Departamento de Engenharia Informática

Relatório de Estágio

Mestrado em Engenharia Informática – Computação Móvel

Web Vessel Tracker: Aplicação Web para Monitorização de Embarcações

José Pedro Roriz Ferreira

Leiria, setembro de 2015

Departamento de Engenharia Informática

Relatório de Estágio

Mestrado em Engenharia Informática – Computação Móvel

Web Vessel Tracker: Aplicação Web para Monitorização de Embarcações

José Pedro Roriz Ferreira

Estágio de Mestrado realizado sob a orientação do Doutor Carlos Fernando Almeida Grilo, Professor da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, setembro de 2015

À Minha Família

Esta página foi intencionalmente deixada em branco

Agradecimentos

Esta página, serve para agradecer às pessoas que deram um contributo muito importante para o fim desta etapa da minha vida académica.

Em primeiro lugar, queria deixar mil obrigados ao Professor Carlos Grilo, pelas horas disponibilizadas, pelo apoio e apoio moral que me deu durante este estágio e numa fase complicada que atravessei, pela preocupação mostrada, pela paciência que teve e por muitas mais coisas, fica mais um obrigado.

Queria agradecer também ao Pedro Borges, da XSealence por ter proposto um projeto muito interessante e pelo apoio e tempo prestado durante o estágio. Quero também deixar um obrigado às pessoas da XSeaelence por me terem recebido de braços abertos e por criar um ambiente amigável.

Um obrigado que não podia deixar de ser dado é aos meus pais, por me terem sempre apoiado e me terem dado a possibilidade de chegar até onde cheguei hoje, obrigado mãe e pai!

Um agradecimento final aos meus colegas de trabalho diário, obrigado João, Joel, Rúben, Jóni e André, assim como, aos meus amigos de longa duração que me ajudaram a ultrapassar este desafio, nomeadamente o Miguel Barros e o Luís Manso e aos restantes por me proporcionarem sempre momentos divertidos e bem passados.

Esta página foi intencionalmente deixada em branco

Resumo

Este documento visa explicar o trabalho que foi realizado para o projeto Web Vessel Tracker. Este projeto, proposto pela empresa XSealence S.A, foi realizado no âmbito da unidade curricular de Estágio do Mestrado em Engenharia Informática – Computação Móvel (MEI-CM) da Escola Superior de Tecnologia e Gestão (ESTG) do Instituto Politécnico de Leiria (IPL).

O Web Vessel Tracker é um projeto que se enquadra na monitorização de embarcações com a qual se tenta resolver diversos problemas, como a pesca ilegal ou a sobrepesca. Estes são problemas que têm ganho bastante atenção, já que põem em causa a quantidade e variedade de peixe disponível numa determinada zona, o que pode levar a graves repercussões tanto a nível ambiental como económico.

O Web Vessel Tracker, permite visualizar num mapa vários detalhes de embarcações que utilizam o dispositivo AIS e/ou MONICAP. A aplicação permite ainda obter informação relativa às rotas realizadas por uma embarcação, assim como, as capturas realizadas por esta.

O Web Vessel Tracker serve para complementar o sistema proprietário de monitorização de embarcações já existente desenvolvido pela XSealence, o Centro de Controlo Integrado.

O trabalho desenvolvido foca-se em utilizar tecnologias do lado do cliente para fazer a representação geográfica de informação relativa às embarcações, através de tecnologias de *web mapping*. Ao mesmo tempo, para obter essa informação, são utilizadas tecnologias do lado do servidor que depois a envia para o cliente.

Palavras-chave: Sobrepesca, Monitorização de Embarcações, *Web Mapping*, Representação Geográfica.

Esta página foi intencionalmente deixada em branco

Abstract

This document describes work that was done for the project Web Vessel Tracker, during the internship at XSealence SA. The internship was done within the Internship course, that is part of the master degree in Computer Science – Mobile Computing from School of Technology and Management from Leiria Polytechnic Institute.

Web Vessel Tracker is a project that falls into the vessel monitorization area, intended to help solving several problems such as, overfishing and illegal fishing. These are problems that have been gaining a lot of attention, as they threaten the variety and quantity of available fish in a certain zone, which can lead to severe environmental and economic problems.

This application, allows the user to see in a map information regarding any ship as long as they are equipped with AIS or MONICAP devices. The application also grants the user additional information about the ships, such as, the routes performed and captured fish that a vessel has done (as long as there is a MONICAP device onboard).

This project is an adition to the existent proprietary system developed by XSealence, the Integrated Control Center.

The work done was focused on using client side technologies to display geographic information, relative to the ships being displayed. At the same time, server side technologies are used in order to obtain that very information.

Key-Words: Overfishing, Vessel Monitorization, *Web Mapping*, Geographic Representation.

Esta página foi intencionalmente deixada em branco

Índice de Figuras

Lista de Siglas

Ao longo deste documento são apresentadas diversas siglas, cujo o significado é descrito na tabela abaixo. No texto principal, as siglas são apresentadas, com o seu significado aquando da sua primeira utilização.

Sigla	**Significado**
AIS	Automatic Identification System
Ajax	Asynchronous JavaScript and XML
API	Application Programming Interface
AToN	AIS Aids to Navigation
C#	C Sharp
CCI	Centro Control Integrado
CSS/CSS3	Cascading Style Sheets
DOM	Document Object Model
ESRI	Environmental Systems Research Institute
FAO	Food and Agriculture Organization of the United Nations
FMC	Fisheries Monitoring Center
GIS	Geographic Information Systems
GML	Geography Markup Language
GPS	Global Positioning System
GUI	Graphical User Interface

HTML/HTML5	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IE	Internet Explorer
IDE	Integrated Development Environment
IIS	Internet Information Services
JSON	JavaScript Object Notation
MCS	Monitoring control and Surveillance
MONICAP	MONItorização e Controlo das Ativdades de Pesca
MVC	Model View Controller
MVVM	Model View ViewModel
NAF	North Atlantic Format
ODbL	Open Data Commons Open Database License
OGC	Open Geospatial Consortium
ORM	Object-Relational Mapping
OSM	OpenStreetMap
S-AIS	Satelite-AIS
SBD	Short Burst Data
SOS	Save Our Souls
UI	User Interface

VMS	Vessel Monitoring System
VTS	Vessel Traffic Service
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
XHR	XmlHttpRequest
XHTML	Extensible HyperText Markup Language
XML	EXtensible Markup Language
ZEE	Zona Económica Exclusiva

Esta página foi intencionalmente deixada em branco

Índice

Esta página foi intencionalmente deixada em branco

Esta página foi intencionalmente deixada em branco

Introdução

A preocupação com a existência de uma pesca sustentável é algo que tem merecido uma atenção crescente ao longo do anos por parte das autoridades responsáveis, sejam elas de caráter nacional ou internacional. Esta preocupação é motivada pelo aparecimento de tecnologias, sistemas e práticas que têm vindo a permitir o aumento do número de capturas. Ou seja, a atividade piscatória tem vindo a tornar-se cada vez mais agressiva levando a que a própria atividade fique ameaçada. De forma a combater estas práticas, foram criadas diversas leis e regulações sobres as pescas e aumentou também a necessidade de monitorizar as embarcações piscatórias. Ao monitorizar uma embarcação, é possível detetar quando é que existe uma captura ilegal, seja esta uma captura de espécies protegidas seja uma captura que exceda o peso máximo permitido para uma determinada espécie.

A aplicação web desenvolvida neste estágio, o Web Vessel Tracker, é mais um mecanismo que poder ser utilizado para que a atividade piscatória seja sustentável, já que permite ao utilizador visualizar de uma forma rápida e simples os detalhes das embarcações piscatórias. O Web Vessel Tracker consegue ainda apresentar ao utilizador, os detalhes das rotas que foram feitas pelas embarcações, a sua posição atual, a carga que contêm a bordo, entre outras. O Web Vessel Tracker junta-se, assim, a outros sistemas já desenvolvidos pela XSealence com os quais se procura garantir um futuro melhor para todos.

1.1 Entidade de Acolhimento

A entidade de acolhimento deste estágio foi a Xsealence S.A. - Tecnologias do Mar S.A, empresa fundada em 2013, sendo a sua principal área de negócio as tecnologias marítimas, nomeadamente de controlo, monitorização e comando. A XSealence pertence ao grupo Trailtec à qual pertence também a Tecmic.

No que respeita a parceiros, tal como ilustra a Figura 1, a XSealence conta com entidades como a Aquafish, JRC, IRPA, COMSAT, IRIDIUM, INMARSAT, NAUTEMA entre outras.

A nível de clientes, conta com a MPA, IRPA, DGRM, entre outras. A área de influência extende-se por Portugal, Angola, Turquia, Irlanda, Espanha e França.

Figura 1 - Parceiros de negócio e tecnológicos da XSealence

1.2 Motivação e objetivos

A pesca é uma atividade que sempre acompanhou a humanidade, desde a altura em que era feita com lanças, até aos dias de hoje onde são utilizadas diversas tecnologias para encontrar cardumes e embarcações capazes de carregar toneladas de peixe. Esta evolução permitiu trazer uma maior quantidade de peixe para o mercado, assim como, maior variedade. Apesar desta evolução ter trazido coisas boas para a humanidade, trouxe também o problema da sobrepesca e da capturas ilegais.

Estes problemas têm ganho cada vez mais atenção nos últimos anos, uma vez que põem em causa a pesca sustentável. A sobrepesca é um fenómeno que não é recente mas que se tem vindo a agravar nas últimas décadas [1]. Este é um fenómeno que pode ser descrito e caracterizado de várias formas, mas a ideia é quase sempre a mesma: fazer capturas excessivas e num pouco espaço de tempo num determinado ecossistema. Deste modo, o ecossistema vai-se degradando aos poucos, uma vez que a quantidade e a variedade de espécies presentes nesse ecossistema vão sendo cada vez menores [2] [3].

Para combater este problema foram criadas várias medidas como, por exemplo, a regulação das capturas que são feitas. Esta regulação pode ir desde, a limitação da quantidade anual que

pode ser capturada [4], à criação de épocas em que uma espécie pode ser capturada.

As capturas ilegais são também um problema para a existência de uma pesca sustentável. Uma captura ilegal pode ser, por exemplo, a captura de uma espécie cujas quotas já atingiram o limite ou a pesca em zonas proibidas. Tal como aconteceu com a sobrepesca, foram também criadas medidas para prevenir e punir a pesca ilegal. Para mais informação sobre regulações da pesca profissional e enquadramentos legais da pesca em Portugal consultar, respetivamente, as referências [5] e [6].

De modo a facilitar uma pesca sustentável e prevenir estes problemas e outros, foram criadas diversas leis, regulações e sistemas. Um dos sistema criados foi o MONICAP (MONItorização e Controlo das Ativdades de Pesca) [7] que é um VMS (Vessel Monotoring System). Os conceitos de MONICAP e VMS são discutidos em mais detalhe no Capítulo 2 mas, de forma resumida, são sistemas instalados numa embarcação que permitem monitorizar a sua posição e capturas realizadas. Essa informação é depois transmitida para um centro de controlo onde a atividade piscatória é monitorizada pelas autoridades competentes, nomeadamente, as autoridades de pesca e as autoridades marítimas [8]. A utilização de um sistema como o MONICAP levou naturalmente ao desenvolvimento de aplicações que permitem monitorizar as capturas das embarcações. Estas aplicações designam-se por *centros de controlo*, onde é possível analisar a informação recolhida pelo MONICAP e as rotas e capturas que uma determinada embarcação fez num determinado período.

Uma dessas aplicações, o Centro de Controlo Integrado (CCI), desenvolvido pela XSealence, é uma aplicação *desktop* onde é possível ao utilizador visualizar a informação que foi enviada pelo MONICAP. Nesta aplicação o utilizador pode criar rotas, áreas de pesquisa, ver informação relativa às embarcações, como é o caso da carga atual de uma embarcação que tenha o dispositivo MONICAP, entre outras. De forma a dar ao utilizador uma alternativa de acesso a esta informação, foi criada posteriormente uma aplicação *web* baseada no CCI, o webCC. Esta aplicação, para além de não restringir o utilizador a um único espaço físico, permite-lhe visualizar e realizar tarefas semelhantes à versão *desktop*, embora as funcionalidades disponibilizadas sejam em menor número, principalmente no que respeita às funcionalidades de pesquisa e visualização de rotas. O webCC, foi desenvolvido com recurso a tecnologias entretanto descontinuadas, como seja o caso do Silverlight da Micorsoft. Esta foi uma das motivações para o desenvolvimento do Web Vessel Tracker. Para além destas versões, foi criada também uma aplicação para Android, o Seawolf Patrol, que utiliza o CCI

de modo a conseguir pesquisar e apresentar informação sobre embarcações. Esta aplicação não fez parte do plano de trabalhos deste estágio.

1.3 Estrutura do documento

Terminamos este capítulo com uma breve descrição dos restantes capítulos deste relatório:

- Capítulo 2: Este capítulo aborda os vários sistemas que existem para a monitorização de navios e das pescas, faz uma descrição geral do projeto webCC e termina com uma caracterização de aplicações web semelhantes ao webCC.

- Capítulo 3: Neste capítulo são introduzidos os sistemas GIS (*Geographic information system*) e depois é feita uma descrição dos geoservidores e plataformas de desenvolvimento para *web mapping* que foram analisadas para o desenvolvimento do Web Vessel Tracker.

- Capítulo 4: Este capítulo explica em detalhe a metodologia de desenvolvimento utilizada, o planeamento e gestão do projeto e uma análise aos requisitos que foram pedidos.

- Capítulo 5: Este capítulo faz uma descrição das tecnologias e ferramentas que foram utilizadas para o desenvolvimento da aplicação, descreve em pormenor a forma como a implementação foi realizada, assim como a justificação das decisões que foram tomadas durante o desenvolvimento.

- Capítulo 6: Neste capítulo são discutidos os resultados que foram obtidos nos testes que foram realizados.

- Capítulo 7: Neste capítulo é realizada uma refleção sobre o trabalho desenvolvido ao longo deste estágio e descrito o trabalho futuro.

Estado da arte

Este capítulo descreve algumas das tecnologias utilizadas hoje em dia para a localização e monitorização de navios. Uma vez descritas as tecnologias, é realizada uma caracterização do webCC, que é a versão anterior do Web Vessel Tracker. No final, é realizada uma análise de aplicações semelhantes ao webCC e WebVessel Tracker.

2.1 Localização de navios

Esta secção aborda os diferentes conceitos/tecnologias que são utilizados na localização de navios. Primeiramente, será introduzido o conceito de *Monitoring Control and Surveillance*. Serão, depois, descritas as tecnologias *Vessel Monitoring System* e *Automatic Identification System*.

2.1.1 Monitoring Control and Surveillance

Monitoring Control and Surveillance (MCS) é um conceito que foi criado pela *Food and Agriculture Organization of the United Nations* (FAO) [9] com o objetivo de tornar sustentável a atividade da pesca.

Como o próprio nome indica, este conceito baseia-se em três outros - monitorização, controlo e vigilância - que na conferência MCS *Conference of experts* de 1981 em Roma [10] foram descritos da seguinte forma:

- A monitorização é um requisito contínuo para a medição dos esforços de pesca e dos rendimentos dos recursos. Este conceito envolve também a recolha de dados relativos à composição das espécies [10], o esforço de pesca, a área de ocupação, as capturas, as capturas acidentais (algo que acontece quando espécies diferentes das pretendidas são capturadas);

- O controlo está relacionado com as condições reguladoras de como é que a exploração

de um determinado recurso pode ser realizada. Estas condições consistem tipicamente num corpo de legislação, regulamentos e acordos internacionais. Tipicamente, estas condições levam ao estabelecimento de áreas em zonas onde é proibido qualquer tipo de pesca; restrições nos equipamentos que são usados para pescar, assim como a entrada de certos tipos de embarcações em zonas de pesca e o controlo da quantidade e tipo de peixe que se pode pescar num determinado período;

- A vigilância, que é o grau e os tipos de observações necessários para manter o cumprimento das atividades de pesca de acordo com os controlos reguladores impostos relativamente a este tipo de atividades.

A imagem seguinte ilustra como é que o conceito MCS ajuda na gestão das pescas.

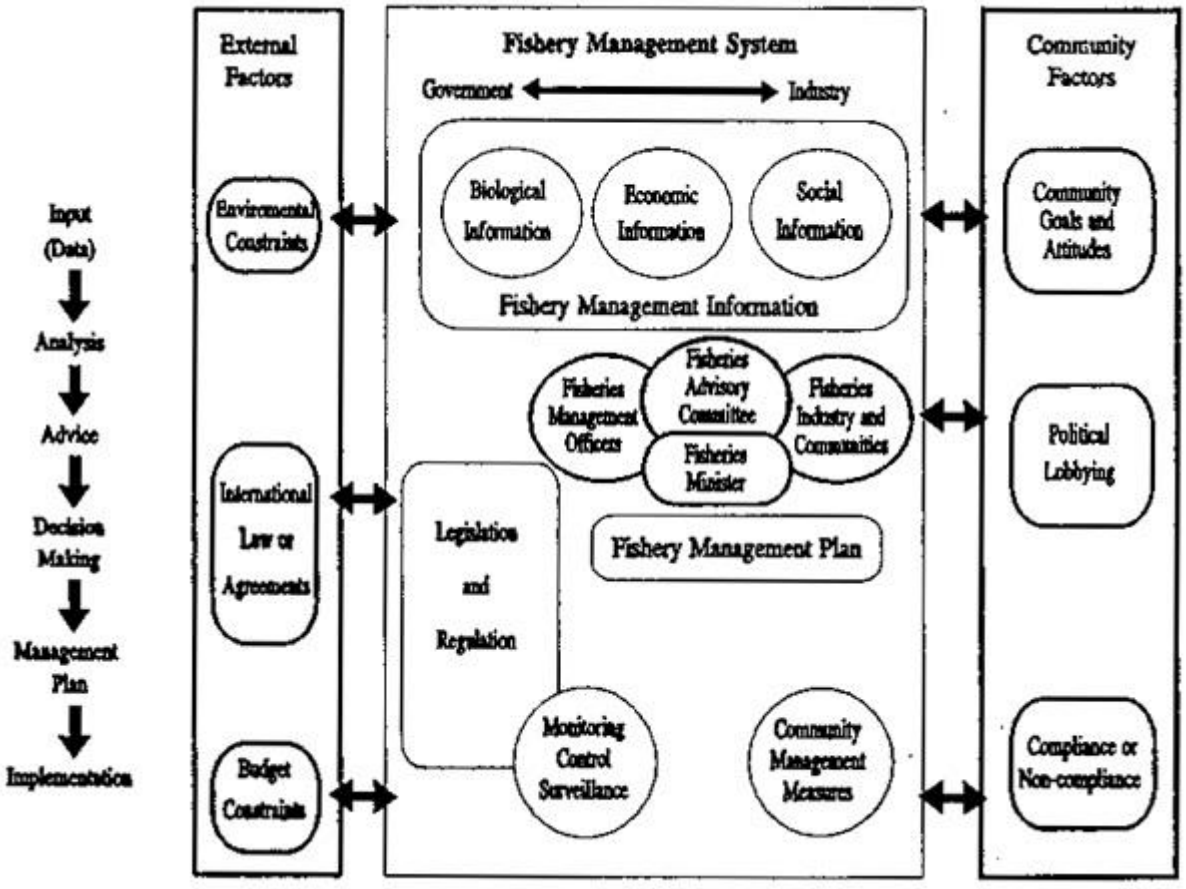


Figura 2 - Conceito MCS

Neste esquema, retirado de [10], é assumido que existem três componentes envolvidos na gestão das pescas. Esses componentes são:

- Uma coleção de dados sobre os aspetos biológicos e económicos das pescas, assim

como informação básica relativa aos pescadores, embarcações e equipamentos. Esta informação incluí ainda informação relativa aos padrões e tendências das pescas. Toda esta informação é utilizada no planeamento e gestão da atividade piscatória;

- Uma componente para a tomada de decisões, que inclui todo o processo de negociação de todos as partes que influenciam as decisões respeitantes à atividade piscatória. O objetivo final é obter planos para a gestão das pescas assim como o apoio político necessário para a implementação desses planos;

- Uma componente que diz respeito à implementação dos planos para a gestão das pescas. Esta componente envolve a monitorização, controlo e vigilância da atividade piscatória, assim como dos pescadores, sendo isto um requisito fundamental para o sucesso da implementação dos planos de gestão. A falta de atenção ou empenho na implementação destas atividades MCS resulta frequentemente em sobrepesca, colapso de recursos e consequentemente numa perda económica para as gerações futuras.

De modo a garantir uma implementação bem-sucedida, são utilizadas diferentes tecnologias como, por exemplo, radares e os *Vessel Monitoring Systems* (ver secção seguinte), entre outros. Estas tecnologias tornam possível descobrir as embarcações que pescam mais do que devem e/ou o que não devem, tentando garantir, assim, a existência de uma pesca sustentável na maior parte do planeta.

2.1.2 Vessel Monitoring Systems

Os VMS foram desenvolvidos para dar resposta ao problema da pesca excessiva e ao impacto que esta ação tem nas reservas de peixe a nível mundial. Estes sistemas têm como objetivo tornar sustentável a atividade da pesca e possibilitar que as organizações reguladoras do ambiente e das pescas monitorizem aspetos como a posição, a velocidade, o rumo das várias embarcações, assim como o peixe que capturam. Os VMS são constituídos por 1) equipamentos físicos que estão instalados nas embarcações e que fazem a recolha da informação sobre a sua atividade (ver Figura 3) [11]; 2) um sistema de comunicação [12] que é responsável por enviar a informação recolhida para estações de monitorização (FMCs – *Fisheries Monitoring Centers*); 3) os FMCs, que funcionam como um centro de controlo (estes serão abordados com maior detalhe mais à frente nesta secção) onde existe uma aplicação que permite analisar a informação recebida das embarcações como, por exemplo, o tipo de peixe capturado, a quantidade, as rotas que uma embarcação realizou, entre outras.

De seguida, é apresentada uma imagem retirada de [13] que ilustra alguns dos componentes que os sistemas VMS usam.

Figura 3 - Exemplos de equipamentos físicos instalados na embarcação

Os equipamentos destes sistemas são formados por uma combinação de um transcetor com uma antena. São estes equipamentos que transmitem a informação tipicamente envolvida num sistema VMS como, por exemplo, o identificador único do sistema VMS, a data e a hora de cada posição com a inclusão da longitude e latitude.

A recolha da informação que diz respeito à posição da embarcação é feita de duas formas. Uma delas consiste em recorrer a uma unidade GPS integrada; A outra consiste em utilizar um sistema de satélites que medem o efeito de Doppler do sinal que receberam do dispositivo localizado na embarcação para descobrir a sua posição.

O rumo e velocidade também podem ser calculados de duas formas diferentes: em simultâneo com a posição ou nos FMCs com recurso à aplicação de monitorização tendo em conta o intervalo de tempo entre cada posição da embarcação.

O funcionamento típico de um sistema VMS pode ser descrito da seguinte forma: os equipamentos presentes na embarcação recolhem a informação que foi já mencionada e enviam-na para um sistema de comunicação como, por exemplo, ARGOS, Inmarsat, Orbcomn, entre outros. Os sistemas de comunicação são responsáveis por enviar a informação recebida das embarcações para os FMCs. Estes podem receber a informação dos sistemas de comunicação com recurso à Internet, ao protocolo de comunicação X.25 (este tem perdido bastante popularidade) ou então aos dois. Depois de recebida, a informação é validada e armazenada, tipicamente, em bases de dados relacionais, para ficarem disponíveis para análise posterior. É também nos FMCs que é realizada a análise dos dados com recurso a uma aplicação que serve de intermediária entre o analista e os equipamentos presentes nas

embarcações. Uma vez que a informação que os FMCs recebem é bastante sensível, estes tomam várias precauções de modo a garantir o não-repúdio e a integridade dessa informação como, por exemplo: emitir um alerta caso haja alguma manipulação dos equipamentos a bordo das embarcações (ex: desligar ou bloquear a antena), ou mesmo interrupções de energia que podem ser ou não acidentais; garantir que cada equipamento pertencente ao sistema VMS tem um identificador único podendo-se, deste modo, prevenir a criação de equipamentos VMS falsificados; a tomada de medidas *anti-spoofing*, isto é, prevenir a inserção de informação falsa. De seguida, é feita uma ilustração adaptada de [11], onde o funcionamento típico de um sistema VMS está esquematizado.

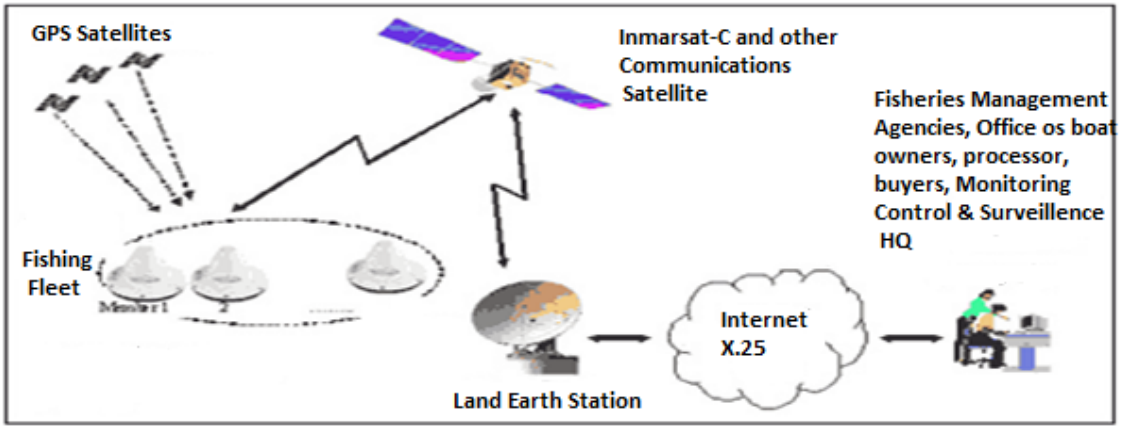


Figura 4 - Os componentes de um sistema VMS

De seguida, são apresentadas algumas aplicações possíveis que derivam da utilização de VMS:

- Como foi já referido, estes sistemas recolhem informação, permitindo a entidades terceiras monitorizar as embarcações e as suas atividades. Com recurso à informação recolhida, é possível saber, por exemplo, se uma embarcação se encontra na ZEE (Zona Económica Exclusiva) de uma nação. Este é um aspeto importante, tanto a nível legal, como para a preservação de reservas de determinadas espécies de peixes;

- Uma vez que os sistemas VMS permitem monitorizar embarcações e os seus movimentos, é possível utilizar essa informação para fazer uma análise do comportamento que as embarcações de pesca têm [14];

- Alguns dos dados recolhidos por sistemas VMS como, por exemplo, dados recolhidos para a vigilância das pescas, podem ser utilizados por equipas de busca e salvamento

para dar auxílio no resgate de embarcações em perigo;

- Os VMS podem ser usados em conjunto com outros equipamentos, como é o caso do sistema AIS (que será introduzido na próxima secção). Desta forma é possível complementar a vigilância das embarcações permitindo, por exemplo, que as aplicações que lidam com a segurança da navegação, derrames tóxicos/nocivos, ameaças ambientais, tenham um grau de vigilância maior;

- A informação obtida com os sistemas VMS como, por exemplo, a posição das embarcações, permite fazer uma análise das rotas que estas fazem em busca do peixe. Com esta informação é possível fazer uma estimativa da quantidade de peixe pescado numa determinada área e ter uma ideia de quando é que se deve restringir o acesso a essa área, evitando a sobrepesca ou o desaparecimento de uma determinada espécie.

2.1.3 Automatic Identification Systems

Os *Automatic Identification Systems* (AIS) são sistemas utilizados pelas embarcações e pelos sistemas costeiros de monitorização de tráfico marítimo como os VTS (*Vessel Traffic Services*), que permitem a identificação e localização automática de embarcações. A troca de informação pode se realizada entre navios ou entre navios e sistemas de monitorização de forma direta ou através de satélite. Neste último caso, utiliza-se a designação de Satelite-AIS (S-AIS). Os VTS são sistemas de monitorização que são usados pelos portos ou pelas autoridades portuárias, na monitorização do tráfego marítimo, algo semelhante com que acontece nos sistemas de tráfego aéreo.

A informação que é disponibilizada pelo AIS pode ser utilizada como um complemento à deteção realizada através dos radares marítimos, para a prevenção de colisões. Esta tecnologia permite identificar individualmente cada embarcação, a sua posição e movimentos. Os navegadores conseguem utilizar essa informação para tomar decisões e evitar colisões com outras embarcações ou formações rochosas.

Chamamos a atenção para o facto de que os sistemas AIS e os VMS partilham algumas funcionalidades, como é o caso da transmissão da posição das embarcações. O facto de terem algumas características funcionais em comum não impede a utilização dos dois equipamentos, uma vez que isso permite uma maior precisão na monitorização das embarcações, além de permitir alguma redundância caso um dos equipamentos falhe.

Tal como ilustra a Figura 5 (retirada de [15]), tipicamente, um sistema AIS consiste num dispositivo localizado na embarcação que transmite informação relativa a esta como, por exemplo, o número MMSI (*Maritime Mobile Service Identity*), que identifica uma embarcação de forma única, as suas dimensões, a posição, o nome, local de partida, destino, tempo estimado de chegada [16], entre outros. Esta informação é recebida e analisada por estações costeiras e por outras embarcações que estejam perto.

