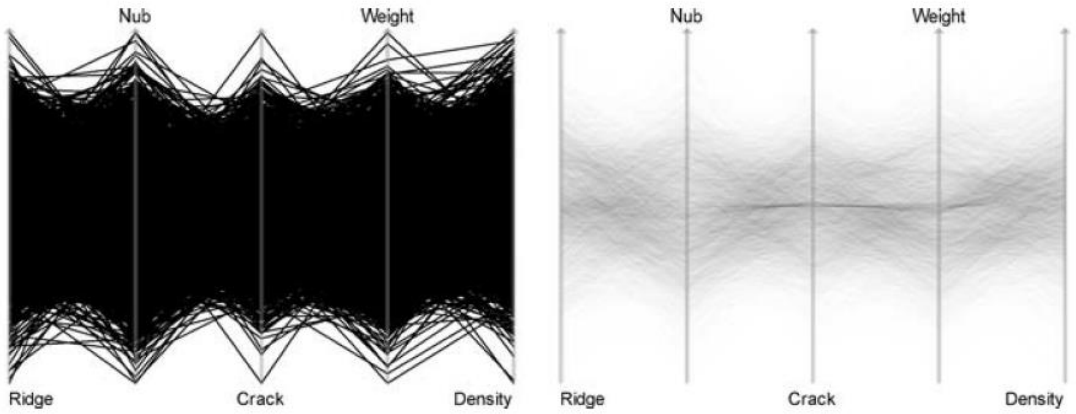


Figura 17 – Gráfico com alpha=1 na direita e alpha=0.005 na esquerda [35].

No exemplo, meramente figurativo, da Figura 17, são apresentados quase 4000 casos caracterizados por 5 variáveis. À direita está representado um gráfico sem a utilização da técnica "α-*Blending*" e à esquerda é representado um gráfico com esta técnica aplicada. À esquerda o gráfico é praticamente ilegível, mas à direita é possível verificar uma linha sólida no centro do gráfico que significa que há uma maior densidade de linhas a passar no local e, assim sendo, uma maior incidência de casos nesta zona [35].

A escala neste tipo de diagrama é importante para a visualização. Por defeito a escala corresponde à representação de todas as variáveis desde o seu mínimo até ao seu máximo em cada eixo. As opções de escala mais importantes estão relacionadas com a escala de cada eixo individualmente ou então usar uma única escala para todos os eixos do gráfico [35].

A ordem neste sistema gráfico é importante para a sua interpretação, pois a ocorrência de padrões, importantes para os pesquisadores, acontecem normalmente entre pares de variáveis. Assim sendo, a ordem dos eixos neste tipo de gráficos pode aumentar significativamente a capacidade de visualização e interpretação de resultados por parte dos investigadores [35].

As representações abaixo, meramente figurativas, mostram três diagramas de "coordenadas paralelas" com os dados do tempo de 155 ciclistas que acabaram o "Tour de France" no ano de 2005. Aqui são mostrados diferentes tipos de escalas e alinhamento dos dados. [35]

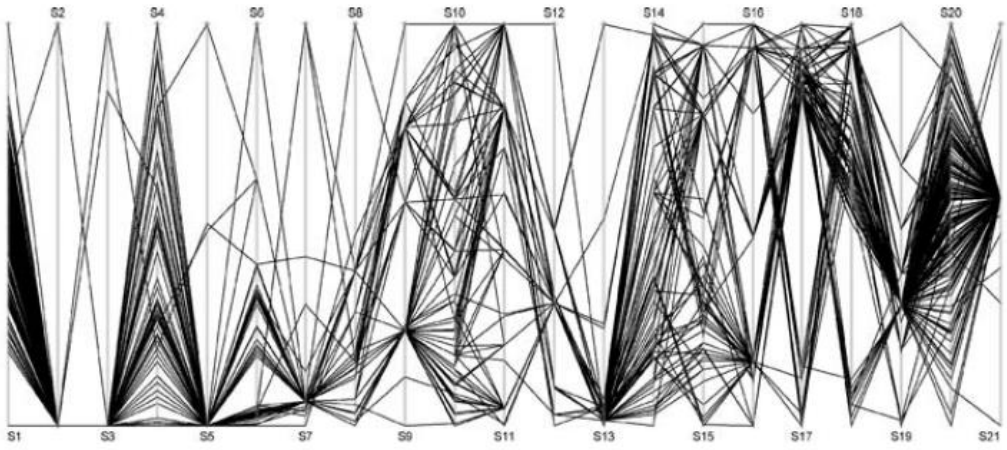


Figura 18 – Coordenadas paralelas 1. [35]

Na Figura 18, cada eixo possui a sua própria escala, que se inicia no valor mínimo dos dados e termina no valor máximo. Desta forma, o mínimo da escala representa o tempo mínimo que um ciclista demorou a terminar a etapa e o máximo corresponde ao maior tempo que um ciclista demorou a terminar a prova. É possível verificar onde existe maior concentração de grupos de ciclistas que corresponde às zonas com mais concentração de traços negros [35].

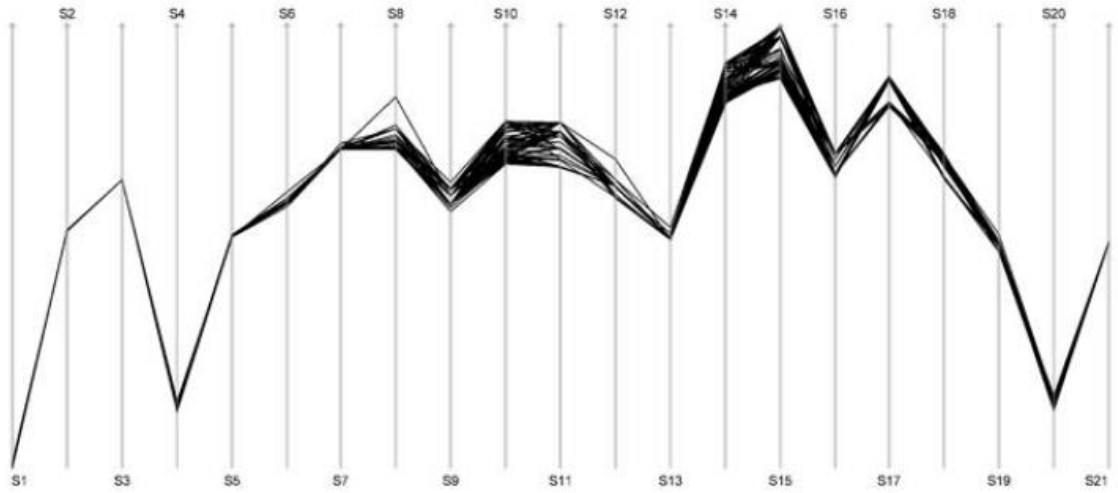


Figura 19 – Coordenadas paralelas 2 [35].

No diagrama da Figura 19 todos os eixos possuem a mesma escala, no entanto, não é fácil, para a maioria das etapas, observar a diferença entre o último ciclista a terminar a etapa e o primeiro classificado. Com a exceção de algumas etapas, onde é possível verificar este facto [35].

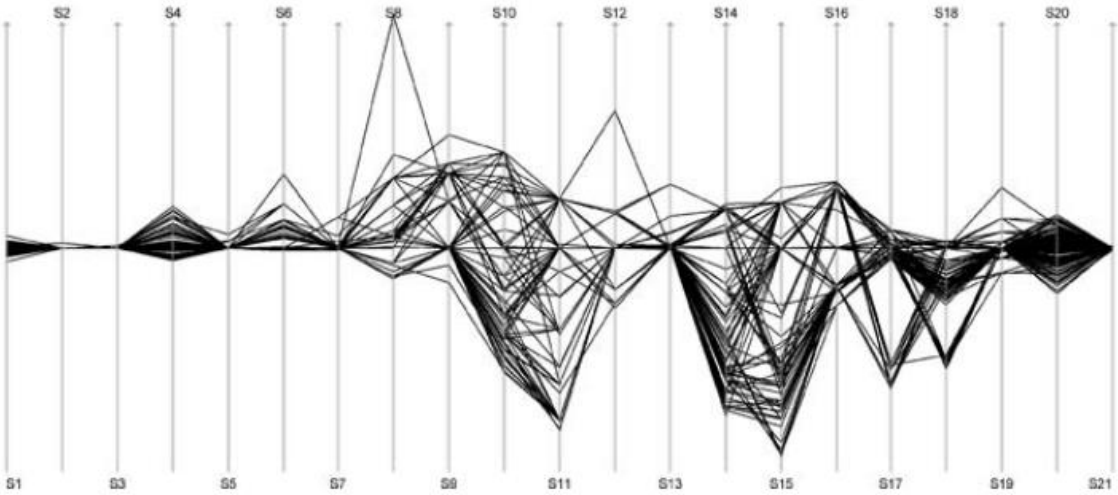


Figura 20 – Coordenadas paralelas 3 [35].

Na Figura 20 os eixos possuem a mesma escala mas todos os dados estão representados através da sua média, isto é, o tempo médio de cada atleta até ao fim de cada etapa. É possível comparar as diferenças de tempo entre os vários ciclistas. Este diagrama, em comparação com os anteriores, é o que revela mais informação [35].

O diagrama do tipo "coordenadas paralelas" não é muito útil sem utilizar as técnicas de otimização de visualização descritas. Porém, cada utilizador através da sua experiência de utilização deve identificar as técnicas que mais se adaptam ao seu problema e que melhor conseguem solucionar a sua questão [35].

Capítulo 4 - Requisitos

Neste capítulo vão ser apresentados todos os requisitos das plataformas desenvolvidas no âmbito do estágio. Os requisitos foram identificados de acordo com a necessidade de implementar uma plataforma de visualização de dados *online*, com o objetivo de preencher uma lacuna existente neste campo.

Para que seja possível descrever os vários requisitos de uma forma específica, o projeto vai ser dividido em três partes. A primeira parte é constituída por um "*Database Updater*", a segunda parte diz respeito ao "*Web-Service*" e, por fim, a terceira corresponderá à aplicação *web* apelidada de MDV (*Mission Data Visualization*). Para obter informações mais pormenorizadas referentes aos requisitos é necessário consultar o Apêndice A.

4.1 – Casos de Uso

Os casos de uso são utilizados para ajudar na análise de requisitos através da forma como o utilizador interage com a aplicação. Deste modo, foram realizadas várias reuniões com o orientador da empresa *VisionSpace Technologies* no sentido de procurar identificar as interações que os utilizadores teriam com os sistemas a serem desenvolvidos. Nesta fase, não é necessário capturar todos os pormenores mas apenas aqueles que se revelam essenciais para compreender os requisitos dos sistemas no seu todo. Os casos de uso podem ser consultados no Apêndice D.

4.2 - Levantamento de Requisitos

O levantamento de requisitos foi realizado durante várias reuniões com o orientador de estágio da empresa *VisionSpace Technologies*. Desta forma, foi possível elaborar um documento, que se encontra em anexo, com todos os pormenores dos requisitos discutidos.

4.3 - Requisitos Funcionais

Nesta secção vão ser apresentados os requisitos funcionais do sistema que foram implementados. São disponibilizadas tabelas com o código de cada requisito e a respetiva descrição.

4.3.1 - Requisitos Funcionais do "*Web-Service*"

Neste ponto são mostrados todos os requisitos funcionais do "*Web-Service*", que resultaram de várias reuniões. Alguns destes requisitos foram alterados, eliminados e acrescentados, de acordo com a evolução dos trabalhos. Este "*Web-Service*" deverá possibilitar a interação com outros sistemas informáticos que necessitem de dados, assim como de utilizadores, como por exemplo, cientistas que estejam à procura de dados de missões. A tabela abaixo representa todos os requisitos funcionais deste sistema:

Código	Descrição
RF-WS-01	O "*Web-Service*" tem de possuir um comando de ajuda. Desta forma, torna-se possível ao utilizador saber como é que este sistema funciona.
RF-WS-02	O "*Web-Service*" tem de redirecionar o pedido do cliente para a página de ajuda caso o método que é chamado não exista.
RF-WS-03	Este sistema deve possuir uma função para enviar todos os métodos suportados, ou seja, todos os métodos que o cliente pode utilizar para interagir com o "*Web-Service*".
RF-WS-04	O sistema tem de enviar para o utilizador toda a informação sobre as missões existentes no "*Web-Service*".
RF-WS-05	O "*Web-Service*" tem de possuir a funcionalidade de enviar uma lista com todas as missões existentes no sistema.
RF-WS-06	É necessário existir uma funcionalidade que envia para o utilizador os nomes de todos os instrumentos de uma dada missão.
RF-WS-07	O sistema tem enviar informações mais específicas sobre cada instrumento, isto é, as características de cada instrumento referente a uma missão.
RF-WS-08	O sistema tem suportar a funcionalidade de enviar ao utilizador todos os parâmetros das leituras de um instrumento existentes no "*Web-Service*".
RF-WS-09	O sistema tem de enviar para o utilizador informação sobre o intervalo de tempo das medições de um instrumento.
RF-WS-10	Tem de ser possível enviar para o utilizador o valor de um parâmetro.
RF-WS-11	Tem de existir uma validação dos parâmetros relativos ao método que é chamado pelo utilizador.
RF-WS-12	Tem de haver uma validação para o caso da informação requerida não existir no sistema.
RF-WS-13	Todas as datas, i.e tempos requisitados, têm de estar representadas como o número de milissegundos a partir do dia 1 de Janeiro de 1970.
RF-WS-14	Tem de existir uma funcionalidade que dê informação sobre o tipo de erro que é enviado ao cliente.
RF-WS-15	Informação geral dos tipos de erro existentes.

Tabela 1 – Requisitos funcionais do "*Web-Service*".

4.3.2 - Requisitos Funcionais da Aplicação *web* - MDV

Nesta secção são mostrados todos os requisitos funcionais da aplicação *web*. Esta aplicação vai interagir com a API pública do "*Web-Service*" como forma de receber todos os dados e informações necessárias para responder às necessidades dos utilizadores. Deste modo, torna-se possível aos clientes visualizarem de forma interativa os dados existentes no sistema. A tabela abaixo representa todos os requisitos da MDV:

Código	Descrição
RF-MDV-01	O utilizador tem de poder explorar os dados das várias missões, instrumentos, domínios e parâmetros de forma visual e interativa, através da visualização de um grafo.
RF-MDV-02	Tem de ser disponibilizada ao utilizador a funcionalidade de criar e gerir listas de parâmetros.
RF-MDV-03	Tem de ser mostrada ao utilizador a janela temporal disponível para os dados que estão inseridos na lista selecionada.
RF-MDV-04	Possibilidade de ver os dados dos parâmetros em formato de texto.
RF-MDV-05	O utilizador pode aceder a informações relativas às missões e aos instrumentos.
RF-MDV-06	Tem de ser possível ao utilizador criar gráficos de linha simples, gráficos de linha com múltiplos eixos, gráficos de dispersão, gráficos de coordenadas paralelas, matrizes de dispersão e gráficos especializados para a representação de *time series*. Também deve possuir a funcionalidade de ver num mapa a órbita do satélite.
RF-MDV-07	Tem de existir a funcionalidade de autenticação e registo na plataforma *online*.
RF-MDV-08	O ambiente de trabalho do utilizador, isto é, as suas listas e configurações devem ser guardadas na base dados. Quando este utilizador voltar a fazer *login*, o seu ambiente de trabalho deve ser descarregado pelo *site* de forma automática.

Tabela 2 – Requisitos funcionais da MDV.

4.3.3 - Requisitos Funcionais do "*Database Updater*"

São apresentados de seguida os requisitos funcionais do "*Database Updater*". Estes são o resultado de algumas reuniões que aconteceram na empresa. A tabela abaixo representa todos os requisitos do "*Database Updater*":

Código	Descrição
RF-DU-01	Este sistema tem de atualizar a base de dados com os dados dos ficheiros das missões.
RF-DU-02	Tem de haver a possibilidade de filtrar os dados.
RF-DU-03	Este sistema tem de abrir os ficheiros, com diferentes extensões, que contêm os dados.

Tabela 3 – Requisitos funcionais "*Database Updater*".

4.4 - Requisitos Não Funcionais

Os requisitos não funcionais estão relacionados com a usabilidade, desempenho, segurança, disponibilidade, tecnologias envolvidas, entre outros. Todas estas características dependem do *software* desenvolvido, assim como do programador envolvido no projeto. Para o cliente, estas características têm de ficar invisíveis no normal uso do sistema. Por exemplo, se ocorrer um erro na comunicação entre os servidores, o *software* tem de resolver a questão sem que o utilizador intervenha nem tenha conhecimento do facto. Estas características têm grande influência na qualidade do *software* a ser desenvolvido, na medida em que estão relacionadas com o tempo de resposta por parte do sistema e se o sistema é amigável para o utilizador, entre outros.

4.4.1 - Requisitos Não Funcionais do "*Web-Service*"

Nesta secção são descritos todos os requisitos não funcionais do "*Web-Service*", sendo estes importantes no desenvolvimento da aplicação. Assim sendo, abaixo está presente uma tabela com os requisitos não funcionais:

Código	Descrição
RNF-WS-01	O "*Web-Service*" tem de ser desenvolvido na linguagem *Java*.
RNF-WS-02	O "*Web-Service*" tem de possuir uma base de dados, utilizando o modelo não relacional, isto é, base de dados de documentos.
RNF-WS-03	O "*Web-Service*" tem de comunicar com a base de dados.
RNF-WS-04	O "*Web-Service*" tem de possuir uma API pública capaz de receber pedidos através do protocolo *http*.
RNF-WS-05	O "*Web-Service*" tem de possuir a funcionalidade de validação de pedidos.

Tabela 4 – Requisitos não funcionais "*Web-Service*".

4.4.2 - Requisitos Não Funcionais da Aplicação *web* - MDV

A descrição dos requisitos não funcionais da plataforma *web* MDV é apresentada na tabela seguinte.

Código	Descrição
RNF-MDV-01	Utilização da API pública do "*Web-Service*".
RNF-MDV-02	A aplicação *web* MDV tem de ser desenvolvida utilizando a *framework Ruby on Rails*.
RNF-MDV-03	O *site* deve ser desenvolvido utilizando *html*, *css* e *Javascript*.

Tabela 5 – Requisitos não funcionais da MDV.

4.4.3 - Requisitos Não Funcionais do "*Database Updater*"

Na tabela abaixo são apresentados os requisitos não funcionais para o "*Database Updater*".

Código	Descrição
RNF-DU-01	O sistema tem de ser desenvolvido na linguagem de programação *Java*.
RNF-DU-02	O sistema tem de comunicar com a base de dados.

Tabela 6 – Requisitos não funcionais do "*Database Updater*".

4.4.4 - Segurança

Como não vão existir informações críticas ou privadas na comunicação entre os servidores e o utilizador final, não vai existir segurança entre os dois intervenientes da comunicação.

4.4.5 - Testes

Como alguns requisitos, em ambos os sistemas (aplicação *web* e serviço *web*), podem ser testados torna-se possível criar os respetivos casos de teste e a sua validação. Para efetuar os testes dos requisitos foram realizados testes de aceitação não automatizados que se baseiam em *Black-Box Testing*. Esta técnica consiste em examinar as funcionalidades do sistema sem "olhar" para as estruturas e os processos internos do sistema (este caso corresponde a *White-Box Testing*). Desta forma, torna-se possível realizar a maior partes dos testes de alto nível às aplicações que vão ser desenvolvidas.

Assim sendo, para testar o sistema não é necessário saber como é que este funciona internamente, como por exemplo, os tipos de dados que são processados ou como são processados. A pessoa responsável pelos testes, nesta abstração, apenas necessita de saber o que é que o sistema faz, ou seja, quando introduz um *input*, sabe que tem de existir um determinado *output* por parte do sistema. Os casos de testes são gerados a partir das

funcionalidades e dos requisitos do sistema. Esta técnica de testes de código está voltada para as partes funcionais do sistema.

4.5 - Requisitos Externos

Nesta secção vão ser apresentados os requisitos externos das aplicações desenvolvidas. Os requisitos externos são fatores extrínsecos ao sistema em si, como por exemplo, a integração ou a dependência de outros sistemas.

4.5.1 - Requisitos Externos do "*Database Updater*"

Existem dois requisitos externos para o "*Database Updater*", são estes:

Código	Descrição
RE-DU-01	Necessidade da existência dos ficheiros que contêm os dados e as informações dos instrumentos.
RE-DU-02	Comunicação com a base de dados.

Tabela 7 – Requisitos externos do "*Database Updater*".

4.5.2 - Requisitos Externos do "*Web-Service*"

O "*Web-Service*" é um servidor responsável pelo envio de informação e dados relativos às várias missões. Deste modo, é uma peça fundamental do projeto desenvolvido. Assim sendo, possui os seguintes requisitos externos para o seu correto funcionamento:

Código	Descrição
RE-SW-01	Comunicação com a base de dados.
RE-SW-02	Necessidade de acesso à Internet.

Tabela 8 – Requisitos externos do "*Web-Service*".

4.5.3 - Requisitos Externos da Aplicação *web* - MDV

Os requisitos externos relativos a este sistema são os seguintes:

Código	Descrição
RE-MDV-01	Comunicação com o "*Web-Service*".
RE-MDV-02	Comunicação com a Internet.
RE-MDV-03	Utilização de um *browser* para aceder aos conteúdos.

Tabela 9 – Requisitos externos da MDV.

4.6 – Comentários Finais

A documentação dos conjuntos de requisitos é essencial na fase inicial do projeto, uma vez que são especificadas as propriedades e as principais funcionalidades necessárias do projeto que se pretende desenvolver.

Nesta secção foram identificados todos os requisitos dos sistemas desenvolvidos, sendo que alguns foram alterados, eliminados e acrescentados de acordo com a sua recolha, análise e compreensão das necessidades da empresa. Para auxiliar o levantamento dos requisitos foram utilizados, como ferramenta, os casos de uso dos sistemas. Estes permitem identificar as principais interações entre o utilizador e as aplicações projetadas.

A importância dos requisitos consiste em tornar exequível o desenvolvimento de uma arquitetura e a sua implementação. Como cada requisito deve poder ser testado, estes foram utilizados para realização de casos de testes funcionais com o propósito de validar as aplicações de *software* desenvolvidas durante o estágio, conforme se descreve no capítulo 8.

Capítulo 5 – Arquitetura

Neste capítulo são apresentadas as arquiteturas relativas aos sistemas desenvolvidos no âmbito do projeto, assim como as principais tecnologias utilizadas em cada componente.

5.1 - Arquitetura Geral

Este projeto é composto por vários sistemas independentes mas que trabalham em conjunto para o correto funcionamento do projeto como um todo. Desta forma, o trabalho desenvolvido é constituído por três sistemas diferentes, denominados "*Web-Server*", "*Database Updater*" e MDV (aplicação *web*). Na Figura 21 está representada a arquitetura global dos sistemas implementados e podemos visualizar alguns intervenientes, assim como as interações que podem ocorrer.

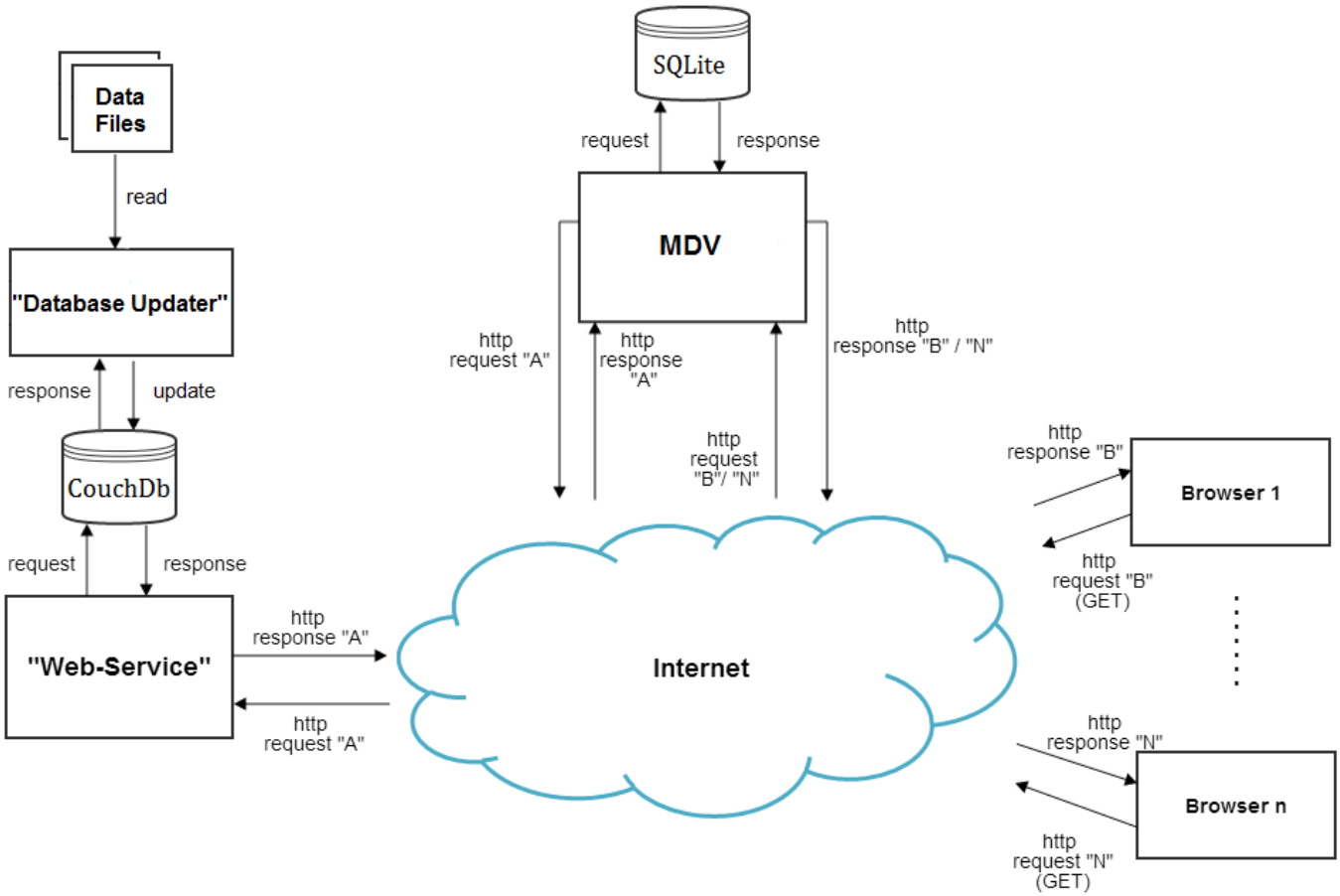


Figura 21 – Arquitetura geral do projeto.

5.2 – Arquitetura do "*Web-Service*"

O "*Web-Service*" implementado é uma peça fundamental do projeto no sentido em que é este sistema que fornece os dados e as informações das missões existentes na base de dados. Este serviço *web* disponibiliza uma API pública que tem a função de receber, tratar e responder aos pedidos provenientes da internet (através de pedidos que utilizem o protocolo *http*). Estes pedidos podem ser realizados por outras aplicações que necessitem dos dados existentes ou por utilizadores (que fazem pedidos, por exemplo, através do *browser* de internet).

Para desenvolver este "*Web-Service*" foram implementados alguns tipos de *design patterns* devido ao facto destes apresentarem soluções gerais e reutilizáveis no desenvolvimento de projetos de *software*. Estes padrões de projeto não são um código final, são uma descrição ou modelo de como resolver um problema e aplicam-se a linguagens de programação orientadas a objetos. Normalmente, estes padrões definem as relações e interações entre as classes ou objetos, sem especificar os detalhes das classes ou dos objetos envolvidos, sendo assim, considerados uma abstração [36]. Ao utilizar *design patterns*, torna-se possível reduzir a complexidade do projeto através da definição de abstrações que estão acima das classes e instâncias, facilitando o trabalho e tornando mais fácil realizar alterações ou adicionar funcionalidades no futuro.

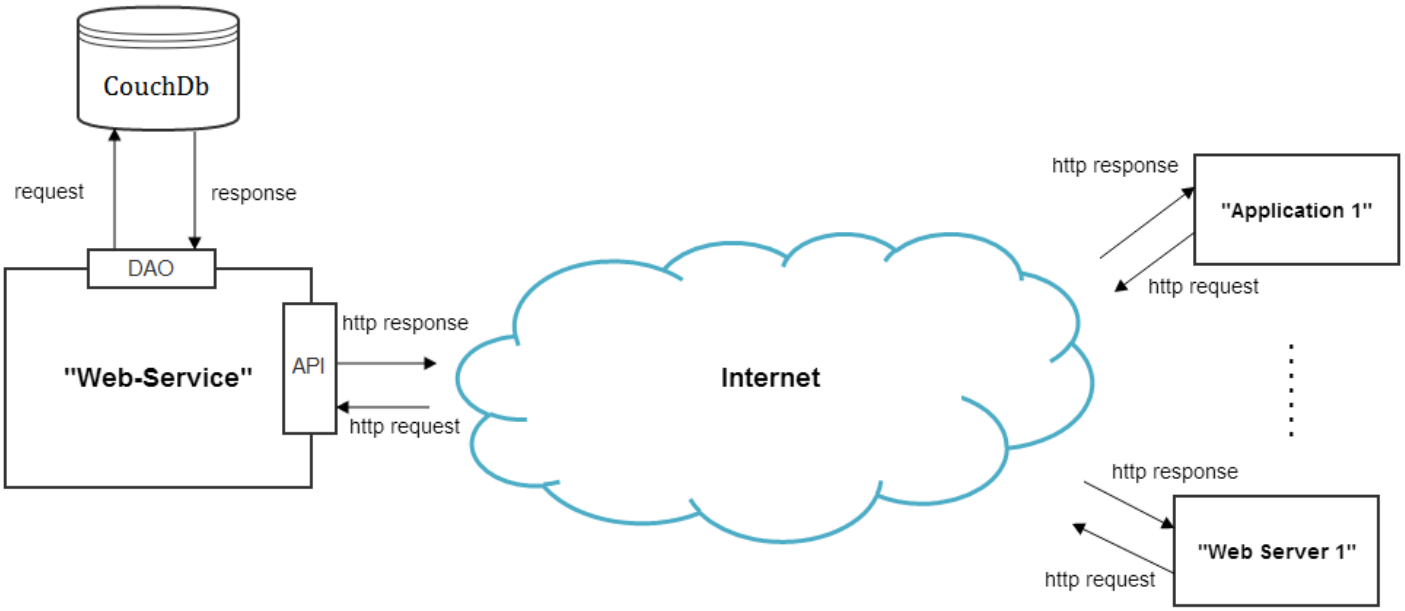


Figura 22 – Diagrama da arquitetura do "*Web-Service*".

Na Figura 22 está representada a arquitetura do *"Web-Service"*. Este realiza pedidos ("*request*") à base de dados e recebe a resposta ("*response*") sob a forma de documento representado no formato JSON. Também é possível ver que podem chegar pedidos ("*http request*") de entidades externas situadas na internet e que comunicam através do protocolo *http*. Estes pedidos são recebidos pela API pública, processados e seguidamente é construída uma resposta ("*http response*") contendo os dados requisitados no formato JSON. A API do "*Web-Service*" possui vários métodos que serão apresentados e discutidos na secção 5.2.2 – API pública.

5.2.1 - Tecnologias utilizadas

Para o desenvolvimento e implementação deste "*Web-Service*" foi necessário instalar e utilizar várias ferramentas que serão descritas de seguida.

O "*Web-Service*" foi desenvolvido e implementado utilizando o servidor *Apache Tomcat* [37]. Esta tecnologia é *open-source*, desenvolvida pela *Apache Software Foundation* e permite desenhar aplicações para a *web*. Não é necessário adquirir licenças para a sua utilização. Foi utilizada a versão *Apache Tomcat7*.

Com este servidor *Apache Tomcat* torna-se possível receber pedidos provenientes da internet através do protocolo *http,* caracterizados por *http requests* e, seguidamente, proceder às respostas utilizando o mesmo protocolo (*http response)*. No caso dos *http requests*, o "*Web-Service*" recebe uma instrução através de um comando GET composta pelos métodos que o cliente está a requisitar ao servidor e cria uma resposta *http response* com os dados requisitados em formato JSON.

Relativamente à base de dados utilizada no projeto, a escolha recaiu sobre a base de dados orientada a documentos chamada *CouchDB* [38]*,* desenvolvida pela *Apache Software Foundation*. É nesta base de dados que vão ser guardadas todas as informações e dados relativos às missões realizadas pela ESA. Esta base de dados tem como principal função receber e responder a pedidos provenientes do "*Web-Service*".

Para o desenvolvimento da API deste serviço foi utilizada a biblioteca *Jersey* [39] desenvolvida na linguagem de programação *Java* que é gratuita. Esta biblioteca permite a criação de serviços *Restful*.

5.2.2 - *API* pública

Foi desenhada uma *API* (*Application Programming Interface*) pública para este "*Web-Service*". É através desta que é possível receber pedidos *http* provenientes da internet, realizar o seu processamento e seguidamente proceder ao envio da resposta com os diversos conteúdos que foram requisitados. Esta resposta também utiliza o protocolo *http* e os dados são formatados em JSON. Todos os métodos criados e disponibilizados possuem o intuito de fornecer um maior número de informações e facilitar o processo de utilização deste sistema. Foram implementados os métodos seguintes:

- *help* – Comando para receber ajuda em geral sobre a utilização da *API*;
- *getMethods* – Comando utilizado para receber todos os métodos disponíveis;
- *getMissions* – Devolve o nome de todas as missões existentes no sistema;
- *getMissionInformation?mission=Mission* – Devolve toda a informação da missão "*Mission*";
- *getInstruments?mission=Mission* – Retorna todos os instrumentos relativos à missão "*Mission*";
- *getInstrumentInformation?instrument=Instrument&Mission=Mission* – Devolve a informação do instrumento "*Instrument*" relativo à missão "*Mission*";

- *getParameters?instrument=Instrument&Mission=mission* – Retorna o nome de todos os parâmetros existentes no sistema do instrumento "*Instrument*" e da missão "*Mission*";

- *getTimeRange?instrument=Instrument&mission=Mission* – Devolve todas as janelas temporais dos dados existentes relativos ao instrumento "*Instrument*" da missão "*Mission*";

- *getValuesParameter?mission=missiont&instrument=instrument&domain=domain&parameter=parameter&initd=inttime&endd=endtime* – Este comando é utilizado para receber o valor dos dados relativos ao parâmetro "*parameter*", missão "*mission*", domínio "*domain*", instrumento "*instrument*", data de inicio da medição "*initd*" e data de fim da medição "*endd*";

- *getErrorInformation?error=Error* – Este método é utilizado para receber mais informações sobre o erro "Error". Desta forma, é uma função muito útil para os utilizadores desta API;

- *getErrorList* – Esta função devolve a lista de todos os erros previstos pelo sistema.

Com os métodos discriminados acima torna-se possível à entidade que utiliza a API obter um grande número de informações e dados, como por exemplo, receber ajuda de como utilizar a API, receber todos os parâmetros de um instrumento ou receber a janela temporal onde estão definidos os dados relativos a um parâmetro. Também existem métodos com a finalidade de fornecer informações sobre erros ou problemas que ocorreram no pedido, como por exemplo, se o utilizador realizar um pedido errado ou então requisitar dados que não existem, é notificado com uma mensagem de erro e com um código numérico que o informa do tipo de erro que ocorreu. Esta funcionalidade é importante pois permite entender qual foi o erro ocorrido com o pedido. Também existe um método que disponibiliza todas as validações (tipos de erros que podem ocorrer) que são suportadas pelo "*Web-Service*".

