Desenvolvimento de ferramentas de software para suporte a operações de investigação marinha

Miguel Ribeiro Correia

Dissertação de Mestrado apresentada à
Faculdade de Ciências da Universidade do Porto em
Área Científica

2020

Desenvolvimento de ferramentas de software para suporte a operações de investigação marinha

Miguel Ribeiro Correia
Engenharia de Redes e Sistemas Informáticos
Ciência de Computadores
2020

Orientador
José Paulo Leal, Professor
Faculdade de Ciências da Universidade do Porto

Supervisor
Carlos Almeida, Investigador
INESC TEC

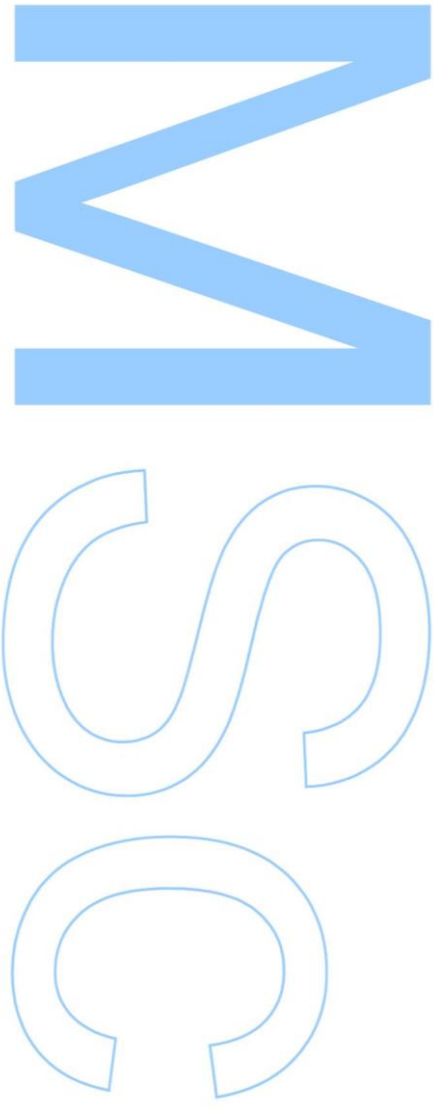
U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____



Abstract

The water that composes all the oceans is not static, it is moved with different intensities, giving origin marine currents. These currents alter the chemical and physical state of the water, where they manipulate the climate on the mainland in a direct and accelerated way. The project Multi-purpose/Multi-sensor Extra Light Oceanography Apparatus (MELOA) has a purpose to develop an innovative family of products WAVY drifters. The main features of these drifters are low-cost manufacturing, easily deployable, high versatility and low maintenance, for in-situ measurements and analyses of marine environments. Were developed tree models of WAVY drifters: basic, littoral and ocean; with different sizes, technologies and resistances, to adapt in a better way on the environment and all have multiples sensors: temperature, pressure, speed, GPS, etc.

This work will focus on the software (WOS) to support the planning, execution and post-processing of scientific campaigns for the study of sea currents, using the WAVY drifters, which transmit in real-time the information collected by it is sensors. Throughout this work, carried out at INESC TEC, were integrated into this software, tools taht allow the analysis, visualization and curation of information collected by the drifters.

However, during the MELOA project, the need arose to integrate log files in WOS, so that later it was possible to calculate Wave Spectrum parameters, which are fundamental for the study of marine currents. These files increased around thirty times the amount of information stored, in a relational database PostgreSQL, which generated many challenges. The challenges were overcome with the integration of a time-series database, InfluxDB, and the integration of a process scheduler so that more dense tasks are performed in another service.

Resumo

A água que compõe todos os oceanos não é estática, toda ela é movimentada com diferentes intensidades, dando origem a correntes marinhas. Estas correntes alteram o estado químico e físico da massa da água, onde de forma direta e acelerada manipulam o clima nos continentes. O projeto *Multi-purpose/Multi-sensor Extra Light Oceanography Apparatus* (MELOA) tem como propósito o desenvolvimento de uma família inovadora de derivadores flutuantes *WAVY*. As principais características destes derivadores são o baixo custo de produção, a facilidade de operação, a grande versatilidade e reduzida manutenção, para operações de medição e análise de correntes marinhas. Foram desenvolvidos três modelos de derivadores flutuantes: básico, litoral e oceânico; com tamanhos, tecnologias e resistências diferentes, para se adaptarem da melhor forma ao seu meio e todos eles incorporam uma série de sensores: temperatura, pressão, velocidade, GPS, etc.

Este trabalho irá incidir sobre o *software* (WOS) de apoio ao planeamento, execução e pós-processamento de campanhas cientificas para o estudo das correntes marinhas, utilizando os derivadores flutuantes *WAVY*, que transmitem em tempo real a informação recolhida pelos seus sensores. Ao longo deste trabalho, realizado no INESC TEC, foram integradas nesse *software* ferramentas que permitem a análise, visualização e curadoria da informação recolhida pelos derivadores.

Contudo, no decorrer do projeto MELOA surgiu a necessidade de integrar ficheiros *Log* no WOS, para que posteriormente fosse possível o calculo de parâmetros de Espectro de Onda, fundamentais para o estudo de correntes marinhas. Estes ficheiros aumentaram cerca de trinta vezes a quantidade de informação armazenada, numa base de dados relacional *PostgreSQL*, o que gerou diversos desafios. Desafios estes, que foram superados com a integração de uma base de dados de série temporais, *InfluxDB*, e a integração de um escalonador de processos, para que tarefas mais densas sejam realizadas noutro serviço.

Conteúdo

Lista de Tabelas

Lista de Figuras

Acrónimos

ADCP *Acoustic Doppler Current Profiler*

AMQP *Advanced Message Queuing Protocol*

API *Application Programming Interface*

ARGOS *Advanced Research and Global Observation Satellite*

ASCII *American Standard Code for Information Interchange*

AUV *Autonomous Underwater Vehicle*

CARTHE *Consortium for Advanced on Transport of Hydrocarbon in the Environment*

CLS *Collecte Localisation Satellites*

CRUD *Create, Read, Update and Delete*

CSV *Comma-Separated Values*

GPRS *General Packet Radio Service*

GPS *Global Positioning System*

GSM *Global System for Mobile*

HDOP *Horizontal Dilution Of Precision*

HTML *HyperText Markup Language*

HTTP *Hypertext Transfer Protocol*

IBM *International Business Machines*

IH Instituto Hidrográfico

IMU *Inertial Measurement Unit*

INESC TEC Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência

IOT *Internet of Things*

ISR Instituto de Sistemas e Robótica

JIT *Just In Time*

JSON *JavaScript Object Notation*

JSX *JavaScript XML*

LSTS Laboratório de Sistemas e Tecnologia Subaquática

MELOA *Multi-purpose/Multi-sensor Extra Light Oceanography Apparatus*

NoSQL *Not Only SQL*

PM2 *Process Manager 2*

OS *Operating System*

SBI	*SmartBay Ireland*		*Catalunya*
SONAR	*Sound Navigation And Ranging*	**UUID**	*Universally Unique Identifier*
SQL	*Structured Query Language*	**VM**	*Virtual Machine*
REST	*Representational State Transfer*	**WOS**	*WAVY Operation Software*
UPC	*Universitat Politècnica de*	**XML**	*Extensible Markup Language*

Capítulo 1

Introdução

O oceano é o coração do planeta, onde cobre mais de dois terços da superfície da Terra, sendo que cerca de 90% é praticamente inexplorado. É quem fornece, entre 70% a 80%, o oxigénio que respiramos [21], regula o clima, oferece segurança alimentar à população mundial e as suas profundezas abrigam vida selvagem, onde habitam algumas das maiores criaturas do planeta. No entanto, várias ameaças comprometem a sua saúde e sustentabilidade.

1.1 Contextualização

A característica mais impressionante do planeta Terra, e deveras importante, quando vista do espaço, é a imensa quantidade de água que cobre o nosso planeta. A sua maioria pertence aos oceanos e uma pequena percentagem aos lagos e rios. A interação entre a água e o nosso planeta, tanto através da superfície da Terra como da atmosfera, origina as nuvens, chuva, neve e correntes. [37]

A água que compõe todos os oceanos não é estática, isto é, toda a água é movimentada com diferentes intensidades, o que dá origem a ondas, marés e correntes marinhas. Tanto as ondas como as marés são meramente movimentos oscilatórios, onde de modo nenhum influenciam a distribuição de temperatura e salinidade, enquanto que as correntes marinhas formam a verdadeira circulação, que altera o estado químico e físico da massa de água e de forma direta manipula o clima nos continentes. [16]

Os movimentos em águas mais profundas também acontecem, porém de forma muito lenta e com tendência a uniformizar as condições térmicas e de salinidade. Nas águas mais superficiais, a movimentação é muito mais acelerada e com uma discrepância maior, o que requer a nossa atenção, sendo este ponto o principal foco do projeto MELOA.

MELOA - "Aparelho de Oceanografia Multifuncional / Multisensor Extraleve" - é um projeto com o propósito de criar uma família inovadora de derivadores flutuantes *WAVY*, de baixo custo, com grande facilidade de implementação e manuseamento, versáteis e de reduzida manutenção, para medições e análise de ambientes marinhos, em qualquer parte do mundo. [40]

Figura 1.1: Protótipos de *WAVY Littoral*

Existem diversas formas e especialidades para o estudo dos oceanos. Porém, neste projeto o principal foco é o estudo e monitorização das correntes marinhas, onde este estudo está diretamente relacionado com o clima do planeta, mas também com todos os ecossistemas existentes.

Entre os dois hemisférios, as correntes marinhas circulam em diferentes sentidos: horário no norte e anti-horário no sul, como se pode confirmar na figura 1.2. Esta direção, é tão importante para o clima, como também para o equilíbrio dos ecossistemas presentes no oceano e atividades relacionadas com a pesca, pois define a deslocação de várias espécies de animais marinhos.

1.1.1 Correntes marinhas

As ondas e marés, por vezes com efeitos fascinantes, simplesmente fazem a deslocação de águas ao longo de pequenas distâncias. Os oceanos, estão continuadamente em movimento e têm capacidades de carregar fragmentos para muito longe da sua origem, isto é chamado de correntes marinhas ou correntes oceânicas 1.2.

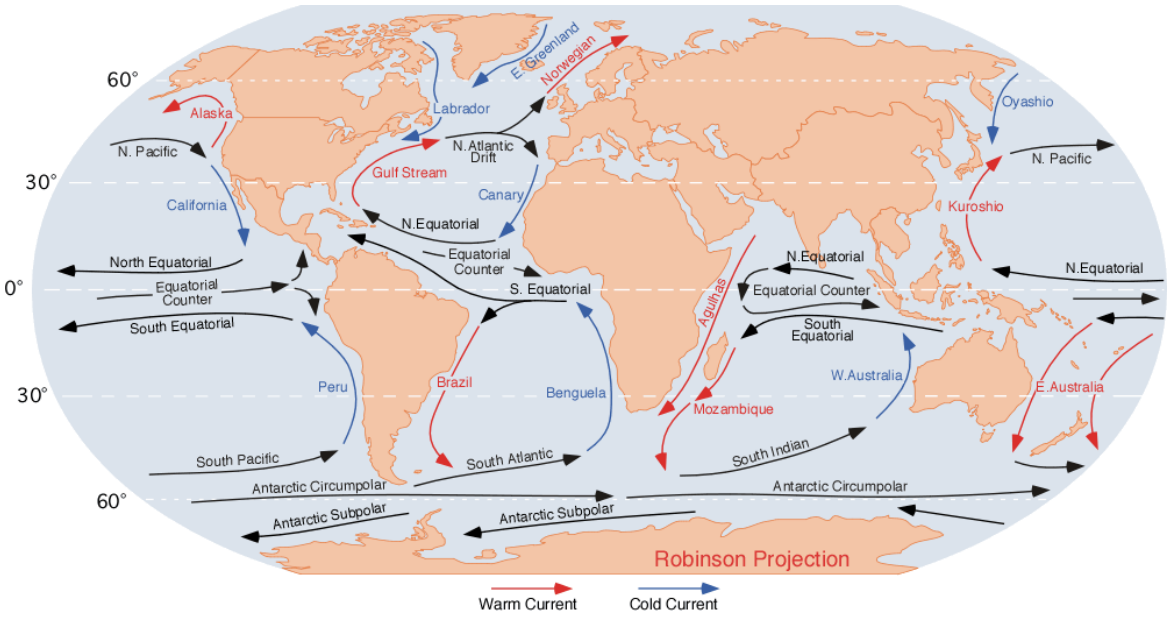


Figura 1.2: Correntes oceânicas [15]

O seu estudo enquadra-se perfeitamente no domínio da oceanografia e está diretamente ligado ao das correntes atmosféricas. Uma corrente marinha é caracterizada, pela sua baixa velocidade (geralmente inferior a 2 nós - 3,7km/h), salinidade, temperatura e direção, sendo que este último tem uma relação direta com a direção do vento, que por sua vez se enquadra no estudo das correntes atmosféricas. [2]

Estas correntes atmosféricas, além de responsáveis pela dinâmica de circulação dos ventos, também o são pela alteração das pressões atmosféricas, pela distribuição do calor e, consequentemente, interferem diretamente no clima da Terra.

1.2 Motivação

Os oceanos controlam o clima da Terra, pois quando aquecem e arrefecem, humidificam e secam o ar. Os padrões climáticos de longo prazo, não só influenciam o suprimento de água e alimentos, como também promovem o crescimento ou desaparecimento de populações. Uma pessoa não consegue escapar nem alterar o clima, mas pode ser capaz de prever o seu impacto e tentar de alguma forma o controlar.

Apesar do oceano desempenhar um papel importante em tudo, desde o ar que respiramos até aos padrões climáticos e meteorológicos, sabemos muito pouco sobre ele. A maior parte do nosso conhecimento sobre oceano está em águas rasas e as águas mais profundas continuam a ser um mistério.

O projeto MELOA espera ajudar não só na resolução de muitos problemas, mas também intensificar e melhorar o estudo neste âmbito. O projeto conta com uma plataforma *Web* (secção 4) de ajuda a lançamentos de derivadores flutuantes *WAVY*. Esta plataforma apenas cobre operações de planeamento e execução de campanhas, onde se pode seguir os derivadores geograficamente através de um mapa.

1.3 Objetivos

O objetivo principal desta dissertação é dotar a plataforma, acima referida, de capacidades para apoiar operações de pós-processamento dos dados recolhidos pelos derivadores flutuantes.

De acordo com os requisitos recolhidos junto dos investigadores e para atingir este objetivo, pretende-se **desenvolver e integrar ferramentas que permitam análise, visualização e curadoria da informação** recolhida pelos derivadores flutuantes *WAVY*, com o intuito de facilitar a exploração dos dados ao longo do tempo. Pretende-se que estas ferramentas sejam um elemento de apoio aos investigadores, que utilizam a plataforma, auxiliando-os em tarefas de curadoria de dados, como **identificação e limpeza de valores atípicos**, gerando novos produtos, mantendo sempre a integridade da informação original.

Pretende-se ainda que a plataforma tenha capacidade de **calcular parâmetros de Espectro de Onda**, essencial no estudo das correntes marinhas, sem necessidade de recorrer a ferramentas externas. Assim como disponibilizar esses mesmos valores na plataforma, integrando-os nas ferramentas de visualização.

1.4 Abordagem

De modo a facilitar a analise dos parâmetros recolhidos pelos derivadores flutuantes *WAVY* ao longo do tempo, irá ser desenvolvida uma **ferramenta baseada em gráficos**. Esta irá complementar as ferramentas de visualização existentes e disponibilizará uma alternativa à identificação de valores atípicos, designados *outliers*.

De maneira a auxiliar os investigadores no processo de curadoria de dados, previamente recolhidos e automaticamente catalogados, irá ser desenvolvida uma **ferramenta que permitirá realizar as tarefas de identificação e remoção de *outliers***. Este trabalho irá produzir um conjunto de dados tratados que poderão ser disponibilizados em forma de produtos de interesse para a comunidade cientifica. Serão integradas as ferramentas de visualização, onde se inclui a ferramenta prevista anteriormente, de forma a permitir a **exploração dos dados tratados** para validação do trabalho de curadoria.

Os dados transmitidos em tempo real pelos derivadores, não são suficientes para o cálculo dos parâmetros de Espectro de Onda. No entanto, os derivadores recolhem e guardam esses dados na sua memória interna, que pode ser extraída sobre a forma de ficheiros *Log*. É necessário dotar a plataforma de capacidades para absorver e tratar esses valores, de modo a ser possível o **cálculo e disponibilização desses parâmetros**. Para isso será necessário **converter os *scripts***, criados e utilizados pelos investigadores para o cálculo desses parâmetros, numa linguagem compatível com a plataforma.

1.5 Estrutura

Com o intuito de alcançar os objetivos apresentados, este trabalho está organizado em capítulos, da seguinte forma:

- Projeto MELOA: Este capítulo detalha a estrutura do projeto MELOA. Aborda todos as parcerias, bem como a divisão das tarefas por cada uma delas. Também explica o funcionamento dos derivadores flutuantes *WAVY*.

- Estado da Arte: Apresenta alternativas ao projeto MELOA, bem como as vantagens de se optar pelos derivadores flutuantes *WAVY*.

- *WAVY Operation Software* (WOS): O nome do atribuído à plataforma de apoio a lançamentos de derivadores flutuantes *WAVY*, referida na Introdução, é *WAVY Operation Software*. Este capítulo descreve o funcionamento de toda a plataforma, antes do início deste trabalho, explicando tecnologias usadas, a sua estrutura e a comunicação dos derivadores flutuantes com a mesma.

- Novas ferramentas do WOS: Neste capítulo é detalhado todo o desenvolvimento na aplicação *Web* do WOS, referente ao primeiro ponto dos objetivos.

- Parâmetros de Espectro de Onda: Este capítulo descreve as funcionalidades implementadas, para posteriormente ser possível integrar na plataforma WOS o calculo de parâmetros de Espectro de Onda.

- Validação inicial: Explica a primeira validação deste trabalho, bem como a metodologia usada para tal. No final são definidos novos objetivos, para superar os desafios encontrados, que serão abordados nos capítulos seguintes.

- Infraestrutura tecnológica: É descrito todo o trabalho de estudo sobre a forma em como se abordaram os desafios apresentados no capítulo anterior.

- Contextualização de dados: Este capítulo descreve a implementação das tecnologias apresentadas no capítulo anterior, de modo a alcançar os novos objetivos.

- Validação e conclusão: De modo a verificar se o trabalho desenvolvido foi bem sucedido, foi realizada uma segunda validação. Este capítulo, além da validação final, apresenta uma resposta que atinja todos os objetivos propostos e faz uma análise geral do trabalho como um todo e de seu possível futuro.

Capítulo 2

Projeto MELOA

Este projeto é financiado pelo programa da União Europeia Horizonte 2020, que começou em dezembro de 2017 e o seu término está previsto ser em fevereiro de 2021. Coordenado pela *Deimos Space*, o MELOA tem como objetivo desenvolver uma solução para aplicar *WAVYs* em diferentes ambientes marítimos, desde o mar profundo a zonas mais costeiras.

Para este projeto, o INESC TEC contribui com o desenvolvimento do software, *WAVY Operation Software* (WOS), para gestão de todo o processo de campanhas, tratamento de dados e integração no catálogo de dados que a *Deimos Space* irá disponibilizar, aumentando os serviços relevantes do programa *Copernicus* (programa europeu para a criação de uma capacidade europeia de observação da Terra).

2.1 Consórcio

O projeto MELOA é uma parceria do Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência (INESC TEC); da *Deimos Space* e *Deimos Engenharia*, empresa de tecnologia do grupo *Elecnor*; do Laboratório de Sistemas e Tecnologia Subaquática (LSTS) da Faculdade de Engenharia da Universidade do Porto; do Instituto Hidrográfico (IH); do Instituto de Sistemas e Robótica (ISR); da *Ocean Scan*, *spin-off* da LSTS; da *Composite Solutions*; da *Collecte Localisation Satellites* (CLS) que gere a rede de satélites ARGOS, da *Universitat Politècnica de Catalunya* (UPC) e da *SmartBay Ireland* (SBI).

2.2 *WAVYs*

Os derivadores flutuantes *WAVY* respeitam a seguinte ideologia: uma esfera de tamanho pequeno com espaço suficiente para acomodar uma fonte de energia, receptor satélite, módulos de comunicação, sensores e processador de dados; flutuabilidade otimizada, para impedir que a trajetória do derivador responda ao vento em vez da corrente, fornecendo exposição suficiente das antenas para garantir a melhor aquisição do sinal satélite na taxa necessária e com comunicações confiáveis em tempo real. [38]

Dada a baixa influência do vento na deslocação dos derivadores flutuantes *WAVY*, o MELOA proporcionará uma maneira barata e eficaz de monitorizar as correntes e os recursos dinâmicos da superfície em qualquer ambiente marinho no mundo. Ao equipar os derivadores com sensores de temperatura em dois níveis diferentes, está aberta a possibilidade de monitorizar gradientes de temperatura da superfície do mar.

Foram desenvolvidos três modelos de derivadores flutuantes *WAVY* (figura 2.1): básico, litoral e oceânico, com tamanhos, tecnologias e resistências diferentes, para se adaptarem da melhor forma ao meio.