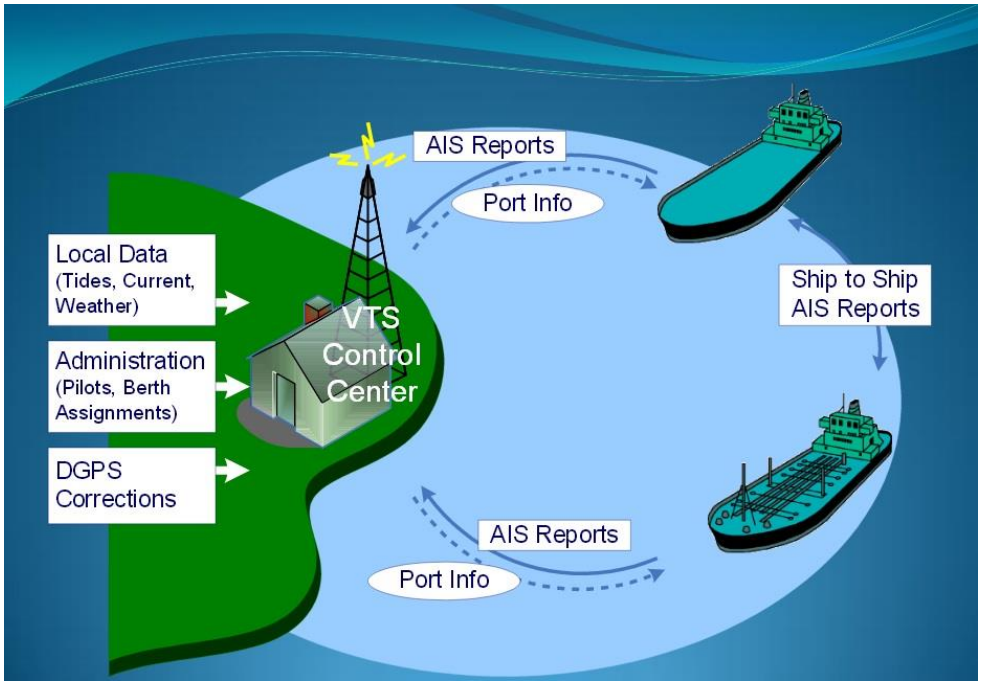


Figura 5 - O ciclo de vida de um sistema AIS

Além de permitir identificar as embarcações, o que consiste num fim por si mesmo, os sistemas AIS podem ser utilizados com outros fins, como sejam:

- Monitorização de embarcações de pesca, algo que é bastante usado pelas autoridades para a monitorização das embarcações de pesca do um determinado país. Tal é possível uma vez que o AIS permite fazer a monitorização ao longo da zona costeira de uma forma barata e fiável;

- Segurança marítima, resultante de uma fusão dos dados obtidos pelos radares e pelos AIS. Esses dados são depois utilizados pelas autoridades de modo a poder identificar e diferenciar embarcações com maior facilidade, assim como criar os padrões de pesca de uma determinada embarcação. Caso a embarcação quebre ou saia desse padrão as autoridades recebem um alerta ficando, deste modo, as possíveis ameaças já sinalizadas. Dentro desta área de segurança marítima é possível também receber

alertas caso uma embarcação entre em águas pertencentes à zona económica exclusiva de uma determinada nação;

- Em situações de busca e salvamento, que são situações em que não há espaço para errar. De modo a conseguir auxiliar as equipas de busca e salvamento, o AIS consegue fornecer os dados necessários assim como melhorar a perceção dos recursos disponíveis;

- Na investigação de acidentes, a informação que o AIS recebe dos VTS é bastante importante uma vez que este fornece dados históricos precisos, a identidade, posições GPS, velocidade, a frequência das viragens, a direção que se seguia, informações estas que não são possíveis de obter por um radar.

De modo a melhorar a forma como o AIS ajuda na navegação, foi criado o *AIS Aids to Navigation* (AtoN). Este sistema permite que os nomes e posições de outros objetos que não sejam embarcações possam ser transmitidos, como é o caso de faróis, boias, etc.

Apesar das várias qualidades que os sistemas AIS têm, existem falhas [17] que permitem, por exemplo, que entidades terceiras possam criar navios falsos, acionar falsos alarmas de SOS ou de colisão, apoderar-se das comunicações de um navio ou mesmo desligar o sistema. Estas vulnerabilidades foram demonstradas por uma equipa da Trend Micro numa conferência sobre segurança de modo a alertar para os eventuais perigos que poderão aparecer.

2.2 O Sistema de Controlo Integrado

O Centro de Controlo Integrado (CCI) desenvolvido pela XSealence é uma aplicação que disponibiliza ao utilizador grande parte das funcionalidades de um sistema VMS. Esta aplicação funciona de forma distribuída e o armazenamento dos dados é realizado numa base de dados centralizada, disponível aos vários componentes do sistema. O CCI permite realizar as seguintes tarefas:

- Monitorização das comunicações das embarcações;

- Controlo das atividades de pesca em áreas relevantes, permitindo o controlo das embarcações que entram/saem nas/das áreas marítimas de um país;

- Assegurar a correta troca de dados com outros FMCs;

- Integração de diversos fornecedores de dados;
- Notificação de alertas e eventos, permitindo que o utilizador seja notificado caso as anomalias definidas no sistema sejam detetadas;
- Administração e configuração centralizadas;
- Auditoria e produção centralizada de relatórios, que permite fazer um registo sobre as operações que foram realizadas pelos utilizadores.

A informação geoespacial sobre as embarcações como, por exemplo, localização (latitude e longitude), rumo, velocidade, se ocorreram falhas de GPS, entre outros, é recolhida pelo sistema MONICAP que a envia para a base de dados de uma estação costeira através de uma rede de satélites que atualmente é a Inmarsat. A informação é, depois, enviada a outra base de dados localizada na empresa que, por conveniência, denominaremos neste documento por base de dados central. O CCI utiliza a informação armazenada nesta base de dados, bem como noutras (por exemplo, com informação acerca de espécies animais) para apresentar, por exemplo, a rota de uma embarcação de uma forma amigável/percetível ao utilizador. Além da informação recolhida pelo MONICAP, a base de dados central armazena outros tipos de informação, nomeadamente, as credenciais dos utilizadores.

A figura seguinte ilustra os vários módulos existentes no CCI.

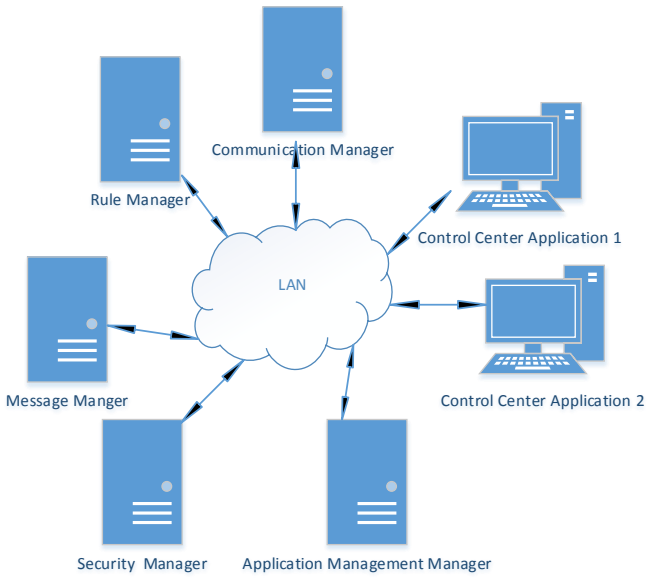


Figura 6 - Módulos existentes no CCI

Os módulos que o CCI apresenta são:

- Um módulo para as regras, que permite monitorizar e controlar de forma autónoma as embarcações de pesca e outros recursos. Este módulo reage a determinados eventos (e.g, quando uma embarcação entra numa zona económica exclusiva), gerando, notificações, alertas ou ações consoante o estado ou informação recebida;

- Um módulo para as comunicações, que assegura que todas as comunicações com entidades externas e entre os módulos do sistema funcionem. Este módulo suporta vários tipos de protocolos e assegura que as mensagens sejam entregues ao destino;

- Um módulo para as mensagens, que permite converter as mensagens que são recebidas pelo sistema para um formato interno e realizar processo inverso para as mensagens a enviar, libertando os restantes módulos desta tarefa. Os protocolos do MONICAP e os protocolos baseados em NAF (*North Atlantic Format*) [18] são dois exemplos de protocolos/formatos que este módulo suporta e traduz para um formato interno. Caso seja recebida uma mensagem num formato não reconhecido, este módulo lança um alerta;

- Um módulo de segurança, responsável por autenticar os utilizadores do sistema e encriptar as mensagens;

- Um módulo para a gestão de aplicações, que tem como objetivo monitorizar o estado das aplicações que estão a correr no servidor do sistema. Este módulo permite que o utilizador (com os devidos privilégios) seja notificado caso ocorra uma anomalia.

2.3 O WebCC até ao início do estágio

O webCC é uma aplicação *web* que permite que um utilizador possa monitorizar os movimentos de uma determinada embarcação de forma bastante detalhada. Na prática, o webCC é uma versão *web* simplificada do CCI.

Como é possível ver na figura seguinte, a forma como os dados são recolhidos nas embarcações é comum à que descrevemos na secção anterior para o CCI. Ou seja, o webCC também utiliza a informação armazenada na base de dados central já mencionada. Além desta informação, o webCC utiliza também os mapas armazenados num geoservidor, que atualmente se designa por GeoServer [19]. Estes mapas permitem, por exemplo, que a rota de uma embarcação possa ser apresentada de uma forma amigável/percetível ao utilizador. Os mapas armazenados no GeoServer são mapas do projeto OpenStreetMap [20]. O CCI utiliza o

TatukGIS [21] para fazer o mapeamento da informação geoespacial, isto é, representar num mapa as rotas das embarcações. O TatukGIS suporta uma grande variedade de tipos de dados para fazer a representação da informação geoespacial como por exemplo: SHP, TAB, DXF, GML, KML, JSON, GeoTIFF, JPEG, PNG, BMP, PixelStore, PostGIS, Microsoft Spatial, Oracle Spatial, ESRI PGDB, MapInfo SpatialWare, Geomedia, SQLite Spatial, entre outros.

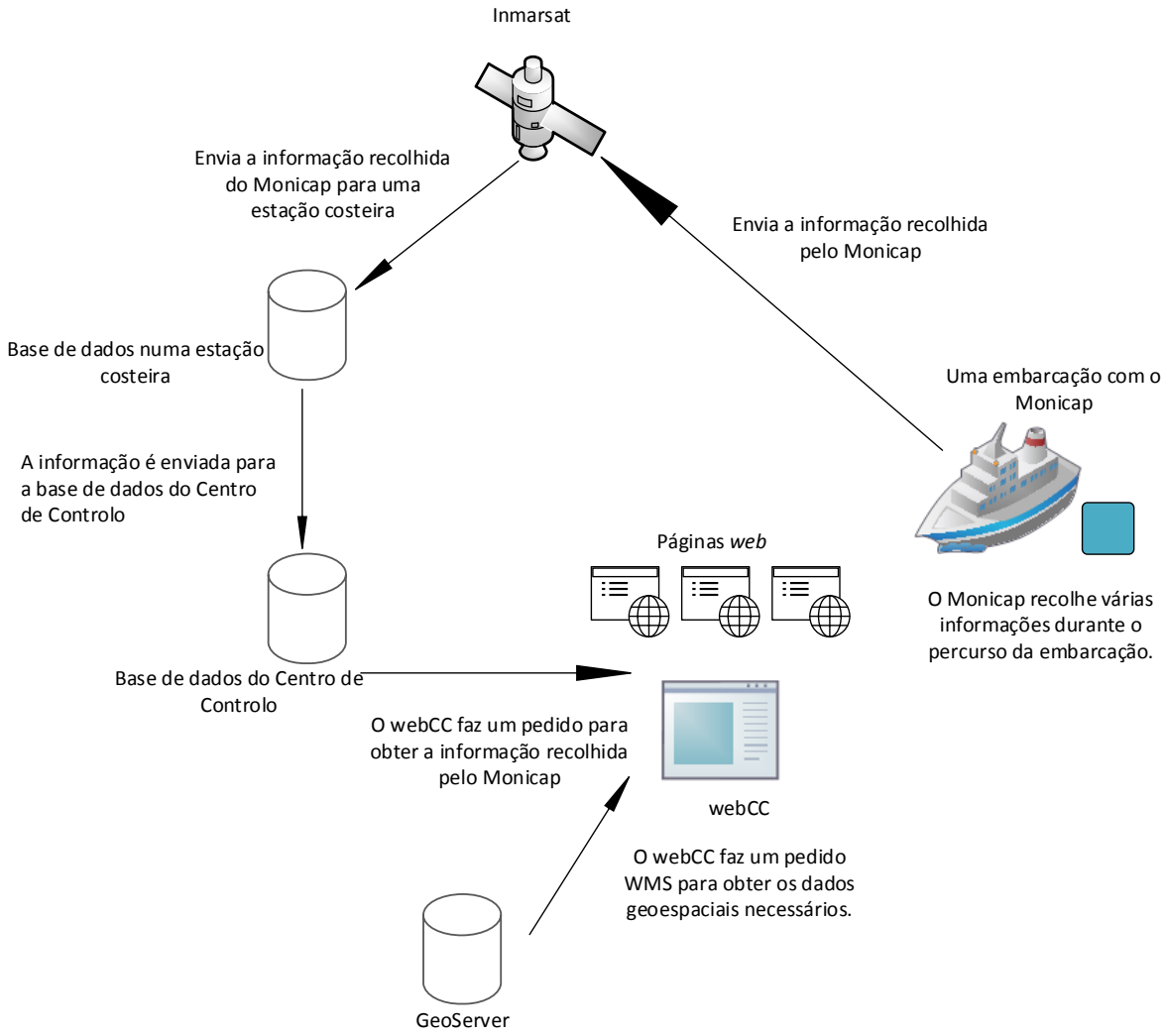


Figura 7 - Ciclo de vida dos dados recolhidos até chegarem ao webCC

A figura seguinte ilustra de uma forma mais detalhada o funcionamento do webCC, isto é os passos que ocorrem até a informação chegar ao utilizador.

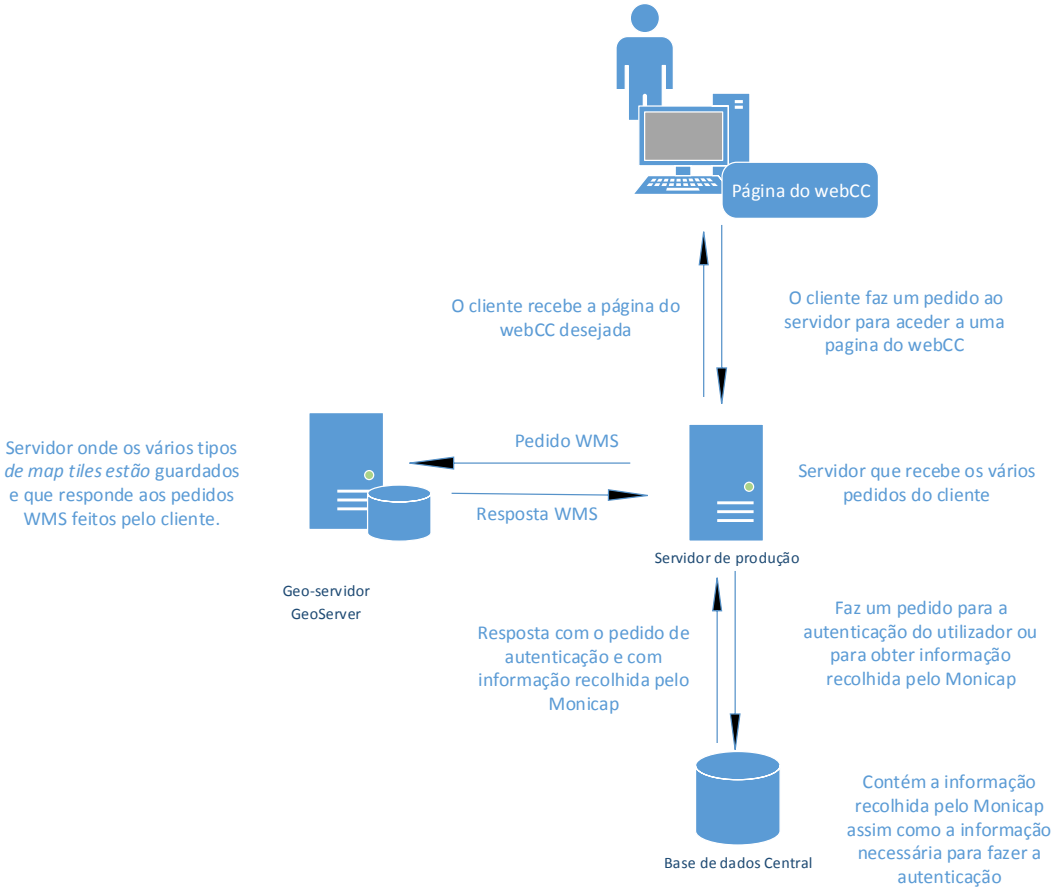


Figura 8 - Passos que ocorrem até a informação chegar ao utilizador no webCC

Uma vez na página principal do webCC, e após o processo de autenticação, o utilizador tem à sua disposição uma variedade de funcionalidades como, por exemplo, visualizar mapas, visualizar rotas de embarcações que utilizem o MONICAP (ou outro sistema de monitorização, como o AIS, radar ou deteção por estações costeiras), pesquisar embarcações em áreas marítimas, entre outras. Uma sessão de utilização típica começa com um pedido do utilizador, por exemplo, para visualizar rotas de embarcações. Estes pedidos são enviados ao servidor de produção que, por sua vez, faz um pedido ao geoservidor para que este lhe envie o mapa solicitado pelo utilizador. Caso o pedido envolva informação sobre embarcações, a informação respetiva é pedida à base de dados central, tendo em conta o intervalo de tempo especificado pelo utilizador (caso este não especifique nenhum intervalo são consideradas as últimas 24 horas).

A Figura 9 ilustra o resultado de o utilizador selecionar a opção de ver todas as embarcações com o MONICAP num período de 24 horas. Como é possível visualizar, existem três pontos de cor diferentes para cada rota de uma embarcação: amarelo, azul e magenta. Cada uma destas cores tem um significado. O amarelo é registado de forma periódica, tipicamente em

intervalos de uma hora. Estes pontos têm associada a latitude e a longitude, a velocidade, o rumo, entre outros. O magenta é registado quando existe um cruzamento de áreas como, por exemplo, a passagem da área marítima de um país para a área marítima de outro país ou quando uma embarcação chega a um porto. O azul é registado de dez em dez minutos e tem associada a velocidade de uma embarcação assim como, a sua latitude e longitude.

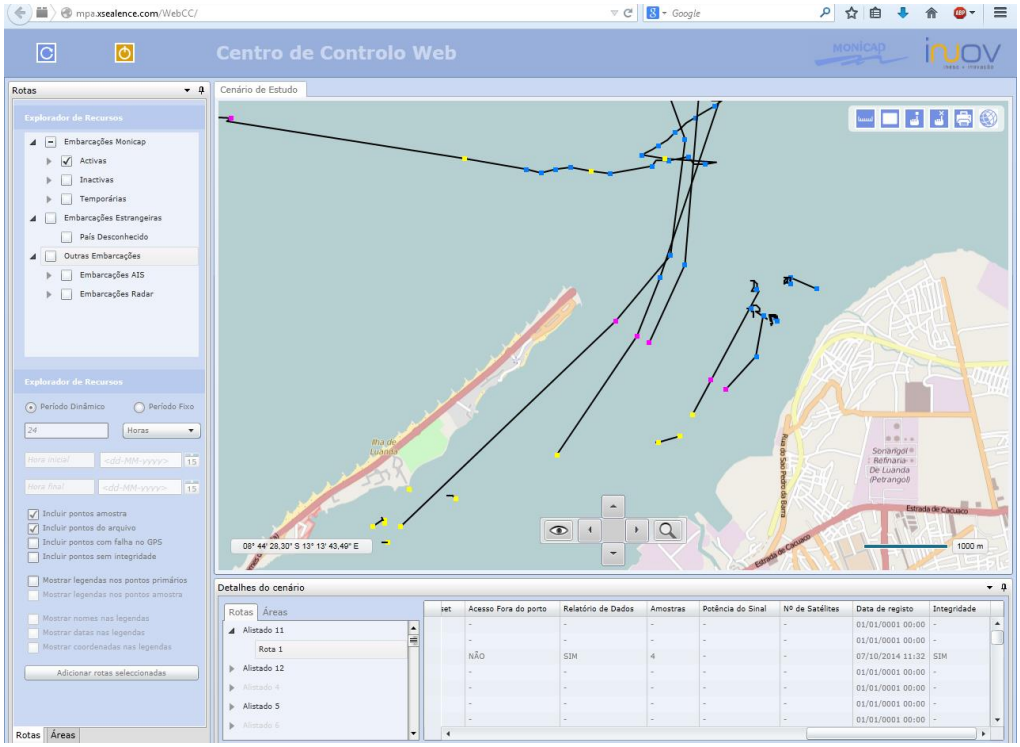


Figura 9 – Todas as embarcações com o MONICAP

Uma outra característica que o webCC apresenta é a possibilidade que o utilizador tem de poder visualizar as várias áreas marítimas que existem. A figura seguinte mostra as áreas marítimas de Portugal apresentadas no webCC.

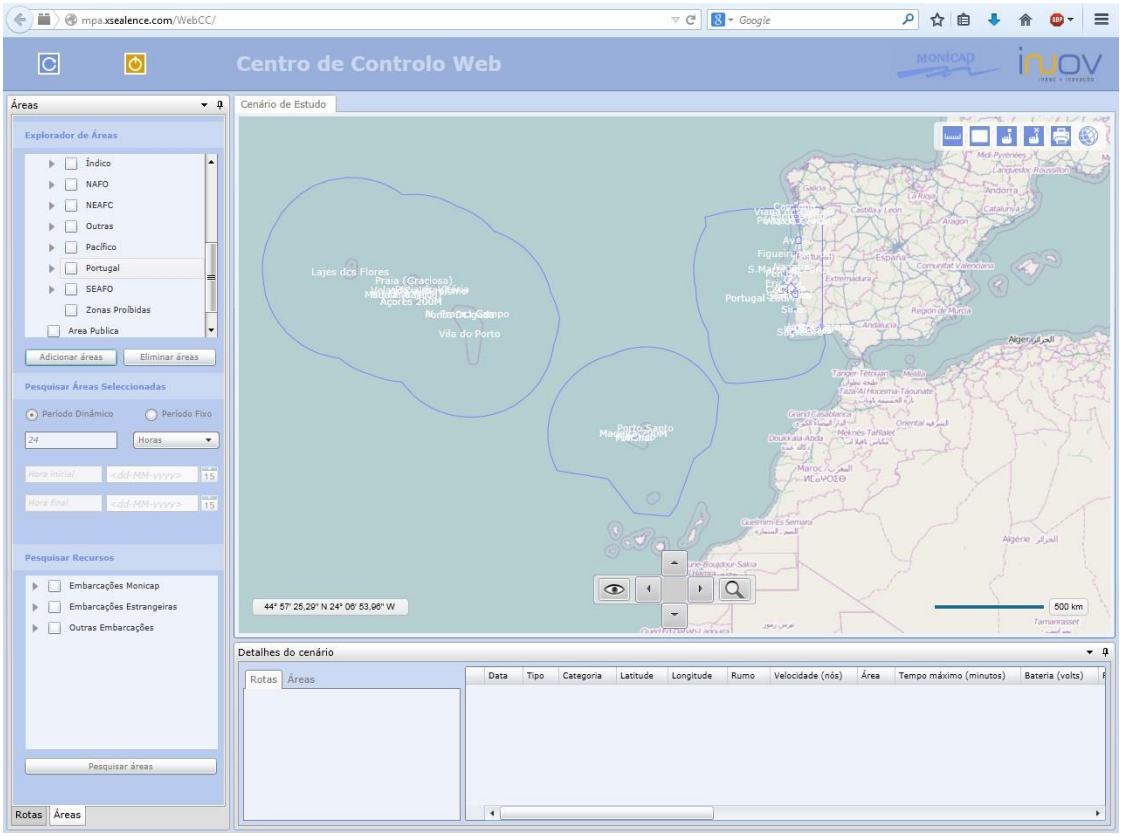


Figura 10 - Áreas marítimas de Portugal

Uma outra possibilidade que o utilizador tem, e que foi já mencionada, é a capacidade de procurar por rotas de embarcações que estejam numa determinada área ou então que tenham passado por essa área num certo período. Para tal o utilizador escolhe qual é a área que pretende pesquisar, o intervalo de tempo e, por fim, qual(ais) o(s) tipo(s) de dispositivo(s) que as embarcações estão a utilizar (Monicap, AIS, radar).

O webCC permite ao utilizador alternar entre mapas do OpenStreetMaps, que são os mapas carregados por omissão, e mapas locais que são mapas "customizados" que pertencem ao utilizador.

2.3.1 O WebCC e o Microsoft Silverlight

O sistema webCC que existia até ao início deste estágio foi desenvolvido com recurso à plataforma Microsoft Silverlight [22], que é uma plataforma aplicacional que permite o desenvolvimento de aplicações *web* ricas (*rich web applications*). O *run-time enviromment* do Silverlight está disponível como um *plugin* para *browsers* que são executados sobre sistemas operativos como o Microsoft Windows o OS X e o Symbian OS. Esta plataforma apresenta, no entanto, algumas limitações como sejam o facto de o suporte para sistemas Android e

iPhone ser quase inexistente. Este aspeto, e o facto de a Microsoft ter anunciado que o suporte para o Silverlight irá terminar em 2021 [23], foram os motivos para o abandono do Silverlight por parte da XSealence no desenvolvimento do WebCC.

2.4 Aplicações semelhantes

De forma a tornar o desenvolvimento do Web Vessel Tracker mais completo foram analisadas algumas aplicações com funcionalidades semelhantes e alguns sistemas VMS. Esta secção aborda vários sistemas existentes relacionados com o projeto desenvolvido, ou seja, sistemas que permitem a visualização da atividade marinha como, por exemplo, a monitorização de embarcações.

A análise de sistemas semelhantes ao Web Vessel Tracker permitiu adquirir conhecimento de natureza vária para o desenvolvimento da aplicação como por exemplo, as melhores formas de apresentar informação ao utilizador.

As aplicações que são apresentadas de seguida permitem fazer a monitorização de embarcações recorrendo ao AIS.

Começamos por descrever o Marine-Traffic, que é a aplicação *web* com mais adesão a nível mundial para a monitorização de embarcações [24]. Esta aplicação permite realizar as seguintes operações:

- Monitorização de embarcações - A aplicação apresenta no mapa uma enorme variedade de embarcações de vários tipos como, por exemplo, embarcações de carga, de pesca, de passageiros, entre outros. Permite ao utilizador visualizar um formulário com detalhes de uma embarcação, como por exemplo, a sua largura e tamanho, o país de origem, fotos da embarcação, as posições sucessivas dessa embarcação, a possibilidade de visualizar uma rota animada dessa embarcação, ver mais detalhes sobre essa embarcação, entre outras. Este conceito de apresentar um formulário com detalhes de uma embarcação acabou por ser incluído no Web Vessel Tracker uma vez que, permite apresentar alguma informação sobre a embarcação de forma simples e fácil de ler. Ao formulário desenvolvido foi ainda adicionada uma opção para visualizar a rota que essa embarcação fez num intervalo de tempo pré-definido. Na eventualidade de o utilizador querer procurar uma embarcação, pode pesquisar por critérios como o país, o tipo de embarcação, a velocidade, o tamanho, o MMSI (número de identificação único de uma embarcação), o nome e se a embarcação esta

ativa ou não. Estes critérios de pesquisa serviram de inspiração para os filtros de pesquisa para as embarcações que foram criados na página Pesquisar;

- Monitorização de portos e marinhas - A aplicação permite visualizar no mapa onde é que as marinhas e os portos estão localizados e quais as embarcações que estão nessa área.

No que toca às tecnologias utilizado pelo Marine-Traffic, foi possível descobrir que este utiliza o AIS para obter informação relativa à localização das embarcações graças a recetores AIS, que estão localizados em bases pertencentes à Marine-Traffic. A informação recebida (que vem encriptada) é depois processada e enviada para uma base de dados central através de um *webservice*. Esta é atualizada de forma contínua (a base de dados), no entanto, existem algumas posições que podem não ser atualizadas imediatamente por vários motivos como, por exemplo, uma embarcação sair do alcance dos recetores. O Google Maps API [25] é utilizado para fazer a representação das embarcações ao utilizador. Para mais informação sobre o Marine-Traffic consultar [26].

No ponto de vista de sistemas VMS semelhantes ao MONICAP foram analisados o Thorium [27] desenvolvido pela CLS America [28] e o Globavista [29], antiga Bluefinger, desenvolvido pela empresa como o mesmo nome.

Os dados que o Thorium recolhe são enviados para um satélite pertencente à rede comercial Iridium, onde é usado o modo de transmissão SBD (*Short Burst Data*). Esta é uma rede de transporte que é utilizada para transmitir mensagens pequenas, entre equipamentos e um sistema de computadores centralizado que serve de *host* [30]. O Thorium vem com uma aplicação para dispositivos móveis como os *tablets* ou *smartphones* que permite ao utilizador comunicar com o sistema e analisar os dados recolhidos por este [31]. Esta aplicação é, essencialmente, o "centro de controlo" do Thorium.

Para além da monitorização de embarcações, o Thorium permite ao seu utilizador ver a previsão meteorológica de uma determinada zona, utilizar formulários eletrónicos para os mais variados fins e enviar *e-mails*, tudo graças à rede Iridium. A figura seguinte ilustra o funcionamento do Thorium.

