Aplicación web para el seguimiento de embarcaciones registradas por dispositivos AIS

Grado en Ingeniería Informática

AUTOR: Jesús Artiles Cidoncha

TUTORIZADO POR:
Javier Sánchez Pérez y Nelson Monzón López

Julio/2021

Agradecimientos

Me gustaría agradecer a todas las personas que me han ayudado durante mi travesía universitaria y que han aportado a la realización de este proyecto.

En primer lugar, me gustaría dar las gracias a mi familia y amigos, por apoyarme diariamente y por ayudarme a alcanzar mis objetivos.

También, me gustaría agradecer a mis tutores Nelson Monzón López y Javier Sánchez Pérez por la ayuda que me han ofrecido durante el proyecto.

Agradecer a todas las personas del equipo QAISC [1], por tener paciencia conmigo y aceptarme en su equipo. Especialmente a Alejandro Cabrera y Sergio Ballines por ayudarme en el desarrollo del back-end necesario para este proyecto y Andrés Alonso-Martirena por ejercer de Product Owner aportando ideas y feedback durante el desarrollo de las funcionalidades.

Finalmente, agradecer a la Plataforma Oceánica de Canarias [2] (PLOCAN), y su red de observación de cámaras costeras, y a la Autoridad Portuaria de Las Palmas de Gran Canaria [3] por ofrecernos el entorno de trabajo necesario para la realización de este TFG.

[1] www.qaisc.com
[2] https://www.plocan.eu/
[3] http://www.palmasport.es/es/

Resumen

En este Trabajo Fin de Grado (TFG) se ha desarrollado una aplicación web que permite visualizar registros de embarcaciones capturados por sistemas de detección automáticos (*Automatic Detection System*, AIS). Desde la aplicación, es posible acceder a registros obtenidos minutos previos al momento de la consulta, así como a registros guardados con días o incluso semanas de anterioridad. Además, se puede consultar información detallada de las embarcaciones representadas, así como utilizar diferentes herramientas de visualización, como representar el recorrido llevado a cabo por una embarcación a lo largo del tiempo o filtrar la representación según el tipo de navío para consultar la información más relevante en cada momento.

Este trabajo se ha desarrollado en colaboración con la empresa Qualitas Artificial Intelligence And Science S.A.U.(QAISC) y la Plataforma Oceánica de Canarias (PLOCAN), que han facilitado el acceso a dos dispositivos AIS localizados en el puerto de la Luz y de Las Palmas y en las oficinas de PLOCAN en la zona de Taliarte (Telde).

Abstract

In this final degree project, we have developed a web application that allows the visualization of vessel records captured by Automatic Detection Systems (AIS). From the application, it will be possible to access records obtained minutes prior to the time of the query, as well as records saved days or even weeks in advance. In addition, detailed information on the represented vessels can be consulted, as well as the use of different visualization tools, such as representing the route carried out by a vessel over time or filtering the representation according to type, to consult the most relevant information at all times.

This work has been developed in collaboration with the company Qualitas Artificial Intelligence And Science S.A.U. (QAISC) and the canary islands oceanic platform (PLOCAN), which have facilitated access to two AIS devices located in Puerto de la Luz y de Las Palmas and at the PLOCAN offices in the Taliarte area (Telde).

Índice general

1. Introducción **9**
1.1. Antecedentes del proyecto . . . 9
1.2. Descripción del proyecto . . . 9
1.3. Objetivos del proyecto . . . 10
1.4. Competencias específicas . . . 11
1.5. Presupuesto . . . 11
1.5.1. Coste hardware . . . 11
1.5.2. Coste personal . . . 12
1.5.3. Presupuesto final . . . 12

2. Estado del arte **13**
2.1. Mapas interactivos . . . 13
2.1.1. Marine Traffic . . . 13
2.1.2. Windy . . . 15
2.2. Frameworks de creación de mapas interactivos . . . 17
2.2.1. Openlayers . . . 17
2.2.2. Leaflet . . . 18
2.2.3. Comparación . . . 18

3. Tecnologías utilizadas **21**
3.1. Angular . . . 21
3.2. Laravel . . . 21
3.3. Bootstrap . . . 22
3.4. Angular Material . . . 22
3.5. MongoDB . . . 23
3.6. MySQL . . . 23
3.7. Visual Studio Code . . . 24

4. Metodología aplicada **25**
4.1. Método Kanban . . . 25
4.1.1. Principios básicos del método Kanban . . . 25
4.1.2. Las 6 prácticas de Kanban . . . 25
4.1.3. Roles principales . . . 27
4.1.4. Gestión del tablero Kanban . . . 27

5. Desarrollo **29**
5.1. Análisis del documento de requisitos 29
5.2. Sprint Zero: Estudio de recursos disponibles 30
5.2.1. Recepción de mensajes AIS 30
5.2.2. Tratamiento de los mensajes en el contexto de la empresa 32
5.2.3. Desarrollo de la pila de producto 36
5.3. Etapas de desarrollo de la aplicación 40
5.3.1. Sprint 1: Creación del mapa 40
5.3.2. Sprint 2: Dibujar las embarcaciones en el mapa 41
5.3.3. Sprint 3: Mostrar información de embarcaciones 46
5.3.4. Sprint 4: Dibujar el recorrido de una embarcación 47
5.3.5. Sprint 5: Filtro de embarcaciones 48
5.3.6. Sprint 6: Sistema de autenticación y registro de usuarios 50
5.3.7. Sprint 7: Selector de mapa base 53
5.3.8. Sprint 8: Desarrollo de barra de tiempo 55

6. Conclusiones y trabajo futuro **59**

Índice de figuras

1.1. Embarcaciones dibujadas en el mapa 10

2.1. Representación de embarcaciones en MarineTraffic 13
2.2. Consulta de información de embarcaciones en MarineTraffic 14
2.3. Consulta de recorrido de una embarcación en MarineTraffic 15
2.4. Otras funcionalidades gratuitas de MarineTraffic 15
2.5. Vista general de la web de Windy 16
2.6. Barra de control de tiempo web de Windy 16
2.7. Barra de botones web de Windy 17

4.1. Esquema flujo de trabajo básico 26
4.2. Tablero KANBAN gestionado con TRELLO. 28
4.3. Ejemplo de historia de usuario utilizando una tarjeta TRELLO 28

5.1. Ejemplo de aspecto general de la web provisto por Product Owner 29
5.2. Ejemplo representación de embarcaciones provisto por Product Owner 30
5.3. Fragmento de JSON de posiciones (mensaje tipo 1,2,3) de barcos 35
5.4. Definición de la tabla ship 36
5.5. Vista del mapa al acceder a la aplicación 40
5.6. Vista normal y en pantalla completa de la web 41
5.7. Barra de navegación 41
5.8. Iconos utilizados para representación de embarcaciones[16] 44
5.9. Resultado del mapa con las embarcaciones dibujadas 45
5.10. Icono AIS desarrollado durante este proyecto 45
5.11. Barra de botones del mapa 45
5.12. Popup al hacer hover sobre embarcación 46
5.13. Popup al hacer click en embarcación 46
5.14. Recorrido de embarcaciones 48
5.15. Filtro de selección de embarcaciones a dibujar 49
5.16. Diagrama caso de uso interacción con embarcaciones 50
5.17. Formularios de login y registro 51
5.18. Campos obligatorios sin rellenar 51
5.19. Error de validación de campos 52
5.20. Error de login y registro 52
5.21. Diagrama caso de uso perfil de usuario 53

5.22. Selector de mapas base 53
5.23. Mapas base disponibles en la app 54
5.24. Vista de la barra de tiempo 55
5.25. Selector de intervalo máximo de la barra de tiempo 55
5.26. Burbuja de texto que indica el instante de tiempo al que equivale la posición de la barra a la que se apunta 56
5.27. Botones adicionales de la barra de tiempo 57
5.28. Diagrama caso de uso barra de tiempo 58

6.1. Relación futura de la tabla ship con tabla file 60

Índice de cuadros

1.1. Coste hardware del proyecto 11
1.2. Coste personal del proyecto 12
1.3. Presupuesto total del proyecto 12

5.1. Parámetros mensajes tipo 1, 2, 3 33
5.2. Parámetros mensaje tipo 5 34
5.3. Pila de producto del proyecto 40

Capítulo 1

Introducción

1.1. Antecedentes del proyecto

Los sistemas de detección automática (*Automatic Identification System*, AIS) son ampliamente utilizados en la detección y seguimiento de embarcaciones que transitan cerca de la costa. Son muy necesarios para que las estaciones portuarias y determinados organismos puedan identificar en instantes determinados, la posición GPS, rumbo y velocidad de dichas embarcaciones a partir de un visor PTZ[1]. Este sistema, tiene una gran importancia a la hora de evitar colisiones entre navíos gracias al alto ritmo de actualización de los datos.

En este contexto, la empresa Qualitas Artificial Intelligence and Science (QAISC), está realizando un proyecto que persigue desarrollar un mapa *online* con la posibilidad de visualizar información de una red de cámaras PTZ situadas en determinadas localizaciones costeras de la isla de Gran Canaria. Además, se tiene como objetivo representar en dicho mapa las embarcaciones detectadas por el AIS.

En este sentido, este Trabajo Fin de Grado (TFG) persigue el desarrollo de un prototipo que permita a distintos usuarios visualizar en directo las embarcaciones detectadas por el AIS y consultar determinada información perteneciente a los navíos cercanos a la costa de Gran Canaria.

1.2. Descripción del proyecto

El prototipo que se ha realizado es una aplicación web que permitirá visualizar en un mapa definido, las embarcaciones detectadas tanto en tiempo real como en instantes de tiempo anteriores. Las embarcaciones serán representadas en el mapa siguiendo una clasificación de

[1]PTZ significa Pan, Zoom y Tilt por sus siglas en inglés. Esto se refiere a las capacidades de las cámaras automatizadas y grabadoras de vídeo. Esencialmente, una cámara PTZ se establece en un lugar en particular, pero tiene la habilidad de girarse, usualmente en un círculo completo, inclinarse en ángulos diferentes hacia arriba y hacia abajo, y ampliar en objetos en particular.[13]

colores en base al tipo de navío, y mediante el uso de una barra de tiempo, será posible acceder a representar embarcaciones detectadas en días anteriores.

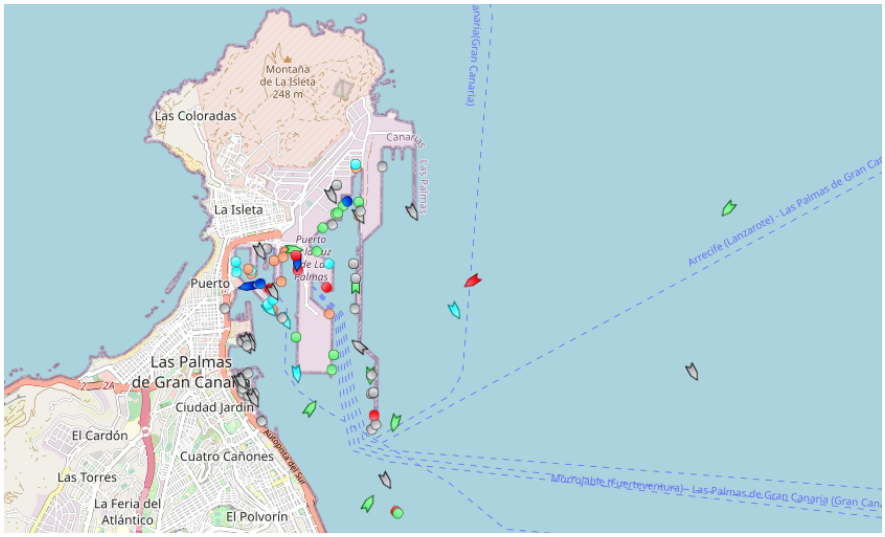


Ilustración 1.1: Embarcaciones dibujadas en el mapa

Además, el usuario será capaz de obtener cierta información básica de la embarcación al deslizar el cursor sobre cualquiera de las embarcaciones representadas y pulsar en ella para obtener un menú con información más detallada. Dicho menú, dará posibilidad al usuario de consultar el recorrido que ha seguido la embarcación a lo largo del tiempo. La web dispondrá de un filtro que permitirá visualizar solo aquellos tipos de embarcaciones que sean relevantes en cada momento.

Para la obtención de los datos necesarios para la correcta representación de los navíos y el control del flujo de usuarios en la web, la empresa dispone de una API, en la que se han realizado ciertas tareas de back-end. Entre estas tareas, se encuentra la de crear un set de peticiones para obtener los registros de embarcaciones según el instante de tiempo y obtener las localizaciones previas de una embarcación en concreto para representar el recorrido de la misma, entre otras que se explicarán en el apartado correspondiente. La información proveniente de las llamadas a la API, tendrá formato JSON.

1.3. Objetivos del proyecto

Los objetivos que se persiguen alcanzar con este proyecto son los siguientes:

- ✔ Familiarizarse con dispositivos AIS y profundizar en mensajes JSON entre cliente y servidor.
- ✔ Aprender nuevas tecnologías, extender conocimientos de ingeniería del software y aplicar técnicas de desarrollo nuevas para el alumno.

- ✔ Aplicación de un flujo de trabajo que aplique técnicas de integración y entrega continúa así como familiarización del alumno con metodologías ágiles en un contexto empresarial.
- ✔ Desarrollo de peticiones a la API de la empresa para crear y obtener la información necesaria para la representación de las embarcaciones.
- ✔ Desarrollo de una aplicación web para la representación de embarcaciones e información perteneciente a las mismas tanto en directo como en instantes pasados.

1.4. Competencias específicas

1. ***IS01.*** *Capacidad para desarrollar, mantener y evaluar servicios y sistemas software que satisfagan todos los requisitos del usuario y se comporten de forma fiable y eficiente, sean asequibles de desarrollar y mantener y cumplan normas de calidad, aplicando las teorías, principios, métodos y prácticas de la ingeniería del software.*
2. ***IS02.*** *Capacidad para valorar las necesidades del cliente y especificar los requisitos software para satisfacer estas necesidades, reconciliando objetivos en conflicto mediante la búsqueda de compromisos aceptables dentro de las limitaciones derivadas del coste, del tiempo, de la existencia de sistemas ya desarrollados y de las propias organizaciones.*
3. ***IS03.*** *Capacidad de dar solución a problemas de integración en función de las estrategias, estándares y tecnologías disponibles.*
4. ***IS04.*** *Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software sobre la base de un conocimiento adecuado de las teorías, modelos y técnicas actuales.*

1.5. Presupuesto

1.5.1. Coste hardware

Hardware	Coste
Ordenador portátil	900€
TOTAL	**900€**

Cuadro 1.1: Coste hardware del proyecto

1.5.2. Coste personal

Personal	Nº Horas	Coste/Hora	Coste
Alumno	300	15€	4500€
Tutor	10	30€	300€
TOTAL			**4800€**

Cuadro 1.2: Coste personal del proyecto

1.5.3. Presupuesto final

Concepto	Coste
Coste del hardware	900€
Coste del personal	4800€
PRESUPUESTO TOTAL	**5700€**

Cuadro 1.3: Presupuesto total del proyecto

Capítulo 2

Estado del arte

2.1. Mapas interactivos

2.1.1. Marine Traffic

La primera y más importante tecnología a describir es la web de MarineTraffic[1]. Se trata del proveedor líder mundial de rastreo de barcos e inteligencia marítima. Permite al usuario monitorizar los movimientos de los barcos en base a datos recopilados de su red de estaciones receptoras AIS costeras, complementados por receptores satelitales.