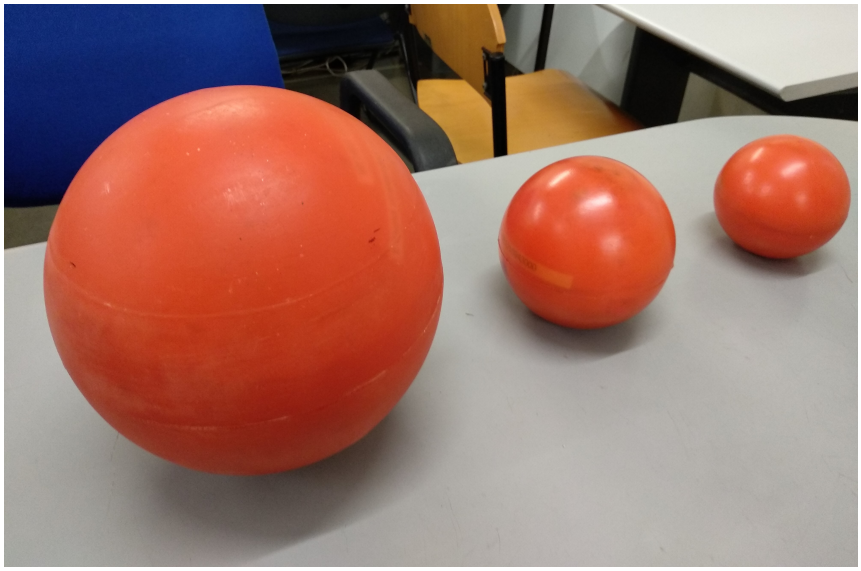


Figura 2.1: Diferentes modelos de *WAVY*

Os modelos básico e litoral comunicam através da rede GSM, enquanto o modelo oceânico irá transmitir via satélite. Contudo, o modelo oceânico ainda está numa fase de prototipagem, sendo que está previsto quando o projeto terminar, este modelo estar operacional.

2.3 Ecossistema MELOA

Todos os componentes deste projeto estão distribuídos pelos vários elementos do consórcio. Como mostra o diagrama da figura 2.2, os derivadores flutuantes *WAVY* tem a capacidade de comunicar por *WiFi*, pela rede GSM ou através da ARGOS.

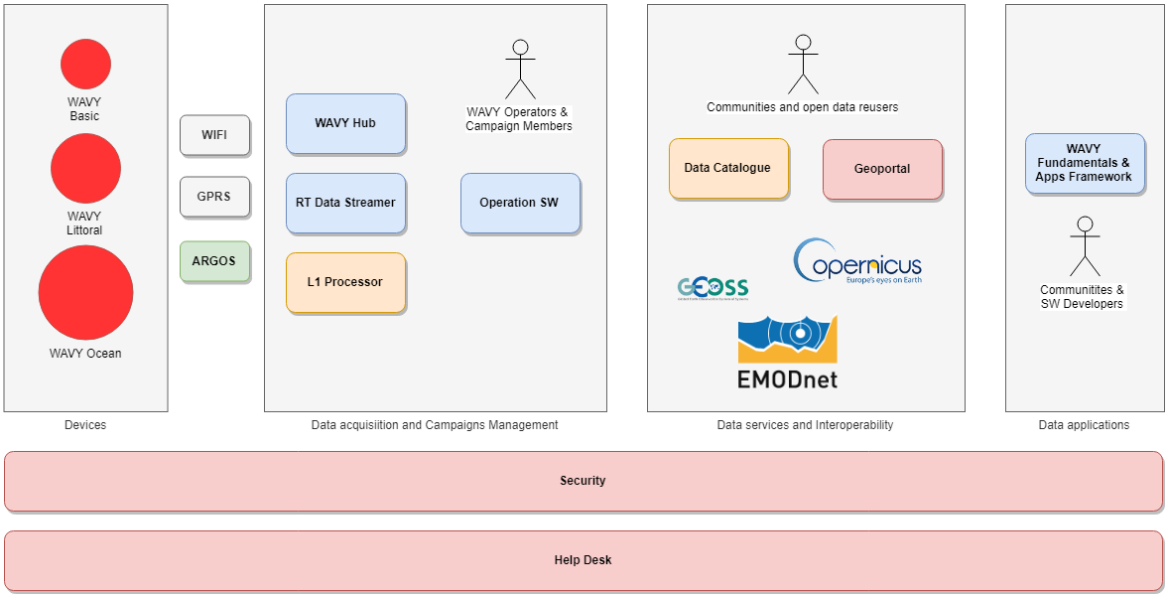


Figura 2.2: Diagrama de *software* do ecossistema MELOA [18]

Através da rede *WiFi*, os derivadores flutuantes *WAVY* comunicam com a aplicação móvel *WAVY Hub*, projetada pela LSTS, que funciona como uma interface entre o dispositivo e o utilizador. Esta aplicação possibilita a configuração de determinados parâmetros como frequências de recolha de dados e ativar/desativar certos sensores.

Com a tecnologia GPRS, que trabalha sobre a rede GSM, estes derivadores comunicam com o *Real Time Data Streamer*, que é um intermediário de mensagens, implementado pelo INESC TEC, que publica as mensagens enviadas pelos derivadores para o seu subscritor, *WAVY Operation Software*. Este *Real Time Data Streamer* utiliza o protocolo AMQP, devidamente autenticado, que garantindo a entrega e autenticidade de todas as mensagens transmitidas pelos dispositivos.

Por último, os modelos oceânicos usam para as suas comunicações a rede de satélites ARGOS, do CLS, para enviar os seus dados para o *L1 Data Processor*, da responsabilidade da *Deimos Engenharia*. Este *software* descodifica e transforma a mensagem binária, proveniente da ARGOS, produzindo mensagens compreensíveis ao *Real Time Data Streamer*, para posteriormente lhe serem transmitidas.

O *WAVY Operation Software* é uma plataforma *Web*, a encargo do INESC TEC, que ajuda a planear, monitorizar e explorar a execução de lançamentos dos derivadores flutuantes *WAVY*. Também é possível nesta plataforma publicar conjuntos de dados, designados de *datasets*, provenientes de lançamentos, no *Data Catalogue*. É no WOS que esta dissertação vai incidir, com o seu melhoramento e desenvolvimento de novas funcionalidades.

Até aqui, todas as plataformas são de domínio privado, sendo que só são acessíveis a utilizadores devidamente registados e autorizados. Por questões de segurança e de forma a haver um serviço centralizado, até ao final do projeto, irá ser desenvolvida uma plataforma para registo e gestão dos utilizadores do ecossistema. Este serviço de autenticação centralizado será integrado em todos os restantes *softwares* - secção *Security* no diagrama da figura 2.2, da responsabilidade da *Deimos Space*.

Uma vez publicados *datasets* no *Data Catalogue*, os mesmos estão acessíveis no *Geoportal*, plataformas desenvolvidas pelas *Deimos Engenharia* e *Deimos Space* respetivamente. Estas plataformas são de domínio público, sendo que qualquer pessoa pode explorar estes dados por elas disponibilizados.

O *WAVY Fundamentals* será uma *framework*, disponibilizada pelo INESC TEC, para que qualquer pessoa consiga desenvolver, de maneira simples e intuitiva, aplicações *Web* e *mobile* que possibilitem a análise e exploração de dados de lançamentos públicos para a diferentes comunidade.

Ainda da responsabilidade da *Deimos Space*, o serviço *Help Desk* é um *software* que disponibiliza ajuda nas diversas plataformas do projeto MELOA e onde utilizadores podem reportar qualquer eventualidade que possa ocorrer.

Capítulo 3

Estado da Arte

O oceano está em constante movimento. Com os inúmeros fatores que afetam este movimento, os cientistas ainda batalham para saber como o oceano se comportará em qualquer parte do globo. Este baixo conhecimento sobre o oceano foi evidente quando, em 2010, houve o maior derrame acidental de petróleo, no golfo do México, onde era necessária uma limpeza rápida e eficaz. Conforme o derrame aumentava, as equipas de resposta a este tipo de desastres precisavam de saber onde é que o petróleo chegaria. Esta previsão foi um enorme desafio. [20]

3.1 Metodos para traçar correntes marinhas

Hoje em dia, os oceanógrafos continuam a estudar as correntes marinhas para estarem melhor preparados para caso aconteça mais alguma catástrofe destas. Para isto, em 2013 foi fundada a CARTHE (*Consortium for advanced on transport of hydrocarbon in the environment*) que é constitui um grupo de investigadores, com o objetivo de prever o destino de possíveis derrames de petróleo. Este grupo tem como objetivo ajudar a informar e orientar equipas de resposta, de forma a proteger e minimizar os danos à saúde humana, à economia e meio ambiente. [10]

De forma a providenciar um melhor estudo neste âmbito, a CARTHE, com parceria com inúmeras instituições e universidades, usa uma variedade de métodos para analisar as correntes marinhas.

3.1.1 *Acoustic Doppler Current Profiler* (ADCP)

Um ADCP é um sensor hidroacústico que mede a velocidade e direção de correntes através do efeito *Doppler* (fundamento físico na base de propagação de ondas sonoras). Como um SONAR, o ADCP envia sons, que são refletidos em partículas que estão em suspensão na água e que estão em movimento com a corrente. As ondas sonoras refletidas são captadas de volta pelo sensor. O efeito de *Doppler* é a diferença entre a frequência emitida e recebida pelo ADCP, que através de fórmulas matemáticas é calcula a velocidade e direção da corrente.

Os ADCP podem ser rebocados em pequenos barcos, mas também incorporados em veículos autónomos subaquáticos. Conhecidos como AUV, estes veículos, que geralmente têm uma forma semelhante a um torpedo, permitem um estudo mais amplo sobre esta matéria, como se pode ver na figura 3.1.

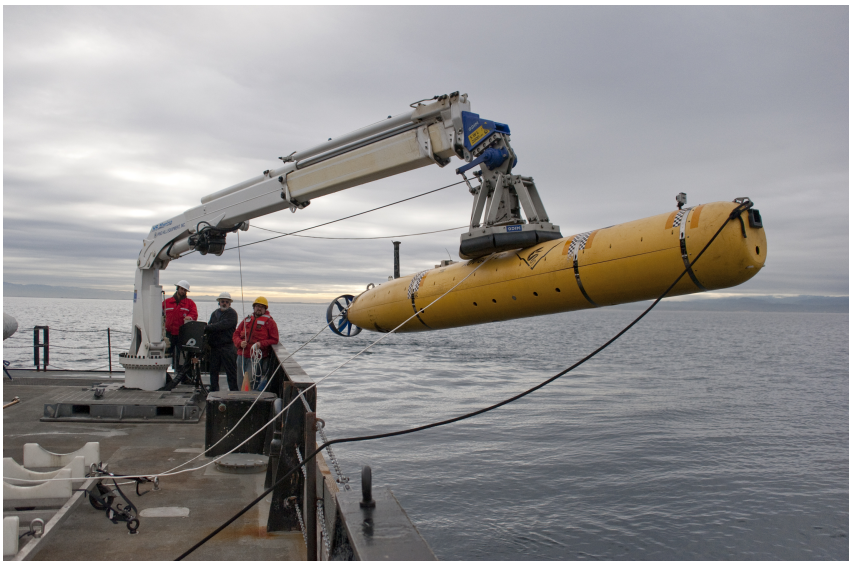


Figura 3.1: *Autonomous Underwater Vehicle* (AUV) [25]

3.1.2 Aviões

Por vezes, a melhor maneira de estudar as correntes oceânicas é através de uma vista aérea. Um simples olhar sobre a superfície do oceano, é possível identificar a frente e a direção da corrente. Com a ajuda de certos instrumentos, é possível detetar mudanças de temperatura na superfície do oceano, que é um indicador para correntes marinhas ou até mesmo atmosféricas, visto que estas estão diretamente relacionadas (subsecção 1.1.1).

3.1.3 Aeróstato e pratos de bambu

Uma outra forma de estudo de correntes marinhas, é com recurso a pequenos discos de bambu biodegradáveis, de tamanho e forma semelhante a um prato convencional, e um balão de ar quente (Aeróstato).

Depois de serem lançados uma série de pratos de bambu pela superfície do oceano, é largado um balão de ar quente ancorado a um barco. Este, equipado com camaras de alta resolução, é capaz de gravar os movimentos dos pratos para posteriormente ser construído um modelo das correntes marinhas, através de algoritmos e processamento de vídeo [9].

3.1.4 *Drones*

Como a zona costeira é um lugar difícil de prever e analisar, os investigadores recorreram a um corante biodegradável para entender melhor o movimento em direção à terra. No entanto, a melhor visualização de toda a cor espalhada pela costa é no céu onde, através de câmaras de alta resolução incorporadas em *drones*, estes captaram toda a dispersão da cor. Para aproveitar o seu uso foi usado novamente o método anterior, onde foram lançados também pratos de bambu.

3.2 Derivadores flutuantes

No início do século XX, como forma de estudar as correntes marinhas, os oceanógrafos lançaram cerca de duas mil garrafas de vidro, contendo uma mensagem. Para quem encontrasse uma garrafa e lesse a mensagem, era-lhe pedido que anotasse a data e local, e retornasse a mensagem para o endereço listado. Este foi o começo de todo o estudo das correntes marinhas. [23]

Hoje em dia, para o estudo de correntes marinhas, já não é necessário tanto trabalho, nem depender de alguém encontrar algo à deriva e ter a sorte de responder a uma mensagem.

3.2.1 Derivador básico

Um pequeno derivador GPS, foi desenvolvido entre 2008 e 2009 para estudo de correntes em zonas costeiras, por dois engenheiros provenientes do Irão. Este derivador, de pequeno porte, simples e de baixo custo, foi desenvolvido para ambientes onde não são necessárias todas as capacidades de derivadores maiores e mais complexos. Este derivador básico e de custo reduzido, figura 3.2, permitiu a realização de testes com grandes quantidades de derivadores, de forma a registar um enorme número de dados. Contudo, houve algumas falhas na fixação do recetor de GPS e no cálculo de velocidades [41], provavelmente pela sua estrutura não ser a mais adequada ao meio.

3.2.2 Derivador CARTHE

A CARTHE também desenvolveu estudos neste campo, onde concluíram que o registo da velocidade e direção da superfície pode ser feito através de vários derivadores flutuantes. Para isto é necessário que estes tenham posicionamento preciso, frequente e com durabilidade para várias semanas.

No entanto, devido à necessidade de serem vários derivadores, na casa das centenas, devem ser de baixo custo, compactos, fáceis de manusear e também construídos com materiais inofensivos ao meio ambiente. Portanto, a CARTHE desenvolveu derivadores biodegradáveis (figura 3.3), aproximadamente 3 meses de vida útil, incorporados com GPS, que transmitem a sua localização, numa frequência de 5 minutos, por satélite enquanto andam à deriva. [33]

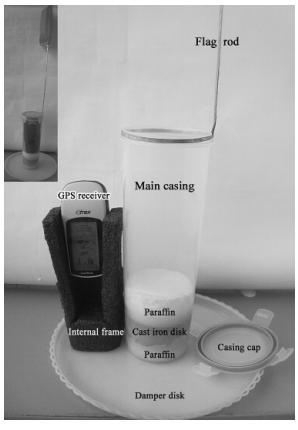


Figura 3.2: Composição de derivador básico [41]

Figura 3.3: Derivador biodegradável CARTHE [36]

3.2.3 Derivador flutuante *WAVY*

Destinado a ser um dispositivo de baixo custo, uma vez fabricado numa escala adequada, o derivador *WAVY* é um derivador de superfície leve e fácil de usar, armazenar e transportar. O modelo básico pode ser usado para estudar processos em praias, estuários e lagoas movidas por marés. A sua forma esférica permite com que flutue naturalmente, com uma reduzida exposição ao vento, e que não encalhe facilmente.

Estes produtos, com inúmeros sensores incorporados (GPS, temperatura, velocidade, etc), juntamente com o *WAVY Operation Software* apresentam inúmeras vantagens em relação a todos os restantes projetos.

Capítulo 4

WAVY Operation Software (WOS)

WAVY Operation Software é uma aplicação que dá apoio às operações de planeamento, execução e pós-processamento de campanhas de carácter cientifico para o estudo das correntes marinhas, utilizando os derivadores flutuantes *WAVY*. [3]

O elemento central do sistema são os dados recolhidos pelos derivadores flutuantes *WAVY*, designados de observações. No âmbito de uma campanha podem ser realizados múltiplos lançamentos com vários derivadores. O intervalo de tempo dos lançamentos é definido pelos utilizadores, sendo por norma caracterizado pelo tempo em que os derivadores fazem a recolha dos dados, ou mesmo um intervalo em que seja conveniente para a análise da informação recolhida. Assim que o lançamento terminar, os dados são automaticamente catalogados, num grupo de observações, designado de *dataset*, que se relaciona o par lançamento/derivador.

Para uma melhor compreensão desta hierarquia, podemos considerar o exemplo da campanha realizada no evento *Business2Sea*, que decorreu na Alfandega do Porto, com início no dia 11 de novembro de 2019. Como o evento teve a duração de três dias e como estava previsto haver um lançamento por dia, foram criados previamente três lançamentos. Sendo que em todos os eles foram lançados 5 derivadores, no final haviam 15 *datasets*, com as observações correspondentes ao intervalo de tempo definido nos lançamentos. Através do *WAVY Operation Software* é possível visualizar o progresso geográfico dos derivadores flutuantes *WAVY* lançados no decorrer do evento, como se pode ver na figura 4.1.

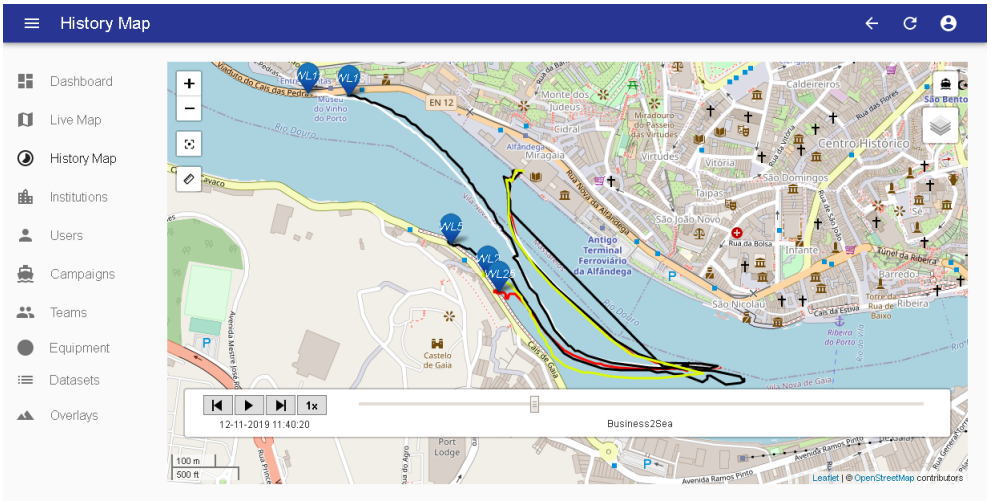


Figura 4.1: Mapa da campanha *Business2sea*, através de uma linha temporal

4.1 Arquitetura

A plataforma WOS é desenvolvida com base na linguagem *JavaScript*, sendo esta uma linguagem de programação simples, que pode ser interpretada ou compilada *Just In Time* (JIT). Suporta paradigmas de programação orientada a objetos, é imperativa e recentemente foi lançada a versão 6 da sua norma *ECMAScript*.[29]

Toda a plataforma WOS segue o modelo de uma arquitetura REST, composta por vários componentes, como se pode ver na figura 4.2.

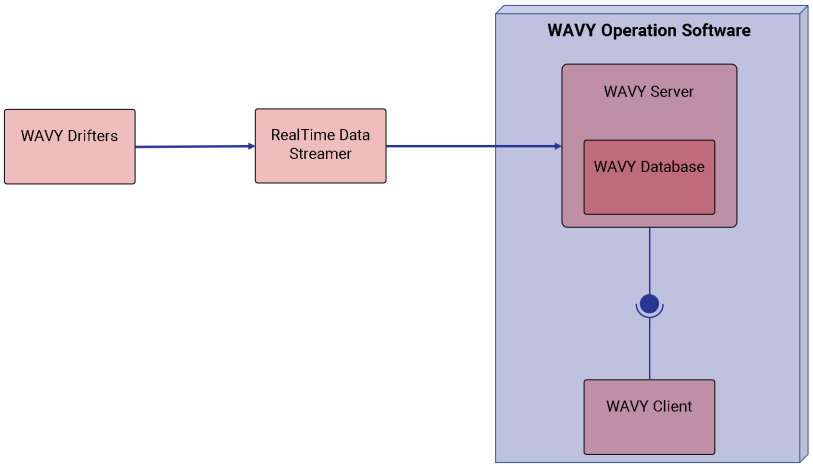


Figura 4.2: Diagrama do sistema WOS

4.1.1 *WAVY Server*

De maneira a uniformizar a linguagem usada, tanto na parte do cliente como da parte do servidor, optou-se pela utilização do *NodeJS*. O *NodeJS* é uma plataforma *JavaScript open source* que utiliza o interpretador *V8 JavaScript engine*, usado pelo *browser Chrome* da *Google*.[44] Através deste interpretador, o *NodeJS* transforma a linguagem virada ao desenvolvimento *frontend*, para a criação de servidores *Web*.

Com o objetivo de desenvolver uma API REST, optou-se por usar a *framework LoopBack* da IBM. Esta *framework* consiste numa estrutura *NodeJS*, que facilita o desenvolvimento de API's.