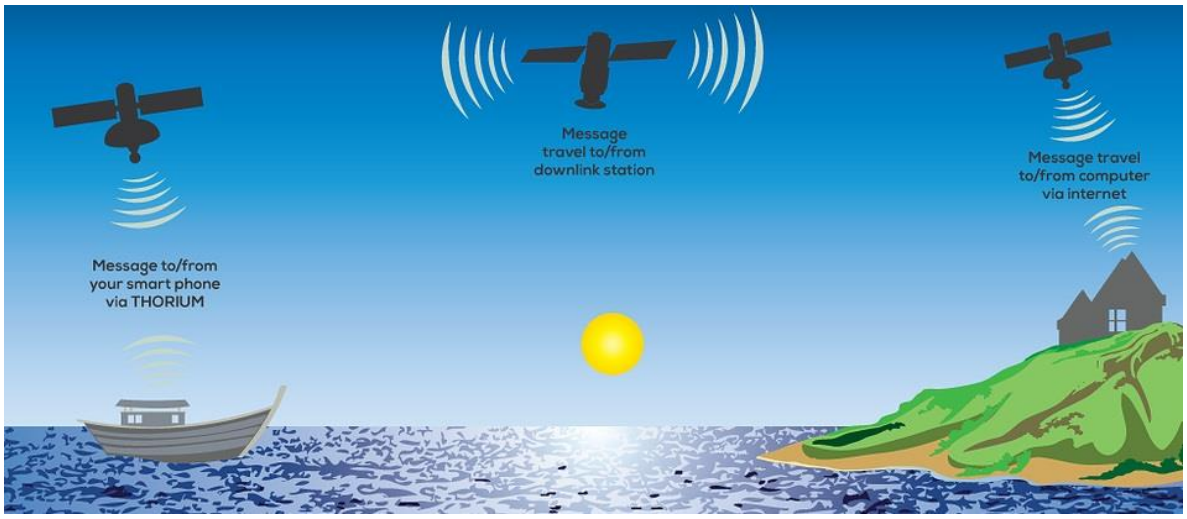


Figura 11 - Processo de recolha de informação do Thorium

Como é possível observar pela imagem (imagem retirada de [32]), a informação que o *Thorium* recolhe é enviada pela rede *Iridium* através do serviço SBD. Esta informação chega ao consumidor final através de um *gateway*. Para mais informações sobre o *Iridium* SBD, consultar a referência [32].

O Globavista é usado pelas autoridades das pescas de países como o Reino Unido e a África do Sul (para a sua zona económica exclusiva) para fazer a monitorização de embarcações. Para além do seu próprio dispositivo, utilizam também o AIS para monitorizar embarcações e as redes de satélites ARGOS [33] e o Inmarsat-C [34]. Apresenta também um sistema que é capaz de guardar a localização e o momento em que a pesca foi feita assim como, a espécie de peixe que foi pescada. Este sistema guarda também o tratamento feito ao peixe pescado, isto é, se foi congelado, estripado ou posto em gelo para um livro de registos digital. Graças a estes registos, o trabalho dos inspetores e reguladores fica mais simplificado uma vez que têm acesso a informação chave, como por exemplo, se as quotas de peixe estão de acordo com a legislação para a preservação da espécie pescada.

Serviços geoespaciais

Este capítulo começa por fazer uma introdução e descrição aos sistemas GIS (*Geographic information system*) e alguns serviços importantes para o desenvolvimento do projeto. De seguida é feita uma descrição do que são geoservidores e o que é o *web mapping*. O capítulo termina com uma descrição feita às plataformas de desenvolvimento para aplicações de *web mapping*.

3.1 Os sistemas GIS

Os sistemas designados de GIS são sistemas computacionais que permitem armazenar, verificar e exibir dados que estão relacionados com posições na superfície da Terra. Este tipo de sistemas que permite que vários tipos de dados sejam mostrados num mapa como, por exemplo, áreas que são bastantes sensíveis à poluição com áreas que produzem bastante poluição. Estes dados, permitem, ao serem analisados, encontrar padrões e relacionamentos que de outro modo seriam mais difíceis de encontrar.

O grau de complexidade que um sistema GIS tem pode variar entre uma simples aplicação *desktop* para um sistema que tenha várias bases de dados dedicadas com vários computadores ligados à mesma rede [35]. Um dos sistemas GIS mais famosos é o ArcGIS [36], que é um *software* proprietário da ESRI (Environmental Systems Research Institute) [37].

Hoje em dia estes sistemas têm uma grande importância em muitas organizações como, por exemplo, os bombeiros, que podem utilizar estes sistemas para planear rotas de emergência ou encontrar possíveis ameaças.

Um dos requisitos com prioridade máxima do Web Vessel Tracker consiste na capacidade de suportar diferentes tipos de mapas (Bing Maps, OpenStreetMaps, etc.). De modo a conseguir encontrar a melhor opção, foram analisadas diversas tecnologias e geoservidores que serão apresentados de seguida.

A OGC (*Open Geospatial Consortium*) [38] é uma organização internacional que conta com a ajuda de vários colaboradores para a criação e desenvolvimento de padrões para conteúdo geoespacial. Os colaboradores são um conjunto de empresas como a Google, várias universidades de todo o mundo e agências governamentais, também de várias partes do mundo [39]. Estas entidades colaboram para que o desenvolvimento e implementação de padrões para conteúdos geoespaciais seja feito de uma forma consensual e aberta.

Dado a crescente necessidade de desenvolver especificações/padrões interoperáveis para tecnologias geoespaciais, foram criados vários protocolos/serviços que permitem essa interoperabilidade.

São apresentados de seguidas dois serviços muito utilizados no desenvolvimento de aplicações GIS:

- O WMS (*Web Map Service*) é um protocolo que é utilizado para mostrar mapas georreferenciados pela Internet a partir de um servidor de mapas que utiliza uma base de dados GIS [40]. Um pedido WMS define a área de interesse assim como as camadas geográficas a serem processadas. As respostas para um destes pedidos são imagens de mapas que conseguem ser exibidas pelo *browser*;

- O WFS (*Web Feature Service*) é um serviço permite a um cliente receber e atualizar dados geoespaciais que estão codificados em GML (*Geography Markup Language*) de vários WFS [41]. O WFS está dividido em dois "tipos", o básico que permite a consulta e obtenção de características, e o transacional (WFS-T), que permite criar, apagar e atualizar características.

3.2 Servidores para *web mapping*

O *web mapping* consiste no processo de apresentar mapas interativos em *browsers* a partir de um sistema GIS. Por seu lado, um servidor *web* que apresenta capacidades para permitir *web mapping* denomina-se por servidor de mapas *web* (*web map server*).

Numa fase inicial do estágio foram analisados vários servidores com capacidade de suportar pelo menos os serviços WMS e WFS. Os servidores que foram analisados foram o GeoServer, o MapServer [42] e o MapGuide [43]:

- O GeoServer é um servidor *open-source* desenvolvido em Java que permite aos seus

utilizadores processar e editar dados geoespaciais. Este servidor suporta serviços da OGC como o WMS e WFS, permitindo desta forma que um utilizador possa publicar mapas baseados em *web* tais como Bing Maps, Google Maps, OpenStreetMaps, entre outros. O servidor vem com a biblioteca de mapeamento *open-source* OpenLayers [44] integrada. Mais abaixo, será realizada uma análise ao OpenLayers. Este servidor foi o que foi utilizado na versão anterior deste projeto, o webCC;

- O MapServer é um projeto *open-source* que foi desenvolvido em C e C++, com o objetivo de permitir o desenvolvimento de aplicações com capacidade geográfica na *web*. Tem suporte para uma variedade de padrões da OGC tais como WMS (cliente/servidor), WFS não transacional (cliente/servidor), WCS, etc. A existência de um ficheiro denominado por *mapfile* permite que o programador possa fazer proceder a configurações como, por exemplo, a forma como os mapas são apresentados, ou seja, é possível configurar as cores, ângulo, entre outros aspetos;

- O MapGuide é um projeto *open-source* que foi desenvolvido para permitir aos seus utilizadores desenvolver e lançar aplicações de mapeamento e *web services* geoespaciais. O MapGuide tem suporte para os serviços WMS e WFS segundo os padrões da OGC. O MapGuide utiliza uma base de dados baseada em XML que suporta a maior parte dos tipos de ficheiros geoespaciais. Esta plataforma pode ser utilizada em ambiente Windows ou Linux e tem suporte para os servidores *web Apache* e IIS (Internet Information Services).

Apesar de esta análise ter sido realizada, na prática, não se verificou a necessidade de utilizar um geoservidor uma vez que a plataforma de *web mapping* utilizada (ver próxima secção) faz também a gestão da renderização dos *map tiles*.

3.3 Plataformas de desenvolvimento para aplicações de *web mapping*

Para o desenvolvimento do Web Vessel Tracker foi necessário analisar plataformas *web mapping*. As plataformas que foram analisadas foram o OpenLayers, o Leaflet [45] e o MapFish [46]:

- O OpenLayers é uma biblioteca JavaScript *open-source* que foi desenvolvida de modo a permitir a exibição de mapas em *browsers*. Esta biblioteca encontra-se na sua 3ª versão desde o dia 29 de agosto de 2014 e apresenta várias características como:

camadas vetoriais; possibilidade de personalizar os controlos de mapas só com CSS; encontra-se já preparada com suporte para as plataformas móveis; e a possibilidade de incluir qualquer plataforma de mapas (ex: Bing Maps) que utilize os padrões da OGC como, por exemplo, WMS e WFS. Esta biblioteca é distribuída sob uma licença 2-Clause BSD [47];

- O Leaflet é uma biblioteca JS open-source que permite o desenvolvimento de aplicações de *web mapping*. No que diz respeito ao suporte de serviços OGC, no período em que a análise foi realizada, esta biblioteca tinha suporte nativo para alguns serviços como o WMS mas não para outros como o WFS ou o GML, sendo assim necessário recorrer a *plugins* de terceiros. Muitos dos *plugins* disponíveis podem ainda ser encontrados no *website* da Leaflet. O Leaflet tem a particularidade de ser bastante leve em comparação com outras bibliotecas. Apresenta também uma licença do tipo 2-Clause BSD;

- O MapFish é uma plataforma de desenvolvimento de *web mapping open-source* que foi estendida da plataforma de desenvolvimento *Pylons* (uma plataforma de Python para *web*) [48]. Esta plataforma utiliza as bibliotecas OpenLayers e GeoExt (a referência [49] contém informações adicionais sobre a plataforma) permitindo, a utilização dos serviços WMS, WFS, WMC, entre muitos outros. Apresenta uma licença do tipo 2-Clause BSD.

Uma vez feita a análise destas três plataformas, foi escolhido o OpenLayers 3 dado que na altura apresentava uma grande variedade de funcionalidades nativas, algo que não acontecia com a restantes. Foi descartada a utilização da versão 2.13.1 [50], a última versão antes da versão 3.0, dado que deixou de ter suporte, isto é *releases* novas, apesar de, na altura, aquela versão possuir mais funcionalidades estáveis. Para além de apresentar uma variedade mais diversificada de funcionalidades, esta plataforma tinha também uma documentação muito rica e bem estruturada, assim como, um conjunto de exemplos interativos de várias funcionalidades suportadas. O OpenLayers apresenta também um grande número de utilizadores, o que gera uma comunidade ativa e atualizações regulares, pormenores que contribuíram para a escolha do OpenLayers em relação à concorrência. A ilustração seguinte mostra a atividade que existe no StackOverflow [51] de cada plataforma, dado que é o maior meio de comunicação de programadores do mundo. Foram pesquisadas as *tags* OpenLayers,Openlayers 3, Leaflet e MapFish. Estes dados foram obtidos no dia 21 de

setembro de 2015.

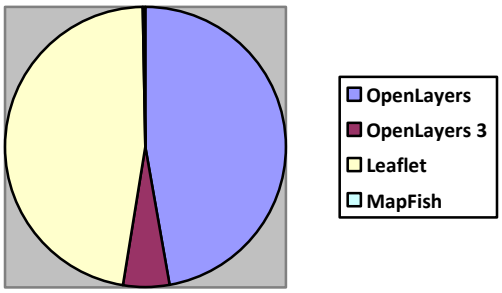


Figura 12 - Atividade das plataformas de *web mapping* analisadas

Como é possível verificar, a diferença entre o OpenLayers e o Leaflet é mínima, no entanto, o OpenLayers existe há mais tempo e estes números contam com as versões anteriores. O facto do OpenLayers se encontrar escrito em JavaScript foi também um factor de escolha dado que contribuiu para uma menor curva de aprendizagem, dado que já existiam conhecimentos prévios de JavaScript.

3.4 Plataformas de mapas

Nesta secção são apresentadas algumas das plataformas de mapas *web* mais usadas a nível mundial. Os mapas apresentados nesta secção são os Bing Maps [52], o Google Maps e por fim os OpenStreetMap. Todas as plataformas de mapas *web* analisadas utilizam a projeção de Mercator. Este é um estilo de mapas que tem uma distorção na área e na escala perto dos polos. No entanto, este tem duas características que compensam esse aspeto: o facto de ser cilíndrica garante que o Norte está sempre em cima, o Sul sempre em baixo, o Este sempre à direita, e o Oeste sempre à esquerda; a outra característica é que consegue preservar imagens pequenas, como é o caso dos edifícios, em que a sua imagem de cima não é distorcida.

Existe uma desvantagem neste estilo de mapas que é o facto da existência de uma latitude máxima de aproximadamente 85,05º, algo que é causado pela projeção se prolongar até ao infinito nos polos, levando a que o globo inteiro não seja visível.

A figura (retirada de [52]) que é apresentada de seguida é uma ilustração da projeção de Mercator.

Figura 13 - A projeção de Mercator

As plataformas de mapas analisadas têm capacidade para *forward* e *reverse geocoding*. O *forward* geocoding é a capacidade de receber uma localidade, por exemplo um centro comercial e conseguir obter as coordenadas da latitude e longitude deste. O *reverse geocoding* é a capacidade de através das coordenadas de latitude e longitude conseguir encontrar um objeto/rua/etc.

Uma outra característica que as plataformas analisadas apresentam é uma grande variedade de documentação e exemplos existentes. Desta forma, a resolução de eventuais problemas de integração destas plataformas podem ser mais facilmente resolvidos.

De seguida é feita uma caracterização das três plataformas de mapas *web* analisadas:

- O Bing Maps, antigo Microsoft Virtual Earth, apresenta níveis de *zoom* mais elevados que o Google Maps e os OMS assim como uma maior variedade de tipos de mapas, como por exemplo, o *bird´s eye* (olho de pássaro), que permite ao utilizador ter uma vista aérea sobre os edifícios com melhor perceção de profundidade, entre muitas outras características. O Bing Maps apresenta dois tipos de licença, uma gratuita e uma paga. A licença gratuita limita o número de transações faturáveis por ano em 125 000, enquanto a licença paga não apresenta essa limitação;

- Das plataformas analisadas o Google Maps é, provavelmente, a mais conhecida devido à popularidade do Google. Esta plataforma apresenta tipicamente uma maior quantidade de informação disponível assim como um maior grau de precisão uma vez que é das plataformas mais antigas no mercado. O Google Maps apresenta dois tipos

de licença: uma gratuita com a qual, se forem carregados mais de 25 000 mapas por dia num espaço de 90 dias, será necessário comprar uma licença para o continuar a utilizar ou então pagar pelo excesso de carregamentos efetuados [53]; A versão paga não apresenta qualquer limitação;

- O OpenStreetMap é uma iniciativa que tem como objetivo criar e distribuir dados geográficos de uma forma gratuita. Esta plataforma de mapas pertence à OpenStreetMap Foundation [54] que é uma organização sem fins lucrativos e que conta com a ajuda dos membros da sua comunidade para a obtenção de dados geográficos. O funcionamento do OSM é bastante parecido com o da Wikipedia, isto é, existe um grupo de pessoas que adicionam conteúdo e ao mesmo tempo existe um grupo de pessoas que garantem que esses conteúdos são inseridos/editados de forma correta. Da mesma foram que a Wikipedia está dependente de donativos e de *crowdfunding*, o OMS também depende dos donativos e *crowdfunding* para a obtenção de fundos. Existem inúmeras entidades que utilizam o OSM como a plataforma para a representação de maps *web*, como é o caso do Banco Mundial, Cruz Vermelha, Wikipedia, entre outros. Como foi já referido, o webCC utiliza os *map tiles* da OMS. Esta plataforma apresenta uma licença do tipo ODbL [55], cuja única implicação que tem é dar crédito ao OpenStreetMaps e aos seus contribuidores.

Metodologia e Gestão

Este capítulo destina-se a descrever a metodologia de desenvolvimento utilizada no desenvolvimento do projeto, assim como o seu planeamento e respetiva gestão.

4.1 Metodologia de desenvolvimento

No desenvolvimento do projeto foi escolhido um modelo baseado no modelo em cascata (waterfall model) [56]. O modelo é tipicamente aplicado quando se sabe à partida que é possível realizar o levantamento completo dos requisitos de um projeto, ou a sua maior parte, e os *inputs* e os *outputs* que este tem com sistemas externos (caso existam) [57], como é o caso deste projeto. Tal acontece porque este é um modelo que está estruturado em etapas não reversíveis por omissão.

A Figura 14 ilustra as várias etapas que compõem o modelo em cascata tradicional.

Como foi já referido, e como é possível observar na figura, existem várias etapas no modelo em cascata que podem ser descritas da seguinte forma:

- Requisitos – Esta é a etapa onde é feita uma análise ao projeto que se pretende desenvolver e onde são recolhidos os requisitos e tecnologias necessárias para o projeto;

- Desenho/prototipagem - Esta fase é caracterizada pelo desenho de vários protótipos que ilustram a forma como o produto irá funcionar. O objetivo desta fase é conseguir "passar para uma tela" as várias ideias de como é que o projeto vai funcionar. Desta forma, é possível evitar eventuais problemas e garantir que o que foi desenhado está de acordo com o pretendido. Nesta fase, é também concebido um desenho da arquitetura do sistema/aplicação e das interações que este terá com fontes externas;

- Implementação – Esta é a fase onde os requisitos funcionais para o projeto que foram

recolhidos na primeira fase são implementados;

- Verificação/Validação - Esta é uma etapa crucial porque é nela que são realizados os testes, como por exemplo, testes de carga, de usabilidade, entre outros. Caso existam erros, é nesta etapa que se espera que sejam descobertos e corrigidos;

- Manutenção – Esta é a etapa em que o projeto, que já está em produção, vai recebendo a manutenção necessária, nomeadamente a correção de erros que não tenham sido identificados em fases anteriores, otimizações e atualizações de ordem vária.

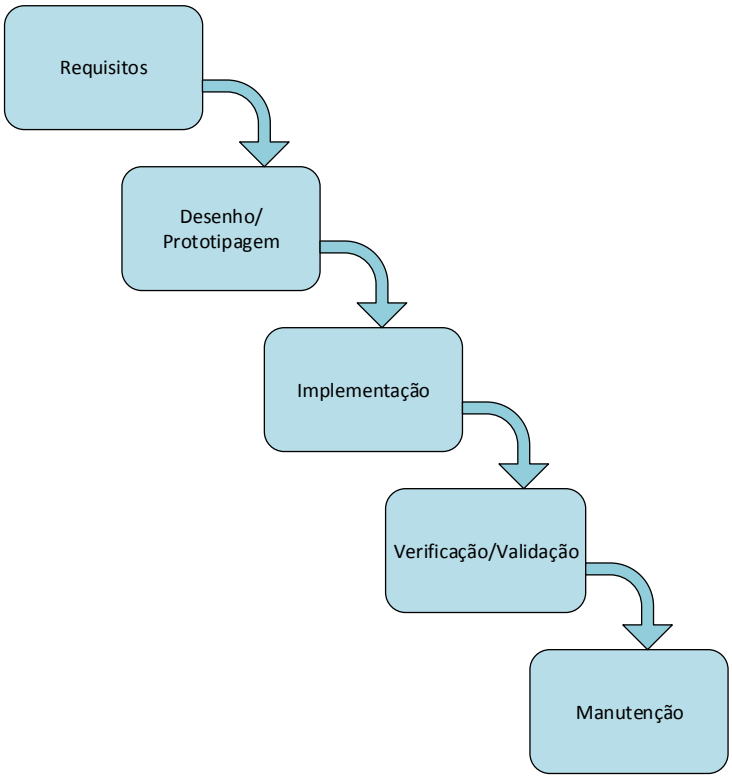


Figura 14 - Passos típicos do modelo em cascata

O modelo em cascata é um modelo que é caracterizado por ser bastante simples de perceber/entender. Este modelo é no entanto considerado um modelo muito teórico e irrealista de seguir na sua totalidade [58]. Tal acontece porque este é um modelo unidirecional, o que significa que, quando seguido à risca, determinadas ações não sejam possíveis, como por exemplo, não se poder voltar a uma etapa anterior, ou então esperar que não hajam alterações nos requisitos. De forma a reduzir os efeitos negativos que o modelo em cascata tradicional apresenta, foi decidido adaptá-lo para um modelo bidirecional. Este modelo, ao contrário do anterior, permite retroceder a fases anteriores, por exemplo, no caso de se verificar que um requisito precisa de ser atualizado/alterado ou na eventualidade de uma funcionalidade não

estar implementada da forma pretendida.

Embora neste projeto não tenha existido um cliente concreto, esse papel foi desempenhado pelo orientador da empresa. Foi com ele que os requisitos foram definidos e aprovados, assim como, a verificação da implementação dos mesmos, isto é, se os mesmos estavam de acordo com o pretendido.

Este novo modelo passa a incluir o "cliente" (de agora em diante cliente refere-se à pessoa que orientou o estágio na empresa) a participar no desenvolvimento do projeto. Deste modo, a probabilidade de aparecerem problemas fica bastante reduzida.

É agora feita uma ilustração e descrição do modelo utilizado para o desenvolvimento do Web Vessel Tracker.

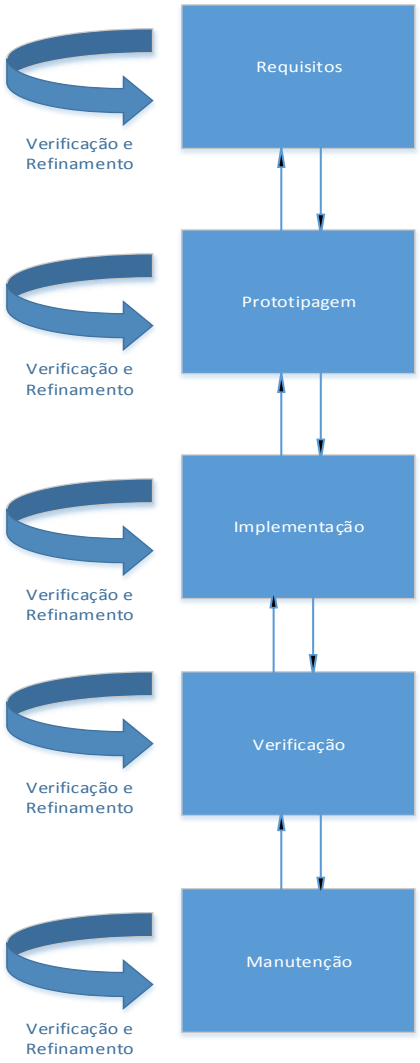


Figura 15 - Metodologia de desenvolvimento utilizada

Como é possível observar pela figura, foram realizadas algumas modificações ao modelo em cascata tradicional, nomeadamente ao nível da passagem das etapas. Neste novo modelo, no decorrer de cada etapa, em particular no final, verifica-se se todos os objetivos delineados foram atingidos e avalia-se a necessidade de retroceder a fases anteriores. Na prática, em cada etapa, os objetivos atingidos eram avaliados regularmente com o orientador da empresa, de forma a receber o seu *feedback* e, em caso de necessidade, retrocedia-se para um fase anterior.

4.2 Gestão do projeto

O estágio teve início a 15 de setembro de 2014 e terminou no dia 15 de junho de 2015. Durante o primeiro mês do estágio foi realizado um levantamento do estado da arte das tecnologias GIS. Esta fase serviu para adquirir conhecimentos sobre as várias tecnologias GIS que existem, a forma como estas podiam ser integradas no projeto e os vários conceitos inerentes a este tipo de tecnologias como, por exemplo, o conceito de *feature* e *layer*, ou então o conceito de *map tile*. Ainda neste primeiro mês foram analisadas diversas aplicações VMS e aplicações *web* que permitem monitorizar embarcações. O objetivo desta fase consistiu em analisar características que pudessem ser utilizadas no Web Vessel Tracker, tendo também facilitado o desenho dos "*mockups*" e, por conseguinte, a entrada na segunda fase do estágio.

A segunda fase começou no dia 16 de outubro de 2014. Esta fase consistiu na análise dos requisitos e no desenvolvimento de "*mockups*" para o Web Vessel Tracker, assim como, na recolha de mais informação sobre os sistemas e aplicações VMS semelhantes ao MONICAP. Esta fase terminou depois de serem aplicadas as melhorias que foram sugeridas depois de uma apresentação dos *mockups* à XSealence em Lisboa.

A terceira fase do estágio começou no dia 11 de novembro com o início da implementação dos requisitos definidos para o Web Vessel Tracker, e prolongou-se até à entrega deste documento. Esta fase, começou pelo desenvolvimento de uma *spike solution*, cujo o objetivo consistiu na adaptação ao OpenLayers. Esta fase serviu para realizar algumas tarefas simples, utilizando dados *dummy* como, por exemplo, criar um ícone no mapa, trocar os *map tiles* e a criação de uma rota. Uma vez terminada a *spike*, começou o desenvolvimento propriamente dito. Esta fase começou com a integração da *spike* com o projeto "real", aapós a qual se dividiu o desenvolvimento de acordo com os requisitos. Isto significa que houve, por exemplo, uma fase para fazer a implementação para as rotas e outra para aos diários de pesca. Durante a fase de desenvolvimento, ocorreram diversas discussões e reuniões com o orientador da empresa, de modo a melhorar a componente de interação com o utilizador.

Desta forma, o estado atual da aplicação encontra-se diferente das *mockups* que foram apresentadas (algumas das *mockups* mais representativas constam do Anexo III).

No Anexo I é possível visualizar num quadro todos os requisitos que foram levantados no âmbito do estágio e no Anexo II é possível visualizar todas as tarefas realizadas relativas ao projeto Web Vessel Tracker.

4.3 Análise de requisitos

Os requisitos que, se encontram descritos no Anexo I, foram levantados após várias conversas e reuniões com o "cliente". Após os requisitos estarem bem definidos, foi necessário fazer uma análise de quais as melhores tecnologias a utilizar no projeto. Apesar de grande parte dos requisitos serem iguais aos do webCC, as tecnologias utilizadas neste não podiam ser usadas porque estavam já ultrapassadas. Um aspeto identificado desde muito cedo foi o facto de grande parte dos requisitos necessitarem de uma biblioteca/*framework* capaz de utilizar o maior número possível de *map tiles*, assim como, representar e manipular informação nesses nos mapas gerados através dos *map tiles*. Como foi já explicado no Capítulo 3, a biblioteca escolhida foi o OpenLayers 3.

O capítulo seguinte aborda o desenvolvimento do projeto em detalhe, assim como, as tecnologias que foram utilizadas.

Desenvolvimento

Este capítulo descreve o desenvolvimento do projeto ao longo do estágio. O capítulo começa por fazer uma introdução à aplicação seguida da aquitetura do projeto. Uma vez terminada esta descrição, são apresentadas e descritas as tecnologias/ferramentas que foram utilizadas. O capítulo termina com a descrição do desenvolvimento propriamente dito da aplicação Web Vessel Tracker.

O Web Vessel Tracker foi desenvolvido de modo a conseguir dar suporte ao maior número de pessoas possível. Desta forma, foi feito um estudo dos *browsers* mais utilizados e qual a sua quota de mercado a nível mundial. A figura seguinte (retirada de [59]) mostra a utilização dos principais *browsers* durante o ano de 2014.

2014	Chrome	IE	Firefox	Safari	Opera
December	61.6 %	8.0 %	23.6 %	3.7 %	1.6 %
November	60.1 %	9.8 %	23.4 %	3.7 %	1.6 %
October	60.4 %	9.5 %	23.4 %	3.9 %	1.6 %
September	59.6 %	9.9 %	24.0 %	3.6 %	1.6 %
August	60.1 %	8.3 %	24.7 %	3.7 %	1.8 %
July	59.8 %	8.5 %	24.9 %	3.5 %	1.7 %
June	59.3 %	8.8 %	25.1 %	3.7 %	1.8 %
May	59.2 %	8.9 %	24.9 %	3.8 %	1.8 %
April	58.4 %	9.4 %	25.0 %	4.0 %	1.8 %
March	57.5 %	9.7 %	25.6 %	3.9 %	1.8 %
February	56.4 %	9.8 %	26.4 %	4.0 %	1.9 %
January	55.7 %	10.2 %	26.9 %	3.9 %	1.8 %

Figura 16 - Estatísticas de utilização de web browsers de 2014

A tabela seguinte mostra as estatísticas de utilização de *browsers* muito conhecidos e utilizados hoje em dia. Como é possível observar, o Google Chrome é o *browser* mais utilizado por uma larga margem à escala mundial. O Firefox encontra-se em segundo lugar com uma média de 24,825% e o IE (Internet Explorer) está em terceiro lugar com uma média de 9,23%. Com isto em mente, o Web Vessel Tracker foi desenvolvido a pensar nesses três *browsers*.

Posteriormente foi feita uma segunda análise à taxa de utilizações dos mesmos *browsers*. A figura seguinte, retirada de [59], mostra essa taxa para o ano de 2015.