Desde el punto de vista del usuario, es posible consultar en directo y de manera gratuita información de las embarcaciones, como se representa en la figura 2.1. La forma de representar las embarcaciones, viene dada en primer lugar por el tipo de embarcación, que decidirá el color del icono a representar. Adicionalmente, se tendrá en cuenta las dimensiones de la embarcación, para representar con mayor o menor tamaño el barco en el mapa.

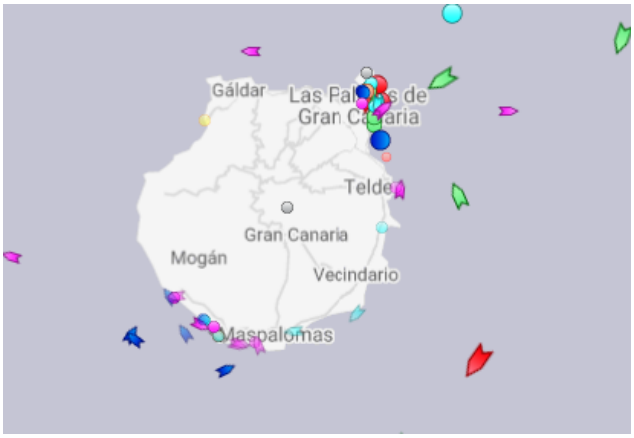
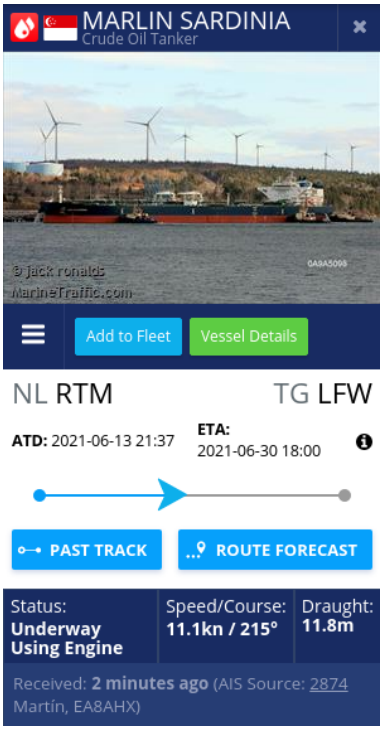


Ilustración 2.1: Representación de embarcaciones en MarineTraffic

[1]https://www.marinetraffic.com/

Además de visualizar en el mapa las embarcaciones, es posible consultar información de las mismas. Al mover el ratón sobre la embarcación, se mostrará un popup con cierta información de la embarcación. Haciendo click sobre el icono, se abrirá un menú, en el que se visualiza la siguiente información.

- ✔ Nombre
- ✔ Tipo
- ✔ Imagen
- ✔ País de procedencia
- ✔ Origen de la ruta
- ✔ Destino
- ✔ Estado en que se encuentra
- ✔ Velocidad/Rumbo
- ✔ Calado
- ✔ Fecha y origen de la información



(a) Menú de información detallada

(b) Popup de información

Ilustración 2.2: Consulta de información de embarcaciones en MarineTraffic

MarineTraffic ofrece también una variedad de filtros de embarcaciones, que permiten

filtrar por tipo de barco, capacidad, estatus, hora de detección etc. Además de la información que observamos a simple vista, es posible consultar información más detallada de la ruta de la embarcación. La web permite al usuario, como se representa en la figura 2.3, visualizar la ruta que ha llevado la embarcación a lo largo del tiempo. También, es posible especificar un segmento de tiempo para visualizar la ruta solo entre los instantes de tiempo que el usuario defina.

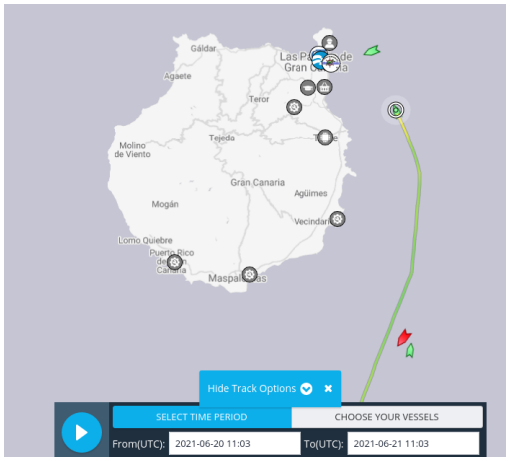
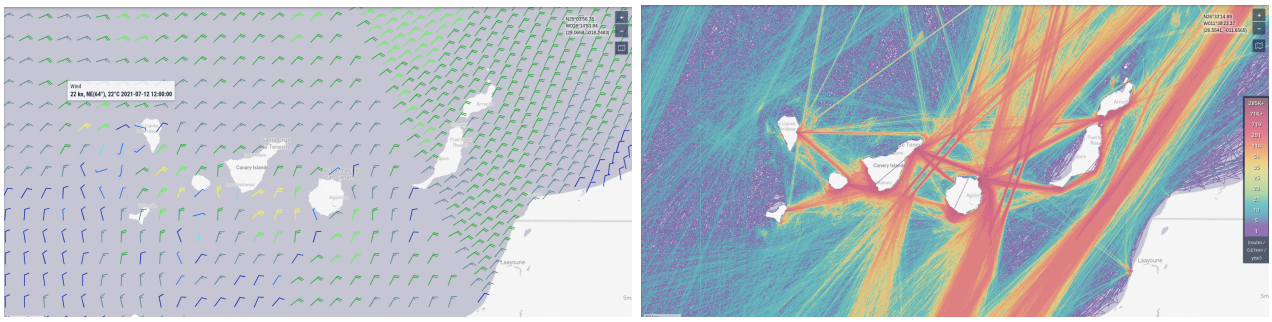


Ilustración 2.3: Consulta de recorrido de una embarcación en MarineTraffic



(a) Mapa de viento

(b) Mapa de densidad

Ilustración 2.4: Otras funcionalidades gratuitas de MarineTraffic

2.1.2. Windy

Windy[2] es un proveedor de datos meteorológicos especializado en la representación de mapas climatológicos animados. Actualmente es gratuita, y está disponible tanto para dispositivos móviles como para uso en la web. Windy destaca principalmente por la calidad gráfica de los mapas que presenta, además de por el diseño actual y funcional tanto de su aplicación como de su web, en la que nos centraremos en este apartado.

[2]https://www.windy.com/

Aunque Windy no es un proveedor de datos de embarcaciones, que es en lo que se centra este proyecto, se ha decido incluir en este apartado debido a que para el desarrollo de la web que conforma a este proyecto, se ha elegido la web de Windy como inspiración por su diseño.

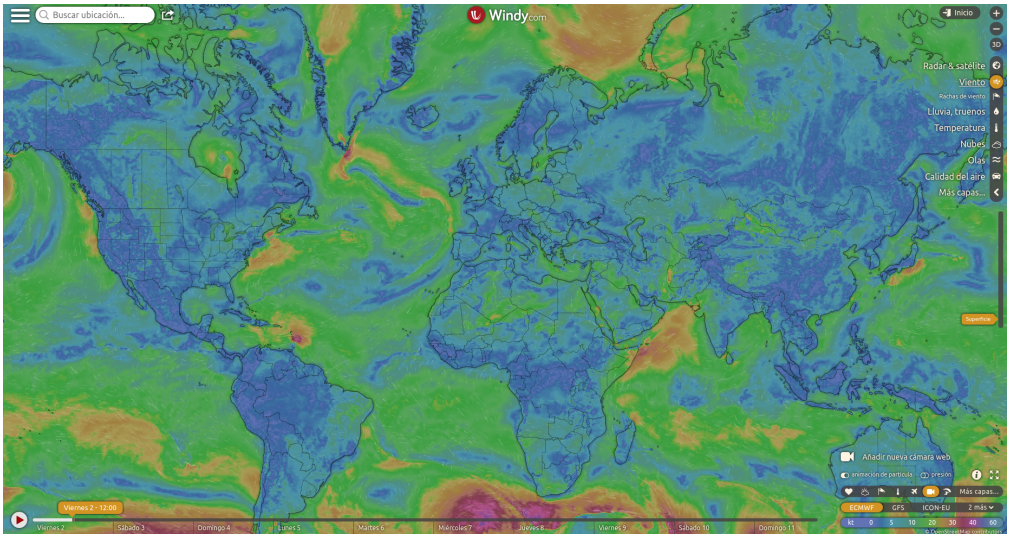


Ilustración 2.5: Vista general de la web de Windy

Como vemos en la figura 2.5, y centrándonos principalmente en la interfaz que muestra la web, vemos que se caracteriza por un diseño evitando el uso de bordes y centrándose en el uso de botones con iconos para acceder a las distintas funcionalidades. Con esto se consigue incrustar una gran variedad de funciones, sin perder visibilidad en el mapa.

Uno de los elementos de la interfaz más relevantes en el contexto de este proyecto es la barra de tiempo representada en la esquina inferior izquierda de la web.

Ilustración 2.6: Barra de control de tiempo web de Windy

Esta barra de tiempo, permite al usuario acceder a los registros meteorológicos tanto de instantes pasados como de previsiones futuras. Como se observa en la figura 2.6, cuando colocamos el cursor sobre la barra, aparece una burbuja de texto que señala el instante de tiempo que representa esa posición de la barra. Adicionalmente, cuando presionamos el botón de play, se visualizan los datos meteorológicos de manera continuada desde la posición en la que se encuentre la barra hasta su final.

Ilustración 2.7: Barra de botones web de Windy

Otro de los elementos a prestar atención, es la barra de botones que encontramos en la esquina superior derecha (figura 2.7). Esta permite al usuario utilizar las distintas herramientas de representación que dispone la web, como mapas de temperatura, precipitaciones, mareas etc. Este tipo de diseño es el que más se adapta a las tecnologías actuales y es una característica diferenciadora, que hace al usuario decantarse por un sistema sobre otro.

2.2. Frameworks de creación de mapas interactivos

2.2.1. Openlayers

OpenLayers[3] facilita la colocación de un mapa dinámico en cualquier página web. Puede mostrar mosaicos de mapas, datos vectoriales y marcadores cargados desde cualquier fuente. Ha sido desarrollado para promover el uso de información geográfica de todo tipo. Es completamente gratuito, publicado bajo la licencia BSD de 2 cláusulas (también conocida como FreeBSD).

Openlayers ha sido creado con la intención de profundizar en el uso de información geográfica de todo tipo. De esta forma, conseguimos una librería con funciones avanzadas pero a costa de obtener una dificultad mayor.

Las principales funcionalidades que ofrece esta librería son:

- ✔ Tiled layers(Capas de mosaico). Obtener mosaicos de OSM, Bing,MapBox y cualquier fuente XYZ.

[3] https://openlayers.org/

- ✔ Vector layers(Capas vectoriales). Permite procesar datos vectoriales de GeoJson, TopojSON, KML, GML, Mapbox y otros formatos
- ✔ Soporte móvil.
- ✔ Fácil personalización y ampliación. Posibilidad de personalizar los controles del mapa con CSS y utilizar bibliotecas de terceros para ampliar la funcionalidad.

2.2.2. Leaflet

Leaflet[4] es actualmente, la biblioteca Javascript de código abierto líder para mapas interactivos compatibles con dispositivos móviles. Contiene todas la funcionalidades de mapeo que la mayoría de los desarrolladores necesitan.

Leaflet tiene un diseño altamente focalizado en la simplicidad y la facilidad de uso. Funciona de manera eficiente en todas las principales plataformas móviles(Safari, Android, Chrome, Firefox y Win8) y de escritorio. Permite también la ampliación de funcionalidades con plugins externos.

Algunas de sus características más relevantes son las siguientes:

- ✔ Tile Layers
- ✔ Vector Layers
- ✔ Marcadores y Popups
- ✔ Funciones visules. Animaciones de zoom y panorámica, animaciones de popups, Diseño agradable de marcadores y controles del mapa etc.

2.2.3. Comparación

En la actualidad, las dos librerías encabezan el mercado[1], estando Leaflet en primera posición y Openlayers en segundo lugar. Trabajando con ambas librerías, nos damos cuenta de la característica diferenciadora que posee Leaflet con respecto a Openlayers, su simpleza.

```
var map = L.map('map').setView([51.505, -0.09], 13);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy;␣<a␣href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>␣contributors'
}).addTo(map);
```

Este es un ejemplo obtenido de la página principal de Leaflet, en el que se muestra el código para la integración de un mapa en una aplicación web. Para ello, solo es necesario la creación de un Map al que le asignamos una View, aportándole unas coordenadas, en las que se centrará la vista del mapa cuando abramos la web, y un valor de zoom. Además, definimos

[4] https://leafletjs.com/es

un mapa base, en este caso haciendo uso de la vista proporcionada por OpenStreetMap[5] y la añadimos al mapa.

```
const map = new Map({
  target: 'map',
  layers: [
    new TileLayer({
      source: new OSM()
    })
  ],
  view: new View({
    center: [0, 0],
    zoom: 0
  })
});
```

En este caso, se muestra la integración en la web de un mapa simple con la ayuda de la librería Openlayers. La diferencia entre las dos muestras de código, está en la cantidad de objetos utilizados, dos por parte de Leaflet(L.Map y L.tileLayer) vs cuatro por parte de Openlayers(ol.Map, ol.layerTile, ol.source.OSM y ol.View).

Otra de las ventajas de Leaflet con respecto a Openlayers, es que incorpora de manera nativa la posibilidad de crear de manera sencilla ventanas emergentes y asignarlas a marcadores.

```
L.marker([51.5, -0.09]).addTo(map)
    .bindPopup('A␣pretty␣CSS3␣popup.<br>␣Easily␣customizable.')
    .openPopup();
```

Visto este ejemplo, una tarea que es ciertamente trivial en Leaflet, se convierte en una tarea bastante tediosa cuando trabajamos con Openlayers. Esto es debido a que esta librería no trae por defecto la posibilidad de crear ventanas emergentes, lo que nos obliga a tener que crearlas desde el archivo HTML.

```
<div id="popup" class="ol-popup">
    <a  id="popup-closer" class="ol-popup-closer"></a>
    <div id="popup-content"></div>
</div>
```

Además de estar obligados a crear la ventana emergente, tenemos otro inconveniente, Openlayers no ofrece ninguna funcionalidad para directamente bindear un marcador con la ventana emergente que hemos creado. Para conseguirlo, debemos hacer uso del método **forEachFeatureAtPixel()**. Esta función que nos proporciona la librería, permite detectar las entidades que cruzan con un píxel en la ventana gráfica y ejecuta una llamada devolviendo la entidad que se ha cruzado.

Por lo tanto, para enlazar un marcador de una ventana emergente tenemos que hacer uso del método explicado en el apartado anterior para obtener en primera instancia la entidad(el

[5] https://www.openstreetmap.org/

marcador) y además, para que el usuario sea capaz de cerrar la ventana, debemos crearnos un método adicional.