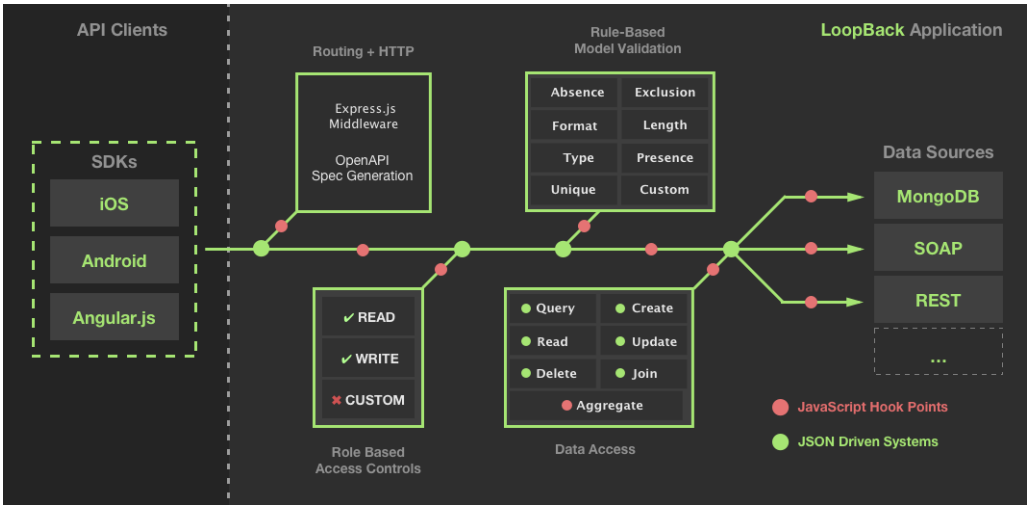


Figura 4.3: Estrutura da *framework LoopBack* [13]

Através de alguns ficheiros de configuração, o *LoopBack* rapidamente se estabelece uma API estável. Estes ficheiros, em formato JSON, definem as características e relações dos modelos que pretendemos para a API. Depois de definidos os modelos e suas relações, o *LoopBack* encarrega-se de gerar código *JavaScript* para satisfazer as necessidades pretendidas.

Esta ferramenta de alto nível, simplifica a criação de modelos relacionais (*hasMany*, *belongsTo* e *hasAndBelongsToMany*), quando conetada uma base de dados relacional. Além disso, conta com conetores para as diversas base de dados que suporta, como *MySQL*, *PostgreSQL*, *Oracle*, entre outras. O *LoopBack* também suporta autenticação e registo de utilizadores, através de *OAuth*, e autorização com controlos baseados em *roles*.

4.1.2 *WAVY Database*

Entre as diversas base de dados relacionais que existem, a escolha foi reduzida aquelas que são *open source* e que são suportadas pela *framework LoopBack*, através dos seus conetores. Deste modo, foi optada a base de dados *PostgreSQL*, pelo grande suporte para *LoopBack*, pela sua confiabilidade e por conter todas as características das principais base de dados existentes.

PostgreSQL é uma base de dados objeto-relacional *open-source*. Foi lançada em 1989 como *Postgres* e em 1996 passou a suportar SQL, onde foi renomeada para *PostgreSQL*. Está implementada em C e com mais de 30 anos de desenvolvimento ativo, ganhou uma forte reputação pela sua robustez de recursos e desempenho. [19]

4.1.3 *WAVY Client*

O Cliente do WOS é uma aplicação *Web* que tem como alicerce *React*. *React* é uma biblioteca *open source JavaScript*, desenvolvida pelo *Facebook*, com foco em criar interfaces interativas para páginas *Web*. Esta biblioteca usa uma sintaxe semelhante a HTML chamada JSX, de modo a tornar a mesma mais simples e sofisticada, sem ser difícil de a entender e usar.

Para facilitar o trabalho de desenvolvimento da aplicação, o cliente do WOS usufrui da *framework React-Admin*. Esta *framework* tira partido de *React*, *MaterialUI*, *React Router* e *Redux*, que facilita na criação aplicações *Web* com base em API REST.

No *WAVY Client* é possível a visualização de conteúdo através de tabelas e mapas, onde para este último foi adotada a *LeafLet*, que é uma biblioteca para mapas interativos com suporte para *React* e com compatibilidade para dispositivos móveis. Foram criados alguns *plugins* em *React*, como uma barra de tempo (figura 4.1), que permite navegar pelas observações ao longo de uma linha temporal, e um medidor de distâncias.

4.2 Estrutura do *WAVY Client*

A plataforma WOS é composta por várias secções e todas elas são acompanhadas de um menu lateral esquerdo, que serve para navegar entre as mesmas, e de uma barra superior, como se pode ver na figura 4.4.

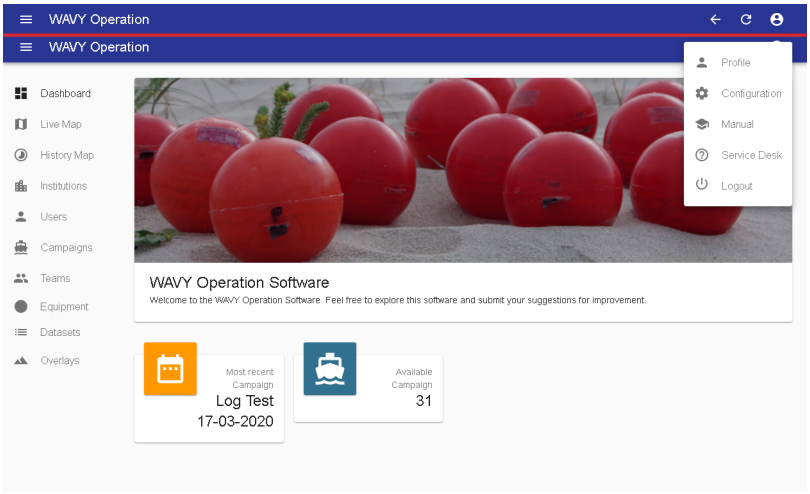


Figura 4.4: *Dashboard* da aplicação *Web* WOS

A barra superior identifica a secção em que o utilizador está e também contém um botão para retroceder nas diversas operações realizadas, outro para atualizar os dados e um último que disponibiliza um menu de utilizador. No menu de utilizador é possível visualizar e editar os detalhes do utilizador (email e nome), configurações da plataforma (linguagem e tema), aceder ao manual da plataforma e por último a possibilidade de realizar *logout*.

No entanto, nem todos os utilizadores terão acesso a todas as secções e operações da plataforma. Desta forma, existem três tipos de níveis de acesso: utilizadores, coordenador de uma instituição e administrador de sistema.

Um utilizador comum pode navegar pelas campanhas em que participou, bem como visualizar os dados recolhidos pelos derivadores nas mesmas, bem como as transmissões em tempo real. Um coordenador de uma instituição, além das permissões de um utilizador comum, pode criar e gerir campanhas, bem como os utilizadores dessa instituição. Este pode também controlar os utilizadores da sua instituição. Enquanto isto, um administrador de sistemas tem acesso total a toda a plataforma.

Visto isto, a plataforma *WAVY Operation Software* está dividida em:

- *Dashboard*

  No Dashboard (painel inicial), o utilizador pode encontrar o número total de campanhas, bem como aceder à mais recente (figura 4.4). No final, quando desenvolvidas todas as funcionalidades, vão ser estudados os dados relevantes de toda a plataforma para serem integrados neste painel inicial.

- *Live Map*

Na secção *Live Map*, é possível acompanhar, através de um mapa, os derivadores flutuantes *WAVY* que estão a transmitir dados em tempo real. Os derivadores são representados por um marcador azul clicável, que quando pressionado disponibiliza numa janela toda a informação transmitida na última comunicação com o servidor.

Este mapa contem um conjunto de botões que permitem as seguintes ações: alterar o *zoom*; visualizar todos os derivadores centrados; uma ferramenta para medir distâncias e escolher o tipo de mapa.

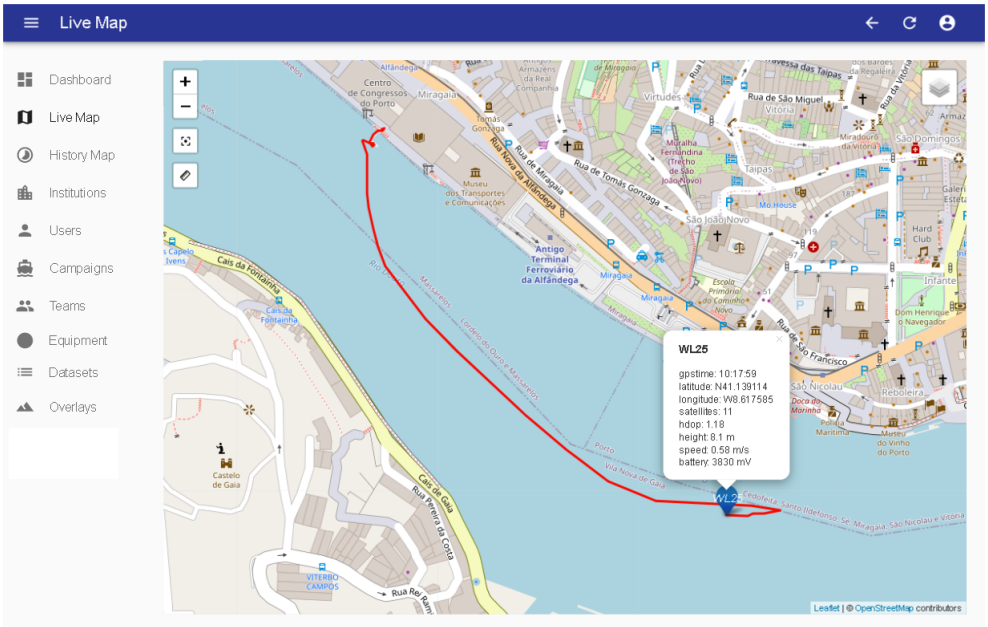


Figura 4.5: Mapa para visualizar os derivadores em tempo real

- *History Map*

Na secção *History Map*, o utilizador pode escolher um lançamento que já tenha sido realizado e visualizar o comportamento geográfico dos derivadores. Como mostra na figura 4.1, é possível navegar pelo lançamento através da linha temporal, localizada na parte inferior.

Da mesma forma que acontece no *Live Map*, quando o utilizador clica no marcador correspondente ao derivador, pode analisar os dados referentes ao tempo que consta na linha temporal. Também contém os restantes botões com as mesmas funcionalidades.

- *Institutions*

  Na secção *Institutions* é possível criar e gerir instituições, em que apenas administradores de sistemas terão acesso a esta secção. Além destas funcionalidades, um administrador de sistemas pode ainda verificar quais os utilizadores associados às mesmas.

- *Users*

  Na secção *Users* é possível associar utilizadores a instituições, quando se tem privilégios de administrador de sistemas e definir permissões, bastando para isso ser administrador da instituição em causa.

- *Campaigns*

  Como já referido, uma campanha consiste num ou mais lançamentos, num local especifico. Toda a informação recolhida no decorrer da mesma, está disponível para os utilizadores que pertencem às equipas que participaram nas campanhas na secção *Campaigns*.

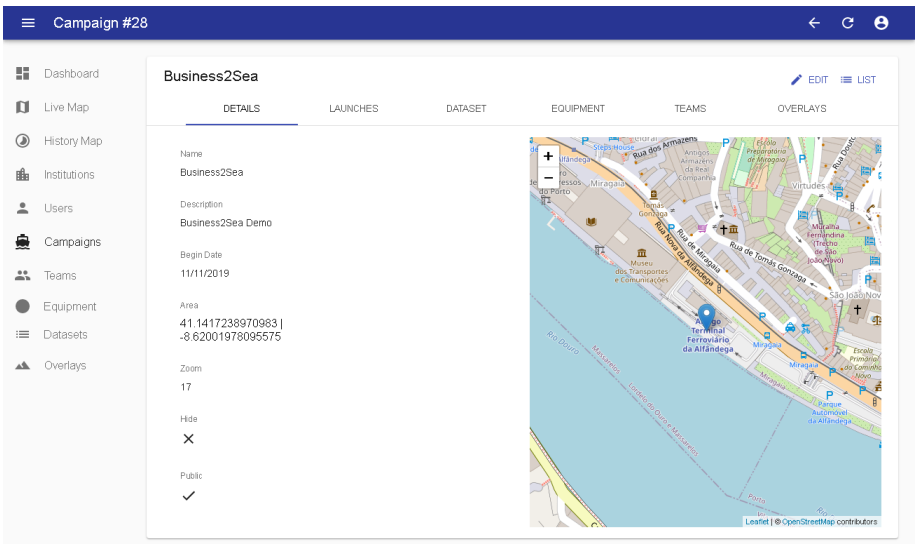


Figura 4.6: Página de detalhes da campanha *Bussiness2sea*

Esta secção tem o propósito de gerir campanhas e lançamentos, associar equipas e equipamentos às mesmas e verificar quais os *datasets* gerados, através de subsecções destinadas a cada uma destas funcionalidades. Neste caso, os utilizadores podem ver toda a informação, mas apenas administradores têm acesso para criar e editar a mesma.

- *Teams*

  Na secção *Teams*, os administradores podem gerir equipas para a sua instituição, associando às mesmas utilizadores.

- *Equipment*

  Para ajudar na logística de campanhas, o *WAVY Operation Software* apresenta um conceito de equipamento, permitindo que outros equipamentos, além dos derivadores flutuantes *WAVY*, possam ser adicionados no sistema.

  Na secção *Equipment*, administradores podem associar equipamentos às suas instituições, indicando qual a sequência dos parâmetros na mensagem enviada em tempo real dos mesmos, para ser possível identificar o derivador em causa, bem como os valores obtidos pelo mesmo.

- *Datasets*

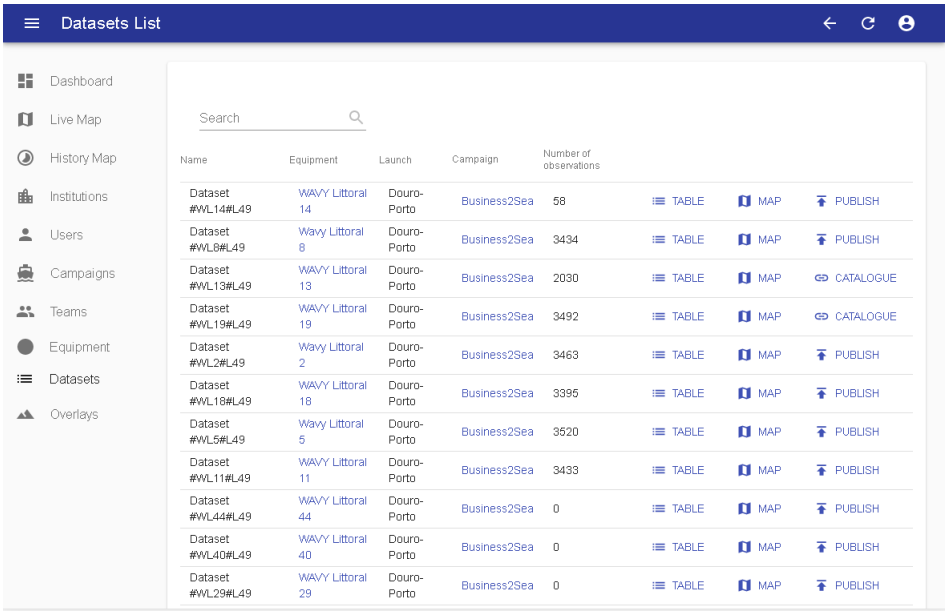


Figura 4.7: Lista de *datasets* de um utilizador que participou no *Bussiness2Sea*

Todos os utilizadores podem ver os *datasets* gerados nas campanhas em que participaram. Existem duas formas de analisar os dados recolhidos pelos derivadores flutuantes *WAVY*: através de uma tabela ou geograficamente num mapa.

Se for selecionada a opção de tabela, a informação é apresentada conforme a figura 4.8.

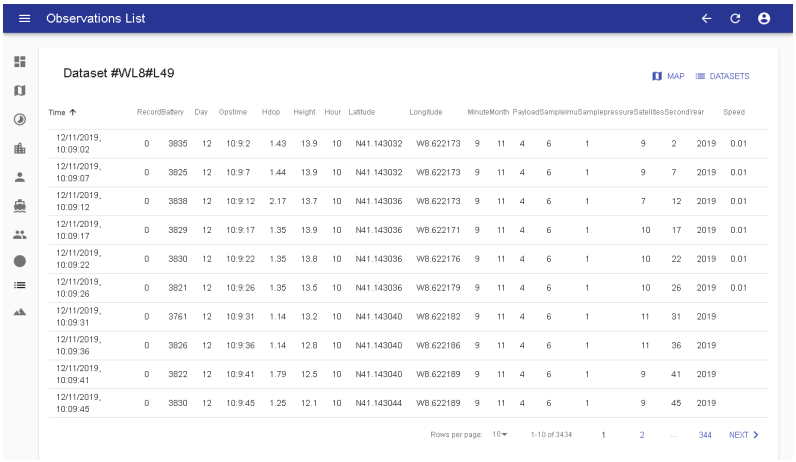


Figura 4.8: Visualização em forma de tabela

De realçar, que o menu lateral pode ser contraído a qualquer momento, mantendo a possibilidade de navegar entre o mesmo.

No caso do mapa, que é acompanhado de uma linha temporal, como acontece na secção *History Map*, porém apenas se visualiza os dados referentes ao derivador associado ao *dataset*. Além disto, o utilizador pode publicar a informação do *dataset* no *Data Catalogue*, para tornar os dados públicos. Após publicados, é disponibilizado uma hiperligação para a página pública.

- *Overlays*

  Uma *overlay* é uma sobreposição de uma imagem no mapa. Aqui um utilizador pode as adicionar através de ficheiros ou a partir de um endereço *Web*. Estas sobreposições são acessíveis, quando associadas a campanhas, no mesmo botão em que o utilizador pode alterar o tipo de mapa.

4.3 Dados em tempo real

O *WAVY Operation Software* foi construído para suportar dados em tempo real, onde geralmente esta transmissão tem uma cadência de uma observação por cada cinco segundos, no limite este valor pode ser ajustado para uma observação por segundo. A configuração destes valores pode ser feita através de uma aplicação *mobile*, *WAVY Hub*, que emparelha com o derivador flutuante *WAVY*, servindo de interface entre o utilizador e o próprio derivador.

Depois dos derivadores flutuantes *WAVY* estarem ligados, começam a comunicar com o servidor do WOS. As mensagens destas comunicações consistem numa *string* (formato ASCII). Após receber uma mensagem, o servidor retorna 'OK' para que o derivador reconheça o envio como bem sucedido.

Cada transmissão corresponde a um pedido GET via HTTP, como é mostrado no exemplo abaixo:

http://server.path.pt:80/u?v=WL12|12:45:54|N41.17842|W8.59680|8|1.23|123.3|0.2|6|20|1|1|3.98

A mensagem transmitida através do parâmetro 'v', contém os dados relativos aos sensores do derivador. A figura 4.9 mostra a estrutura desta mensagem ASCII, onde cada parâmetro está dividido por uma barra vertical ' | ', seguido de uma explicação do seu conteúdo. Além de ser um requisito do fabricante, existem limitações nos derivadores flutuantes *WAVY*, que obrigam a usar este tipo de pedido HTTP com os parâmetros no endereço *Web*.

A	B	C	D	E	F	G	H	I	J	K	L	M
WB12	12:45:54	N41.17842	W8.59680	8	1.23	123.3	0.2	6	20	1	1	3.98

Figura 4.9: Mensagem transmitida por um derivador flutuante *WAVY*

- A - Corresponde ao identificador do derivador flutuante (ex: WL12 = WL[ID])
- B - Horário fixo do GPS (ex: 12:45:54)
- C - Latitude (ex: N41.17842)
- D - Longitude (ex: W8.59680)
- E - Número fixo de GPS (ex: 8)
- F - GPS HDOP (ex: 1.23)
- G - Altitude em metros (ex: 123.3)
- H - Velocidade em metros por segundo (ex: 0.2)

- I - Opções de configuração (ex: 6 = sensores de IMU e Temperatura ligados)

  A tabela 4.1 explica os valores possíveis neste parâmetro (*payload*), sendo que 0 e 1 significam que sensor está desligado ou ligado, respetivamente.

IMU	TEMP	ATM	Valor
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Tabela 4.1: Opções de configuração (*payload*)

- J - Número de amostras que o sensor IMU faz por segundo (ex: 20 por segundo)
- K - Intervalo de tempo em que o sensor de temperatura realiza uma amostra (ex: 1 amostra por segundo)
- L - Intervalo de tempo em que o sensor de pressão (ATM) realiza uma amostra (ex: 1 amostra por segundo)
- M - Nível de bateria em volts (ex: 3.98 V)

A mensagem explicada anteriormente, especificada pelo fabricante dos derivadores flutuantes *WAVY*, é referente a uma transmissão de um modelo litoral. No entanto, esta não é a estrutura da mensagem transmitida pelos restantes modelos, como o caso do *WAVY Ocean*.

De modo a tornar o sistema dinâmico e a suportar múltiplos modelos dos derivadores flutuantes, foi introduzida no WOS uma funcionalidade que permite a configuração do formato da mensagem transmitida, para cada modelo. Portanto, quando a estrutura da mensagem é definida, são obrigatórios os campos: tempo (*timestamp*), número de série e a posição (latitude e longitude).

Os campos obrigatórios garantem o relacionamento das observações com um derivador flutuante e que as mesmas possam ser representadas temporal e geograficamente. No entanto, o campo número de série é obrigatório estar na primeira posição da mensagem, de modo a permitir saber de imediato a que modelo *WAVY* pertence a mensagem, que por sua vez possibilita a sua descodificação.

Depois do servidor saber os campos referidos no paragrafo anterior, guarda os mesmos na base de dados juntamente com a mensagem intacta (*content*) e um campo de validação *status*, como se pode ver na figura 4.10.

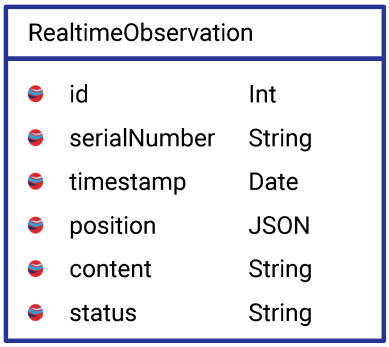


Figura 4.10: Tabela das observações de tempo real

O campo de validação *status*, toma o valor de erro quando algum dos campos obrigatórios seja inválido. Isto pode acontecer, por exemplo, quando algum valor de posição tome valores fora do esperado (a latitude para ser válida tem de estar entre os -90 e 90 graus, enquanto que a longitude entre -180 e 180 graus).

Como referido anteriormente, a mensagem é guardada conforme chega ao servidor. Isto acontece para assegurar que toda a informação é registada. Pois, se algum parâmetro for acrescentado ou até mesmo esquecido o seu registo na plataforma, o mesmo está salvaguardado.