2015	Chrome	IE	Firefox	Safari	Opera
August	64.0 %	6.6 %	21.2 %	4.5 %	2.2 %
July	63.3 %	6.5 %	21.6 %	4.9 %	2.5 %
June	64.8 %	7.1 %	21.3 %	3.8 %	1.8 %
May	64.9 %	7.1 %	21.5 %	3.8 %	1.6 %
April	63.9 %	8.0 %	21.6 %	3.8 %	1.5 %
March	63.7 %	7.7 %	22.1 %	3.9 %	1.5 %
February	62.5 %	8.0 %	22.9 %	3.9 %	1.5 %
January	61.9 %	7.8 %	23.4 %	3.8 %	1.6 %

Figura 17 - Estatísticas de utilização de web browsers de 2015

Observando a imagem é possível verificar um crescimento de utilização do Google Chrome, um ligeiro crescimento do Safari e do Opera, enquanto que o IE e o Firefox perderam alguma utilização. Apesar disto, a escolha de dar suporte para o Google Chrome, Firefox e IE continua a ser o suficiente para chegar a mais de 90% das pessoas.

5.1 Arquitetura do sistema

Nesta secção, é descrita a arquitetura do Web Vessel Tracker, bem como o funcionamento do sistema como um todo. É importante referir que o projeto utiliza vários outros projetos, que foram desenvolvidos pela XSealence, que possibilitam que a informação recolhida através dos sistemas AIS e MONICAP possa ser utilizada pelo Web Vessel Tracker. Com recurso a estes projetos é possível mostrar ao utilizador vários tipos de informação, como é o caso do percurso que uma embarcação fez num determinado período, o tipo/quantidade de peixe que uma embarcação que esteja a utilizar o MONICAP capturou, entre outras. De seguida, é feita

uma ilustração do percurso que os dados recolhidos pelo MONICAP fazem até chegar ao centro de controlo para depois serem utilizados pelo Web Vessel Tracker.

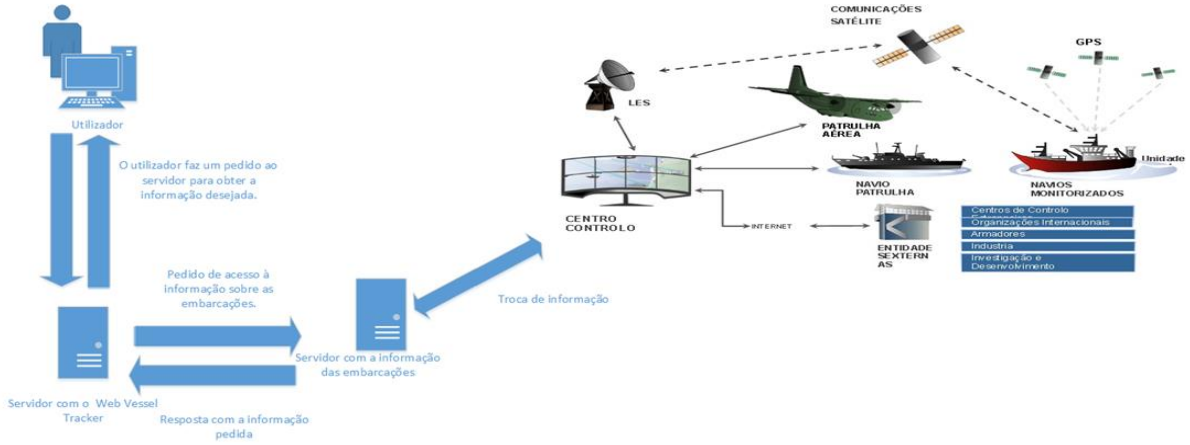


Figura 18 - Percurso dos dados recolhidos pelo MONICAP até chegar à aplicação

Esta "viagem" que a informação faz foi já descrita na Secção 2.4, uma vez que a forma como a informação chega até ao servidor não mudou, isto é, continua a ser igual ao do webCC.

O sistema MONICAP recolhe diversos dados das embarcações como, por exemplo, a localização, rumo, velocidade, peixe pescado, entre outros. Esses dados são depois enviados através de uma rede comercial de satélites para o centro de controlo, onde podem ser posteriormente utilizados pelo Web Vessel Tracker.

A Figura 19 ilustra a arquitetura da aplicação. Como é possível observar, as respostas que o cliente recebe do servidor vêm em formato JSON, enquanto que os pedidos que este faz ao servidor são feitos em Ajax (*Asynchronous JavaScript and XML*). Estes são pedidos assíncronos que o *browser* faz ao servidor sem ser necessário ter que refrescar a página corrente. Estes pedidos são, por vezes, também desginados por pedidos XHR (XmlHttpRequest).

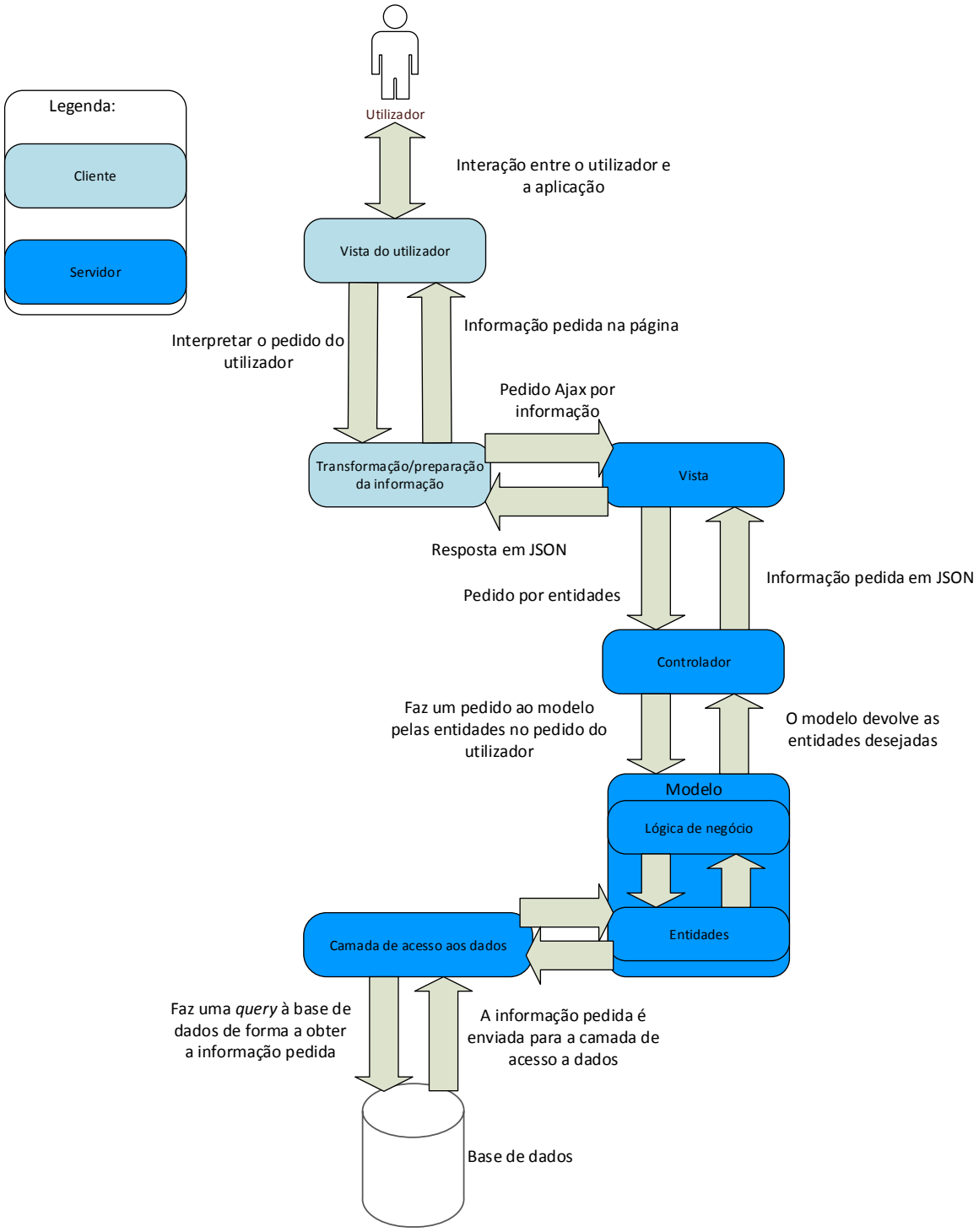


Figura 19 - Esquema da arquitetura da aplicação

No Web Vessel Tracker o acesso à base de dados (a base de dados central) não é feito de forma direta, mas sim através de uma camada intermédia que desempenha o papel de camada de acesso a dados. Esta camada foi desenvolvida pela XSealence e permite que os dados desejados sejam devolvidos, por exemplo, numa arraylist. Devido à necessidade de desenvolver uma aplicação funcional o mais rápido possível, não foi utilizado nenhum mecanismo de ORM (Object-relational mapping) no desenvolvimento. A razão principal

passa pelo facto de a base de dados existente ser muito complexa, o que implicaria um investimento considerável de tempo para a conhecer e dominar, e o facto de a camada intermédia existente permitir evitar esse investimento.

Esta camada devolve dados que têm de ser tratados, isto é, os objetos que são devolvidos pela camada intermédia contêm informação a mais ou que necessita de ser traduzida de inglês para outra língua, como por exemplo, português. Uma vez concluído o "tratamento" dessa informação nas classes do modelo responsáveis pela lógica de negócio, são criadas as respetivas instâncias das entidades pertencentes ao modelo.

Sempre que a vista necessita de informação nova, seja para fazer um refrescamento da posição de uma embarcação, seja para ver a rota de uma embarcação, é realizado um novo pedido à base de dados através de uma tecnologia cliente, o Jquery. Através desta tecnologia é enviado um pedido Ajax ao controlador pela informação desejada. Por sua vez, o controlador envia ao modelo um pedido para obter a informação. No modelo, as classes responsáveis pela lógica de negócio obtêm a informação desejada através da camada intermédia. Esta comunica com a base de dados utilizando as *queries* necessárias para obter a informação e envia-a de volta para as classes da lógica de negócio. Nestas classes são feitas as transformações necessárias à informação obtida, de modo a conseguir criar as entidades necessárias para satisfazer o pedido. Uma vez criadas, as entidades são enviadas ao controlador, que cria o JSON correspondente que é depois enviado para a vista. No caso de se tratar de um pedido por uma página, esta é obtida para depois ser renderizada no cliente. Caso se trate de um pedido Ajax, após a resposta do servidor em JSON, este é processado no cliente com recurso a tecnologias como, por exemplo, o JavaScript e OpenLayers e o resultado incorporado na página que é apresentada ao utilizador com a informação pedida.

5.2 Tecnologias utilizadas

Para o desenvolvimento do Web Vessel Tracker foram utilizadas várias tecnologias e ferramentas diferentes. A plataforma de desenvolvimento utilizada foi o .NET (versão 4.5.1), uma vez que é o ambiente de desenvolvimento mais utilizado na XSealence. Desta forma, foi possível integrar facilmente no Web Vessel Tracker os projetos (daqui em diante o termo projeto refere-se a um projeto que foi criado no Visual Studio) que permitem receber informação relativa às embarcações (estejam estas a utilizar o MONICAP ou AIS) que tinham já sido desenvolvidos em C# pela XSealence.

O ASP.NET MVC 5 foi a plataforma de desenvolvimento web utilizada, dado que esta implementa o padrão MVC (as características deste padrão serão abordadas numa secção mais adiante), o que leva a uma melhor organização da aplicação. O IDE (*Integrated Development Environment*) que foi utilizado para o desenvolvimento da aplicação foi o Visual Studio 2013 [60], que é proprietário da Microsoft e que no início da implementação era a versão mais recente do Visual Studio. Foi ainda utilizado o HTML5, isto é, algumas *tags* específicas do mesmo direta e indiretamente como, por exemplo, a *tag canvas* que é utilizada pelo OpenLayers para renderizar um mapa. É importante referir que algumas *tags*, como por exemplo, *details* foram descartadas devido à falta de compatibilidade com alguns dos *browsers* escolhidos para dar suporte. Foi também utilizado o Razor que é um *view engine* utilizado para criar páginas *web* dinâmicas com recurso a C#. O Razor permite que não seja necessário ter *tags* ou blocos explícitos para comunicações com o servidor, o que resulta numa sintaxe mais compacta e limpa [61].

Como foi já referido anteriormente (na Secção 3.3), a plataforma de desenvolvimento para aplicações de *web mapping* utilizada foi o Openlayers 3. Foi com esta plataforma que foram implementadas várias funcionalidades relativas à representação geográfica das embarcações como, por exemplo, a representação das rotas que estas fizeram num determinado período de tempo.

Para fazer a representação da informação relativa aos vários pontos de uma embarcação foi utilizada a biblioteca Knockout JS [62]. Esta é uma implementação Javascript do padrão MVVM (*Model View ViewModel*), e permite fazer uma separação dos dados do domínio, das componentes da vista e dos dados a serem disponibilizados. Com o Knockout foi criada a tabela que apresenta ao utilizador os detalhes da rota de uma embarcação na página principal. Quando o utilizador decide visualizar a rota de uma embarcação diferente, a tabela é atualizada com a informação desejada de uma forma transparente e rápida. De modo a melhorar o aspeto da aplicação, foi utilizada a biblioteca de CSS 3 (*Cascading Style Sheet*) Bootstrap [63]. Esta biblioteca já vem incluída com a instalação do Visual Studio. Para o desenvolvimento do Web Vessel Tracker foi utilizada a linguagem de programação C#. De forma a tratar de comunicação com o cliente, foi utilizada a linguagem de programação JavaScript [64]. Com recurso a esta linguagem de programação foram desenvolvidos *scripts*, que permitem "capturar" as interações que o utilizador tem com a aplicação e gerar as devidas respostas. As respostas geradas podem ser ou não do lado do cliente, isto é, o utilizador pode fazer uma acção que necessite de alguma informação que esteja do lado do servidor. A forma

como a informação que está do lado do servidor é tratada/recebida será apresentada na próxima secção. Para simplificar o desenvolvimento de código JavaScript foi também utilizada a biblioteca JavaScript conhecida como Jquery [65]. Esta biblioteca permite que operações como a manipulação de eventos, animações e pedidos Ajax sejam realizadas de forma mais rápida e intuitiva para o programador.

A figura seguinte ilustra as principais tecnologias utilizadas no desenvolvimento do Web Vessel Tracker.

Figura 20 - Tecnologias utilizadas no desenvolvimento

Como é possível observar, grande parte das tecnologias utilizadas são *cliente-side*, como é o caso do JavaScript, jQuery, OpenLayers 3 e o Knockout. A utilização de tecnologias do lado do cliente permitem que a carga do servidor seja aliviada e que esse código seja executado de forma mais imediata, entre outras vantagens.

Apesar de não terem sido usadas diretamente, as bibliotecas Respond e Modernizr js foram também utilizadas. Estas bibliotecas permitem que os *browsers* consigam fazer o tratamento de elementos CSS e JavaScript que estes não suportem. Para mais informação sobre como cada uma destas bibliotecas funciona, consultar a referência [66] para o Modernizr e a referência [67] para o Respond.

5.3 O Servidor

Esta secção começa por fazer uma descrição ao padrão utilizado durante o desenvolvimento. É depois feita uma descrição das decisões tomadas no desenvolvimento do modelo, seguido das vistas e terminando por fim nos controladores.

Para o desenvolvimento deste projeto foi utilizada a plataforma de desenvolvimento

ASP.NET MVC5 e utilizado o *web server* IIS para fazer o alojamento da aplicação. Esta plataforma é uma implementação do MVC "tradicional" para aplicações *web*. De seguida é feita uma descrição de cada componente deste MVC.

5.3.1 O padrão MVC

O MVC "tradicional" é um padrão de desenvolvimento que tem como objetivo estruturar e simplificar a construção de aplicações. Este padrão de desenvolvimento foca-se em estruturar uma aplicação em três camadas: o *Model* (o modelo), a *View* (a vista) e o *Controller* (o controlador). Esta separação é feita de modo a que o desenvolvimento de uma aplicação, ou projeto, seja mais organizada. A versão ASP .NET MVC5 continua a promover esses objetivos no desenvolvimento de aplicações *web*. De seguida é feita uma pequena caracterização de cada uma destas camadas que existem tanto no MVC "tradicional" como na versão ASP.NET MVC 5.

- O Modelo é a camada que é responsável por fazer o tratamento e gestão dos dados da aplicação. Tipicamente são os objetos do modelo que vão obter ou armazenar informação da/na base de dados [68]. Por exemplo, um objeto do tipo fishingVessel representa e obteria da base de dados informação relativa a barcos de pesca;

- A Vista é a camada que é responsável pela interação com o utilizador, a UI (*user interface*). A informação que é mostrada pode vir de um pedido que é feito ao modelo, caso seja algo que esteja na base de dados. Por exemplo, a lista de todos os tipos de embarções existentes na aplicação Web Vessel Tracker;

- O Controlador é a camada que é responsável por tratar dos pedidos que o *browser* faz ao servidor, especificar quais as vistas que devem de ser geradas/apresentadas ao utilizador e obter os modelos de dados necessários.

De seguida é feita uma ilustração do ciclo de vida de uma aplicação desenvolvida em ASP.NET MVC 5.

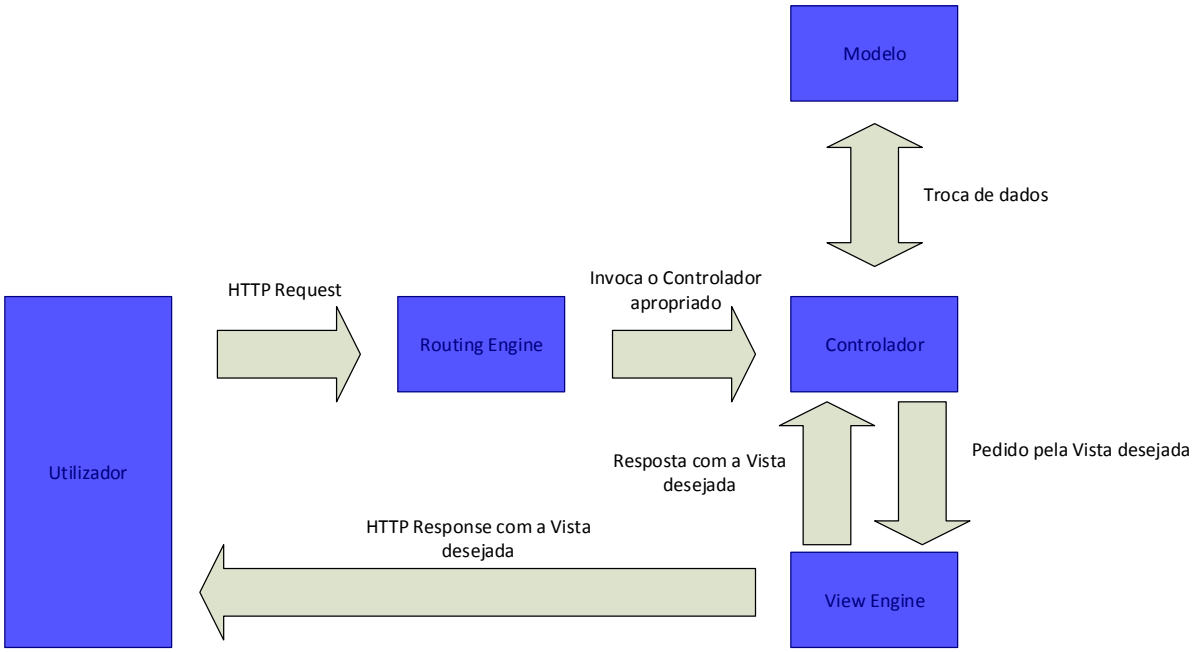


Figura 21 - Ciclo de vida de uma aplicação ASP.NET MVC5

Como é possível ver pela figura (adaptada de [69]), o ciclo de vida de uma aplicação desenvolvida em ASP.NET MVC 5 é algo complexo. O utilizador começa por fazer um pedido, um HTTP *Request*, que chega ao *Routing Engine*. Este, baseado na configuração que foi realizada vai invocar o controlador apropriado, no caso do Web Vessel Tracker é o LoginController. Uma vez invocado o controlador correto, este comunica com o modelo de modo a obter os dados necessários (caso seja necessário), para serem depois apresentados ao utilizador. O controlador, tendo a informação necessária, tem de fazer um pedido ao ViewEngine correto para que este devolva a vista desejada, no entanto, podem existir vários ViewEngines, pelo que é necessário procurar aquele que renderiza a vista desejada. Quando o ViewEngine correto é encontrado, a vista é devolvida para o controlador que, por sua vez, envia a envia através de um HTTP *response*. É nesta altura que o utilizador tem acesso ao pedido que fez.

5.3.2 O Modelo

Para o modelo, foi decidido fazer uma divisão entre as entidades do modelo de domínio, e as classes que tratam da gestão da lógica de negócio. A razão, para esta divisão passou por querer ter as entidades num estado "puro", isto é, sem qualquer lógica de negócio.

Agora, é feita uma descrição de como o modelo de domínio está organizado, seguida de uma descrição das classes que tratam da lógica de negócio.

5.3.2.1 As entidades

De forma a satisfazer os requisitos, foram criadas e usadas as seguintes entidades para o modelo de domínio (os requisitos estão disponíveis no Anexo I):

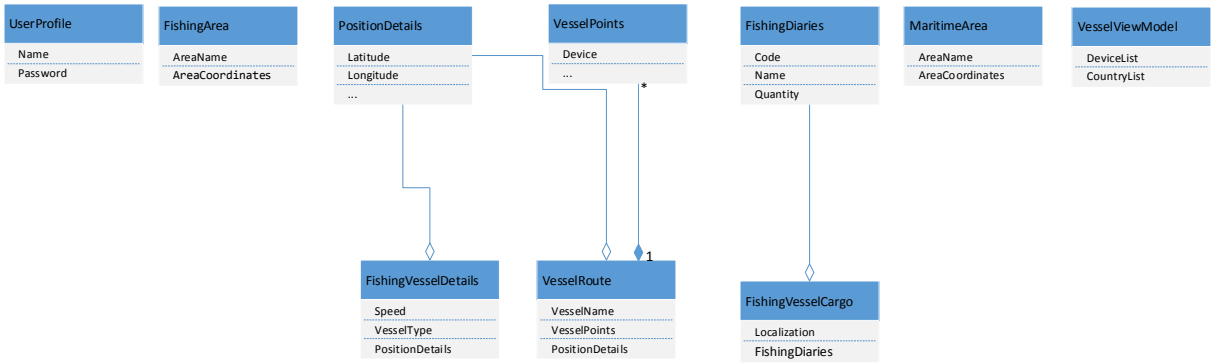


Figura 22 - Modelo de domínio da aplicação

- FishingVesselDetails, representa uma embarcação na página principal e na página de pesquisa para mostrar ao utilizador informação referente às embarcações que estão no mapa como, por exemplo, localização, última atualização, entre outros. Grande parte da informação que é mostrada ao utilizador é armazenada em instâncias desta entidade, uma para cada embarcação. A utilização de uma lista desta entidade nas duas páginas mencionadas, justifica-se pelo facto de o mesmo conteúdo ser apresentado em ambas sem a existência de atrasos. A entidade *FishingVesselDetails* só contém informação de embarcações que tenham tido alguma atividade nas últimas 48 horas;

- VesselRoute, representa a informação relativa a uma rota de uma embarcação num determindo intervalo de tempo, tal como o estado do(s) dispositivo(s) utilizado(s) pela embarcação, o nome da embarcação, a sua localização, entre outros. Esta entidade consiste, essencialmente, numa lista de pontos registados durante a rota de uma embarcação;

- FishingDiaries, representa os diários de pesca. A informação que esta entidade armazena consiste na quantidade de cada espécie diferente de peixe que foi pescado por uma embarcação, o nome comum desse peixe e o respetivo código FAO (código de identificação de uma espécie de peixe). Os diários de pesca estão disponíveis apenas para embarcações que tenham o dispositivo MONICAP. Por este motivo, a aplicação não mostra informação sobre o peixe capturado por embarcações que

utilizem outro tipo de dispositivo;

- FishingVesselCargo, representa todo o peixe que está a bordo de uma embarcação num determinado momento. A informação armazenada consiste numa lista de FishingDiaries, assim como a latitude, longitude e a data em que o peixe foi pescado/registado;

- VesselPoints, representa a informação relativa aos pontos da rota de uma embarcação num determinado intervalo de tempo. Contém também informação sobre o dispositivo que está a bordo de uma embarcação, assim como a latitude, longitude, estado do(s) dispositivo(s) a bordo e a data de registo, entre outras. Esta entidade já existia aquando do início do estágio e era utilizada noutros projetos da XSealence;

- VesselPoint, representa a informação relativa a um ponto de uma determinada rota. Esta entidade contém informação sobre o dispositivo que está abordo de uma embarcação, assim como, a latitude, a longitude, o estado do(s) dispositivo(s), a data de registo de ponto, entre outros. O VesselPoint já existia aquando do início do estágio e é utilizado noutros projetos desenvolvidos pela XSealence;

- UserProfile, representa a informação relativa a um utilizador autenticado da aplicação. Esta entidade contém o nome do utilizador, o apelido e a sua palavra passe;

- MaritimeArea, representa a informação necessária para fazer a representação de áreas de sistema e públicas. As áreas de sistema são um tipo de área que representa um conjunto de áreas "reais". Estas áreas existem para todos os utilizadores da aplicação e são imutáveis. As áreas públicas são parecidas com as áreas de sistema com a particularidade de algumas só aparecerem conforme o nível de privilégio do utilizador. Mais abaixo faremos referência a outros tipos de áreas existentes no sistema. No lado do servidor não existe essa distinção, dado que o cliente não tem forma de pedir áreas de sistema e áreas públicas em simultâneo. O cliente, quando faz um pedido, indica qual o tipo de área desejado para que o modelo obtenha essa informação da camada intermédia. A distinção destas áreas só é necessária no lado do cliente;

- PositionDetails, representa a informação relativa à posição de uma embarcação, como é o caso das coordenadas, velocidade e rumo.

- VesselViewModel, representa a informação relativa aos filtros presentes na página de

pesquisar.

5.3.2.2 A lógica de negócio

A imagem abaixo ilustra as classes que tratam da lógica de negócio bem como dos controladores que com elas interagem, seguida da descrição de cada classe.

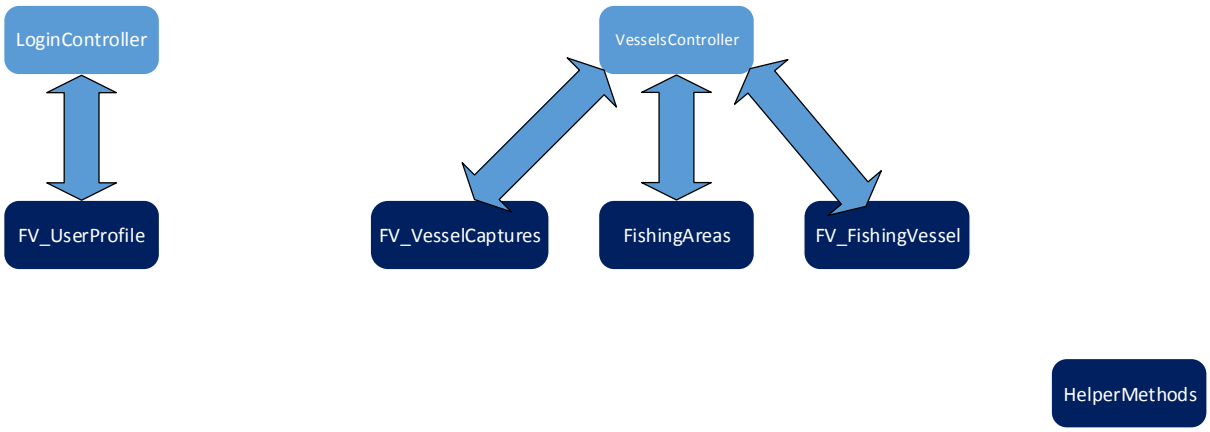


Figura 23 - Modelo de classes que tratam da lógica de negócio

- FV_UserProfile, trata da gestão do *login* e autenticação de um utilizador;
- FV_VesselCaptures, trata da gestão das capturas de uma determinada embarcação. É nesta classe que são obtidas as capturas de uma embarcação, a carga atual de uma embarcação o FishingVesselCargo e os diários de pesca relativos a uma rota realizada;
- FishingAreas, trata da gestão das áreas da aplicação, isto é MaritimeArea;
- FV_FishingVessel, trata da gestão das embarcações e das suas rotas. É esta classe que obtém a foto da embarcação, caso exista, para ser utilizada na página principal e onde é feita a gestão de FishingVesselsDetails e dos VesselRoutes;
- HelperMethods, é onde se encontram os métodos auxiliares para o tratamento da informação necessária entre a informação que é recebida do servidor e a que o utilizador vai receber. Todas as classes à excepção do FishingAreas e FV_UserProfile utilizam métodos desta classe.

5.3.3 As Vistas

É feita agora uma descrição das vistas que a aplicação tem, assim como, uma breve descrição

das interações que o utilizador pode ter com estas.

A LoginPage, como o nome indica, é a página onde o *login* e autenticação de um utilizador é realizado. Esta é uma página simples, com um formulário onde o utilizador insere o nome e palavra passe. Caso algum desses *inputs* esteja errado, é apresentada uma mensagem de erro. Caso contrário, o utilizador é redirecionado para a página principal. Esta vista, o LoginPage, é apresentada por omissão ao utilizador, dado que foi essa a configuração feita no *RouteConfig* (ficheiro responsável pelo *routing* da aplicação).

A MainPage é a página que permite ao utilizador visualizar informação relativa às embarcações. É nesta página que o utilizador vê a carga que está a bordo das embarcações, que define e visualiza as rotas das embarcações, onde desenha as suas próprias áreas, filtra as embarcações que estão presentes no mapa, onde troca as *map tiles* do mapa, entre outras funcionalidades. Para esta vista foi criada alguma lógica em JavaScript, Razor e Knockout que facilita algumas das interações que o utilizador tem com a aplicação. Um destes exemplos é uma funcionalidade que permite fazer a pesquisa de embarcações no mapa.