En resumen, Leaflet es un producto bastante minimalista cuando lo comparamos con Openlayers, esto se demuestra cuando observamos el tamaño que ocupan, 140KB en el caso de Leaflet frente a 644KB en el caso de Openlayers. En la actualidad, ambas librerías son opciones completamente viables a la hora de plantear la integración de un mapa en una aplicación web, puesto que plantean características muy similares y ambas reciben cantidad de actualizaciones para seguir ampliando las funcionalidades que ofrecen y adaptarse a las tecnologías actuales.

En lo que concierne a este proyecto, se ha hecho uso de la librería Openlayers, puesto que el código desarrollado, será posterior a su presentación, integrado en el sistema que está siendo desarrollado por la empresa en la actualidad. Este sistema, está utilizando esta librería en concreto, por lo que, para facilitar en todo lo posible dicha integración, se ha decidido optar por Openlayers.

Capítulo 3

Tecnologías utilizadas

3.1. Angular

Angular es un framework de Javascript, creado con el propósito de desarrollar aplicaciones frontend actuales. Con Angular, se desarrollan aplicaciones con el estilo SPA(Single Page Application), lo que significa que en la misma página tenemos varias vistas que se irán intercambiando, produciendo el efecto de que tienes varias páginas[10].

Para el desarrollo en Angular, se utiliza generalmente Typescript, un superset de Javascript que añade principalmente tipos estáticos y objetos basados en clases. Typescript extiende la sintaxis de Javascript, por lo que cualquier código desarrollado en Javascript, debería funcionar en Typescript.

Para construir aplicaciones en Angular, se dispone de templates HTML, componentes que gestionan dichos templates mediantes propiedades variables y funciones, servicios que se encargan de gestionar la lógica de la aplicación y módulos que organizan los componentes y servicios[3].

3.2. Laravel

Laravel[11] es un framework de PHP. Es uno de los frameworks más utilizados y de mayor comunidad en internet. Tiene como filosofía el desarrollo de código PHP de forma elegante y simple, poniendo énfasis además en la facilidad de mantenimiento y la escalabilidad.

Gran parte de Laravel está formado por dependencias, especialmente de Symfony, lo que produce que el mantenimiento del framework dependa en gran medida del desarrollo de determinadas dependencias.

Utiliza como estilo arquitectónico MVC, promoviendo la separación de archivos archivos con un orden correcto definido, además de hacer énfasis en la separación de código por responsabilidades. Laravel destaca sobre sus competidores principalmente por su facilidad de uso, Artisan(sistema de comandos de consola), gran solidez y su completa variedad de documentación y tutoriales.

3.3. Bootstrap

Bootstrap[6] es una biblioteca de código abierto para el desarrollo frontend de aplicaciones web. Contiene plantillas personalizadas para la mayoría de elementos HTML, CSS y Javascript, como formularios, botones, menús etc.

Permite crear interfaces limpias y adaptables a la mayoría de dispositivos y pantallas, incluyendo dispositivos móviles. Destaca por su modularidad, componentes reutilizables y en uso de la librería jQuery para los componentes de Javascript, que proveen elementos adicionales además de añadir funcionalidad a los elementos ya existentes.

3.4. Angular Material

Angular Material es una librería de componentes orientada a desarrolladores en Angular. Ofrece, de manera similar a Bootstrap, componentes que permiten crear páginas web funcionales con un diseño atractivo y actual, manteniendo los principios del diseño web moderno, como portabilidad a navegadores e independencia del dispositivo[20].

3.5. MongoDB

MongoDB es una base de datos distribuida, basada en documentos y de uso general. Al ser documental, almacena los datos en forma de documentos tipo JSON, frente al tradicional modelo de filas y columnas.

Mantiene que trabajar con JSON, es una forma más natural y productiva de trabajar con datos, integrando esquemas dinámicos y flexibles para adaptarse a las nuevas tecnologías como la nube. Destaca por la posibilidad de realizar transacciones ACID completas y admite combinaciones en las consultas, que al ser también JSON, son más fáciles de programar y evitan la práctica de concatenar cadenas para generar consultas.

3.6. MySQL

MySQL[7] es un sistema de gestión de bases de datos relacional desarrollado por Oracle Corporation. Es considerado el sistema de bases de datos de código abierto más popular del mundo. Al ser relacional, utiliza tablas múltiples que se conectan entre si para organizar la información.

Dispone de una interfaz visual que ayuda a los desarrolladores administrar todos los datos de forma más sencilla. Está disponible en gran cantidad de plataformas y sistemas y ofrece gran seguridad.

3.7. Visual Studio Code

Visual Studio Code[8] es un editor decódigo fuente desarrollado por Microsoft. Es gatuito y de código abierto. Tiene varias opciones de personalización como cambio de tema, atajos de teclado y soporte de plugins. Además, incluye soporte para la depuración, control integrado de Git, refactorización, finalización inteligente de código etc.

Es compatible con los lenguajes de programación actuales, incluendo Python, Java, C++, JavaScript, PHP entre otros.

Capítulo 4

Metodología aplicada

4.1. Método Kanban

El método Kanban [21] es una las metodologías ágiles más utilizadas en el entorno empresarial para la gestión de proyectos. Su objetivo principal es brindar de un sistema de gestión visual que permita analizar el flujo de trabajo en un proyecto y el estado de las tareas y las funcionalidades que se quieren desarrollar, normalmente descritas como historias de usuario [5, 2].

4.1.1. Principios básicos del método Kanban

- ✔ **Calidad garantizada.** Kanban no premia la rapidez, sino el resultado final de las tareas, por lo que se busca en todo momento que el trabajo realizado salga bien a la primera, sin margen de error.
- ✔ **Mejora continua.** Este método trata de obtener pequeños y continuos cambios incrementales y evolutivos en el desarrollo de proyectos.
- ✔ **Flexibilidad.** Las tareas a realizar a continuación, vienen dadas por el *backlog*(tareas pendientes), con la posibilidad de priorizar otras tareas entrantes en caso de necesidad, de esta forma el equipo es capaz de reaccionar frente a tareas imprevistas.
- ✔ **Reducción del desperdicio.** Kanban opta por centrarse en una tarea en cada momento, reduciendo todo aquello considerado secundario, consiguiendo así un resultado de mayor calidad.

4.1.2. Las 6 prácticas de Kanban

Para implementar Kanban de forma eficiente, es necesario llevar a cabo unas prácticas identificadas por Favid J. Anderson(líder de pensamiento de la adopción del Lean/Kanban)[12].

1. **Definir el flujo de trabajo**
   Es necesaria la creación de un tablero, accesible por todos los miembros involucrados en el desarrollo del proyecto. Cada columna, será equivalente a un estado del flujo de desarrollo, para tener claro en todo momento en qué situación se encuentra cada tarea(Pendiente, en revisión, finalizada...).

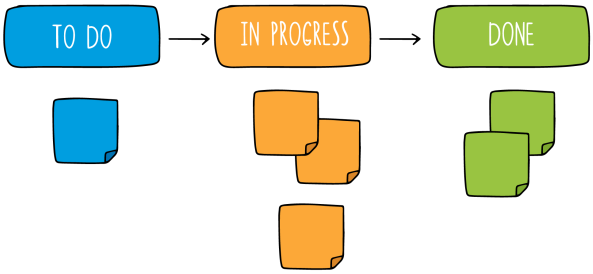


Ilustración 4.1: Esquema flujo de trabajo básico
[18]

2. **Visualizar las fases de producción**
   Kanban se basa en la división de una tarea en distintos pasos para obtener una mayor claridad en el trabajo a desarrollar. Cada tarea, se escribe en una tarjeta, en la que se especifica una descripción detallada del trabajo a realizar así como una estimación en horas, acompañado en ocasiones de las personas relacionadas en la tarea. Una vez definida la tarjeta, se añadirá al tablero en la columna a la que pertenezca.

3. **Eliminar interrupciones**
   La idea de Kanban es priorizar el trabajo que está en curso en lugar de empezar otras tareas. El cambio de enfoque al comenzar una tarea, puede llegar a producir pérdidas, por lo que esta práctica, se centra en limitar la cantidad de trabajar en proceso.

4. **Gestionar el flujo**
   Esto implica, identificar los trabajos que están estancados, para apoyar a las personas relacionadas con dicho trabajo y evitar los cuellos de botella. No vale la pena empezar trabajos nuevos sin primero asegurarse de que los miembros del equipo están listos para llevar a cabo la tarea en buenas condiciones.

5. **Implementar ciclos de retroalimentación**
   La filosofía Kanban promueve las reuniones regulares, normalmente diarias para sincronizar el equipo. Cada miembro comparte con los demás lo que va a hacer durante el día en concreto. Además, también se realizan reuniones periódicas con el Product owner para revisar el trabajo que se ha desarrollado.

6. **Mejora colaborativa**
   Para conseguir una mejora continua dentro de un equipo, es necesario controlar y estudiar el trabajo llevado realizado, para lograr que todos los miembros del mismo tengan claro los problemas que existen y entender como solventarlos.

4.1.3. Roles principales

Estos son los roles[9] que han tomado parte en el desarrollo de este proyecto. Si bien estos roles son propios de la metodología SCRUM, son ampliamente utilizados también en sistemas KANBAN.

1. **Service Request Manager (Product Owner).**
   Este rol representa la voz del cliente y, por tanto, la perspectiva de negocio. Su aportación reside en entender y trasmitir al resto de roles, cual es la forma más acertada para conseguir un producto que se acerque a las necesidades reales de los clientes y usuarios finales de un determinado entorno de negocio. Participa en las reuniones periódicas en las que proporcionará feedback del trabajo realizado por el equipo.

2. **Service Delivery Manager (Scrum Master).**
   Este rol se ocupa de ayudar al equipo a que las tareas salgan adelante. Su función principal es servir de facilitador para que los miembros del equipo tomen decisiones. También será el encargado de coordinar las reuniones diarias.

3. **Equipo de desarrollo**
   Son las personas encargadas de entregar el producto final. Participarán tanto en las reuniones diarias, como en las reuniones periódicas con el Product Owner para lograr alcanzar un resultado acorde a los objetivos iniciales.

4.1.4. Gestión del tablero Kanban

Para la gestión del tablero Kanban, se ha decidido hacer uso de la herramienta TRELLO para hacer un seguimiento diario de las funcionalidades, descritas como historias de usuario. Como se observa en la figura 4.2, en una primera instancia, se define la pila de producto, incluyendo en cada tarjeta la historia de usuario a desarrollar. Cada tarjeta contiene una breve descripción del trabajo, los denominados criterios de validación, que desglosan las tareas que serán necesarias realizar para completar el trabajo y una estimación del tiempo necesario para completarlo en horas.

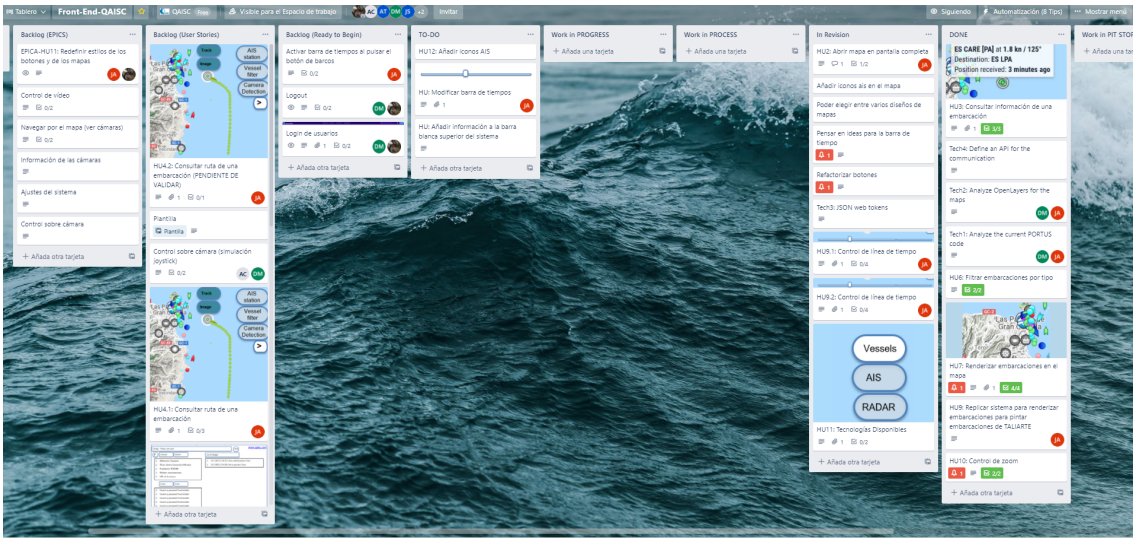


Ilustración 4.2: Tablero KANBAN gestionado con TRELLO.

(a) Ejemplo de descripción de una tarjeta

(b) Ejemplo de criterios de validación de una tarjeta

Ilustración 4.3: Ejemplo de historia de usuario utilizando una tarjeta TRELLO

Capítulo 5

Desarrollo

5.1. Análisis del documento de requisitos

Durante las fases iniciales del proyecto, el CEO de QAISC, quién ejerció de *Product Owner*, presentó un documento de requisitos que sirvió de guía inicial para las especificaciones que debían estar presentes en la aplicación web. Posteriormente, este trabajo se descompuso en colaboración con los tutores del TFG en las distintas historias de usuario necesarias para el correcto desarrollo del proyecto. En este apartado, exponemos brevemente los principales requisitos obtenidos tras el análisis del documento.

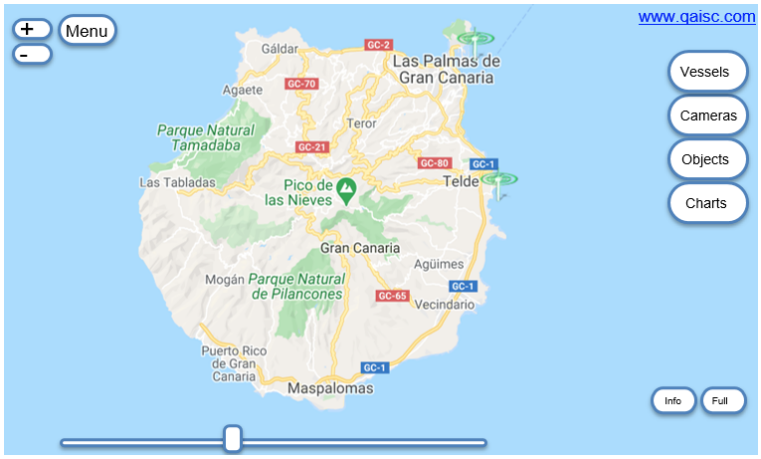


Ilustración 5.1: Ejemplo de aspecto general de la web provisto por Product Owner

El primero de los requisitos analizados es el diseño de la aplicación. Se espera una página de entrada con un aspecto sencillo, provisto de una serie de botones que encapsulen las funcionalidades principales del sistema. Entre ellas, se ha especificado que debe estar presente un botón, en el caso de la figura 5.1 representado por la palabra *Menú*, en el que el usuario podrá acceder a los ajustes principales del sistema.