4.4 Contextualização de dados

Depois de um lançamento ser realizado é feita uma contextualização dos dados recolhidos pelos derivadores. São automaticamente criados grupos, designados de *datasets*, que têm uma relação direta com o lançamento e o equipamento. Todos os dados recolhidos pelo equipamento no intervalo de tempo definido no lançamento, são atribuídos ao *dataset* relacionado com esse equipamento e lançamento.

Inicialmente é feita uma seleção de todos os derivadores flutuantes *WAVY* associados ao lançamento, bem como a data de inicio e fim do mesmo. Posteriormente, por cada derivador, é realizada uma pesquisa pelo identificador e intervalo de tempo do lançamento, na tabela dos dados de tempo real.

```
content = {
        "serialNumber": "WB12",
        "gpstime":"12:45:44",
        "latitude":"N41.17842",
        "longitude":" W8.59680",
        "satellites":"8",
        "hdop":"1.23",
        "height":"123.3",
        "speed":"0.2",
        "payload":"6",
        "sampleimu":"20",
        "sampletemperature":"1",
        "samplepressure":"1",
        "battery":"3.98",
}
```

Figura 4.11: Mensagem recebida em forma de objeto

Uma vez selecionadas as observações, a mensagem é transformada num objeto, relacionando os parâmetros com o seu valor, como mostra a figura 4.11. Depois disto, este objeto, juntamente com o tempo e a posição, são adicionados a uma outra tabela na base de dados (figura 4.12) com um identificador do *dataset* correspondente (*datasetId*).

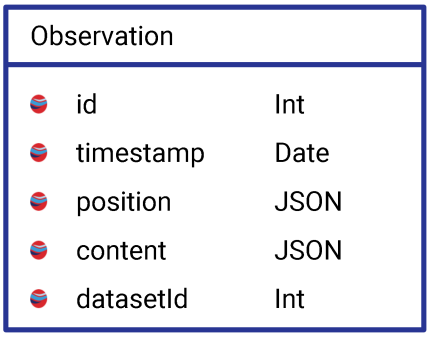
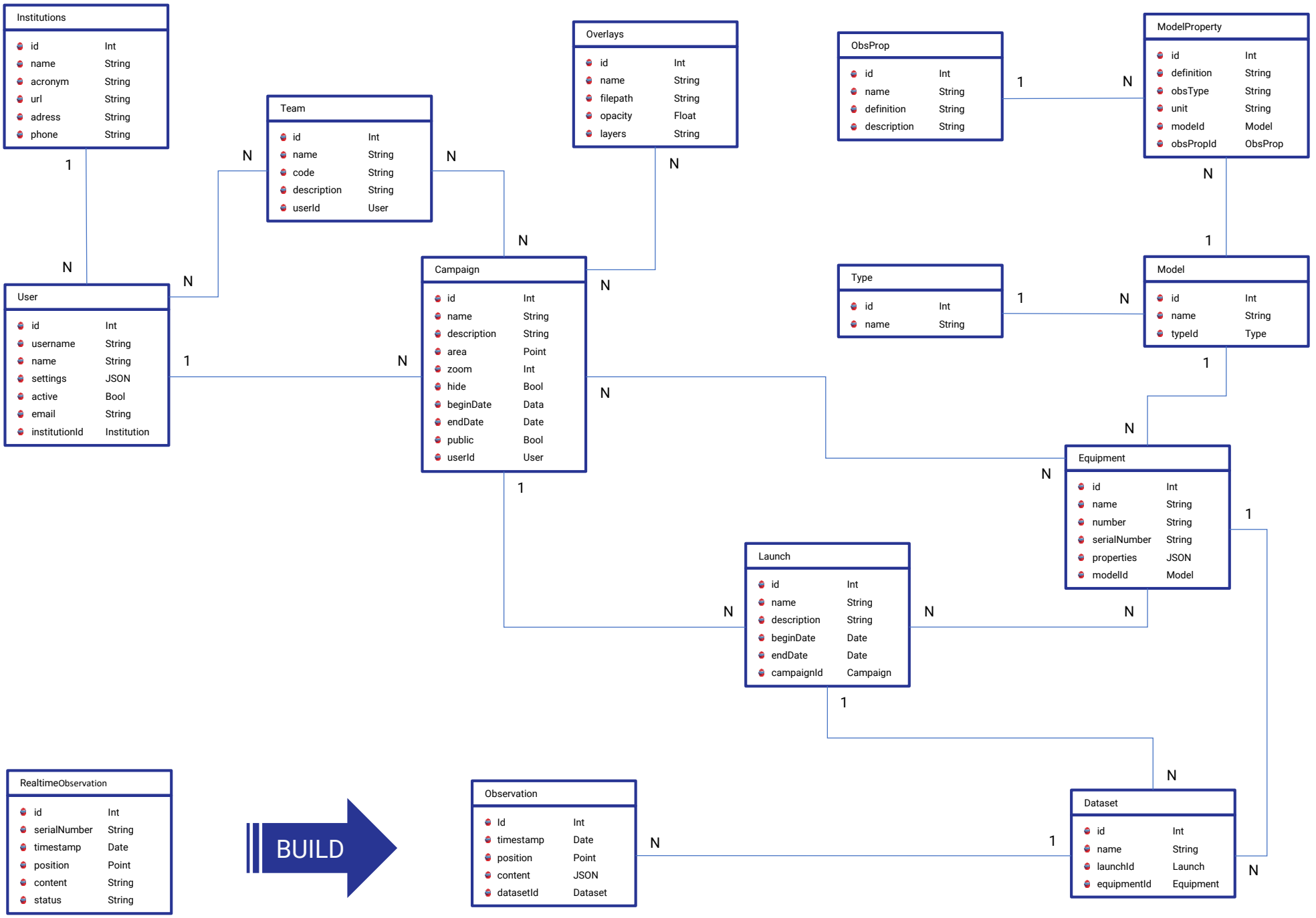


Figura 4.12: Tabela das observações pertencentes a um *dataset*

Neste ponto, ainda numa fase de desenvolvimento, foi criado um botão na secção de lançamentos que ativa o processo de contextualização de dados (*BUILD*). No final, este processo será ativado automaticamente caso o lançamento acabe, seja criado posteriormente ou haja alguma alteração de datas relacionadas com o mesmo.

4.5 Versão 1 do sistema WOS

Na página seguinte é mostrado um diagrama que representa toda a estrutura relacional do *WAVY Operation Software* na sua fase inicial (versão 1). O mesmo irá acontecer, no final dos próximos capítulos, para as alterações que irão acontecer ao longo deste trabalho serem mais percetíveis.


Institutions
id Int
name String
acronym String
url String
adress String
phone String
Team
id Int
name String
code String
description String
userId User
Overlays
id Int
name String
filepath String
opacity Float
layers String
ObsProp
id Int
name String
definition String
description String
ModelProperty
id Int
definition String
obsType String
unit String
modelId Model
obsPropId ObsProp
User
id Int
username String
name String
settings JSON
active Bool
email String
institutionId Institution
Campaign
id Int
name String
description String
area Point
zoom Int
hide Bool
beginDate Data
endDate Date
public Bool
userId User
Type
id Int
name String
Model
id Int
name String
typeId Type
Equipment
id Int
name String
number String
serialNumber String
properties JSON
modelId Model
Launch
id Int
name String
description String
beginDate Date
endDate Date
campaignId Campaign
RealtimeObservation
id Int
serialNumber String
timestamp Date
position Point
content String
status String
BUILD
Observation
Id Int
timestamp Date
position Point
content JSON
datasetId Dataset
Dataset
id Int
name String
launchId Launch
equipmentId Equipment
1
N

Capítulo 5

Novas ferramentas do WOS

Este capítulo é dedicado ao desenvolvimento de ferramentas para a análise, visualização e curadoria da informação recolhida pelos derivadores flutuantes *WAVY*, na plataforma WOS.

Numa primeira etapa, será integrada uma nova forma de visualização de dados, através de gráficos, com o propósito de explorar os diferentes parâmetros recolhidos pelos sensores dos derivadores ao longo do tempo. Esta visualização gráfica dos dados tem como objetivo auxiliar os investigadores no trabalho de análise e limpeza da informação, permitindo uma fácil identificação de dados anómalos (*outliers*).

Posteriormente, na segunda etapa será criada uma secção, chamada de *Derived Datasets*. Esta destina-se a dar apoio aos investigadores que utilizam a plataforma em tarefas de curadoria de dados, como identificação e limpeza de valores anómalos, gerando novos produtos, mantendo sempre a integridade da informação original.

5.1 Gráficos

Com o objetivo de expressar visualmente dados recolhidos pelos derivadores flutuantes *WAVY*, em alternativa às já existentes, pretende-se implementar uma visualização baseada em gráficos que facilitem a compreensão dos dados recolhidos.

Seguindo os requisitos para integração desta forma de visualização gráfica, foi utilizada a *framework open source amCharts*.

5.1.1 *amCharts*

Esta é uma ferramenta voltada para visualização de gráficos e mapas, que possui diversas funcionalidades e um bom desempenho.[5] Além disto, esta biblioteca *JavaScript* para gráficos, é responsiva e adapta-se a dispositivos moveis. A ferramenta *amCharts*, integra-se nativamente com aplicações implementadas em *Angular*, *Vue*, *React* e *JavaScript* simples.

A primeira versão deste *software* foi lançada em 2004, com o objetivo de disponibilizar uma ferramenta *open source*. Hoje em dia, já se encontra na sua versão 4, cada vez mais dinâmica e com novas funcionalidades, mantendo sempre suporte para todos os *browsers*. Esta *framework* é capaz de produzir praticamente todos os tipos de gráficos, com diversas formas de interação.[5]

5.1.2 Implementação

A idealização deste componente é produzir um gráfico de linhas simples e intuitivo, que permita explorar o comportamento de cada parâmetro ao longo do tempo. Deste modo, o primeiro passo foi adicionar um botão na secção *Datasets*, que redirecionasse para uma página onde é gerado o gráfico (figura 5.1).

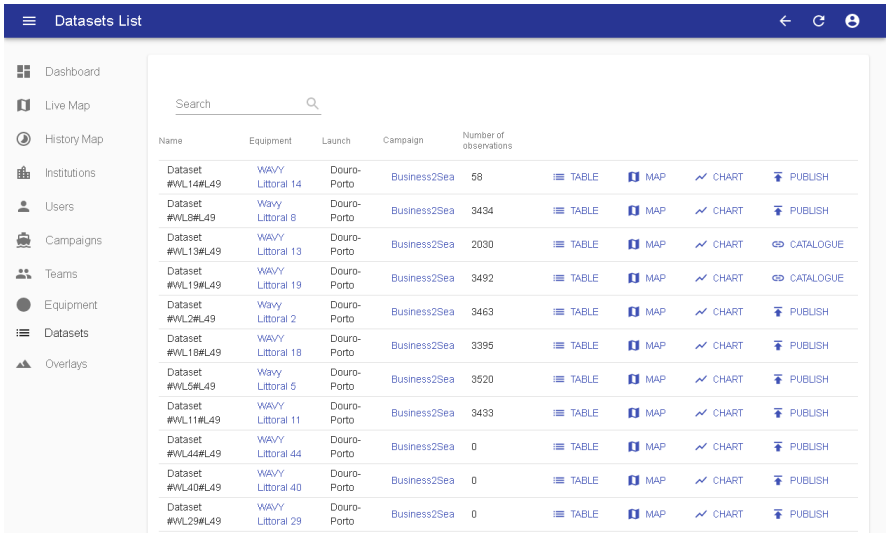


Figura 5.1: Secção *Datasets* com botão para gráficos

De forma a tornar esta ferramenta reutilizável e de fácil implementação pela plataforma, foi criado um componente dinâmico para posteriormente ser integrado na secção *Derived Datasets*. A criação de componentes genéricos é comum em aplicações *Web*, para a criação de elementos reutilizáveis [28], como acontece no caso dos mapas, onde são utilizados em diversas secções.

O componente referido é responsável por gerar o gráfico, consoante as configurações definidas para: interação entre os parâmetros incorporados nos dados, definições de *design*, *zoom* e determinados comportamentos em eventos, como ocultar e disponibilizar parâmetros.

A API disponibiliza rotas que possibilitam a obtenção dos dados (observações), agrupados por *dataset*, no entanto estes dados necessitam de uma transformação para que possam ser ingeridos pela ferramenta *amCharts*.

Esta ferramenta apresenta diversas vantagens, sendo as mais importantes e cruciais para analise de dados, o facto da escala ser automaticamente ajustada consoante os valores tomados pelos parâmetros e quando é selecionado um intervalo de tempo.

De maneira a facilitar a visualização dos parâmetros, pois estes diferem entre si na ordem de grandeza, apenas são visíveis quando clicados na legenda correspondente. Ou seja, é possível mostrar e esconder os parâmetros, com o objetivo de apresentar apenas os valores que um utilizador pretenda, como se pode ver na figura 5.2.

Figura 5.2: Visualização gráfica

Como se pode verificar, através da visualização gráfica da velocidade do derivador, facilmente se identificam *outliers* na parte final do lançamento. Neste caso, esta discrepância tem origem na recolha do derivador através de uma mota de água, que se não for tratada, pode causar uma contaminação quando os dados forem usados para o estudo das correntes marinhas.

5.2 Secção *Derived Datasets*

De forma a minimizar valores anómalos, como o exemplo da figura 5.1.2, a introdução desta nova secção vem adicionar um novo conceito. A secção *Derived Datasets*, será dedicada à manipulação de dados de um *dataset*. Este novo conceito possibilita a adição de observações provenientes de diversos *datasets*, desde que contenham os seguintes elementos comuns: campanha e equipamento.

Quando escolhida a visualização em tabela na secção *Datasets* (figura 4.8), é nos disponibilizada uma simples lista com todas as observações ligadas ao mesmo. De modo a possibilitar a manipulação dos dados, a ideia para esta funcionalidade passa por adicionar a opção de selecionar observações dessa mesma lista e adiciona-las a um *derived dataset*, previamente criado.

Depois disto, a página dedicada à exploração destes dados manipulados, será semelhante à dos *datasets*, no sentido em que se possa optar pela visualização em mapas, tabelas ou até através dos gráficos desenvolvidos anteriormente. Além disto, deverá conter uma opção para criar e gerir *derived datasets*, para posteriormente se poder associar observações aos mesmos.

Aqui não só é necessária uma intervenção na aplicação *Web*, mas também no lado do servidor, visto que não suporta este novo conceito.

5.2.1 Implementação no *WAVY Server*

Antes de começar o desenvolvimento no lado do cliente, foi necessário preparar o servidor para estar apto para cuidar de toda a informação proveniente desta nova secção. Assim sendo, foi necessário desenvolver os modelos que suportem este conceito pela *framework LoopBack*.

Deste modo, foi criado o modelo *Derivedset*, de forma a associar um nome e uma descrição dos *derived datasets*. Depois disto e tendo em conta que uma observação pode estar em múltiplos *derived datasets* e estes podem ter múltiplas observações, vemos que estamos perante uma relação de "muitos para muitos". A figura 5.3 mostra esta relação.

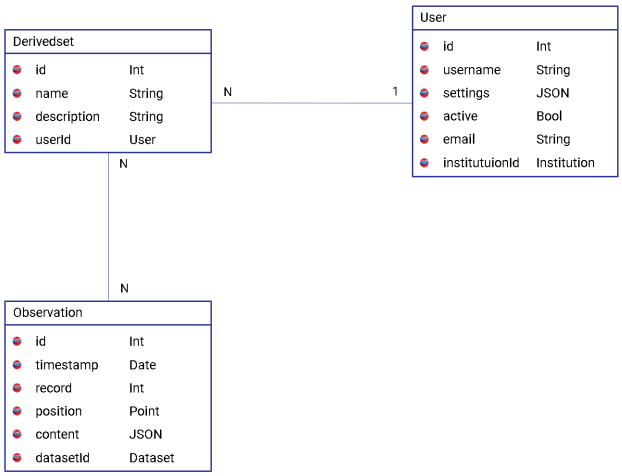


Figura 5.3: Relações do modelo *Derivedset*

Feito isto, foi necessário disponibilizar rotas para que este novo modelo estivesse disponível na API. Como o *LoopBack* facilita neste processo, foram disponibilizadas automaticamente as rotas básicas (figura 5.4), para responder à gestão dos *derived datasets*, e criadas outras para associar e desassociar observações.

Figura 5.4: Rotas CRUD geradas pelo *LoopBack*

As rotas básicas CRUD são automaticamente geradas pelo *LoopBack*, tendo em conta o modelo previamente criado. Estas rotas incluem a associação de observações e *derived datasets*, porém esta só permite uma inserção por pedido. Como na aplicação *Web* pretende-se associar múltiplas observações, isto resultaria em múltiplos pedidos, o que consequentemente poderia tornar o sistema lento. De maneira a ultrapassar esta questão, desenvolveu-se uma rota especifica, com capacidade de relacionar de uma só vez, múltiplas observações com um *derived dataset*.

5.2.2 Implementação no *WAVY Client*

Após o desenvolvimento das rotas necessárias para o funcionamento desta nova secção, foi a vez de avançar para o desenvolvimento na aplicação *Web*. Para isso foi necessário desenvolver um componente para os *derived datasets*, que permitisse operações de gestão como criar, editar e apagar, como se pode ver na seguinte figura 5.5.

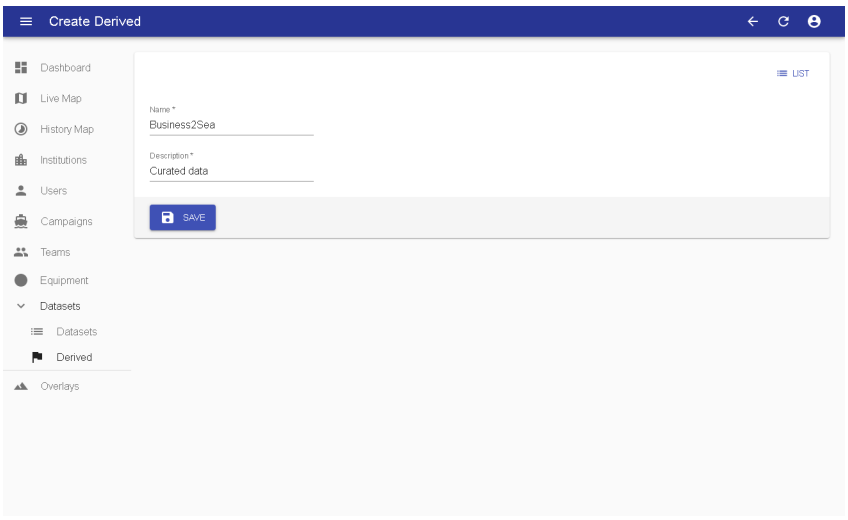


Figura 5.5: Criar um derivado de *dataset*

Como esta nova secção é semelhante à dos *datasets* foi integrada a possibilidade de visualização destes novos dados, através de tabelas, mapas e gráficos. De modo a organizar as secções e melhorando a experiência de utilização, no menu lateral foi integrado um submenu, onde agrupa os *datasets* e os seus derivados.

De seguida, foi adicionada uma funcionalidade onde permitisse ao utilizador selecionar múltiplas observações na tabela dos *datasets*, como mostra a figura 5.6.

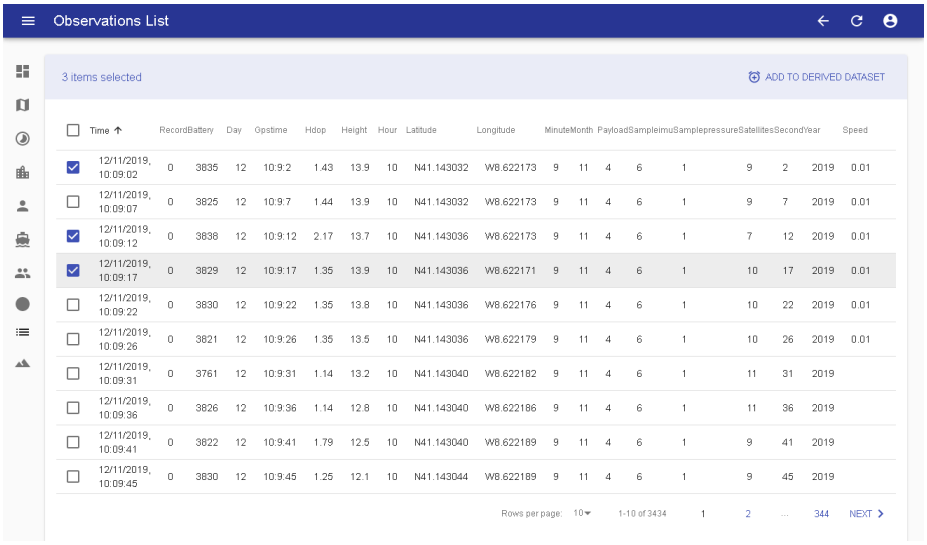


Figura 5.6: Tabela com opção de selecionar observações

Após selecionadas as observações na tabela dos *datasets* (figura 5.6), é disponibilizado ao utilizador um botão, onde após pressionado, apresenta uma janela com os *derived datasets* disponíveis, como mostra a figura 5.7. De realçar que apenas são apresentados os *derived datasets*, que contenham nenhuma ou observações com os elementos campanha e o equipamento comuns.