A SearchPage é a página que possibilita fazer pesquisas. Esta página apresenta uma tabela devidamente paginada com todas as embarcações que estão renderizadas na página principal. A paginação da tabela é conseguida através do PagedList.Mvc [70], que é um pacote NuGet, que facilita a paginação de uma tabela. A razão pela qual este foi escolhido foi por ser fácil e simples de utilizar e por ser o que tinha a maior comunidade (na altura) dentro de pacotes para facilitar a paginação. Nesta página o utilizador pode utilizar os filtros disponíveis tais como, país, tipo de embarcação e tipo de dispositivo utilizado, para facilitar a procura de uma embarcação. Estes filtros, implementados com recurso ao HTML *Helper* DropDownList [71], permitem ainda ao utilizador escolher vários elementos com a utilização do MultiSelect [72] para a sua pesquisa. Os HTML Helpers são funções simples que permitem especificar qual o tipo de HTML a ser gerado numa vista. O utilizador, tem ainda a opção de ver uma determinada embarcação na página principal (existe um redirecionamento para a mesma quando a opção é selecionada). Esta ação, redireciona o utilizador para a página principal e centra o mapa na embarcação selecionando-a e gerando a *popup*. Isto é conseguido através da utilização de Razor, OpenLayers, JavaScript e jQuery. É também utilizado o `Html.ActionLink` [71], que permite fazer o redirecionamento da página passando o ID da embarcação para a página seguinte. O JavaScript e o jQuery são necessários para detetar o redirecionamento e obter as coordenadas através do ID que "capturaram". As coordenadas são obtidas

pesquisando na VesselLayer por uma *feature* com esse ID. Quando a *feature* é encontrada, as coordenadas são depois enviadas para uma função que, para centrar o mapa na embarcação, seleciona essa embarcação e gera uma *popup* com detalhes sobre a mesma.

5.3.4 Os Controladores

Esta secção descreve os controladores que foram desenvolvidos para o projeto. A imagem seguinte serve para ilustrar quais as páginas que comunicam com os controladores.

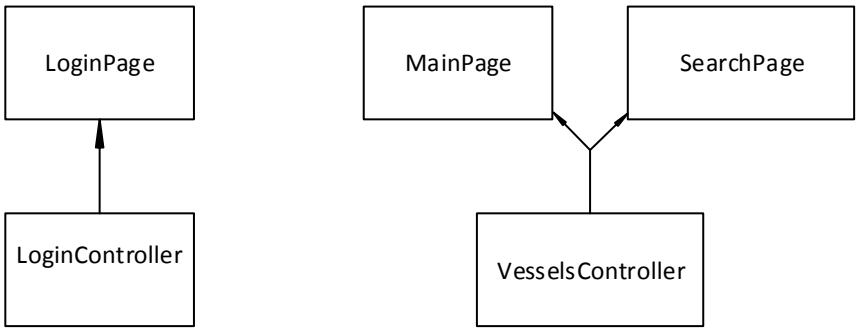


Figura 24 - Controladores e VIstas

Na figura acima é possível visualizar os dois controladores utilizados, o LoginController e o VesselsController. O LoginController é responsável por servir de intermediário no processo de autentição de um utilizador, e de devolver as vistas apropriadas consoante as credenciais do utilizador. O processo de autenticação é descrito com mais detalhe na secção, dedicada à autenticação e segurança. O VesselsController é o controlador que gere a página principal e a pesquisa. Para a página principal, trata de pedir ao modelo a informação desejada pelo utilizador. No caso da página de pesquisa, é ele que trata da lógica da apresentação, isto é da forma como a tabela reage aos filtros que o utilizador define com recurso aos elementos que foram explicados na Secção 5.3.3.

É com recurso a `System.Web.HttpContext.Current.Cache` [73] que este controlador consegue representar a mesma informação relativa às embarcações (FishingVesselDetails) e ter as páginas principal e de pesquisa em sintonia.

Esta abordagem foi seguida em deterimento da utilização do Session State [74], por poder tornar os pedidos não determiníticos [75]. Outra alternativa seria a utilização do TempData [76]. No entanto, esta abordagem é indicada para situações em existe apenas um redirecionamento, o que não é o caso da página do pesquisar. Sempre que é realizado um refrescamento das embarcações presentes no página principal, essa alteração é refletida também na página de pesquisa. A informação, apesar de ser a mesma em ambas as vistas, é

apresentada de forma diferente consoante a página onde o utilizador se encontra. Na página principal o utilizador consegue ver o local no mapa onde a embarcação está, enquanto que na página de pesquisa é apresentada uma tabela com várias informações sobre as embarcações.

5.4 O Funcionamento do OpenLayers na aplicação

Esta secção serve para introduzir e explicar termos que são utilizados com frequência na secção das funcionalidades. Para grande parte da informação que o utilizador vê na aplicação é necessário recorrer a uma combinação de JavaScript e OpenLayers para que a informação desejada seja renderizada no mapa.

De seguida é feita uma descrição da *pipeline* que existe para a renderização das embarcações no mapa.

O cliente, com recurso ao Jquery, faz um pedido Ajax [77] a um método (GetFishingVessels) do controlador FishingVesselsController para obter as embarcações que tenham tido alguma atividade nas últimas 48 horas, isto é, uma lista de *FishingVesselDetails*. Este pedido é recebido e a lista é depois obtida através de um pedido ao modelo. A resposta é devolvida através de JSON, uma vez que o método do controlador é um System.Web.MVC JsonResult [78]. A resposta, por ser em JSON torna-se mais leve do que usar XML (EXtensible Markup Language) e facilita a representação de objetos complexos, daí ter sido feita a escolha de utilizar métodos do tipo JsonResult. O cliente quando recebe a resposta "interpreta" o JSON e gera as denominadas *features*. Uma *feature* refere-se a uma instância da classe `ol.Feature` [79], que permite fazer a representação de um objeto no mapa, neste caso, uma embarcação.

Antes de uma *feature* estar pronta para poder ser representada, é necessário configurar a sua localização no mapa. Para tal, são utilizadas as coordenadas da embarcação para criar uma `ol.geom.Geometry`[80], que cria uma geometria vetorial (uma figura/objeto) para fazer a representação da embarcação. Uma *feature* pode ser configurada para conter diversas propriedades como, por exemplo, a velocidade, o rumo, as últimas coordenadas, entre outros.

De seguida é feita uma ilustração de como as propriedades de uma *feature* são configuradas.

```
var iconFeature = new ol.Feature({
    image: vesselDetailsInfo[i].Image,
    name: vesselDetailsInfo[i].Name,
    type: vesselDetailsInfo[i].VesselType,
    device: vesselDetailsInfo[i].Device,
});
```

Figura 25 - Configuração e algumas propriedades de uma *feature*

A imagem que está ilustrada acima encontra-se dentro de um ciclo, tendo algumas das propriedades sido omitidas para que a imagem não ocupasse demasiado espaço. Neste caso, o ciclo está a gerar uma *feature* nova por cada elemento presente em vesselDetailsInfo, que é uma lista resultante da leitura do JSON que o cliente recebeu.

A feature é configurada para ter uma representação específica, sendo tido em conta a velocidade e o tipo de dispositivo a bordo da embarcação. A distinção do tipo de dispositivo é conseguida através de um esquema de cores diferente aplicadas ao ícone, neste caso, azul para o MONICAP e vermelho para o AIS. Na eventualidade de a embarcação ter AIS e MONICAP é atribuída a cor azul, já que as embarcações com dispositivo MONICAP disponibilizam o peixe pescado a bordo, algo que tem interesse para o utilizador. No caso da velocidade, se a embarcação em questão se está a deslocar abaixo de 0.4 milhas náuticas é tratada como se tivesse a 0. Nesta situação, é utilizado um ícone diferente para simbolizar que a embarcação em questão está parada. Estes ícones continuam a utilizar o esquema de cores para o dispositivo que está a bordo. Uma vez definido qual o ícone a ser utilizado, é realizada mais uma verificação: se a embarcação tiver uma velocidade superior a 0.4 milhas náuticas, então o ícone é rodado consoante o rumo que a embarcação está a seguir. Quando o ícone para representar a embarcação estiver definido já é possível adicionar essa *feature* ao mapa. De modo a tornar possível essa representação, é necessário adicionar a *feature* que acabou de ser criada a uma `ol.source.Vector`[81]. Esta, por sua vez, serve de fonte (*source*) para a `ol.layer.Vector`[82], que é a camada ou *layer* onde as *features* serão apresentadas ao utilizador. É com recurso às funções da `ol.source.Vector` que é possível manipular as *features* a serem removidas ou adicionadas do/ao mapa, através de funções como *removeFeature*() e *addFeature*(). Na `ol.layer.Vector` é possível definir a que resolução/*zoom* do mapa a *layer* deve de ser renderizada. Um exemplo seria a camada das embarcações a qual só tem interesse em ser apresentada quando a resolução estiver a dois mil pixeis ou quando o *zoom* estiver a oito.

A imagem abaixo ilustra de uma forma simples como o JSON que é recebido pelo cliente é

transformado numa *feature* e depois como esta é apresentada no mapa.

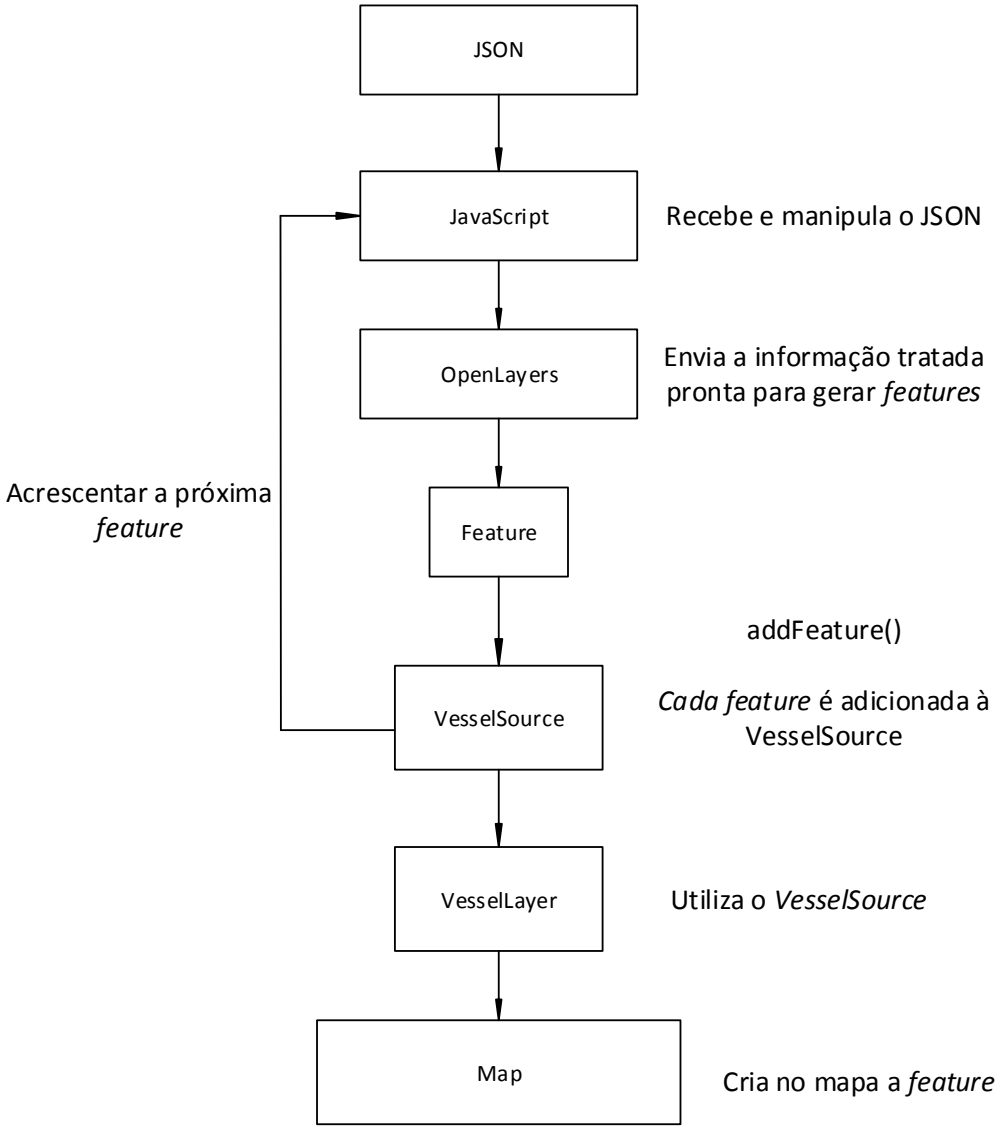


Figura 26 - Passos na criação de uma feature no mapa

Do JSON, que é uma lista de FishingVesselDetails, vão ser lidas cada uma das embarcações presentes. Os detalhes de cada embarcação são utilizados para gerar uma *feature* que representa essas propriedades, como é o caso da velocidade, rumo, tipo de embarcação e dispositivo a bordo, que são utilizadas para definir qual o ícone a ser gerado para a embarcação em questão. Cada *feature* criada é adicionada ao VesselSource, através da função do OpenLayers *addFeature()*, sendo depois renderizada no ecrã. Como o VesselSource é referenciado pelo VesselLayer, qualquer alteração ao *VesselSource* é refletida no VesselLayer.

A secção seguinte faz a descrição de algumas das funcionalidades desenvolvidas.

5.5 As funcionalidades

Para uma melhor compreensão sobre a forma como cada funcionalidade foi desenvolvida, esta secção aborda o desenvolvimento das funcionalidades mais relevantes, descrevendo para cada uma, todo o seu *pipeline*.

Para grande parte das funcionalidades desenvolvidas foi necessário guardar alguma dessa informação em *cache*, como por exemplo os detalhes das embarcações. Para tal foram analisadas algumas formas de implementar uma *cache* no lado do cliente, nomeadamente *cookies* [83], Local Storage e Session Storage [84]. O Local e Session Storage são a opção mais segura, permitem o armazenamento de dados de grande dimensão sem afetar o desempenho da aplicação. A diferença entre o Local e Session Storage reside na forma como os dados armazenados são apagados. No Local Storage não existe uma data de validade e a informação continua guardada mesmo que o *browser* seja fechado. O Session Storage mantém a informação até o utilizador fechar a janela/*tab*. Com isto em mente foi decidido utilizar o Session uma vez que não havia a necessidade de guardar informação durante muito tempo/longos períodos. Foram também utilizadas as chamadas *callback functions* [85] em conjunto com alguns pedidos Ajax, nomeadamente no caso do diários de pesca. A razão passou pela necessidade de poder fazer o pedido Ajax e receber a informação antes de editar o HTML referente aos diários de pesca.

Salientamos que, quando se refere um pedido do cliente ao servidor está-se a referir um pedido Ajax, a menos que seja explicitada outra forma.

5.5.1 Pesquisar por nomes de embarcações no mapa

Esta é uma funcionalidade que foi desenvolvida para a página principal da aplicação. Esta funcionalidade permite a um utilizador procurar por uma determinada embarcação que esteja no mapa. Esta funcionalidade tem um mecanismo de *auto-complete,* feito em Jquery, que facilita a procura de uma embarcação ao utilizador. Quando a embarcação desejada é selecionada, o mapa é centrado no ponto dessa embarcação, é feito um *zoom-in* e a embarcação passa para o modo “selecionada” para facilitar a sua visualização.

Em seguida é feita uma ilustração da ação de pesquisar.

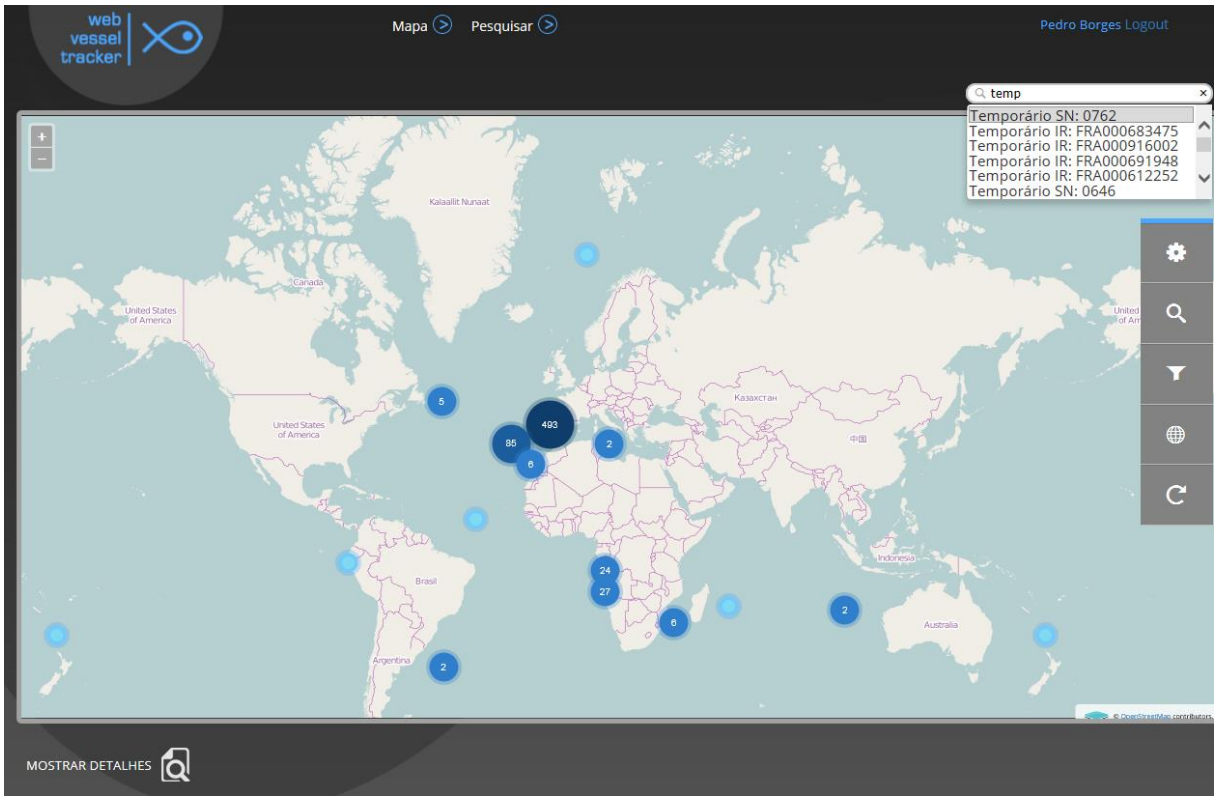


Figura 27 - Pesquisar o nome de uma embarcação (antes)

O preenchimento da tabela com os nomes das embarcações é realizado aquando da criação de uma *feature* representativa de uma embarcação. Desta forma, e em conjunto com o *auto-refresh* existente, é possível ter a lista de nomes sempre atualizada.

A imagem seguinte ilustra o resultado final depois de um nome ser escolhido. Como é possível constatar, foi feito um *zoom-in*, que é possível ver pelo maior detalhe dos *map tiles* e por um maior afastamento das embarcações. É possível ainda ver que o mapa foi centrado na embarcação selecionada e que esta está no modo "selecionada".

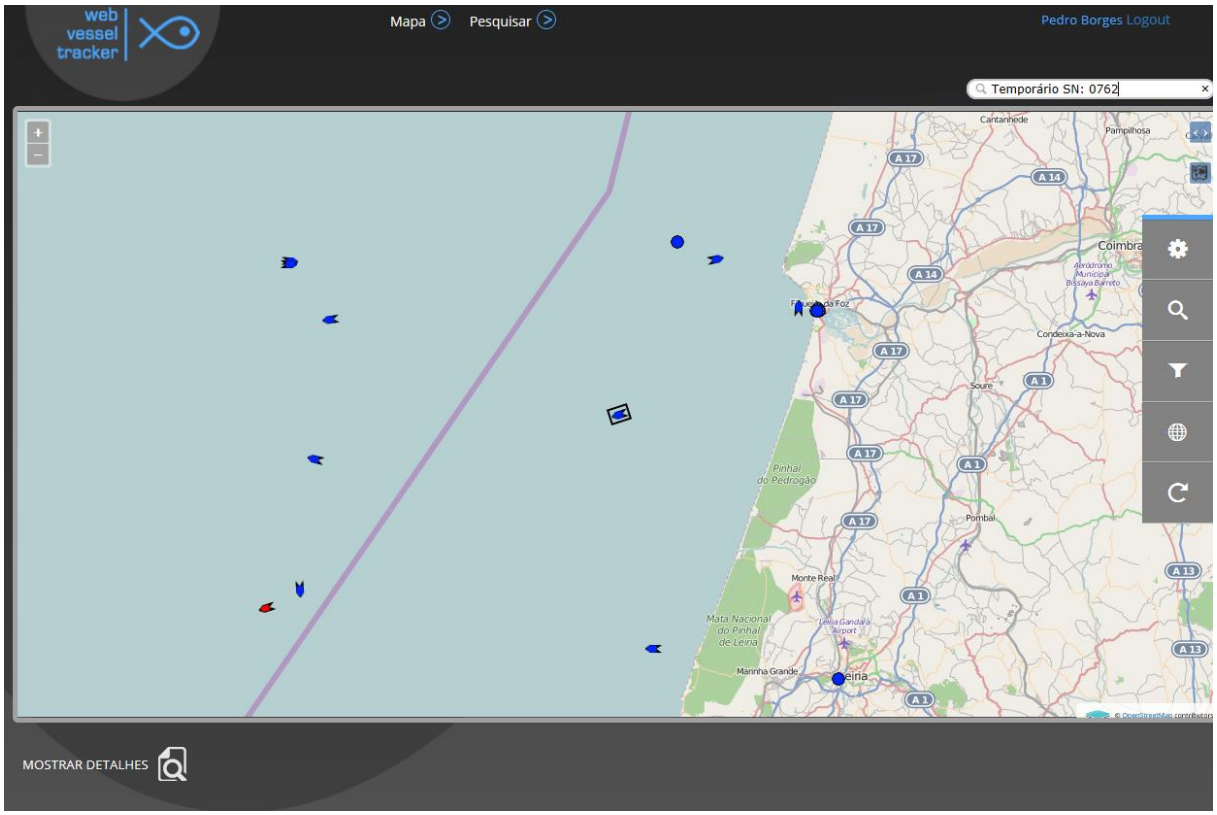


Figura 28 - Pesquisar o nome de uma embarcação (depois)

5.5.2 A troca de *map tiles*

Na página principal, é possível ao utilizador trocar de *map tiles* em tempo real sem perder o contexto do que estava a fazer, isto é, as embarcações, rotas, entre outros aspetos, não se perdem aquando da troca de *map tiles*. Foi utilizada a biblioteca LayerSwitcher [86] para fazer a implementação desta funcionalidade. Os *map tiles* que são suportados neste momento são os Bing Maps, MapQuest, OSM, Watercolor Stamen [87] e mapas desenvolvidos pela XSealence, com a possibilidade de acrescentar mapas adicionais. Os *map tiles* da Google foram propositadamente postos de parte, uma vez que a sua integração com o OpenLayers, apesar de possível, não é recomendada [88] dado que é algo complexo e torna a aplicação instável.

5.5.3 As *tooltips*

As *tooltips* das embarcações mostram ao utilizador o nome da embarcação. Uma *tooltip* é criada quando o utilizador tem o cursor por cima de uma *feature*. Quando este evento ocorre, é feita uma "captura" à *feature* semelhante ao que ocorre na criação da *popup* mas a única propriedade que interessa neste caso é o nome, uma vez que que a *tooltip* não precisa de mais informação. As *tooltips* foram desenvolvidas com base na referência [89], que é um exemplo

do OpenLayers.

5.5.4 O agrupamento das embarcações

De forma a reduzir o "ruído visual" que seria causado por todas as embarcações serem apresentadas logo à partida, foi decidido ter uma camada inicial com *clustering* (agrupamento). Esta camada é apresentada quando a resolução do mapa está acima de um determinado valor, como por exemplo, 1000 pixeis. O *clustering* é conseguido utilizando `ol.source.Cluster` [90], que é uma classe que, para além da criação de uma *layer* de *clusters*, permite também fazer a configuração da resolução mínima que este pode ter antes de "desaparecer" e de como é que os *clusters* devem estar organizados e ser apresentados. A apresentação refere-se ao estilo que é utilizado para o *cluster*, ou seja, quantos mais elementos existirem no *cluster*, maior e mais escura será a sua representação. Os elementos referem-se ao número de embarcações existentes num *map tile* num determinado momento. Foram criadas várias gamas de valores para que um *cluster* deva de ser apresentado. Esses valores foram: uma embarcação num *map tile* é um *cluster* e depois a cada intervalo de vinte e cinco embarcações é escolhido um *cluster* diferente, até chegar às 100 embarcações por *map tile*, nessa situação é sempre escolhido o mesmo *cluster*, para fazer a representação quando estão mais de 100 embarcações. A imagem seguinte mostra a *cluster layer* que é apresentada aquando de uma autenticação bem sucedida.

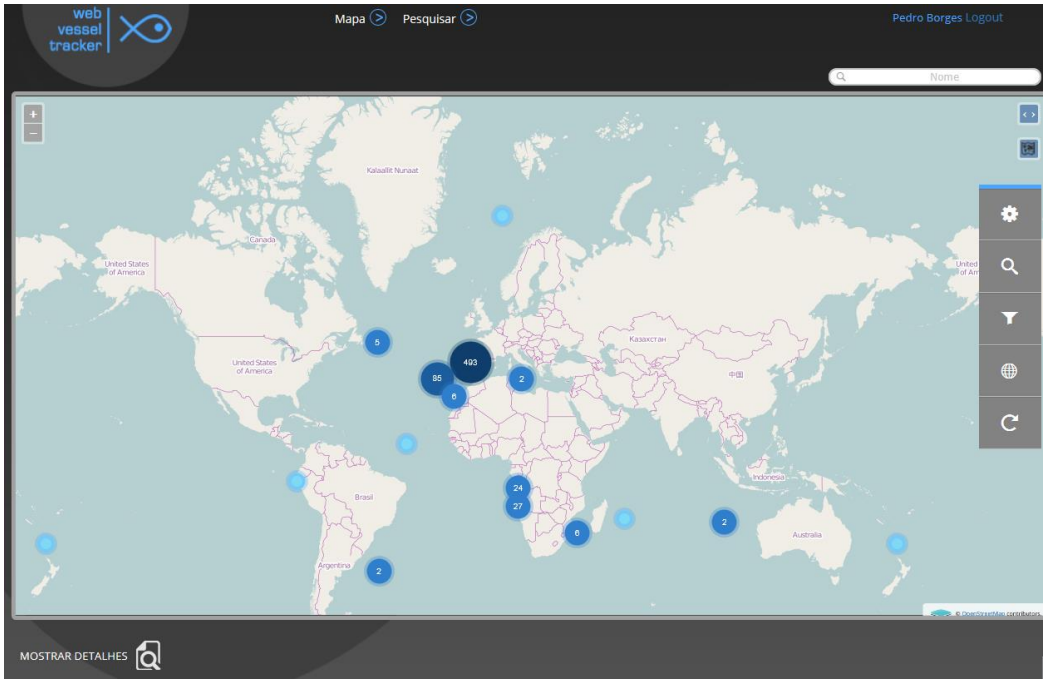


Figura 29 - A *cluster layer*

A razão pela qual é criado um *cluster* quando só existe uma embarcação num *map tile* passa

pela separação de *layers* que foi realizada. Neste caso, existem dois tipos de *layers*, a das embarcações e a dos *clusters*. O que acontece é que ambas as *layers* existem em simultâneo, mas só uma é que está "ativa" num determinado momento, dependendo da resolução atual do mapa. No capítulo referente ao trabalho futuro existe uma secção dedicada ao trabalho a realizar relativamente aos *clusters* que têm só uma embarcação.

A imagem seguinte mostra como as *layer* das embarcações e das *clusters* interagem uma com a outra.

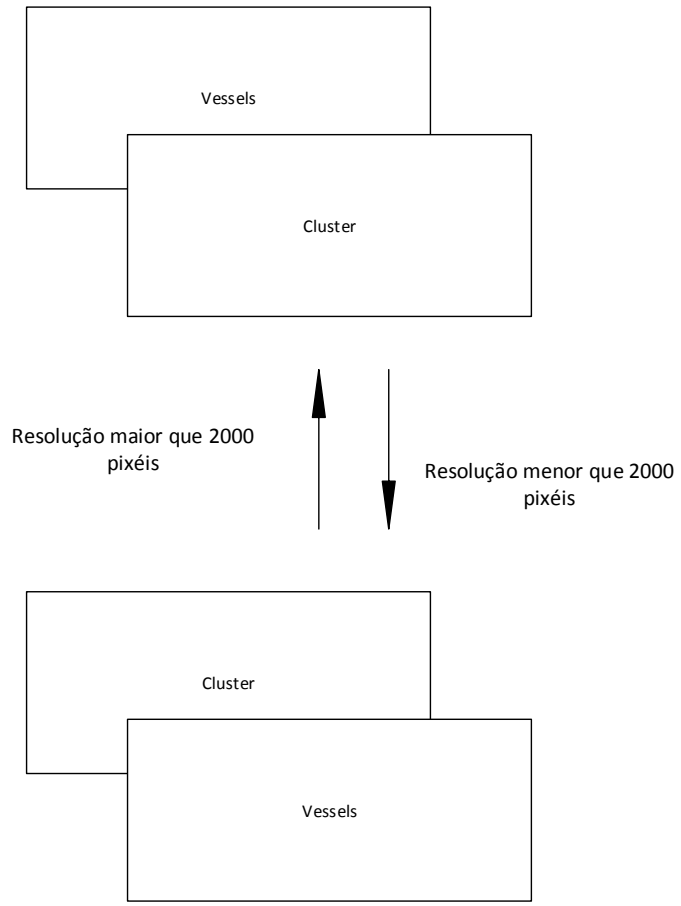


Figura 30 - As camadas das embarcações e do agrupamento das embarcações

Na figura apresentada é possível observar que a layer Cluster "passa para a frente" da layer Vessels quando a resolução é maior que 2000 pixéis, verificando-se o contrário quando a resolução é menor que 2000 pixéis.