En la parte derecha de la web, se espera encontrar una barra de botones que represente las tecnologías disponibles, como acceso a cámaras, embarcaciones y gráficas. Finalmente, se hace mención a la necesidad de una barra de tiempo, que se representará en la parte inferior de la página.

El documento, también contiene requisitos sobre la funcionalidad concreta de cada elemento. Al pulsar sobre *Vessels*, aparecerán los barcos, además de la opción de *Vessel filter*. Especifica también la necesidad de un elemento que permita la opción de visualizar la página en pantalla completa. Con respecto a la barra de tiempos, se establece que debe ser configurable para admitir diferentes intervalos de representación temporal y que este elemento permitirá, al elegir mediante la barra un tiempo concreto, visualizar el último set de datos existente de la fecha en cuestión.

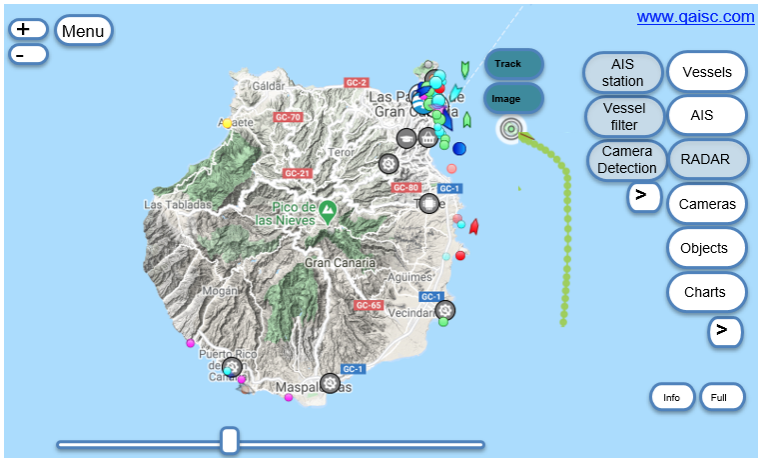


Ilustración 5.2: Ejemplo representación de embarcaciones provisto por Product Owner

La representación de las embarcaciones en la página vendrá dada por un determinado código de colores, (verde = cargo, rojo = tanker, azul = passenger, amarillo = high speed craft, azul claro = tugs and special craft, naranja = fishing, violeta = pleasure craft, cuadrado rosa = navigation aids, gris = unspecified ship). Mediante el *Vessel Filter*, se podrá filtrar la representación de embarcaciones, para mostrar únicamente las del tipo que sean relevantes para el usuario. Además al pulsar sobre una embarcación, el usuario podrá visualizar información referente a la misma, además de tener la posibilidad de consultar el recorrido seguido por la misma, como se representa en la figura 5.2 mediante puntos de color verde.

5.2. Sprint Zero: Estudio de recursos disponibles

5.2.1. Recepción de mensajes AIS

Como se explicó brevemente en la introducción del proyecto, el AIS en un sistema que funciona como emisora de VHF por la que embarcaciones transmiten de manera periódica sus datos de navegación, para así, que las demás embarcaciones sepan de su existencia y

así evitar colisiones. Existen, dos tipos de sistema AIS dependiendo del transpondedor que emite la información, clase A y clase B. La diferenciación de dichas clases[15] reside en las capacidades del AIS, tanto con respecto al alcance, como a la complejidad y el precio.

La información proveniente de los AIS de clase A siempre será prioritaria respecto a los de clase B y por tanto las embarcaciones con este sistema siempre estarán visibles para las demás embarcaciones. Esto se debe a un cuestión de seguridad, las embarcaciones de peso superior a 300 toneladas están obligadas a llevar consigo el sistema AIS clase A[14]. Estos buques de gran envergadura, utilizan un tipo de AIS de clase A denominado SOTDMA (Self-Organized TDMA) para evitar que se pierdan registros al colisionar mensajes de diferentes embarcaciones al mismo tiempo.

Para esta tarea, se hace uso de un algoritmo, que permite al transmisor AIS recibir en primer lugar los registros de las demás embarcaciones, para de esta manera, adaptar la transmisión al mismo patrón de estas. En caso de que existan demasiadas embarcaciones con sistemas de clase A en un área concreta y que esto acarreé una saturación del ancho de banda, el sistema se encargará de limitar el área de recepción para descartar los registros de las embarcaciones más lejanas al receptor.

Los sistemas AIS de clase B, se corresponden con embarcaciones de menor tamaño como por ejemplo veleros o yates. Estos, usan un sistema AIS de clase B de menor costo llamado CSTDMA (Carrier Sense TDMA) que transmiten información de manera menos frecuente a los sistemas de clase A. Además, la información perteneciente a esta clase de sistema, será mostrada a otras embarcaciones solo en caso de que exista espacio en el canal de transmisión, para ello escucha el canal durante algunos milisegundos para así definir si existe espacio suficiente para emitir el mensaje o no. Las embarcaciones con sistema de clase A solo tienen la capacidad de visualizar la posición de las de tipo B, pero no conocen su identidad y pueden elegir en su visor si mostrarlas o no.

Existen diferentes tipos de mensajes emitidos por los sistemas AIS[17], aunque en el contexto de este proyecto, no se entrará en detalle en aquellos que carecen de interés para la aplicación que buscamos lograr. En primer lugar, cabe mencionar los **mensajes de posición**, estos mensajes buscan informar sobre la posición física, además del movimiento que están llevando las embarcaciones en un instante de tiempo.

- ✔ Mensajes 1, 2 y 3 provenientes de sistemas AIS de clase A
- ✔ Mensaje 4 recibido desde estaciones base
- ✔ Mensaje 18 provenientes de sistemas AIS de clase B
- ✔ Mensaje 27 proveniente tanto de sistemas de clase A como B(solo para aplicaciones de alto rango)

A continuación, tenemos los **mensajes estáticos**, que muestran información respecto a las características normalmente invariables de las embarcaciones durante su ruta.

- ✔ Mensajes 5 proveniente de sistemas AIS de clase A
- ✔ Mensaje 24 proveniente de sistemas AIS de clase B

Existe un gran variedad adicional de tipos de mensajes, puesto que están numerados del 1 al 26, pero que, en este caso, no aportan información relevante. Los receptores AIS envían los paquetes de datos ASCII como un flujo de bytes a través de líneas seriales o USB, utilizando los formatos de datos NMEA 0183 o NMEA 2000(National Marine Electronics Association).

5.2.2. Tratamiento de los mensajes en el contexto de la empresa

Para el proceso de tratamiento de los datos provenientes de los sistemas AIS, debemos saber, que la información codificada en el fichero .nmea, viene dada por paquetes AIVDM/AIVDO, los paquetes AIVDM son informes de otros barcos y los paquetes AIVDO son informes de su propio barco. Este sería un paquete de datos AIVDM típico[19]:

$$!AIVDM, 1, 1“B, 177KQJ5000G?TO‘K > RA1wUbN0TKH, 0 * 5C$$

1. **!AIVDM**: Prefijo que identifica un paquete AIVDM.
2. **1**: Número de fragmentos en el mensaje que se está acumulando.
3. **1**: Número que identifica el fragmento de la oración
4. **vacío**: Id mensajes que contienen más de una oración.
5. **B**: Código que identifica el canal de radio utilizado en la transmisión.
6. **177KQJ5000G? TO‘K¿RA1wUbN0TKH**: Contiene la carga útil de donde obtendremos la información relevante de las embarcaciones.
7. **0**: Cantidad de bits necesarios para rellenar el mensaje hasta llegar a los 6 bits necesarios.

Adicionalmente, el fichero .nmea, facilita otro protocolo, que codifica el timestamp completo de la recepción de información por parte del AIS. Esta mensaje se denomina ZDA, y aquí podemos ver un ejemplo del mismo.

$$\$GPZDA, 14450{,}86, 23, 10, 2017, 00, 00 * 02X$$

1. **$GPZDA**: Este parámetro sirve como identificador del tipo de mensaje recibido.
2. **14450.86**: Representa la Fecha en formato UTC.
3. **23**: Día del mes, en este caso 23(entre 1 y 31).
4. **10**: Mes del año, en este caso octubre (entre 1 y 12).
5. **2017**: Año, en este caso 2017.
6. **00**: Diferencia de horas respecto a UTC.
7. **00**: Diferencia de minutos respecto a UTC.
8. ***02X**: *C*hecksum del mensaje ZDA.

En este momento, desde QAISC solo hacemos uso de los mensajes asociados a sistemas de clase A (1,2,3 y 5). Los datos relevantes se obtienen de decodificar el parámetro de carga útil proveniente del mensaje AIVDM. Una vez decodificado, se obtienen los siguientes campos[4]:

Cuadro 5.1: Parámetros mensajes tipo 1, 2, 3

Parámetro	Descripción
Message ID	Identifica si el mensaje es de tipo 1, 2 o 3
Repeat indicator	Indica las veces que ha sido repetido el mensaje
User ID	MMSI number (identificador del buque)
Navigational Status	Identifica en que estado de navegación se encuentra la embarcación(anclado, en tierra, son algunos ejemplos)
Rate of turn	Indica información sobre la maniobra que está ejerciendo la embarcación
SOG	Velocidad en knot steps(nudos)
Position accuracy	Indica cuán precisa es la posición emitida por el transmisor
Longitude	Longitud en la que se encuentra la embarcación
Latitude	Latitud en la que se encuentra la embarcación
COG	Rumbo percibido en tierra
True heading	Rumbo real de la embarcación en grados
Time stamp	Segundos UTC cuando se genero el mensaje
Special maneuvre indicator	Indica si la embarcación está realizando una maniobra especial
Spare	No utilizado
Number of bits	Indica los bits que ocupa el mensaje

Por otro lado, los mensajes de tipo 5 acarrean los siguientes parámetros:

Cuadro 5.2: Parámetros mensaje tipo 5

Parámetro	Descripción
Message ID	Identifica el tipo de mensaje(5 en este caso)
Repeat indicator	Indica cuantas veces ha sido repetido el mensaje
User ID	número MMSI (identificador del buque)
AIS version indicator	Indica la versión del AIS que transmite el mensaje
IMO number	número IMO(identificador de buque)
Call sign	Cadena de caracteres que ayuda a diferenciar embarcaciones de nombre similar
Name	Nombre de la embarcación
Type of ship	Identifica el tipo de embarcación
Dimension	Identifica las dimensiones físicas de la embarcación
Type of electronic position fixing device	Identifica el tipo de sistema de identificación de posición.
ETA	Tiempo estimado de llegada a destino
Maximum present static draught	Distancia desde el nivel del mar hasta el final del casco del barco
Destination	Destino de la embarcación
DTE	Indica la disponibilidad de la terminal de datos de la embarcación emisora
Spare	No utilizado
Number of bits	Indica los bits que ocupa el mensaje

El primer paso para el tratamiento de estos datos que se realiza desde la empresa, por parte de Sergio Ballines y Alejandro Cabrera, es acceder al fichero .nmea y comprobar si los datos obtenidos desde la llamada anterior son relevantes (mensajes 1,2, 3 o 5). En caso de que lo sean, se extraerán los datos de cada embarcación y se añadirán los campos necesarios a un objeto tipo json. Para ello, debemos saber que los datos relativos a la posición(mensajes 1,2 y 3), no tienen el timestamp completo (solo los segundos en UTC), por lo que en primera instancia, se lee el mensaje ZDA y se obtiene la fecha completa de recepción de la información. Una vez hecho esto, se decodifica el parámetro de carga útil del mensaje AIVDM y se obtienen los datos relevantes de posición para lograr formar el json correctamente. Finalmente, se guardan el json en una base de datos en MongoDB, que dispone actualmente de dos colecciones para preservar embarcaciones captadas por el AIS del dique Nelson Mandela y el AIS situado en Taliarte. Usamos Mongo debido a que su estructura hace muy sencillo preservar objetos de tipo json.

```
"_id": {
  "$oid": "609111c910e16e19df8ca9a0"
},
"timestamp": "1620119889",
"data": [
  {
    "type": 1,
    "repeat": 0,
    "mmsi": "311000084",
    "status": 0,
    "turn": 0,
    "speed": 0,
    "accuracy": true,
    "lon": -15.398871666666667,
    "lat": 28.157091666666666,
    "course": 79,
    "heading": 0,
    "second": 3,
    "maneuver": 0,
    "raim": true,
    "radio": 32922,
    "timestamp_GPZDA": "09:18:59.140000",
    "day_GPZDA": "4",
    "month_GPZDA": "5",
    "year_GPZDA": "2021",
    "timestamp": "1620119939.14"
  },
```


Ilustración 5.3: Fragmento de JSON de posiciones (mensaje tipo 1,2,3) de barcos

El JSON expuesto en la figura 5.3, es un fragmento de un registro de la base de datos de posiciones, captado por el AIS situado en el dique Nelson Mandela. Como se observa, cada registro contiene un id que lo identifica, así como un timestamp (representado en formato UNIX), que representa la fecha en la que el registro fue capturado por el AIS. El parámetro data, contiene el conjunto de embarcaciones que se desea representar en el mapa, aunque en la figura 5.3 solo se visualiza una embarcación, cada registro puede contener del orden de 80 embarcaciones a representar.

En el caso de que los datos json pertenezcan a un mensaje de tipo estático (tipo 5), el proceso se complica. En lugar de guardar todos los registros obtenidos como en el caso de la posición, aprovechando que estos datos son estáticos, guardamos la información de las embarcaciones en una base de datos relacional en MySQL. Esto es, debido a la necesidad de controlar qué embarcaciones han sido guardadas previamente y cuáles debemos añadir. El proceso por tanto, hará una llamada a la API, que se encargará de comprobar si la embarcación está registrada en la base de datos (mediante el mmsi) y en caso de que no lo esté, se guardará.

ship	
id	**int**
mmsi	varchar
imo	varchar
shipname	varchar
shiptype	varchar
length	float
breadth	float

Ilustración 5.4: Definición de la tabla ship

Prestando atención a la figura 5.4, se puede observar que aunque originalmente el mensaje tipo 5 contiene 17 parámetros pertenecientes a la información estática de las embarcaciones, la tabla que guardamos apenas contiene 7 columnas, de esta forma evitamos guardar información que no es relevante y conseguimos que sea más sencillo visualizar los datos.

5.2.3. Desarrollo de la pila de producto

Teniendo claro, tanto la metodología a llevar a cabo en el desarrollo, como las principales tecnologías a utilizar, así como los requisitos del proyecto, en este punto, se ha definido la pila de producto.