5.2.3 – Base de dados

A base de dados possui um papel importante no projeto, na medida em que vai armazenar e gerir todos os dados das leituras dos instrumentos. No processo de escolha da base de dados é necessário ter em atenção vários fatores, como por exemplo, os tipos de dados que são importantes, as relações entre estes dados, os requisitos do sistema e o paradigma a utilizar.

Os termos *SQL* e *NoSQL* não se referem à utilização da linguagem de consulta estruturada (*Structured Query Language*) aplicada nas bases de dados relacionais. Ao contrário do modelo relacional, o termo *NoSQL* indica que estas bases de dados têm a característica de serem não relacionais e orientadas a documentos, o que se opõe ao paradigma das bases de dados relacionais.

Uma característica de notar nas bases de dados *NoSQL* é o facto de não ser necessário um esquema pré-definido de dados, tabelas e relacionamentos entre estes. Com este fator torna-se possível uma maior escalabilidade da base dados, no sentido em que para fazer algum tipo de alteração nos dados ou nos campos dos documentos, não é necessário atualizar todos os documentos do mesmo tipo, o que acontece nas tabelas das bases de

dados relacionais. Com este paradigma também não há a necessidade de verificar a integridade e os relacionamentos do modelo de base de dados relacional.

Entre os dois paradigmas anteriormente referidos, a escolha do modelo de base de dados para este projeto, recai sobre uma base de dados *NoSQL*. Esta escolha foi tomada devido a vários fatores, entre os quais:

- A importância de uma base de dados sem um esquema pré definido, deixando a possibilidade de ocorrem alterações, o que torna possível a modificação ou incorporação de mais campos com o decorrer do projeto, sem que seja necessário fazer uma alteração em toda a base de dados;

- A possibilidade de suportar vários formatos, como por exemplo, colocar uma imagem dentro de um documento, na base de dados;

- A necessidade de ter os documentos estruturados no formato JSON (*JavaScript Object Notation*), por ser fácil ao ser humano ler e escrever este tipo de ficheiros, facilitando também a comunicação com o "*Web-Service*" e com a aplicação *web*. Este formato possibilita a melhor estruturação, por hierarquia, dos dados que vão ser manipulados.

A base de dados *CouchDB* foi criada pela organização *Apache Software Foundation*. Está disponível deste o ano de 2005 e foi adotada por várias empresas, como por exemplo, BBC, *Skechers*, *Dimagi*, entre outras. Esta possui fatores favoráveis, como por exemplo:

- É caracterizada por ser uma base de dados orientada a documentos e, desta forma, não exige um esquema de dados previamente definidos. Sendo assim, é caracterizada por ser uma base de dados muito versátil.

- As operações de escrita na base de dados não bloqueiam as leituras, o que torna possível a escrita de grandes quantidades de informação sem que deixe de ser possível a leitura da base de dados. Para que isto seja permitido, esta base de dados aplica o conceito ACID [40] (Atomicidade, Consistência, Isolamento e Durabilidade). Existe uma replicação de dados e a deteção de conflitos, tornando assim os dados mais consistentes e oferecendo uma maior tolerância a falhas.

- É guardada uma versão antiga dos documentos, caso haja alguma falha de consistência de dados e seja necessário recuperá-los;

- Possui uma API REST acessível pelo protocolo *http* utilizando os métodos *POST*, *GET*, *PUT*, *DELETE* para as operações CRUD (*Create*, *Read*, *Update*, *Delete*);

- Funções de *Map/Reduce* através da linguagem de programação *JavaScript*.

Neste projeto, o tamanho dos dados pode crescer muito, pois os dados de várias missões ao longo de anos acarretam informação que ocupa vários *Terabytes* de espaço. Estes dados vão ser guardados em documentos do tipo JSON com hierarquia, definida sob a forma de relacionar dados com a informação relativa à data em que as medições foram efetuadas. A integridade e consistência dos dados tem de ser garantida pela base de dados. Desta forma, vão existir muitos documentos e relações de um para muitos. Estas relações dos documentos não são do tipo grafo.

Para consultar os documentos existentes na base de dados foi necessário proceder à criação de várias funções na linguagem *javascript* designadas de *views* (*map/reduce*). O "*Web-Service*"

realiza uma chamada às *views* para receber os dados ou as informações que foram requisitadas.

5.2.4 - Data Access Object

Neste projeto foram utilizados vários tipos de *design patterns*, nesta secção vai ser abordado o padrão designado por DAO (*Data Access Object*). Este tem a característica de possuir todos os mecanismos de acesso à base de dados, como por exemplo, conexão à base de dados, execução de comandos (*queries*) e mapeamento de dados para objetos *Java*.

Este padrão cria uma interface abstrata para ser utilizada pelas camadas lógicas que estão acima desta, resultando que todos os acessos à base de dados têm obrigatoriamente de passar por objetos DAO. Assim, toda a lógica que se encontra nas camadas superiores não necessita de ter conhecimento do tipo de base de dados existente ou do processo de aquisição dos dados [41].

As vantagens da utilização deste padrão consistem na separação rigorosa entre as partes importantes da aplicação que não podem conhecer quase nada uma da outra e, assim, conseguem evoluir independentemente. Caso a camada de lógica que se encontra acima sofra alterações, não implica nenhuma alteração na camada de acesso aos dados. Por outro lado, se for necessário alterar a camada de acesso aos dados (DAO) como por exemplo, alterar a base de dados, apenas têm de ser alterados os objetos DAO [41].

Na Figura 22 é possível visualizar a camada de acesso aos dados do "*Web-Service*", sendo uma abstração de como funciona o padrão DAO no projeto. É através destes objetos que são realizados os pedidos à base de dados e são recebidos e tratados as respostas com os dados. Quando as camadas lógicas acima dos objetos DAO necessitam de dados ou informações invocam os métodos dos objetos DAO e estes é que vão efetuar os pedidos, processar as respostas enviadas pela base de dados e posteriormente enviar os dados processados para a camada acima.

5.3 – Arquitetura da aplicação *web* - MDV

A aplicação *web* implementada fornece todas as funcionalidades para a visualização dos dados através de um *website*. Este é fornecido pelo servidor quando os utilizadores fazem um pedido ao domínio através do seu *browser*. Este servidor utiliza o "*Web-Service*" para fazer vários pedidos de dados e informações que são necessários para a representação gráfica dos dados.

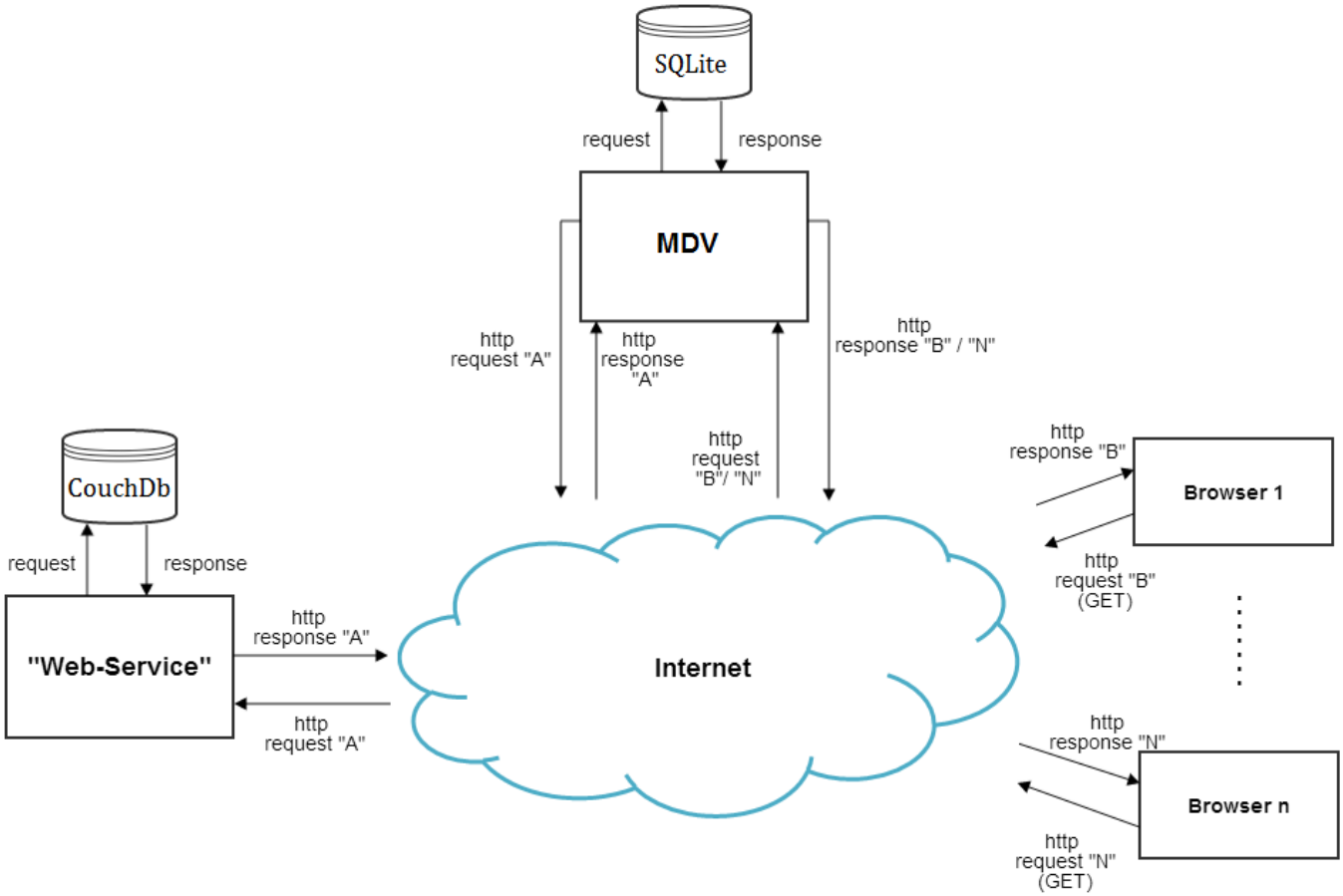


Figura 23 – Diagrama da arquitetura do "*Web-Server*".

No diagrama da arquitetura de alto nível da aplicação *web* (MDV) apresentado na Figura 23 é possível mostrar a existência de vários *browsers*, representados à direita, que fazem pedidos ao servidor onde está implementado a aplicação *web*. Estes pedidos (*GET*) são caracterizados, primeiramente, pela solicitação do ficheiro *html* referente ao *web site*. Os pedidos seguintes são realizados de acordo com a interação do utilizador com este *web site*.

Verifica-se que os pedidos são feitos de acordo com o protocolo *http*. A representação no diagrama da resposta ("*http request B/N*") e pedido ("*http response B/N*") significa que o "*Web-Server*" está a responder separadamente às solicitações provenientes do "*Browser B*" e do "*Browser N*". Os *Browsers*, entidades mais à direita do diagrama, são representações de navegadores que possibilitam o acesso à *Internet* dos utilizadores, ou seja, são os utilizadores que estão a realizar pedidos através dos diferentes *browsers*. Como exemplo destes *browsers* temos o *Google Chrome*, *FireFox*, entre outros.

Ao interagir com a aplicação *web* de visualização de dados, os utilizadores estão constantemente a requisitar informações e algumas destas estão presentes na base de dados e acessíveis através do "*Web-Service*". Desta forma, com a interação dos utilizadores, a aplicação MDV realiza vários pedidos de dados e informações ao "*Web-Service*". Estes pedidos podem ser vistos no diagrama da Figura 23 com o nome de "*http request A*" e é efetuada a resposta pelo "*Web-Service*" chamada de "*http response A*". Assim, através deste sistema distribuído torna-se possível fornecer todas as funcionalidades de visualização de dados que estão disponíveis aos utilizadores em geral.

Na arquitetura da Figura 23 verifica-se que a aplicação MDV possui uma base de dados *SQLite,* a comunicação é realizada através de pedidos ("*request*") e de respostas ("*response*"). Com esta base de dados relacional foi possível criar as funcionalidades de registo e autenticação na plataforma *online*, esta característica foi desenvolvida com o propósito de guardar as listas de parâmetros criadas pelo utilizador. Quando o utilizador faz a autenticação no *site* as suas listas de dados são carregadas automaticamente. Esta funcionalidade é extremamente útil para o utilizador no sentido em que guarda o contexto das variáveis configuradas para que na próxima visita ao *site* não seja necessário configurar as listas de dados novamente.

5.3.1 - Tecnologias utilizadas

Para o desenvolvimento desta aplicação *web* foram utilizadas diversas ferramentas, que vão ser descritas seguidamente.

A *framework Ruby on Rails* [42] foi utilizada para criar a aplicação *web* do projeto (plataforma de visualização *online*). Este utiliza a linguagem de programação *Ruby* e baseia-se no padrão de arquitetura MVC (*Model-View-Controller*). A *framework* é *open-source*, logo não é necessária a compra da licença para o desenvolvimento e disponibilização ao público. A decisão de utilizar esta *framework* foi um requisito da empresa *VisionSpace Technologies*.

Todos os conteúdos relativos ao *back-end* foram programados recorrendo à linguagem de programação *Ruby* que suporta o paradigma de programação orientada a objetos. Foi usada a linguagem de marcação *Html5* para produzir o *html* do *web site*, a linguagem de estilo *CSS* para controlar o *design* do *site* e, para suportar as ações relativas ao *Html5* recorreu-se à linguagem de programação *Javascript*. Para atualizar os conteúdos de forma dinâmica e proceder à comunicação com o servidor foi utilizada tecnologia *ajax*. O utilizador não tem a noção de todas as comunicações que são realizadas, o conteúdo que necessita visualizar é atualizado de forma assíncrona e não há a necessidade de refrescar o *site*.

Para aplicar as várias técnicas de visualização de dados na plataforma *web* foram utilizadas algumas bibliotecas desenvolvidas na linguagem *Javascript*. Com estas torna-se possível representar os dados graficamente e de forma interativa, permitindo a implementação de várias funcionalidades, como por exemplo, aplicar a visualização do tipo coordenadas paralelas ou a seleção e adição de parâmetros para as listas. Seguidamente são apresentadas as ferramentas utilizadas:

- *jQuery* – Esta biblioteca, desenvolvida utilizando a linguagem de programação *Javascript,* foi lançada no ano de 2006 pelo seu autor John Resig, possui uma licença gratuita e é *open-source*. Tem como objetivo a simplificação da programação do lado do cliente (*cliente-side*), facilita navegação e seleção de elementos do DOM, possibilita a criação de animações, gere eventos e torna mais fácil o

desenvolvimento de aplicações *ajax*. Esta também oferece a possibilidade da criação de *plugins* [43];

- *Arbor.js* – É uma biblioteca de visualização de grafos desenvolvida em *Javascript* que utiliza o *jQuery* e o algoritmo *force directed*, é *open-source* e possui uma licença gratuita. Com esta biblioteca torna-se possível criar uma visualização de um grafo interativo que mostra a organização e estrutura geral dos dados existentes. Permite também criar vários tipos de interação com o utilizador, como por exemplo, expandir um nó selecionado ou então adicionar um parâmetro à lista de dados [44];

- *Highcharts* – Esta é uma biblioteca de criação de gráficos desenvolvida na linguagem de programação *Javascript*. Possui uma licença gratuita se a aplicação desenvolvida não possuir fins comerciais e como este projeto é *open-source* não é necessário comprar a licença de desenvolvimento. O *Highcharts* permite a criação de vários tipos de representações gráficas para serem colocadas numa página *web* ou numa aplicação *web*, como por exemplo, gráficos de linha simples, gráficos de barras, gráficos de dispersão, gráficos circulares, gráficos de área, entre outros. Também possui um tipo de visualização gráfica dedicada à representação de dados temporais (*time series*) que inclui funcionalidades de navegação, apresentação do intervalo de tempo e opções de seleção temporal [33];

- *D3.js* – Também é uma biblioteca desenvolvida em *Javascript* que tem o propósito de controlar e criar visualizações interativas para uma página *web* ou para uma aplicação *web*. Através desta ferramenta foi possível utilizar dois *plugins* (*paracoords* e *splom*) no sentido de implementar a visualização de dados do tipo coordenadas paralelas e matrizes de dispersão de dados. Esta biblioteca possui uma licença BSD que impõem restrições mínimas à distribuição do código [45];

- *DateTimePicker* – É um *plugin* para *jQuery* desenvolvido na linguagem Javascript e tem como funcionalidade principal a criação de um método interativo de visualizar e permitir a seleção de datas. A sua licença é gratuita [46];

- *contextMenu* – Esta biblioteca é um *plugin* para *jQuery* que permite a criação de um menu de contexto personalizado quando o utilizador clica no botão direito do rato sobre certos elementos que estão dispostos na página *web*. A sua licença é gratuita [47];

- *Notify.js* – *Plugin* para *jQuery* que fornece notificações ao utilizador. Esta biblioteca possui licença gratuita [48].

Para proceder à implementação da aplicação *web* em modo de produção numa máquina virtual, alojada no servidor da empresa *VisionSpace Technologies*, foi utilizada a ferramenta *Phusion Passenger*. Esta é caracterizada por ser um servidor *web* que possui suporte para *Ruby on Rails* e é desenhada para integrar com o *Apache http Server*. Com estes pacotes de *software* instalados, a aplicação *web* está a correr numa máquina da rede interna da empresa.

5.3.2 – Base de Dados

A base de dados da aplicação *web* MDV permite guardar os dados de todos os clientes que utilizam esta ferramenta de visualização. Têm de ser guardadas as credenciais dos utilizadores e o seu ambiente de trabalho, ou seja, todas as listas, configurações e parâmetros que o utilizador cria ou altera.

É utilizada a ferramenta *SQLite* [49], que se trata de uma base de dados relacional desenvolvida na linguagem de programação C. É gratuita e não necessita de configurações nem de um sistema de gestão de base de dados (SGBD). Esta característica, por um lado, é positiva na medida que não é necessário instalar, configurar e gerir um sistema SGBD mas, por outro lado, uma desvantagem baseia-se no facto de que um SGBD fornece uma melhor proteção de *bugs* provenientes da aplicação (cliente), no sentido em que previne que a memória da base de dados seja corrompida.

Esta é uma base de dados completa no sentido que possui múltiplas tabelas, índices, *triggers* e *views* contidos num ficheiro alojado no disco. Desta forma, esta base de dados faz as leituras e escritas diretamente neste ficheiro de dados. O *SQLite* fica embutido na aplicação *web* através da *framework Ruby on Rails*.

5.3.3 – Modelo MVC

A *framework Ruby on Rails* baseia-se na arquitetura *Model-View-Controller*.

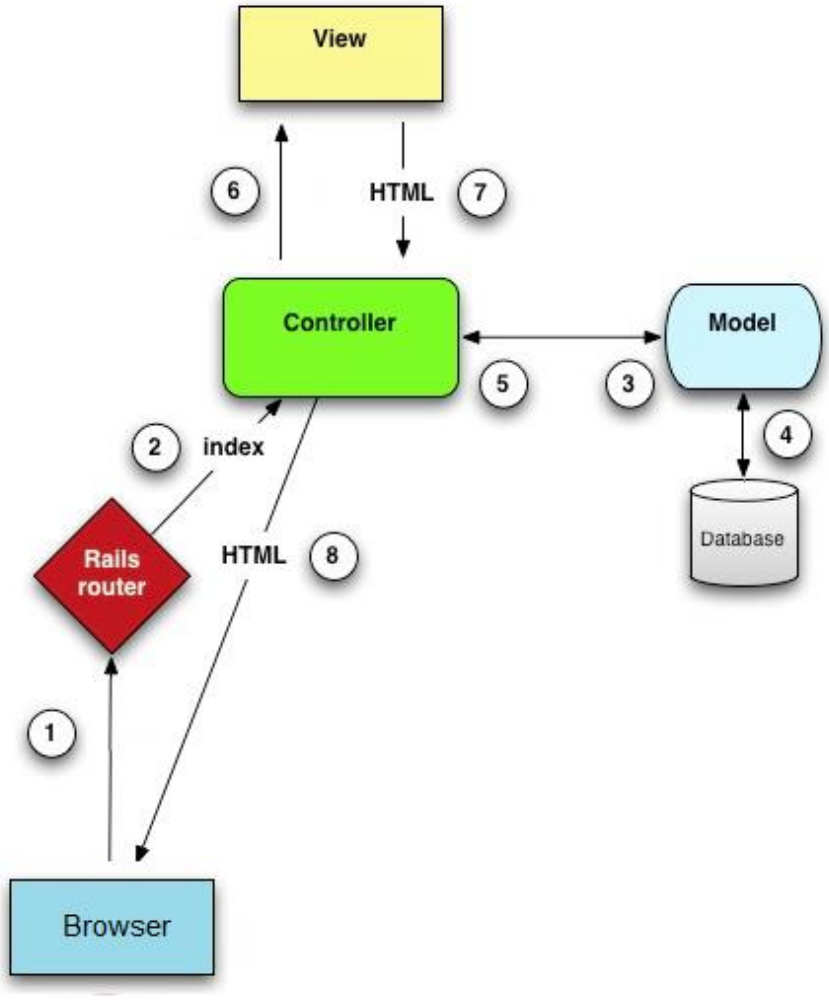


Figura 24 – Modelo MVC *Ruby on Rails*. [50]

A Figura 24 mostra o funcionamento da arquitetura *Model-View-Controller* presente na framework *Ruby on Rails*. Em primeiro lugar o cliente, através do seu *browser,* faz um pedido

ao URL (1) onde está a correr o servidor da aplicação *web*. Este é recebido pela *Rails routes* que direciona (2) o pedido ao *controller* indicado (este é responsável pela próxima ação a tomar). Posteriormente (3) o *controller* pode fazer uma chamada ao *model* (representa a informação) que consulta a base de dados (4) e esta responde com os dados (5). Estes são passados à *view* (template que é convertido para html) que constrói (6) a página *html* (7) e, por fim, o *controller* envia o *html* ao *browser* do cliente (8) [50].

5.4 – Arquitetura do "*Database Updater*"

O "*Database Updater*" tem a finalidade de atualizar a base de dados com os dados dos ficheiros que contêm as várias medições dos instrumentos utilizados pelas missões. Em primeiro lugar são abertos os ficheiros que contêm os dados, seguidamente é criado um ficheiro do tipo *JSON* com todos os elementos lidos e a base de dados é atualizada.

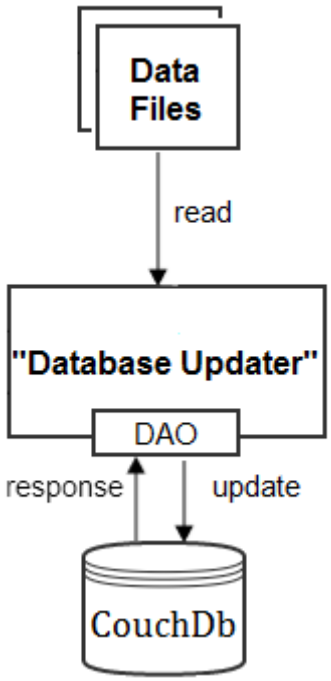


Figura 25 – Diagrama da arquitetura do "*Database Updater*"

5.4.1 - Tecnologias Utilizadas

O "*Database Updater*" foi desenvolvido utilizando a linguagem de programação orientada a objetos *Java*.

5.4.2 – Data Access Object

Neste sistema também foram utilizadas vários tipos de *design patterns*, nomeadamente o padrão DAO para a atualização da base de dados, como já foi descrito na secção 5.3.4 – Data Access Object.

5.5 – Comentários finais

Nesta secção foram apresentadas as arquiteturas de todos os sistemas que fazem parte do projeto de visualização de dados. Em cada um dos sistemas existem várias tecnologias envolvidas que são cruciais para cada funcionalidade implementada. Algumas tecnologias e ferramentas utilizadas fazem parte dos requisitos propostos pela empresa *VisionSpace Technologies*. No Capítulo 6 serão debatidos assuntos relativos ao desenvolvimento destes sistemas.

Capítulo 6 – Desenvolvimento

No decorrer deste capítulo vai ser especificado o desenvolvimento de todos os sistemas que foram criados no âmbito deste projeto.

6.1 – Desenvolvimento do "*Web-Service*"

O "*Web-Service*", como já foi referido anteriormente, possui uma API pública que tem o objetivo de responder a pedidos externos com informações e dados relativos às missões. Desta forma, este comunica com uma base de dados orientada a documentos que contém todos os dados e informações, nesta secção também vai ser explicado o desenvolvimento desta componente. O funcionamento deste serviço pode ser consultado no manual do utilizador presente no Apêndice B.

6.1.1 – Diagrama de *Packages*

Nesta secção vão ser explicadas todas as *packages* relativas ao desenvolvimento do "*Web-Service*". A partir da Figura 26 conseguimos obter uma ideia de alto nível do sistema implementado.

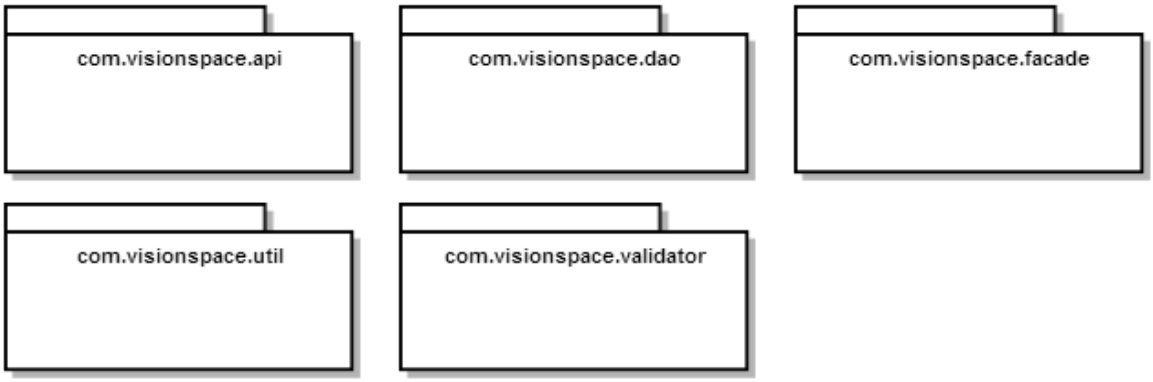


Figura 26 – Diagrama de *packages* "*Web-Service*"

Existem cinco *packages* relativas ao serviço *web*, cada uma possui uma função dentro deste "*Web-Service*". Explicação de cada uma das *packages*:

- com.visionspace.api : Esta *package* possui a *servlet* que implementa toda a API pública do servidor. É através desta que todos os pedidos são recebidos, filtrados e são invocados os métodos para o seu processamento, posteriormente a resposta é enviada ao cliente.

- com.visionspace.dao : *Package* que possui todas as classes DAO (*Data Access Objects*), ou seja, é através destas que a comunicação com a base de dados *CouchDB* é realizada no sentido de obter os dados requisitados pelos clientes. Toda a comunicação com a base de dados é realizada através destes objetos;

- com.visionspace.facade : Esta *package* possui uma classe que é uma API simplificada com o intuito realizar chamadas a outros métodos necessários para processar os

pedidos provenientes do exterior. Também possui uma classe que permite criar uma estrutura de dados do tipo JSON com os dados dos parâmetros;

- com.visionspace.util : Esta possui várias classes úteis para o funcionamento do projeto. Existe uma classe que possui métodos de resposta com ajuda (*html*) ao cliente, também possui a classe que fornece a ligação com a base de dados;

- com.visionspace.validator : Esta *package* possui todos os métodos com o intuito de fornecer a informação de erros existentes nos pedidos à API do "*Web-Service*".

6.1.2 – Diagrama de Classes

O diagrama de classes da aplicação "*Web-Service*" pode ser consultada na seguinte imagem (Figura 27).

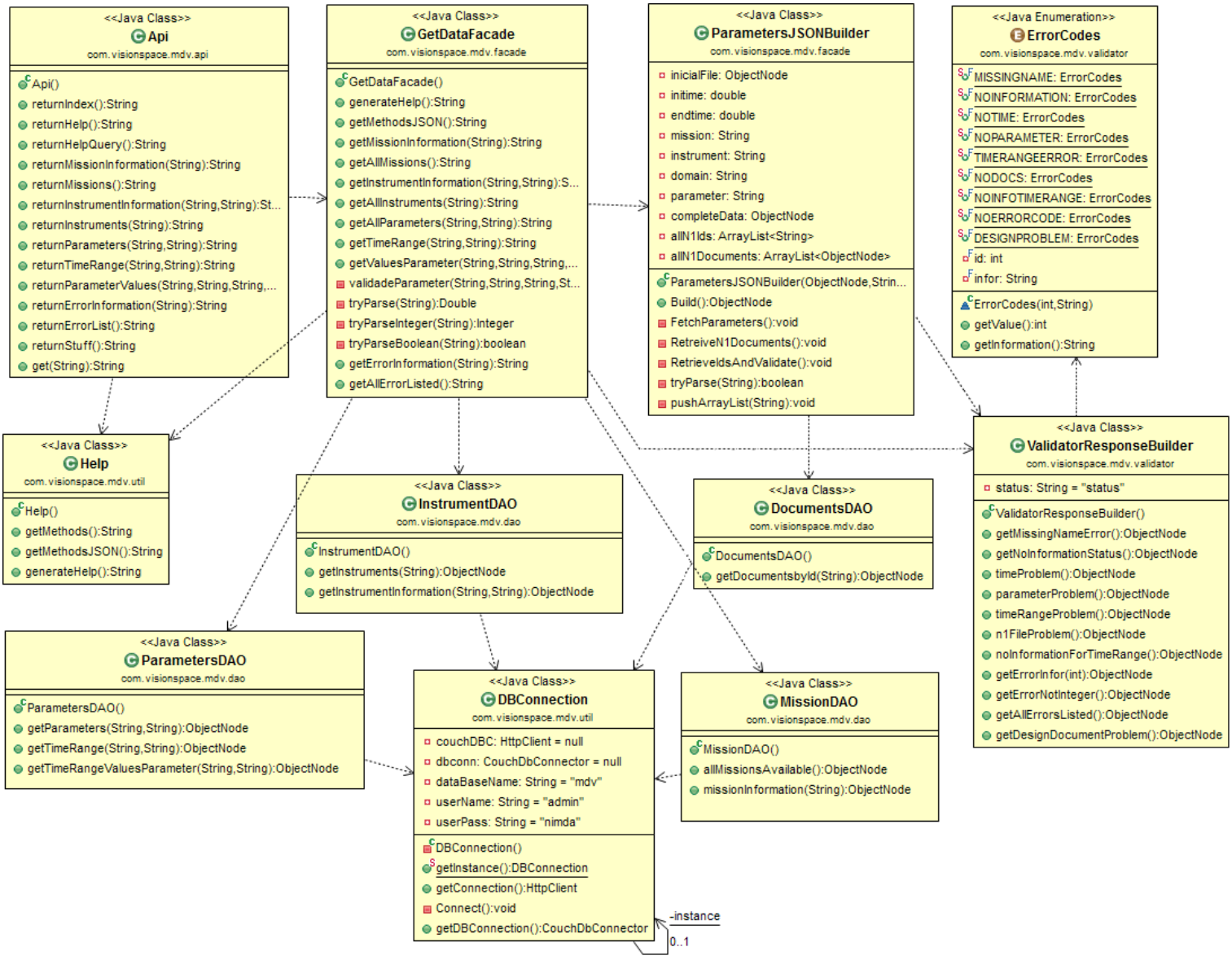


Figura 27 – Diagrama de classes do "*Web-Service*"

6.1.3 – *Design Patterns* Implementados

Como foi referido no capítulo 5, mais propriamente na secção 5.2 – arquitetura do "*Web-Service*", foram implementadas alguns tipos de *design patterns* devido ao facto destes apresentarem soluções gerais e reutilizáveis no desenvolvimento de projetos de *software*. Pode ser salientado ainda que estes padrões de projeto não são um código final, são uma descrição ou modelo de como resolver um problema e aplicam-se a linguagens de programação orientadas a objetos.

Como forma de aceder e requisitar dados e informações da base de dados (*CouchDB*) foi implementado o padrão DAO (*Data Access Object*). Este padrão está explicado na arquitetura do "*Web-Service*" secção 5.2.4 – *Data Access Object*. De forma resumida, este padrão tem a característica de possuir todos os mecanismos de acesso à base de dados separando todos os métodos de acesso aos dados. Este padrão cria uma interface abstrata para ser utilizada pelas camadas lógicas que estão acima desta.

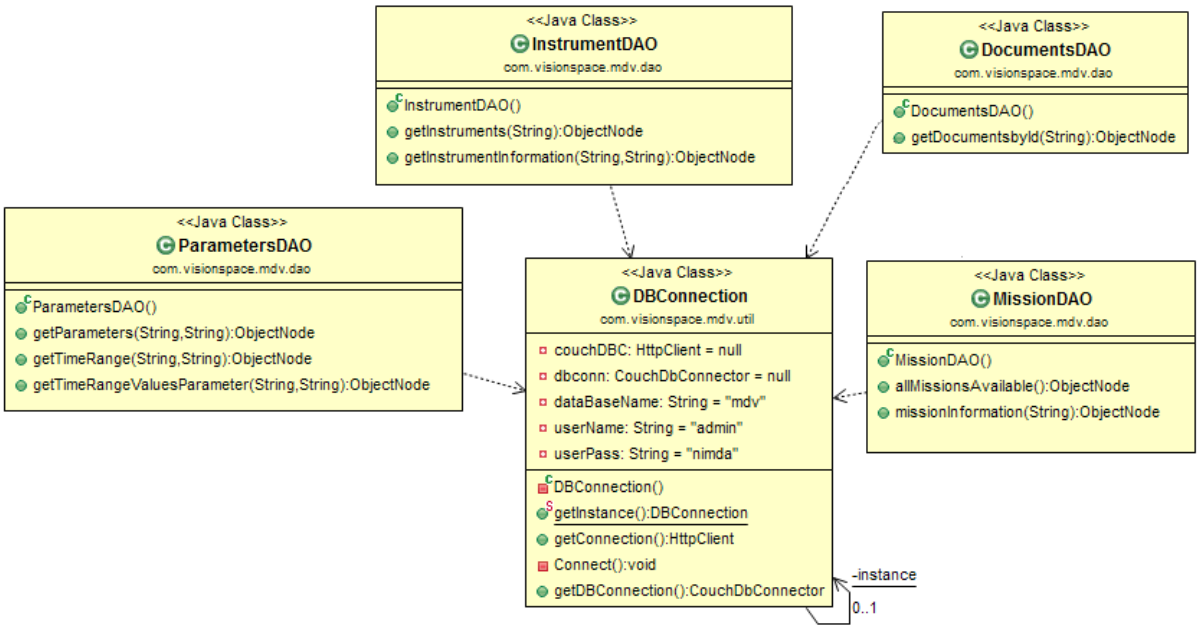


Figura 28 – Diagrama de classes DAO.

Na Figura 28 estão representadas as quatro classes que foram implementadas de acordo com o padrão de desenho DAO. O acesso aos dados foi dividido nas quatro classes com o intuito de separar e estruturar o código. Desta forma, todo o acesso à base de dados é feito através da utilização destas classes pelas camadas de lógica diretamente acima. A classe *DBConnection,* que aparece no diagrama de classes, tem o propósito de fornecer a conexão à base de dados.

A classe *DBConnection* foi desenvolvida de acordo com o padrão de desenho *singleton*, este padrão pode ser utilizado em projetos onde existem algumas classes que apenas devem ter uma instância. Desta forma, este objeto tem de ter a responsabilidade de criar e fornecer a sua própria instância, assim sendo, é possível garantir que apenas uma instância seja criada, além de fornecer um único ponto de acesso para esta instância [36].