5.5.5 Os detalhes das embarcações

As *popups* das embarcações são responsáveis por mostrar ao utilizador detalhes sobre as embarcações no mapa. As *popups* podem implicar entre zero a dois pedidos ao servidor dependendo se, a embarcação tem o MONICAP ou se a embarcação tem uma imagem. A

imagem da embarcação (caso exista) só é obtida/pedida quando o utilizador seleciona essa embarcação, isto é, quando a *popup* é gerada. Quando tal acontece, é feito um pedido para obter a imagem da embarcação que está guardada na base de dados. Na eventualidade de a embarcação não ter uma imagem, é utilizado um ícone de uma embarcação genérica. Uma vez obtida a imagem, ela é convertida para uma *string* de base 64 para depois voltar a ser convertida para JSON a fim de ser devolvida para o lado do cliente. O cliente quando recebe a resposta do pedido converte o JSON de volta numa *string* de base 64 para conseguir fazer a representação dessa imagem na *popup*. O outro pedido que é feito, é relativo ao peixe que uma embarcação tem a bordo nesse momento. Lembramos que existe a restrição de apenas ser possível obter informação (tipo e quantidade) sobre o peixe pescado. A *popup* é criada com recurso a um `ol.Overlay` [91], que é um elemento criado por cima do mapa. Uma vez criado o `ol.Overlay`, é necessário injetar o HTML, que tem a estrutura da *popup* a ser gerada na vista atual. A injeção só é realizada quando o utilizador "clica" numa embarcação. Quando esse evento é detetado, são utilizadas algumas das funcionalidades do OpenLayers para obter a *feature* que foi clicada, neste caso a embarcação. Quando a *feature* é "capturada", algumas das suas propriedades como, por exemplo, a velocidade rumo, nome, localização, imagem, tipo de embarcação, entre outras, são adicionadas ao HTML que irá ser injetado na vista. As coordenadas onde a *popup* é apresentada (tendo como referência o seu canto superior direito) são calculadas com base na posição da embarcação no ecrã. Na eventualidade de o utilizador selecionar uma embarcação diferente, e caso haja uma *popup* ativa, essa é fechada antes de a nova ser gerada.

Esta implementação foi beaseada num exemplo disponibilizado pelo OpenLayers na referência [92]. A imagem seguinte ilustra o estado em que a *popup* se encontrava na altura da entrega deste documento.

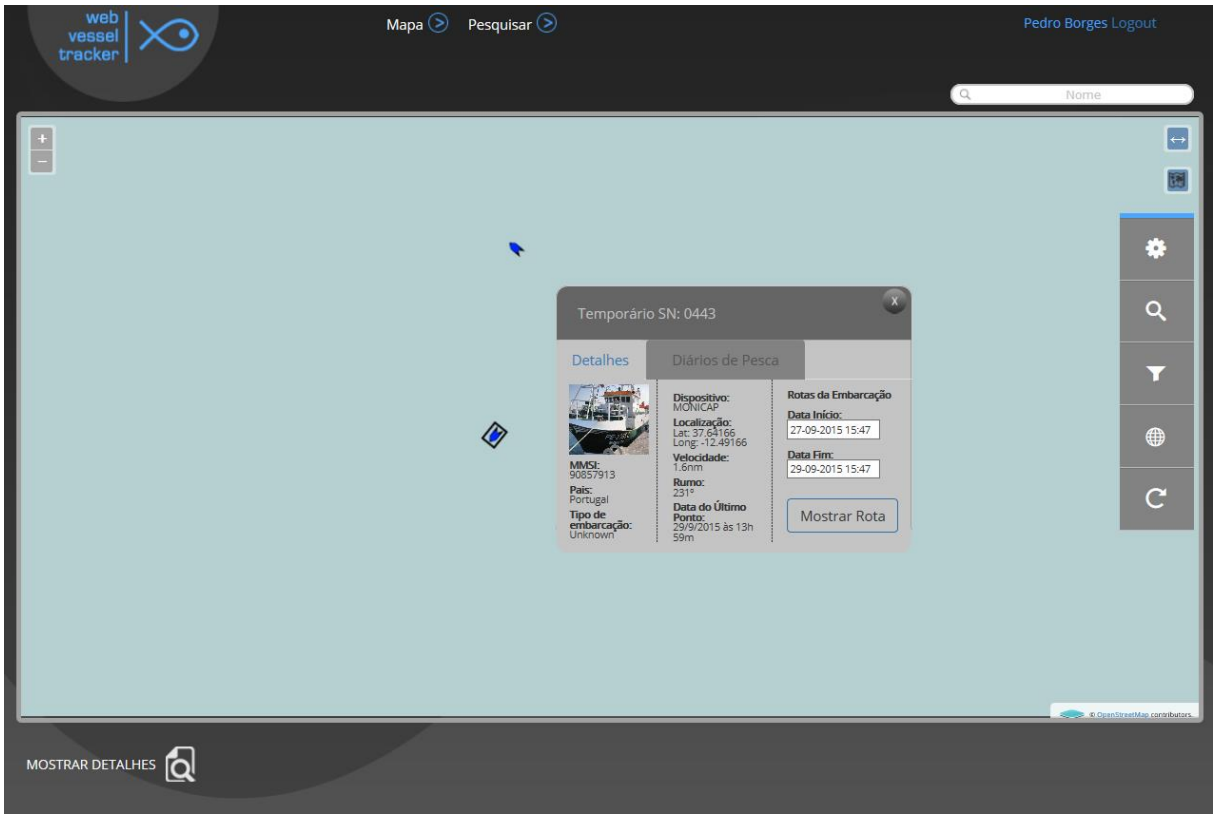


Figura 31 - Uma *popup* de uma embarcação no Web Vessel Tracker

5.5.6 As rotas das embarcações

A rota de uma embarcação é caracterizada/definida pelos pontos que foram registados pelo aparelho que esta tem a bordo (MONICAP ou AIS) num determinado intervalo de tempo. Por exemplo, para poder visualizar uma rota da embarcação A, é necessário selecionar essa embarcação no mapa e escolher duas datas, a data do início de rota e a de fim, sendo que esta tem de ser sempre posterior àquela. Essa informação (a data de início, a data de fim e o ID da embarcação) é então enviada para o FV_FishingVessel que tem o método necessário para obter a rota que essa embarcação fez no intervalo de tempo definido. A informação é depois convertida para JSON e enviada para o cliente, onde essa informação é tratada para ser depois apresentada ao utilizador. O tratamento que é feito pelo cliente passa por várias fases. Numa primeira fase, é criada a rota que é apresentada no mapa com recurso ao OpenLayers, que recebe os pontos da rota e gera no mapa uma linha (`ol.geom.LineString`[93]) que é composta por esses mesmos pontos. O último ponto dessa linha (que é o ponto mais recente) tem o ícone correspondente ao tipo de embarcação em questão. Na segunda fase, é necessário guardar a informação relativa a cada ponto e da rota em si, para o utilizador poder visualizar essa mesma rota com mais detalhe. Para tal, é utilizado o padrão MVVM (no lado do cliente)

com o Knockout de modo a conseguir criar uma tabela com a informação desejada que é responsiva e transparente para o utilizador. De seguida é feita uma breve explicação do padrão MVVM.

O MVVM (*Model View ViewModel*) é um padrão que deriva do MVC, que facilita a separação do desenvolvimento da parte gráfica, a GUI (*graphical user interface*), do desenvolvimento da lógica de negócio.

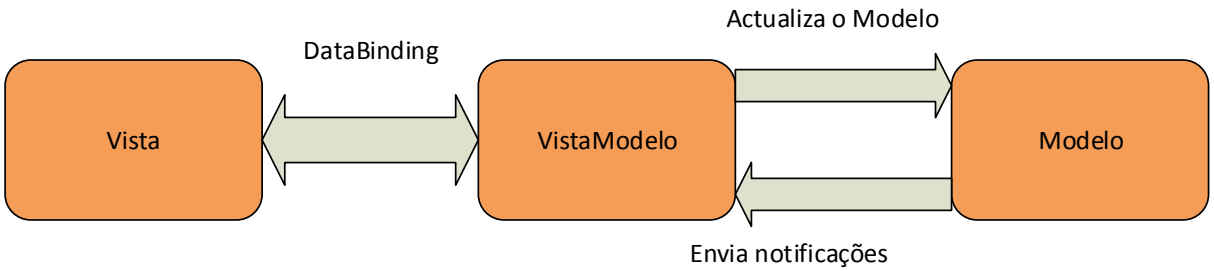


Figura 32 - Padrão MVVM

Como é possível observar na figura acima (adaptada de [94]), este padrão é consituído por três camadas, tal como o padrão MVC.

- A Vista neste padrão funciona de forma igual à do padrão MVC. Ou seja, é a camada que é responsável pela interface que o utilizador está a ver/utilizar (a GUI);

- A VistaModelo expõe os dados e os comandos que a Vista necessita. Esta camada funciona como intermediária entre a vista e o modelo, ou seja, qualquer alteração que haja na vista reflete-se no modelo e vice-versa;

- O Modelo representa o modelo de domínio que incluí um modelo de dados assim como, lógica de negócio e de validação.

Para mais informação sobre este padrão consultar a referência [94].

As entidades que são apresentadas de seguida representam a informação das rotas para o utilizador.

- chosenVesselsPoints, esta entidade representa o nome da embarcação e uma lista com os detalhes dos pontos de uma determinada rota definida;

- chosenVesselsCargo, esta entidade representa as capturas que foram feitas nos pontos

de uma determinada rota que foi definida; esta entidade só é aplicável a embarcações com o dispositivo MONICAP.

Com a informação referida acima em mente, é agora explicado o processo que ocorre quando o JSON que é recebido do servidor é processado pelo Knockout. O JSON começa por ser *binded* para dois *observableArrays*, (um para os pontos da rota e outro para a rota) que são elementos do Knockout. Um *observableArray* [95] é um elemento que é utilizado quando se quer detetar alguma mudança/interação, como por exemplo, mudar o elemento selecionado dentro de uma *drop-down list* e responder com a informação desejada. Uma vez finalizado o *bind*, a *View* é notificada para atualizar a *drop-down list* de rotas com detalhes da rota mais recente que foi adicionada. É com recurso a esta *drop-down list* que o utilizador consegue visualizar as várias rotas que tem no mapa. Quando o utilizador escolhe uma rota, o *Knockout* deteta essa ação e gera uma tabela com todos os detalhes obtidos do servidor. Na eventualidade de o utilizador eliminar uma rota, a lista é então atulizada e, se a tabela estiver renderizada, esta é removida. É com recurso a estas ações que o *Knockout* possibilita ao utilizador ter acesso à informação de uma rota de forma quase instantânea.

5.5.7 O refrescamento das embarcações

De forma a garantir que o mapa apresenta as embarcações nas suas posições mais recentes, é realizado um pedido ao servidor a cada 60 segundos em *background* para obter as novas posições das embarcações, resultando no correspondente refrescamento das embarcações no mapa. No cliente é feita uma camparação entre a lista de embarcações antiga e a nova que resultou do último pedido ao servidor. Primeiramente, é verificado se a embarcação "antiga" está na lista mais recente, isto é, se essa embarcação teve alguma atividade nas últimas 48 horas; se tal não tiver acontecido, essa embarcação é removida da *layer* das embarcações e, consequentemente, do mapa. Após a análise das embarcações antigas é então utilizada a lista mais recente.

A imagem seguinte ilustra a forma como a substituição com a lista mais recente é feita na VesselLayer.

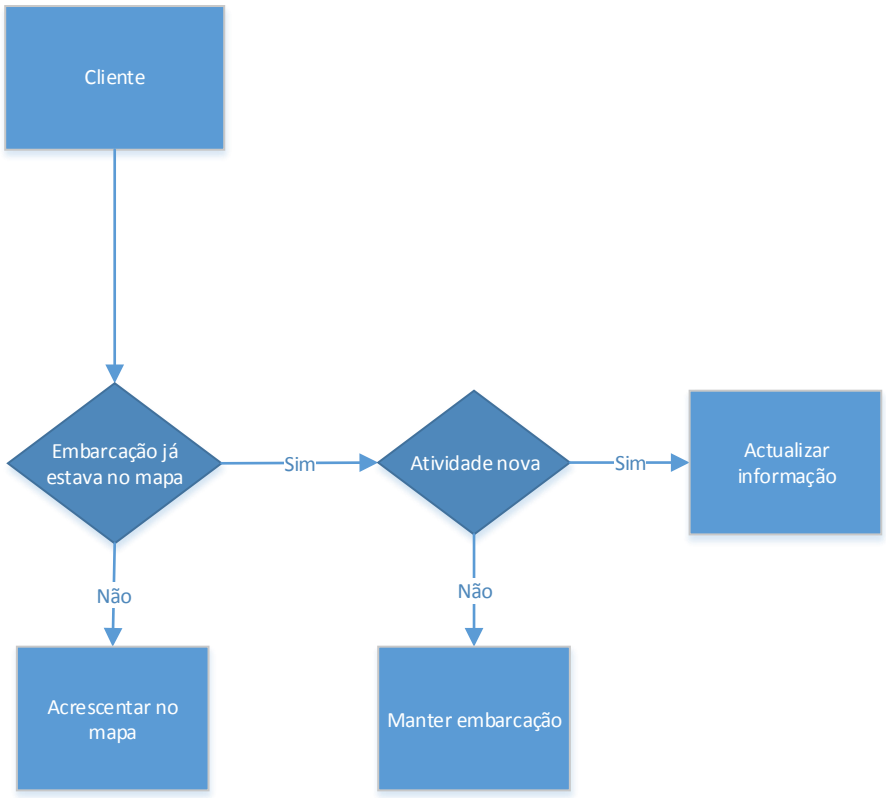


Figura 33 - Refrescamento das embarcações

Como é possível ver pela imagem acima, começa-se por verificar se uma embarcação está ou não no mapa; se não estiver, é criada a *feature* com as características correspondentes e essa *feature* é adicionada à VesselLayer. Se a embarcação estiver no mapa, e verificado se houve alguma atividade, isto é se ela fez alguma captura ou se mudou de posição, se sim, essa informação é atualizada, caso contrário, não se faz nada. Este processo é repetido até todas as embarcações da lista nova serem "analisadas".

5.5.8 A visualização das áreas

Existem três tipos diferentes de áreas: As áreas de sistema, as áreas públicas e as áreas pessoais.

As áreas do sistema, que são áreas predefinidas no sistema e que não sofrem alterações. Por esse motivo, foi decidido guardá-las em *cache* assim que o primeiro pedido de uma sessão para as visualizar estiver concluído. As áreas de sistema, quando são apresentadas ao utilizador, encontram-se agrupadas conforme a região onde estão. Desta forma, é mais fácil para um utilizador que não tenha muita experiência encontrar a área que deseja visualizar. Por exemplo, se o utilizador quiser ver a área AM_Portugal_120M (nome fictício), sabe que ela está algures dentro do grupo Portugal.

As áreas públicas são áreas que foram criadas por utilizadores com um determinado nível de privilégio. Estas são depois visíveis para todos os outros utilizadores da aplicação. As áreas públicas são muito parecidas com as áreas pessoais, uma vez que só o autor é que pode modificar/eliminar mas, para efeitos de visualização, em vez de estarem restritas a uma só pessoa todos os restantes utilizadores têm acesso a elas.

As áreas pessoais são criadas durante uma sessão de um utilizador devidamente autenticado. Neste caso, o utilizador tem permissão para as alterar como, por exemplo, mudar o nome ou apagar essa mesma área. As áreas pessoais estão mais destinadas aos utilizadores que tipicamente visualizam uma região/área no mapa com alguma frequência como, por exemplo, a região entre o porto de Lisboa e o de Faro. Uma vez que estas áreas são criadas pelo utilizador, são apresentadas pela sua ordem de criação.

Uma vez que existem três tipos diferentes de áreas no sistema, decidiu-se dar a cada uma delas tons de cor diferentes para facilitar a distinção das mesmas. Assim, as áreas de sistema têm uma cor mais escura, as áreas públicas uma intermédia e as pessoais uma cor mais clara. Estas definições estão nas *layers* a que cada tipo de área está associada. Outra decisão que foi tomada durante o desenvolvimento do módulo das áreas foi a de incluir o *zoom* (referimo-nos ao nível de *zoom* que o mapa tinha aquando da criação da área) como um dos parâmetros da área (só se aplica nas áreas pessoais). Quando o utilizador seleciona a opção de ver uma determinada área, a aplicação centra o mapa no ponto central da área e aplica o *zoom* de criação. Desta forma, é possível reduzir o "ruído visual" da aplicação uma vez que essas áreas só são geradas quando o utilizador se encontrar num *zoom* inferior em uma unidade do valor inicial. Por exemplo, se a área ABC é criada com o *zoom* a 5, esta só fica visível quando o *zoom* estiver a 4. Com isto, o número de áreas que estão a ser representadas em simultâneo diminui e o utilizador tem sempre a área "ativa" que está renderizada para *zooms* mais pequenos, caso pretenda maior detalhe. De seguida é feita uma ilustração da hierarquia destas três áreas na aplicação.

Figura 34 - A hierarquia das áreas

A forma como as áreas de sistema e públicas são desenhadas é diferente da forma como são desenhadas as áreas pessoais. Nas áreas pessoais o utilizador assiste em 1ª pessoa à criação da área, uma vez que é ele que a desenha e que lhe dá o nome. Quando o utilizador atribui o nome à área, a geometria dessa área é depois utilizada para criar a *feature* que a vai representar. Esta é a descrição do processo de criação de uma área pessoal do ponto de vista das ações do utilizador. Do ponto de vista técnico, é utilizado o ol.interaction.Draw [96], que é uma interação do OpenLayers para permitir o desenho de polígonos num mapa. Uma vez desenhado o polígono, é chamado um evento do ol.interaction.Draw, o drawend, assim que o utilizador indicar que terminou de desenhar a nova área. É nesta fase que é obtida a geometria desse polígono para que seja aplicada a uma *feature*. Uma vez tendo a *feature*, procede-se à sua associação com o tipo de estilo que esta tem, para depois ser representada no mapa através da camada (*layer*) correspondente ao tipo de área em questão.

No caso das áreas de sistema e públicas é realizado um pedido ao servidor de modo a obter os pontos que constituem essa área. Com esses pontos é desenhada uma área "fantasma" em *background*. Quando a área já se encontra desenhada, a sua geometria é obtida de modo a ser utilizada pela *feature* para fazer a representação dessa mesma área. De uma forma mais detalhada, o que acontece é que nas áreas de sistema e públicas não é utilizado o `ol.interaction.Draw` de forma direta, isto é, pelo utilizador. Para este tipo de áreas o `ol.interaction.Draw` é utilizado para criar o polígono em *background* utilizando as coordenadas que recebeu do servidor. As coordenadas têm de ser convertidas pelo OpenLayers de modo a conseguir criar o polígono, o qual não é visível para o utilizador. Uma

vez criado o polígono, é obtida a sua geometria para poder ser utilizada pela *feature* que irá fazer a representação dessa área. A esta *feature* é-lhe dado o estilo e associada a *layer* correspondente para ser depois renderizada.

Durante o desenvolvimento optou-se por passar o polígono criado para uma *feature*. Nas áreas de sistema e públicas o que muda é a fase inicial. Para estas é feito um pedido ao servidor o qual devolve uma lista com as coordenadas em formato decimal dessas áreas. De forma, a conseguir aplicar a mesma lógica de ter uma *feature* para fazer a representação dessas áreas, é criado um polígono em *background* antes da criação da *feature*. Uma vez criado o polígono, este é associado à *feature,* assim como, o nome dessa área. Por fim, essa informação fica associada à *layer* das áreas de sistema e é guardada em *cache,* dado que o pedido para as áreas de sistema só é feito uma vez porque estas não sofrem alterações.

Cada tipo de área pertence a uma *layer* diferente. A razão pela qual se faz esta divisão passa por facilitar a eventual atribuição de atributos novos às áreas no futuro, assim como, tornando também mais fácil ao utilizador a distinção entre os diferentes tipos de área.

5.5.9 Os diários de pesca

Os diários de pesca são os registos do peixe capturado por uma embarcação que tenha o MONICAP. Na aplicação existem duas formas de um utilizador poder visualizar o peixe que uma embarcação tenha a bordo. A primeira passa por selecionar uma embarcação no mapa e depois selecionar a opção para ver os diários (o peixe que está a bordo). A informação que é apresentada nesta situação é o peixe que está atualmente a bordo dessa embarcação, ou seja, caso tenha havido alguma descarga, esta não é apresentada. Esta informação é apresentada ao utilizador com recurso a um pedido Ajax para obter o peixe que a embarcação tem nesse instante. Uma vez feito o pedido, o sevidor devolve ao cliente a informação pedida em formato JSON. O cliente, quando recebe a informação sobre o peixe a bordo da embarcação, faz o devido tratamento, isto é, procede à sua formatação e ao cálculo da carga total que a embarcação tem, apresentando essa informação ao utilizador. A segunda só é apresentada quando o utilizador cria uma rota. Ao visualizar informação sobre essa rota e, caso seja uma embarcação MONICAP, existe a opção de poder ver o peixe que estava a bordo num determinado ponto da rota. Com isso é possível poder acompanhar a evolução da carga que uma embarcação teve durante uma determinada rota. A forma como esta informação é enviada para o utilizador passa mais uma vez por um pedido Ajax e uma resposta em JSON; o que muda é a fase do tratamento. O JSON que é devolvido ao cliente contém todo o peixe que

foi pescado durante a rota de uma embarcação num determinado período. Desta forma, é possível mostrar o peixe que a embarcação tinha nesse período de tempo de uma forma muito mais rápida . O tratamento que é feito para este efeito passa por utilizar o Knockout e criar um *observable array* com o peixe que estava a bordo em cada ponto. Assim, sempre que o utilizador seleciona um ponto da rota, o Knockout deteta essa ação e mostra ao utilizador o peixe do ponto "novo" (caso exista; caso contrário, é apresentada uma mensagem a avisar que não foi feita nehuma captura nesse ponto). A figura seguinte ilustra o diário de pesca de uma embarcação, isto é o peixe que ela tem a bordo nesse momento.

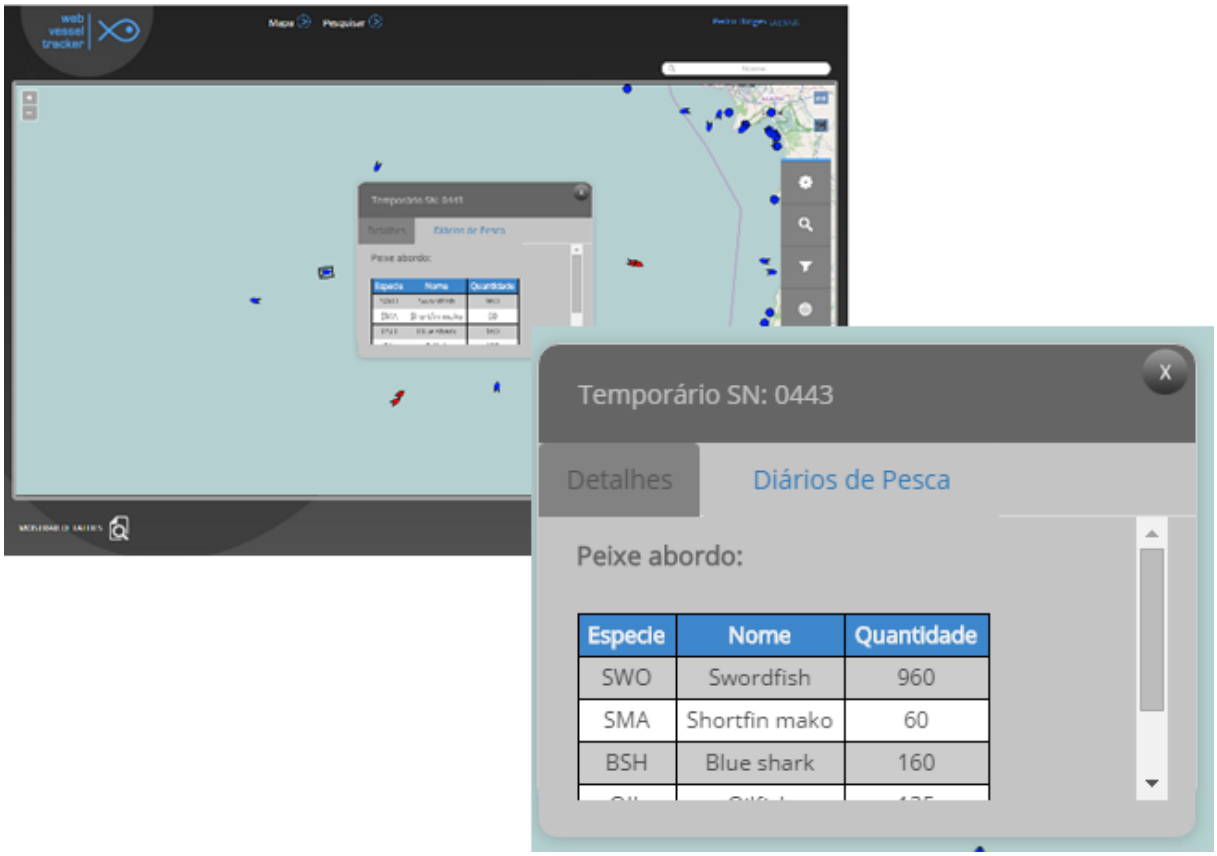


Figura 35 - Diários de pesca

Como é possível observar pela figura, o utilizador consegue visualizar qual o peixe a bordo capturado, a quantidade dele e o peso total do peixe a bordo se fizer *scroll down*.

5.6 Autenticação/Segurança

A segurança é sempre um fator importante numa aplicação, que lide com informação sensível ou importante. No entanto, a necessidade de tornar uma aplicação segura pode ser incrementada dependendo de como esta é acedida, isto é: se é uma aplicação que vai correr numa rede interna ou se é uma aplicação a que qualquer pessoa pode ter acesso. Uma

aplicação que esteja "aberta" ao público, terá que ter um cuidado extra em comparação com uma que esteja numa rede interna. Tal acontece devido à facilidade de acesso que existe quando algo é aberto ao público.

O Web Vessel Tracker faz parte do primeiro grupo, já que apenas utilizadores que estejam ligados à rede do sistema é que podem aceder à aplicação. A autenticação de um utilizador é feita com recurso a um projeto desenvolvido pela XSealence que devolve o primeiro e último nome do utilizador caso a autenticação seja válida. As bases de dados utilizadas para fazer a consulta ao utilizador encontram-se também num projeto à parte. De forma a tornar esta autenticação mais segura, são usados os atributos do ASP.NET HttpPost [97] e ValidateAntiForgeryToken [98], quando se trata de *inputs*.

Quando a autenticação é bem sucedida, o modelo cria um UserProfile que é utilizado nas restantes páginas para mostrar ao utilizador que este está autenticado na aplicação. É criada depois uma variável de sessão com o nome do utilizador. Esta variável tem um *timeout* definido para dez horas. É com esta variável que um utilizador tem acesso às páginas da aplicação, isto é, enquanto essa variável continuar com um valor não nulo o utilizador tem acesso a todas as páginas. Esta "autorização" é feita com recurso a um `AuthorizeAttribute` [99] customizado. Este verifica se o utilizador está autenticado isto é, se a variável de sessão está ou não nula e, se não estiver, este tem acesso às páginas que tenham o atributo `VesselTrackerAuthorization`.

Por omissão, o ASP.NET guarda as páginas do cliente em cache, o que é algo que pode ser problemático quando se trata de informação sensível. De forma a resolver este problema, foram analisadas duas soluções: prevenir que as páginas tenham *cache* [100], levando a uma perda de *performance*, uma vez que seriam necessários mais pedidos ao servidor. A segunda abordagem consiste em detetar quando é que o utilizador termina a sua sessão e, nessa altura, redirecioná-lo para a página de autenticação enquanto um utilizador válido não for autenticado. Deste modo, se alguém com intenções maliciosas aceder ao computador e recuar uma página, não tem acesso à informação dessa página mas é redirecionado para a página de autenticação. Por essas razões, esta acabou por ser a abordagem que foi seguida.

Para além dos atributos mencionados, existe outro que é utilizado nas aplicações *web* que é o AllowAnonymous. Este atributo permite que qualquer pessoa, estando autenticada ou não, possa aceder a uma página, como por exemplo, página de contactos ou "sobre nós". Este atributo é utilizado para a *View* do *login*, permitindo que qualquer pessoa possa aceder a essa

página, uma vez que esta não tem informação sensível.

Não foi implementado nenhum mecanismo de "lembrar-se de mim?", dado que é uma funcionalidade que o próprio *browser* consegue fazer.

Testes realizados

Uma aplicação necessita de ser alvo de testes para remover erros que escaparam durante o desenvolvimento ou então corrigir problemas de usabilidade. Os testes que foram realizados, serviram para procurar problemas de usabilidade, e testar o estado atual da robustez da aplicação. Esta não será a última fase de testes do Web Vessel Tracker, dado que é uma aplicação ainda em desenvolvimento, onde serão feitos mais testes, como, por exemplo, de carga e de desempenho até chegar à "versão de lançamento". Neste capítulo é feita uma descrição dos testes que foram realizados à aplicação e as conclusões retiradas, assim como, uma breve descrição dos problemas que foram encontrados.