Id	Nombre	Descripción	Criterios de validación	Estimación
1	Login de usuarios	Como usuario web quiero poder logearme en el sitio web desde la pantalla de inicio de sesión para poder acceder a mi cuenta y disponer funcionalidades de adicionales.	-El botón mostrará el mensaje login cuando no se esté logueado. -El botón mostrará el nombre del usuario una confirma que se ha logueado.	4h

2	Abrir mapa en pantalla completa	Como usuario web quiero abrir el mapa en pantalla completa para poder visualizar y navegar mejor sobre el mapa	-Al pulsar el botón full del menú de botones de la derecha, se pondrá el mapa en pantalla completa. -Debemos confirmar que funciona con distintos navegadores (firefox, chrome, etc)	2h
3	Consultar información de una embarcación	Como usuario web quiero consultar la información de una embarcación para poder conocer detalles como mmsi, nombre y posición de la embarcación	-Al hacer hover con el ratón, se hará highlight al icono de la embarcación.-Al hacer hover sobre cualquier embarcación renderizada en el mapa, saldrá un popup con cierta información sobre la embarcación.-La información que debe estar representada en dicho popup será: Nombre de la embarcación, posición y cuando se produjo la última actualización.	10h
4	Consultar ruta de una embarcación	Como usuario web quiero consultar la ruta de una embarcación para poder conocer el recorrido que ha seguido a lo largo del tiempo	-Al hacer click sobre cierta embarcación saldrá un botón llamado consultar ruta.-Al acceder a consultar ruta, se renderizará en el mapa la ruta que ha seguido dicha embarcación como se puede ver en la imagen adjuntada.-La ruta se puede consultar con detalle con el ratón visualizando LAT y LON en esa posición y hora registrada	10h

5	Filtrar embarcaciones por tipo	Como usuario web quiero filtrar las embarcaciones del mapa por tipo para poder representar en el mapa solo aquellas que deseo.	-Al presionar el botón Vessel finder, saldrá un desplegable con el tipo los tipos de embarcaciones y se podrá seleccionar uno en concreto.-Una vez seleccionado, cuando presionemos el botón de renderizar embarcaciones solo se renderizarán las embarcaciones del tipo que tengamos marcado.	10h
6	Renderizar embarcaciones en el mapa	Como usuario web quiero renderizar las embarcaciones en el mapa para poder visualizar las embarcaciones capturadas por el sistema AIS y visualizar su posición en el mapa.	-Al presionar el botón AIS Station, se dibujarán las embarcaciones en el mapa.-El icono de cada embarcación cambiará conforme su tipo.- Se tendrá en cuenta el "Vessel finder", para renderizar solo algún tipo de barco en caso de que el usuario lo haya especificado.-Se tendrá en cuenta el valor de la línea de tiempo, para renderizar las embarcaciones en el instante de tiempo que desee el usuario.	15h
7	Replicar sistema AIS TALIARTE	Replicar el sistema de la HU5, para conseguir obtener registros de embarcaciones captados por el AIS situado en Taliarte.	Mismas que HU5	1h

8	Control de zoom	Como usuario web quiero aumentar y/o disminuir el nivel de zoom del mapa.	-Cuando se pulse en + se acercará una cierta cantidad de píxeles (determinar) y no hará nada una vez cubierto ese tope. -Cuando se pulse - se irá alejando hasta el tope del mapa (determinar con Luis) y no hará nada una vez cubierto ese tope.-Mediante la rueda del ratón, también se debe poder controlar el nivel de zoom	2h
9	Control de línea de tiempo	Como usuario web quiero adelantar y/o atrasar la línea de tiempo para poder visualizar registros de embarcaciones de instancias de tiempo anteriores.	-Cuando se mueva la barra de tiempo, se cambiará el instante de tiempo de representación de las embarcaciones dependiendo de la cantidad de píxeles (determinar) que se haya movido.-El intervalo de representación temporal de la barra de tiempos del mapa será configurable, por defecto se tomará la de dos días de tiempo pasado (futuro derecha) -La resolución temporal será configurable entre 1 minuto y 1 hora, tomamos 5 minutos por ahora.-El tamaño de la barra de tiempo será adaptable al tamaño del mapa (opcional)	10h
10	Añadir iconos AIS	Como usuario web quiero visualizar los iconos del AIS para poder visualizar la posición en la que se encuentran los receptores	-Al pulsar sobre el botón de mostrar embarcaciones, se mostrarán con un icono representativo, las posiciones de los receptores AIS	0.5h

11	Selección de mapa base	Como usuario web quiero poder seleccionar entre distintos tipos de mapa base para poder disponer de variedad de diseño a la hora de representar embarcaciones.	-En el menú de ajustes, habrá una opción de mapa base donde se podrá seleccionar entre distintos tipos de mapa.	3h

Cuadro 5.3: Pila de producto del proyecto

5.3. Etapas de desarrollo de la aplicación

Para el desarrollo de la aplicación web especificada en los objetivos del proyecto, se ha elegido la herramienta Angular para gestionar las vistas de la aplicación, además de la librería Openlayers para la integración del mapa en la web. En esta sección describimos el desarrollo de los principales componentes que conforman la aplicación.

5.3.1. Sprint 1: Creación del mapa

Para la integración del mapa en la web, como indicado en secciones anteriores, se ha usado la librería Openlayers. Se ha conseguido, mediante la creación de un componente, en el cual definimos un objeto de la clase *Map*[1]. A dicho objeto, le asignamos determinados parámetros para su funcionamiento, entre los cuales está *View*. Este objeto, admite parámetros como zoom, longitud y latitud, y causará, que en la web, se visualice el mapa centrado en una determinada localización en base a los valores aportados. En este caso se ha centrado en la parte norte de la isla de Gran Canaria.

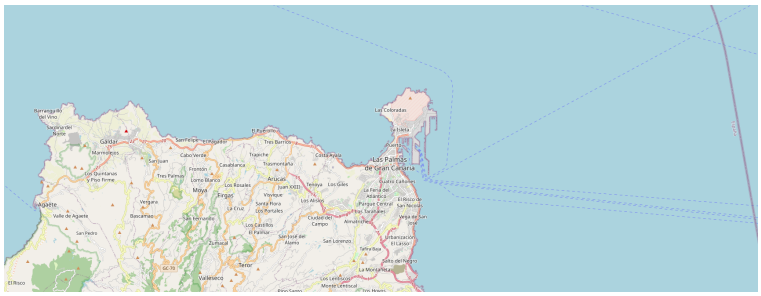
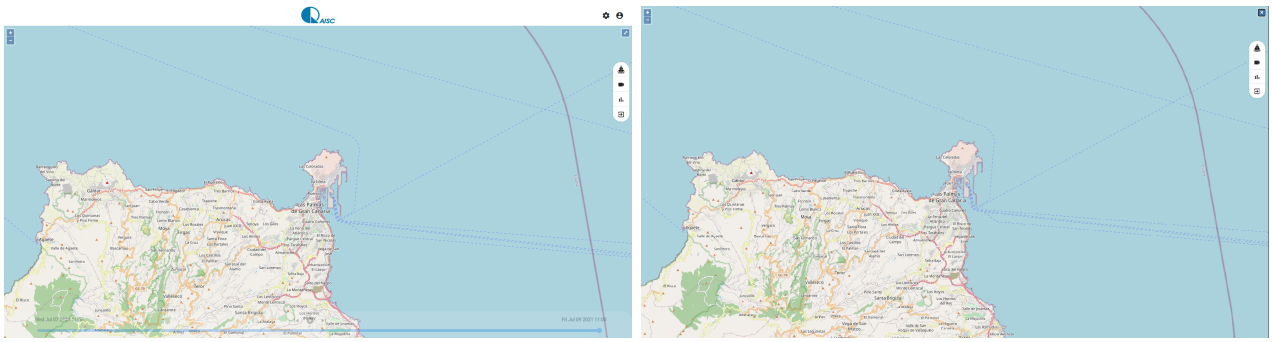


Ilustración 5.5: Vista del mapa al acceder a la aplicación

[1]https://openlayers.org/en/latest/apidoc/module-ol_Map-Map.html

Otro parámetro necesarios que se ha asignado al objeto *Map*, es la capa base del mapa. Por defecto, se mostrará un mapa provisto por OpenStreetMap, como se observa en la figura 5.5. Además, debemos añadir otros parámetros como el control de pantalla completa que nos ofrece OpenLayers y los popups que utilizaremos para incrustar información encima del mapa(overlays). Gracias al botón situado en la parte superior derecha de la web, el usuario podrá visualizar el contenido de la misma en pantalla completa.



(a) Vista de la web (b) Vista en pantalla completa

Ilustración 5.6: Vista normal y en pantalla completa de la web

Ilustración 5.7: Barra de navegación

Como se aprecia en la figura 5.7, se ha incluido en la parte superior, una barra de navegación con el logo de QAISC y algunos botones cuya utilidad se ha desarrollado durante los siguientes Sprints.

Otra función que nos ofrece Openlayers por defecto, es la integración de los controles de zoom, tanto mediante los botones situados en la esquina superior derecha, como utilizando la rueda del ratón.

5.3.2. Sprint 2: Dibujar las embarcaciones en el mapa

Como se había indicado en los objetivos, la funcionalidad más importante que se quiere conseguir, es la posibilidad de dibujar en el mapa las embarcaciones obtenidas en los registros captados por el AIS. Adicionalmente, la representación de las embarcaciones se desea llevar a cabo siguiendo un código de colores en el que cada color representará un tipo de barco determinado (verde = cargo, rojo = tanker, azul = passenger, amarillo = high speed craft, azul claro = tugs & special craft, naranja = fishing, violeta = pleasure craft, cuadrado rosa = navigation aids, gris = unspecified ship). En primer lugar, necesitamos desarrollar las peticiones necesarias a la API para obtener tanto la información estática de las embarcaciones, como la información de su recorrido actual. En segundo lugar, podemos desarrollar los métodos de la aplicación para representarlas.

Desde QAISC, se ha definido una API con las peticiones necesarias para el correcto funcionamiento de diferentes procesos. Se trata de una API construida en laravel, en la que se han definido ciertos métodos que han sido fundamentales para desarrollo de este TFG.

5.3.2.1. Petición a la API para obtener un registro de posiciones de embarcaciones

Este método busca obtener un registro de embarcaciones específico, aportando como parámetros a la petición, un instante de tiempo y el nombre de un radar AIS, con el objetivo de obtener como resultado, el registro obtenido por el AIS especificado, cuyo timestamp sea el más cercano al pasado como parámetro. Estos registros que se desean obtener, están preservados en MongoDb como se especificó anteriormente. Para obtenerlos desde laravel, accedemos al archivo *databases.php* y añadimos la conexión a Mongo de la siguiente forma:

```
'mongodb' => [
            'driver'   => 'mongodb',
            'host'     => env('MONGO_DB_HOST', '192.168.22.12'),
            'port'     => env('MONGO_DB_PORT', 27017),
            'database' => env('MONGO_DB_DATABASE'),
        ]
```

Una vez definida la conexión, se crea un nuevo controlador, donde tras realizar la validación de archivos, se define la conexión con Mongo y se eleje la colección en la que se realizará la búsqueda, en este caso las colecciones tienen el nombre ais. seguido del nombre del Ais(mandela/taliarte).

```
protected $connection = "mongodb";
protected $collection = "ais.'.$request->camera.'";
```

Haciendo uso del paquete de Laravel-MongoDb[2], se realiza un filtrado de los registros de la colección, obteniendo en primer lugar, aquellos en los que el timestamp sea menor o igual que el pasado como parámetro. Finalmente se ordenan y se obtiene el primer registro de la lista, que será transformado a formato json y pasado como resultado.

```
$success            =   $var::orderBy('timestamp', 'abs')
                                ->where('timestamp', "<=", $request->timestamp)
                                ->take(1)
                                ->get();

return response()->json(['success'=>$success], $this->successStatus);
```

5.3.2.2. Petición a la API para guardar en la base de datos la información estática de embarcaciones

Este método recibe como parámetro un mensaje tipo JSON con un conjunto de registros de información estática de barcos y tiene como objetivo recorrer dicho conjunto, obteniendo cada registro individual, que en este caso, contendrá información estática de una embarcación

[2]urlhttps://github.com/jenssegers/laravel-mongodb

concreta. En caso de que dicha embarcación no esté presente en la tabla de datos, cuya definición se representa en la ilustración 5.4, se guardará en dicha tabla.

Para ello, empezamos obteniendo un array con todos los mmsi de las embarcaciones pertenecientes a la tabla de la base de datos.

```
$mmsi = Ship::get(['mmsi'])->map(function($mmsi) {
            return $mmsi['mmsi'];
        })->toArray();
```

Y finalmente, recorremos individualmente las embarcaciones del json pasado como parámetro y comprobamos si el mmsi coincide con alguno de los presentes en el array. Si no está presente, guardamos en la tabla la información necesaria de la siguiente manera:

```
foreach($request->all() as $key => $response){
            if(in_array($key, $mmsi)!= 1){
                $ship     = new Ship();
                $ship->mmsi = $response['mmsi'];
                $ship->imo = $response['imo'];
                $ship->shipname = $response['shipname'];
                $ship->shiptype = $response['shiptype'];
                $ship->length = $response['length'];
                $ship->breadth = $response['breadth'];

                $ship->save();
            }
        }
```

5.3.2.3. Petición a la API para obtener toda la información estática de las embarcaciones guardadas en la base de datos

Este método, no requiere de ningún parámetro, y su único objetivo es el de devolver un json con la información de todas las embarcaciones preservadas en la tabla ship de la base de datos. Será de vital importancia a la hora consultar información de cualquiera de las embarcaciones que dibujemos en nuestro mapa.

Es muy sencillo de conseguir, ya que disponemos de la función all() de la que obtenemos un array con todos los elementos de la tabla en la que se utilice.

5.3.2.4. Gestión de la representación de barcos en la web

Esta funcionalidad, está controlada por un método llamado **printAis()**, que se encarga de llamar al método de la API que nos devolverá la información requerida para representar los barcos. En primer lugar llamará al método **getShipJson()**, que obtendrá el JSON con la información estática de las embarcaciones guardada en la tabla ship de la base de datos de QAISC(5.4). Este JSON, será tratado por otro método, que guardará toda la información en un array para llamarla siempre que sea necesario. En segundo lugar, llamará al método **getPositionJson()**, que se encarga de obtener el JSON con el registro de la posición de los barcos en un momento determinado. Este método será en nuestro caso llamado dos veces, una

para obtener las embarcaciones captadas por el AIS del dique Nelson Mandela y otra para obtener los registros del AIS de Taliarte. Al igual que el método anterior, una vez obtenido el JSON, se llamará a la función **loadShips()**, que descompondrá su información, la insertará en un array y llamará al método **addShip()**, para pintar el barco en el mapa.

La función **addShip()** tiene cierta complejidad, puesto que para dibujar en el mapa una embarcación debemos tener en cuenta diferentes elementos, como la rotación, el color a representar(según el tipo de barco), la velocidad, y la posición. El parámetro de rotación viene dado por el JSON de posición por un parámetro llamado heading. Este parámetro representa a qué orientación está apuntando la embarcación y viene dado en grados. Para dibujar sobre el mapa la embarcación utilizaremos el objeto de Openlayers Icon[3], este objeto tiene una propiedad llamada rotation dada en radianes, que utilizaremos para orientar correctamente la embarcación. Multiplicaremos heading por $\frac{\pi}{180}$ para obtener el valor en radianes.