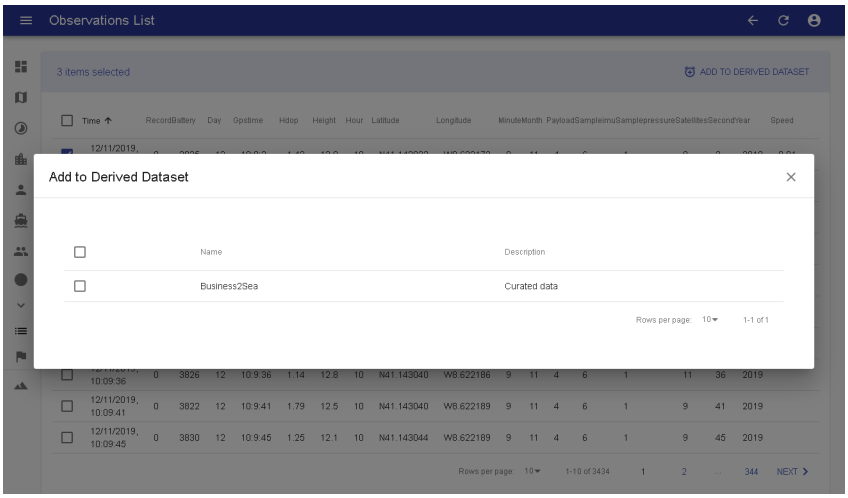
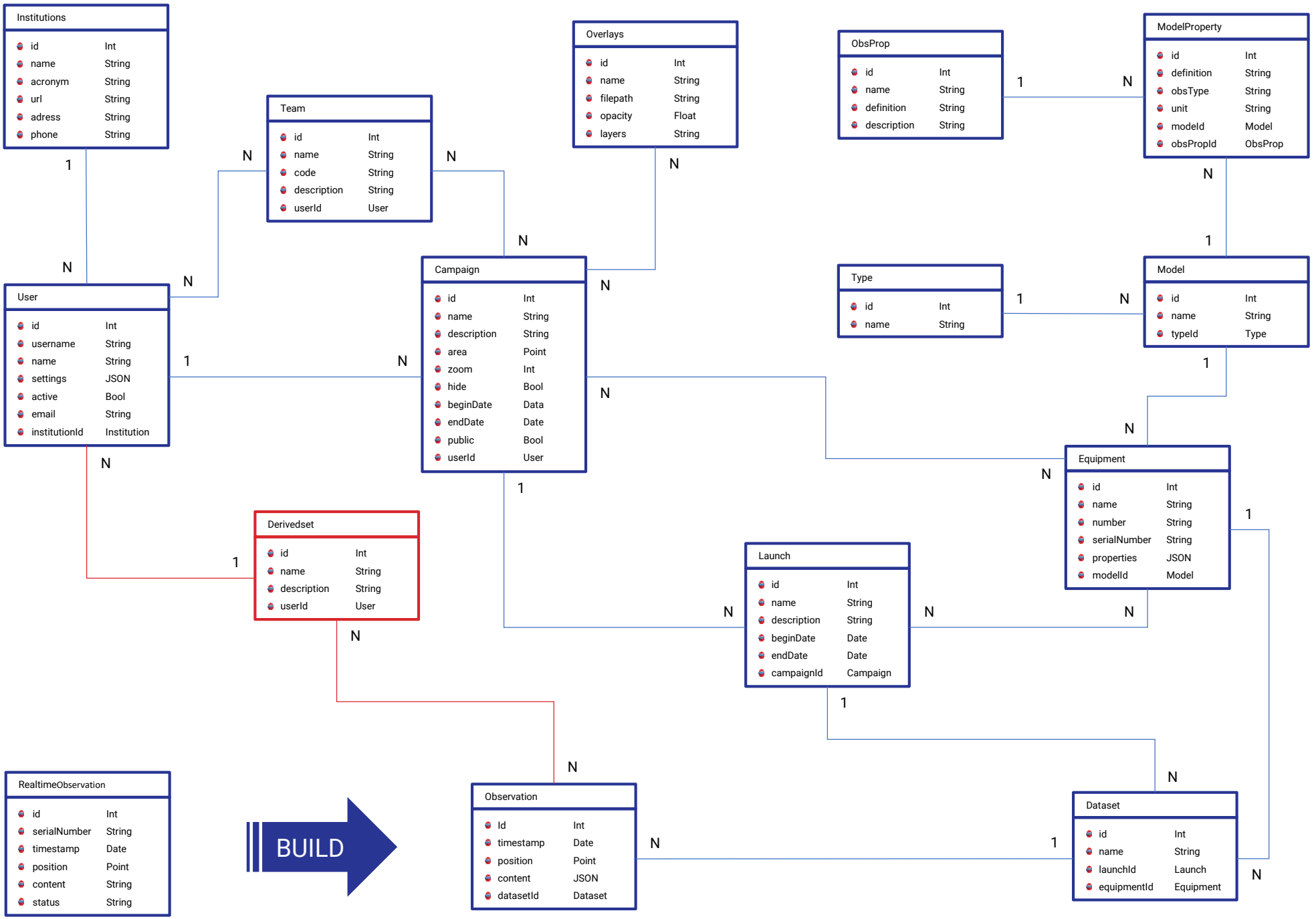


Figura 5.7: Escolha de *derived dataset*

5.3 Versão 1.1 do sistema WOS

Depois da implementação da visualização de dados através de gráficos e da secção para os *derived datasets*, foi realizada uma validação de usabilidade. O objetivo desta validação é identificar possíveis deficiências desta versão, do ponto de vista dos seus utilizadores. A validação de usabilidade da aplicação *Web* é detalhada no capítulo 7.

A página seguinte mostra o diagrama que representa toda a estrutura relacional do *WAVY Operation Software* no final do desenvolvimento deste capítulo. As alterações estão apresentadas numa cor diferente para ser mais percetível.


Institutions
id Int
name String
acronym String
url String
adress String
phone String
Team
id Int
name String
code String
description String
userId User
Overlays
id Int
name String
filepath String
opacity Float
layers String
ObsProp
id Int
name String
definition String
description String
ModelProperty
id Int
definition String
obsType String
unit String
modelId Model
obsPropId ObsProp
User
id Int
username String
name String
settings JSON
active Bool
email String
institutionId Institution
Campaign
id Int
name String
description String
area Point
zoom Int
hide Bool
beginDate Data
endDate Date
public Bool
userId User
Type
id Int
name String
Model
id Int
name String
typeId Type
Equipment
id Int
name String
number String
serialNumber String
properties JSON
modelId Model
Derivedset
id Int
name String
description String
userId User
Launch
id Int
name String
description String
beginDate Date
endDate Date
campaignId Campaign
RealtimeObservation
id Int
serialNumber String
timestamp Date
position Point
content String
status String
BUILD
Observation
Id Int
timestamp Date
position Point
content JSON
datasetId Dataset
Dataset
id Int
name String
launchId Launch
equipmentId Equipment

Capítulo 6

Parâmetros de Espectro de Onda

A plataforma *WAVY Operation Software* foi desenhada para suportar operações simples, em que apenas existiriam dados transmitidos pelos derivadores flutuantes *WAVY*, em tempo real. Contudo, durante o decorrer do projeto MELOA surgiu a necessidade de integrar nestes derivadores um sistema de registos (ficheiros *Log*), para que posteriormente seja possível o calculo de parâmetros de Espectro de Onda, fundamentais para o estudo de correntes marinhas. Este capitulo descreve as funcionalidades desenvolvidas para possibilitar o cálculo desses parâmetros.

Os ficheiros *Log*, depois de submetidos na plataforma WOS, irão complementar os dados transmitidos em tempo real. A integração destes ficheiros não só vai possibilitar o calculo dos parâmetros pretendidos, mas também enriquecer a toda informação existente na plataforma, uma vez que estes contêm mais informação do que a transmitida em tempo real.

6.1 Sistema de registos

O processador incluído no derivador flutuante, corre um processo que recolhe os dados dos sensores e os transmite, em tempo real, para o Real Time Data Streamer. Paralelamente existe outro processo, não sincronizado com o anterior, que guarda os valores dos sensores na memória do derivador (ficheiro *Log*), a uma cadência superior, por volta dos 6Hz.

Depois de recolhido o derivador flutuante, o utilizador pode recorrer à aplicação *WAVY Hub* para extrair os ficheiros *Log*, em formato CSV, que pode posteriormente enviar para o *WAVY Operation Software*.

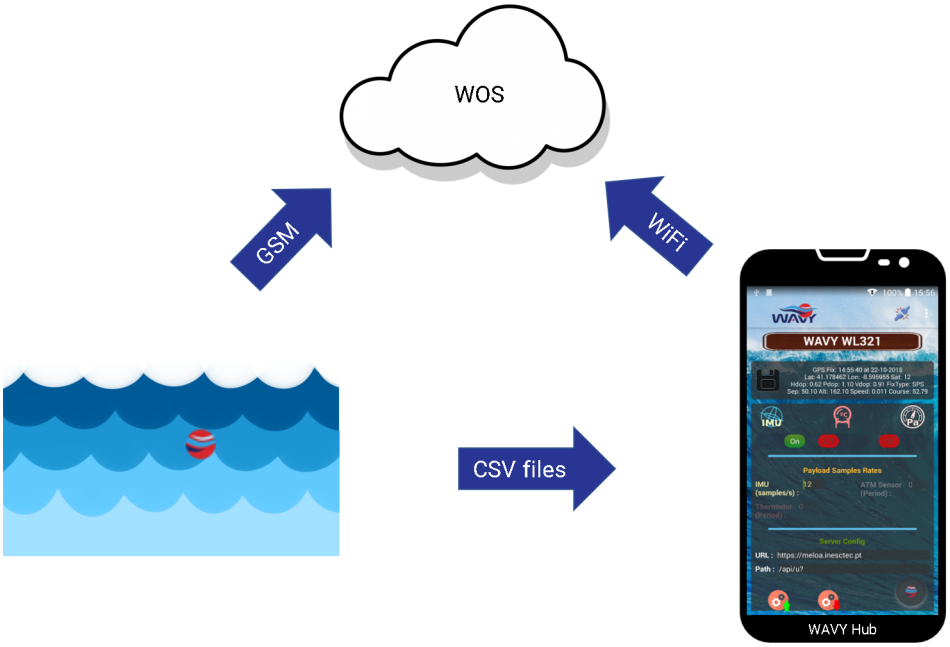


Figura 6.1: Diferentes métodos de envio de dados [18]

A integração dos ficheiros *Log* no *WAVY Operation Software* irá aumentar a quantidade de informação recolhida em cerca de 30 vezes, quando a cadência enviada em tempo real for uma observação a cada 5 segundos e a do sistema de registos a 6 observações por segundo (6Hz).

6.2 Implementação

6.2.1 Ficheiros *Log*

Uma vez carregado o ficheiro *Log*, através da aplicação *WAVY Hub*, o servidor irá registar os meta-dados relativos ao mesmo, para um posterior processamento. Para isso foi criado um modelo que associa os meta-dados a um derivador flutuante *WAVY*, através do seu número de série.

Depois de guardado o ficheiro, é necessário processá-lo de forma a extrair a informação nele contido. Para suportar os dados processados, foi criado um modelo auxiliar (*FileObservation*), semelhante ao modelo que suporta os dados em tempo real. O objetivo deste novo modelo é persistir os dados dos ficheiros *Log*, devidamente descodificados.

Da mesma forma que acontece nas mensagens em tempo real, são verificados se os valores de tempo e localização contém informação inválida. Caso existam erros, é adicionada uma *tag* de erro no campo *status*. Após esta analise dos ficheiros, são criadas e guardadas as observações, no formato apresentado na figura 6.2.

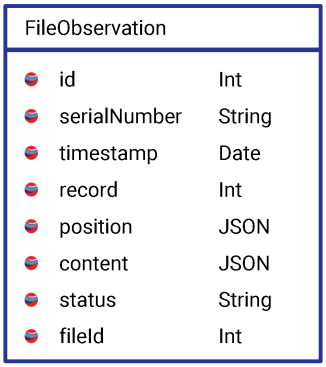


Figura 6.2: Esquema da tabela das observações contidas nos ficheiros

Como se pode ver, o tipo de dados do campo *content* difere do seu equivalente na tabela *realtime* (Figura 4.10), uma vez que são guardados dos valores em forma de objeto, de modo a facilitar todo o processo de obtenção dos parâmetros. Também é visível que existe um novo campo (*record*), que consiste num inteiro que indica a ordem das observações para o mesmo tempo. Como por definição são guardados seis registos para o mesmo segundo, este campo *record*, que assume valores entre 0 e 5, identifica a ordem cronológica em que os dados foram registados.

De frisar que o método de registar várias observações no mesmo segundo, com recurso à variável *record*, é uma especificação do fabricante dos derivadores flutuantes *WAVY*.

6.2.2 Processo de contextualização

O processo de contextualização será acionado quando: um lançamento termina, são inseridos novos dados ou existe uma alteração no intervalo de tempo do lançamento. Quando acionado, são selecionadas todas as observações relativas ao derivador, dentro do intervalo de tempo do lançamento, que por sua vez são relacionadas com um *dataset* previamente criado, como referido no capítulo 4. No entanto, como já referido, nesta fase de desenvolvimento, este processo de contextualização dos dados é ativado através de um botão existente na plataforma.

Após todas as observações, referentes aos ficheiros *Log*, estarem devidamente guardadas num modelo auxiliar (*FileObservation*), foi necessário alterar este processo de modo a integrar estas no modelo *Observation*. Anteriormente este processo consistia na obtenção e transformação das observações, provenientes da tabela *RealtimeObservation*, num objeto que posteriormente era guardado na tabela *Observation*. No entanto, com a inserção dos ficheiros de registos, é necessária uma fusão dos dados.

Antes de começar o desenvolvimento, foi necessário adicionar um novo campo (*record*) no modelo das observações, como se pode ver na figura 6.3.

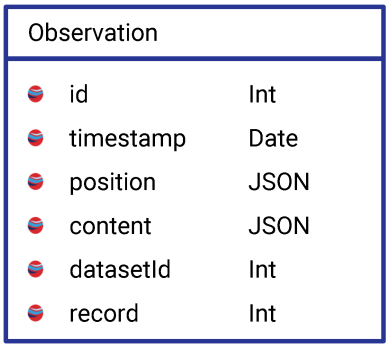


Figura 6.3: Esquema da nova tabela das observações

Primeiramente, através de um método do modelo dos *datasets* (*find or create*), é feita uma pesquisa pelo par dos identificadores do lançamento e do equipamento. Este método retorna um elemento, *dataset*, com as características indicadas, criando-o caso não exista na base de dados. Após esta operação é feita uma pesquisa pelos detalhes do lançamento onde é recolhido o intervalo de tempo do mesmo.

Em seguida, é feita uma pesquisa das observações na tabela *Observation*, relacionadas com o identificador do *dataset*. Depois de selecionadas, estas observações são guardadas em memória, num *array* com o nome de *existentObservations*.

Depois disto são também selecionados os dados da tabela de tempo real, correspondentes ao equipamento no intervalo de tempo do lançamento. Estes, são guardados num *array* designado de *realtimeObservations*. De frisar que o campo *content* deste modelo está no formato *string*, onde os parâmetros estão separados por barras verticais, como explicado no capítulo 4.

Como já explicado, não se sabe a que milésimo de segundo corresponde a observação em tempo real e os seis registos dos dados contidos nos ficheiros *Log*, no caso do derivador flutuante *WAVY Littoral*. Este aspeto foi discutido internamente no projeto, onde foi acordado relacionar a observação de tempo real com o registo (campo *record*) 0.

Antes de continuar este processo, são criados dois outros *arrays*: *update* e *create*. De seguida, são percorridas todas as observações contidas no *array realtimeObservations*, onde são transformadas em objeto, como anteriormente explicado, e adicionado o campo *record* com o valor 0. Enquanto isto, por cada observação é verificado se existe alguma correspondência no *array existentObservations*. Se existir, esta sofre uma fusão dos dados com a observação correspondente e é adicionada ao *array update*, caso contrário é adicionada ao *array create*. Após isto, são criadas ou atualizadas na tabela *Observation*, as observações correspondentes aos respetivos *arrays* (*create* e *update*).

Na segunda e última fase deste processo é feito um procedimento idêntico, porém com as observações dos registos dos ficheiros *Log*. São carregadas e comparadas, as observações da tabela *Observation* e *FileObservation*. Para finalizar são criadas as que ainda não existem e atualizadas as restantes.

Uma alternativa a este processo, seria apagar todas as observações provenientes da tabela *Observation* e executar posteriormente o processo de contextualização dos dados. Esta alternativa foi descartada pelo facto do modelo das observações (figura 6.3) conter um identificador único, auto incremental, que identifica cada observação. Como os valores tomados pelos inteiros no *PostgreSQL*, tem um limite de 32 *bits*, é um valor facilmente atingível quando lidamos com um grande volume dados, mais a possibilidade de apagar e inserir os mesmos.

6.2.3 Espectro de Onda

Como referido anteriormente, no decorrer do projeto MELOA surgiu a necessidade de calcular os parâmetros de Espectro de Onda. De forma a aproveitar o trabalho desenvolvido pelos investigadores do IH da Marinha Portuguesa, serão convertidos os *scripts* por eles desenvolvidos, em *Matlab*, de modo a ser possível integrar o cálculo destes parâmetros no *WAVY Operation Software*. Estes *scripts* incorporam algoritmos e processos complexos como *Lanczos*, *Simpson's rule*, entre outros.

Esta necessidade surgiu após a obtenção dos parâmetros de Espetro de Onda, no modelo oceânico. Este modelo de derivadores flutuantes *WAVY*, tem capacidade de processamento que permite calcular os parâmetros de Espetro de Onda *on board* pelo facto de, em comparação aos restantes modelos, ser maior, possibilitando a integração de um processador superior.

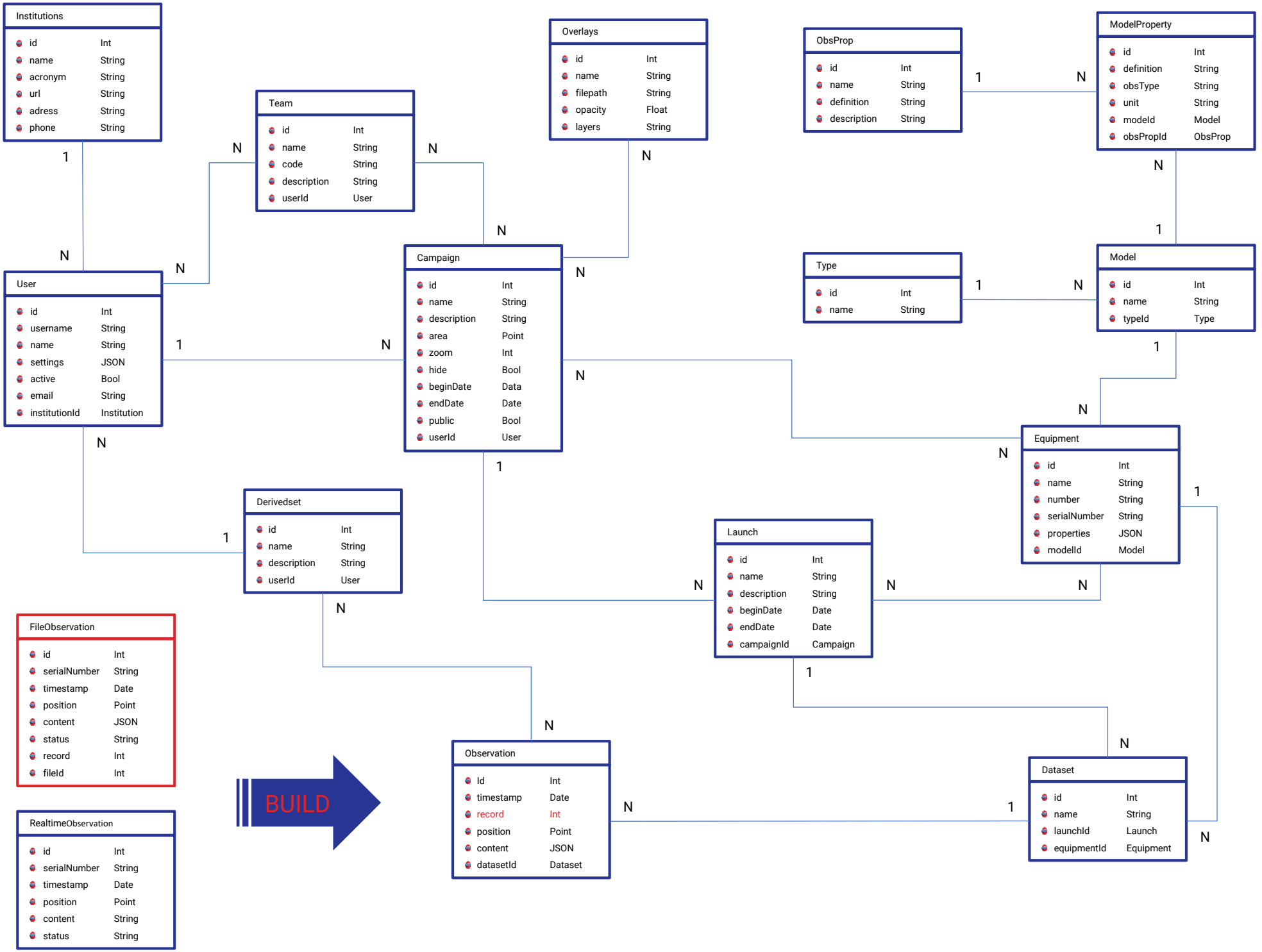
Não havendo a capacidade de processamento para calcular os parâmetros nos restantes modelos (básico e litoral), os *scripts* serão integrados no *WAVY Operation Software*. Desta forma, os *scripts Matlab* serão convertidos para *JavaScript*, para serem facilmente integrados e calcular parâmetros de Espectro de Onda, no WOS.

Os *scripts* tiram partido de diversas variáveis contidas nos ficheiros *Log*, sendo estas obtidas pelo sensor IMU dos derivadores flutuantes *WAVY*. Este sensor possui três acelerómetros, três giroscópios e três magnetómetros e permite uma taxa de amostragem de pelo menos 20Hz, o que, segundo os investigadores, é mais do que suficiente para calcular o Espectro de Onda.

Sendo assim, os *scripts* aceitam como *input* um conjunto de observações provenientes da tabela *FileObservation*, onde depois de executados, devolvem um conjunto de parâmetros de Espectro de Onda, entre os quais a altura máxima e significativa da onda, períodos médios e de pico, direção média e propagação.