Na Figura 28 pode ser visualizado a classe *DBConnection*. Esta possui o seu construtor privado e um método público (*getInstance*) que tem como objetivo permitir o acesso a esta instância. Desta forma, as instâncias das classes DAO utilizam o objeto *singleton* para efetuarem a comunicação com a base de dados. Com este padrão de desenho é possível garantir um ponto único de comunicação com a base de dados.

Foi também implementado um padrão de desenho chamado de *facade*, este disponibiliza uma interface simplificada que permite tornar o código mais fácil de usar e de entender. Desta forma, este *design pattern* pode ser visto no diagrama de classes da Figura 27, a sua classe possui o nome de *GetDataFacade*, esta cria uma abstração e uma interface simplificada para obter e processar todos os dados que são requisitados pelas instâncias da classe Api [36].

6.1.4 – Bibliotecas Externas e Ferramentas Utilizadas

Para realizar a comunicação com a base de dados foi utilizada a biblioteca externa *Ektorp*. Esta é uma biblioteca gratuita que fornece uma interface simplificada para toda a interação com a base de dados *CouchDB*. Esta também possui métodos de mapeamento de ficheiros no formato JSON para facilitar o processamento dos documentos recebidos pela base dados.

Para fazer a implementação (*deployment*) deste serviço *web* foi utilizada a ferramenta de *software* denominada de *Apache tomcat7*. Esta aplicação foi instalada numa máquina virtual na empresa *VisionSpace Techonologies* e o "*Web-Service*" encontra-se em execução neste servidor.

6.1.5 – Base de Dados CouchDB

Como já foi referido na arquitetura do "*Web-Service*" presente no capítulo 5.2, a base de dados instalada é denominada de *CouchDB* e tem a particularidade de ser uma base de dados orientada a documentos, esta possui algumas características que necessitam de ser implementadas. Nesta secção serão explicados o tipo de documentos que estão presentes na base de dados, assim como, os métodos para acedê-los.

Em primeiro lugar é fundamental explicar a forma de como os dados são requisitados. Para obter os dados é necessário fazer um pedido a um documento especial que está presente na base de dados, este documento é do tipo *_design* e tem a característica de possuir várias *views*. Estas *views* podem ser explicadas como sendo funções de *map/reduce*, os seus métodos são programados utilizando a linguagem *Javascript,* têm de receber como argumento um documento e necessitam de ter um método chamado *emit*(*key*,*value*). Assim sendo, para cada *view* existente na base de dados são processados todos os documentos e o resultado do método *emit* é guardado numa *B+Tree*. Quando um documento é alterado ou modificado apenas este é processado pelas funções *view* e a árvore *B+Tree* é atualizada. Desta forma, para efetuar um pedido de dados à base de dados *CouchDB* é necessário realizar uma chamada ao documento *_design* mencionando a *view* pretendida. A base de dados consulta a árvore *B+Tree* e devolve os resultados presentes nesta sob a forma de um documento JSON.

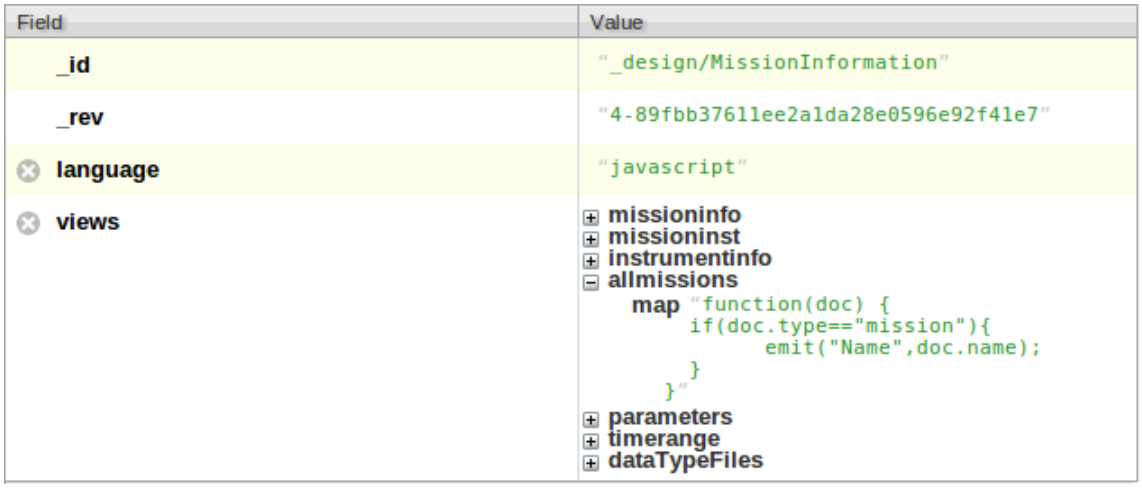


Figura 29 – Representação de um ficheiro do tipo *_design*.

Na Figura 29 está representado o documento do tipo *_design* com todas as *views* desenvolvidas no projeto, verifica-se que a *view allmissions* está expandida e é através desta que torna-se exequível obter a informação de todas as missões presentes na base de dados. Também é possível visualizar o código implementado utilizando a linguagem de programação *Javascript*.


```
{
   "_id": "db32369c7a26ab4733a4ea8c89000654",
   "_rev": "201-ede72f2a9ff4147cf5d2c40a4b37c1a4",
   "type": "mission",
   "name": "EnviSat",
   "information": "Information about EnviSat",
   "instruments": {
       "GOMOS": {
           "information": "instrument info",
           "parameters": {
               "mph": [
                   "product",
                   "proc_stage"
               ],
               "sph": [
                   "sph_descriptor",
                   "start_time"
               ],
               "nl_local_species_density": [
                   "dsr_time",
                   "quality_flag"
               ] "sun_azimuth_tangent"
               ]
           },
           "readings": [
               {
                   "tinit": "1.305154816546E12",
                   "tend": "1.305154858543E12",
                   "nlentry": {
                       "id": "cdd77f24-30d6-452b-b86a-938f8e6ae2e8"
                   }
               },
               {
                   "tinit": "1.305155255206E12",
                   "tend": "1.305155300703E12",
                   "nlentry": {
                       "id": "cc779420-db3b-4270-8157-6809b97ce532"
                   }
               }
           ]
       }
   }
}
```


Figura 30 – Documento do tipo *mission*.

Todos os documentos presentes na base de dados têm de estar formatados de acordo com o tipo JSON. A Figura 30 tem o propósito, meramente figurativo, de explicar o esqueleto de um documento do tipo missão existente na base de dados. De acordo com o documento (Figura 30) verifica-se que possui um campo (*key*) *id* que é o seu identificador único, ou seja, é como uma chave primária numa base de dados relacional e, a partir deste

id, o documento pode ser descarregado da base de dados sem ser necessário recorrer aos métodos presentes nas *views*. Também é possível identificar que a missão é denominada de *Envisat* e possui um campo (*key*) chamado de *instruments*, o valor deste contém todos os instrumentos relativos à missão assim como os seus domínios, parâmetros e as referências para os documentos de dados relativos ao instrumento. Desta forma, para aceder às leituras (dados dos parâmetros) do instrumento denominado GOMOS é necessário adquirir os vários *ids* (identificadores dos documentos) que estão presentes no campo (*key*) chamado *n1entry*.

```
{
   "_id": "cbf2435a90f3f31c489b006470000a98",
   "_rev": "7-d86a9746446bb95b07b4e5b920248212",
   "type": "data",
   "mission": "EnviSat",
   "instrument": "Doris",
   "DORIS_PRECISE_ORBITS": {
       "1": [
           {
               "index": "1",
               "fieldname": "utc_time",
               "value": "1401455126813",
               "units": "UTC"
           },
           {
               "index": "7",
               "fieldname": "x_pos",
               "value": "1663441.593",
               "units": "m"
           },
           {
               "index": "9",
               "fieldname": "y_pos",
               "value": "1071773.273",
               "units": "m"
           },
           {
               "index": "11",
               "fieldname": "z_pos",
               "value": "-16892085.607",
               "units": "m"
           }
       ],
```


Figura 31 – Estrutura de um documento de dados.

Para além dos ficheiros que foram referidos anteriormente existem na base de dados documentos que contêm os dados resultantes das medições efetuadas pelos instrumentos. Desta forma, verifica-se na Figura 31 um exemplo, meramente figurativo, da estrutura de um documento de dados no formato JSON. Podemos visualizar que se trata da missão *Envisat*, instrumento DORIS e que o domínio dos dados é o "DORIS_PRECISE_ORBITS". Podemos ainda constatar que os valores dos parâmetros estão organizados como sendo vários objetos JSON inseridos num *array*. Este irá possuir todos os valores relativos a um ficheiro de medições, ou seja, inclui os valores de um ficheiro de dados fornecido pela ESA (como por exemplo um ficheiro .N1).

6.2 – Desenvolvimento da aplicação *web* – MDV

A aplicação *web* é a componente do projeto onde irá ocorrer uma elevada interatividade com os utilizadores, assim sendo, a plataforma de visualização de dados possui vários métodos e várias funcionalidades que auxiliam os utilizadores no processo de exploração de dados. É através desta plataforma que os vários utilizadores podem selecionar os dados das missões, agrupá-los em listas e posteriormente utilizar as funcionalidades de visualização dos dados selecionados.

Para não tornar este subcapítulo demasiado extenso não vão ser apresentadas imagens de exemplo dos tipos de visualização implementados na aplicação *web*. Porém, foi desenvolvido um manual do utilizador que possui todas as informações (incluindo imagens) relativas à utilização e funcionamento da plataforma de visualização *online*. Este cobre todas as funcionalidades e inclui representações gráficas de exemplo. Desta forma, este manual encontra-se no Apêndice C e pode ser consultado para compreender como a plataforma funciona.

6.2.1 – *Framework Ruby on Rails*

Como já foi referido no capítulo 5 (arquitetura dos sistemas), a *framework Ruby on Rails* segue o padrão MVC (*Model-View-Controller*) e foi utilizada para desenvolver a aplicação *web*. De acordo com esta metodologia foi necessário, em primeiro lugar, proceder à configuração do ficheiro *routes,* sendo neste descritas as ações a tomar quando um pedido é recebido pelo servidor, ou seja, quando chega um novo pedido à aplicação *web* o ficheiro *routes* direciona-o para o controlador correto, este executa os métodos implementados podendo recorrer aos *models* e seguidamente constrói a *view*.

Nesta secção vão ser apresentados os diagramas dos *controllers*, *models* e *views* desenvolvidos na aplicação *web*. Estes têm o propósito de fornecer uma visão mais técnica sobre o sistema implementado. Relativamente aos controladores e modelos vão ser apresentados os seus diagramas contendo os seus atributos e métodos, por outro lado, quanto às *views* apenas vão ser descritas pois estas tratam-se de *templates* que, posteriormente, são convertidos para *html*. Desta forma, foram implementados os seguintes controladores:

- *application_controller* : Controlador que constrói o *template* que contém a *view* principal da aplicação *web*;
- *pages_controller* : Controlador que possibiliza a construção da *view* onde existe o conteúdo *html* relativo à *home page* da aplicação *web* e é através desta que toda a aplicação é inicializada;
- *data_controller* : Este controlador possui vários métodos que possibilitam o processamento e a troca de dados entre a aplicação *web,* que está a correr no servidor, e a página *html* que está a correr do lado do utilizador. Quando o cliente necessita dos dados relativos a um parâmetro realiza um pedido a este controlador que, por sua vez, efetua um pedido ao "*Web-Service*". Os dados são recebidos e processados, seguidamente o controlador envia-os de volta à aplicação que está a correr do lado do cliente.

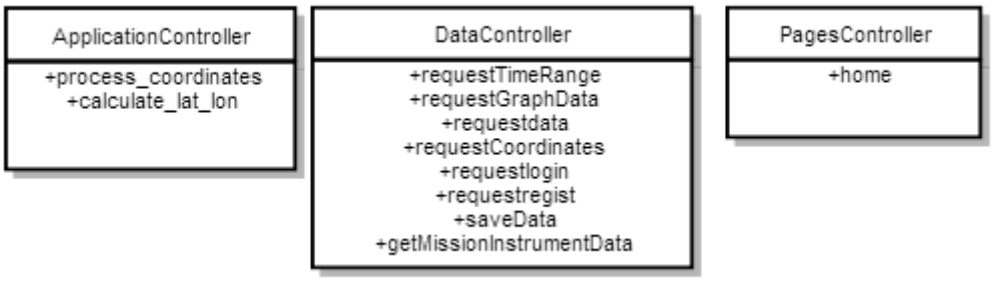


Figura 32 – Representação dos *controllers* existentes na aplicação *web*.

Na Figura 32 está a representação dos controladores da aplicação *web* do projeto. Verifica-se que o *PagesController* possui o método *home*. Este é chamado quando o cliente faz um pedido ao servidor no sentido de requisitar o *html* da aplicação *web*, ou seja, existe uma configuração no ficheiro *routes* indicando que este método deve ser executado quando um cliente requisita o *html* da página. O controlador *DataController* recebe pedidos (do tipo POST) e consome um objeto do tipo JSON. Novamente o ficheiro *routes* foi configurado para permitir que os métodos deste controlador sejam executados corretamente. Este controlador tem a característica de fazer as chamadas ao "*Web-Service*", processar a resposta e seguidamente efetua a resposta. Explicação dos métodos deste controlador:

- *requestTimeRange*: Tem o objetivo de responder com o intervalo temporal das medições realizadas por um determinado instrumento. Desta forma, realiza vários pedidos ao "*Web-Service*", processa e agrupa os dados para enviar como resposta ao cliente;

- *requestGraphData*: Este método tem como objetivo fornecer os dados para a construção do grafo onde é apresentado a estrutura geral dos dados existentes. Para que isto seja possível são realizados vários pedidos ao "*Web-Service*" com o propósito de construir um objeto no formato JSON;

- *requestdata*: Possui como principal objetivo enviar os dados de um parâmetro. Para isto necessita de receber todas as informações necessárias para recorrer ao "*Web-Service*" e assim, obter os dados necessários, processá-los de acordo com o formato de séries temporais e criar um objeto no formato JSON;

- *requestCoordinates*: Este método possui a função de enviar as coordenadas ao cliente num objeto com formato JSON;

- *requestlogin*: Método que tem a função de receber e validar os dados de um utilizador que esteja a tentar fazer a autenticação na aplicação *web* que está em execução no *browser* do cliente;

- *requestregist*: Esta função tem como objetivo realizar o registo de um cliente na plataforma *web*;

- *saveData*: Método que possui a função de guardar o ambiente de trabalho de um cliente que esteja a utilizar a aplicação *web*;

- *getMissionInstrumentData*: Este método tem como funcionalidade fornecer todas as informações das missões e dos seus instrumentos. Para que isto seja possível é necessário recorrer à API do "*Web-Service*", construir um objeto JSON e responder ao cliente com a informação.

Nesta aplicação *web* foi desenvolvido um modelo que representa todos os utilizadores registados no sistema. Neste são guardados os dados relativos ao *email*, *password* e o seu ambiente de trabalho, este representa todas as listas criadas, configurações efetuadas e parâmetros selecionados. Este modelo tem o nome de *Users* e viabiliza a funcionalidade de guardar o ambiente de trabalho criado e configurado pelo utilizador durante a seleção e exploração dos dados. Sempre que o utilizador efetua alguma alteração nas listas ou nas suas configurações, estes dados são atualizados recorrendo a este modelo. Desta forma, quando o utilizador volta à plataforma de visualização de dados e realiza a sua autenticação, todo o seu ambiente de trabalho é descarregado e colocado à sua disposição para que este

consiga proceder à sua visualização sem perdas de tempo na criação e configuração de novas listas de parâmetros.

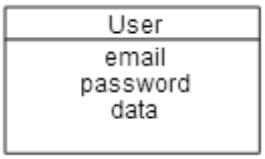


Figura 33 – Tabela *User*.

O modelo presente na Figura 33 é denominado *User* e possui os campos: *email*, *password* e a *data*. Este modelo representa os dados guardados na base de dados e tem como principal objetivo guardar o ambiente de trabalho do cliente. O processo de guardar o ambiente de trabalho resulta numa funcionalidade essencial para a interação com o utilizador porque sempre que o utilizador voltar a utilizar a ferramenta de visualização de dados, todo o seu ambiente de trabalho que foi guardado é descarregado e colocado à sua disposição de uma forma transparente.

Relativamente às *views* existentes na aplicação *web*, a fundamental é a *view home* que pertence ao controlador *PagesController* pois é esta que possui o código *html* que serve de esqueleto para a plataforma de visualização. Outra *view* importante é a *aplication* (relativa à *aplicationController*) pois é nesta que são incluídos no seu *html* os ficheiros *javascript* e *css* que desempenha funções diferentes.

6.2.2 – Conteúdos *Javascript*

Quando o *html* da aplicação *web* é totalmente carregado pelo *browser* do utilizador, através da renderização da *view home,* a aplicação do lado do cliente passa a ser totalmente controlada pela linguagem de programação *javascript*. Esta linguagem fornece as ações necessárias para controlar o *html* e foi utilizada porque todas as bibliotecas de visualização de dados foram desenvolvidas nesta linguagem. Assim, o processamento da visualização é realizado no lado do cliente através de *javascript*.

A biblioteca *jQuery* tem o propósito de facilitar a navegação e o acesso aos elementos do DOM, criar e gerir eventos (como por exemplo, cliques do rato em elementos da página) e facilitar o desenvolvimento de métodos *ajax*. Desta forma, esta biblioteca foi usada para gerir a interação do utilizador com a aplicação (baseada em cliques do rato), adicionar e remover novos elementos *html* do DOM (acrescentando novos conteúdos ao *site*) e recorrer a pedidos assíncronos *ajax*. Estes pedidos *ajax* são realizados da aplicação que está em execução no *browser* do cliente e tem como destino o servidor onde está a decorrer a execução da aplicação *web*. Possuem a característica de serem assíncronos e através desta tecnologia são requisitados os dados dos parâmetros, as janelas temporais onde os parâmetros selecionados estão enquadrados e todas as informações relativas às missões, instrumentos e parâmetros. Ao receber estes pedidos a aplicação *web,* que está a "correr" no servidor, realiza pedidos à API do "*Web-Service*" para conseguir os dados requisitados. Depois de recebida a resposta do serviço *web*, a informação é processada e enviada para o *browser* do utilizador, sendo devolvida ao método *ajax* desenvolvido.

Foi desenvolvida uma estrutura de dados com o nome de *ListObj* que vai representar cada lista de parâmetros criada pelo utilizador, ou seja, esta estrutura vai possuir um nome, um *array* de parâmetros onde são introduzidos todos os parâmetros selecionados pelo

utilizador, os dados relativos a estes parâmetros, assim como, o tempo inicial e tempo final escolhido pelo utilizador (esta janela temporal diz respeito ao intervalo de tempo requerido para os valores dos parâmetros). Esta estrutura vai conter vários métodos que foram implementados no sentido de permitir manipulações numa lista, assim, possui métodos de introdução de parâmetros, remoção de parâmetros, introdução dos tempos inicial e final, entre outros. Esta estrutura de dados vai ser utilizada para aceder aos conteúdos criados e selecionados pelo utilizador, assim como, aceder aos valores dos parâmetros existentes em cada lista com o propósito de gerar representações gráficas de dados.

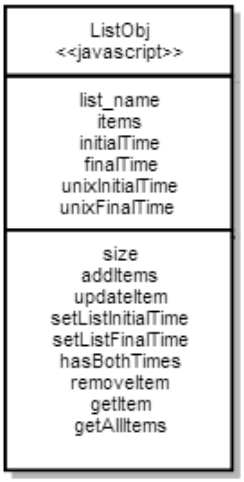


Figura 34 – Representação da classe "ListObj".

Para manipular e criar os objetos do tipo *ListObj* (presente na Figura 34) foi desenvolvida uma classe com o nome de *UserLists*. Esta contém um *array* de *ListObj* e, através de um objeto desta classe, vão ser realizadas todas as interações com os objetos do tipo *ListObj* (listas de dados), como por exemplo, criar, apagar, renomear ou adicionar dados aos parâmetros das listas. Foi implementada uma funcionalidade nesta classe que permite descarregar os dados dos parâmetros de forma transparente ao utilizador, a partir de métodos *ajax*. Assim o utilizador cria listas, adiciona parâmetros, procede à configuração do tempo inicial e do tempo final da lista. Estas ações permitem reunir todas as condições necessárias para que um pedido de dados seja realizado. Quando a aplicação *web* recebe o pedido *ajax* recorre à API pública do "*Web-Service*" como forma de requisitar os dados. Posteriormente, quando os dados são devolvidos ao serviço *web*, estes são processados e organizados tornando possível o seu envio ao método *ajax* que está em execução no *browser* do cliente. Desta forma, os parâmetros incluídos nas listas são atualizados com os dados e podem ser representados através de vários métodos de visualização. Na Figura 35 pode ser visto o diagrama que relaciona as duas classes acima descritas.

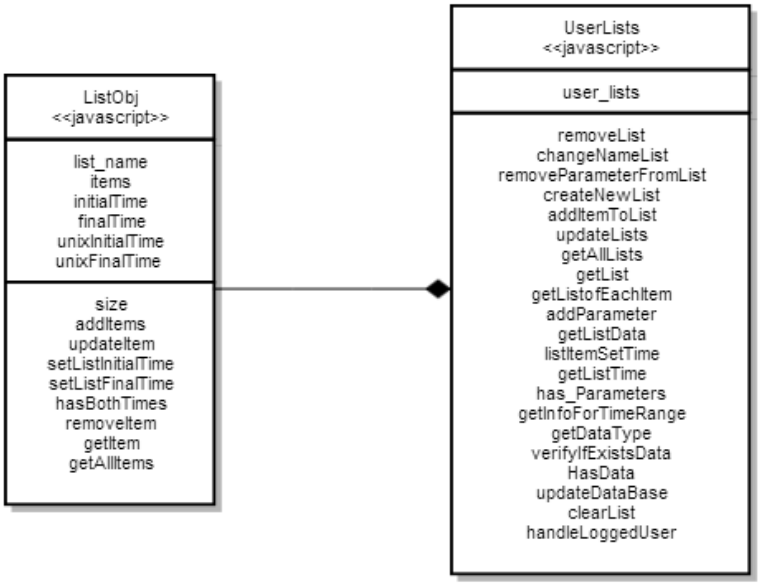


Figura 35 – Diagrama que representa as classes *ListObj* e *UserLists*.

6.2.3 – Visualização do Grafo

Foi utilizado um método baseado na interação com um grafo para mostrar a estrutura dos dados existentes, ou seja, através desta técnica é possível representar de forma gráfica e interativa os parâmetros, domínios, instrumentos e missões que existem no sistema de base de dados do projeto. Desta forma, o utilizador possui uma visão de alto nível sobre a organização geral dos dados e consegue ter uma melhor perceção das estruturas que contêm todos os dados das missões realizadas. Para isto foi utilizada uma biblioteca escrita na linguagem de programação *javascript* denominada de *arbor.js*, esta torna possível a criação de um grafo recorrendo à técnica *force-directed* que tem a característica de introduzir nos vértices e arestas propriedades, como por exemplo, repulsão e rigidez. Para realizar a implementação desta biblioteca no contexto do projeto de visualização de dados foi desenvolvida uma classe em *javascript* com o propósito de criar e gerir o grafo, denominada por *index_visual*. Para criar o grafo é necessário possuir toda a informação relativa às missões, instrumentos, domínio e parâmetros organizadas num objeto hierárquico, onde no topo da hierarquia está o nome da missão e na base estão todos os parâmetros. Esta informação é conseguida através de um pedido *ajax* à aplicação *web* que está em execução no servidor. Esta ao receber o pedido utiliza a API pública do "*Web-Service*" para requisitar as informações necessárias. De seguida, processa e organiza as informações no sentido de criar um objeto do tipo JSON enviado como resposta ao *browser* do cliente.

Para a implementação deste tipo de visualização recorri à funcionalidade *canvas* do *html5* e de uma biblioteca externa denominada de *contextMenu* com o intuito de criar o menu de contexto quando o cliente clica num parâmetro com o botão direito do rato. Esta funcionalidade permite ao utilizador selecionar os parâmetros e introduzi-los nas listas que foram criadas. Para isto, o objeto que procede à construção do grafo (*index_visual)* recorre ao objeto *UserLists* com o intuito de inserir os parâmetros nas listas selecionadas pelo cliente. Por defeito, os vértices do grafo são representados por círculos coloridos, estes foram substituídos por imagens alusivas às missões no espaço com o propósito de aproximar a visualização ao contexto do projeto. Desta forma, foi utilizada uma imagem de um satélite para representar cada missão existente, outras imagens foram utilizadas para representar o instrumento e o domínio. Relativamente à especificação dos argumentos

relativos à repulsão e rigidez que por defeito estão definidos como: repulsão = 1000 e rigidez = 600 e foram alterados para: repulsão = 30 e rigidez = 500. Estas alterações foram introduzidas porque com os valores de repulsão elevados e muitos vértices a representação gráfica deixa de funcionar devido ao facto dos vértices saírem da janela de visualização, uma vez que existe um nível elevado de repulsão entre estes. Assim, após algumas tentativas o valor fixado foi de: repulsão = 30.

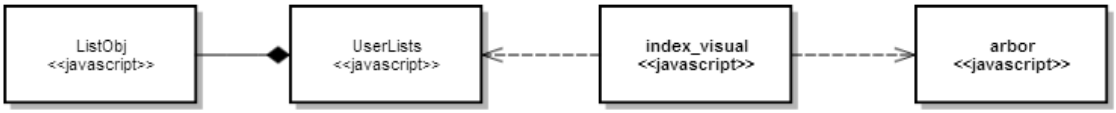


Figura 36 – Diagrama de classes da visualização do grafo.

Na Figura 36 é possível visualizar o diagrama de classes. Esta representação é simples na medida que não possui os atributos nem os métodos das classes. Apenas apresenta as relações entre as classes importantes para este tipo de visualização.

6.2.4 – Visualização do Intervalo de Tempo

Para mostrar os intervalos temporais que existem, relativos às medições dos instrumentos, foi utilizada uma técnica de visualização recorrendo a um gráfico de linha com muitas alterações. Esta funcionalidade tem como objetivo informar de forma visual e interativa os intervalos de tempo que existem no sistema relativos aos parâmetros que foram selecionados para uma certa lista. Assim, o utilizador necessita de recorrer a este tipo de visualização para conseguir configurar o tempo inicial e o tempo final que pretende para os parâmetros selecionados. Deste modo, são representados graficamente linhas que significam que o valor existe num determinado intervalo temporal, ou seja, se o utilizador visualizar uma linha contínua de uma data até a outra significa que existem dados para os parâmetros indicados no intervalo temporal indicado. Também vão existir lacunas nas linhas que representam o tempo das medições, isto significa que não existem medições no espaço temporal onde a linha não está definida. Cada linha é codifica com uma cor diferente com o objetivo de representar, com maior diferenciação, os intervalos temporais existentes, isto acontece devido ao facto de poderem existir parâmetros que não possuam medições no mesmo intervalo temporal. Para selecionar o tempo inicial e final pretendido o utilizador apenas necessita de clicar e arrastar o rato no intervalo de tempo pretendido diretamente na janela do gráfico. Com esta ação, automaticamente são atualizados os campos relativos ao tempo inicial e tempo final da lista que está selecionada.

A classe desenvolvida para realizar este tipo de visualização é denominada de *ListManager* e utiliza a classe *UserLists* com o propósito de realizar as alterações do tempo inicial e tempo final, relativo à lista onde foram selecionados os tempos. Existe um método que realiza a comunicação com a aplicação *web* que está a executar no servidor utilizando a ferramenta *ajax*. Desta forma, são requisitados todos os intervalos temporais para os parâmetros que existem na lista selecionada e, assim, a aplicação *web* recorre à API do "*Web-Service*" com o propósito de receber o intervalo temporal requisitado. Ao receber os dados, a aplicação *web* necessita de os processar, organizar e criar um objeto JSON com os intervalos temporais. A aplicação que está em execução no *browser* do cliente recebe todas as informações que necessita e procede à representação gráfica do intervalo temporal existente. A ferramenta utilizada para a representação do gráfico é denominada de *Highcharts* e o tipo de gráfico tem o nome de *StockChart*. Este gráfico foi alterado com o intuito de remover os eixos dos y,

pois estes não representam qualquer valor e foram removidas também as informações relativas a cada variável que ficavam na parte inferior do gráfico.

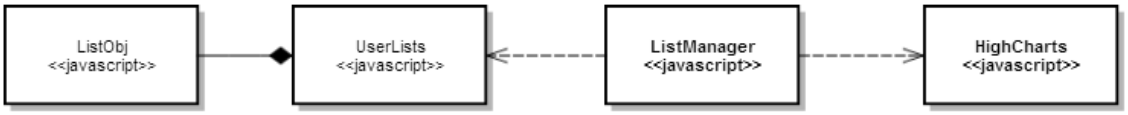


Figura 37 – Classes da visualização do intervalo temporal.

Na Figura 37 é possível visualizar o diagrama de classes relativo a este tipo de representação gráfica sob uma perspetiva simplista onde apenas é mostrado os relacionamentos das classes necessárias para este tipo de visualização.

6.2.5 – Visualização em Tabela

Para tornar possível a visualização de uma forma crua de todos os dados existentes em cada parâmetro, existe o método de visualização através de tabelas que contêm os valores dos parâmetros e o seu tempo. Desta forma, foram utilizadas as tabelas que existem em *html* e foi desenvolvida uma classe denominada de *ShowParameters_handler,* que recorre à classe *UserLists* que é indicada para a requisição dos dados dos parâmetros, caso existam. Assim sendo, os valores dos parâmetros são requisitados e incluídos nas células da tabela seguindo a lógica de apresentar o seu valor e o respetivo tempo em que a medição foi efetuada.

Na vista principal da visualização de dados com tabelas é possível verificar a existência de todos os parâmetros, aqui é introduzida a funcionalidade de remover um parâmetro da lista. Foi utilizada a biblioteca *contextMenu* com o propósito de criar um menu de contexto quando o utilizador clica com o botão direito sobre o parâmetro. Se o utilizador realizar a ação de remover o parâmetro da lista é utilizada a classe *UserLists* para concluir esta ação.

Figura 38 – Diagrama visualização em trabela.

De acordo com a Figura 38 é verificado que são apresentadas as classes e os seus relacionamentos essenciais para este tipo de visualização. Desta forma, não foram apresentados os métodos ou os atributos de cada uma das classes.

6.2.6 – Gráfico de Linhas

Foi desenvolvido nesta aplicação *web* um tipo de representação gráfica capaz de apresentar visualmente dados no contexto de multivariável, para isto foi utilizado um tipo de gráfico muito comum: gráfico de linhas. Para tornar exequível esta funcionalidade foi implementada a classe *SimpleLineGraphic* que recorre à biblioteca externa *Highcharts* e à classe *UserLists*. Desta forma, a utilização da classe *UserLists* prende-se com a necessidade de aceder aos dados dos parâmetros selecionados pelo utilizador neste tipo de visualização. Quanto à biblioteca *Highcharts* é utilizada para gerar o tipo de visualização do tipo gráfico de linhas.

Inicialmente quando o utilizador seleciona um parâmetro, este é apresentado de forma visual de acordo com a colocação da variável independente (o tempo) no eixo dos x e a variável dependente no eixo os y. Isto acontece porque as variáveis são do tipo séries temporais, ou seja, os parâmetros estão definidos em relação ao tempo. Para tornar esta funcionalidade exequível são requisitados os dados através da classe *UserLists*. Após receber os dados, estes são processados no sentido de colocá-los no formato aceite pelo tipo de gráfico, ou seja, existe uma função que percorre os dados e retira a informação relativa ao valor e ao tempo da medição. Desta forma, os dados são colocados em séries onde existem *arrays* que possuem a informação do tempo da medição e a informação do valor da variável. Caso o utilizador selecione outro parâmetro da lista existe a funcionalidade de acrescentar um novo eixo ao gráfico, podendo ser adicionada uma nova dimensão à visualização. Assim, o novo parâmetro é introduzido no mesmo gráfico e possui o mesmo tratamento do anterior (criação de uma série com os novos dados), tal acontece porque continua a haver a representação dos valores dos parâmetros em relação ao tempo, logo o eixo do x continua o mesmo. A nova informação é codificada com uma cor diferente e com um glifo diferenciado permitindo a sua diferenciação. A cor é gerida por uma classe que foi desenvolvida e que possui a denominação de *ColorManager* (existe um limite de cores disponíveis, após passar esse limite a variável é codificada com uma cor já utilizada). Desta forma, podem ser adicionados vários parâmetros à mesma janela de visualização podendo ou não acrescentar uma nova dimensão à visualização e, assim, permitir a comparação dos valores representados visualmente. Quando o utilizador pretende remover um parâmetro que já se encontra visível na representação gráfica é necessário aceder a todas as séries existentes na visualização, identificar a que é para remover e removê-la da visualização.

Foi implementada a funcionalidade de compactar os parâmetros que são representados na janela do gráfico, ou seja, se a próxima variável que for introduzida no gráfico for do mesmo tipo (unidades) da variável que já está representada, existe a possibilidade de representar os parâmetros no mesmo eixo dos y, para isto é necessário aceder a todos os eixos que estão dispostos, encontrar o eixo com o mesmo tipo e alterar a sua série de dados (acrescentando a nova série), caso não seja encontrado um eixo do mesmo tipo (unidades do parâmetro) é criado um novo eixo e adicionado ao gráfico. Um fator positivo associado a esta funcionalidade consiste no aproveitamento do espaço relativo ao gráfico, pois ao ser adicionado um novo eixo o espaço de visualização fica reduzido e com esta funcionalidade o aproveitamento de um eixo é exequível e resulta numa poupança de espaço. Por outro lado, é de notar um problema relativo às dimensões das variáveis, pois uma pode reduzir a outra porque estão a ocupar o mesmo eixo dos y. Quando é pretendido remover um parâmetro que está disposto no mesmo eixo de outro parâmetro é necessário aceder às séries do eixo, identificar a série que necessita de ser removida e assim, removê-la.

Existe ainda outra funcionalidade que permite relacionar os parâmetros, para isto acontecer, o utilizador terá de fixar um parâmetro ao eixo dos x e fixar outros parâmetros ao eixo dos y. Este relacionamento pode ser representado visualmente, pois se os valores das variáveis possuírem o tempo de medição podem ser relacionados. Assim, o valor do x é a variável que foi fixada no eixo dos x e a o valor do y corresponde à variável que foi fixada no eixo dos y. Foi desenvolvido um método que permite remover as linhas que conectam os dados e assim, é criado o gráfico do tipo *scatterplot*, ou seja, um gráfico de dispersão. Para isto, foi necessário aceder a todas as séries existentes na visualização e retirar a linha da configuração do gráfico. Embora seja removida a linha, a codificação dos pontos continua a ser pela cor e pelo glifo, isto é, os pontos continuam a possuir a mesma cor e o mesmo glifo e assim, torna-se possível a sua diferenciação. O tipo de gráfico foi configurado para que o utilizador possa fazer *zoom* em relação ao x e ao y.