Para representar los distintos tipos de embarcaciones, se ha utilizado los iconos mostrados a continuación(figura 5.8)

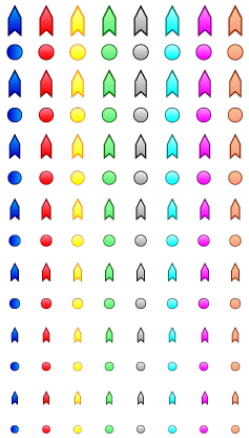


Ilustración 5.8: Iconos utilizados para representación de embarcaciones[16]

Utilizando las propiedades offset y size del objeto Icon, puedo obtener cualquier icono individual de los representados en la imagen. Teniendo en cuenta el tipo de embarcación, elegiré un color específico y teniendo en cuenta la velocidad del barco, utilizaré el icono redondeado para representar una embarcación quieta, o el icono triangular para indicar que la embarcación está en movimiento. Finalmente, obtenido el icono con los datos correctos, creo un objeto VectorLayer[4], le asigno la longitud y latitud dada por los datos obtenidos del JSON de posición, añado el icono a dicho objeto y le asigno a la propiedad nombre el mmsi del barco, para poder posteriormente saber que embarcación representa cada icono. Cuando añadimos al mapa todas las embarcaciones obtenemos el siguiente resultado(figura 5.9).

[3]https://openlayers.org/en/latest/apidoc/module-ol_style_Icon-Icon.html

[4]https://openlayers.org/en/latest/apidoc/module-ol_renderer_canvas_VectorLayer-CanvasVectorLayerRenderer.html

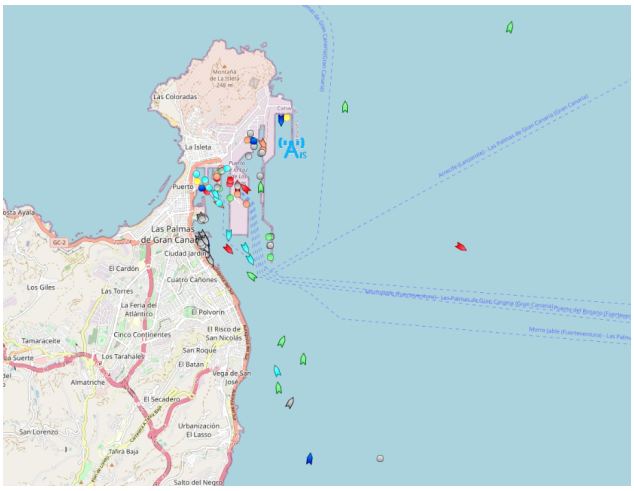


Ilustración 5.9: Resultado del mapa con las embarcaciones dibujadas

Como se aprecia en la figura 5.9, se ha añadido un icono para representar la posición donde se encuentra el receptor AIS. La figura 5.10 muestra más en detalle el citado icono.

Ilustración 5.10: Icono AIS desarrollado durante este proyecto

Para dibujar y borrar las embarcaciones del mapa, se hace uso del primer botón de la barra de botones representado en la figura 5.11. Al presionar sobre el mismo una vez los barcos están dibujados, las embarcaciones dibujadas se borraran del mapa, así como cualquier otro elemento presente en el mapa como popups o rutas.

Ilustración 5.11: Barra de botones del mapa

5.3.3. Sprint 3: Mostrar información de embarcaciones

El objetivo de esta aplicación, no solo consiste en la representación de las embarcaciones, sino también en la posibilidad de consultar información detallada de cada una de ellas. Para ello, ha sido necesario hacer uso de la información estática de los barcos obtenida gracias los métodos de la API descritos en las secciones anteriores (5.3.2.2 y 5.3.2.3).

5.3.3.1. Gestión de la representación de información de las embarcaciones en la web.

La representación de información de las embarcaciones en la web, es efectuada de dos maneras. En primer lugar, moviendo el cursor sobre las embarcaciones, sin hacer click, obtenemos un Popup con el mmsi, nombre, posición y fecha de recepción de las mismas. Cuando hacemos click, obtenemos un menú en el que obtenemos información mucho más detallada.

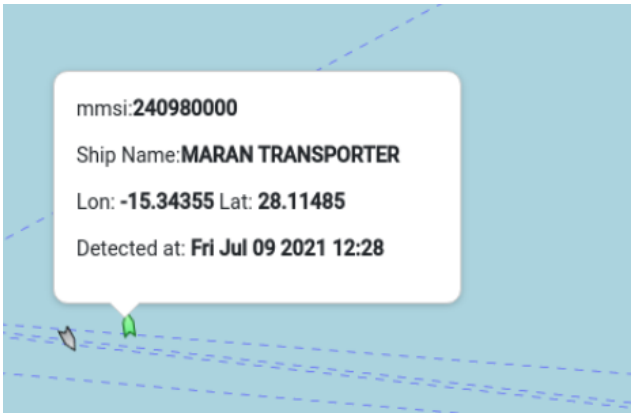


Ilustración 5.12: Popup al hacer hover sobre embarcación

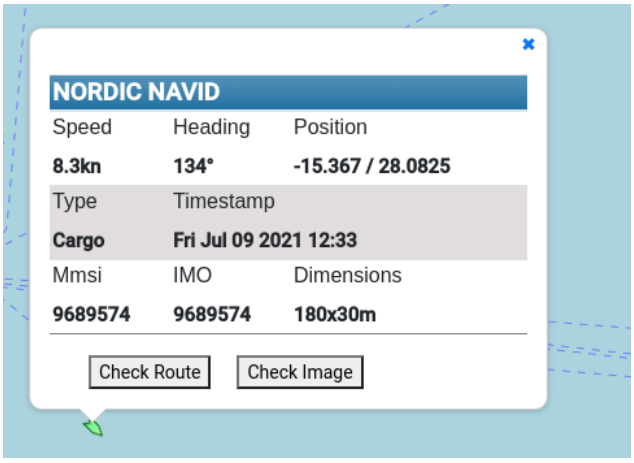


Ilustración 5.13: Popup al hacer click en embarcación

Para conseguir estos resultados, añadimos al mapa dos eventos, uno en pointermove y otro en click. Ambos eventos, hacen uso del método **forEachFeatureAtPixel()** al cual hice referencia en el capítulo 2 (sección 2.2.3). Gracias a esto, obtenemos el nombre del *VectorLayer* que representa la embarcación y mediante el método **getShip()** obtendré toda la información de la misma. Con la información del barco obtenida, se llama al método **createPopup()**, que obtiene el elemento HTML referente al Popup y mediante la propiedad innerHTML se encarga de escribir los datos del barco que hemos obtenido y dibujar el popup en la posición correcta.

5.3.4. Sprint 4: Dibujar el recorrido de una embarcación

Para lograr representar el recorrido de una embarcación, necesitamos desarrollar una petición a la API que nos permita obtener sus posiciones pasadas, para posteriormente, desarrollar en nuestra aplicación un método que nos permita dibujarlo.

5.3.4.1. Petición a la API para obtener la ruta que ha realizado una embarcación

El objetivo de este método, es obtener varios registros del mismo barco (pasado por parámetro), a lo largo del tiempo, para lograr reconstruir uniendo las posiciones de los distintos registros, la ruta que ha realizado la embarcación. Este método es similar al de recogida de posiciones de embarcaciones (5.3.2.1), ya que usaremos la misma base de datos y colecciones, por lo que, los dos primeros pasos serán idénticos.

```
$arrResponse            = $var::orderBy('timestamp', 'abs')
                                ->where('timestamp', "<=", $request->timestamp)
                                ->where('data.mmsi',"=", $request->ship)
                                ->take(60)
                                ->get();
```

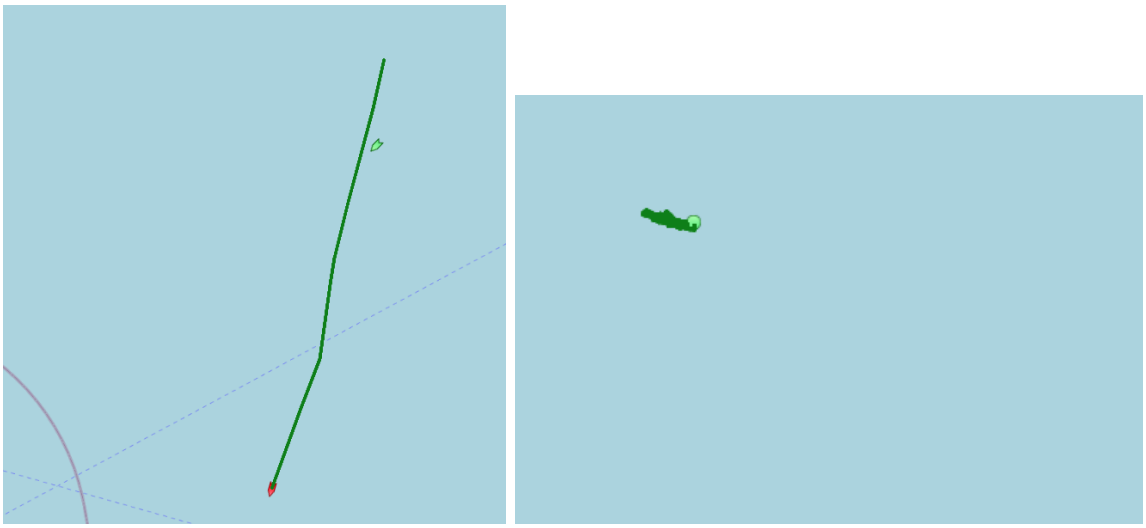
A diferencia del método que devuelve el registro de posición de todos las embarcaciones, en este caso debemos añadir la condición que el barco del registro sea uno en específico, que pasaremos como parámetro al método y además cogeremos 60 registros en lugar de uno. Y para finalizar, construiremos un json que contendrá todos los registros del mismo barco obtenidos por el filtrado.

```
foreach($arrResponse as $response){
            $arr                    = [];
            $arr['_id']             = $response['_id'];
            $arr['timestamp']       = $response['timestamp'];
            foreach($response['data'] as $ship){
                if($ship['mmsi']==$request->ship){
                    $arr['data']    = $ship;
                    break;
                }
            }
            $success[] = $arr;
        }

        return response()->json(['success'=>$success], $this->successStatus);
```

5.3.4.2. Gestión de la representación del recorrido de una embarcación en la web

Al presionar sobre el botón **Check Route** del popup descrito en el apartado anteriormente, tenemos como objetivo que se visualice el recorrido que ha realizado la embarcación. Este botón será un trigger de la función **printRoute()**, que al igual que con la representación de embarcaciones, se encargará de obtener el json con los registros de posicion del barco a lo largo del tiempo mediante la llamada al método de la API. El json, será descompuesto en el método **loadShipRoute()**, y se guardará en un array las posiciones de la embarcación a lo largo del tiempo(60 registros). Con la información de las posiciones ya guardadas, se llama a la función **showRoute()**, que utiliza el objeto de Openlayers *MultilineString*[5] para trazar una linea entre los distintos puntos de posición captados.



(a) Recorrido de una embarcación en movimiento

(b) Recorrido de un embarcación estacionaria

Ilustración 5.14: Recorrido de embarcaciones

El suceso representado en la figura 5.14b, se debe a que aunque la embarcación no esté en movimiento, debido al batir de las olas, siempre se producirá un ligero cambio en la posición de la embarcación, por lo que nunca estará completamente estática y en consecuencia, la posición captada por el AIS variará con el tiempo.

5.3.5. Sprint 5: Filtro de embarcaciones

El filtro de embarcaciones, permite al usuario elegir qué tipo de embarcaciones se dibujarán en el mapa. Cuando presionamos el botón para dibujar los barcos, aparecerá un botón adicional a la izquierda del mismo, que al presionarlo, abrirá un menú en el que se podrá elegir las embarcaciones a representar.

[5] https://openlayers.org/en/latest/apidoc/module-ol_geom_MultiLineString-MultiLineString.html

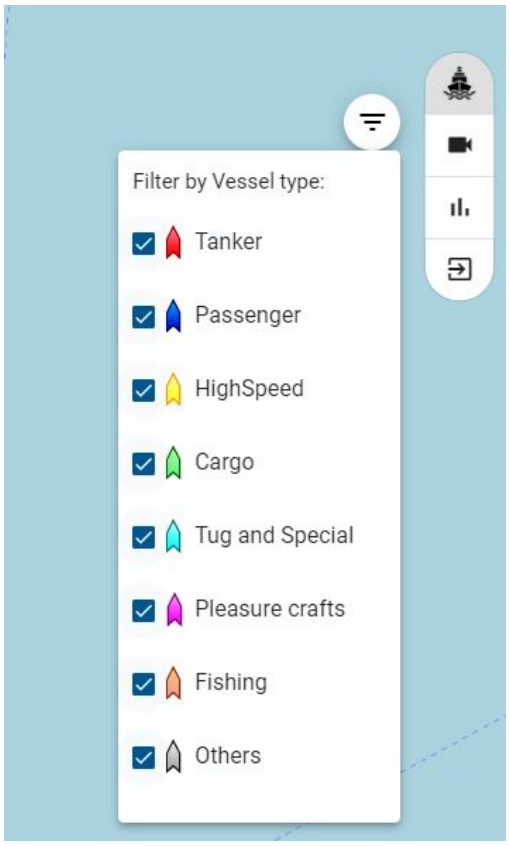


Ilustración 5.15: Filtro de selección de embarcaciones a dibujar

Al presionar sobre el último botón de la barra de botones representado en la parte derecha de la figura 5.15, haremos que el filtro deje de ser visible, al presionarlo de nuevo, volverá a estar disponible. Cada vez que el usuario selecciona o des selecciona un tipo de embarcación, se llama al método **filterEvent()**, que se encarga de insertar el tipo en cuestión en un array que funciona como lista negra. De esta manera, cuando llamamos al método **printAis()**, este se encargará, antes de dibujar la embarcación, de comprobar si el tipo de la misma no está presente en la lista negra para así no dibujarlo.