6.1 Testes de validação de HTML

Os testes de validação de HTML, como o nome indica, são testes que têm como objetivo validar o HTML de uma página, isto é, se ele cumpre os *standards* da web impostos pelo *W3 Consortium* [101]. O validador que foi utilizado encontra-se na seguinte referência [102].

Os resultados obtidos com as páginas do Web Vessel Tracker nesse validador são apresentados no Anexo IV e de seguida é feita uma pequena descrição de como correram os testes de validação para cada página:

Página de login:

Esta validação correu bem com 0 avisos e 0 erros.

Página principal:

No primeiro teste de validação foram detetados vários erros e avisos, desde Ids repetidos, caracteres inválidos dentro das *tags*, e CSS definidos dentro do próprio HTML, que é uma má prática de desenvolvimento *web*. Esses IDs repetidos foram passados a classes e os CSS que estavam definidos dentro das *tags* passaram para os ficheiros de CSS apropriados.

Uma vez corrigidos estes erros foi possível ter uma validação positiva, isto é 0 erros e avisos.

Página de pesquisa:

Na página de pesquisa a validação correu também bem à primeira com 0 erros e avisos. Uma das principais razões para este resultado é o facto de esta página apresentar a mesma página de *layout* que a página principal apresenta. Dado que a maior parte dos erros da página principal estavam nessa página, a página de pesquisa já não os apresentou.

Uma reflexão que pode ser feita em relação à validação do HTML de toda a aplicação é que um dos motivos pela qual resultaram poucos erros deveu-se à utilização do validador de *markup* do Visual Studio 2013. Este tem um modo de funcionamento igual a uma ferramenta de deteção de erros ortográficos, para mais informação sobre este validador consultar a referência [103].

6.2 Testes de macaco

Os testes de macaco (Monkey Testing), são testes automatizados onde os campos são preenchidos com *inputs* aleatórios e os botões premidos de forma aleatória. O objetivo deste tipo de testes é analizar a robustez de uma aplicação. Para a realização deste teste foi utilizada a biblioteca Gremlins js [104]. O Gremlins é uma biblioteca escrita em JavaScript que permite a realização de testes de macaco. O objetivo deste teste era encontrar alguma coisa que fosse capaz de causar erros de JavaScript ou então de provocar um erro irrecuperável na aplicação.

Não foram detetados erros irrecuperáveis na aplicação, no entanto, foram encontrados erros de JavaScript, tanto na página principal como na de pesquisar, que serão corrigidos. A grande maioria dos erros encontrados consistiram na não verificação de *inputs* inválidos, nomeadamente de coordenadas e nomes. Desta forma, será necessário fazer testes mais controlados com as coordenadas da rota ou de uma embarcação, dado que podem existir algumas coordenadas inválidas. No que toca aos erros de *inputs*, é necessário verificar as funções que deram erro e melhorar a forma como elas fazem a validação desses mesmos *inputs*.

6.3 Testes de usabilidade

Os testes de usabilidade são muito importantes pois ajudam a encontrar os erros de usabilidade que uma aplicação tem. Neste caso, pode ser algo tão simples como alterar a localização de um botão para ser necessário só um clique para que a ação associada a este seja

executada. Durante o desenvolvimento da aplicação várias alterações foram sendo feitas de modo a corrigir problemas de usabilidade que iam sendo identificados. Na altura da entrega do documento foi realizado um teste de usabilidade mais detalhado, o guia encontra-se disponível no Anexo V. Este teste foi feito a 10 pessoas. Dessas 10, 3 acompanharam o desenvolvimento da aplicação Web Vessel Tracker durante bastante tempo. Após a realização destes testes, foram pedidas várias sugestões sobre o que mudariam na aplicação para esta ficar mais do seu agrado.

De seguida, são apresentadas algumas das sugestões realizadas para cada página e respetivas funcionalidades presentes. As razões das sugestões dadas são também incluídas a não ser que a mesma seja auto explicativa.

Para a página do login:

- Ter uma menssagem de "*Loading*" quando a aplicação passa da página de *Login* para a página principal.

Para a página principal:

- Utilizar uma cor que destaque melhor uma embarcação que se encontra selecionada;
- Passar o nome do utilizador para o canto superior direito;
- Mudar a *label* "mapa" para página principal; o utilizador pode pensar que "mapa" é uma página diferente quando este se encontra na página principal, quando não é o caso;
- Ter uma mensagem de "*loading"* na criação de uma rota; existem rotas mais longas que outras; deste modo, o utilizador é notificado de que uma rota está a ser gerada;
- Centrar a embarcação quando esta é selecionada; por vezes pode ser díficil distinguir a embarcação selecionada das restantes, principalmente quando esta se encontra rodeada de muitas embarcações;
- Na criação de uma rota, calcular o ponto médio do ponto inicial ao ponto final e centrar o mapa nesse ponto;
- Ter uma opção para sinalizar os portos; esta seria uma opção que por omissão estaria

ligada;

- Destacar mais os títulos das funcionalidades na *sidebar*;
- Ter a opção “mostrar área" a centrar o mapa num ponto intermédio;
- Distinguir a rota de cada embarcação/mostrar só uma rota; na versão atual todas as rotas das embarcações têm a mesma cor;
- Caso a rota esteja criada e uma embarcação altera a sua posição, atualizar também a rota;
- Reduzir o espaço que a *popup* das áreas tem, existe espaço desnecessário;
- Reduzir e simplificar a informação que é mostrada sobre a rota de uma embarcação, neste momento existe demasiada informação;
- Melhorar a forma como o *mouse over* em cima de embarcações é detetado, na implementação atual está pouco preciso. Acontece principalmente nas embarcações paradas;

Para a página de pesquisa:

- Tornar a cor do fundo da página do pesquisar mais clara;
- Tornar as *labels* dos botões de pesquisa mais auto-explicativas;
- Dar relevo aos botões;
- Incluir uma imagem em vez de texto na funcionalidade “ver embarcação no mapa”. O texto atual pode ficar com uma *hint* sobre a imagem;
- Ordenar a tabela das embarcações pela última actualização;
- Apresentar informação sobre qual o critério pela qual a tabela se encontra ordenada no momento, isto é, qual o campo e se esse campo está por ordem decrescente ou crescente.
- Ter a pesquisa avançada mais auto-explicativa;

- Refazer a forma como os filtros estão a ser mostrados ao utilizador. Da forma atual causam confusão e dão a ideia que todos têm de ser escolhidos;

- Ter o chosen [105], que é um plugin JS que torna as selectboxes mais user friendly, com *scroll* da forma a não haver necessidade de lista tabular se alterar, nem de utilizar o *scroll* da página;

- Na eventualidade de ser necessário, apresentar uma *popup* com os detalhes relevantes da embarcação selecionada;

- Ter um "minimapa" na página, se for necessário, só com a embarcação, de modo a não perder o contexto de pesquisa;

- Ter a tabela do pesquisar igual à tabela dos mostrar detalhes, para haver mais consistência na aplicação.

Após esta análise foram descobertos alguns erros/falhas que na altura de desenvolvimento não foram detetados, como foi o caso da falta de consistência de cores ou algo que permita fazer a diferenciação de rotas. Esses erros serão corrigidos assim que possível. Com esta análise foi também possível recolher boas sugestões, indo de pequenas mudanças a funcionalidades complementares que serão apresentadas à XSealence num futuro próximo para serem discutidas.

Conclusão e Trabalho Futuro

Com a realização deste estágio os objetivos inicialmente propostos foram atingidos, isto é uma aplicação *web* que permite ao utilizador monitorizar a atividade piscatória de uma determinada embarcação. Com esta aplicação é criado mais um mecanismo para regular e controlar as pescas, de modo a conseguir desenvolver e manter uma pesca sustentável para todos.

Durante o estágio realizado, foi desenvolvida uma aplicação com uma arquitetura escalável, que permite responder às necessidades dos utilizadores da aplicação. O WebVessel Tracker permite mostrar ao utilizador a informação desejada, a atividade piscatória, de uma forma transparente e rápida.

A aplicação foi implementada com recurso a tecnologias da Microsoft, nomeadamente a plataforma de desenvolvimento ASP.NET MVC 5, e a tecnologias *cliente.side* como é o caso do Knockout, OpenLayers 3, JavaScript e jQuery. Esta aplicação, conta ainda com a utilização de vários projetos que foram desenvolvidos pela XSealence.

O WebVessel Tracker foi desenvolvido num contexto empresarial onde, apesar da entrega deste documento, o ciclo de desenvolvimento da aplicação vai continuar, seja para melhorar algumas funcionalidades, seja para acrescentar funcionalidades extra. Algumas dessas melhorias e acrescentos são apresentados e descritos de seguida, como trabalho futuro.

- Incluir o SignalR [106]; Esta é uma biblioteca para o ASP.NET que simplifica o processo de acrescentar funcionalidades em tempo real, disponibilizando uma API que permite a criação de RPCs (Remote Procedure Calls) servidor-cliente, que invocam funções JavaScript no *browser* do cliente através de código presente do lado do servidor. A utilização desta biblioteca na aplicação deverá, nomeadamente, simplificar a forma como o refrescamento das embarcações e a procura por novas áreas é realizada;

- Incluir na aplicação suporte para multi-língua; A aplicação só tem suporte para português. Esta é uma funcionalidade que tem como objetivo fazer a aplicação chegar a mais pessoas de diferentes nacionalidades. Esta implementação vai detetar qual a linguagem do utilizador através da sua região (país) com recuro a propriedades como Culture [107] e UICulture [108]. Caso o utilizador pretenda mudar a lingua por algum motivo, ser-lhe-á também permitido fazê-lo Para mais informação sobre como se espera que esta funcionalidade venha a ser implementada, consultar a referência [109];

- Acrescentar mais ícones de tipos de embarcações que existem no sistema AIS. Estas alterações consistem em adicionar ícones novos e alterar o esquema de cores de modo a distinguir embarcações com o dispositivo AIS de embarcações com o MONICAP. Deste modo, será possível mostrar ao utilizador, de uma forma mais amigável, os vários tipos de embarcações que estão numa determinada zona;

- Associar aos *clusters* que representam só uma embarcação, uma função que aplique o *zoom* necessário de modo a passar da *layer* dos *clusters* para a *layer* das embarcações. Esta alteração permitirá melhorar a experiência de utilização, dado que implicará menos um passo para o utlizador mudar as *layers;*

- Criação de uma secção de estatísticas onde o utilizador terá acesso a várias estatísticas sobre as embarcações. As estatísticas apresentadas poderão ser , por exemplo, o peixe mais pescado por uma embarcação, o carregamento médio de uma embarcação, a(s) área(s) que aparecem mais vezes nas rotas de pesca, entre outras;

- Criação de rotas de velocidade que permitam ao utilizador acompanhar a evolução da velocidade de uma embarcação ao longo da sua rota;

- Adaptar a aplicação para incluir o protocolo HTTPS (Hypertext Transfer Protocol Secure). A inclusão deste protocolo no Web Vessel Tracker irá aumentar a sua segurança;

- Implementar um mecanismo de *timeout*, isto é, caso o utilizador fique com a página aberta durante um determinado período de tempo sem nenhuma atividade (30 minutos, por exemplo), a aplicação faz *logout*. Desta forma, é acrescentado mais um obstáculo no acesso à informação, no caso de existir uma conta que esteja ainda autenticada sem que o utilizador esteja presente.

- Incluir *bundles* de modo a reduzir o tempo de carregamento de uma página;

- Realizar testes de carga e de desempenho; o objetivo deste tipo de testes é verificar se a aplicação tem algum *bottleneck*, verificar como é que a aplicação se comporta quando são feitos muitos pedidos em simultâneo e analisar a robustez da aplicação. Para uma eventual realização destes testes foi analisada a ferramenta de *load* e *performance testing* do Visual Studio 2013 [110]. Para estes testes será também necessário definir quais os critérios mais importantes a serem analisados;

- Os testes realizados com o Gremlin.Js provaram que existem funções JavaScript que precisam de ser adaptadas/corrigidas de modo a informar ao utilizador quando este faz um determinado *input* corretos;

- Incluir algumas das sugestões que foram dadas durante os testes de usabilidade.

Como balanço final, o estágio desenvolvido na XSealence foi uma experiência muito enriquecedora, que serviu para adquirir aptidões novas, relativamente a tecnologias e ao desenvolvimento de aplicações GIS. Desta experiência foram também criadas novas relações, tanto a nível pessoal como profissional.

Bibliografia

[1] "Overfishing -- Pristine Seas -- National Geographic." [Online]. Available: http://ocean.nationalgeographic.com/ocean/explore/pristine-seas/critical-issues-overfishing/. [Accessed: 23-Sep-2015].

[2] S. Murawski, "Definitions of overfishing from an ecosystem perspective," *ICES J. Mar. Sci.*, vol. 57, no. 3, pp. 649–658, Jun. 2000.

[3] "Overfishing - A global environmental problem, threat to our oceans and disaster." [Online]. Available: http://overfishing.org/pages/what_is_overfishing.php. [Accessed: 23-Sep-2015].

[4] O. of S. Fisheries, "Ending Overfishing Through Annual Catch Limits :: Office of Sustainable Fisheries."

[5] "Pesca Profissional — ICNF." .

[6] "Enquadramento Legal — ICNF." .

[7] "MONICAP tm - ADI_Monicap.pdf." [Online]. Available: http://www.adi.pt/docs/ADI_Monicap.pdf. [Accessed: 25-Sep-2015].

[8] "Autoridade Marítima Nacional - Homepage." [Online]. Available: http://www.amn.pt/Paginas/Homepage.aspx. [Accessed: 28-Sep-2015].

[9] "Food and Agriculture Organization of the United Nations." [Online]. Available: http://www.fao.org/home/en/. [Accessed: 23-Sep-2015].

[10] "An introduction to monitoring, control and surveillance systems for capture fisheries." [Online]. Available: http://www.fao.org/docrep/003/V4250E/V4250E03.htm. [Accessed: 23-Sep-2015].

[11] "FAO Fisheries & Aquaculture - Components - VMS." [Online]. Available: http://www.fao.org/fishery/topic/18101/en. [Accessed: 23-Sep-2015].

[12] "FAO Fisheries & Aquaculture - Communications systems - VMS." [Online]. Available: http://www.fao.org/fishery/topic/18103/en. [Accessed: 23-Sep-2015].

[13] "FAO Fisheries & Aquaculture - Shipboard equipment - VMS." [Online]. Available: http://www.fao.org/fishery/topic/18102/en. [Accessed: 23-Sep-2015].

[14] H. Gerritsen and C. Lordan, "Integrating vessel monitoring systems (VMS) data with daily catch data from logbooks to explore the spatial distribution of catch and effort at high resolution," *ICES J. Mar. Sci.*, vol. 68, no. 1, pp. 245–252, Sep. 2010.

[15] “Slide 1 - nmea collision avoidance through ais.pdf.” [Online]. Available: http://www.nmea.org/Assets/nmea collision avoidance through ais.pdf. [Accessed: 23-Sep-2015].

[16] “RIS :: AIS - Frequently Asked Questions.” [Online]. Available: http://ris.vlaanderen.be/html_en/AIS/ais_vragen.html. [Accessed: 23-Sep-2015].

[17] “Vulnerabilities Discovered in Global Vessel Tracking Systems.” [Online]. Available: http://blog.trendmicro.com/trendlabs-security-intelligence/vulnerabilities-discovered-in-global-vessel-tracking-systems/. [Accessed: 23-Sep-2015].

[18] “The North Atlantic Format.” [Online]. Available: http://www.naf-format.org/. [Accessed: 23-Sep-2015].

[19] “GeoServer.” [Online]. Available: http://geoserver.org/. [Accessed: 23-Sep-2015].

[20] “OpenStreetMap.” [Online]. Available: https://www.openstreetmap.org/#map=5/51.500/-0.100. [Accessed: 23-Sep-2015].

[21] “TatukGIS - Geographic Information Systems.” [Online]. Available: http://www.tatukgis.com/. [Accessed: 23-Sep-2015].

[22] “Microsoft Silverlight.” [Online]. Available: http://www.microsoft.com/silverlight/. [Accessed: 23-Sep-2015].

[23] “End of Silverlight.” [Online]. Available: https://support.microsoft.com/en-us/lifecycle?c2=12905. [Accessed: 23-Sep-2015].

[24] “Top 3 Websites to Track Your Ship.” [Online]. Available: http://www.marineinsight.com/marine/marine-news/headline/top-3-websites-to-track-your-ship/. [Accessed: 23-Sep-2015].

[25] “Google Maps APIs | Google Developers.” [Online]. Available: https://developers.google.com/maps/?hl=en. [Accessed: 23-Sep-2015].

[26] “Live Ships Map - AIS - Vessel Traffic and Positions - AIS Marine Traffic.” [Online]. Available: http://www.marinetraffic.com/en/. [Accessed: 23-Sep-2015].

[27] “Thorium Mobile VMS Vessel Monitoring System.” [Online]. Available: http://www.thoriumvms.com/. [Accessed: 23-Sep-2015].

[28] “CLS America .” [Online]. Available: http://www.clsamerica.com/. [Accessed: 23-Sep-2015].

[29] “Globavista - Solutions, Software and Technologies for Global Tracking and Positioning.” [Online]. Available: http://www.globavista.co.uk/. [Accessed: 23-Sep-2015].

[30] “Iridium Short Burst Data.” [Online]. Available: http://www.satcomglobal.com/iridium-short-burst-data. [Accessed: 23-Sep-2015].

[31] “Thorium - Android Apps on Google Play.” [Online]. Available: https://play.google.com/store/apps/details?id=com.clsa&hl=en. [Accessed: 23-Sep-2015].

[32] “Thorium - How it works.” [Online]. Available: http://www.thoriumvms.com/#!thorium-how-it-works/c1hx8. [Accessed: 23-Sep-2015].

[33] “Argos - Worldwide tracking and environmental monitoring by satellite.” [Online]. Available: http://www.argos-system.org/?nocache=0.9469674903675851. [Accessed: 23-Sep-2015].

[34] “Inmarsat - Home.” [Online]. Available: http://www.inmarsat.com/. [Accessed: 23-Sep-2015].

[35] “What is Geographic Information Systems (GIS)? Webopedia.” [Online]. Available: http://www.webopedia.com/TERM/G/GIS.html. [Accessed: 23-Sep-2015].

[36] “ArcGIS | Main.” [Online]. Available: https://www.arcgis.com/features/. [Accessed: 23-Sep-2015].

[37] “Esri - GIS Mapping Software, Solutions, Services, Map Apps, and Data.” [Online]. Available: http://www.esri.com/. [Accessed: 23-Sep-2015].

[38] “Welcome to the OGC | OGC.” [Online]. Available: http://www.opengeospatial.org/. [Accessed: 23-Sep-2015].

[39] “OGC Members | OGC.” [Online]. Available: http://www.opengeospatial.org/ogc/members. [Accessed: 23-Sep-2015].

[40] “Web Map Service | OGC.” [Online]. Available: http://www.opengeospatial.org/standards/wms. [Accessed: 23-Sep-2015].

[41] “Using a Web Feature Service | OGC Network.” [Online]. Available: http://www.ogcnetwork.net/wfstutorial. [Accessed: 23-Sep-2015].

[42] “Welcome to MapServer — MapServer 7.0.0 documentation.” [Online]. Available: http://mapserver.org/. [Accessed: 23-Sep-2015].

[43] “MapGuide Project Home | MapGuide Open Source.” [Online]. Available: https://mapguide.osgeo.org/. [Accessed: 23-Sep-2015].

[44] “OpenLayers 3 - Welcome.” [Online]. Available: http://openlayers.org/. [Accessed: 23-Sep-2015].

[45] “Leaflet - a JavaScript library for interactive maps.” [Online]. Available: http://leafletjs.com/. [Accessed: 23-Sep-2015].

[46] “MapFish — MapFish.” [Online]. Available: http://mapfish.org/. [Accessed: 23-Sep-2015].

[47] “BSD 2-Clause License (FreeBSD/Simplified) Explained in Plain English - TLDRLegal.” [Online]. Available: https://tldrlegal.com/license/bsd-2-clause-license-(freebsd). [Accessed: 23-Sep-2015].

[48] “The 2012 free and open source GIS software map – A guide to facilitate research, development, and adoption - Open_source_GIS_CEUS_2013.pdf.” [Online]. Available: http://www.geo-informatie.nl/courses/grs32806/course/2015/Staff_presentations/Open_source_GIS_CEUS_2013.pdf. [Accessed: 23-Sep-2015].

[49] “JavaScript Toolkit for Rich Web Mapping Applications — GeoExt v1.1.” [Online]. Available: http://www.geoext.org/. [Accessed: 23-Sep-2015].

[50] “OpenLayers 2.” [Online]. Available: http://openlayers.org/two/. [Accessed: 23-Sep-2015].

[51] “Stack Overflow.” [Online]. Available: http://stackoverflow.com/. [Accessed: 23-Sep-2015].

[52] “Bing Maps Tile System.” [Online]. Available: https://msdn.microsoft.com/en-us/library/bb259689.aspx. [Accessed: 23-Sep-2015].

[53] “FAQ | Google Maps APIs | Google Developers.” [Online]. Available: https://developers.google.com/maps/faq. [Accessed: 23-Sep-2015].

[54] “OpenStreetMap Foundation Wiki.” [Online]. Available: https://wiki.osmfoundation.org/wiki/Main_Page. [Accessed: 23-Sep-2015].

[55] “ODC Open Database License (ODbL) Explained in Plain English - TLDRLegal.” [Online]. Available: https://tldrlegal.com/license/odc-open-database-license-(odbl). [Accessed: 23-Sep-2015].

[56] W. Royce, “Managing the development of large software systems,” *Proc. IEEE WESCON*, 1970.

[57] “Beyond the waterfall : software development at Microsoft,” 1995.

[58] “Waterfall Methodology: There’s no such thing!” [Online]. Available: http://www.idinews.com/waterfall.html. [Accessed: 25-Sep-2015].

[59] “Browser Statistics.” [Online]. Available: http://www.w3schools.com/browsers/browsers_stats.asp. [Accessed: 23-Sep-2015].

[60] “Visual Studio - Microsoft Developer Tools.” [Online]. Available: https://www.visualstudio.com/en-us. [Accessed: 23-Sep-2015].

[61] “ScottGu’s Blog - Introducing ‘Razor’ – a new view engine for ASP.NET.” [Online]. Available: http://weblogs.asp.net/scottgu/introducing-razor. [Accessed: 23-Sep-2015].

[62] “Knockout : Home.” [Online]. Available: http://knockoutjs.com/. [Accessed: 23-Sep-2015].

[63] “Getting started · Bootstrap.” [Online]. Available: http://getbootstrap.com/getting-started/. [Accessed: 23-Sep-2015].

[64] “JavaScript | MDN.” [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript. [Accessed: 23-Sep-2015].

[65] “jQuery.” [Online]. Available: https://jquery.com/. [Accessed: 23-Sep-2015].

[66] “Modernizr: the feature detection library for HTML5/CSS3.” [Online]. Available: https://modernizr.com/. [Accessed: 23-Sep-2015].

[67] “respond.js | JavascriptOO.” [Online]. Available: http://www.javascriptoo.com/respond-js. [Accessed: 23-Sep-2015].

[68] “ASP.NET MVC Overview.” [Online]. Available: https://msdn.microsoft.com/en-us/library/dd381412%28v=vs.108%29.aspx. [Accessed: 27-Sep-2015].

[69] “The MVC Pattern and ASP.NET MVC - Back to Basics.” [Online]. Available: http://www.dotnetcurry.com/aspnet-mvc/922/aspnet-mvc-pattern-tutorial-fundamentals. [Accessed: 23-Sep-2015].

[70] “NuGet Gallery | PagedList.Mvc 4.5.0.” [Online]. Available: https://www.nuget.org/packages/PagedList.Mvc. [Accessed: 23-Sep-2015].

[71] “ASP.NET MVC HTML Helpers.” [Online]. Available: http://www.w3schools.com/aspnet/mvc_htmlhelpers.asp. [Accessed: 23-Sep-2015].

[72] “MultiSelectList Class (System.Web.Mvc).” [Online]. Available: https://msdn.microsoft.com/en-us/library/system.web.mvc.multiselectlist(v=vs.118).aspx. [Accessed: 27-Sep-2015].

[73] “HttpContext.Cache Property (System.Web).” [Online]. Available: https://msdn.microsoft.com/en-us/library/system.web.httpcontext.cache(v=vs.110).aspx. [Accessed: 27-Sep-2015].

[74] “SessionStateAttribute Class (System.Web.Mvc).” [Online]. Available: https://msdn.microsoft.com/en-

us/library/system.web.mvc.sessionstateattribute(v=vs.118).aspx. [Accessed: 27-Sep-2015].

[75] “Did Your Session Really Expire? - Brian Hartman’s Report Viewer Blog - Site Home - MSDN Blogs.” [Online]. Available: http://blogs.msdn.com/b/brianhartman/archive/2009/02/15/did-your-session-really-expire.aspx. [Accessed: 28-Sep-2015].

[76] “TempDataDictionary Class (System.Web.Mvc).” [Online]. Available: https://msdn.microsoft.com/en-us/library/system.web.mvc.tempdatadictionary(v=vs.118).aspx. [Accessed: 27-Sep-2015].

[77] “jQuery.ajax() | jQuery API Documentation.” [Online]. Available: http://api.jquery.com/jquery.ajax/. [Accessed: 27-Sep-2015].

[78] “JsonResult Class (System.Web.Mvc).” [Online]. Available: https://msdn.microsoft.com/en-us/library/system.web.mvc.jsonresult%28v=vs.118%29.aspx. [Accessed: 23-Sep-2015].

[79] “OpenLayers 3 API Reference - Class: Feature.” [Online]. Available: http://openlayers.org/en/master/apidoc/ol.Feature.html. [Accessed: 23-Sep-2015].

[80] “OpenLayers 3 API Reference - Class: Geometry.” [Online]. Available: http://openlayers.org/en/master/apidoc/ol.geom.Geometry.html. [Accessed: 23-Sep-2015].

[81] “OpenLayers 3 API Reference - Class: Vector.” [Online]. Available: http://openlayers.org/en/master/apidoc/ol.source.Vector.html. [Accessed: 23-Sep-2015].

[82] “OpenLayers 3 API Reference - Class: Vector.” [Online]. Available: http://openlayers.org/en/master/apidoc/ol.layer.Vector.html. [Accessed: 23-Sep-2015].

[83] “Cookie FAQ.” [Online]. Available: http://www.aboutcookies.org/default.aspx?page=5. [Accessed: 27-Sep-2015].

[84] “HTML5 Web Storage.” [Online]. Available: http://www.w3schools.com/html/html5_webstorage.asp. [Accessed: 27-Sep-2015].

[85] “Declaring and Using Callbacks - Mozilla | MDN.” [Online]. Available: https://developer.mozilla.org/en-US/docs/Mozilla/js-ctypes/Using_js-ctypes/Declaring_and_Using_Callbacks. [Accessed: 29-Sep-2015].

[86] “OpenLayers 3 - LayerSwitcher.” [Online]. Available: http://rawgit.com/walkermatt/ol3-layerswitcher/master/examples/layerswitcher.html. [Accessed: 23-Sep-2015].

[87] “maps.stamen.com.” [Online]. Available: http://maps.stamen.com/#toner/12/37.7706/-122.3782. [Accessed: 23-Sep-2015].

[88] “Google layer: Where is new ol.source.Google ? - Google Groups.” [Online]. Available: https://groups.google.com/forum/?utm_medium=email&utm_source=footer#!msg/ol3-dev/ZV8LB-F5904/x1sspizm3xMJ. [Accessed: 23-Sep-2015].

[89] “Earthquakes in KML.” [Online]. Available: http://openlayers.org/en/v3.9.0/examples/kml-earthquakes.html?q=tooltip. [Accessed: 27-Sep-2015].

[90] “OpenLayers 3 API Reference - Class: Cluster.” [Online]. Available: http://openlayers.org/en/master/apidoc/ol.source.Cluster.html. [Accessed: 23-Sep-2015].

[91] “OpenLayers 3 API Reference - Class: Overlay.” [Online]. Available: http://openlayers.org/en/master/apidoc/ol.Overlay.html. [Accessed: 23-Sep-2015].

[92] “Vector Icon Example.” [Online]. Available: http://openlayers.org/en/v3.9.0/examples/icon.html. [Accessed: 24-Sep-2015].

[93] “OpenLayers 3 API Reference - Class: LineString.” [Online]. Available: http://openlayers.org/en/master/apidoc/ol.geom.LineString.html. [Accessed: 23-Sep-2015].

[94] “The MVVM Pattern.” [Online]. Available: https://msdn.microsoft.com/en-us/library/hh848246.aspx. [Accessed: 23-Sep-2015].

[95] “Knockout : Observable Arrays.” [Online]. Available: http://knockoutjs.com/documentation/observableArrays.html. [Accessed: 23-Sep-2015].

[96] “OpenLayers 3 API Reference - Class: Draw.” [Online]. Available: http://openlayers.org/en/master/apidoc/ol.interaction.Draw.html. [Accessed: 23-Sep-2015].

[97] “HttpPostAttribute Class (System.Web.Mvc).” [Online]. Available: https://msdn.microsoft.com/en-us/library/system.web.mvc.httppostattribute(v=vs.118).aspx. [Accessed: 27-Sep-2015].

[98] “ValidateAntiForgeryTokenAttribute Class (System.Web.Mvc).” [Online]. Available: https://msdn.microsoft.com/en-us/library/system.web.mvc.validateantiforgerytokenattribute(v=vs.118).aspx. [Accessed: 27-Sep-2015].

[99] “AuthorizeAttribute Class (System.Web.Mvc).” [Online]. Available: https://msdn.microsoft.com/en-us/library/system.web.mvc.authorizeattribute(v=vs.118).aspx. [Accessed: 27-Sep-2015].