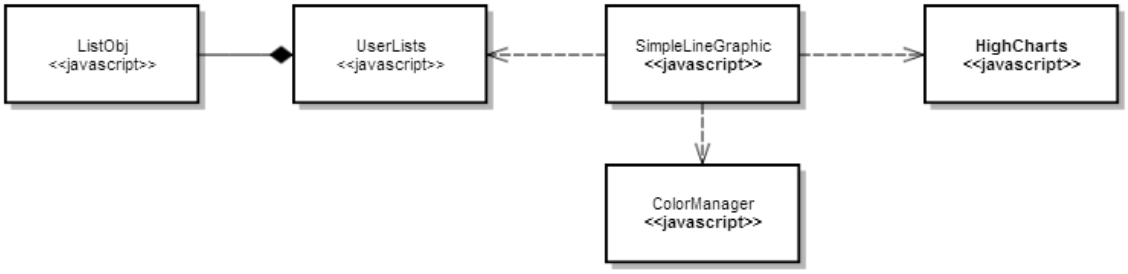


Figura 39 – Diagrama de classes do gráfico de linhas.

É possível visualizar na Figura 39 o diagrama de classes simplificado relativo às principais classes utilizadas para tornar possível a representação do gráfico de linhas.

6.2.7 – Visualização em Séries Temporais

O tipo de visualização denominado *time series* é uma especialização do gráfico de linhas simples pois é específico para representar séries temporais. Nota-se na janela de visualização do gráfico que existem várias componentes dedicadas ao tempo, na parte inferior do gráfico existe uma janela de visualização interativa e permite ao utilizador ajustar a janela temporal à sua maneira, na parte superior direita também existe uma alusão ao tempo, e duas datas, uma inicial e outra final que podem ser alteradas e o gráfico adapta-se a estas datas. Este tipo de visualização foi conseguido através da utilização da classe *TimeSeriesGraphic,* com o auxílio da biblioteca externa *Highcharts* (gráfico do tipo *HighStock*) e com o acesso à classe *UserLists*. Assim, os dados são requeridos ao objeto *UserLists* e é utilizado um método para retirar o valor dos dados e o tempo em que os dados foram medidos. Desta forma, é criada uma série com vários *arrays* que possuem a informação relativa ao valor do tempo em que foi realizada a medição e o valor do parâmetro, esta série é colocada no eixo do gráfico e este é representando graficamente na janela de visualização. Caso o utilizador selecione mais parâmetros para serem apresentados visualmente, são criadas novas séries de dados e adicionadas ao gráfico. Por outro lado, o utilizador pode necessitar de remover um parâmetro da janela de visualização e, assim, a série de dados tem de ser identificada e removida. Neste tipo de visualização também foi implementado a funcionalidade de compactar os eixos. Assim sendo, quando o utilizador ativa esta funcionalidade e adiciona um novo parâmetro à visualização são geradas novas séries com os novos dados e existe a necessidade de utilizar um método que encontra o eixo com os mesmos tipos de dados para que as séries sejam associadas. Caso não exista um eixo com o tipo de dados (unidades) é criado um novo eixo e adicionado à visualização. Quando é necessário remover um parâmetro, que já esteja representado no mesmo eixo de outro parâmetro, é necessário aceder às séries, identificar a série que deve ser removida e removê-la.

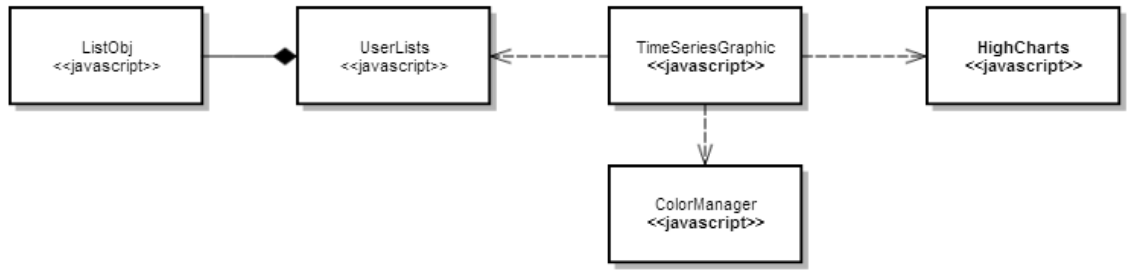


Figura 40 – Diagrama de classes, visualização séries temporais.

No diagrama da Figura 40 estão presentes as relações das principais classes essenciais para a representação gráfica em séries temporais. Desta forma, este apresenta uma visão simplista na medida que não apresenta os atributos nem os métodos das classes representadas.

6.2.8 – Gráfico Coordenadas Paralelas

Existe um tipo de visualização que é pouco comum, trata-se do tipo de representação gráfica denominada de coordenadas paralelas. Foi desenvolvida uma classe *ParallelCoordinates* com o propósito de implementar este tipo de gráfico e recorreu-se à biblioteca externa *d3* e à classe *UserLists* como forma de acesso aos dados dos parâmetros requisitados. Os dados dos parâmetros selecionados pelo utilizador são relacionados utilizando o método semelhante ao do gráfico de dispersão (*scatterplots*) e é criado um *array* que contém vários *arrays* que possuem dados. Seguidamente o gráfico é representado na janela de visualização. Foi adicionado uma funcionalidade denominada de *alpha-blending* que permite diminuir ou aumentar a transparência de cada linha do gráfico, estes valores variam de 0 (transparente) até 1 (sem transparência), o que torna possível verificar zonas de maior densidade de linhas que representam a relação dos dados. Também foi acrescentado uma tabela com todos os dados relacionados onde é possível identificar na visualização um conjunto de dados (uma linha) na representação gráfica.

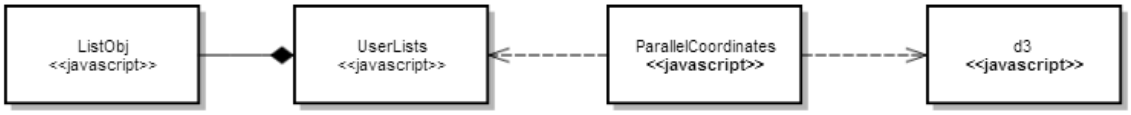


Figura 41 – Diagrama visualização coordenadas paralelas.

Pode ser visto no diagrama da Figura 41 a representação das classes essenciais para a representação deste tipo de gráfico, assim como, os seus relacionamentos.

6.2.9 – Gráfico SPLOM

Foi implementado o tipo visualização de dados referente a matrizes de gráficos de dispersão ou SPLOMS (*scatterplot matrices*). Desta forma, foi desenvolvida uma classe *Splom_* que utiliza a biblioteca externa *d3* e requisita os dados da classe *UserLists*. Quando o utilizador seleciona os parâmetros que pretende visualizar recorrendo, a este tipo de gráfico, os dados são requisitados ao objeto *UserLists* e são formatados através de um método que cria um *array* de objetos *javascript* onde existe uma correspondência entre o nome do parâmetro e o valor deste parâmetro, ou seja, caso existam dois parâmetros a serem relacionados, o objeto vai conter o nome dos dois parâmetros e os seus valores que estão relacionados. Neste tipo de gráfico os parâmetros são apresentados na diagonal descendente e todos os parâmetros são relacionados uns com os outros e, assim, são formadas todos os gráficos de dispersão que estão presentes na matriz de gráficos de dispersão.

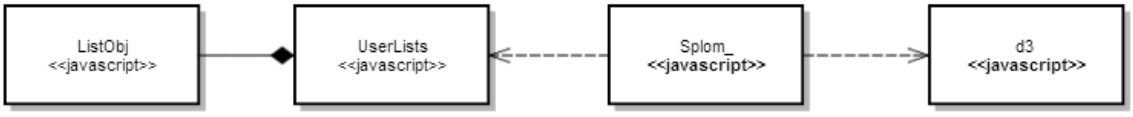


Figura 42 - Diagrama de classes do gráfico splom.

É possível visualizar na Figura 42 o diagrama de classes deste tipo de gráfico. Desta modo, são representadas de forma simples as principais classes envolvidas, assim como, os seus relacionamentos.

6.2.10 – Representação em Mapa

Foi implementada uma classe em *javascript* denominada de "*SatellitePosition*" que possui o propósito de fornecer um tipo de visualização que utilizasse um mapa e onde fosse possível ver a órbita do satélite. Desta forma, foi utilizada a API do *Google Maps* para tornar exequível a construção de um mapa-mundo interativo. Para representar o satélite foi usada uma imagem alusiva a um satélite e o ícone do gráfico foi substituto por esta. Para receber os dados das coordenadas foi necessário utilizar um método que comunicasse com a aplicação *web* em execução do lado do servidor. Assim, os dados são processados no lado do servidor e é criado um objeto do tipo JSON que é enviado ao método que fez o pedido do lado do cliente, esta comunicação é realizada utilizando *ajax*.

Figura 43 – Diagrama classe da representação em mapa.

No diagrama da Figura 43 verifica-se que, através de um diagrama de classes simples, as relações entre as principais classes que são utilizadas para recorrer a este tipo de visualização.

6.2.11 – Informação de Missões e Instrumentos

Com o propósito de fornecer informações relativas às missões e aos seus instrumentos foi desenvolvida a classe "*ExploreMissions*". Esta recorre à aplicação *web* em execução no lado do servidor com o intuito de receber todas as informações relativas às missões e seus instrumentos armazenadas na base de dados. Para conseguir obter as informações a aplicação *web* utiliza a API do "*Web-Service*". Ao receber todas as informações existentes estas são processadas e formatadas para um objeto do tipo JSON. Este objeto é enviado para a aplicação que está a executar no *browser* do cliente e assim, é possível mostrar todas as informações relativas às missões e seus instrumentos de uma forma interativa onde o cliente utiliza um menu que contém todas as missões e outro onde existem os instrumentos destas missões.

6.3 – Desenvolvimento do "*Database Updater*"

É através deste sistema que se torna exequível adicionar os dados das missões à base de dados *CouchDb*. Estes dados são relativos às medições realizadas pelos instrumentos ao longo do tempo e estão disponíveis devido às políticas da ESA que consistem em fornecer acesso total aos dados de várias missões. Desta forma, estes dados estão acessíveis através de um registo simples no *site* da ESA e organizados em vários ficheiros, como por exemplo, os ficheiros .N1 que contêm os dados da missão *Envisat*. Para fazer *download* dos ficheiros é necessário recorrer ao servidor FTP da ESA onde se encontram estes ficheiros de dados. É através deste sistema que a base de dados é atualizada e possibilita que o "*Web-Service*"

tenha acesso aos dados e consiga responder aos pedidos recebidos. Para realizar esta tarefa, os dados têm de ser processados e organizados em objetos com o formato JSON, posteriormente são introduzidos na base de dados.

6.3.1 – Diagrama de packages

Neste sistema foram desenvolvidos seis *packages* como forma de organizar classes relacionadas e é possível obter uma ideia de alto nível do *software* implementado.

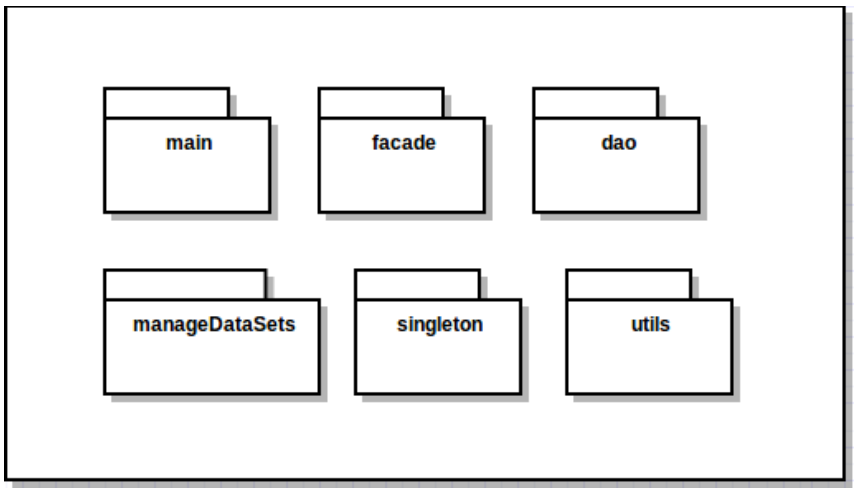


Figura 44 – Diagrama de *packages* relativas ao "Database Updater".

De acordo com a Figura 44 verifica-se a existências de várias *packages* e cada uma possui uma função no sistema. Desta forma, segue-se a explicação de cada uma destas:

- *main*: Esta *package* possui a classe *main* e é o ponto de entrada do código *Java*;
- *facade*: É caracterizada por ser uma API simplificada onde são chamados vários métodos;
- *dao*: Trata-se da *package* onde existem as classes que possibilitam o acesso aos dados e informações da base de dados;
- *manageDataSets*: *Package* que possui as classes que guardam a informação das missões e *datasets*;
- *singleton* : Trata-te da *package* onde existe a classe *singleton* que permite realizar a comunicação com a base de dados.
- *utils* : Esta *package* contém as várias classes que permitem a manipulação e criação de objetos JSON com o propósito de atualizar a base de dados.

6.3.2 – *Design Patterns* Implementados

Como já foi referido na arquitetura do projeto neste sistema foram implementados alguns tipos de padrões de desenvolvimento de *software*. Foi utilizada esta abordagem porque possibilita a reutilização do *software* produzido quando for necessário acrescentar novas funcionalidades ou até alterar as que já existem.

Pelo diagrama de packages (Figura 44) torna-se possível identificar os padrões de *software* desenvolvidos através do seu nome. Desta forma, neste sistema foram implementados os *design patterns* denominados de DAO (*Data Access Object*), *Facade* e *Singleton*. A informação sobre estes pode ser consultada na secção "6.1.3 – *Design Patterns* Implementados" relativa ao desenvolvimento do "*Web-Service*".

6.3.3 – Características da Implementação

Para ser exequível atualizar a base de dados *CouchDb* com os dados provenientes dos ficheiros que contêm as leituras é necessário realizar várias manipulações nestes dados antes de proceder à sua inserção. Desta forma, os dados têm de ser formatados para um objeto do tipo JSON que posteriormente é inserido na base de dados.


```
{
   "_id": "cbf2435a90f3f31c489b006470000a98",
   "_rev": "7-d86a9746446bb95b07b4e5b920248212",
   "type": "data",
   "mission": "Envisat",
   "instrument": "Doris",
   "DORIS_PRECISE_ORBITS": {
       "1": [
           {
               "index": "1",
               "fieldname": "utc_time",
               "value": "1401455126813",
               "units": "UTC"
           },
           {
               "index": "7",
               "fieldname": "x_pos",
               "value": "1663441.593",
               "units": "m"
           },
           {
               "index": "9",
               "fieldname": "y_pos",
               "value": "1071773.273",
               "units": "m"
           },
           {
               "index": "11",
               "fieldname": "z_pos",
               "value": "-16892085.607",
               "units": "m"
           }
       ],
```


Figura 45 – Estrutura de um documento de dados.

De acordo com a Figura 45 podemos visualizar a estrutura de um documento do tipo JSON que representa alguns dados da missão *Envisat*, instrumento *Doris* e domínio "DORIS_PRECISE_ORBITS". Assim sendo, este sistema necessita de organizar os dados nesta estrutura JSON.

Para aceder aos ficheiros de dados existem diretorias definidas de acordo com as missões e instrumentos. As leituras dos ficheiros são realizadas de forma recursiva que possibilita a existência de ficheiros em várias pastas ou subpastas. Desta forma, os ficheiros são localizados e enviados para o processamento. Como resultado deste processamento surge um objeto JSON com todas as informações necessárias para a introdução deste objeto na base de dados, assim como, os valores dos dados codificados de forma correta (exemplo da Figura 45, os dados têm de estar codificados como objeto e inseridos num *array*). Depois de possuir este objeto, em primeiro lugar, é necessário identificar a missão, o instrumento, os *datasets* (domínios) e os vários parâmetros. Desta forma, o objeto JSON é introduzido na base de dados e o seu *id* é referenciado no ficheiro do tipo missão a que este pertence.

6.3.4 – Bibliotecas Externas

Foi desenvolvida pela *VisionSpace Technologies* uma biblioteca que utiliza o software CODA [51], este faz parte de um projeto financiado pela ESA, é gratuito e possui uma API para a

linguagem *Java*. Este software possibilita realizar o *parse* dos ficheiros .N1 relativos à missão *Envisat*. Para a utilização desta ferramenta é necessário proceder à sua instalação. Desta forma, este programa vai efetuar o *parse* dos ficheiros indicados pelo sistema "*Database Updater*", os resultados são consumidos pela biblioteca mdv-n1-parse que devolve um objeto JSON com os dados das leituras do ficheiro .N1.

Para a comunicação e transferência de documentos com a base de dados foi utilizada a biblioteca *Ektorp*. Esta fornece vários métodos para efetuar a comunicação com o *CouchDb* e possibilita que os documentos sejam descarregados, apagados ou então modificados.

6.3.5 – Diagrama de Classes

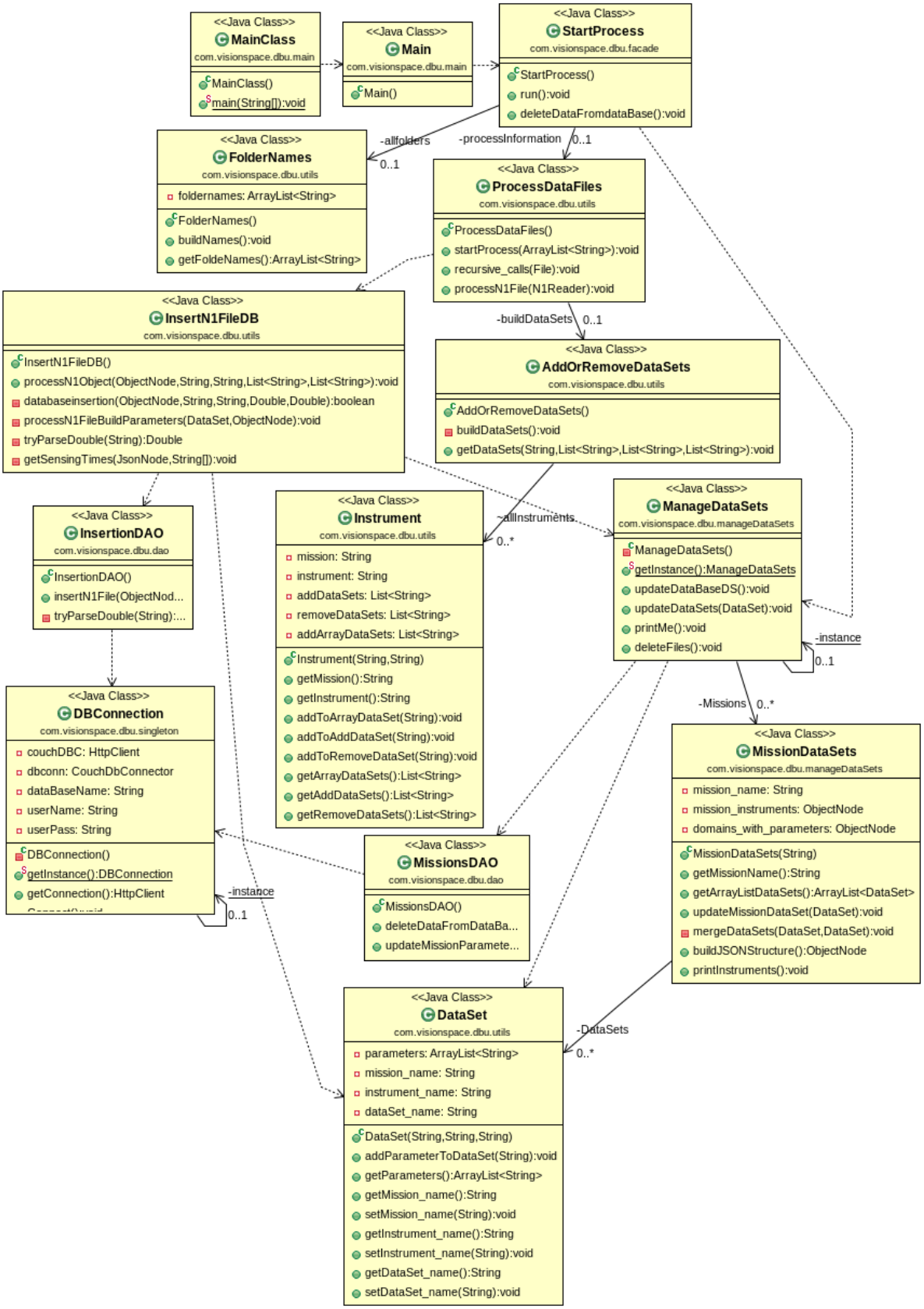


Figura 46 – Diagrama de classes "Database Updater".

Capítulo 7 – Metodologia e Planeamento

Neste capitulo é apresentada a metodologia seguida no desenvolvimento do projeto, assim como o planeamento inicial e final do estágio desenvolvido.

7.1 – Metodologia

O desenvolvimento do projeto seguiu uma abordagem que incorporou características do modelo clássico *waterfall* [52]. De acordo com este modelo sequencial de desenvolvimento de *software* existem as seguintes fases: especificação dos requisitos, *design*, codificação, integração, testes, instalação e manutenção. Neste projeto, em primeiro lugar, foi realizada uma análise relativa aos sistemas a implementar resultando num documento com a especificação dos requisitos das aplicações requeridas pela empresa. Com este documento foi possível proceder ao *design* dos sistemas gerando uma arquitetura do projeto. Seguiu-se a fase de implementação do *software*, instalação das aplicações e, por fim, a realização de testes funcionais.

Durante o estágio foram realizadas várias reuniões semanais com o orientador da empresa de modo a verificar o trabalho desenvolvido, discussão sobre o trabalho a realizar e o trabalho que ficou por realizar. Desta forma, existiu uma elevada interação entre o programador e o cliente. Relativamente à gestão do projeto foi utilizada a ferramenta *web* denominada *redmine*.

7.2 - Planeamento Inicial

	Nome	Duração	Início	Fim
1	Enquadramento	1d	30/09/2013	30/09/2013
2	Planeamento estudo	1d	01/10/2013	01/10/2013
3	**Estudo e pesquisa do tema**	**24d**	**02/10/2013**	**04/11/2013**
4	Missões da ESA	2d	02/10/2013	03/10/2013
5	Envisat	9d	04/10/2013	16/10/2013
6	Cryosat	7d	17/10/2013	25/10/2013
7	SMOS	6d	28/10/2013	04/11/2013
8	Estado da arte	14d	05/11/2013	22/11/2013
9	Levantamento de requisitos	5d	25/11/2013	29/11/2013
10	**Arquitetura**	**15d**	**02/12/2013**	**20/12/2013**
11	Estudo de ferramentas	4d	02/12/2013	05/12/2013
12	Estudo base de dados	3d	06/12/2013	10/12/2013
13	"Web-Server" e "Web-Site"	3d	11/12/2013	13/12/2013
14	"Web-Service"	3d	16/12/2013	18/12/2013
15	"Parser de Informação"	2d	19/12/2013	20/12/2013
16	Elaboração do relatório intermédio	18d	02/01/2014	27/01/2014
17	Preparação defesa intermédia	4d	28/01/2014	31/01/2014
18	**Desenvolvimento do "Web-Service"**	**35d**	**10/02/2014**	**28/03/2014**
19	Criação base de dados	5d	10/02/2014	14/02/2014
20	Desenvolvimento	30d	17/02/2014	28/03/2014
21	Desenvolvimento do "Parser de Informação"	10d	31/03/2014	11/04/2014
22	Desenvolvimento do "Web-Server" e "Web-Site"	32d	14/04/2014	27/05/2014
23	Testes e resolução de bugs	10d	28/05/2014	10/06/2014
24	Elaboração relatório final	15d	11/06/2014	01/07/2014
25	Preparação defesa final	4d	02/07/2014	07/07/2014

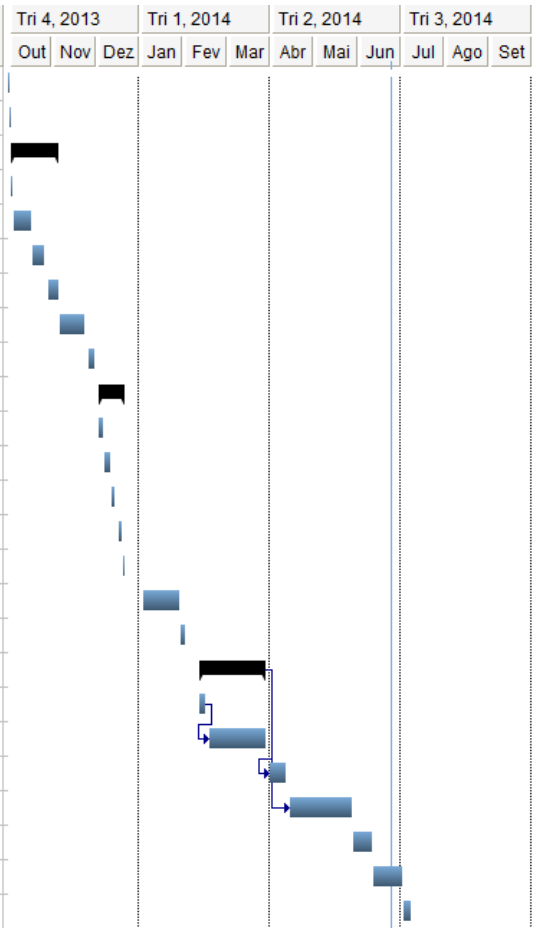


Figura 47 – Planeamento inicial do projeto.

Na Figura 47 é apresentado o planeamento inicial do projeto. As atividades relativas ao primeiro semestre eram de carácter preparatório consistindo no estudo de várias missões, no levantamento dos requisitos e no desenvolvimento das arquiteturas do projeto (Estudo e pesquisa do tema, levantamento de requisitos e arquitetura). Relativamente ao segundo semestre, o trabalho focava-se no desenvolvimento dos sistemas presentes nos requisitos (Desenvolvimento do "*Web-Service*", Parser de Informação e Desenvolvimento do "Web-Server" e "Web-Site").

7.3 - Planeamento Final

	Nome	Duração	Início	Fim
1	Enquadramento	1d	30/09/2013	30/09/2013
2	Planeamento estudo	1d	01/10/2013	01/10/2013
3	**Estudo e pesquisa do tema**	**24d**	**02/10/2013**	**04/11/2013**
4	Missões da ESA	2d	02/10/2013	03/10/2013
5	Envisat	9d	04/10/2013	16/10/2013
6	Cryosat	7d	17/10/2013	25/10/2013
7	SMOS	6d	28/10/2013	04/11/2013
8	Estado da arte	14d	05/11/2013	22/11/2013
9	Levantamento de requisitos	5d	25/11/2013	29/11/2013
10	**Arquitetura**	**15d**	**02/12/2013**	**20/12/2013**
11	Estudo de ferramentas	4d	02/12/2013	05/12/2013
12	Estudo base de dados	3d	06/12/2013	10/12/2013
13	Aplicação Web	3d	11/12/2013	13/12/2013
14	"Web-Service"	3d	16/12/2013	18/12/2013
15	Database Updater	2d	19/12/2013	20/12/2013
16	Elaboração do relatório intermédio	18d	02/01/2014	27/01/2014
17	Preparação defesa intermédia	4d	28/01/2014	31/01/2014
18	Estudo Visualização de Dados	40d	17/02/2014	11/04/2014
19	**Desenvolvimento do Web Service**	**36d**	**10/02/2014**	**31/03/2014**
20	**Base de dados**	**5d**	**10/02/2014**	**14/02/2014**
21	Instalação e leitura de documentação	1d	10/02/2014	10/02/2014
22	Criação de Views	2d	10/02/2014	11/02/2014
23	Integração com Java	3d	12/02/2014	14/02/2014
24	**Implementação do Web Service**	**27d**	**17/02/2014**	**25/03/2014**
25	Desenvolvimento da API	5d	17/02/2014	21/02/2014
26	Desenvolvimento das DAO e Singleton	13d	24/02/2014	12/03/2014
27	Desenvolvimento das camadas intermédias (Facade)	9d	13/03/2014	25/03/2014
28	Implantação do Web Service	3d	26/03/2014	28/03/2014
29	JavaDoc	1d	31/03/2014	31/03/2014
30	**Desenvolvimento Aplicação Web MDV**	**42d**	**01/04/2014**	**28/05/2014**
31	Instalação e estudo da ferramenta Ruby on Rails	5d	01/04/2014	07/04/2014
32	Criação do design (Mockups) e implementação	10d	07/04/2014	18/04/2014
33	Comunicação e Gestão de Dados	5d	21/04/2014	25/04/2014
34	Desenvolvimento Visualização de Dados	23d	28/04/2014	28/05/2014
35	Desenvolvimento do Database Updater	7d	29/05/2014	06/06/2014
36	Testes e resolução de problemas	5d	09/06/2014	13/06/2014
37	Desenvolvimento do Relatório Final	16d	09/06/2014	30/06/2014
38	Elaboração da Defesa Final	4d	02/07/2014	07/07/2014

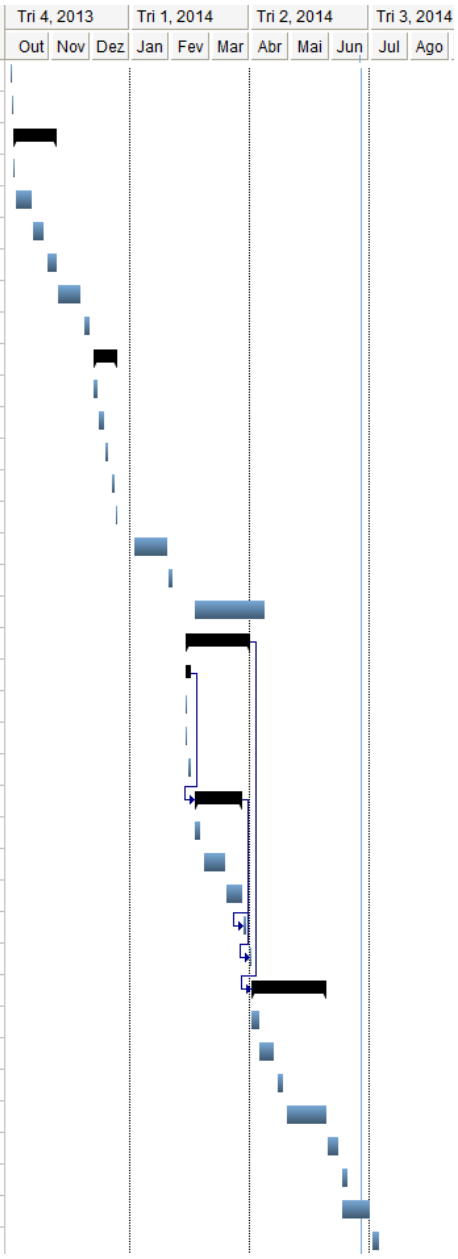


Figura 48 – Planeamento Final.

O planeamento apresentado na Figura 48 contém as novas atividades inseridas com o propósito de colmatar falhas existentes no projeto. Desta forma, o planeamento inicial foi ligeiramente alterado, tendo sido introduzida uma atividade chamada "Estudo da Visualização de Dados" que decorreu em simultâneo com outras atividades de desenvolvimento de *software*. Esta veio introduzir um estudo teórico sobre visualização de dados em geral. Ocorreu uma mudança dos nomes de algumas atividades. A atividade "Desenvolvimento do Parser de Informação" passou a ser designada por "Desenvolvimento do Database Updater" e a atividade "Desenvolvidmento do "Web-Server e "Web-Site"" passou a ser denominada de "Desenvolvimento aplicação Web MDV". Também a ordem foi alterada. Assim, no decorrer do segundo semestre foram desenvolvidas atividades teóricas e de desenvolvimento de *software*.

Capítulo 8 – Testes

Neste capítulo vão ser apresentados os testes realizados às aplicações desenvolvidas com o propósito de encontrar *bugs* e validar as mesmas. Este processo decorreu de acordo com o planeamento do projeto.

8.1 – Testes de aceitação

Foram realizados testes de aceitação (*User Acceptance Testing* – UAT) não automatizados com o propósito de validar se todos os requisitos funcionais foram implementados e se o seu funcionamento está de acordo com o esperado pela empresa. Desta forma, os requisitos funcionais foram mapeados para casos de teste.

8.1.1 – Planeamento dos testes

Para efetuar os testes de aceitação foram desenvolvidos vários casos de testes recorrendo a folhas de cálculo *Excel* (podem ser consultadas no Apêndice E). Estes casos de teste foram separados de acordo com os sistemas e requisitos desenvolvidos. Cada caso de teste possui um código e, assim, é possível fazer a associação com os requisitos. Para cada caso de teste existe uma descrição relativa a cada passo e o seu resultado esperado. Assim, quem estiver a realizar os testes tem a informação sobre o resultado que é esperado em cada teste e consegue responder se o teste passou ou não.

Código do Requisito	Código do Caso de Teste
RF-WS-01	WS-TC-02
RF-WS-02	WS-TC-03
RF-WS-03	WS-TC-04
RF-WS-04	WS-TC-05
RF-WS-05	WS-TC-05
RF-WS-06	WS-TC-05
RF-WS-07	WS-TC-05
RF-WS-08	WS-TC-06
RF-WS-09	WS-TC-07
RF-WS-10	WS-TC-08
RF-WS-14	WS-TC-10
RF-WS-15	WS-TC-09

Tabela 10 – Códigos de Casos de Teste: "*Web-Service*".

Código do Requisito	Código do Caso de Teste
RF-MDV-01	WA-TC-01
RF-MDV-02	WA-TC-02
RF-MDV-03	WA-TC-02
RF-MDV-04	WA-TC-07
RF-MDV-05	WA-TC-09
RF-MDV-06	WA-TC-03
RF-MDV-06	WA-TC-04
RF-MDV-06	WA-TC-05
RF-MDV-06	WA-TC-06
RF-MDV-06	WA-TC-08
RF-MDV-07	WA-TC-10
RF-MDV-08	WA-TC-10

Tabela 11 - Códigos de Casos de Teste: aplicação *web*.

Na Tabela 10 e 11 são identificadas as relações entre os requisitos funcionais e os casos de teste. Na Tabela 10 estão presentes os códigos dos casos de teste necessários para testar o "*Web-Service*" utilizando este método. Por outro lado, na Tabela 11 estão os códigos dos casos de teste necessários para testar as funcionalidades da aplicação *web* MDV. Existem duas folhas de cálculo do *Excel* que cotêm os casos de teste relativos aos dois sistemas acima referidos. Estas são entregues aos responsáveis pelos testes e estes têm de executá-los de acordo com a ordem que aparecem na folha de cálculo. Desta forma, os testes são realizados e os possíveis problemas que ocorram podem ser descritos na própria folha de cálculo e enviada ao responsável pelo processo de testes.

Os testes foram realizados por dois colaborados e pelo orientador da empresa *VisionSpace Technologies*. Estes testes tiveram lugar na empresa e foi utilizada uma máquina interna onde estão implementados os sistemas desenvolvidos. Para além dos testes, todos os sistemas desenvolvidos no âmbito do projeto foram apresentados ao orientador da empresa e aos responsáveis desta em várias reuniões de demonstração.

8.1.1 – Resultados

Todos os casos de teste realizados pelos vários colaboradores foram aprovados e, assim, podemos considerar que os requisitos funcionais foram implementados com sucesso.

Capítulo 9 – Conclusão e Trabalho Futuro

A realização deste relatório final de estágio, como uma das componentes do estágio referente ao Mestrado em Engenharia Informática, veio aprofundar o meu conhecimento sobre a área do espaço, mais precisamente, sobre a *ESA* e alguns dos seus satélites e instrumentos. Com este estudo foi possível perceber o impacto positivo que esta área oferece a todos os habitantes do planeta terra. Desta forma, impulsiona avanços tecnológicos, melhora as comunicações e cria um vasto leque de dados e informações que são utilizados por cientistas. Através do estudo de fenómenos como o clima, correntes marítimas e o ciclo da água, possibilita a elaboração de relatórios que podem prever futuras alterações do clima terrestre e, possivelmente, salvar vidas humanas.