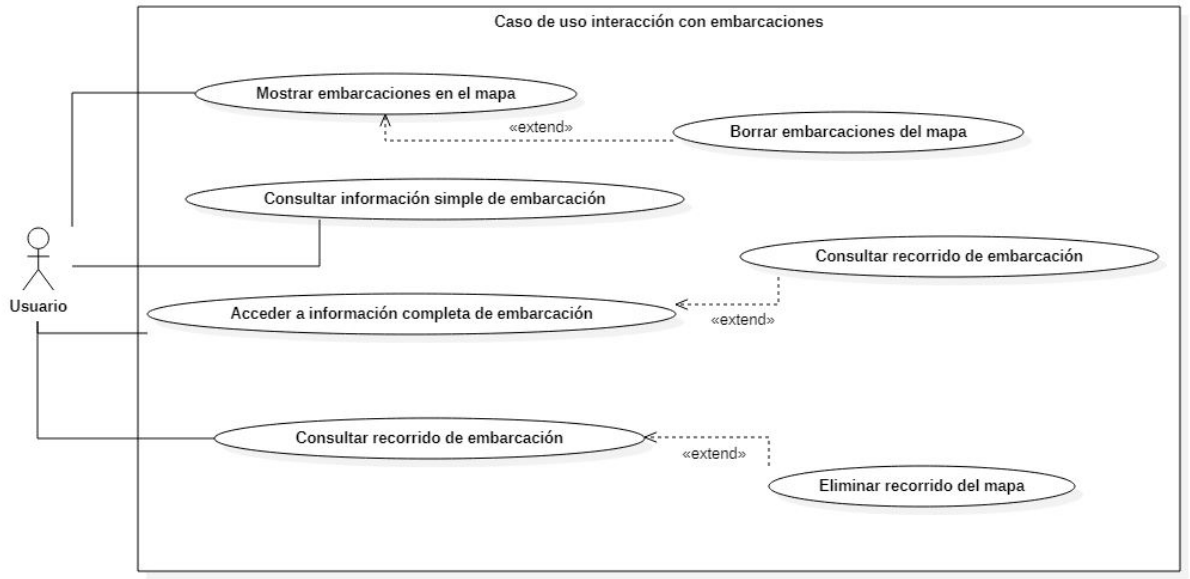
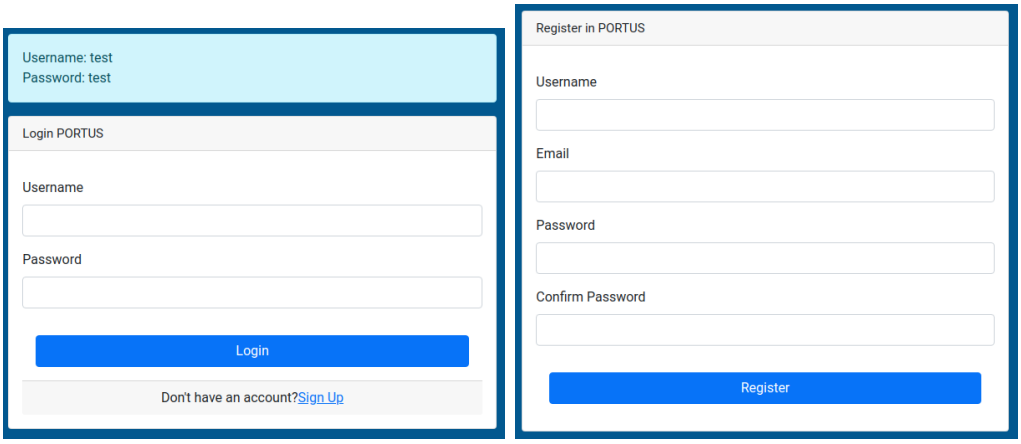


Ilustración 5.16: Diagrama caso de uso interacción con embarcaciones

5.3.6. Sprint 6: Sistema de autenticación y registro de usuarios

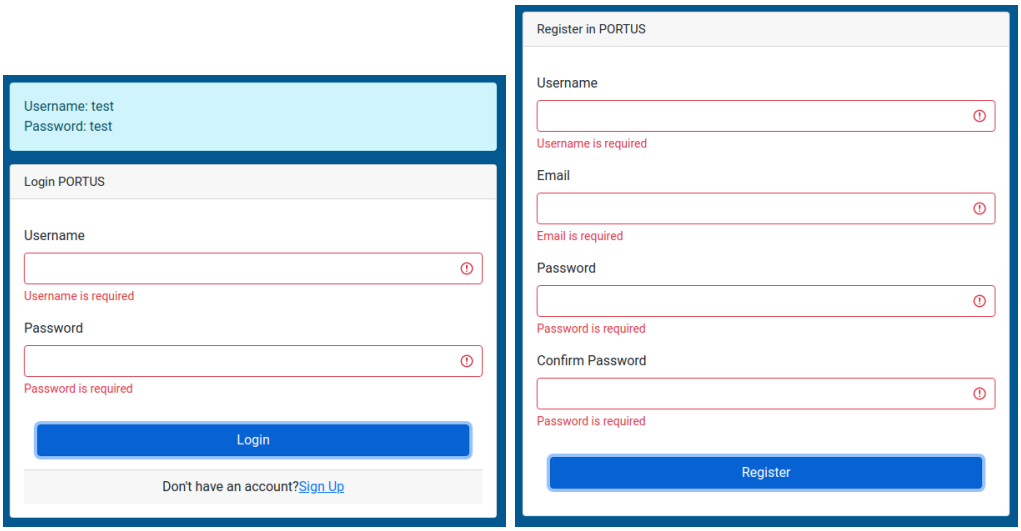
Para el sistema de autenticación, se ha desarrollado en el proyecto un componente denominado login, cuya función recae en la recogida de los datos que el usuario escribirá en el formulario de inicio de sesión, y el posterior envío de dicha información, que en este caso será el nombre de usuario y contraseña, a un servicio que se ha desarrollado llamado auth.service.

Este servicio, se encarga de realizar una llamada al endpoint de la API que se encarga de la autenticación de los datos, y que devuelve, en caso de que los datos enviados sean validados, la key asociada al usuario, que dará al mismo los permisos para utilizar las herramientas del mapa que se explicarán a continuación. Cuando se obtiene la respuesta de la API con la key asociada al usuario, se guarda en el **localStorage**, herramienta que nos permite almacenar datos en el navegador relativos al usuario y que persistan en el tiempo hasta que este decida cerrar la sesión. De esta forma, podemos guardar la key de la API de forma segura y evitar malas prácticas que pondrían en peligro la seguridad, como insertar la clave directamente en el código.



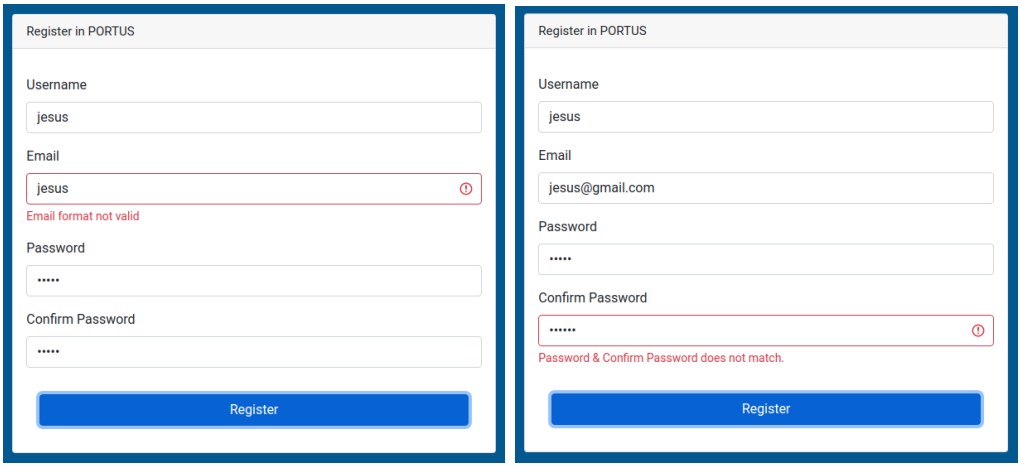
(a) Formulario Login (b) Formulario de registro

Ilustración 5.17: Formularios de login y registro



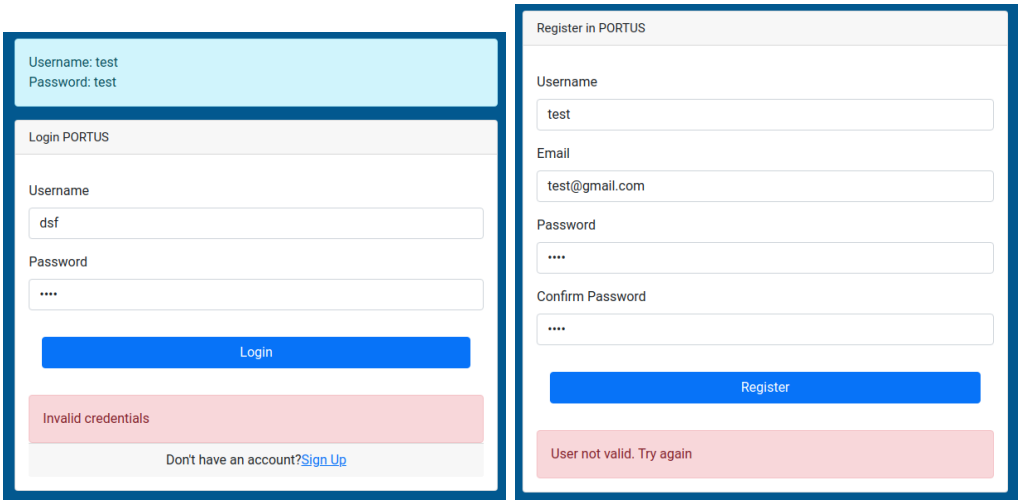
(a) Formulario Login (b) Formulario de registro

Ilustración 5.18: Campos obligatorios sin rellenar



(a) Formato incorrecto de email (b) Contraseñas no coincidentes

Ilustración 5.19: Error de validación de campos



(a) Error de inicio de sesión (b) Error de registro

Ilustración 5.20: Error de login y registro

Debido a que esta aplicación es solo un prototipo, se ha añadido una etiqueta con los datos de login en la parte superior del formulario para facilitar a las personas que deseen probar la aplicación el inicio en la misma. En caso de que se presione el botón de Login o registro sin acceder los parámetros necesarios, obtendremos un aviso (figura 5.18) y nos impedirá el login o registro hasta que se acceda la información necesaria. En la pantalla de registro, es necesario utilizar parámetros adecuados, como un email con formato válido, además de que ambas contraseñas coincidan, si alguna de estas condiciones no se cumple, obtendremos otro aviso (figura 5.19). En el caso de intentar acceder con unos datos incorrectos, obtendremos un error (figura 5.20), que indica que los datos introducidos no se relacionan a ningún usuario

registrado. En la parte inferior del formulario de inicio de sesión, tenemos el enlace que nos permitirá acceder a la vista de registro de usuarios.

Para desarrollar ambos componentes, se ha utilizado Bootstrap para añadir los elementos y colocarlos de forma adecuada, y angular para las condiciones de validación de campos.

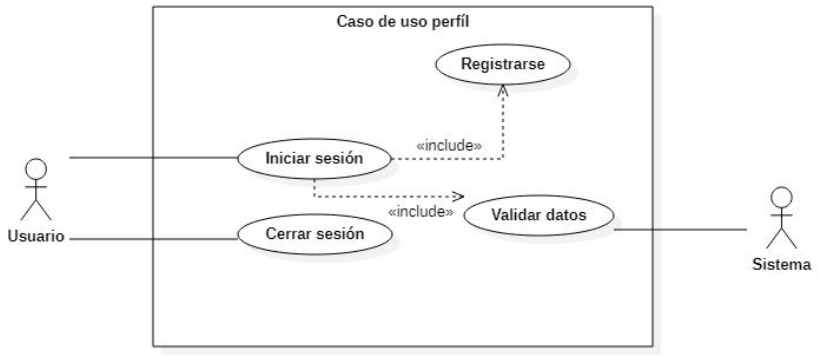


Ilustración 5.21: Diagrama caso de uso perfil de usuario

5.3.7. Sprint 7: Selector de mapa base

Esta funcionalidad, permite al usuario elegir entre un listado de mapas, que actuarán como base de la aplicación. Desde la barra de navegación situada en la parte superior de la web, en el botón de ajustes, se podrá desplegar un menú en el que se podrán elegir cinco tipos de mapas distintos. Por defecto, estará el mapa provisto por Open Street Map, y los cuatro siguientes, serán provistos por Bing Maps.

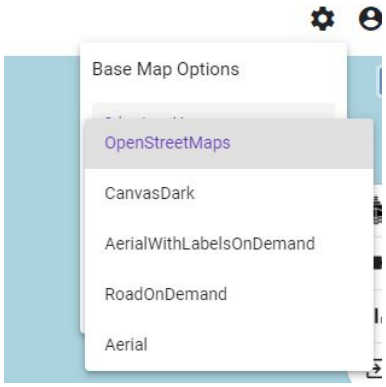


Ilustración 5.22: Selector de mapas base

Ilustración 5.23: Mapas base disponibles en la app

Para desarrollar este selector, se ha necesitado de la creación de un servicio que sirva como intermediario entre la barra de navegación y el mapa, debido a que son componentes separados y no pueden comunicarse entre si. Este servicio llamado message.service, define un objeto de tipo Subject, este objeto hereda ambos Observable y Observer, de esta forma, podemos utilizar este objeto para observar los cambios que se produzcan en el componente barra de navegación, y enviar un mensaje al mapa. El primer paso es definir en el servicio un método para enviar el mensaje entre componentes.

```
sendMessage(message: string) {
        this.subject.next({ text: message });
    }
```

El segundo paso, es definir un método en el servicio que permita devolver el objeto subject como Observable, para poder detectar cuando se produce un cambio en la selección de mapa. Para ello definimos este método en el servicio.

```
 onLayerMessage(): Observable<any> {
      return this.subject.asObservable();
    }
```

En el constructor del componente mapa(llamado map), utilizaremos el método anterior para detectar el cambio en la selección desde el componente barra de navegación y llamar al método encargado de realizar el correspondiente cambio de mapa base.

```
this.subscription = this.messageService.onLayerMessage().subscribe(message => {
      if (message) {
        this.changeBaseMapLayer(message.text)
```

```
  }
});
```

Desde la creación inicial del mapa, hemos insertado todos mapas base disponibles, y cuando se cambia de mapa desde el selector, el servicio manda un mensaje hasta el componente mapa con el nombre del mapa seleccionado. En el método **changeBaseMapLayer()** se recorrerá la lista de mapas base disponibles y utilizando la función **setLayerGroup()** provista por Openlayers y pasándole como parámetro el mapa base cuyo nombre concuerde con el recibido por mensaje, obtenemos el mapa deseado.

5.3.8. Sprint 8: Desarrollo de barra de tiempo

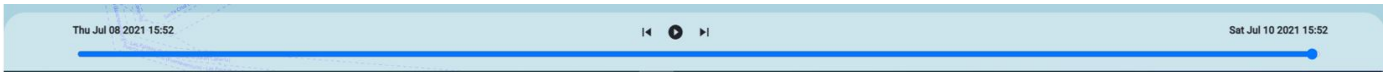


Ilustración 5.24: Vista de la barra de tiempo

Como se observa en la figura 5.24, se ha desarrollado un barra de selección de tiempo, que permite al usuario, acceder a registros captados en instantes de tiempo anteriores. Esta barra, está inicialmente invisible, y no se visualizará hasta que el usuario mueva el ratón por la parte inferior de la pantalla. Por defecto, el usuario podrá visualizar registros de embarcaciones con hasta un máximo de dos días de anterioridad, pero se ha desarrollado un selector utilizando el mismo método que el expuesto en el apartado del selector de mapa base, para elegir el intervalo máximo de tiempo que se desee en cada momento(2 días, 1 semana o un mes).

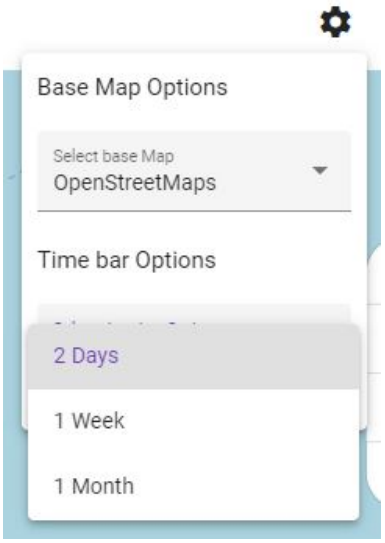


Ilustración 5.25: Selector de intervalo máximo de la barra de tiempo

Para facilitar el poder elegir el registro que se desee, se ha añadido una burbuja de texto,

que se mostrará al mover el cursor sobre la barra, indicando a que instante de tiempo equivale la posición de la barra a la que se está apuntando en ese momento.