[100] “Setting the Cacheability of a Page.” [Online]. Available: https://msdn.microsoft.com/en-us/library/w9s3a17d.aspx. [Accessed: 27-Sep-2015].

[101] “World Wide Web Consortium (W3C).” [Online]. Available: http://www.w3.org/. [Accessed: 23-Sep-2015].

[102] “The W3C Markup Validation Service.” [Online]. Available: https://validator.w3.org/. [Accessed: 23-Sep-2015].

[103] “Markup Validation in Visual Studio.” [Online]. Available: https://msdn.microsoft.com/en-us/library/f940516c(v=vs.110).aspx. [Accessed: 24-Sep-2015].

[104] “Completing the JavaScript Test Stack: Introducing Gremlins.js, a Monkey Test Library For Web Apps.” [Online]. Available: http://www.redotheweb.com/2014/01/07/completing-the-js-test-stack-introducing-greminsjs.html. [Accessed: 23-Sep-2015].

[105] “Chosen: A jQuery Plugin by Harvest to Tame Unwieldy Select Boxes.” [Online]. Available: https://harvesthq.github.io/chosen/. [Accessed: 27-Sep-2015].

[106] “Introduction to SignalR | The ASP.NET Site.” [Online]. Available: http://www.asp.net/signalr/overview/getting-started/introduction-to-signalr. [Accessed: 23-Sep-2015].

[107] “Page.Culture Property (System.Web.UI).” [Online]. Available: https://msdn.microsoft.com/en-us/library/system.web.ui.page.culture.aspx. [Accessed: 23-Sep-2015].

[108] “Page.UICulture Property (System.Web.UI).” [Online]. Available: https://msdn.microsoft.com/en-us/library/system.web.ui.page.uiculture.aspx. [Accessed: 23-Sep-2015].

[109] “Nadeem Afana’s blog · ASP.NET MVC 5 Internationalization.” [Online]. Available: http://afana.me/post/aspnet-mvc-internationalization.aspx. [Accessed: 23-Sep-2015].

[110] “Run performance tests on your app.” [Online]. Available: https://msdn.microsoft.com/en-us/library/dn250793%28v=vs.120%29.aspx. [Accessed: 23-Sep-2015].

Anexo I, Análise de requistos

De seguida, é apresentada a análise dos requisitos inicial que foi feita para o Web Vessel Tracker. Com o decorrer do desenvolvimento da aplicação alguns destes requisitos acabaram por ser agrupar, outros foram postos de parte, e outros acabaram por ficar como trabalho futuro. Os requisitos que se encontram a cinzento não foram implementados neste ciclo de desenvolvimento. Primeiramente, são apresentados os requisitos não funcionais e de seguida os funcionais.

Requisitos não funcionais	**Prioridade**
O sistema operar por HTTP e HTTPS.	Alta
O lado do servidor é desenvolvido com recurso ao ASP.NET MVC 5.	Alta
O lado do cliente utiliza tecnologias de *wep mapping* para a representação geográfica de informação.	Alta
O sistema devolve dados em formato JSON.	Alta
O sistema pode ser alojado no servidor IIS.	Média
O sistema é desenvolvido no IDE Visual Studio 2013.	Baixa

Requisitos funcionais	**Prioridade**
Autenticação	
A aplicação deve permitir a autenticação do utilizador.	Alta
Uma vez autenticado, deverá aparecer alguma informação a indicar que as	Média

embarcações estão a ser carregadas.	
A aplicação deve mostrar uma mensagem de aviso caso a autenticação não seja bem sucedida.	Alta
Rotas e embarcações	
Deve de ser possível criar rotas no mapa com recurso à informação recolhida pelo Monicap e/ou pelo AIS.	Alta
A aplicação deve permitir escolher o período para ver a rota de uma determinada embarcação.	Alta
A aplicação deve apresentar ao utilizador o ponto mais recente de cada embarcação tendo em conta um intervalo de tempo pré-definido.	Alta
A aplicação deve apresentar ícones diferentes para os diferentes tipos de embarcações.	Média
A aplicação deve apresentar um *tooltip* nos ícones das embarcações com o seu nome.	Média
Os ícones devem ser gerados de forma a indicar ao utilizador o rumo da embarcação.	Baixa
A aplicação deve permitir mostrar o número de embarcações carregadas para o mapa.	Baixa
A aplicação deve permitir ao utilizador ver os diários de pesca de uma embarcação.	Alta
A aplicação deve apresentar uma *popup* com informação relativa à embarcação quando o utilizador seleciona um ícone.	Alta
Quando uma rota de uma embarcação é gerada, a informação relativa a essa rota deve de ser exposta numa tabela na mesma página.	Alta
Na tabela com informação relativa à rota das embarcações deve ser possível	Alta

escolher qual a rota da embarcação que se pretende visualizar.	
A cor da rota de uma embarcação deve ficar mais itensa à medida que o último ponto é aproximado.	Média
Áreas	
A aplicação deve criar áreas pré-definidas.	Alta
Pesquisas e filtros	
A aplicação deve permitir pesquisar por rotas numa área tendo em conta um determinado período.	Alta
A aplicação deve permitir pesquisar por embarcações que utilizam o Monicap e/ou AIS como sistemas de monitorização	Alta
A aplicação permite pesquisar embarcações por país de origem.	Baixa
A aplicação deve permitir a pesquisa de embarcações por MMSI ou pelo seu nome.	Média
A aplicação permite pesquisar embarcações pelas áreas definidas no sistema.	Média
Na página destinada à pesquisa das embarcações deve existir uma tabela que é carregada com informação relativa a todas as embarcações que estão representadas no mapa.	Alta
A aplicação deve desenhar a rota de todas as embarcações filtradas que foram selecionadas pelo utilizador na tabela das embarcações.	Alta
A aplicação deve transitar para a embarcação que foi escolhida pelo utilizador na tabela das embarcações.	Média
Caixa de ferramentas/utilidades	
A aplicação deve permitir ao utilizador mudar os *Map tiles.*	Alta
Deve de ser possível ao utilizador imprimir o mapa.	Baixa

Deve de ser possível ao utilizador calcular a distância entre dois pontos.	Média
A aplicação deve permitir o desenho das rotas tendo em conta a evolução da velocidade da embarcação.	Alta
A aplicação deve de permitir o desenho das rotas tendo em conta a evolução da quantidade de peixe pescado pela embarcação.	Alta
A aplicação deve suportar o número máximo de *map tiles/map providers* possíveis.	Alta
A aplicação deve permitir aos utilizadores criar áreas.	Alta
A aplicação deve permitir editar áreas criadas pelos utilizadores	Média
A aplicação deve permitir remover áreas criadas pelos utilizadores.	Média
Opções e funcionalidades genéricas	
A aplicação deve permitir fazer *zoom in* e *zoom out* a localizações no mapa.	Baixa
A aplicação deve mostrar as coordenadas do cursor no mapa.	Baixa
A aplicação deve permitir a navegação do mapa através do cursor.	Alta
A aplicação deve mostrar a escala que o mapa está a utilizar num determinado momento.	Baixa
A aplicação deve permitir ao utilizador repor o mapa para o estado "original".	Média
A aplicação deve permitir que um utilizador possa voltar à sua sessão anterior depois de fechar a aplicação.	Média
A aplicação deve apresentar uma foto, se possível, da embarcação que se está observar assim como alguns dos seus detalhes.	Média
A aplicação deve mostrar visualmente a percentagem de ocupação da embarcação com capturas.	Média

Anexo II, Planeamento

Este anexo ilustra a tabela das tarefas realizadas no projeto desenvolvido.

Task Name	Duration	Start	Finish
Estudo do estado da arte do ponto de vista das metodologias de desenvolvimento ASP.NET MVC. Avaliar integração com outras plataformas de mapas.	**23 days**	**Mon 15-09-14**	**Wed 15-10-14**
Estudo das diferentes metodologias de desenvolvimento em ASP.NET MVC	2 days	Mon 15-09-14	Tue 16-09-14
Estudo das tecnologias GIS	**11 days**	**Wed 17-09-14**	**Wed 01-10-14**
Estudo das várias plataformas de mapas existentes	5 days	Wed 17-09-14	Tue 23-09-14
Estudo dos vários geoservidores	3 days	Wed 24-09-14	Fri 26-09-14
Estudo das várias plataformas de desenvolvimento para web mapping	3 days	Mon 29-09-14	Wed 01-10-14
Estudo dos sistemas de localização de embarcações	**10 days**	**Thu 02-10-14**	**Wed 15-10-14**
Estudo dos sistemas VMS	5 days	Thu 02-10-14	Wed 08-10-14
Análise a vários sistemas VMS	4 days	Thu 09-10-14	Tue 14-10-14
Estudo dos sistemas AIS	1 day	Wed 15-10-14	Wed 15-10-14
Elaboração de um documento de desenho do trabalho a desenvolver, incluindo a elaboração de "mockups" das páginas do site a desenvolver.	**18 days**	**Thu 16-10-14**	**Mon 10-11-14**
Análise de requisitos	3 days	Thu 16-10-14	Mon 20-10-14
Recolha de informação sobre o webCC e o Monicap	5 days	Tue 21-10-14	Mon 27-10-14
Desenho de mockups	10 days	Tue 28-10-14	Mon 10-11-14
Implementação de uma "spike" para utilizar funcionalidades básicas do openlayers 3	6 days	Tue 11-11-14	Tue 18-11-14
Desenvolvimento da aplicação Web Vessel Tracker	**149 days**	**Wed 19-11-14**	**Mon 15-06-15**

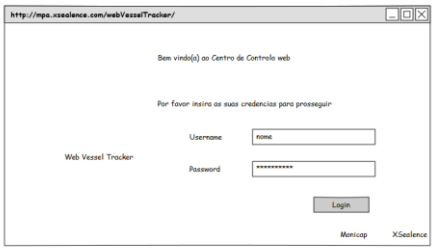
Desenvolvimento da parte gráfica da aplicação (as views)	**19 days**	**Wed 19-11-14**	**Mon 15-12-14**
Integração da spike desenvolvida com o projeto	2 days	Wed 19-11-14	Thu 20-11-14
Criação do esqueleto da página "mapa"	7 days	Fri 21-11-14	Mon 01-12-14
Criação do esqueleto da página “opções”	1 day	Tue 02-12-14	Tue 02-12-14
Implementação de uma opção para mudar os icones das embarcações	4 days	Wed 03-12-14	Mon 08-12-14
Criação do esqueleto da página pesquisar	5 days	Tue 09-12-14	Mon 15-12-14
Implementação das funcionalidades para ver as rotas e detalhes das embarcações Monicap e AIS	**39 days**	**Tue 16-12-14**	**Fri 06-02-15**
Implementação das rotas das embarcações	8 days	Tue 16-12-14	Thu 25-12-14
Implementação dos icones e dos tooltips para as embarcações	5 days	Fri 26-12-14	Thu 01-01-15
Implementação de um popup com informação relativa à embarcação selecionada	7 days	Fri 02-01-15	Mon 12-01-15
Implementação de uma tabela que apresenta a informação das rotas das embarcações	2 days	Tue 13-01-15	Wed 14-01-15
Implementação de uma opção para visualizar intervalos de tempo nas rotas	7 days	Thu 15-01-15	Fri 23-01-15
Implementação de um mecanismo para atualizar automaticamente a posição das embarcações	10 days	Mon 26-01-15	Fri 06-02-15
Implementação das funcionalidades auxiliares	**24 days**	**Fri 06-02-15**	**Wed 11-03-15**
Implementação de uma opção para mudar os "map tiles"	2 days	Fri 06-02-15	Mon 09-02-15
Implementação do desenho e criação de áreas.	17 days	Tue 10-02-15	Wed 04-03-15
Implementação de uma funcionalidade que permite fazer o cálculo da distância de pontos	2 days	Thu 05-03-15	Fri 06-03-15
Implementação de uma opção para repôr o mapa para o estado original	2 days	Mon 09-03-15	Tue 10-03-15
Implementação de uma funcionalidade que permite filtrar embarcações no mapa por dispositivo	1 day	Wed 11-03-15	Wed 11-03-15
Implementação da página Pesquisar	**37 days**	**Thu 12-03-15**	**Fri 01-05-15**
Implementação de filtros para pesquisar por país de origem, tipo de embarcação, dispositivo utilizado	16 days	Thu 12-03-15	Thu 02-04-15
Implementação de um filtro para pesquisar uma embarcação por nome	1 day	Fri 03-04-15	Fri 03-04-15
Implementação de uma tabela com as embarcações que foram carregadas para o mapa	3 days	Mon 06-04-15	Wed 08-04-15
Implementação de uma funcionalidade para ordenar a tabela por nome, rumo, velocidade, data do último ponto ou país de	12 days	Thu 09-04-15	Fri 24-04-15

origem			
Implementação de uma funcionalidade que permita ao utilizador visualizar uma embarcação selecionada na tabela da página pesquisar no mapa.	5 days	Mon 27-04-15	Fri 01-05-15
Implementação de um sistema de autenticação	3 days	Mon 04-05-15	Wed 06-05-15
Implementação dos diários de pesca	5 days	Thu 07-05-15	Wed 13-05-15
Realização de testes de usabilidade e melhorias à interface gráfica	5 days	Thu 14-05-15	Wed 20-05-15
Documentação técnica	18 days	Thu 21-05-15	Mon 15-06-15

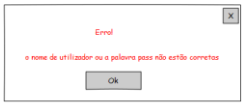
Anexo III, Mockups relevantes

Este anexo ilustra alguns dos *mockups* que foram desenvolvidos para aplicação. Dado que existem muitas só algumas é que são apresentados, nomeadamente os que ilustram as funcionalidades mais relevantes da aplicação.

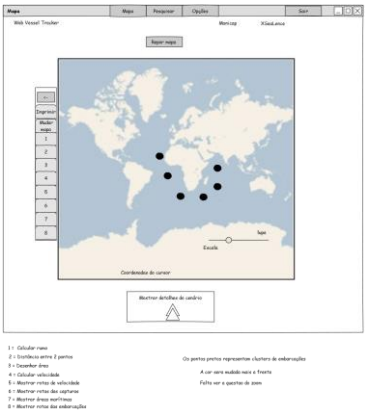
Login:



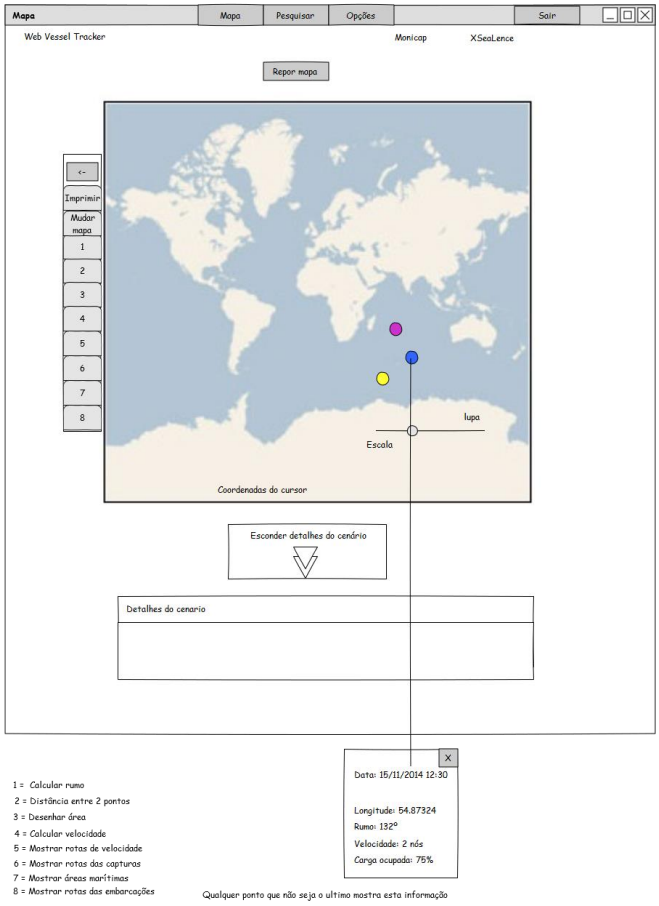
caso a autenticação falhe



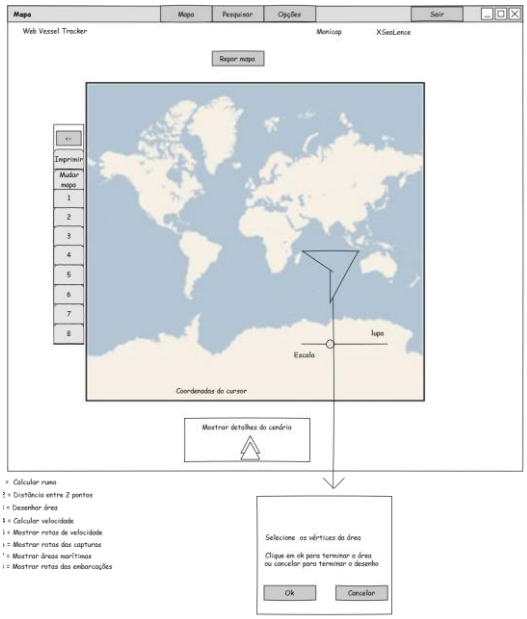
Mapa inicial:



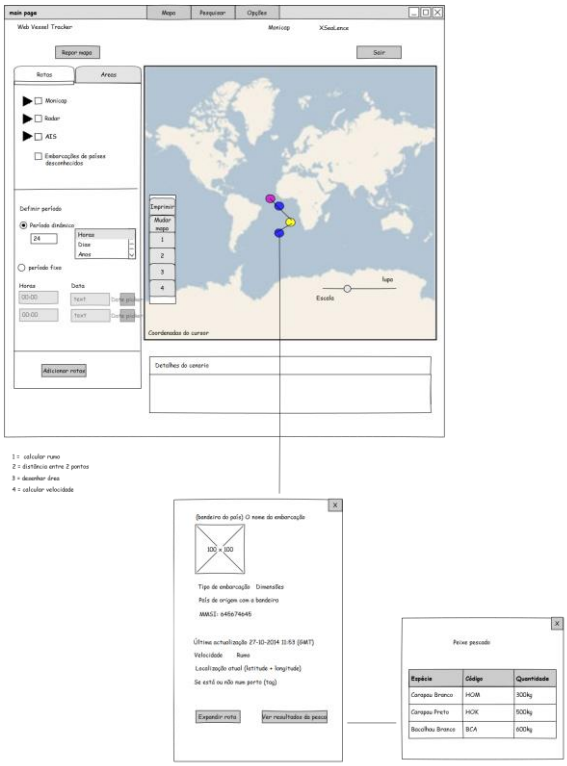
Embarcação selecionada:



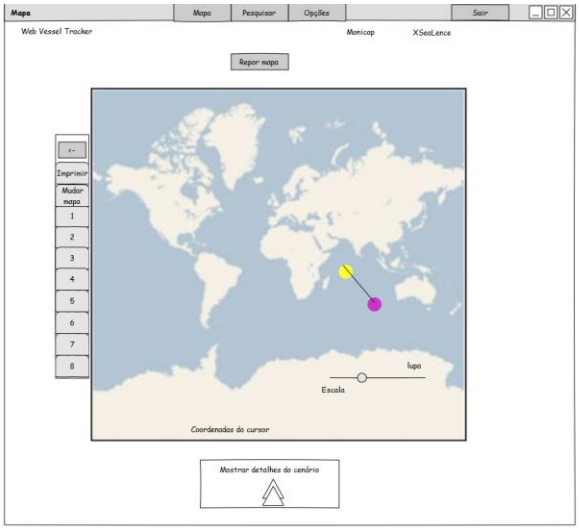
Desenhar área:



Diários de pesca:



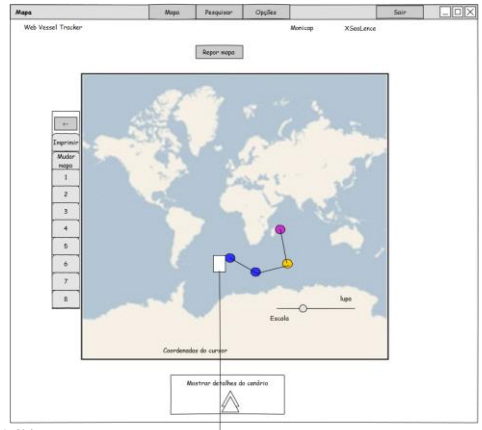
Troca de mapas:



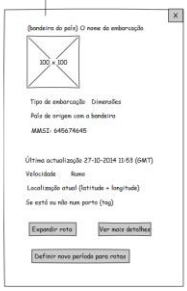
1 = Calcular rumo
2 = Distância entre 2 pontos
3 = Desenhar área
4 = Calcular velocidade
5 = Mostrar rotas de velocidade
6 = Mostrar rotas das capturas
7 = Mostrar áreas marítimas
8 = Mostrar rotas das embarcações

Estas treeviews têm depois outras opções para os map tiles de cada fornecedor

Rota:



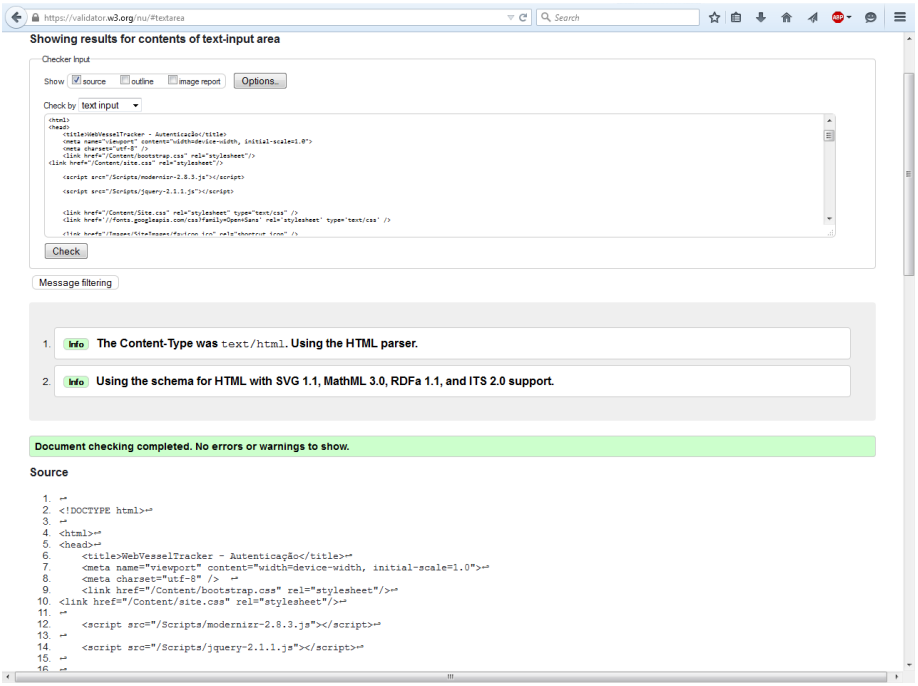
1 = Calcular rumo
2 = Distância entre 2 pontos
3 = Desenhar área
4 = Calcular velocidade
5 = Mostrar rotas de velocidade
6 = Mostrar rotas das capturas
7 = Mostrar áreas marítimas
8 = Mostrar rotas das embarcações



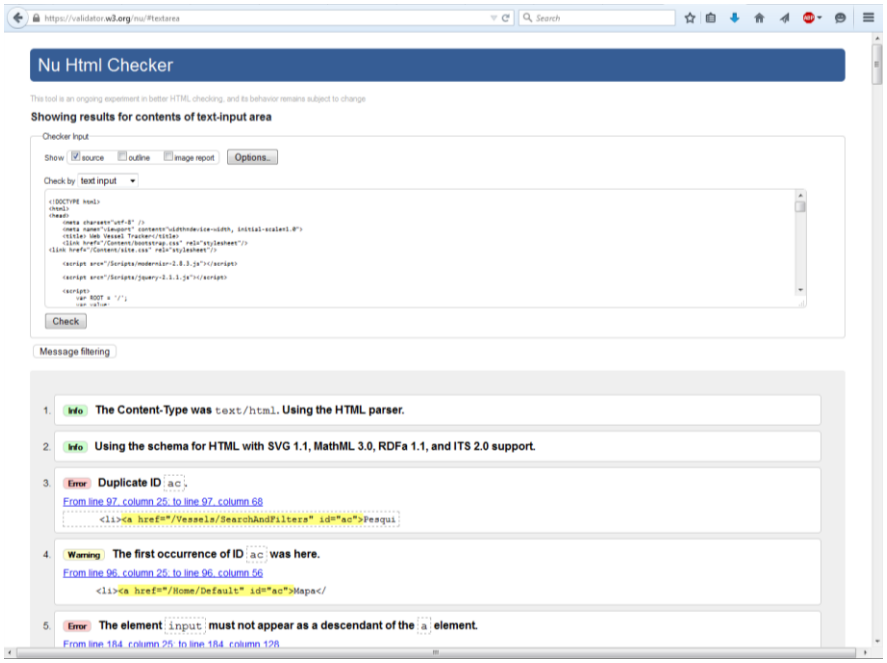
Anexo IV, Resultados da validação de HTML

Este anexo serve para ilustrar os resultados obtidos da validação de HTML da aplicação.

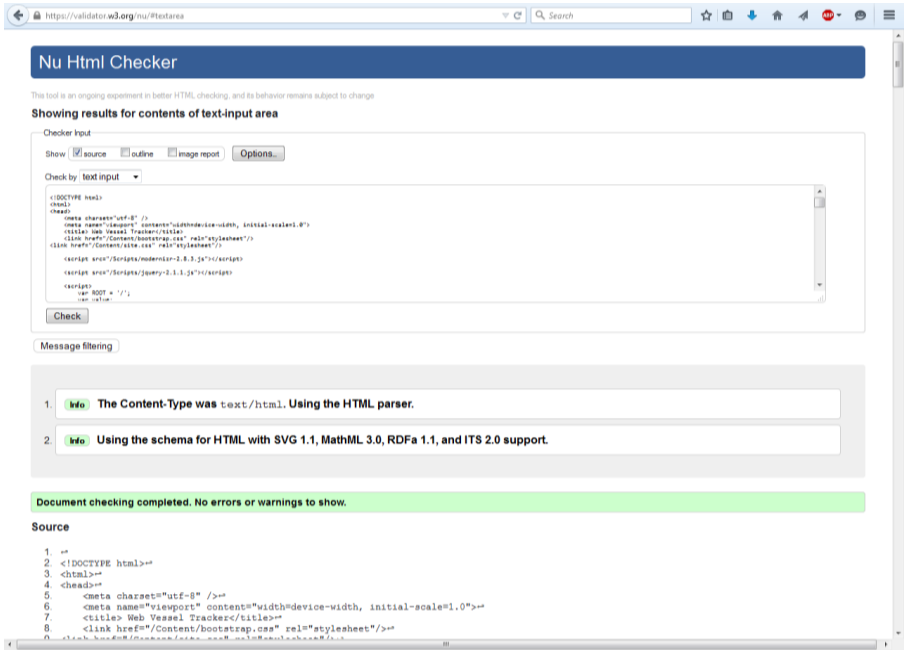
Página de login:



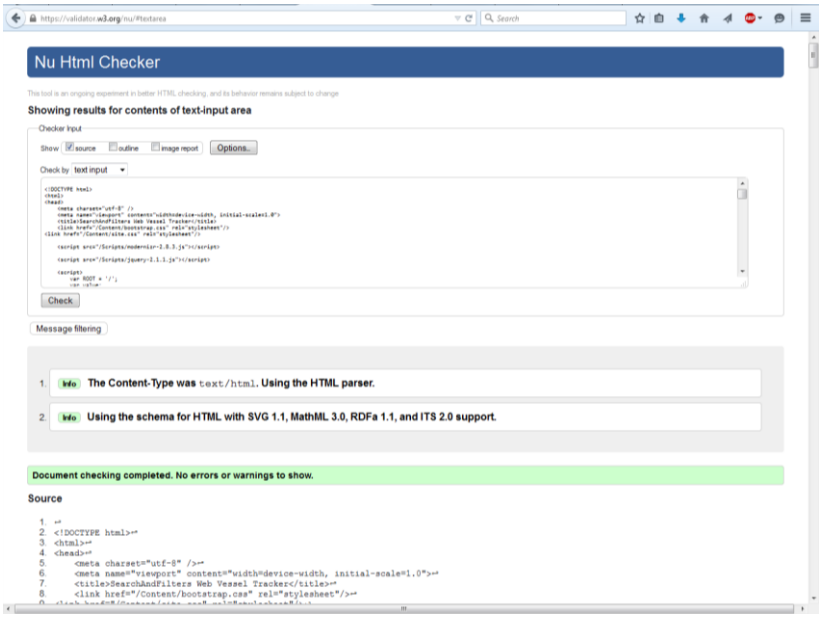
Página principal:



Teste final:



Página Pesquisar:



Anexo V, Guia de utilização

De forma a testar a usabilidade do projeto foi desenvolvido um guia de teste de utilização com as seguintes tarefas:

1. Fazer o login.
2. Ver a informação de uma embarcação no mapa.
3. Pesquisar por uma embarcação (nome).
4. Filtrar embarcações.
5. Ver a carga que uma embarcação tem a bordo.
6. Ver uma rota dessa embarcação.
7. Ver os detalhes da rota de uma embarcação.
8. Ver uma área do sistema ou pública.
9. Desenhar uma área pessoal.
10. Ver essa área pessoal.
11. Pesquisar áreas (nome muito mal, tenho de descrever melhor esta acção).
12. Esconder/Apagar uma área pessoal.
13. ágina do Pesquisar.
14. Filtrar uma embarcação por tipo é país.
15. Ver essa embarcação no mapa da pgina principal.
16. Fazer logout.