Ao longo do primeiro semestre, dedicado ao aprofundamento do meu conhecimento sobre esta área, deparei-me com vários problemas derivados à dificuldade de encontrar informações sobre os vários satélites, instrumentos e os seus dados. Foi um grande desafio entender como obter os ficheiros que contêm os dados e as ferramentas utilizadas para a visualização dos mesmos. Relativamente a estes pacotes de *software* foi complicado compreender o seu funcionamento, na medida em que são orientados para o uso científico.

O principal objetivo deste estágio consistiu no desenvolvimento de uma plataforma *online* de visualização de dados. Para isto, foi realizado um estudo aprofundado sobre várias técnicas de visualização de dados com o propósito de serem utilizadas no projeto. Também foi necessário implementar um serviço *web* com uma API pública com o intuito de fornecer dados e informações relativas às missões existentes na base de dados. Através da realização de pedidos à API do serviço *web,* torna-se possível à aplicação *web* MDV receber e criar vários tipos de visualização de dados.

Com a implementação destes sistemas distribuídos consegui aplicar muito do conhecimento obtido durante o curso de Engenharia Informática. A utilização de novas ferramentas, como por exemplo, a *framework Ruby on Rails* ou base de dados orientada a documentos *CouchDb,* assim como, todas as novas técnicas de representação de dados em contexto de multivariável (coordenadas paralelas) contribuíram para um aumento das minhas competências no que se refere a novas tecnologias de informação e a novas formas de visualização de dados.

Como trabalho futuro do projeto é necessário incluir dados de outras missões, como por exemplo, os dados da missão *CryoSat*, SMOS, entre outras. Relativamente à aplicação *web* MDV pode ser melhorada a interação com o utilizador, como por exemplo, a melhoria da funcionalidade denominada de *Compact*.

Para finalizar, a realização deste estágio permitiu uma aproximação à realidade do que é o mercado de trabalho numa empresa com dimensão internacional, tendo proporcionado uma experiência inovadora e enriquecedora tanto a nível pessoal quanto a nível profissional.

Capítulo 10 - Referências

[1] J. Krige e A. Russo, A History of the European Space Agency, AG Noordwijk: ESA Publications Division, 2000.

[2] ESA, [Online]. Available: http://www.esa.int/Our_Activities/Operations/Kourou_station. [Acedido em 15 Janeiro 2014].

[3] "wikipedia," [Online]. Available: http://en.wikipedia.org/wiki/European_Space_Agency. [Acedido em 15 Janeiro 2014].

[4] "ESA Member States," [Online]. Available: http://www.esa.int/About_Us/Welcome_to_ESA/New_Member_States. [Acedido em 15 Janeiro 2014].

[5] "ESA Budget 2013," [Online]. Available: http://www.esa.int/About_Us/Welcome_to_ESA/Budget_as_presented_during_DG_press_conference_24_January_2013.

[6] ESA, CONVENTION for the establishment of a European Space Agency & ESA Council RULES OF PROCEDURE, AG Noordwijk: ESA Publications Division, 2003.

[7] ESA, *ESA Data Policy for ERS, Envisat and Earth Explorer missions,* 2012.

[8] "ESA ENVISAT," [Online]. Available: https://earth.esa.int/web/guest/missions/esa-operational-eo-missions/envisat. [Acedido em 15 Janeiro 2014].

[9] "ENVISAT END MISSION," [Online]. Available: https://earth.esa.int/web/guest/news/-/asset_publisher/G2mU/content/good-bye-envisat-and-thank-you. [Acedido em 15 Janeiro 2014].

[10] ESA, MIPAS Products User Guide.

[11] ESA, ASAR Product Handbook.

[12] "Imagem "Prestige"," [Online]. Available: https://earth.esa.int/services/sample_products/. [Acedido em 15 Janeiro 2014].

[13] ESA, MERIS User Guide.

[14] ESA, AATSR HANDBOOK.

[15] ESA, ENVISAT RA-2 AND MWR HANDBOOK.

[16] ESA, GOMOS Handbook.

[17] ESA, MIPAS HANDBOOK.

[18] ESA, SCIAMACHY HANDBOOK.

[19] "ESA DORIS," [Online]. Available: https://earth.esa.int/web/guest/missions/esa-operational-eo-missions/envisat/instruments/doris. [Acedido em 16 Janeiro 2014].

[20] "ESA LRR," [Online]. Available: https://earth.esa.int/web/guest/missions/esa-operational-eo-missions/envisat/instruments/lrr. [Acedido em 16 Janeiro 2014].

[21] ESA, CryoSat Mission and Data Description, 2007.

[22] ESA, Mission Objectives and Scientific Requirements of the Soil Moisture and Ocean Salinity (SOMOS) Mission, 2002.

[23] "SMOS INFO," [Online]. Available: https://earth.esa.int/web/guest/missions/esa-operational-eo-missions/smos/space-segment/satellite. [Acedido em 16 Janeiro 2014].

[24] "ESA SMOS MIRAS," [Online]. Available: https://earth.esa.int/web/guest/missions/esa-operational-eo-missions/smos/space-segment/instrument. [Acedido em 16 Janeiro 2014].

[25] E. COLET e D. AARONSON, "Visualization of multivariate data: Human-factors considerations," 1995.

[26] G. Michailidis, "Data Visualization Through Their Graph Representations," em *Handbook of Data Visualization*, Verlag Berlin Heidelberg, Springer, 2008, pp. 104 - 118.

[27] M. Friendly, "A Brief History of Data Visualization," em *Handbook of Data Visualization*, Verlag Berlin Heidelberg, Springer, 2008, pp. 16 - 48.

[28] N. Iliinsky e J. Steele, Designing Data Visualizations, 1005 Gravenstein Highway North, Sebastopol, CA 95472.: O'Reilly Media, Inc., 2011.

[29] A. Unwin, "Good Graphics?," em *Handbook of Data Visualization*, Verlag Berlin Heidelberg, Springer, 2008, pp. 58 - 77.

[30] S. J. Sándor Kromesch, *High Dimensional Data Visualization.*

[31] M. O.Ward, "Multivariate Data Glyphs: Principles and Practice," em *Handbook of Data Visualization*, Verlag Berlin Heidelberg, Springer, 2008, pp. 180 - 195 .

[32] L. Wilkinson, "Graph-theoretic Graphics," em *Handbook of Data Visualization*, Verlag Berlin Heidelberg, Springer, 2008, pp. 122 - 148.

[33] "http://www.highcharts.com/," [Online].

[34] L. V. f. V. Exploration, "Adalbert Wilhelm," em *Handbook of Data Visualization*, Verlag Berlin Heidelberg, Springer, 2008, pp. 200 - 214.

[35] M. Theus, "High-dimensional Data Visualization," em *Handbook of Data Visualization*, Verlag Berlin Heidelberg, Springer, 2008, pp. 152 - 175.

[36] E. Gamma, R. Helm, R. Johnson e J. Vlissides , Design Patterns - Elements of Reusable Object-Oriented Software, Pearson Education, 1994.

[37] "http://tomcat.apache.org/index.html," [Online].

[38] "http://couchdb.apache.org/," [Online].

[39] "https://jersey.java.net/," [Online].

[40] [Online]. Available: http://docs.couchdb.org/en/latest/intro/overview.html. [Acedido em 16 Janeiro 2014].

[41] M. Dan , A. Deepak e J. Crupi, Core J2EE™ Patterns: Best Practices and Design Strategies, Pearson Education, 2001.

[42] "http://rubyonrails.org/," [Online].

[43] "http://jquery.com/," [Online].

[44] "http://arborjs.org/," [Online].

[45] "http://d3js.org/," [Online].

[46] "http://xdsoft.net/jqplugins/datetimepicker/," [Online].

[47] "http://medialize.github.io/jQuery-contextMenu/," [Online].

[48] "http://notifyjs.com/," [Online].

[49] "http://www.sqlite.org/," [Online].

[50] M. Hartl, RUBY ON RAILS TUTORIAL, Addison-Wesley Professional, 2012.

[51] "www.stcorp.nl/coda/," [Online].

[52] W. W. Rovce, *Managing the development of large software systems.*

Apêndice A

Requisitos funcionais "*Web-Service*"

Os requisitos funcionais relativos ao "*Web-Service*" estão descritos nas seguintes tabelas:

Código	**RF-WS-01**
Descrição	O "*Web-Service*" tem de possuir um comando de ajuda. Desta forma, torna-se possível ao utilizador saber como é que este sistema funciona.
Informações	O utilizador deve receber em HTML toda a informação de como utilizar este "*Web-Service*" de uma forma correta.
Regras	Através de um pedido ao servidor, por exemplo, chamada do método *help* da API o utilizador deve receber o HTML com a informação de como utilizar o serviço.

Código	**RF-WS-02**
Descrição	O "*Web-Service*" tem de redirecionar o pedido do cliente para a página de ajuda caso o método que é chamado não exista.
Informações	Quando um cliente faz um pedido errado através de um método GET, o "*Web-Service*" deve redirecionar este cliente para a página de ajuda existente. Desta forma, o cliente pode visualizar os métodos existentes e criar um novo pedido.
Regras	Quando o "*Web-Service*" recebe um método errado proveniente de um GET do cliente tem de redirecionar este pedido para a página de ajuda existente.

Código	**RF-WS-03**
Descrição	Este sistema deve possuir uma função para enviar todos os métodos suportados, ou seja, todos os métodos que o cliente pode utilizar para interagir com o "*Web-Service*".
Informações	Esta funcionalidade é útil quando o utilizador não quer ler a página de ajuda, necessitando apenas da informação sobre todos os métodos suportados pelo sistema. Lembrando que este "*Web-Service*" pode ser utilizado por outro sistema informático, desta forma, este não necessita de uma página de ajuda, mas sim de uma lista de comandos compatíveis.
Regras	O utilizador tem de enviar o comando *getMethods* para obter a resposta de todos os métodos por parte do "*Web-Service*". Esta resposta deve estar formatada em JSON.

Código	**RF-WS-04**
Descrição	O sistema tem de enviar para o utilizador toda a informação sobre as missões existentes no "*Web-Service*".
Informações	O utilizador pode estar interessado em saber detalhes de missões (p.e data de início, data de fim ou o propósito da missão) existentes no "*Web-Service"*. Deste modo, é necessário a existência de um método que devolva essa informação.
Regras	Para receber esta informação, o utilizador necessita de enviar o comando *getMissionInformation?mission=Mission* para o "*Web-Service*", onde *Mission* é a missão que pretende informação. Esta informação deve ser enviada no formato JSON.

Código	**RF-WS-05**
Descrição	O "*Web-Service*" tem de possuir a funcionalidade de enviar uma lista com todas as missões existentes no sistema.
Informações	O utilizador pode necessitar de receber todas as missões existentes na base de dados, é com esta informação que o cliente fica saber quais as missões existentes e com quais pode trabalhar.
Regras	É necessário criar um método para devolver todas as missões, este pode ser definido como *getMissions*, que devolve a informação em formato JSON.

Código	**RF-WS-06**
Descrição	É necessário existir uma funcionalidade que envia para o utilizador os nomes de todos os instrumentos de uma dada missão.
Informações	Como todas as missões possuem vários tipos de instrumentos, tem de haver a possibilidade de enviar para o utilizador a informação de todos os instrumentos existentes na base de dados referentes à missão requisitada.
Regras	Para que seja possível enviar essa informação ao utilizador, é necessário a existência de um método *getInstruments?mission=Mission*. Deste modo, o sistema vai enviar uma resposta em JSON com os nomes de todos os instrumentos da missão (*Mission*).

Código	**RF-WS-07**
Descrição	O sistema tem enviar informações mais específicas sobre cada instrumento, isto é, as características de cada instrumento referente a uma missão.
Informações	Esta funcionalidade tem como objetivo enviar as informações de cada instrumento que existe associado a uma missão, pois cada missão possui vários instrumentos e cada instrumento tem as suas próprias funcionalidades.
Regras	O sistema tem de integrar a funcionalidade de responder ao pedido do utilizador através do método *getInstrumentInformation? instrument=Instrument&mission=Mission*, assim sendo, torna-se possível enviar ao utilizador em formato JSON a informação detalhada de cada instrumento (*Instrument*) que está incluído numa missão (*Mission*).

Código	**RF-WS-08**
Descrição	O sistema tem suportar a funcionalidade de enviar ao utilizador todos os parâmetros das leituras de um instrumento existentes no "*Web-Service*".
Informações	Esta funcionalidade tem a característica de enviar para o utilizador todos os parâmetros, isto é, o nome dos parâmetros existentes no sistema de um instrumento que está presente numa missão. De notar que existem vários parâmetros nas leituras do instrumento.
Regras	Para utilizar esta funcionalidade o utilizador tem de enviar um comando para o sistema, *getParameters?instument=Insturment&mission=Mission*. Desta forma, o sistema vai devolver em formato JSON todos os parâmetros existentes do instrumento (*Instrument*) que faz parte da missão (*Mission*).

Código	**RF-WS-09**
Descrição	O sistema tem de enviar para o utilizador informação sobre o intervalo de tempo das medições de um instrumento.
Informações	O utilizador necessita de saber o intervalo de tempo das medições de um determinado instrumento, para que, posteriormente seja possível realizar uma procura enquadrada numa janela temporal.
Regras	Para obter o intervalo de tempo das medições realizadas pelo instrumento, é necessário realizar um comando, "«*getTimeRange?instrument=Instrument&Mission=mission*. Assim, o "*Web-Service*" vai responder ao utilizador com a janela temporal existente das medições do instrumento (*Instrument*) inserido na missão (*Mission*) em formato JSON. É necessário introduzir a missão porque pode haver o mesmo instrumento em várias missões.

Código	**RF-WS-10**
Descrição	Tem de ser possível enviar para o utilizador o valor de um parâmetro.
Informações	Esta funcionalidade tem o objetivo de enviar os dados do valor da leitura que o instrumento realizou relativo a um único parâmetro que é selecionado pelo utilizador. Este pedido está inserido numa janela temporal, que corresponde a um tempo inicial e a um tempo final.
Regras	Para que seja enviado o valor de um parâmetro, é necessário que o utilizador realize um pedido ao servidor com o seguinte comando: *getValuesParameter?mission=Missiont&instrument=Instrument&domain=Domain&parameter=Parameter&initd=Inttime&endd=Endtime*. O parâmetro que o utilizador deseja tem de estar presente no campo *Parameter*, o instrumento que realizou a medição tem de estar no campo *Instrument*, o domínio que está inserido o parâmetro tem de estar mencionado no *Domain*, a missão tem de estar no campo *Mission*. Quanto ao tempo inicial da leitura tem ser especificado no campo *Inittime* e o fim da leitura no campo *Endtime*. Com esta funcionalidade torna-se possível responder aos pedidos dos utilizadores em formato JSON, com o valor do parâmetro medido pelo instrumento de uma missão numa determinada janela temporal.

Código	**RF-WS-11**
Descrição	Tem de existir uma validação dos parâmetros relativos ao método que é chamado pelo utilizador.
Informações	Existindo vários métodos que requerem vários parâmetros, tem de existir algum tipo de validação quando um método é chamado e os parâmetros introduzidos pelo utilizador ou sistema informático estão errados.

	Com esta funcionalidade o utilizador recebe uma mensagem com o tipo de erro que ocorreu no seu pedido.
Regras	Um método possui erros no parâmetro quando, por exemplo o nome do parâmetro está mal escrito ("getInstruments?missio=") ou quando falta o valor do paramento ("getInstruments?mission="). Desta forma, o "*Web-Service*" deve enviar ao utilizador uma mensagem no formato JSON com o tipo de erro que ocorreu no seu pedido. Este erro tem obedecer ao seguinte formato: " {*status: ERRO*} ", onde "ERRO" é o número do erro que equivale a este tipo de validação.

Código	**RF-WS-12**
Descrição	Tem de haver uma validação para o caso da informação requerida não existir no sistema.
Informações	Quando um cliente faz um pedido ao "*Web-Service*" a informação requisitada pode não existir no sistema, assim sendo, este cliente deve receber uma mensagem que informe este facto.
Regras	Se a informação requisitada pelo utilizador não existir na base de dados, então o "*Web-Service*" terá de responder com uma mensagem no formato *JSON* que obedece à seguinte forma: " {status: ERRO} " onde "ERRO" é o número equivalente a este tipo de validação.

Código	**RF-WS-13**
Descrição	Todas as datas, i.e tempos requisitados, têm de estar representadas com o número de milissegundos a partir do dia 1 de Janeiro de 1970.
Informações	Esta funcionalidade diz respeito ao pedido de valores da leitura do satélite, quando a data inicial e final são introduzidas devem estar no formato de tempo que corresponde aos milissegundos a partir de 1/1/1970.
Regras	Ao utilizar o método *getValuesParameter* a data inicial e final introduzida nos parâmetros *initd=inttime&endd=endtime* tem de estar no formato de tempo em milissegundos, caso contrário o utilizador tem de receber a informação que o formato da data introduzido não é suportado. A resposta deve obedecer ao seguinte formato: " {status: ERRO} " onde "ERRO" é o número equivalente a este tipo de validação.

Código	**RF-WS-14**
Descrição	Tem de existir uma funcionalidade que dê informação sobre o tipo de erro que é enviado ao cliente.
Informações	Quando o cliente utiliza de forma errada os métodos públicos disponibilizados pelo "*Web-Service*", como por exemplo a falta de parâmetros introduzidos ou quando não existe a informação requisitada, este recebe uma mensagem com o tipo de erro. Desta forma, tem de existir um método específico para o cliente utilizar e receber informações sobre o erro existente.
Regras	Esta funcionalidade pode ser definida pelo método público *getErrorInformation?error=Error*. Assim, quando o utilizador recebe uma mensagem com um erro e quer saber especificamente qual foi o erro gerado, utiliza este método com o "Error" igual ao erro recebido.

Código	**RF-WS-15**
Descrição	Informação geral dos tipos de erro existentes.
Informações	Esta funcionalidade diz respeito ao envio da informação sobre todos os tipos de erro que são levantados pela validação do "*Web-Service*".
Regras	Podemos definir esta funcionalidade com o método *GetErrorList*. Desta forma, quando este método é chamado pelo utilizador o sistema envia-lhe uma lista com todos os erros levantos pelo sistema de validação do "*Web-Service*" e as suas respetivas informações.

Requisitos funcionais da aplicação *web* - MDV

Nas tabelas abaixo, estão presentes os requisitos funcionados da aplicação *web* MDV.

Código	**RF-MDV-01**
Descrição	O utilizador tem de poder explorar os dados das várias missões, instrumentos, domínios e parâmetros de forma visual e interativa, através da visualização de um grafo.
Informações	O sentido de explorar os dados das várias missões, neste caso, é referente à exploração de forma interativa e gráfica da organização geral dos dados, como por exemplo, existe a missão *EnviSat* que possui 10 instrumentos. Desta forma, existe o foco na exploração da estrutura relativa à forma como os parâmetros, domínios, instrumentos e missões estão organizados. Através da visualização de um grafo da estrutura dos dados torna-se possível ao utilizador escolher com facilidade os parâmetros que está à procura. Também torna possível a visualização de uma forma simples e interativa da organização dos dados relativos a cada missão.
Regras	Para tornar a exploração dos dados exequível e interativa podemos recorrer à representação da estrutura dos dados através de um grafo, neste caso um grafo utilizando a técnica *force directed*. Com este tipo de grafo os vértices são repelidos e atraídos por outros vértices. Assim sendo, o utilizador tem de conseguir explorar o grafo e selecionar os parâmetros que pretende para a visualização dos dados, que são relativos às medições do instrumento.

Código	**RF-MDV-02**
Descrição	Tem de ser disponibilizada ao utilizador a funcionalidade de criar e gerir listas de parâmetros.
Informações	Como forma de guardar e organizar os parâmetros escolhidos, o utilizador tem de utilizar a funcionalidade de criação e gestão de listas disponível no *web site*.
Regras	Estas listas são caracterizadas pelo facto de possuírem os vários parâmetros que são escolhidos pelo utilizador. Através desta funcionalidade, o utilizador pode criar novas listas, estas listas não podem ter o mesmo nome, assim sendo, cada lista nova tem de possuir um novo nome, caso contrário tem de ser levantado um erro e mostrado ao utilizador para que este corrija o erro. O utilizador pode gerir cada lista de forma a adicionar ou remover novos parâmetros, assim como, escolher a janela temporal associada a esta lista.

	Deve ser apresentado ao utilizador um calendário para este escolher a data de início e fim que pretende das leituras efetuadas pelo instrumento.

Código	**RF-MDV-03**
Descrição	Tem de ser mostrada ao utilizador a janela temporal disponível para os dados que estão inseridos na lista selecionada.
Informações	Esta funcionalidade tem o propósito de auxiliar, de forma interativa, a escolha da janela temporal ao utilizador. Assim, é apresentado num gráfico as janelas temporais relativas aos parâmetros que estão presentes na lista selecionada. Com isto, pretende-se que a gestão da lista seja amigável ao utilizador.
Regras	Recorrendo a bibliotecas de *Javascript* torna-se possível o desenvolvimento desta funcionalidade. Deve ser mostrado a janela temporal no gráfico de cada parâmetro que esteja presente na lista selecionada pelo utilizador. O gráfico deve ser iterativo, assim sendo, deve possibilitar a visualização das datas de início e de fim relativas às medições existentes de cada parâmetro.

Código	**RF-MDV-04**
Descrição	Possibilidade de ver os dados dos parâmetros em formato de texto.
Informações	Como forma de visualização de dados, deve ser possível visualizar os dados de forma crua, isto é, em formato de texto. Esta funcionalidade é importante pois existem algumas *flags* e dados de cabeçalhos que pode ser necessário a sua visualização sob a forma de texto.
Regras	Esta funcionalidade deve estar disponível através do *site* onde a informação selecionada deve aparecer numa tabela de dados.

Código	**RF-MDV-05**
Descrição	O utilizador pode aceder a informações relativas às missões e aos instrumentos.
Informações	O utilizador pode fazer aceder à lista de missões e de instrumentos para obter as suas informações.

Regras	O utilizador vai recorrer ao *site* para selecionar as missões e instrumentos existentes no sistema. Desta forma, são apresentadas todas as informações relativas à missão e instrumento selecionado.

Código	**RF-MDV-06**
Descrição	Tem de ser possível ao utilizador criar gráficos de linha simples, gráficos de linha com múltiplos eixos, gráficos de dispersão, gráficos de coordenadas paralelas, matrizes de dispersão e gráficos especializados para a representação de *time series*. Também deve possuir a funcionalidade de ver num mapa a orbita do satélite.
Informações	Através do *site* o utilizador tem de ter à sua disposição vários tipos de gráficos. Desta forma, para visualizar os dados pretendidos, o utilizador seleciona o gráfico e os dados para estes serem representados graficamente.
Regras	O utilizador do *site* depois de ter realizado o processo de escolha dos parâmetros pode proceder à sua visualização. Para isto, este cliente tem de ter à sua disposição vários tipos de gráficos (linha, dispersão, *time series*, coordenadas paralelas, splom). Desta forma, este interactivamente escolhe as variáveis que pretende visualizar e devem existir métodos que permitem o relacionamento de parâmetros.

Código	**RF-MDV-07**
Descrição	Tem de existir a funcionalidade de autenticação e registo na plataforma *online*.
Informações	Através do *email* e da *password* o utilizador pode fazer registo ou *login* na plataforma de visualização.
Regras	Na plataforma *online* tem de existir a funcionalidade de login e registo. Desta forma, o utilizador deve colocar nos campos de *input* o seu *email* e a sua *password* , caso clique no botão de *login* tem de receber a informação de que foi autenticado com sucesso, isto se as suas credenciais forem corretas, caso isto não aconteça, deve receber uma notificação com o erro que ocorreu. Se o utilizador clicar no campo de registo, este deve receber uma notificação que o registo correu de forma correta, caso contrário deve ser mostrada uma notificação de erro. Os *emails* devem ser únicos e toda a informação dos utilizadores deve ser armazenada na base de dados SQLite.

Código	**RF-MDV-08**

Descrição	O ambiente de trabalho do utilizador, isto é, as suas listas e configurações devem ser guardadas na base dados. Quando este utilizador voltar a fazer login, o seu ambiente de trabalho deve ser descarregado pelo *site* de forma automática.
Informações	Esta funcionalidade requer que o utilizador esteja registado e tenha realizado a autenticação. Quando o utilizador está no processo de procura e seleção dos dados, geração de listas e configuração das mesmas, é necessário guardar o seu ambiente de trabalho na base de dados, ou seja, é preciso guardar todas as listas de parâmetros e suas configurações. Quando o utilizador voltar à aplicação *web* e realizar a autenticação com sucesso, todas as suas listas e configurações devem ser descarregadas e disponibilizadas ao utilizador de forma automática.
Regras	Esta funcionalidade requer que o utilizador esteja ligado e autenticado na aplicação *web*. Ao criar listas, adicionar parâmetros e realizar as configurações (escolha do time range) estes dados são guardados de forma automática e quando o utilizador voltar a fazer a autenticação o seu ambiente de trabalho é descarregado de forma automática e sem a interação deste.

Requisitos funcionais do "*Database Updater*"

Seguidamente são apresentados os requisitos funcionais do "*Database Updater*".

Código	**RF-DU-01**
Descrição	Este sistema tem de atualizar a base de dados *CouchDb* com os dados dos ficheiros das missões.
Informações	Como forma de automatizar o processo de introdução de dados na base de dados, este sistema tem de possuir a funcionalidade de criar documentos do tipo JSON e atualizar a base de dados com estes objetos que contêm os dados e as informações das missões.
Regras	A base de dados *CouchDb* deve ser atualizada com os objetos no formato JSON que possuem dos dados das missões. Ao aceder à base de dados é possível verificar a existências dos ficheiros.

Código	**RF-DU-02**
Descrição	Tem de existir a possibilidade de filtrar os dados.
Informações	Pode não ser conveniente guardar os dados na sua totalidade na base de dados *CouchDb* sendo necessário aplicar um filtro para escolher os tipos de dados a guardar.

Regras	O filtro deve ser implementado a nível de código *Java* para permitir que certos conjuntos de dados sejam filtrados.

Código	**RF-DU-03**
Descrição	Este sistema tem de abrir os ficheiros que contêm os dados.
Informações	Existem vários ficheiros que contêm dados sobre as medições dos instrumentos. Estes ficheiros devem estar localmente na máquina.
Regras	Ler os ficheiros do disco e abrir o ficheiro, para que seja possível retirar os dados pretendidos para o projeto.

Requisitos não funcionais "*Web-Service*"

Seguidamente são apresentados os requisitos não funcionais do "*Web-Service*".

Código	**RNF-WS-01**
Descrição	O "*Web-Service*" tem de ser desenvolvido na linguagem *Java*.
Informações	Este servidor tem de ser desenvolvido na linguagem de programação *Java*. Desta forma, possibilita a utilização do paradigma de desenvolvimento de *software* orientado a objetos. Embora não seja a única linguagem para o desenvolvimento orientado a objetos, esta decisão foi tomada pela empresa *VisionSpace Technologies*.

Código	**RNF-WS-02**
Descrição	O "*Web-Service*" tem de possuir uma base de dados, utilizando o modelo não relacional, isto é, base de dados de documentos.
Informações	A base de dados para o "*Web-Service*" tem de possuir a característica de ser orientada a documentos. Será utilizado o sistema de base de dados *CouchDB*. Esta decisão foi tomada pela *VST*.

Código	**RNF-WS-03**
Descrição	O "*Web-Service*" tem de comunicar com a base de dados.
Informações	Através da livraria *Ektorp* desenvolvida em *Java* torna-se possível aceder à base de dados *CouchDB* e, assim sendo, fazer pedidos de dados e receber os dados para que sejam processados e posteriormente enviados ao utilizador.

Código	**RNF-WS-04**
Descrição	O "*Web-Service*" tem de possuir uma API pública capaz de receber pedidos através do protocolo *http*.
Informações	Com o desenvolvimento da API pública torna-se exequível utilizar este "*Web-Service*" com o propósito de enviar vários tipos de informação e dados relativos às missões existentes na base de dados. Assim sendo, esta plataforma fica disponível para receber pedidos de clientes, podendo estes estarem localizados na internet.
Regras	Através da utilização de um *web-browser* tem de ser possível realizar um pedido *http* (GET) à API pública do "*Web-Service*".

Código	**RNF-WS-05**
Descrição	O "*Web-Service*" tem de possuir a funcionalidade de validação de pedidos.
Informações	Tem de existir uma funcionalidade de validação dos pedidos realizados pelos utilizadores. Desta forma, caso o utilizador introduza o valor parâmetro de maneira incorreta, como por exemplo se a data final for inferior a data inicial, este problema tem de ser detetado e logo de seguida o utilizador deve ser informado através de uma mensagem formatada em JSON.
Regras	Ao realizar um pedido *http* inválido (através de um *browser)* à API pública do "*Web-Service*" este deve responder com uma mensagem de erro.

Requisitos não funcionais da aplicação web - MDV

Abaixo estão todas as tabelas com informação relativa aos requisitos não funcionais da aplicação *web* mdv.

Código	**RNF-MDV-01**
Descrição	Utilização da API pública do "*Web-Service*".
Informações	Como todos os dados e informações referentes às missões estão na base

	de dados do "*Web-Service*" é necessário utilizar a sua API pública para receber e mostrar estes mesmos dados aos utilizadores.
Regras	A aplicação *web* deve realizar vários pedidos *http* à API do "*Web-Service*" e, assim, receber dos dados requisitados.

Código	**RNF-MDV-02**
Descrição	A aplicação *web* MDV tem de ser desenvolvida utilizando a *framework Ruby on Rails*.
Informações	Este requisito foi decidido pela empresa *VisionSpace Technologies*. Desta forma vai ser utilizada a linguagem de programação *Ruby* e a *framework Ruby on Rails*.

Código	**RNF-MDV-03**
Descrição	O *site* vai ser desenvolvido utilizando *html*, *css* e *Javascript*.
Informações	Para conseguir atingir a visualização de dados pretendida, terá de ser utilizada a linguagem de marcação *html5* com recurso ao *css* para fazer a apresentação dos documentos *html*. A linguagem *Javascript* vem dar o suporte necessário para implementar e usar algumas livrarias disponíveis para a visualização de dados.

Requisitos não funcionais do "*Database Updater*"

Todos os requisitos não funcionais do "*Database Updater*" estão descritos nas tabelas abaixo.

Código	**RNF-DU-01**
Descrição	O sistema tem de ser desenvolvido na linguagem de programação *Java*.
Informações	Será utilizada a linguagem de programação *Java* para recorrer à programação orientada a objetos. Requisito este decido pela *VST*.

Código	**RNF-DU-02**
Descrição	O sistema tem de comunicar com a base de dados.

Informações	Através da utilização da biblioteca *Ektorp* é possível comunicar e aceder à base de dados *CouchDB*, o que permite realizar as actualizaçõe.

Requisitos externos do "*Database Updater*"

Abaixo estão presentes as tabelas contendo a informação relativa aos requisitos externos do "*Database Updater*".

Código	**RE-DU-01**
Descrição	Necessidade da existência dos ficheiros que contêm os dados e as informações dos instrumentos.
Informações	Para que seja exequível a introdução ou atualização da base de dados é necessário que estejam disponíveis os ficheiros com os dados e informações. Estes ficheiros são disponibilizados pela ESA.

Código	**RE-DU-02**
Descrição	Comunicação com a base de dados.
Informações	É necessário que a base de dados *CouchDB*, onde irão ser guardados todos os campos dos ficheiros pretendidos, esteja a funcionar no sistema e assim, a comunicação entre os dois sistemas é realizada.
Regras	Através da ferramenta *Ektorp* é possível realizar a comunicação com a base de dados, caso esta comunicação não efetuada, pode significar que a base de dados não está a funcionar no sistema.

Requisitos externos do "Web-Service"

Os requisitos externos do "Web-Service" estão representados nas tabelas abaixo.

Código	**RE-SW-01**
Descrição	Comunicação com a base de dados.
Informações	Um importante requisito externo do "*Web-Service*" resulta da necessidade do funcionamento da base de dados onde estão guardadas todas as informações importantes para o projeto. A comunicação entre estes dois sistemas é crucial. Se esta comunicação falhar, torna-se impossível

	responder aos utilizadores com a informação que por eles foi requisitada.
Regras	Através da ferramenta *Ektorp* torna-se possível realizar a comunicação entre a base de dados e o serviço *web*. Caso esta comunicação falhe, ou não seja possível realizar a comunicação, pode significar que a base de dados não está a funcionar no sistema.

Código	**RE-SW-02**
Descrição	Necessidade de acesso à Internet.
Informações	Como este servidor tem como principal objetivo responder a pedidos de utilizadores, que podem estar situados em regiões diferentes do globo, é necessário que o servidor esteja conectado à internet. Caso exista algum problema ou falha na comunicação com a Internet, o servidor deixa de comunicar e torna-se impossível responder a qualquer pedido que venha de fora da sua rede.
Regras	Utilizando uma máquina que possua acesso à internet, pode ser realizado, através de um *browser*, um pedido *http* ao serviço web. Caso este pedido retorne uma mensagem de não encontrado ou de erro, significa que o "*Web-Service*" não se encontra a comunicar com a internet e, assim sendo, deixa de ser possível responder a pedidos vindos da internet.

Requisitos externos da aplicação *web* - MDV

Nas seguintes tabelas então representados os requisitos externos da aplicação *web*.

Código	**RE-MDV-01**
Descrição	Comunicação com o "*Web-Service*".
Informações	Como requisito externo do "*Web-Server*" é necessário a comunicação com o "*Web-Service*". Isto deve-se ao facto da aplicação *web* recorrer ao "*Web-Service*" através do protocolo *http* para pedir os diferentes tipos de dados existentes em base de dados. Estes dados, dizem respeito às medições dos instrumentos, que são necessários para apresentar ao utilizador da página *Web*.
Regras	Caso não seja possível estabelecer uma comunicação com o "Web-Service", a aplicação *web* não tem forma de mostrar dados aos utilizadores.

Código	**RE-MDV-02**
Descrição	Comunicação com a Internet.
Informações	Se existirem falhas na comunicação com a internet, deixa de ser possível receber visitas de utilizadores que estejam fora da rede.
Regras	Utilizando uma máquina com acesso à internet e recorrendo a um *browser* realiza-se um pedido à aplicação *web*, caso a resposta seja "não encontrado" ou "erro" significa que a aplicação *web* não se encontra disponível.