6.3 Versão 1.2 do sistema WOS

Foram realizados testes com o objetivo de validar todo o trabalho desenvolvido ao longo deste capítulo. Estes testes, bem como os seus resultados, estão descritos no capítulo seguinte. O diagrama apresentado na próxima página representa a evolução de toda a estrutura do *WAVY Operation Software* no desenrolar deste capítulo (versão 1.2). As alterações estão apresentadas numa cor diferente para ser mais percetível.


Institutions
id Int
name String
acronym String
url String
adress String
phone String
Team
id Int
name String
code String
description String
userId User
Overlays
id Int
name String
filepath String
opacity Float
layers String
ObsProp
id Int
name String
definition String
description String
ModelProperty
id Int
definition String
obsType String
unit String
modelId Model
obsPropId ObsProp
User
id Int
username String
name String
settings JSON
active Bool
email String
institutionId Institution
Campaign
id Int
name String
description String
area Point
zoom Int
hide Bool
beginDate Data
endDate Date
public Bool
userId User
Type
id Int
name String
Model
id Int
name String
typeId Type
Equipment
id Int
name String
number String
serialNumber String
properties JSON
modelId Model
Derivedset
id Int
name String
description String
userId User
Launch
id Int
name String
description String
beginDate Date
endDate Date
campaignId Campaign
FileObservation
id Int
serialNumber String
timestamp Date
position Point
content JSON
status String
record Int
fileId Int
RealtimeObservation
id Int
serialNumber String
timestamp Date
position Point
content String
status String
BUILD
Observation
Id Int
timestamp Date
record Int
position Point
content JSON
datasetId Dataset
Dataset
id Int
name String
launchId Launch
equipmentId Equipment
1
N

Capítulo 7

Validação inicial

O objetivo do trabalho descrito nos capítulos anteriores é disponibilizar ferramentas que permitam a análise, visualização e curadoria da informação recolhida pelos derivadores flutuantes *WAVY* e também para calcular parâmetros de Especto de Onda.

No final deste capítulo são definidos novos objetivos para esta dissertação, o que implica uma segunda validação. No entanto, o propósito desta é avaliar a usabilidade da aplicação *Web*, com as funcionalidades implementadas nos capítulos anteriores, e criar um termo de comparação do processo de contextualização dos dados, onde este irá sofrer alterações e uma segunda validação.

7.1 Avaliação de usabilidade

Para verificar o impacto das alterações, a validação da usabilidade e a satisfação desta nova versão foi realizada através de um experimento que consistiu na utilização do sistema por parte dos investigadores que usufruem da plataforma *WAVY Operation Software*. No final de uma reunião do projeto MELOA, constituída por um grupo com cerca de 15 pessoas, os participantes foram convidados a preencher um questionário *on-line*, baseado no modelo de *Nielsen* [32], através da plataforma *Google Forms*. Das 15 pessoas, apenas 6 responderam e de forma anónima.

A figura 7.1 mostra os resultados agrupados pelas heurísticas de *Nielsen* da nova versão, que inclui a secção para curadoria de dados e as ferramentas de visualização incorporadas no capítulo 5.

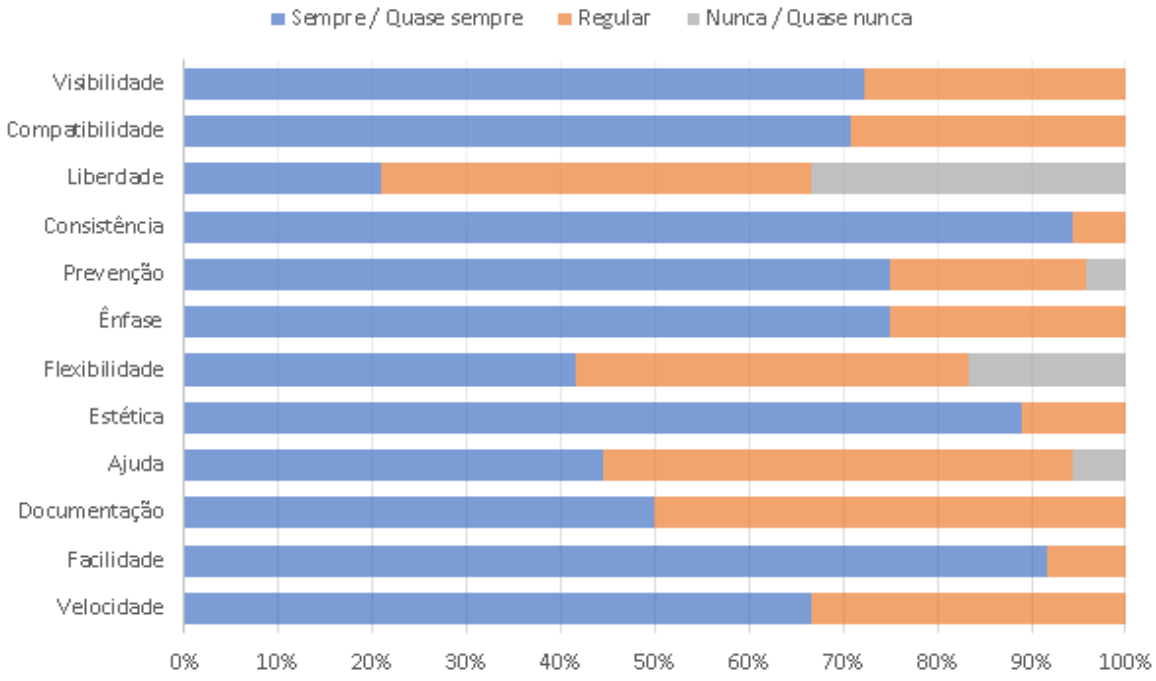


Figura 7.1: Gráfico de resultados da validação de usabilidade

Como fica patente no gráfico da figura 7.1, a usabilidade da plataforma *WAVY Operation Software* é razoavelmente adequada. Todas as heurísticas obtiveram bons resultados, à excepção da liberdade e controlo do utilizador. Este resultado menos bom, deve-se ao facto de não haver possibilidade de parar um processo de contextualização de dados e apagar *datasets*. Por outro lado, a plataforma mostra consistência, uma boa estética e facilidade na sua utilização.

O formulário também incorporava uma secção para sugestões e melhorias, no qual os investigadores propuseram a inclusão de anotações e mais documentação. O tema das anotações irá ser abordado nas futuras reuniões do projeto, de modo a perceber qual será a melhor abordagem a ter sobre este assunto.

7.2 Parâmetros de Espectro de Onda

A integração dos dados provenientes dos ficheiros *Log* na plataforma teve um grande impacto, no sentido em que houve um enorme aumento de informação (cerca de 30 vezes), que além de armazenada, pode passar por vários processos de contextualização.

Depois da integração dos novos dados no *WAVY Operation Software*, é possível os visualizar, juntamente com os transmitidos em tempo real, em mapas, tabelas e gráficos, tal como pretendido. Visto isto, o objetivo foi alcançado, mas no entanto surgiram novos desafios que são apresentados nas subsecções seguintes.

7.2.1 Melhoria no tempo de execução

Com a elevada quantidade de dados que agora existe, quando o processo de contextualização é executado, pode ser bastante demorado e em alguns casos pode mesmo sofrer *overflow*. A tabela 7.1 mostra os resultados dos vários testes realizados:

	REALTIME	FILE	TOTAL	BUILD	(RE)BUILD
1	491	18405	9361	4	19
2	553	0	553	0.8	2
3	2289	0	2289	2	4
4	43931	67841	64596	19	79
5	3625	175693	119745	45	OVERFLOW
ALL	2255034	2945974	1759853	900	OVERFLOW
	observations	observations	observations	seconds	seconds

Tabela 7.1: Resultados do processo de contextualização dos dados da versão 1.2

As colunas *REALTIME* e *FILE* contêm o número de observações, de tempo real e contidas nos ficheiros *Log* respetivamente, pertencentes a um equipamento num intervalo de tempo definido no lançamento. A coluna *TOTAL* representa o número das observações relacionadas com *dataset* no final do processo.

Os campos *BUILD* e *(RE)BUILD* representam ambos o tempo de execução deste processo, em segundos. O primeiro corresponde à primeira iteração do processo, enquanto que o segundo representam todas as iterações seguintes.

De realçar, que a diferença entre a primeira iteração e as seguintes, deve-se ao facto de na primeira não existirem dados na tabela *Observation* correspondentes ao *dataset* que vai ser processado.

Assim sendo, nas iterações seguintes já existem dados nessa tabela e todos eles são comparados, juntamente com os dados provenientes das tabelas *RealtimeObservation* e *FileObservation*. Este aumento de informação é refletido no tempo de processamento mostrado na tabela 7.1.

7.2.2 Remover dados duplicados

Ao longo da integração dos ficheiros de registos, na plataforma WOS, foram identificados valores duplicados provenientes dos ficheiros de registo. Essa duplicação tem origem em múltiplos carregamentos do mesmo ficheiro no servidor, através da aplicação *WAVY Hub*.

Quando o servidor recebe um ficheiro, o mesmo é descodificado e inserido na respetiva tabela, não é realizada nenhuma verificação que permita saber que se o mesmo já existe.

Estes dados duplicados, além de ocuparem memória, reduzem o desempenho de todo este processo, pois todos eles são comparados entre si. Como se pode ver na linha 1 da tabela 7.1, existe uma grande discrepância de observações, entre o numero total e provenientes de ficheiros. Depois de realizada uma verificação aos ficheiros recebidos pelo servidor, constatou-se que existem dois ficheiros iguais.

7.2.3 Tornar os dados coerentes

Com a realização de diferentes testes, notou-se que com alterações do intervalo dos lançamentos podem originar incoerências nos dados dos *datasets*.

Isto porque, no processo de contextualização dos dados, se o intervalo de tempo do lançamento for reduzido, as observações já registadas não serão verificadas. Este processo, fará com que as observações referentes ao intervalo de tempo, que foi removido do lançamento, persistam no *dataset*. Como explicado no final do capítulo 6, isto acontece porque os dados nunca são apagados.

7.3 Análise de resultados

Depois de apresentados os desafios anteriores correspondentes à nova versão do processo de contextualização dos dados, proveniente do objetivo inicial, desenvolver funcionalidades para o cálculo Espectro de Onda, foram definidos novos objetivos para este trabalho.

7.3.1 Objetivos

O elevado tempo de execução do processo de contextualização dos dados, vem acompanhado de um baixo desempenho de todo o processo. Este baixo desempenho torna a plataforma lenta, podendo a mesma ficar indisponível na eventualidade de um *overflow*.

Primeiramente pretende-se **proteger e garantir a fluidez em toda a plataforma**, de modo que este processo não tenha interferência com o funcionamento dos restantes serviços.

Posteriormente, quer-se ultrapassar os desafios apresentados: **melhorar o desempenho e tempo de execução deste processo; impedir que hajam dados duplicados e incoerentes**.

7.3.2 Abordagem

De modo a garantir a fluidez e evitar sobrecarga no *WAVY Operation Software*, **o processo de contextualização dos dados irá ser integrado num serviço independente e paralelo ao já existente**.

Para ultrapassar todos os desafios apresentados, serão estudadas **alternativas à base de dados relacional *PostgreSQL*, tendo em conta que estes se enquadram com o tema *Big Data***.

Capítulo 8

Infraestrutura tecnológica

Este capítulo é dedicado ao estudo das tecnologias que vão ser utilizadas na implementação da nova versão do processo de contextualização dos dados. Todas elas são *open source* e abrem portas à possibilidade de integrar novos serviços no projeto MELOA.

8.1 Servidor do WOS

Como explicado anteriormente, o sistema WOS é composto por dois ou mais componentes, dos quais se destaca a API com base na *framework Loopback*, desenvolvida em *NodeJS*. Este componente deverá estar sempre acessível, num servidor, de forma a alimentar os diversos clientes.

8.1.1 Virtualização

A virtualização de sistemas é um conceito recorrente que se baseia na representação, com base em *software*, de computadores.[8] É a maneira mais eficaz de aumentar a eficiência e agilidade para outros recursos.[1]

Dos vários tipos de virtualização que existem, todos procuramos o que melhor se adequa para o propósito. No entanto, os que são mais conhecidos pela sua robustez e confiabilidade são as máquinas virtuais e virtualização por *containers*.

8.1.2 *Virtual Machine* vs *containers*

As máquinas virtuais (*Virtual Machine* (VM)) foram projetadas para executar *software* em cima de servidores físicos para emular um sistema de *hardware* específico. Um *hypervisor* é um *software* que fica entre o *hardware* e a máquina virtual, sendo este necessário para virtualizar o servidor.

É possível correr várias VM num mesmo servidor físico, porém dentro de cada uma é executado um único sistema operativo (OS) independente e isolado, de modo que o que seja executado na mesma seja invisível para o servidor físico e vice-versa. Em suma, uma máquina virtual possui os seus próprios binários, bibliotecas e de certa forma é independente do servidor físico.[42] Contudo estas consomem muitos recursos da máquina original e como funcionam através de um OS por inteiro, por vezes a portabilidade para outras máquinas gera vários problemas.

A virtualização do OS cresceu nas últimas décadas[8] para permitir que o *software* funcione previsivelmente e bem quando movido entre servidores. No entanto os *containers* fornecem uma maneira de executar esses sistemas isolados num único sistema operativo.

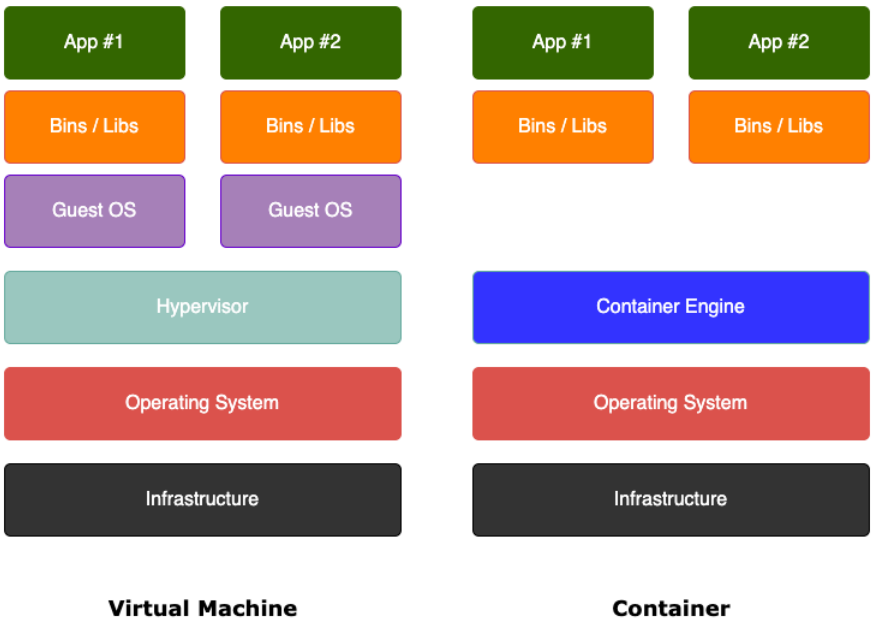


Figura 8.1: *Virtual Machine* vs *container* [27]

Como se pode ver na figura 8.1, o *software container* trabalha sobre um sistema operativo, onde cada um deles compartilha o kernel do OS. Componentes partilhados são somente leitura. Assim, os *containers* são excepcionalmente "leves" e levam apenas alguns segundos a iniciarem.[26]

Os *containers* também reduzem o trabalho de gestão. Como partilham um sistema operativo comum, apenas este precisa de cuidados, pois os *containers* são serviços otimizados e que apenas usufruem de bibliotecas necessárias para o seu funcionamento.[11] Muitas vezes, estas bibliotecas são instaladas no OS para minimizar reinstalações e desperdício de memória.

De acordo com um estudo recente da *451 Research*, a adoção de *containers* irá crescer 30% por ano até 2022. Os *containers* estão a facilitar o desenvolvimento rápido e ágil como nunca antes.[39]

8.1.3 *Docker Engine*

Docker Engine é uma tecnologia *open source* para criação e gestão de *containers*. É sobre ela que prevalece todo o sistema *WAVY Operation Software*. Assim sendo, neste ponto existem três *containers* para todo o sistema funcione nas melhores condições:

- *Nginx*

  A tecnologia *open source Nginx* é um intermediário que mapeia internamente todos os pedidos recebidos para o serviço correspondente.

- *PostgreSQL*

  Este *container* é exclusivo para a base de dados do *WAVY Operation Software*, apresentada no capítulo 4.

- *NodeJS*

  Para correr todo o desenvolvimento feito e disponibilizar a API do WOS, é usado este terceiro *container* que tem como base o interpretador *NodeJS* sob uma distribuição *Alpine* (*Linux*), juntamente com o *software Process Manager 2* (PM2). Este último, tem como objetivo facilitar o desenvolvimento de aplicações *NodeJS*, pois, entre outras, tem a capacidade de reiniciar automaticamente os processos à medida que ocorrem alterações no código fonte.[4]

8.1.4 Escalonador de processos

Para prevenir falhas, sobrecarga e dar suporte a um escalonador de processos foram integrados mais dois *containers*. Um deles, é igual a um já existente (*NodeJS*) que integrará, com ajuda da *framework Agenda*, o escalonador de processos, para que tarefas mais densas sejam realizadas neste novo serviço. O outro, uma base de dados *MongoDB*, foi criado para suportar a *framework Agenda*, sendo possível obter da mesma uma listagem dos processos e seus estados.

Feito isto, o processo de contextualização dos dados foi migrado para o novo *container*, que garante a fluidez em todos os serviços do WOS.

8.2 Base de dados e *Big Data*

Conceptualmente, o termo *Big Data* é caracterizado por um enorme volume e velocidade de dados, com uma grande variedade (diversidade) de tipos e fontes de dados, e pelos rigorosos requisitos de veracidade de dados (fidelidade).[22]

Com este conceito e supondo dezenas de derivadores flutuantes *WAVY* a transmitir e guardar dados, facilmente se percebe que este desafio de processamento de dados se enquadra em *Big Data*. Neste mundo, as soluções para combater a enorme variedade de tipos de dados que existem, todas elas são desenhadas para fazer frente a um problema especifico. É comum fundir várias destas soluções para resolver, em alto nível, um desafio de determinado produto ou serviço.[14]

Contudo, neste trabalho, os dados provenientes dos derivadores flutuantes *WAVY* podem ser reduzidos a uma estrutura simples, que são dados numéricos ou *strings* ao longo do tempo. Exatamente por esta possibilidade de os estruturar de uma maneira mais simples, a solução para este processamento é outra. Sendo assim, a principal necessidade é a escolha de uma base de dados em que insira e retorne os dados de uma forma rápida, mas também que os suporte em grandes quantidades.

8.2.1 Base de dados relacionais

Existe uma vasta família de base de dados, sendo no momento as relacionais e objeto-relacionais mais conhecidas e usadas, onde esta última uma evolução da primeira.

Um modelo relacional é um tipo de base de dados que implementam a visão de várias tabelas, com colunas (propriedades e atributos) e linhas (dados). Estas tabelas interrelacionam-se entre si através de chaves estrangeiras, criando assim relações de fácil acesso entre dados. Outra característica deste tipo de base de dados, é a obrigatoriedade de uma estrutura previamente definida.[35]

Enquanto que o modelo relacional apenas suporta tipos dados simples (*boolean*, *int*, etc.), o modelo objeto-relacional, como o próprio nome indica, aceita dados do tipo objetos. O WOS usa a base de dados relacional *PostgreSQL* para persistência dos dados transmitidos pelos derivadores flutuantes, assim como os dados relacionados com o planeamento de campanhas e lançamentos.

8.2.2 Base de dados NoSQL

As base de dados relacionais encontraram grandes desafios com a evolução e necessidade de trabalhar com *Big Data*, onde são requeridos: o consumo de uma ampla variedade de dados a grandes velocidades, estruturas dinâmicas e suporte para grandes volumes.[17]

Com a prática para encontrar soluções para estes problemas, diversas organizações desenvolveram as suas próprias tecnologias, como a *Apache* com o *HBase*, *Amazon* com o *Dynamo* e *Google* com o *BigTable*. Assim o *Not Only SQL* (NoSQL) encontrou muitos casos de uso e atualmente é o alicerce dessas empresas.[14]

8.2.3 Base de dados de séries temporais

Uma base de dados de séries temporais é uma base de dados otimizada para dados com registo de data e hora, geralmente baseadas em NoSQL.[31] Os dados de séries temporais são simplesmente medições ou eventos que são monitorizados e agregados ao longo do tempo. Este modelo está otimizado para lidar com grandes volumes de dados e é muito usado para guardar conteúdo produzido por sensores, principalmente aqueles gerados pelos IOT.

Estes modelos são caracterizados pela alteração do conceito tabela para medições (*measurements*), pois simplesmente representa uma lista de valores ao longo do tempo, e o forte *design* voltado para uma resolução simplificada de conflitos.[14]

8.2.4 Seleção

Como já referido, o *WAVY Operation Software* tira partido da base de dados *MongoDB*, modelo NoSQL, para armazenar dados referentes ao escalonador de processos. Uma das hipóteses que surgiu, foi a utilização desta base de dados, tendo a vantagem de já estar a ser usada neste software. No entanto, esta opção foi descartada pelo facto dos derivadores flutuantes *WAVY* registarem os dados numa linha temporal, o que nos leva a optar por uma base de dados baseada em séries temporais em vez de uma NoSQL. E segundo um estudo[12], *InfluxDB* (base de dados de séries temporais) tem significativamente um melhor desempenho quando comparada com *MongoDB*.

Além do estudo acima referido, de acordo com a página de classificações de base de dados[43], que dispõe de um complexo algoritmo citado no próprio *website*, na secção de modelos de séries temporais, *InfluxDB* ocupa o primeiro lugar da tabela. Assim sendo, esta base de dados foi a escolhida para armazenar os dados recolhidos pelos derivadores flutuantes *WAVY*.