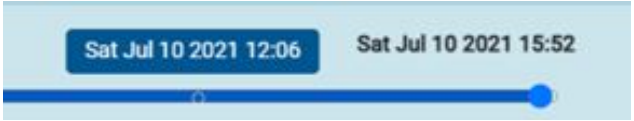
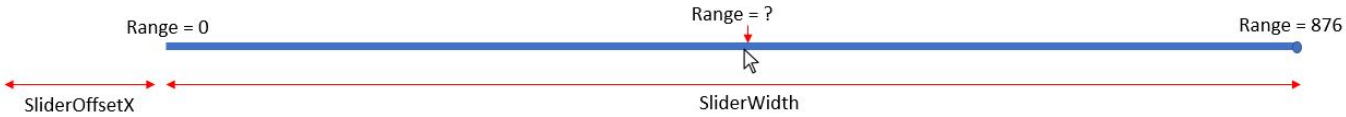


Ilustración 5.26: Burbuja de texto que indica el instante de tiempo al que equivale la posición de la barra a la que se apunta

La barra de tiempo, se trata de un elemento input de HTML de la clase range, que permite incorporar a la web una barra de selección con unos parámetros personalizados. Los registros de embarcaciones, se guardan en base de datos cada 5 minutos, por ello, ya que cada valor de la barra equivale a un registro, la barra tendrá tantos puntos como segmentos de 5 minutos. En el caso de elegir 2 días de intervalo máximo, la barra tendrá 576 puntos(12 segmentos de 5 minutos en una hora multiplicado por 48 horas), 2016 en el caso de 1 semana y 8064 en el caso de un mes de máximo.

El inconveniente que tiene este elemento HTML, es que no posee ningún evento de detecte el movimiento del cursor sobre el componente. Esto dificulta en gran medida calcular a que valor de la barra equivale el punto al que estamos apuntado con el cursor.



Para solventar este problema, en primer lugar, calculo la distancia que separa el comienzo en el eje X de la web, con el comienzo del elemento range(SliderOffsetX). Después, obtengo el tamaño total en píxeles del elemento range(SliderWidth). El siguiente paso, es calcular la posición en la que se encuentra el cursor respecto a al elemento range, (Posición del ratón en X + Número de píxeles que la página esta desplazada en X - SliderOffsetX).Con todo lo anterior, ya podemos calcular el valor del elemento que corresponde con la posición del cursor, para ello:

$$\text{SliderValueAtPos} = \frac{currentMouseXPos}{sliderWidth * rangeMax} \tag{5.1}$$

- ✔ $currentMouseXPos$: Posición del cursor en X respecto a la barra
- ✔ $sliderWidth$: Tamaño de la barra en píxeles
- ✔ $rangeMax$: Rango máximo de valores de la barra(576, 2016 o 8064)

✔ $SliderValueAtPos$: Valor de la barra que representa la posición del cursor

Teniendo el valor apuntado, podemos calcular cuanto tenemos que restarle a la fecha actual, para encontrar la fecha que representa el valor que hemos calculado, para ello:

$$\text{dif} = \frac{(rangeMax - SliderValueAtPos) * 5}{60} \tag{5.2}$$

✔ dif: Minutos que necesito restar a la hora actual para obtener la fecha a la que equivale el valor de la barra apuntado por el cursor.

✔ $rangeMax$: Rango máximo de valores de la barra(576, 2016 o 8064).

✔ $SliderValueAtPos$: Valor de la barra que representa la posición del cursor.

✔ Se multiplica por 5 debido a que cada punto de la barra equivale a 5 minutos.

Con todo lo anterior, solo falta obtener la fecha actual, restarle los minutos que hemos calculado, establecer la posición de la burbuja de texto en el lugar correcto y definir el texto de la burbuja, que equivaldrá a la fecha calculada después de restar el valor calculado.

Cuando el usuario elige un determinado registro, haciendo click sobre la barra, se llama a la función **timeInput()** que se encargará de llamar al método **updateAis()** encargado de refrescar el mapa con el registro de embarcaciones equivalente a la posición de la barra seleccionada por el usuario.

Debido a que cuando aumentamos el intervalo máximo de tiempo de la barra a un mes, es complicado acceder a registros específicos debido a la gran cantidad de puntos, se han desarrollado botones para acceder tanto al registro inmediatamente anterior(5 minutos en el pasado), como al inmediatamente siguiente(5 minuto sen el futuro). Además de un botón que permite reproducir automáticamente los registros.

(a) Botones adicionales de la barra de tiempo en pausa

(b) Botones adicionales de la barra de tiempo reproduciendo

Ilustración 5.27: Botones adicionales de la barra de tiempo

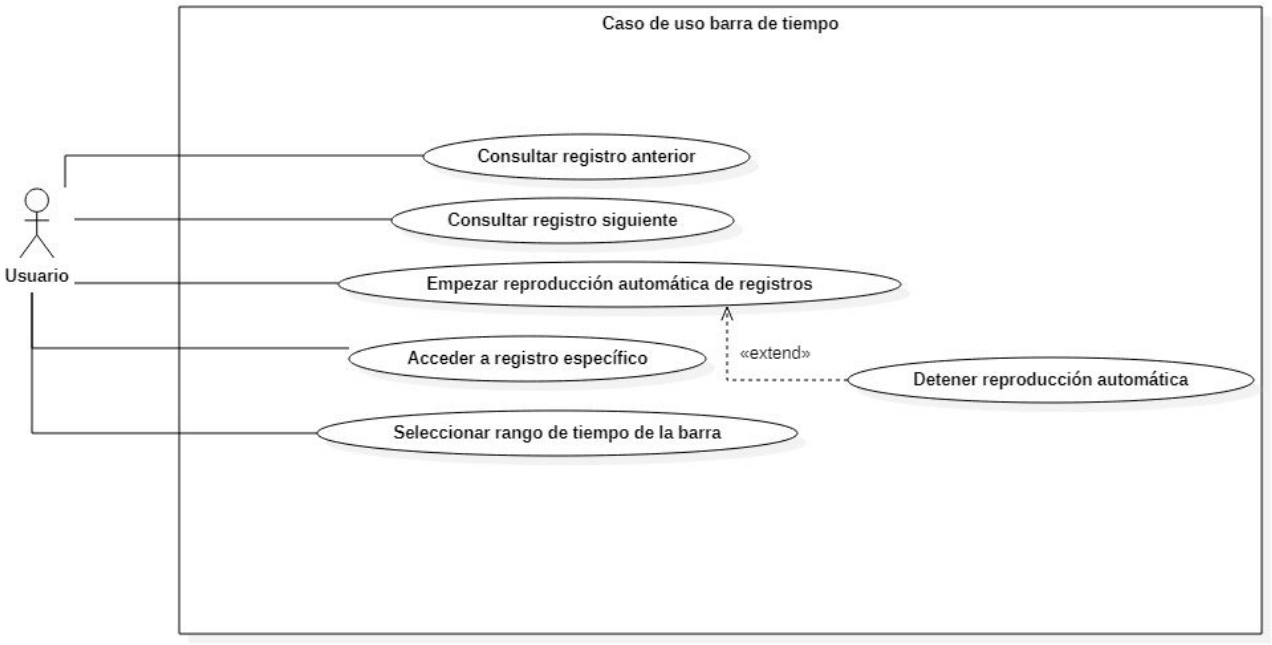


Ilustración 5.28: Diagrama caso de uso barra de tiempo

Capítulo 6

Conclusiones y trabajo futuro

En este Trabajo Fin de Grado se ha desarrollado una aplicación que permite el seguimiento de embarcaciones a partir de la información en tiempo real capturada por dispositivos AIS implantados en zonas costeras canarias, tales como el Puerto de la Luz y de Las Palmas y Taliarte. Esta información se ofrece mediante un mapa que permite distintas funcionalidades, entre las que destaca, una barra de tiempo que nos permite analizar la evolución del entorno marítimo-portuario en intervalos de minutos, días e incluso meses.

Describimos todo el proceso realizado mediante Kanban e Historias de usuario para, desde una fase inicial de visión de producto a partir de los requisitos especificados por el CEO de una empresa, trasladar esa visión a un producto que ha ido mejorando y añadiendo funcionalidades a lo largo de distintas iteraciones o Sprints.

Se ha conseguido aplicar los conceptos de desarrollo ágil, participando en numerosas reuniones con el CEO de QAISC para revisar el trabajo realizado y conseguir una mejora continua. Además, se ha trabajado en conjunto con varias personas de la empresa, reforzando el trabajo en equipo y las buenas prácticas con los tutores del trabajo y el equipo de desarrollo de la empresa.

Aunque se han conseguido en gran medida los objetivos iniciales que se habían planteado, aun existen apartados en los que se puede mejorar y ampliar la aplicación. En este apartado, voy a plantear ciertas propuestas en las que me gustaría continuar trabajando en relación al proyecto expuesto:

- ✔ Implementar una nueva funcionalidad, que permita al usuario visualizar imágenes de las embarcaciones captadas por las cámaras de QAISC. Mediante un script, las cámaras tomarán imágenes de las embarcaciones y estas se guardarán en OpenStackSwift para posteriormente ser representadas en la web. Para ello, sería necesario establecer en la base de datos una relación de la tabla ship representada en la figura 5.4, con una tabla file, que representa un fichero. De esta forma, conseguiríamos relacionar una embarcación con su imagen.

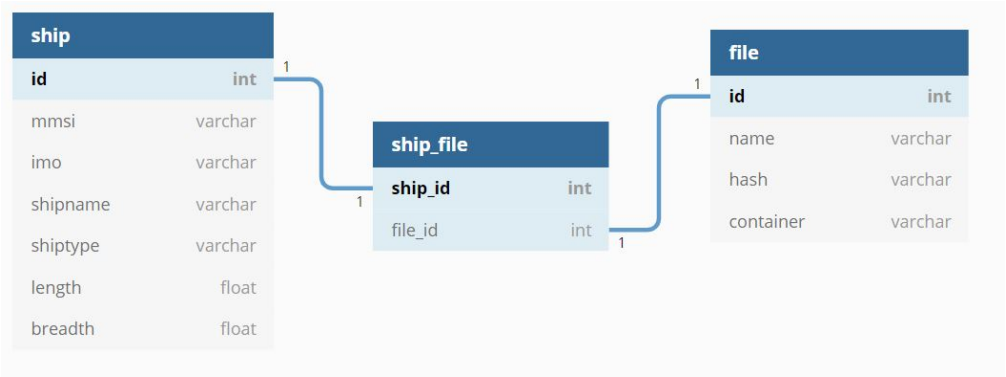


Ilustración 6.1: Relación futura de la tabla ship con tabla file

- ✔ Poder visualizar en el mapa embarcaciones pequeñas, registradas por dispositivos AIS clase B.
- ✔ Desarrollar algunas funcionalidades a futuro, que permitirán al usuario visualizar gráficas referentes a datos provistos por la empresa, así como cartas náuticas etc.
- ✔ Añadir más opciones de personalización al sistema, que permitan personalizar aspectos como la zona horaria, idioma etc.
- ✔ Integrar esta aplicación en PORTUS (aplicación web desarrollada por miembros del grupo QUALITAS al que pertenece QAISC).

Bibliografía

[1] Comparing Mapbox, OpenLayers and Leaflet. https://bachasoftware.com/custom-software-development/, 2019. [Online; accessed 13-July-2021].

[2] Juan Palacio Miguel Ángel Sobrino Rubén Álvarez y Verónica Rivas. Alexander Menzinsky, Gertrudis López. Historias de Usuario. https://scrummanager.net/files/scrum_manager_historias_usuario.pdf, 2020.

[3] Jorge Cano. Angular: Mucho más que un framework. https://sg.com.mx/revista/56/angular, 2018. [Online; accessed 13-July-2021].

[4] U.S. Coast Guard Navigation Center. Class A AIS Position record(Messages 1,2,3). https://www.navcen.uscg.gov/?pageName=AISMessagesA, 2008. [Online; accessed 13-July-2021].

[5] Mike Cohn. User Stories Applied: For Agile Software Development, 2004.

[6] colaboradores de Wikipedia. Bootstrap (framework). https://es.wikipedia.org/w/index.php?title=Bootstrap_(framework)&oldid=135824302, 2021. [Online; accessed 13-July-2021].

[7] colaboradores de Wikipedia. MySQL. https://es.wikipedia.org/w/index.php?title=MySQL&oldid=136660516, 2021. [Online; accessed 13-July-2021].

[8] colaboradores de Wikipedia. Visual Studio Code. https://es.wikipedia.org/w/index.php?title=Visual_Studio_Code&oldid=136990285, 2021. [Online; accessed 13-July-2021].

[9] Javier Martín de Agar Tirado. Los grandes desconocido del Método Kanban: roles y eventos. https://mamaqueesscrum.com/2019/05/30/los-grandes-desconocido-del-metodo-kanban-roles-y-eventos/, 2019. [Online; accessed 13-July-2021].

[10] DesarrolloWeb. Manual de Angular. https://desarrolloweb.com/home/angular, 2019. [Online; accessed 13-July-2021].

[11] DesarrolloWeb. Manual de Laravel. https://desarrolloweb.com/home/laravel, 2019. [Online; accessed 13-July-2021].

[12] Kanbanize. Qué es Kanban: Definición, Características y Ventajas. https://kanbanize.com/es/recursos-de-kanban/primeros-pasos/que-es-kanban, 2018. [Online; accessed 13-July-2021].

[13] Tyler Lacoma. ¿Qué es una cámara PTZ? https://techlandia.com/camara-ptz-sobre_76400/, 2012. [Online; accessed 13-July-2021].

[14] Localizatodo.com. Introducción al AIS. http://www.localizatodo.com/html/pdf/ais.pdf, 2008. [Online; accessed 13-July-2021].

[15] Ocean Time Marine. Class A And Class B Automatic Identification System (AIS). https://www.oceantimemarine.com/class-a-and-class-b-automatic-identification-system-ais/, 2019. [Online; accessed 13-July-2021].

[16] MarineTraffic. MarineTraffic: Global Ship Tracking Intelligence — AIS Marine Traffic. https://www.marinetraffic.com/, 2021. [Online; accessed 14-July-2021].

[17] Spire Sense Maritime. Available AIS Message Types. https://faq.spire.com/available-ais-message-types, 2018. [Online; accessed 13-July-2021].

[18] Patricia Parreira. Métricas Kanban: qué son y cómo te ayudan a optimizar tus procesos. https://netmind.net/es/metricas-kanban-que-son/, 2020. [Online; accessed 13-July-2021].

[19] Eric S. Raymond. Decodificación de protocolo AIVDM / AIVDO. https://gpsd.gitlab.io/gpsd/AIVDM.html, 2021. [Online; accessed 14-July-2021].

[20] Tutorialspoint. Angular Material Tutorial. https://www.tutorialspoint.com/angular_material/index.htm, 2016. [Online; accessed 13-July-2021].

[21] Daniel S. Vacanti. Kanban Guides. https://kanbanguides.org/, 2020. [Online; accessed 13-July-2021].