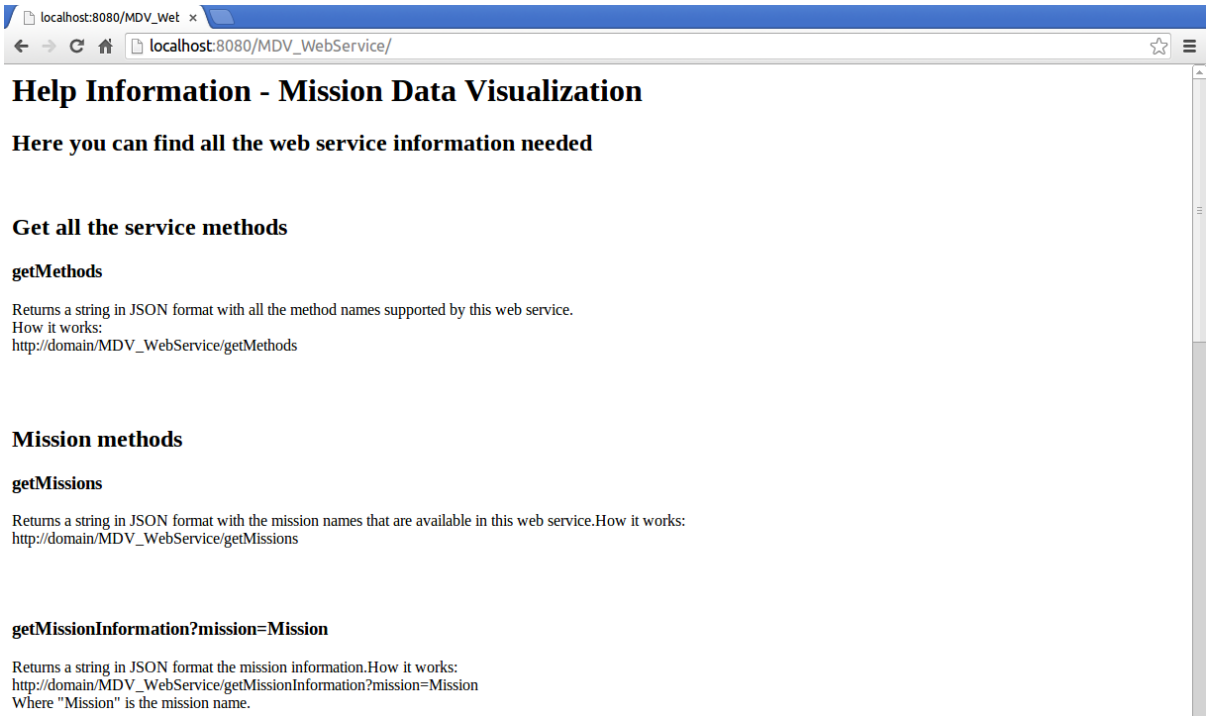
Código	**RE-MDV-03**
Descrição	Utilização de um *browser* para aceder aos conteúdos.
Informações	Para a aceder aos conteúdos da aplicação *web*, ou seja, ver a informação de forma interativa e com as representações gráficas dos dados, é necessário que o cliente utilize um *browser* de internet.

Apêndice B

"*Web-Service*" User Manual

This user manual will cover the basics of using the web service that was implemented under the Mission Data Visualization Project. This service provides a public API in order to receive data requests that comes from a user or other system. There is no need to make any registration on the service in order to request data.

In this user manual the web service is hosted at the localhost, when it is deployed in one server connected to the Internet it will have an URL which can be accessed from any country in the world. In this manual the URL to access the web service is "http://localhost:8080/MDV_WebService/", when it gets the final URL simply replace the URL ("http://localhost:8080/MDV_WebService/") with the new one. When the user accesses the web service link using the internet browser he will receive an html with the help he needs.



How to get the help page

To get the html help page, same page like figure 1 shows, users can call the method "help" of the web service public API. How it works:

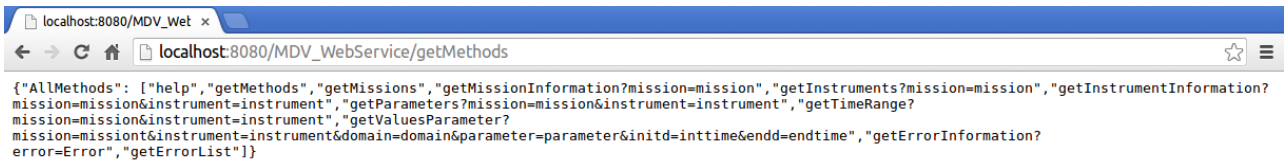
Method expected by the API: help;

Call example: http://localhost:8080/MDV_WebService/help;

Response: Help html.

How to get all the API methods

In order to get all the web service API methods, in JSON format, the user needs to call the method called "getMethods". How it works:



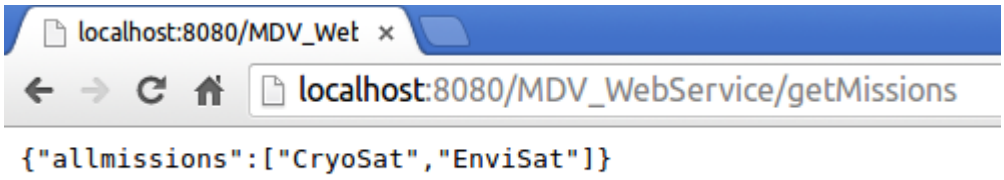
Method expected by the API: getMethods;

Call example: http://localhost:8080/MDV_WebService/getMethods;

Response: All the methods in JSON format. This JSON is an object that contains a key called "allMethods" and the value is an array of methods.

How to get all the existing missions

To get all the missions archived in the web service, the users need to call the API method "getMissions". How it works:



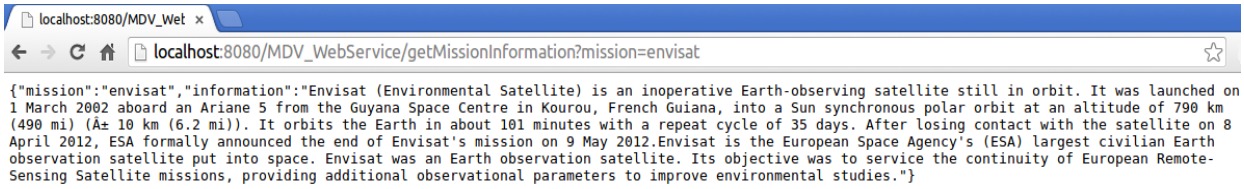
Method expected by the API: getMissions;

Call example: http://localhost:8080/MDV_WebService/getMissions;

Response: All the missions in JSON format. This JSON object contains a key called "allmissions" and the value is an array with all the mission names.

How to request the mission information

In order to request the information about one specific mission, the web service API provides one method called "getMissionInformation". How it works:



Method expected by the API: getMissionInformation?mission=MISSION, where "MISSION" is the mission name;

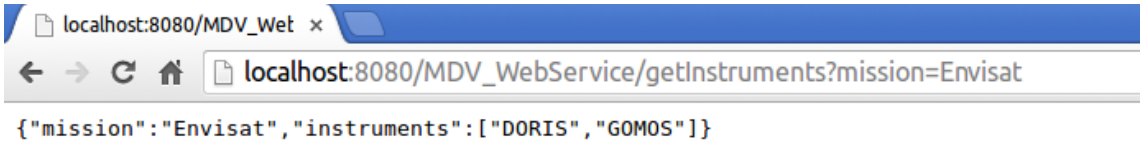
Call example:

http://localhost:8080/MDV_WebService/getMissionInformation?mission=envisat;

Response: The mission information in JSON format. This response contains a JSON object that has one key named "mission" and the value is the mission name, the other key is "information" and his value is a string with the information.

How to request all the instruments of a mission

To request the instruments of a mission, the web service API provide the method called "getInstruments". This way the user can get all the instruments of a specific mission. How it works:



Method expected by the web service API: getInformation?mission=MISSION, where "MISSION" is the mission name;

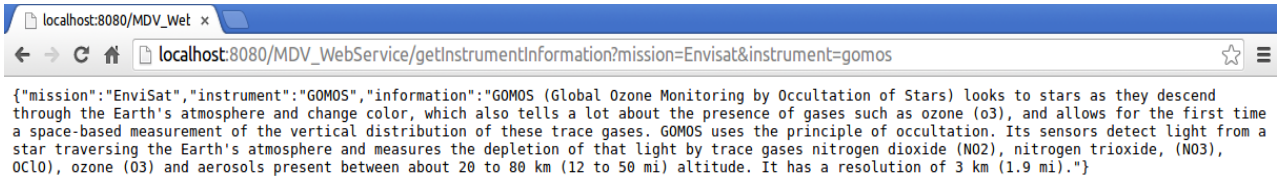
Call example:

http://localhost:8080/MDV_WebService/getInformation?mission=envisat;

Response: All the mission instruments coded in JSON format. In this JSON object exists two keys, the first one is the "mission" and the value is the mission name, the second key is "instruments" and has an array with the instrument names.

How to request the information of a specific instrument

To get the information of a specific instrument, which exists in some mission, there is a method called "getInstrumentInformation". How it works:



Method expected by the web service API:

http://localhost:8080/MDV_WebService/getInstrumentInformation?mission=MISSION &instrument=INSTRUMENT, where "MISSION" is the mission name and "INSTRUMENT" is the instrument name;

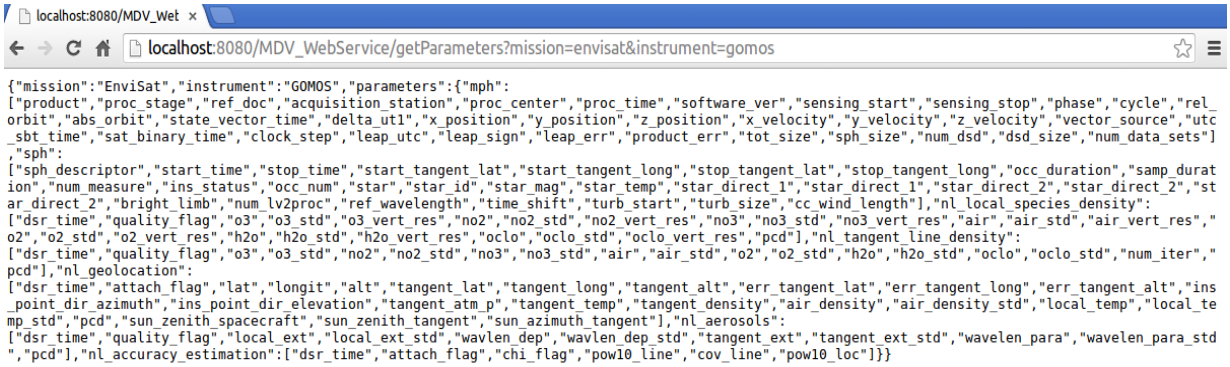
Call Example:

http://localhost:8080/MDV_WebService/getInstrumentInformation?mission=Envisat&instrument=gomos;

Response: The instrument information coded in JSON format.

How to get all the parameters inside one instrument

The instruments of a mission makes measurements of some kind, this generate lots of data that is organized as parameters. With this web service method is it possible to get the available parameters inside the requested instrument. The method is called "getParameters" and it works this way:



{"mission":"EnviSat","instrument":"GOMOS","parameters":{"mph":
["product","proc_stage","ref_doc","acquisition_station","proc_center","proc_time","software_ver","sensing_start","sensing_stop","phase","cycle","rel_
orbit","abs_orbit","state_vector_time","delta_ut1","x_position","y_position","z_position","x_velocity","y_velocity","z_velocity","vector_source","utc
_sbt_time","sat_binary_time","clock_step","leap_utc","leap_sign","leap_err","product_err","tot_size","sph_size","num_dsd","dsd_size","num_data_sets"]
,"sph":
["sph_descriptor","start_time","stop_time","start_tangent_lat","start_tangent_long","stop_tangent_lat","stop_tangent_long","occ_duration","samp_durat
ion","num_measure","ins_status","occ_num","star","star_id","star_mag","star_temp","star_direct_1","star_direct_1","star_direct_2","star_direct_2","st
ar_direct_2","bright_limb","num_lv2proc","ref_wavelength","time_shift","turb_start","turb_size","cc_wind_length"],"nl_local_species_density":
["dsr_time","quality_flag","o3","o3_std","o3_vert_res","no2","no2_std","no2_vert_res","no3","no3_std","no3_vert_res","air","air_std","air_vert_res","
o2","o2_std","o2_vert_res","h2o","h2o_std","h2o_vert_res","oclo","oclo_std","oclo_vert_res","pcd"],"nl_tangent_line_density":
["dsr_time","quality_flag","o3","o3_std","no2","no2_std","no3","no3_std","air","air_std","o2","o2_std","h2o","h2o_std","oclo","oclo_std","num_iter","
pcd"],"nl_geolocation":
["dsr_time","attach_flag","lat","longit","alt","tangent_lat","tangent_long","tangent_alt","err_tangent_lat","err_tangent_long","err_tangent_alt","ins
_point_dir_azimuth","ins_point_dir_elevation","tangent_atm_p","tangent_temp","tangent_density","air_density","air_density_std","local_temp","local_te
mp_std","pcd","sun_zenith_spacecraft","sun_zenith_tangent","sun_azimuth_tangent"],"nl_aerosols":
["dsr_time","quality_flag","local_ext","local_ext_std","wavlen_dep","wavlen_dep_std","tangent_ext","tangent_ext_std","wavelen_para","wavelen_para_std
","pcd"],"nl_accuracy_estimation":["dsr_time","attach_flag","chi_flag","pow10_line","cov_line","pow10_loc"]}}

Method expected by the web service API:

http://localhost:8080/MDV_WebService/getParameters?mission=MISSION&instrument=INSTRUMENT, where "MISSION" is the mission that contains the instrument and "INSTRUMENT" is the instrument name.

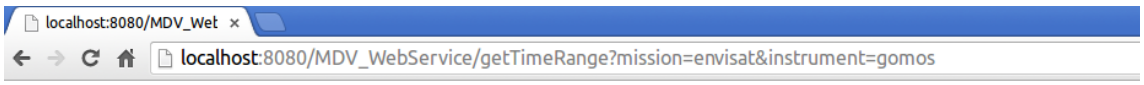
Call Example:

http://localhost:8080/MDV_WebService/getParameters?mission=envisat&instrument=gomos;

Response: All the parameters inside their domains (datasets).

How to get the instrument time range

The web service API provides a method to retrieve the time range available of each instrument, with this time range it is possible to get the data available in some time window range. This method is called "getTimeRange" and works this way:



{"mission":"envisat","instrument":"gomos","timerange":[{"tinit":"1.296600239606E12","tend":"1.296600285602E12"},
{"tinit":"1.296600483109E12","tend":"1.296600524106E12"},{"tinit":"1.296600684945E12","tend":"1.296600726942E12"},
{"tinit":"1.296600809087E12","tend":"1.296600855584E12"},{"tinit":"1.296601028622E12","tend":"1.296601078618E12"},
{"tinit":"1.296601288894E12","tend":"1.296601339391E12"},{"tinit":"1.2966019422E12","tend":"1.296602005695E12"},
{"tinit":"1.296602192922E12","tend":"1.296602242919E12"},{"tinit":"1.296602589665E12","tend":"1.296602630663E12"},
{"tinit":"1.296603467625E12","tend":"1.296603515122E12"},{"tinit":"1.296603802522E12","tend":"1.296603843519E12"},
{"tinit":"1.296604085412E12","tend":"1.296604412391E12"},{"tinit":"1.296604278579E12","tend":"1.296604315077E12"},
{"tinit":"1.296604679421E12","tend":"1.296604743417E12"},{"tinit":"1.305154816546E12","tend":"1.305154858543E12"},
{"tinit":"1.305155255206E12","tend":"1.305155300703E12"},{"tinit":"1.305155400276E12","tend":"1.305155452772E12"},
{"tinit":"1.305155511248E12","tend":"1.305155566745E12"},{"tinit":"1.305155661298E12","tend":"1.305155715295E12"},
{"tinit":"1.305155853755E12","tend":"1.305155916751E12"},{"tinit":"1.305155962648E12","tend":"1.305156018644E12"},
{"tinit":"1.305156128766E12","tend":"1.305156167764E12"},{"tinit":"1.305156244029E12","tend":"1.305156284526E12"},
{"tinit":"1.305156578686E12","tend":"1.305156625683E12"}]}

Method expected by the web service API:

http://localhost:8080/MDV_WebService/getTimeRange?mission=MISSION&instrument=INSTRUMENT, where the "MISSION" is the mission name and the "INSTRUMENT" is the instrument name";

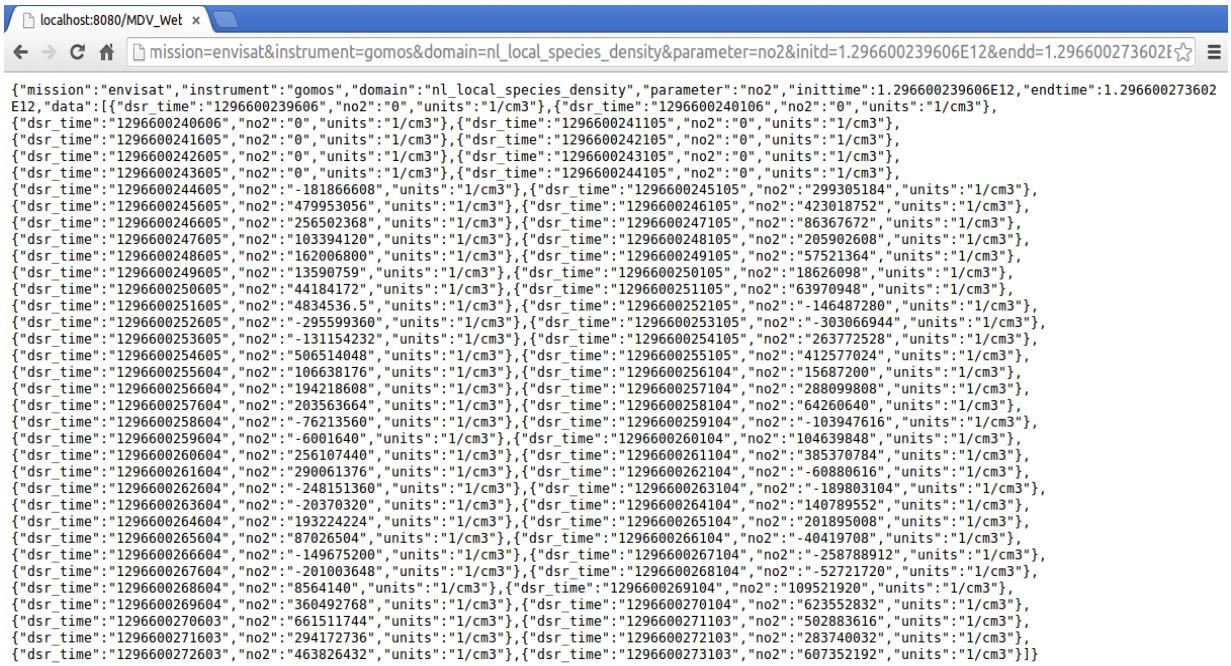
Call Example:

http://localhost:8080/MDV_WebService/getTimeRange?mission=envisat&instrument=gomos;

Response: The response in this API method has all the time range available of each sequence of measurements, this way it is shown "tinit" which mean the time initial of the measurement and the "tend" which means the end time for that measurement. The time comes in milliseconds from january, 1 of the year 1970;

How to get parameters data from the web service

In order to get the parameters data available inside the database that is accessible to the web service there is a method called "getValuesParameter". With this API method the user can get the data inside some time frame. How this method work:



Method expected by the web service API:

http://localhost:8080/MDV_WebService/getValuesParameter?mission=MISSION&instrument=INSTRUMENT&domain=DOMAIN&parameter=PARAMETER2&initd=ITIME&endd=ETIME, this means that the web service API in order to give the data need to have the mission name ("MISSION"), the instrument name ("INSTRUMENT"), the domain name ("DOMAIN"), the parameter name ("PARAMETER"), the initial time ("ITIME") and final time ("ETIME") of the measurement. Notice that the initial time and final time is the time expressed in milliseconds.

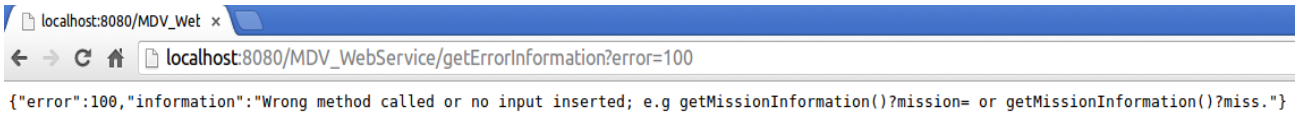
Call example:

http://localhost:8080/MDV_WebService/getValuesParameter?mission=envisat&instrument=gomos&domain=nl_local_species_density&parameter=no2&initd=1.296600239606E12&endd=1.296600273602E12; this method requests the mission envisat, instrument gomos, domain nl_local_species_density, parameter no2, initd (initial date) and endd (end date).

Response: A sequence of data expressed in time series, where for each timestamp there is an associated value, in our case no2 field contains the data and the field units represents the units of the requested data.

How to get error information

In some cases it could happen that the methods are called with some kind of errors, it can be the method name that is wrong, the mission, instrument, domain and parameter doesn't exist, or the time range requested is not available. This way the web service respond to the user request with on error message. The error message is formatted in JSON and its like "{status:ERROR}, the "ERROR" is the error number. This feature gives a better understanding of the existing problem. If the user is informed of one error he can use the method "getErrorInformation" that exists in the API to get the error information.



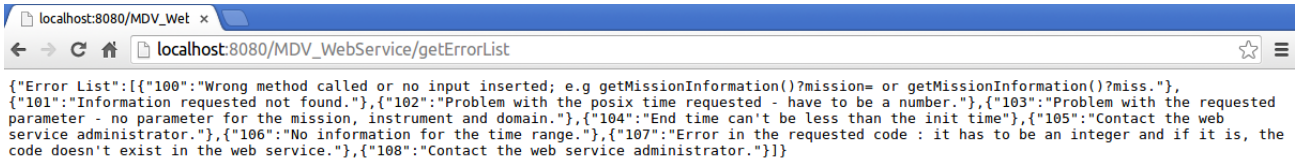
Method expected by the API: getErrorInformation=?error=100;

Call example:http://localhost:8080/MDV_WebService/getErrorInformation?error=100 ;

Response: The web service return in a JSON formatted string the error information.

How to get the list of error

In order to visualize all the errors that can occur the web service provides a method called "getErrorList".



Method expected by the API: getErrorList;

Call example: http://localhost:8080/MDV_WebService/getErrorList;

Response: The web service responds with a string formatted as JSON with all the error information.

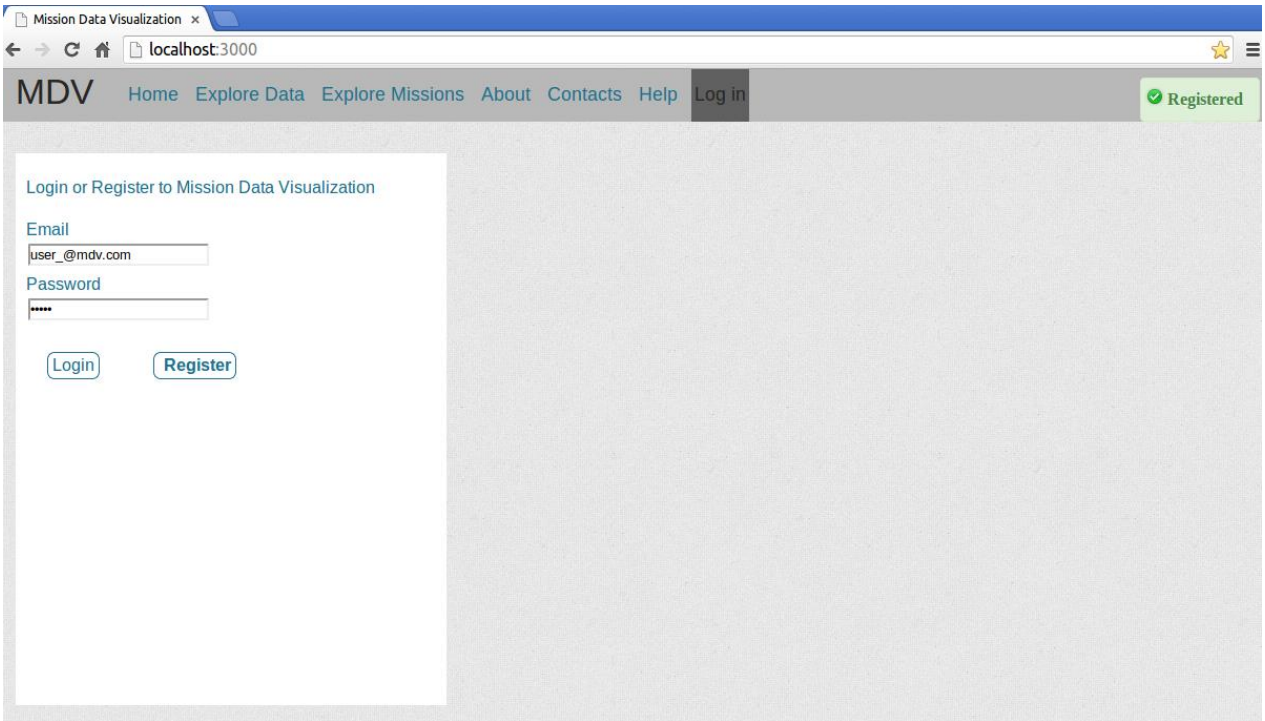
Apêndice C

User Manual for MDV web application

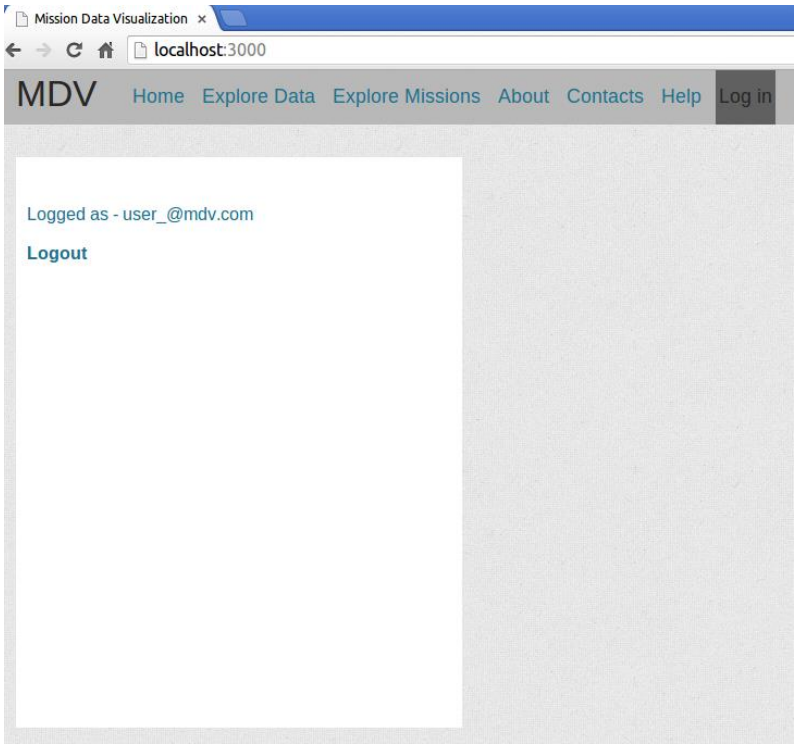
This manual was built in order to provide some useful information's in how to use the MDV web application. This platform was developed to provide the user the possibility of visualization over the available mission data, this manual will show how to perform all the visualization types in order to explore the data and select it. It also will explain how to do simple tasks like register and login to the web application.

Register and Login

In order to make a registration to the web application or perform a login. In main menu select the "Log in" tab. If the user wants to make a registration, he needs to fill the email and password with his data and click on "register", if one green confirmation appear it means that the user correctly registered himself, otherwise the user need to change the email name. If the user wants to make a login, just type the email and passwords in the fields and click the "login" button, if one red notification shows up in the screen verify the email and the password, if the user wants to logout just click in the "logout" button.

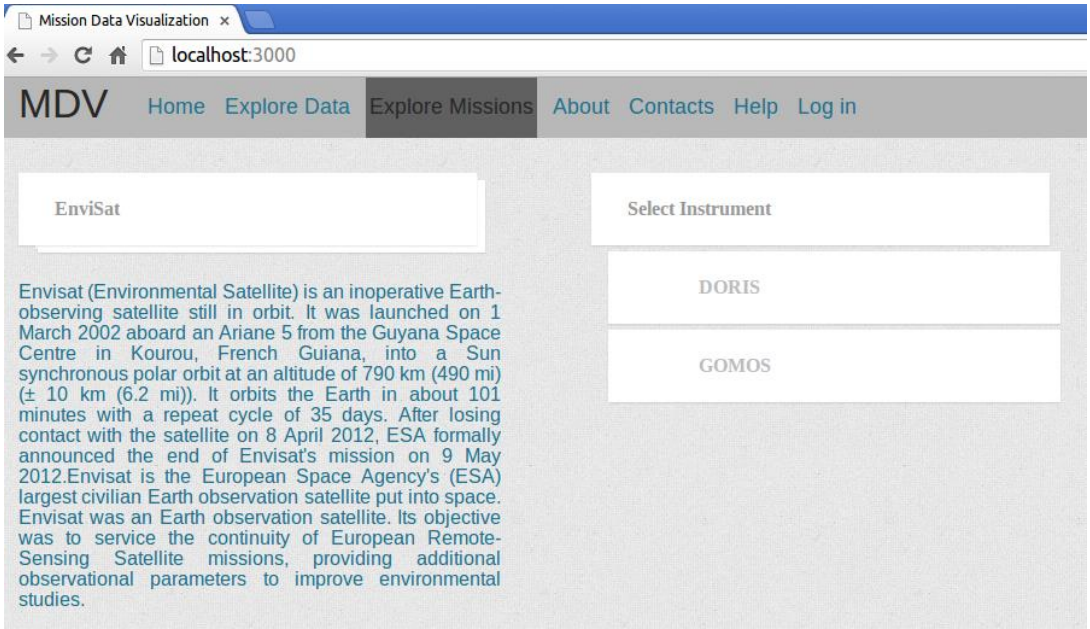


Why register in MDV web application? In this platform was developed a feature that saves all the user context, this means that all the lists builded, parameters added to this lists and time range selected will be automatically saved to the web application database and when the user reenter and perform the login action, all his environment is downloaded and putted into the right place. With this feature, every time the user uses this platform, there is no need to rebuild all the lists and enter parameters, it is already in place and no time wasted doing that.



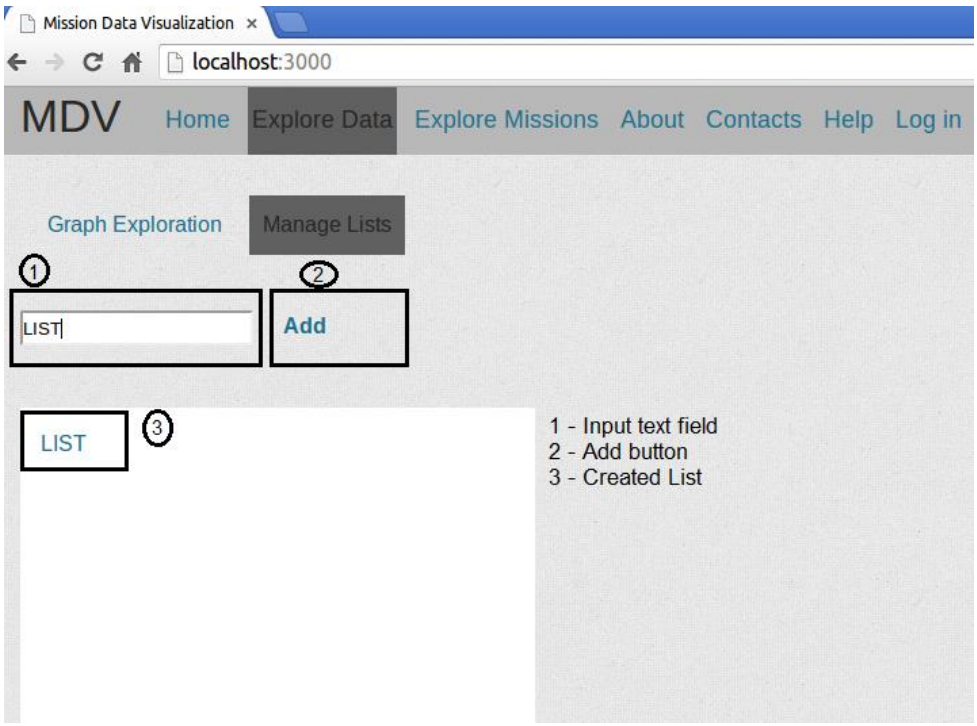
See Mission and Instrument information

In order to understand a little bit more about the mission and all his instruments, the MDV web application builded a feature to show this information. To access it, in the main menu select the "Explore Missions" tab, now choose the mission on the first drop down menu and select the instrument in the right drop down menu when it appears.



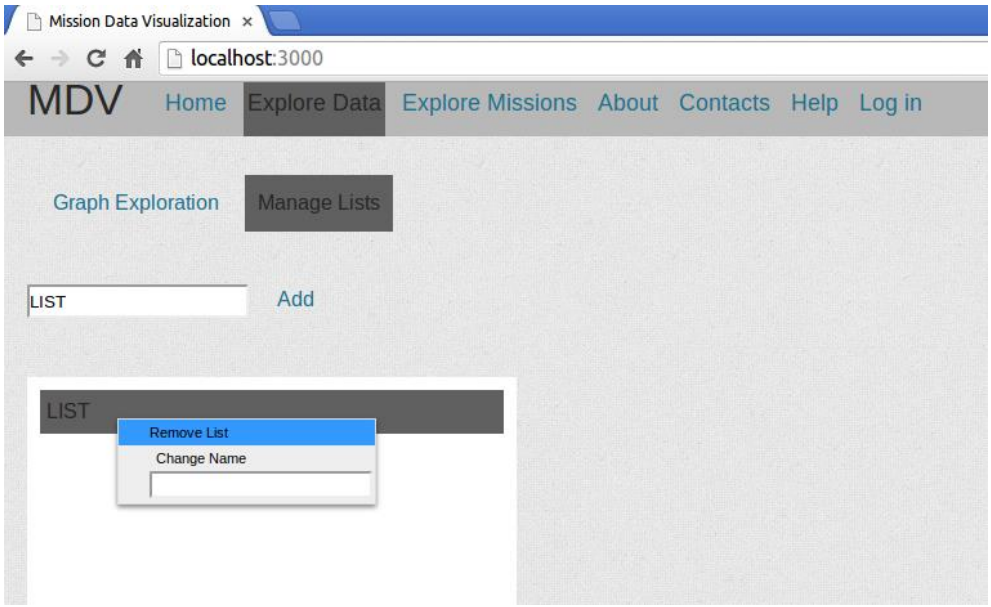
Create and Manage Lists

A list is a container that can hold parameters. With this web application the user can create, rename and delete lists. In order to create a list, the user need to click the "Explore Data" tab in the main menu and click in the "Manage Lists" button, next he can fill the input field with the list name and click "add".



In this example it was created a list named "LIST". All the lists have to be named with a different name, case one list already exists (same name as other list), the user will see a red notification in the web page.