8.2.5 InfluxDB

InfluxDB é uma base de dados de séries temporais *open source*. Lançada em 2013 e implementada em *GO*, esta está otimizada para armazenar e disponibilizar informação de uma forma rápida, eficaz e facilmente escalável.[34] Ao contrário de outras bases de dados deste tipo, *InfluxDB* suporta uma linguagem semelhante a *SQL*, tipos de dados numéricos e sequências de caracteres (*strings*). Não obriga a definir esquemas antecipadamente, o que possibilita serem adicionados novos campos em qualquer altura.

Um registo de data e hora identifica um único ponto em qualquer série de dados, semelhante a uma chave primária num modelo relacional. Além do tempo, existem outras colunas que são internamente indexadas, denominadas de *tags* que são exclusivamente do tipo *string* e só podem tomar 100 mil valores diferentes para cada tabela (este limite pode ser alterado). As restantes colunas são chamadas de *fields*.[24]

Outras vantagens que *InfluxDB* tem para oferecer, é uma grande variedade de funções de agregação, aceita *querys* com expressões regulares e conta com um grande suporte para a tecnologia *Grafana*. O ponto fraco desta tecnologia é a falta de suporte oficial da sua integração com *LoopBack*, o que irá requerer um maior esforço de desenvolvimento na integração com WOS.

8.3 Grafana

O *Grafana* é um *software* de visualização de dados interativa, baseada em painéis e rico em recursos. Suporta muitas fontes de dados, porém tem uma relação forte, juntamente com um grande suporte, com *InfluxDB*.[7]

Esta tecnologia permite criar diferentes painéis, todo o tipo de gráficos com a possibilidade de os ajustar à visualização pretendida, formas de visualização dos dados e diversas operações, de modo a garantir uma boa organização. Além disto, quando ligado ao *InfluxDB*, realiza pedidos com qualquer iteração que seja feita em gráficos, gerindo automaticamente os dados consoante a escala que se esteja a visualizar. Com a ligação direta ao *InfluxDB* é possível tirar proveito das funções de agregação que esta base de dados incorpora.

Depois de analisada, houve um grande interesse pela parte da coordenação do projeto em usufruir desta tecnologia, que leva à possibilidade de desenvolver um novo serviço com base em *Grafana*.

Capítulo 9

Contextualização de dados

No capitulo 8 foi descrita a migração do processo de contextualização para um serviço independente. No entanto, é necessário melhorar o desempenho e ultrapassar os novos desafios apresentados no capitulo 7.

Para atingir os novos objetivos, será integrado um modelo de base de dados de séries temporais, *InfluxDB*, que ajudará a lidar com o aumento de informação, juntamente com melhorias no processo de contextualização dos dados. Contudo, apenas alguns modelos serão migrados, sendo eles: *RealtimeObservation*, *FileObservation* e *Observation*. Posteriormente será necessária uma solução que permita representar a relação entre observações e *derived datasets*, visto que ambos os modelos pertencem a diferentes bases de dados.

9.1 Instalação

Para esta nova base de dados (*InfluxDB*) foi adicionado no projeto um novo *Docker container*, de modo a estabelecer uma ligação com os restantes *containers* em funcionamento. Depois de a mesma estar operacional e dentro do sistema *WAVY Operation Software*, através de um modulo do *InfluxDB* para *NodeJS*, foi realizada a comunicação entre este componente e a API.

De modo a disponibilizar a ferramenta *Grafana*, como referido no capítulo capitulo 8, foi adicionado um novo software *container*, de modo que ao longo da integração do *InfluxDB* no sistema WOS, sejam realizados estudos para o uso do *Grafana*.

9.2 Migração para *InfluxDB*

A *framework LoopBack* não possui nenhum conetor que suporte a ligação com a base de dados *InfluxDB*. Posto isto, foram criadas as funções e respectivas rotas que permitem as operações necessárias ao funcionamento dos modelos.

Foram também criados *scripts* para migrar os dados dos dois modelos *PostgreSQL*, *RealtimeObservation* e *FileObservation*, para *InfluxDB*. Depois de realizada a migração e mantendo a mesma estrutura de dados percebeu-se que, como a base de dados *InfluxDB* não permite entradas duplicadas, houve uma redução de cerca de um milhão de entradas na tabela *FileObservation*, comparativamente com a mesma no *PostgreSQL*. Como o *InfluxDB* substitui as entradas repetidas, o desafio de evitar duplicação de dados foi superado com sucesso.

Como *InfluxDB* não suporta objetos *JavaScript*, a alternativa é guardar os campos *content* e *position*, da tabela *FileObservation* (figura 6.2), numa *string* em formato JSON.

O passo seguinte foi alterar o processo de contextualização dos dados, de forma a substituir os modelos migrados da base de dados relacional, *PostgreSQL*, para a base de dados de séries temporais, *InfluxDB*.

Uma vez que o *InfluxDB* não suporta identificadores auto incrementáveis, optou-se por utilizar um identificador único universal (UUID), para que seja possível identificar cada entrada. Para gerar este identificador único é utilizado um módulo para *NodeJS*, em que usa o tempo em nanossegundos juntamente com outros cálculos aleatórios, cria um valor de 128 *bits* em 32 dígitos hexadecimais. [6] Deste modo, cada observação da tabela *Observation* passa a ter um identificador do tipo *string*.

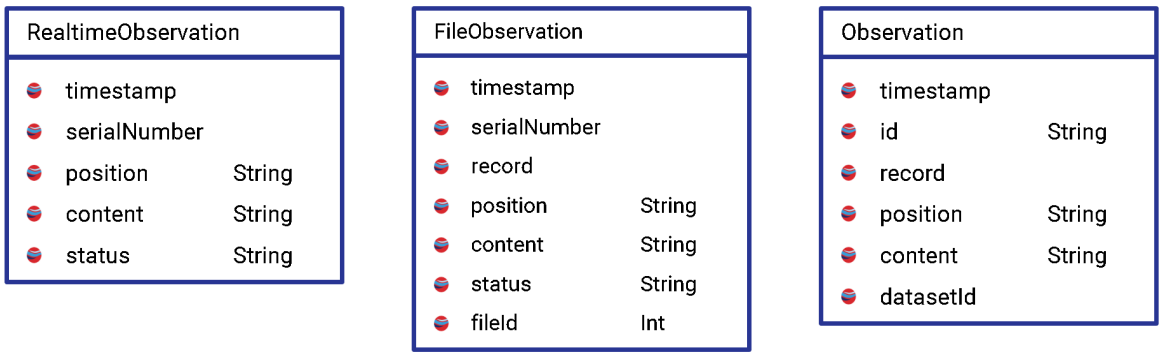


Figura 9.1: Modelos das tabelas do *InfluxDB*

Na figura 9.1, percebemos que alguns campos não têm qualquer tipo dados associado. Estes campos, à exceção do *timestamp*, são *tags*. Como já explicado, estes campos são internamente indexados e forçosamente do tipo *string*. Se repararmos, ao juntar as *tags* com o *timestamp*, obtemos todos os dados que identificam uma única observação. Isto é, não existe mais do que uma observação que tenha todos estes valores iguais. Se isto acontecesse, significava que os dados seriam duplicados e eram substituídos pelo *InfluxDB*.

O campo *content*, que contem toda a informação recolhida pelos derivadores flutuantes *WAVY*, é constituído por uma *string* no formato JSON. Como as funções de agregação do *InfluxDB* e o *Grafana* não suportam os dados neste formato, originou um novo desafio.

9.3 Decomposição do campo *content*

Para superar este novo desafio e possibilitar a utilização das funções de agregação, a solução passou por decompor o objeto *content* e introduzir cada parâmetro num novo campo na base de dados. Deste modo e para que posteriormente se possa distinguir os parâmetros recolhidos dos restantes, foi adicionado o caráter '_' no inicio de cada campo, como mostra a figura 9.2.

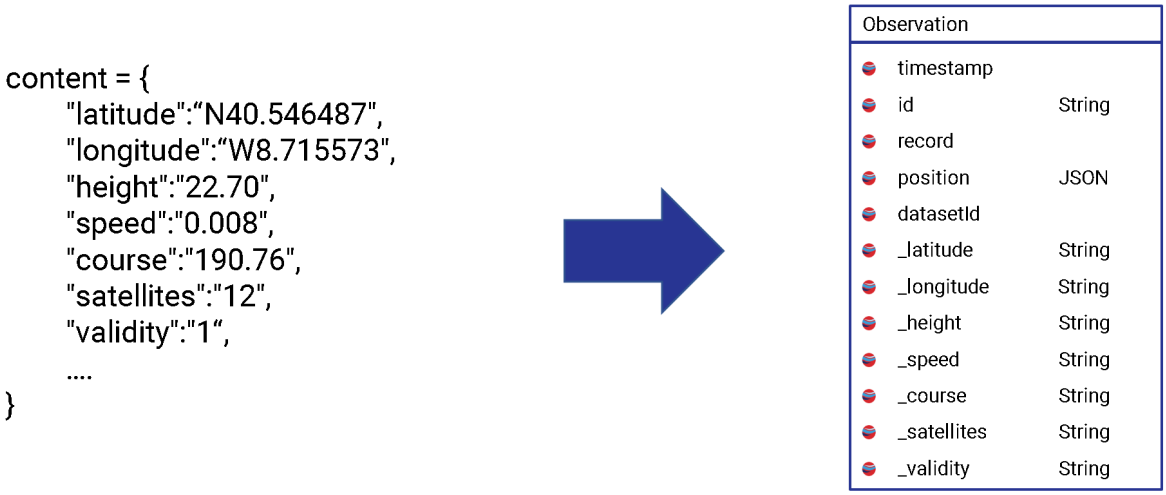


Figura 9.2: Esquema optado para decompor o campo *content*

O processo de contextualização de dados foi reestruturado, de modo a eliminar dados incoerentes. Depois de uma análise ao processo concluiu-se que seria possível reduzir o numero de operações realizadas, aumentando assim o desempenho do mesmo. Assim sendo e aproveitando as capacidades da base de dados *InfluxDB*, quando este processo é acionado, toda a informação referente ao *dataset* é eliminada, o que não acontecia anteriormente. Como o identificador do *dataset* na tabela das observações é uma *tag* e o *InfluxDB* está otimizado para lidar sob estas, rapidamente se apagam todas as observações relacionadas com esse identificador.

O processo de contextualização de dados depois de adicionados os ficheiros *Log*, como explicado no final do capítulo 6, resumidamente consistia nos seguintes passos:

1. São selecionados todos os dados da tabela *RealtimeObservation*
2. São selecionados todos os dados da tabela *Observation*
3. Estes dados são comparados e agregados em dois arrays
4. Posteriormente são criadas ou atualizadas as observações na tabela *Observation*
5. Numa segunda iteração, são selecionados todos os dados da tabela *FileObservation*
6. São também selecionados todos os dados da tabela *Observation*
7. Estes dados são comparados e agregados em dois arrays
8. No final são criadas ou atualizadas as observações na tabela *Observation*

Como facilmente se percebe, há um conjunto de passos repetidos e outros agora desnecessários. Visto isto, nesta nova versão desaparecem: os passos 2 e 6 , pelo facto de no inicio deste processo todos os dados relacionados com o *dataset* serem apagados; o passo 4, pois os dados já estão carregados em memória e pelo desaparecimento do passo 6. Assim, este processo torna-se mais simples e menos demorado com estas alterações.

Como se pode verificar na figura 9.2, todos os parâmetros foram adicionados à base de dados em formato *string* para de modo a proporcionar o maior dinamismo ao sistema. Contudo, esta alteração não é suficiente para permitir o funcionamento das funções de agregação e do *Grafana*, o que originou um novo desafio.

9.4 Alteração dos tipos de dados

Este novo desafio foi superado com facilidade. Foram definidos quais os campos que são obrigatoriamente *string*, nomeadamente a latitude e longitude, enquanto que os restantes passam a ser do tipo numérico, *float*, como se pode ver na figura 9.3. Optou-se por adicionar todos os restantes valores a este tipo de dados para prevenção de erros. Caso existam alterações na transmissões de dados

Quando se insere um novo campo em *InfluxDB*, este assume automaticamente o tipo de dados em que o mesmo é enviado. No entanto, se posteriormente tentar inserir nesse mesmo campo um tipo de dados diferente, o *InfluxDB* adapta o valor, no caso de valores numéricos, ou retorna um erro, nos restantes tipos de dados.

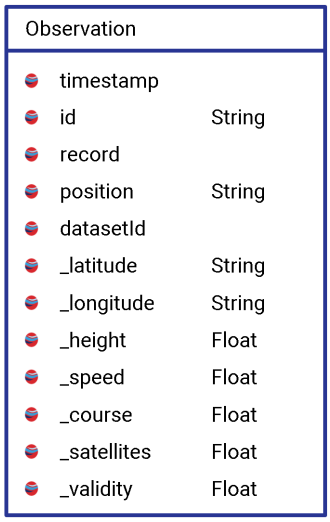


Figura 9.3: Esquema da tabela das observações com novos tipos de dados

Depois das alterações realizadas, é necessário proceder à relação destes dados como os *derived datasets*. Como anteriormente estes dependiam de relações entre os modelos *Observation* e *Derivedset*, no *PostgreSQL*, e este primeiro modelo foi migrado para *InfluxDB*, foram realizados vários testes.

Nesta fase, optou-se por relacionar estes dois modelos, presentes nas duas base de dados, através do UUID e do identificador da tabela *DerivedSet*. Isto consiste numa tabela intermédia, criada na base de dados *PostgreSQL*, que relaciona um *derived dataset* e uma observação, como se pode ver na figura 9.4).

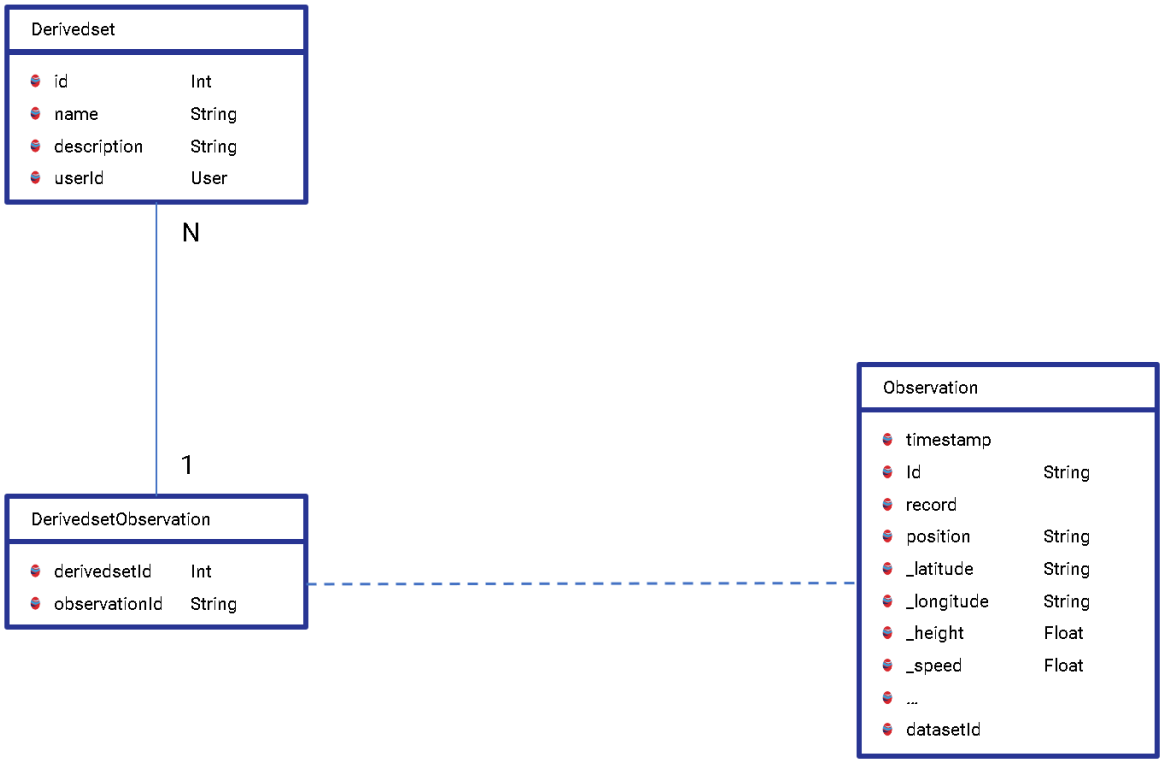


Figura 9.4: Simulação de relação entre diferentes base de dados

Foram então refeitas as rotas de inserir e consultar observações relativas aos *derived datasets*. Na operação de relacionar uma observação com um *derived dataset*, o *WAVY Server* recebe os identificadores das observações pretendidas por parte do cliente. No entanto, depois de receber os identificadores, é necessário confirmar se os mesmos existem. Através de uma *query* à tabela *Observation* do *InfluxDB*, facilmente se confirma a veracidade das observações e se podem ser inseridas na tabela *DerivedsetObservation* no *PostgreSQL*. Já na operação de consulta, é realizada uma pesquisa na tabela *DerivedsetObservation*, no *PostgreSQL*, pelo identificador de um *derived dataset*, no qual este devolve todos os identificadores das observações. Depois disto, é construída uma *query* para obter todas as observações que têm o identificador igual a algum dos obtidos.

Depois de implementadas as devidas alterações e reestruturadas as rotas de inserir e consultar observações, relativas aos *derived datasets*, realizaram-se alguns testes que são apresentados na tabela 9.1.

	VERSION 1.2		VERSION 2.3
25	2	INSERT	12
25	0.5	GET	32
50	3	INSERT	25
50	1	GET	58
observations	seconds	operation	seconds

Tabela 9.1: Resultados das operações de consulta e inserção através de ”relações”

A versão 1.2 da tabela é referente ao estado do servidor na primeira validação. Já a versão 2.3 corresponde à implementação atual, com uma "relação"entre as diferentes base de dados. Como se pode verificar, as operações são excessivamente demoradas e isso deve-se a vários fatores:

- O identificador das observações ser um campo normal e não uma *tag*. De realçar que o identificador não pode ser uma *tag* pelo facto de nunca tomar valores repetidos. O objetivo das *tags* é possibilitar a agregação de dados através das mesmas e deve-se evitar que as mesmas sejam muito variadas, como o que acontece com identificadores. [30]

- Estão a ser usadas duas base de dados diferentes para as operações.

9.5 *InfluxDB* e expressões regulares

De forma a melhorar a integração dos *derived datasets*, foi implementada uma nova abordagem. Na secção anterior foi evidente que este processo não pode envolver muitas operações e diferentes base de dados. Nesta fase foi testada a integração através de expressões regulares no *InfluxDB*.

Esta implementação consiste em adicionar um novo campo na tabela *Observation* no *InfluxDB*, designado de *derivedsetIds*. Este campo, do tipo *string*, contem a sequência de identificadores de *derived datasets* com que a observação se relaciona, separados por barras verticais, como se pode verificar na figura 9.5.

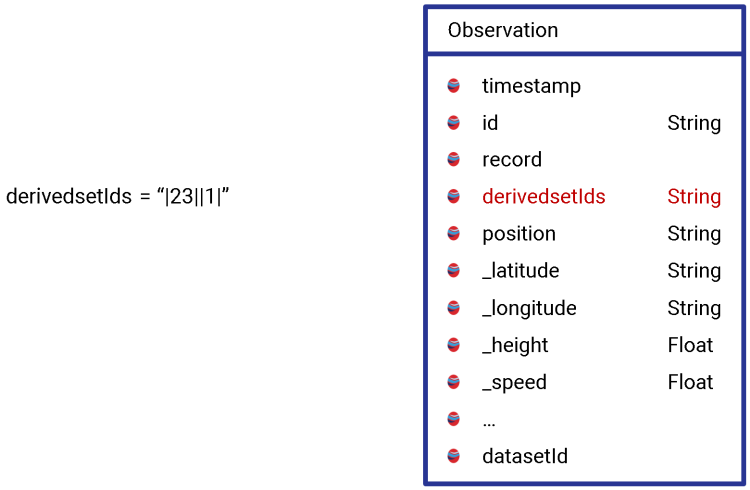


Figura 9.5: Implementação de expressões regulares

Esta abordagem permite criar uma expressão regular, com a qual se consegue obter todas as observações relacionadas com um determinado *derived dataset*. No entanto, para inserir o identificador do *derived dataset* é necessário obter a observação, adicionar o identificador ao novo campo (se este não existir) e atualizar essa informação na base de dados. A tabela 9.2, mostra os resultados desta implementação.

	VERSION 1.2		VERSION 2.4
25	2	INSERT	8
25	0.5	GET	3
50	3	INSERT	15
50	1	GET	5
observations	seconds	operation	seconds

Tabela 9.2: Resultados das operações de consulta e inserção através de expressões regulares

Como referido anteriormente, a versão 1.2 da tabela é referente ao estado do servidor na primeira validação, enquanto que a versão 2.4 corresponde à implementação atual, através de expressões regulares. Concluí-se então que através de expressões regulares foi possível reduzir o tempo de execução das operações, quando comparadas com a abordagem anterior. No entanto, os tempos de execução não são satisfatórios, pelo que de seguida se irá optar por uma abordagem alternativa.

9.6 Escalabilidade do *InfluxDB*

Como explicado na segunda secção deste capítulo, todas as *tags* juntamente com o *timestamp*, representam um identificador de uma única observação. Deste modo, foi necessário adaptar a aplicação *Web*, de maneira a relacionar observações com *derived datasets*. Nesta nova forma de relacionamento, em vez de enviado o identificador da observação correspondente, passam a ser enviados os campos *tag* e *timestamp*.

Feito isto e de modo a superar o desafio da demora das operações, optou-se por criar uma nova tabela, em *InfluxDB*, para guardar todas as observações relacionadas com *derived datasets*. Esta abordagem de incorporar uma nova tabela foi utilizada em último recurso, pelo facto de duplicar a informação existente. Como *InfluxDB* é uma base de dados não relacional e uma vez que as observações já estão guardadas, a única opção é armazená-las novamente, adicionando um campo *tag* que as relaciona com um *derived dataset*, como mostra a figura 9.6.