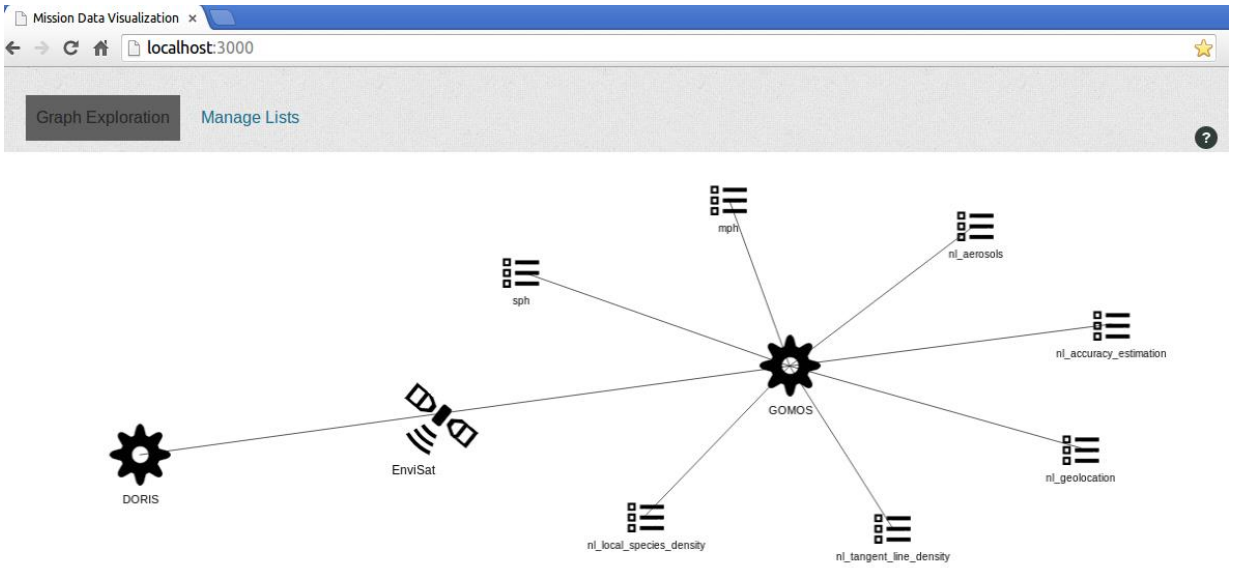
In order to rename or delete a list, right click on the list and choose the action in the context menu. If the action is to change the list name, put the new list name in the input box that appear in the context menu and hit enter.



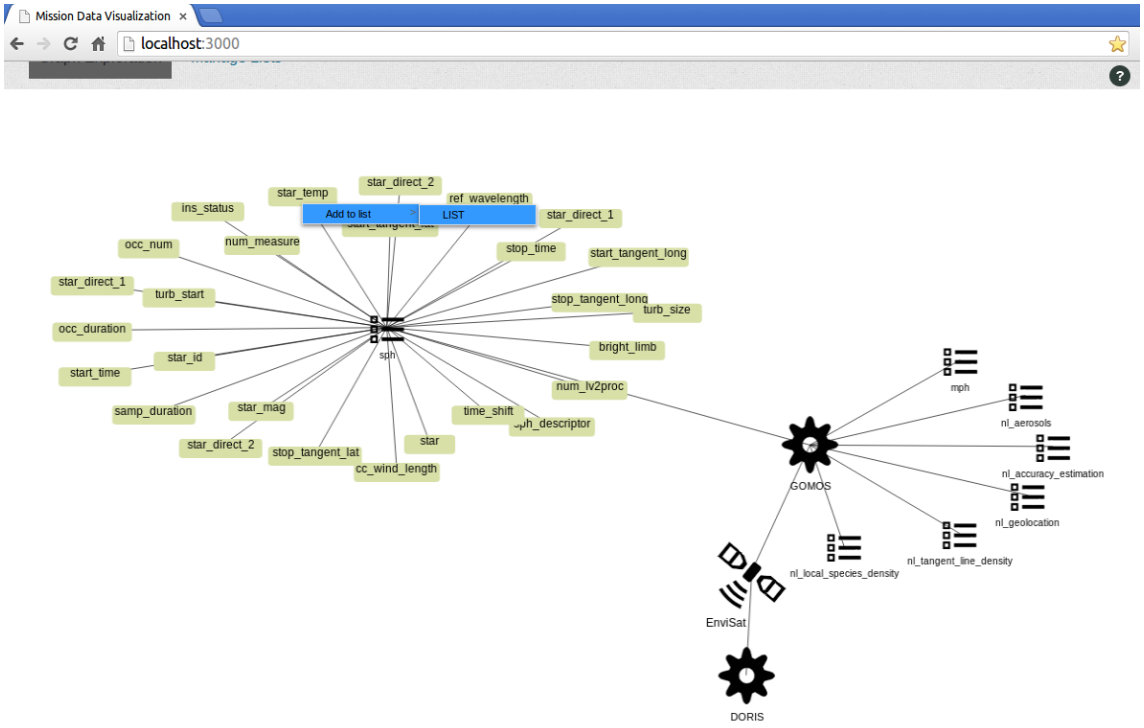
Search and add data

The MDV web application provides an interactive graph that represents all the missions, instruments, domains and parameters of the data that exists in the system. This way the user can explore the mission, visualizing all its parameters and domains. This provides a better look over the mission data structure. The visualization of the interactive graph can be found under the "Explore Data" tab in the main menu and by clicking the "Graph Exploration" button.

When the user finds one parameter that he is interested, he can include this parameter to some data list in order to visualize it. To add a parameter to the list, the user first need to find and right click the parameter in the graph view, doing this it will appear one context menu with the existing lists and now the user can include the parameter in some list by clicking the list name.



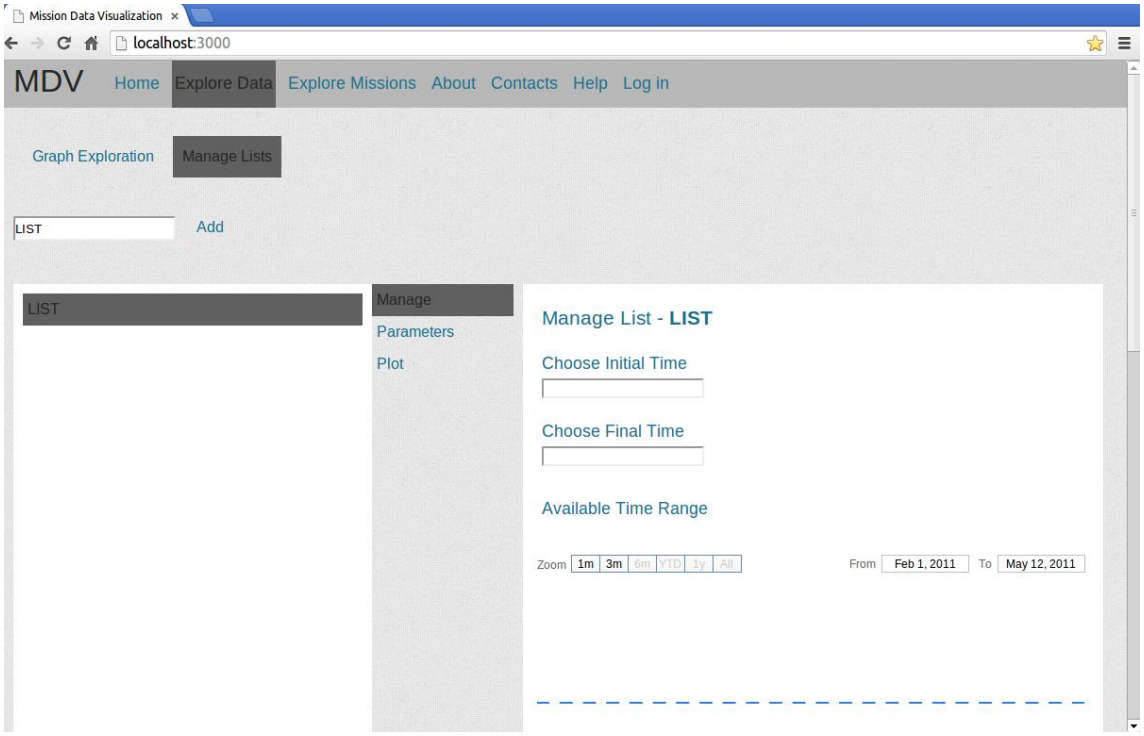
In this image is it possible to view the envisat mission expanded, this shows the GOMOS and DORIS instruments. The GOMOS instrument is expanded and his domains are displayed.



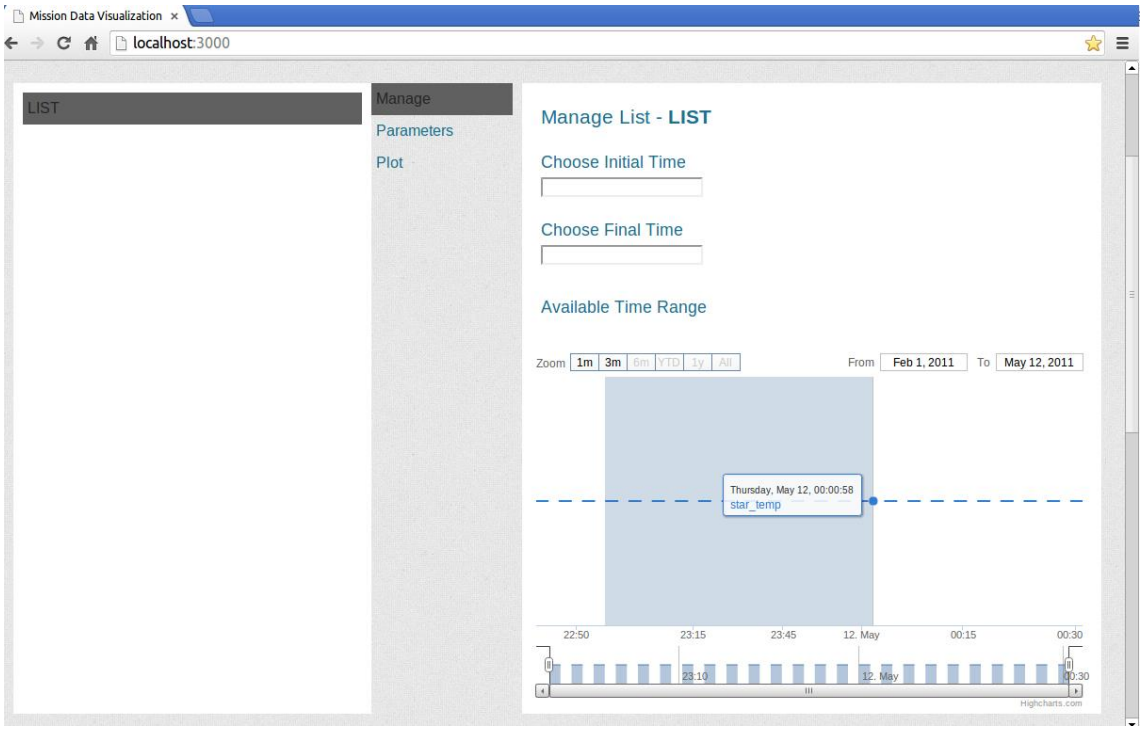
The user can only include once the parameter in some list, if the list already has this element the user can't add him to that list again.

Get the Data

In order to get the data (parameters data) first the user need to create a list and add parameters to that list. Now that there are lists and parameters inside them, the user can select the time range that the parameters chosen are available. To see this the user need to click the "Manage Lists" button under the "Explore Data" tab, click the list he is interested and click "Manage", this can be seen in the next image.



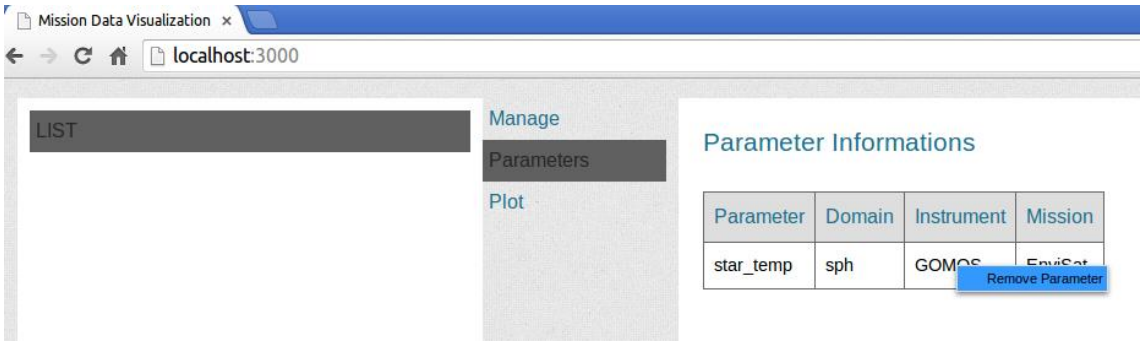
Now the user can select the time range he needs, so in the "Manage List" section on the right it is possible to see a visualization showing some intervals of data. This intervals means that the parameter which is represented by the color in the graphic exists under that time frame. To select the time frame the user only need to click and drag in the graphic to select the data. Figure 10 shows this action.



Now the list time range is selected, it can be seen on the "Initial Time" and "Final Time" the time interval that was chosen.

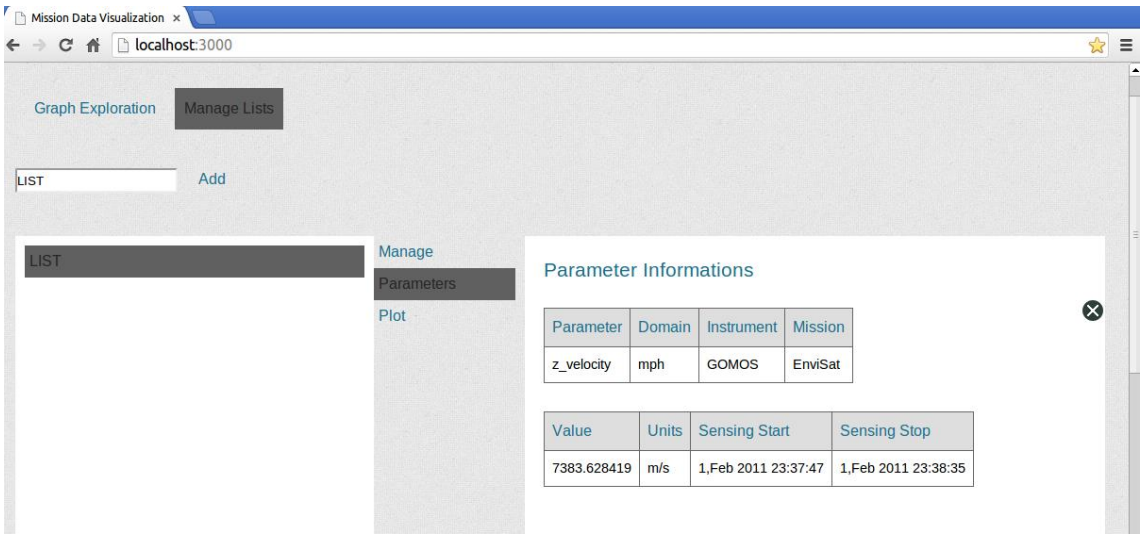
Remove a parameter from the List

If the user is no longer interested in one list parameter, there is a way to remove it. Select the "Parameters" button and right click on the parameter name that is to remove and then click remove. If the user wants to put the same parameter again on the list, he has to go to the interactive graph and select it again.



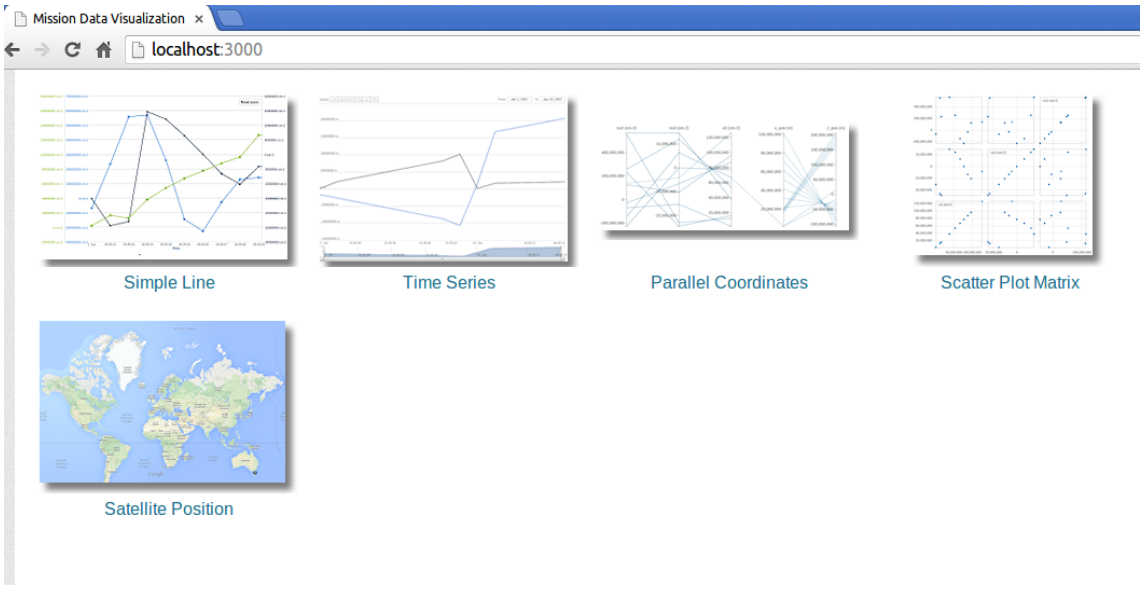
View the data in table view (raw data)

The user can see the data in a table format visualization, this way it is represented the raw data over all lines of the table.



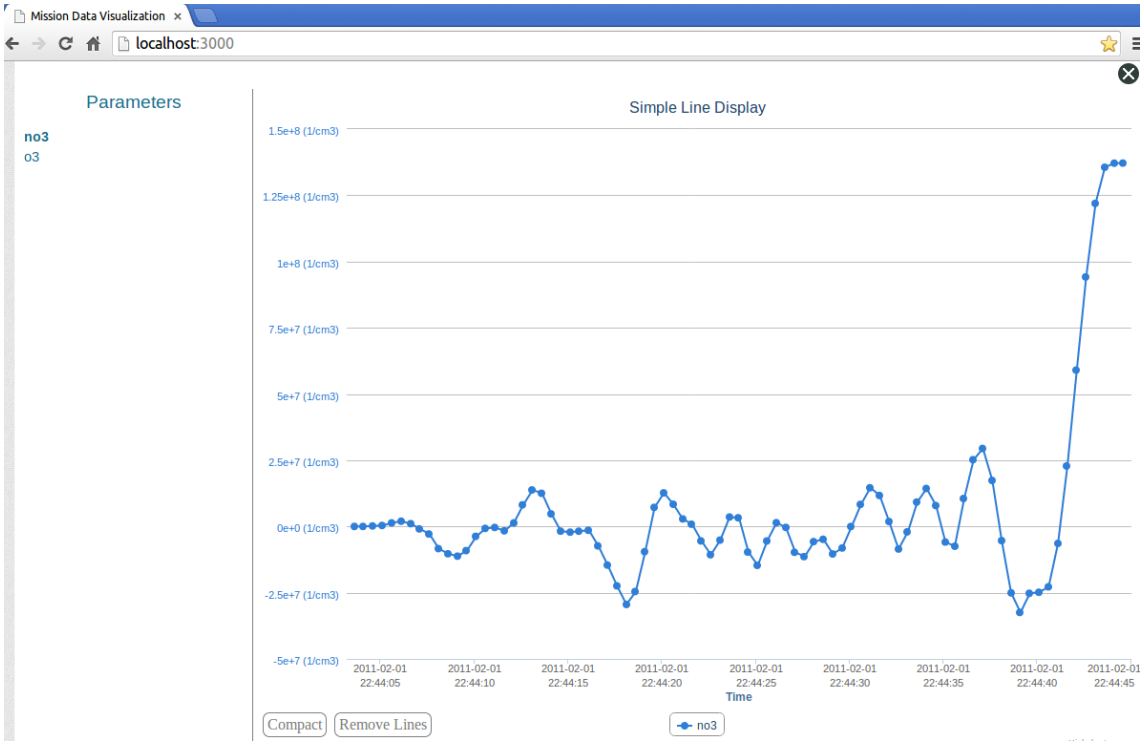
Plot the data available

When the user finishes creating lists, adding parameters and managing them he can select the plot button in order to see all the MDV web application visualizations types available. To do this he needs to click on the "Plot" button and he will be placed on the available plots section.

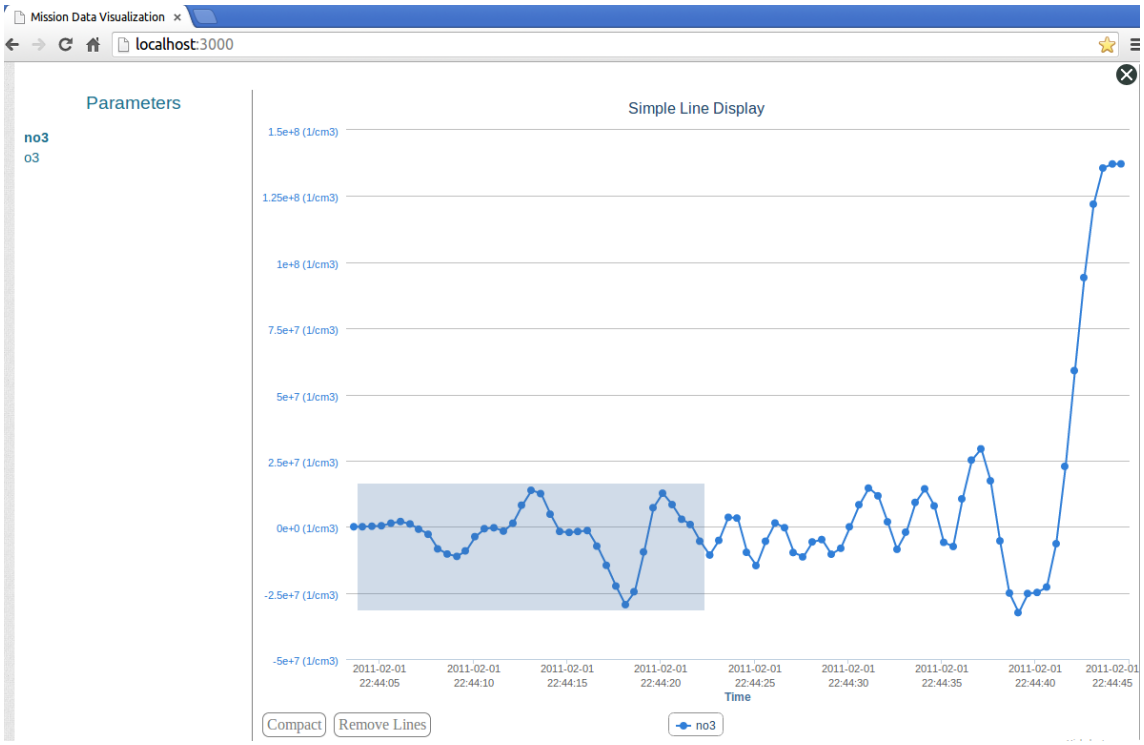


Simple Line Visualization

If the user selects the "Simple Line" graphic type he can use this type of visualization to create simple line graphics and scatter plots with the chosen data (that is in the selected list). In simple line visualization the user can selected the parameter on the left in order to build a visualization type with the selected data.



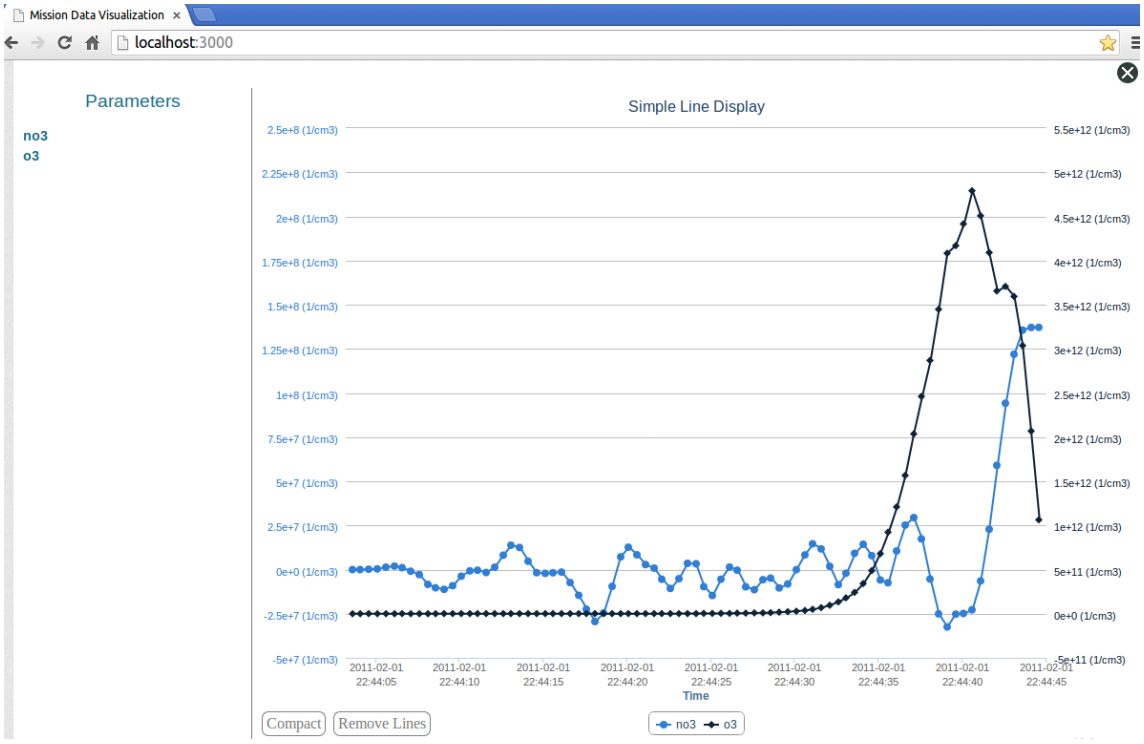
In the figure above it is possible to see that the parameter "no3" is shown in the simple line graphic here the x axis is the time that is defined and the y axis is the unit type of the parameter, in this case "1/cm^3". The user can perform a zoom in the plot, just click and drag the section in order to zoom it.



In the figure above is shown the click and drag over the plot to perform a zoom in it.

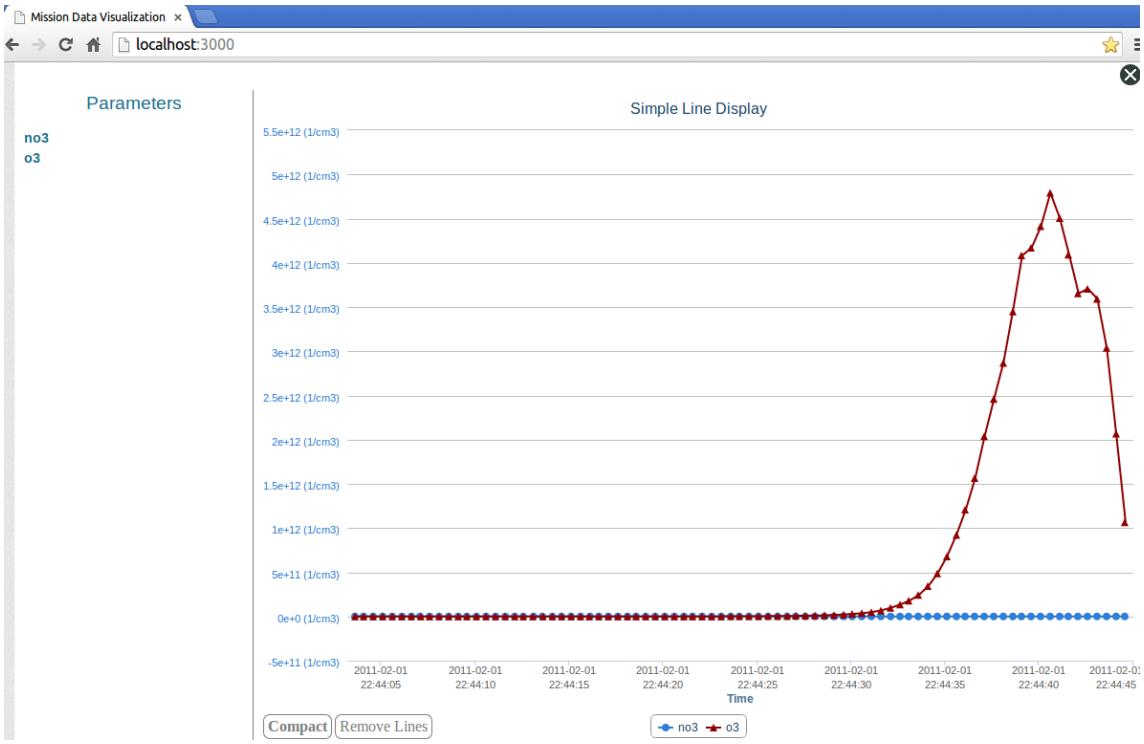
In the figure above the data is zoomed in.

If the user wants to represent another parameter in the same graphic it only has to select the parameter on the left by clicking on it. This way another dimension can be created on the plot, also another axis is added by default.



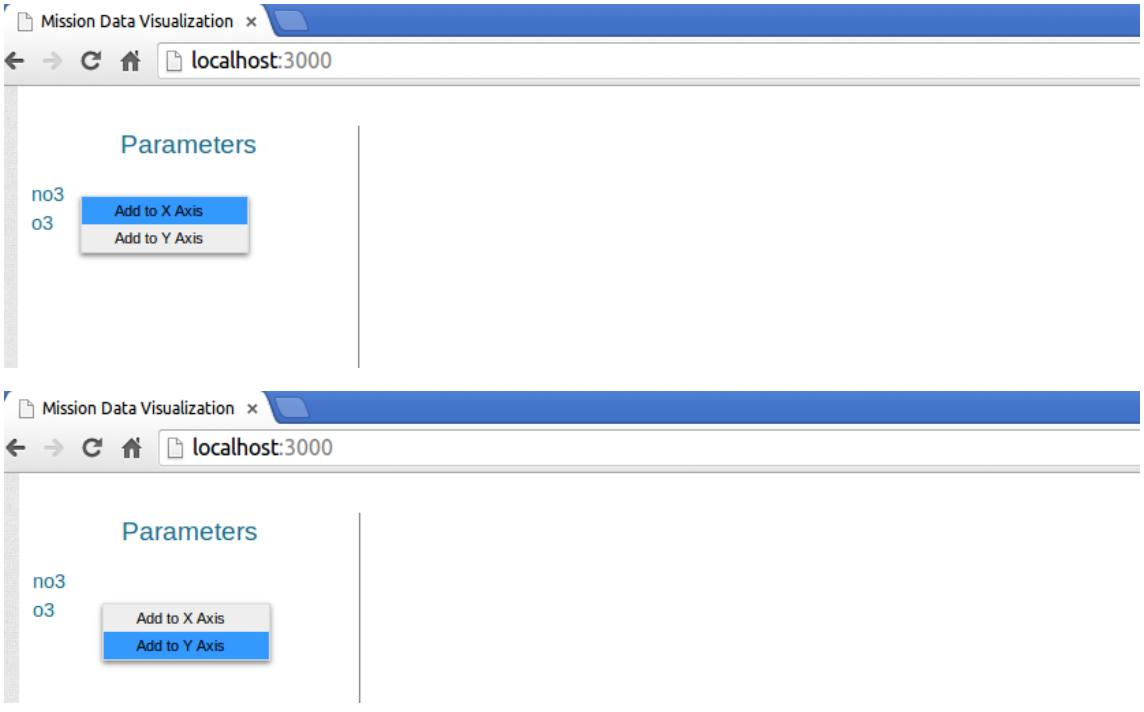
In the figure above is shown the visualization of two variables represented in the same plot, there are two y axis and one x axis. But if the user wants to represent the two variables in the same y axis, the first condition is that the parameters need to have the same data units

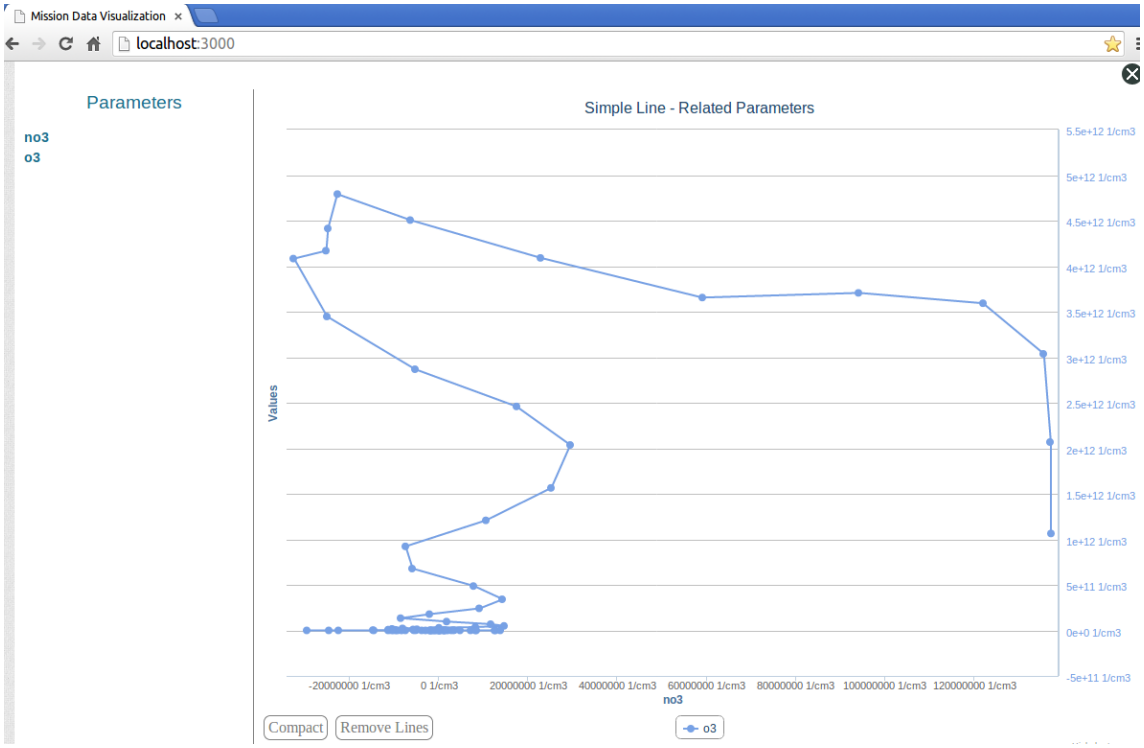
(in this case it is the same, "1/cm^3"), if they haven't the same data units the graphic will create another axis y. To do this, the user need to select the "Compact" button in the graphic window and then add the parameters.



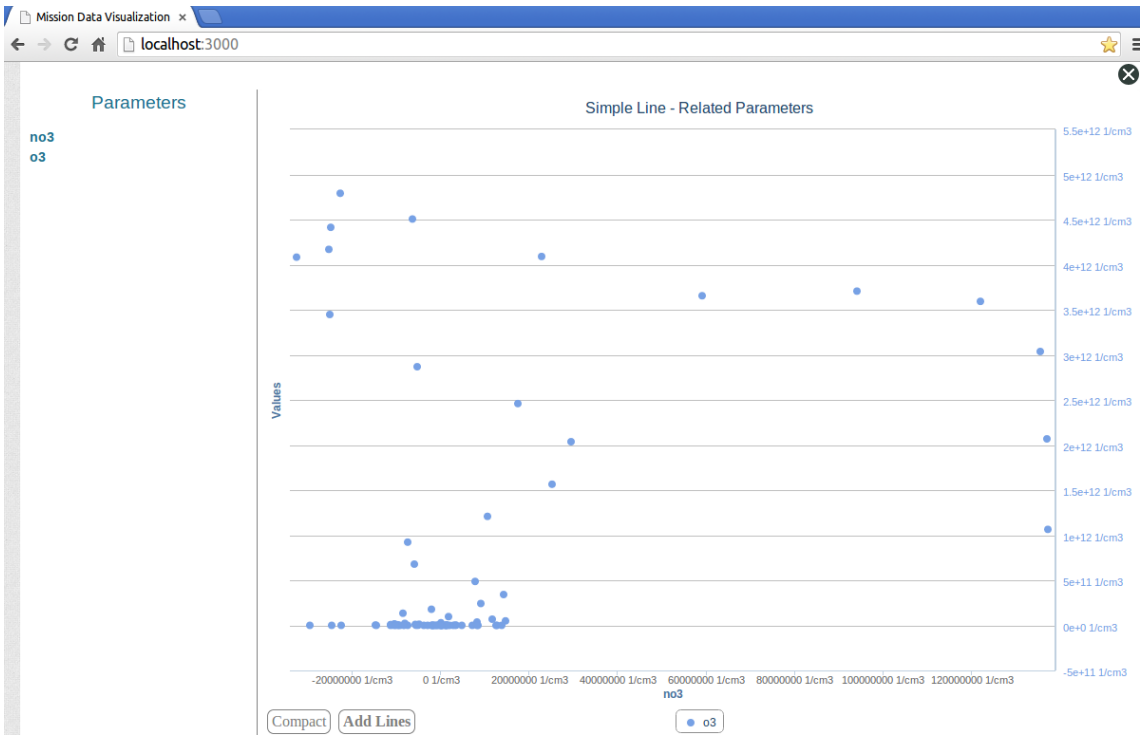
The figure above represent the "Compact" visualization type, there are only one y axis that is shared by the two variables "no3" and "o3". The time is represent in the x axis.

The user can fix one variable to the x axis and relate this variable against another one. To do this the user needs to right click the variable he wants to put in the x axis and then click on "Add to x axis". To place the other variable on the y axis, doing this, the user is relating the variables, he needs to right click the parameter and then "add to y axis".





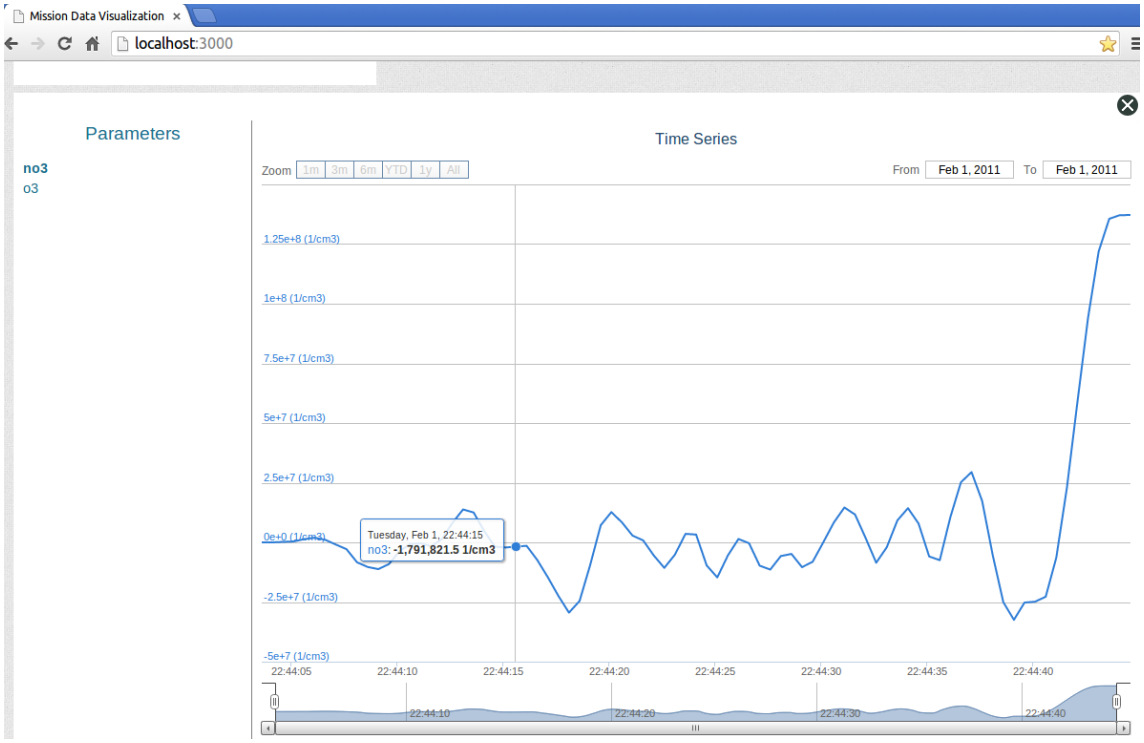
In the figure above can be seen the representation of the relation between the values of "no3" (plotted on the x axis) and the "o3" (plotted in the y axis). This visualization is shown with lines connecting the data, but it is possible to remove the lines, creating a scatter plot. In order to make this visualization, the user only needs to click the "Remove lines" on the graphic window, this action can be seen in the image bellow.



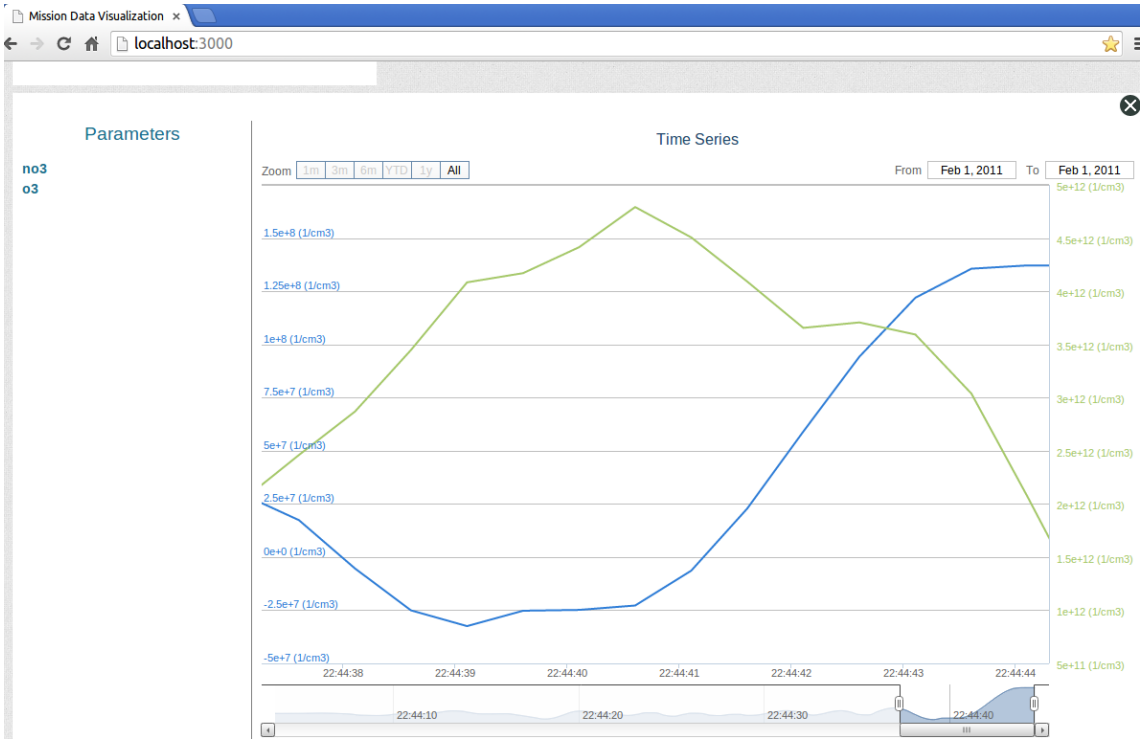
The user can zoom in the graphic as well. If the user wants to occult one variable it is possible by clicking in the button that says the variable name in the bottom of the graphic window. It can be added other variables as well.

Time Series Visualization

The users can select the specialized graphic for time series visualization, to do this, the user needs to choose the "Time Series" graphic type shown in the list of plots available. To visually represent the data, the procedures are the same as the simple line graphics.

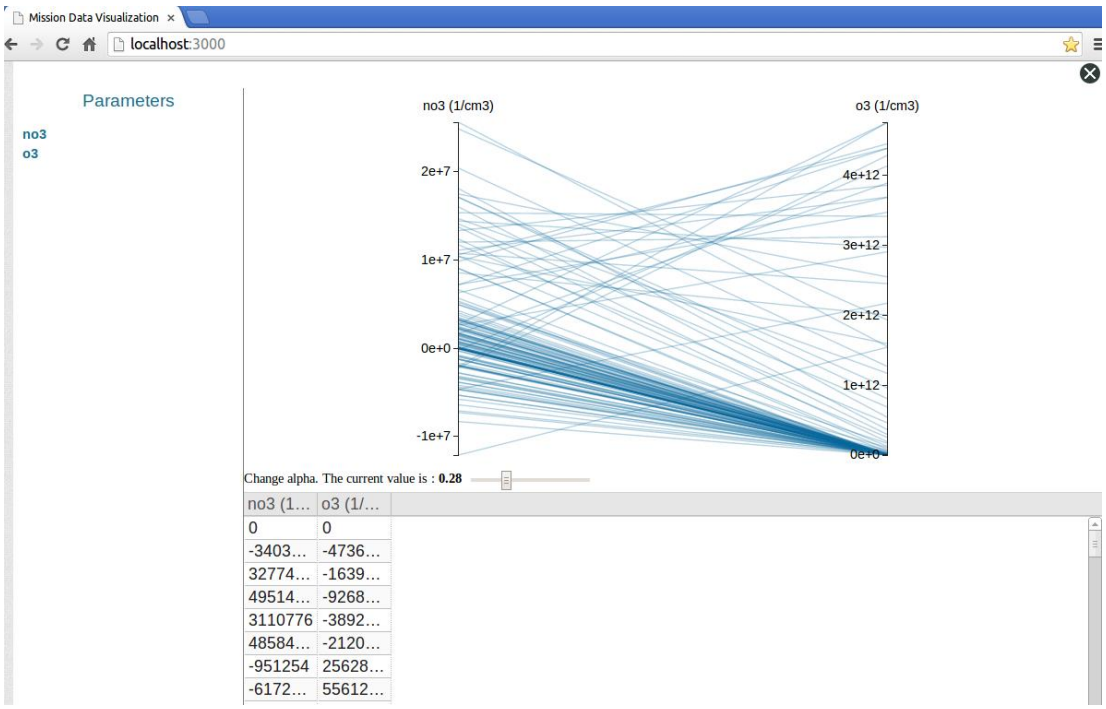


This kind of representation have one specific window that can be used to make borders with the initial time and final time (can be seen in the bottom of the image bellow) and select the date on the date picker.

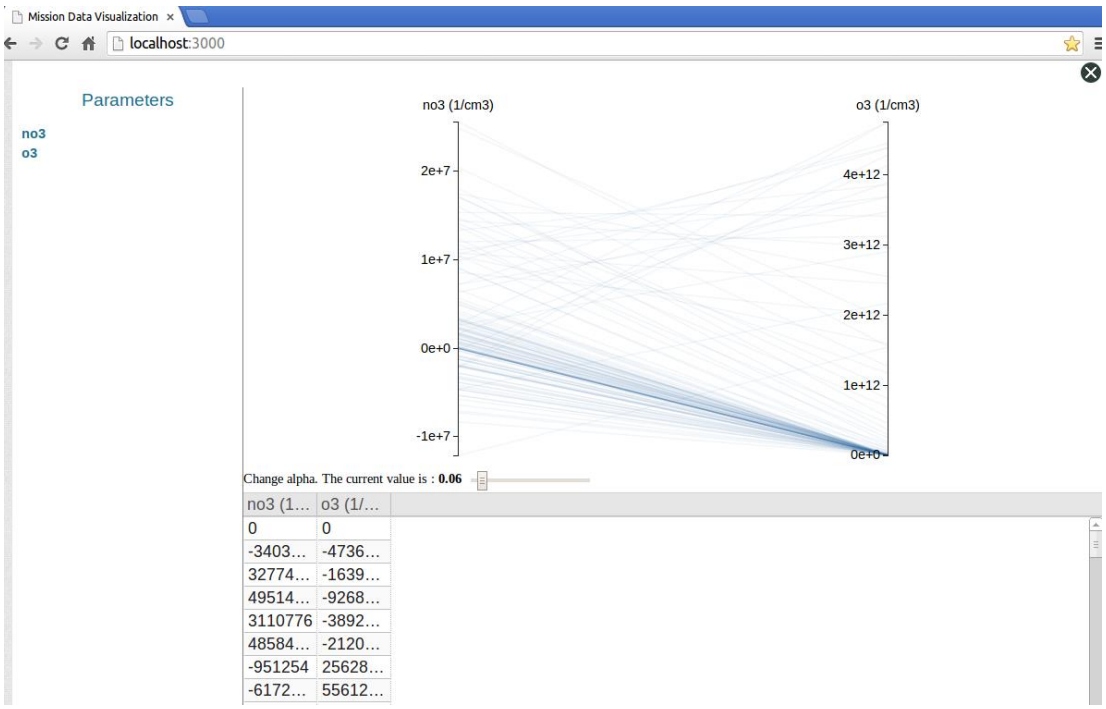


Parallel Coordinates

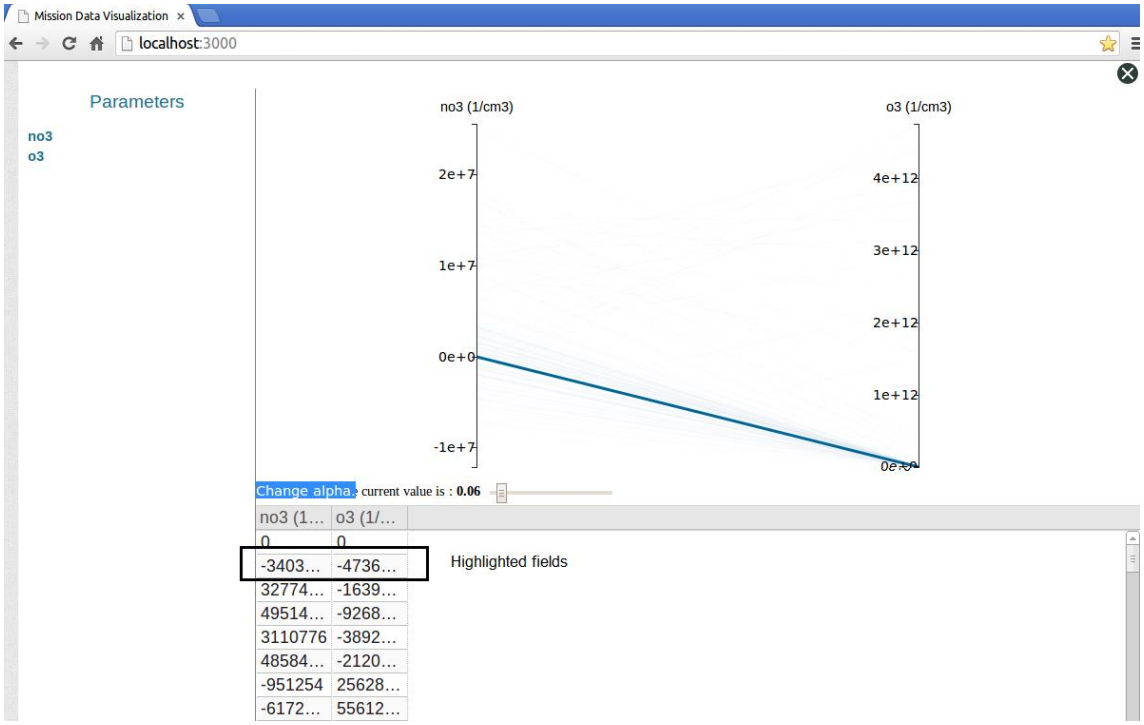
The parallel coordinates is a type of graphic dedicated to multi variable representation. This way the user can select the "Parallel Coordinates" graphic type to perform this type of visualization. Again it will be shown on the left all the variables, this way the user needs to click on each parameter to plot it on the graphic.



In the above figure is shown the relations between the "no3" and "o3" variables. The technique called alpha blending is implemented so the user can change the transparency of all the lines present in the visualization plot. To do this, the user needs to change the value in the "Change Alpha" bar. Also is shown below the list of relations, if the user put the mouse over one relation it is highlight in the graphic.



In the figure above is shown the change in the alpha (alpha blending technique) and it is possible to see the places where density of lines is bigger.

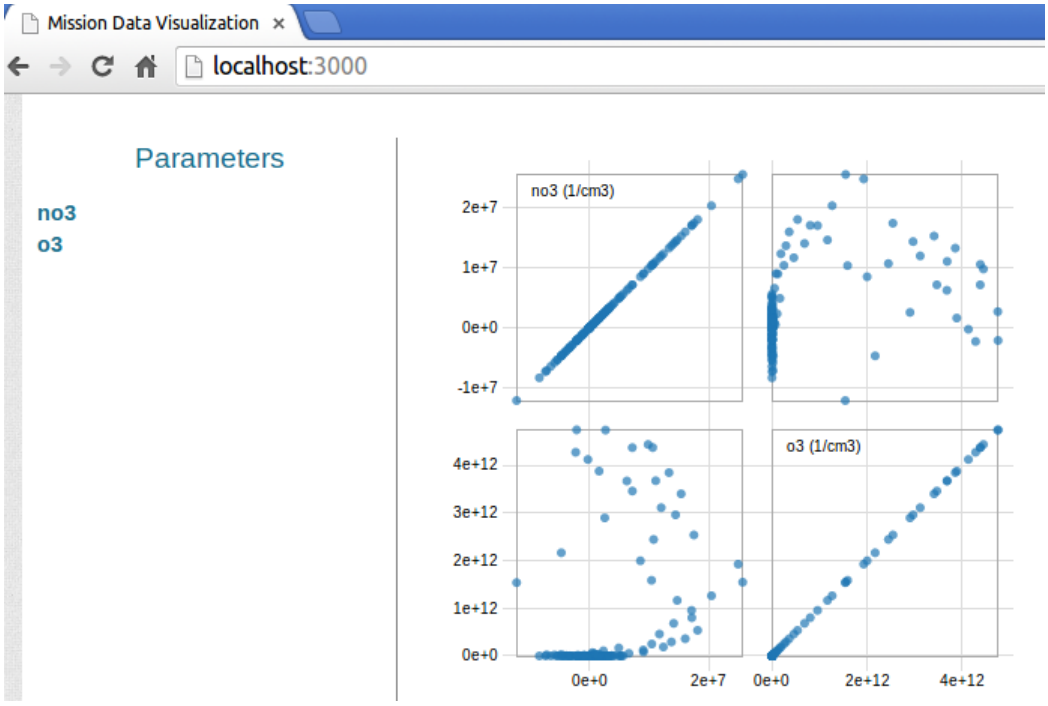


no3 (1…	o3 (1/…
0	0
-3403…	-4736…
32774…	-1639…
49514…	-9268…
3110776	-3892…
48584…	-2120…
-951254	25628…
-6172…	55612…

Above it is possible to see the highlight of the value where the mouse pointer is located (in the table of relations).

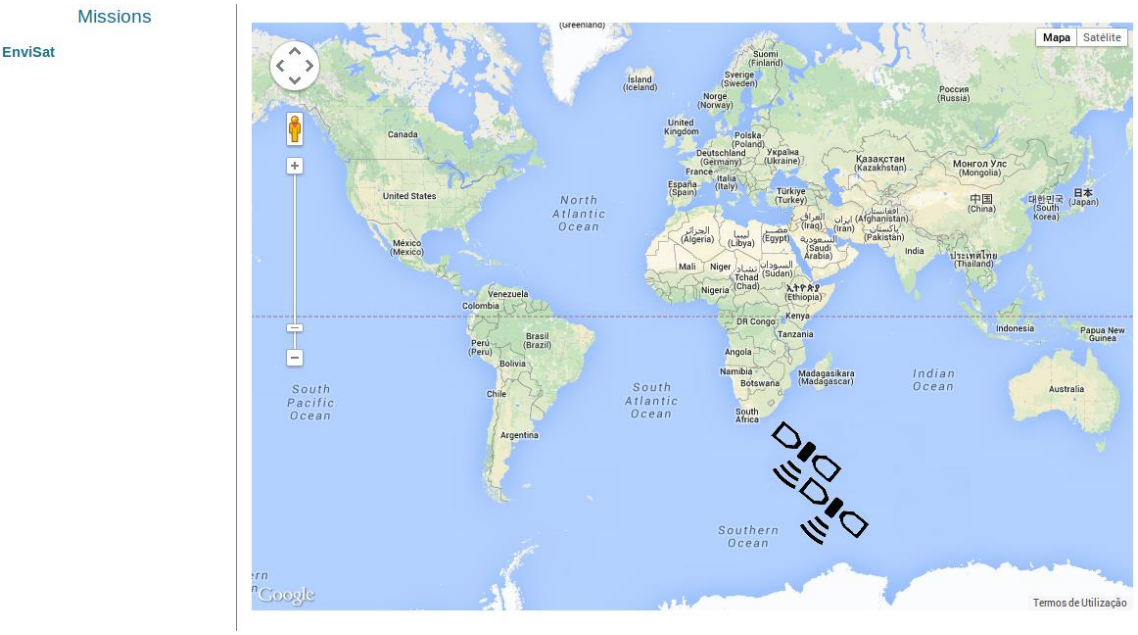
SPLOMS

The scatterplot matrices is available in order to visualize multidimensional data. This way if the user clicks in the "Scatter Plot Matrix" in order to perform this type of representation. In the left is shown all the parameters and by clicking on it, the parameters are compared and shown as scatter plots.



Satellite Position

To be able to see the satellite orbit, the user select the "Satellite Position" and the mission, this way the orbit is shown in a world map.



Apêndice D

Casos de Uso – "Web-Service"

Nesta secção vai ser apresentado o caso de uso geral do "*Web-Service*".

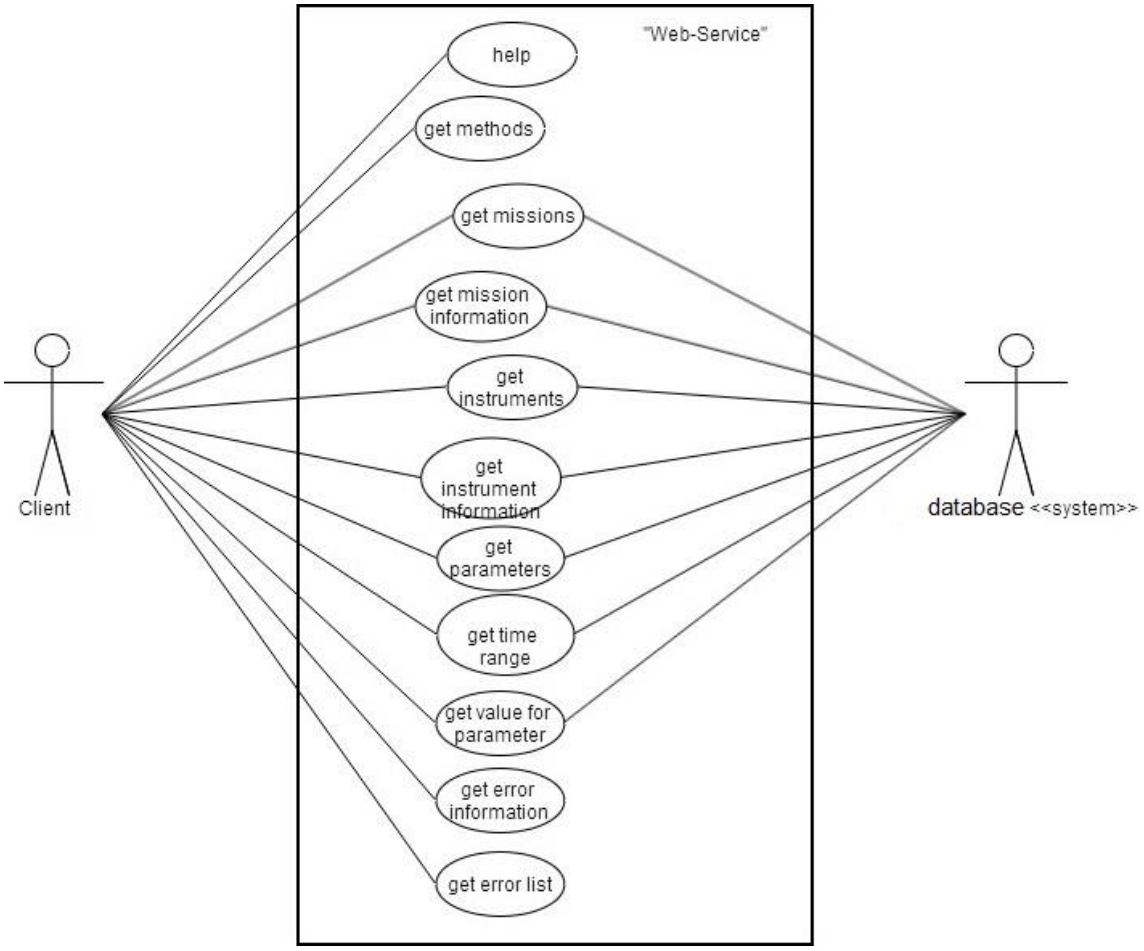


Figura 1 – Casos de uso relativos ao "*Web-Service*".

Explicam-se seguidamente, os casos de usos relativos ao serviço *web*, assim como, a identificação dos atores.

- Identificação de todos os atores:
    - *Client* – Pessoa ou sistema que utiliza o serviço *web*;
    - *Database* – Base de dados com a qual o serviço *web* comunica.
- Caso de uso – *help*:
    - Atores – Client;
    - Descrição – Este caso de uso permite que o utilizador (*Client*) receba em *html* todas as informações de ajuda relativas a este serviço *web*. Para isto, este deve realizar um pedido ao serviço de *help*.

- Caso de uso – *get Methods*:
    - Atores – *Client*;
    - Descrição – Caso que uso que torna possível o envio, em formato *JSON*, de todos os métodos disponíveis na API do serviço *web*. Desta forma, o cliente (*Client*) deve fazer um pedido ao serviço de *getMethods*.
- Caso de uso – *get Missions*:
    - Atores – *Client* e *database*;
    - Descrição - Este caso de uso tem o propósito de fornecer ao utilizador (*Client*) todas as missões existentes na base de dados. Para que isto seja possível o cliente tem de efetuar um pedido *getMissions* ao serviço *web*, este serviço vai consultar a base de dados (*database*), formatar a resposta em JSON e por fim enviar este resultado ao cliente.
- Caso de uso – *get Mission Information*:
    - Atores – *Client* e *database*;
    - Descrição - Este possui o propósito de fornecer ao utilizador (*Client*) toda a informação relativa à missão requisitada, para isto, o utilizador tem de enviar um pedido ao "*Web-Service*" de *getMissionInformation* acompanhado do nome da missão. O serviço *web* realiza uma consulta à base de dados, formata a resposta em JSON e envia de novo para o utilizador.
- Caso de uso – *get Instruments*:
    - Atores – *Client* e *database*;
    - Descrição – Caso de uso que torna possível ao utilizador (*Client*) receber todos os instrumentos relativos a uma missão. Desta forma, o utilizador necessita de efetuar um pedido *getInstruments* ao "*Web-Service*" acompanhado do nome da missão. De seguida, o serviço web consulta a base de dados, realiza a formatação da resposta em *JSON* e envia para o utilizador.
- Caso de uso – *get Instrument Information*:
    - Atores – *Client* e *database*;
    - Descrição – Este torna possível ao utilizador (*Client*) receber todos a informação de um instrumento relativo a uma missão. Desta forma, o utilizador necessita de efetuar um pedido *getInstrumentInformation* ao "*Web-Service*" acompanhado do nome da missão e do nome do instrumento. De seguida, o serviço web consulta a base de dados, realiza a formatação da resposta em *JSON* e envia para o utilizador.
- Caso de uso – *get Parameters*:
    - Atores – *Client* e *database*;
    - Descrição – Caso de uso que torna possível ao utilizador (*Client*) receber todos os parâmetros existentes nos instrumentos relativos a uma missão. Desta forma, o utilizador necessita de efetuar um pedido *getParameters* ao

"*Web-Service*" acompanhado do nome da missão e do nome do instrumento. De seguida, o serviço web consulta a base de dados, realiza a formatação da resposta em *JSON* e envia para o utilizador.

- Caso de uso – *get Time Range*:
    - Atores – *Client* e *database*;
    - Descrição – Caso de uso que torna possível ao utilizador (*Client*) receber todos os intervalos tempo relativos às medições existentes na base de dados. Desta forma, o utilizador necessita de efetuar um pedido *getTimeRange* ao "*Web-Service*" acompanhado do nome da missão e do nome do instrumento. De seguida, o serviço web consulta a base de dados, realiza a formatação da resposta em *JSON* e envia para o utilizador.
- Caso de uso – *get Value for Parameter*:
    - Atores – *Client* e *database*;
    - Descrição – Caso de uso que torna possível ao utilizador (*Client*) receber todos valores de um parâmetro relativos à missão, instrumento, domínio e uma janela temporal (tempo inicial e tempo final). Para isto o utilizador necessita de efetuar um pedido *getValuesParameter* ao "*Web-Service*" acompanhado do nome da missão, nome do instrumento, nome do domínio, nome do parâmetro, tempo inicial e tempo final. De seguida, o serviço web consulta a base de dados, realiza a formatação da resposta em *JSON* e envia para o utilizador.
- Caso de uso – *get error information*:
    - Atores – *Client*;
    - Descrição – Torna possível ao utilizador (*Client*) receber a informação a respeito de um certo tipo de erro que ocorreu. Para isto o cliente necessita de efetuar um pedido *getErrorInformation* com o número do erro. Desta forma, o serviço web realiza a formatação da resposta em *JSON* e envia para o utilizador.
- Caso de uso – *get error list*:
    - Atores – *Client*;
    - Descrição – Caso de uso que torna possível ao utilizador (*Client*) receber a toda a informação relativa às validações (erros) que são suportados pelo "*Web-Service*". Para isto o cliente necessita de efetuar um pedido *getErrorList* e o serviço web realiza a formatação da resposta em *JSON* e envia para o utilizador.

Casos de Uso – Aplicação WEB - MDV

Nesta secção é apresentado o caso de uso geral da aplicação *web* MDV.

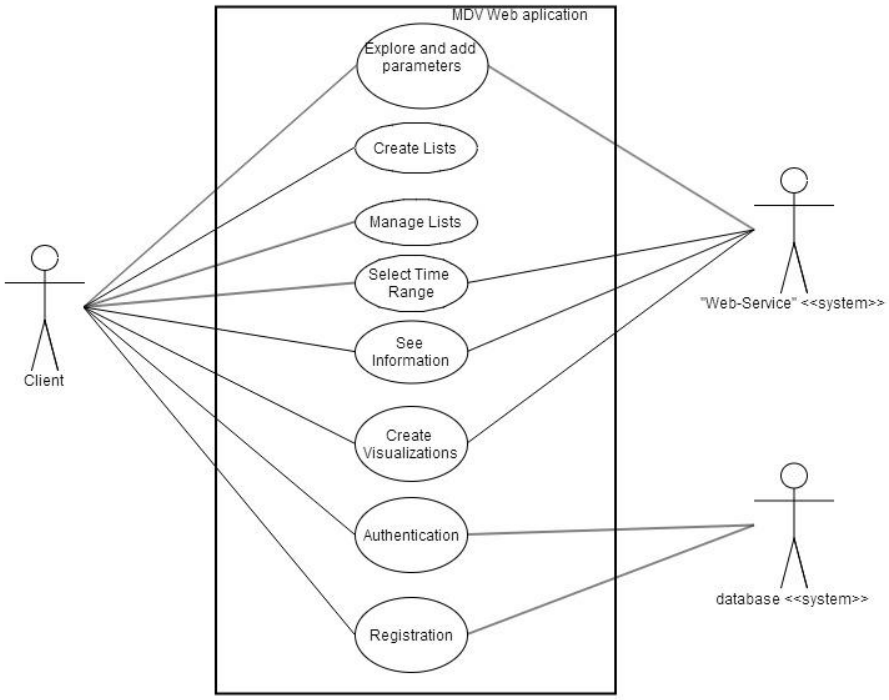


Figura 2 – Casos de uso relativos à aplicação *web*.

Seguidamente vão ser explicados os casos de uso e dos atores da aplicação *web* MDV que pode ser visto na figura 2.

- Identificação de todos os atores:
    - *Client* – Pessoa que utiliza a aplicação *web* MDV;
    - *Database* – Base de dados da aplicação *web*.
    - *"Web*-Service" – Serviço *web* que fornece acesso às informações existentes.
- Caso de uso – *Explore and add parameters*:
    - Atores – *Client* e *"Web-Service"*;
    - Descrição – Caso de uso que fornece ao utilizador (*Client*) um método de visualização interativo baseado em grafos que permite a exploração das estruturas de dados para que este tenha a possibilidade de selecionar um parâmetro. Para que isto seja possível a aplicação *web* necessita de realizar pedidos ao "*Web-Service*" com a finalidade de receber todas as missões, instrumentos, domínios e parâmetros existentes.
- Caso de uso – *Create Lists*:
    - Atores – *Client*;

  - Descrição – Caso de uso que permite ao utilizador (*Client*) criar várias listas. Através destas listas torna-se possível, ao utilizador organizar os parâmetros selecionados.

- Caso de uso – *Manage List*:
  - Atores – *Client*;
  - Descrição – Este caso de uso permite ao utilizador (*Client*) gerir as várias listas existentes, podendo este mudar o seu nome e proceder à sua eliminação.

- Caso de uso – *Select Time Range*:
  - Atores – *Client* e *"Web-Service"*;
  - Descrição – Caso de uso que fornece ao utilizador (*Client*) uma forma, visual e interativa, de selecionar um intervalo temporal para uma lista existente. Para que isto seja possível, a aplicação *web* realiza vários pedidos ao "*Web-Service*" para fornecer os dados relativos às janelas temporais existentes.

- Caso de uso – *See Information*:
  - Atores – *Client* e *"Web-Service"*;
  - Descrição – Este caso de uso possibilita fornecer informações relativas às missões e aos instrumentos que existem no "*Web-Service*". Desta forma, o utilizador (*Client*) seleciona a missão e instrumento e a aplicação web apresenta em formato de texto a informação. Para que isto seja possível a aplicação *web* realiza vários pedidos de informação ao "*Web-Service*".

- Caso de uso – *Create Visualization*:
  - Atores – *Client* e *"Web-Service"*;
  - Descrição – Este caso de uso possibilita ao cliente (*Client*) a criação de vários tipos de representações gráficas para os dados que foram selecionados. Para isto, são mostrados ao utilizador os vários tipos de gráficos suportados pela aplicação *web*, o utilizador necessita de selecionar o que está interessado e posteriormente escolher os parâmetros para a representação gráfica. Para que isto seja possível, a aplicação *web* realiza pedidos ao "*Web-Service*" para receber os dados dos parâmetros.

- Caso de uso – *Authentication*:
  - Atores – *Client* e *database*;
  - Descrição – Este caso de uso permite ao utilizador realizar a sua autenticação, para que isto seja possível, o utilizador necessita de fornecer o seu *email* e a sua password. A aplicação web consulta a base de dados (*database*) e faz a validação dos dados inseridos, se estiverem corretos a autenticação é realizada com sucesso, caso contrário, não é feita a autenticação.

- Caso de uso – *Registration*:

- Atores – *Client* e *database;*
- Descrição – Este caso de uso permite ao utilizador (*Client*) fazer o registo na aplicação *web* MDV. Para que isto seja possível é necessário que este introduza um *email* e uma *password*. Seguidamente, a aplicação *web* vai fazer verificações na base de dados (*database*) e se não existir nenhum utilizador com as mesmas credenciais, então o registo é efetuado com sucesso.

Apêndice E