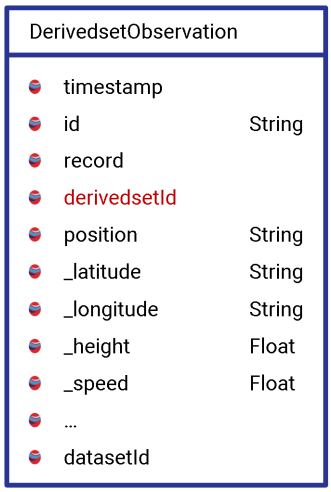


Figura 9.6: Nova tabela para as observações de *derived datasets*

Juntamente com a criação da tabela anterior e as devidas alterações para as operações de inserir e consultar, foi desenvolvida a opção para desassociar observações de *derived datasets*, através da remoção das mesmas na tabela *DerivedsetObservation*. No final desta implementação, foram realizados testes para validar esta implementação.

	VERSION 1.2		VERSION 2.5
25	2	INSERT	0.2
25	0.5	GET	0.1
25	0.5	DELETE	6
50	3	INSERT	0.3
50	1	GET	0.2
50	2	DELETE	15
observations	seconds	operation	seconds

Tabela 9.3: Resultados de todas as operações através de uma nova tabela

Após a análise da tabela 9.3, é visível que esta implementação, nas operações de inserir e consultar observações, foi a que obteve melhores resultados. No entanto, as operações de apagar são demasiado demoradas, o que requer uma melhoria neste caso. O *InfluxDB* está otimizado para inserir e consultar dados a grandes velocidades e escalas, não tendo as mesmas capacidades para as operações de apagar. Essas operações são demoradas e criaram um novo desafio.

De acrescentar que o *InfluxDB* incorpora uma *query* que possibilita apagar rapidamente dados através de uma única *tag*, como acontece no inicio do processo de contextualização de dados. No entanto, para apagar uma única observação, são necessárias múltiplas *tags* e por este motivo torna as operações mais demoradas.

9.7 Escalonador de processos

Como referido anteriormente, o *InfluxDB* está otimizado para inserir e consultar dados a grandes velocidades. Para solucionar este desafio, da demora das operações de apagar, foi inserido um novo campo, *visible* que toma os valores 0 e 1, onde indica se a observação está marcada para ser apagada. Deste modo, sempre que uma observação for adicionada, este campo toma o valor 1, por definição.

No entanto, para que não exista informação duplicada a não ser usada e tirando proveito do escalonador de processos, sempre que se altera o valor do campo *visible* para 0, é acionado um processo, no escalonador, que apagará a mesma da tabela *DerivedsetObservation*. A tabela 9.4 mostra os resultados desta implementação.

	VERSION 1.2		VERSION 2.6
25	2	INSERT	0.2
25	0.5	GET	0.1
25	0.5	DELETE	0.3
50	3	INSERT	0.3
50	1	GET	0.2
50	2	DELETE	0.5
observations	seconds	operation	seconds

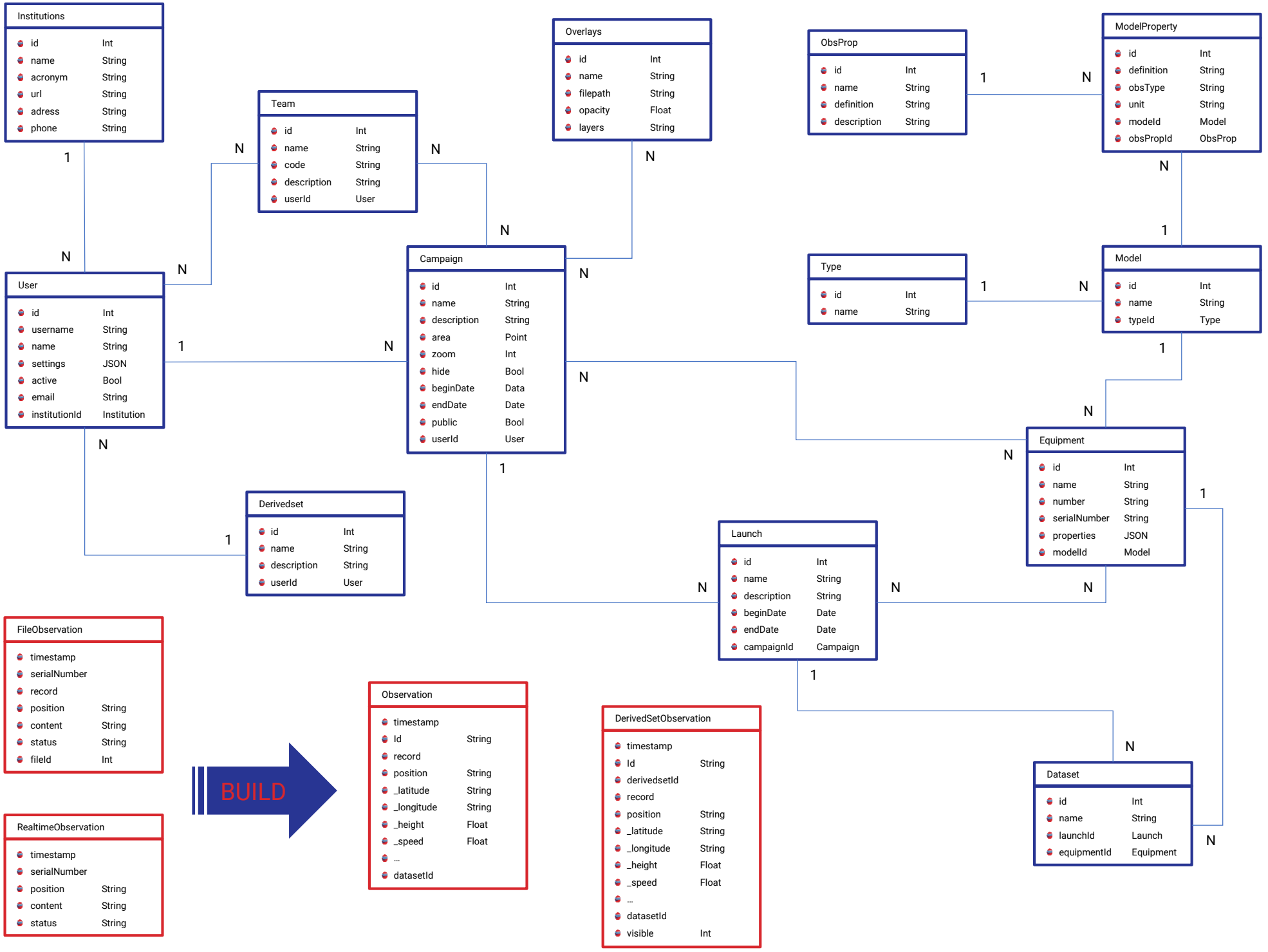
Tabela 9.4: Resultados de todas as operações com a nova tabela e o escalonador

A tabela traduz uma melhoria significativa dos tempos de execução das operações, quando comparadas com o seu estado inicial. Podemos assim concluir que os desafios foram superados com sucesso, melhorando o tempo de execução das operações e tornando a aplicação *Web* do *WAVY Operation Software* mais fluida.

9.8 Versão 2.6 do sistema WOS

Foram realizados testes com o objetivo de validar todo o trabalho desenvolvido ao longo deste capítulo. De modo a ser possível comparar a versão 2.6 com a versão 1.2, foram realizados os mesmos testes demonstrados no capítulo capítulo 7, que estão descritos no capítulo seguinte.

O diagrama apresentado na página seguinte, mostra todas as mudanças que ocorreram no sistema *WAVY Operation Software* com desenrolar deste capítulo (versão 2.6). Os modelos presentes na base de dados *InfluxDB* estão representados numa cor diferente, de modo a ser mais percetível.


Institutions
id Int
name String
acronym String
url String
adress String
phone String
Team
id Int
name String
code String
description String
userId User
Overlays
id Int
name String
filepath String
opacity Float
layers String
ObsProp
id Int
name String
definition String
description String
ModelProperty
id Int
definition String
obsType String
unit String
modelId Model
obsPropId ObsProp
User
id Int
username String
name String
settings JSON
active Bool
email String
institutionId Institution
Campaign
id Int
name String
description String
area Point
zoom Int
hide Bool
beginDate Data
endDate Date
public Bool
userId User
Type
id Int
name String
Model
id Int
name String
typeId Type
Equipment
id Int
name String
number String
serialNumber String
properties JSON
modelId Model
Derivedset
id Int
name String
description String
userId User
Launch
id Int
name String
description String
beginDate Date
endDate Date
campaignId Campaign
Dataset
id Int
name String
launchId Launch
equipmentId Equipment
FileObservation
timestamp
serialNumber
record
position String
content String
status String
fileId Int
RealtimeObservation
timestamp
serialNumber
position String
content String
status String
BUILD
Observation
timestamp
Id String
record
position String
_latitude String
_longitude String
_height Float
_speed Float
...
datasetId
DerivedSetObservation
timestamp
Id String
derivedsetId
record
position String
_latitude String
_longitude String
_height Float
_speed Float
...
datasetId
visible Int

Capítulo 10

Validação e conclusão

10.1 Validação final

O objetivo do trabalho descrito nos capítulos anteriores é melhorar o processo de contextualização dos dados, de modo a tornar todo o sistema *WAVY Operation Software* mais fluido, garantindo que este processo não tenha interferência com o funcionamento dos restantes serviços. Depois de apresentadas e implementadas estas melhorias, este capítulo descreve todos os testes realizados para avaliar se esse objetivo foi alcançado.

De modo a avaliar o trabalho desenvolvido no processo de contextualização de dados, foram realizados os mesmos testes apresentados no capítulo 7. Esse capitulo, serve como termo de comparação nos desenvolvimentos que lhe sucederam. Desta forma, os resultados dos testes realizados são apresentados na tabela 10.1.

	REALTIME	FILE	TOTAL	BUILD	(RE)BUILD
1	491	9305	9361	1	1
2	553	0	553	0.2	0.4
3	2289	0	2289	0.4	0.9
4	43891	25827	64596	9	9
5	3623	119745	119745	22	22
ALL	3017573	1762359	1839422	302	341
	observations	observations	observations	seconds	seconds

Tabela 10.1: Resultados do processo de contextualização dos dados da versão 2.6

As colunas *REALTIME* e *FILE* apresentam a contagem de observações, em tempo real e presentes nos ficheiros *Log* respetivamente, pertencentes a um equipamento num intervalo de tempo definido no lançamento. A coluna *TOTAL* representa o número das observações relacionadas com *dataset* no final do processo.

Os campos *BUILD* e *(RE)BUILD* representam ambos o tempo de execução deste processo, em segundos. O primeiro corresponde à primeira iteração do processo, enquanto que o segundo representa todas as iterações seguintes.

Depois da análise da tabela 10.1, quando comparada com a tabela 7.1, é possível verificar que esta versão não sofre de *overflow*. Outro aspeto notório, é o facto de haver pouca diferença no tempo de execução, entre a primeira iteração e as seguintes, do processo de contextualização de dados. Contudo, todo o trabalho desenvolvido nos capítulos anteriores, reduziram o tempo de execução deste processo para cerca de 1/4 e é garantido que a fluidez dos restantes serviços é mantida. Os testes realizados no capítulo anterior, também mostram uma melhoria no tempo de resposta da API, que por sua vez é refletido na aplicação *Web* (*WAVY Client*).

10.2 Conclusão e trabalho futuro

Este trabalho teve a duração de 10 meses e permitiu trabalhar com as diversas tecnologias que foram apresentadas nesta dissertação, enriquecendo o meu conhecimento nesta área. Contudo, apesar dos desafios que surgiram no desenrolar deste trabalho e depois de realizadas as devidas validações, todos os objetivos definidos foram cumpridos com sucesso.

A integração, na aplicação *Web* do WOS, de gráficos e da secção dedicada aos dados manipulados de um *dataset*, possibilita uma melhor análise dos dados recolhidos pelos derivadores flutuantes *WAVY*, assim como facilita a deteção e remoção de dados anómalos.

Este trabalho permitiu demonstrar que as base de dados de séries temporais, quando usadas para fornecer suporte a dispositivos IOT, como o caso dos derivadores flutuantes *WAVY*, têm melhor desempenho que as base de dados relacionais. Apesar das base de dados de séries temporais só estarem otimizadas para leitura e escrita a grandes velocidades, a demora das operações de apagar pode ser ultrapassada incluindo uma coluna, que assinale a informação a ser apagada posteriormente por um processo.

Além disto, existem diversas ferramentas que possibilitam a monitorização e visualização dos dados armazenados neste tipo de base de dados, como o caso da tecnologia *Grafana*, apresentada no capítulo capítulo 8. De modo a dar continuidade a este trabalho, está a ser desenvolvido, com base em *Grafana*, um novo serviço para integrar no projeto MELOA.

Depois da integração dos ficheiros de registos e das melhorias desenvolvidas no processo de contextualização dos dados, serão convertidos os *scripts Matlab* para *JavaScript* e integrados na API do WOS. Juntamente com isto, será desenvolvida na aplicação *Web* uma secção que permita integrar os cálculos de parâmetros de Espectro de Onda.

Referências

[1] Alexandra Aguiar and Fabiano Hessel. Embedded systems' virtualization: The next challenge? *Proceedings of 2010 21st IEEE International Symposium on Rapid System Protyping*, pages 1–7, 2010.

[2] José Alberto Afonso Alexandre. As correntes marinhas. *[acedido: 15.11.2019] http://www.educadores.diaadia.pr.gov.br/arquivos/File/2010/artigos_teses/GEOGRAFIA/Artigos/correntes_marinhas.pdf*, page 17, 1996.

[3] Carlos Almeida, Artur Rocha, Lino Oliveira, and João Correia Lopes. Wavy operation software: an observation centric system for multi-sensor devices. *Instrumentation viewpoint*, (20):44–44, 2018.

[4] Tim Ambler and Nicholas Cloud. *JavaScript Frameworks for Modern Web Dev.* Apress, USA, 1st edition, 2015.

[5] A.M.Ravishankkar, D.P.Sriridanya, N.Sarmila Devi, A.Sripriya, and K.Santhoshini. Data analysis and report generation in enterprise mobility solution. *International Research Journal of Engineering and Technology (IRJET)*, 04(3):525–530, 2017.

[6] Zoran Balkić, Damir Šoštarić, and Goran Horvat. *GeoHash and UUID Identifier for Multi-Agent Systems.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[7] Eugen Betke and Julian Kunkel. *Real-time I/O-monitoring of HPC applications with SIOX, elasticsearch, Grafana and FUSE.* International Conference on High Performance Computing, 2017.

[8] Fernando Camargos and Gabriel Girard. Virtualization of linux servers. *Proceedings of the Linux Symposium*, 01 2008.

[9] Daniel F. Carlson, Tamay Özgökmen, Guillaume Novelli, Cedric Guigand, Henry Chang, Baylor Fox-Kemper, Jean Mensa, Sanchit Mehta, Erick Fredj, Helga Huntley, A. D. Kirwan, Maristella Berta, Mike Rebozo, Milan Curcic, Ed Ryan,

Björn Lund, Brian Haus, Jeroen Molemaker, Cameron Hunt, Shuyi Chen, Laura Bracken, and Jochen Horstmann. Surface ocean dispersion observations from the ship-tethered aerostat remote sensing system. *Frontiers in Marine Science*, 5:479, 2018.

[10] CARTHE. Consortium for advanced research on transport of hydrocarbon in the environment. *[acedido: 15.02.2020] http://carthe.org/.*

[11] Emiliano Casalicchio and Vanessa Perciballi. *Auto-scaling of containers: The impact of relative and absolute metrics*. 2017 IEEE 2nd International Workshops on Foundations and Applications of Self* Systems (FAS* W), 2017.

[12] Chris Churilo. Influxdb is 2.4x faster vs. mongodb for time series workloads. *[acedido: 22.05.2020] https://www.influxdata.com/blog/influxdb-is-27x-faster-vs-mongodb-for-time-series-workloads/*, 2018.

[13] IBM company. Loopback - node.js framework. *[acedido: 30.04.2020] https://loopback.io/lb3.*

[14] Rodrigo Dlugokensky. Bancos de dados para monitoramento de desempenho de grandes redes. *[acedido: 02.06.2020] https://lume.ufrgs.br/handle/10183/147636*, 2016.

[15] Scott Jones Dr. Michael Pidwirny. Surface and subsurface ocean currents: Ocean current map. *[acedido: 26.12.2019] http://www.physicalgeography.net/fundamentals/8q_1.html*, 2006.

[16] Eduardo de Freitas. Movimento das águas oceânicas. *[acedido: 15.11.2019] https://mundoeducacao.uol.com.br/geografia/movimento-das-aguas-oceanicas.htm.*

[17] Katarina Grolinger, Wilson Higashino, Abhinav Tiwari, and Miriam AM Capretz. Data management in cloud environments: Nosql and newsql data stores. *Journal of Cloud Computing: Advances, Systems and Applications*, 2013.

[18] Deimos Elecnor group. Meloa confluence. *[acedido: 02.06.2020] https://confluence.elecnor-deimos.com/.*

[19] The PostgreSQL Global Development Group. Postgresql database. *[acedido: 30.04.2020] https://www.postgresql.org/.*

[20] Danielle Hall. Five methods for tracking the ocean's motion. *[acedido: 10.01.2020] https://ocean.si.edu/conservation/gulf-oil-spill/five-methods-tracking-oceans-motion*, 2019.

[21] Dr. Jack Hall. The most important organism? *[acedido: 28.12.2019] http://www.ecology.com/2011/09/12/important-organism/*, 2011.

[22] Rui Han, Zhen Jia, Wanling Gao, Xinhui Tian, and Lei Wang. Benchmarking big data systems: State-of-the-art and future directions. *ArXiv*, abs/1506.01494, 2015.

[23] Justin Huggler. World's oldest message in a bottle washes up in germany after 108 years at sea. *The Telegraph*, Aug 2015.

[24] InfluxData. Configuring influxdb oss. *[acedido: 15.02.2020] https://docs.influxdata.com/influxdb/v1.7/administration/config/.*

[25] Monterey Bay Aquarium Research Institute. Autonomous underwater vehicles. *[acedido: 08.01.2020] https://www.mbari.org/at-sea/vehicles/autonomous-underwater-vehicles/*, 2018.

[26] A. M. Joy. *Performance comparison between Linux containers and virtual machines*. 2015 International Conference on Advances in Computer Engineering and Applications, 2015.

[27] Suresh Kumar. Docker - difference between virtual machine and container. *[acedido: 28.04.2020] https://www.idiotinside.com/2020/01/25/virutal-machine-vm-vs-docker-containter-difference/.*

[28] Mozilla and individual contributors. Componentes da web. *[acedido: 04.05.2020] https://developer.mozilla.org/pt-PT/docs/Web/Componentes_Web*, 2019.

[29] Mozilla and individual contributors. Javascript. *[acedido: 08.05.2020] https://developer.mozilla.org/pt-PT/docs/Web/JavaScript*, 2020.

[30] Syeda Noor Zehra Naqvi, Sofia Yfantidou, and Esteban Zimányi. Time series databases and influxdb. *Studienarbeit, Université Libre de Bruxelles*, 2017.

[31] Mohammad Nasar and Mohammad Abu Kausar. Suitability of influxdb database for iot applications. *International Journal of Innovative Technology and Exploring Engineering*, 8(10):1850–1857, 2019.

[32] Jakob Nielsen. *Finding Usability Problems through Heuristic Evaluation*. CHI '92. Association for Computing Machinery, New York, NY, USA, 1992.

[33] Guillaume Novelli, Cédric M. Guigand, Charles Cousin, Edward H. Ryan, Nathan J. M. Laxague, Hanjing Dai, Brian K. Haus, and Tamay M. Özgökmen. A Biodegradable Surface Drifter for Ocean Sampling on a Massive Scale. *Journal of Atmospheric and Oceanic Technology*, 34(11):2509–2532, 11 2017.

[34] Sergio Trilles Oliver, Alberto González-Pérez, and Joaquín Huerta Guijarro. An IoT proposal for monitoring vineyards called SEnviro for agriculture. *Proceedings of the 8th International Conference on the Internet of Things*, pages 1–4, 2018.

[35] Oracle. what is a relational database? *[acedido: 15.02.2020] https://www.oracle.com/database/what-is-a-relational-database/*.

[36] Inc Pacific Gyre. Carthe gps drifter. *[acedido: 10.01.2020] https://www.pacificgyre.com/carthe-drifter.aspx*.

[37] Rodolfo F. Alves Pena. Hidrosfera. Características da hidrosfera terrestre. *[acedido: 15.05.2020] https://mundoeducacao.uol.com.br/geografia/hidrosfera.htm*.

[38] MELOA PROJECT. Meloa. *[acedido: 18.04.2020] https://www.ec-meloa.eu/*.

[39] 451 Research. 451 research says application containers market will grow to reach $4.3bn by 2022. *[acedido: 01.06.2020] https://451research.com/451-research-says-application-containers-market-will-grow-to-reach-4-3bn-by-2022*.

[40] Community Research and Development Information Service. Meloa project information. *[acedido: 15.05.2020] https://cordis.europa.eu/project/id/776825*.

[41] Behnam Shafiei Sabet and Gholam Abbas Barani. Design of small gps drifters for current measurements in the coastal zone. *Ocean & Coastal Management*, 54(2):158 – 163, 2011.

[42] Prateek Sharma, Lucas Chaufournier, Prashant Shenoy, and YC Tay. Containers and virtual machines at scale: A comparative study. *Proceedings of the 17th International Middleware Conference*, pages 1–13, 2016.

[43] solid IT. Knowledge base of relational and nosql database management systems. *[acedido: 22.05.2020] https://db-engines.com/en/*.

[44] S. Tilkov and S. Vinoski. Node.js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, 14(6):80–83, 2010.


U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO