



## Trabajo de Fin de Grado

---

# Sea Calm, planificador de buques en infraestructuras portuarias

*Sea Calm, vessel planner in port infrastructures*

Bruno Lorenzo Arroyo Pedraza

---

La Laguna, 13 de julio de 2023

D. **Christopher Expósito Izquierdo**, con N.I.F. 78.851.649-J profesor Ayudante Doctor adscrito al Departamento Ingeniería Informática y de Sistemas de la Universidad de La Laguna como tutor

D. **Israel López Plata**, con N.I.F. 42.193.801-W profesor Ayudante Doctor adscrito al Departamento Ingeniería Informática y de Sistemas de la Universidad de La Laguna como cotutor

## **C E R T I F I C A N**

Que la presente memoria titulada:

*"Sea Calm, planificador de buques en infraestructuras portuarias"*

ha sido realizada bajo su dirección por Don **Bruno Lorenzo Arroyo Pedraza**, con N.I.F. 79.358.495-P.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 13 de julio de 2023

# Agradecimientos

Agradecer profundamente a los profesores Christopher Expósito Izquierdo e Israel López Plata por su paciencia, tiempo, guía y dedicación. Gracias por haberme dedicado las herramientas y conocimientos necesarios para sacar adelante el presente trabajo, no podría tener mejores tutores.

También agradecer a toda mi familia, tanto la que se encuentra a mi lado como la que se encuentra lejos, por ser uno de mis pilares fundamentales y por todo su apoyo para así conseguir llegar a donde estoy ahora.

Por último, agradecer a todos mis amigos y compañeros por brindarme momentos de descanso y risas para así desconectar en los momentos que lo he necesitado.

# Licencia



©Esta obra está bajo una licencia de Creative Commons Reconocimiento-NoComercial-SinObraDerivada 4.0 Internacional.

## Resumen

*Una de las mayores problemáticas que existe en el ámbito del comercio internacional es que los buques mercantes lleguen a destiempo a las infraestructuras portuarias. La llegada a destiempo a los puertos se debe a diversos factores, pero termina desembocando en la existencia de congestión en el tráfico marítimo, lo que provoca a su vez grandes pérdidas económicas.*

*El trabajo de fin de grado Sea Calm, planificador de buques en infraestructuras portuarias, consiste en el diseño y desarrollo de una aplicación software mediante una aproximación full- stack. La aplicación se encuentra principalmente enfocada en la gestión y planificación de la llegada de los buques mercantiles a las infraestructuras portuarias, con el objetivo de reducir el problema de la congestión marítima.*

*El desarrollo de la aplicación se ha realizado mediante la implementación de la arquitectura hexagonal tanto en el front-end como en el back-end. La comunicación entre front-end y back-end se realiza con un sistema API REST permitiendo abstraer la lógica de la aplicación de la interfaz del usuario. El front-end se implementa con Vue.js 3 junto con Vuetify para tener un diseño Material Design y el back-end se programa en Java, utilizando Spring Boot. Para la obtención de los buques mercantiles se utiliza una conexión WebSocket y la persistencia de datos se realiza con MongoDB. Cada componente o parte de la aplicación se encuentra encapsulada en contenedores Docker para su posterior despliegue.*

**Palabras clave:** Software, Buques, Puertos, Arquitectura hexagonal, Aplicación Web, Full-stack.

## **Abstract**

One of the biggest problems in international trade is the mistime arrival of merchant ships at port infrastructures. The mistime arrival can be due to several factors, but it ends up leading to the existence of maritime traffic congestion, causing great economic losses.

The end-of-degree project Sea Calm, vessel planner in port infrastructures, consists of the design and development of a software application using a full stack approach. The application is mainly focused on the management and planning of the arrival of merchant ships to port infrastructures, reducing the problem of maritime congestion.

The development of the application has been carried out by implementing the hexagonal architecture both in the front-end and back-end. Communication between front-end and back-end is done using a REST API system, allowing the application logic to be abstracted from the user interface. The front-end has been developed using Vue.js 3 along with Vuetify to achieve a Material Design layout, while the back-end has been programmed in Java using Spring Boot. Merchant ships are obtained through WebSocket, and data persistence is achieved using the MongoDB NoSQL database. Each component or part of the application is encapsulated in Docker containers for subsequent deployment.

**Keywords:** *Software, Vessels, Ports, Hexagonal Architecture, Web Application, Full-stack.*

# Índice general

<b>1. Introducción</b>	<b>1</b>
1.1. Introducción . . . . .	1
1.2. Antecedentes . . . . .	2
1.3. La congestión marítima . . . . .	4
1.4. Objetivos . . . . .	5
1.5. Planificación . . . . .	5
1.6. Estructura del documento . . . . .	7
<b>2. Estado del arte</b>	<b>8</b>
2.1. Tecnologías desarrolladas . . . . .	8
2.1.1. Sistemas de Identificación Automática (AIS) . . . . .	8
2.1.2. Radar de Puntero Automático (ARPA) . . . . .	9
2.1.3. Sistemas de imágenes de satélites . . . . .	9
2.1.4. Radares terrestres . . . . .	9
2.2. Aplicaciones existentes . . . . .	9
2.2.1. Maritime Portal . . . . .	10
2.2.2. Vesselfinder . . . . .	11
2.2.3. MarineTraffic . . . . .	12
<b>3. Diseño e implementación</b>	<b>13</b>
3.1. Arquitectura Hexagonal . . . . .	13
3.2. Diseño e implementación de la aplicación . . . . .	16
3.3. Back-end, diseño e implementación . . . . .	17
3.3.1. Capa de dominio . . . . .	18
3.3.2. Capa de aplicación . . . . .	20
3.3.3. Capa de adaptadores . . . . .	21
3.4. Front-end, diseño e implementación . . . . .	24
3.4.1. Capa de dominio . . . . .	24
3.4.2. Capa de aplicación . . . . .	25
3.4.3. Capa de adaptadores . . . . .	25
<b>4. Desarrollo</b>	<b>31</b>
4.1. Pruebas . . . . .	31
4.1.1. Pruebas del back-end . . . . .	31
4.1.2. Pruebas del front-end . . . . .	33
4.2. Documentación . . . . .	34
4.3. Despliegue . . . . .	35
4.4. Producto final . . . . .	36
4.4.1. Cambiar el idioma . . . . .	36

4.4.2. Monitorizar el tráfico marítimo (Live tracker) . . . . .	37
4.4.3. Comprobar el estado de un buque . . . . .	38
4.4.4. Comprobar el estado de un puerto . . . . .	41
4.4.5. Visualizar estadísticas globales . . . . .	44
<b>5. Experimentación</b>	<b>45</b>
5.1. Preparación . . . . .	45
5.2. Resultados . . . . .	45
<b>6. Presupuesto</b>	<b>50</b>
<b>7. Conclusiones y líneas futuras</b>	<b>51</b>
7.1. Conclusiones . . . . .	51
7.2. Líneas futuras . . . . .	51
<b>8. Conclusions and Future Work</b>	<b>52</b>
8.1. Conclusions . . . . .	52
8.2. Future Lines . . . . .	52
<b>Bibliografía</b>	<b>52</b>



# Índice de figuras

1.1. Crecimiento marítimo internacional y PIB . . . . .	3
1.2. Estadía en puerto . . . . .	4
1.3. Diagrama de Gantt . . . . .	6
2.1. Página de inicio de Maritime Portal . . . . .	10
2.2. Interfaz de Vesselfinder . . . . .	11
2.3. Interfaz de MarineTraffic . . . . .	12
3.1. Diseño básico arquitectura hexagonal . . . . .	14
3.2. Capa de dominio . . . . .	14
3.3. Capa de aplicación . . . . .	14
3.4. Capa de adaptadores . . . . .	15
3.5. Diagrama de la aplicación . . . . .	16
3.6. Diagrama del back-end . . . . .	17
3.7. Entidades . . . . .	18
3.8. Esquemas conexión API y WebSocket . . . . .	23
3.9. Diagrama Front-end . . . . .	24
4.1. Resultados test unitarios . . . . .	32
4.2. Resultados PortHandlerIntegrationTest . . . . .	32
4.3. Resultados VesselHandlerIntegrationTest . . . . .	32
4.4. Resultados UserHandlerIntegrationTest . . . . .	32
4.5. Resultados test end-to-end HomeComponent . . . . .	33
4.6. Resultados test end-to-end PortComponent . . . . .	34
4.7. Documentación en Swagger . . . . .	34
4.8. Diagrama de despliegue . . . . .	35
4.9. Barra de navegación . . . . .	36
4.10 Opciones de usuario . . . . .	36
4.11 Pie de página . . . . .	37
4.12 Live tracker mapa . . . . .	37
4.13 Live tracker tabla . . . . .	38
4.14 Lista de buques . . . . .	39
4.15 Datos de un buque . . . . .	39
4.16 Seguimiento del buque . . . . .	40
4.17 Historial del buque . . . . .	40
4.18 Puerto de salida . . . . .	40
4.19 Puerto de destino . . . . .	41
4.20 Gráfica de velocidades del buque . . . . .	41
4.21 Lista de puertos . . . . .	42
4.22 Vista de un puerto con llegadas planificadas . . . . .	42

4.23	Historial del puerto . . . . .	43
4.24	Planificación del puerto . . . . .	43
4.25	Próximas llegadas . . . . .	43
4.26	Buques en puerto . . . . .	43
4.27	Salidas . . . . .	44
4.28	Estadísticas generales . . . . .	44
5.1.	Live Tracker al terminar la simulación . . . . .	46
5.2.	Estadísticas generales simulación . . . . .	46
5.3.	Puertos distribución por tipo y servicios ofrecidos . . . . .	47
5.4.	Puertos distribuidos por países . . . . .	47
5.5.	Top puertos con mayor capacidad . . . . .	48
5.6.	Buques distribuidos por tipos . . . . .	48

# Índice de tablas

5.1. Rendimiento de la aplicación . . . . . 49

6.1. Presupuesto . . . . . 50

# Capítulo 1

## Introducción

### 1.1. Introducción

El transporte marítimo de mercancías es el más extendido en el comercio mundial debido a la gran capacidad de carga de los buques mercantes respecto a otros tipos de vehículos terrestres o aéreos.

En el transporte de mercancías existen diversas alternativas, una de ellas es tomar rutas marítimas, y es en este aspecto es donde entran los buques mercantes. Gracias a su gran capacidad de transporte que no poseen otros tipos de vehículos terrestres o aéreos, los buques mercantes son embarcaciones destinadas al transporte de grandes cantidades de mercancías. Debido a las diferencias que tiene cada mercancía, existen diversos tipos de buques mercantes según la carga que transporta<sup>1</sup>.

- *Buque de carga general*: También conocidos como buques multipropósito, se encarga de transportar mercancía seca pero mayoritariamente son utilizados para transportar mercancía suelta. No realizan transportes especializados como los otros tipos de buques, pero cuenta con equipo especializado para la carga y descarga de mercancías.
- *Buque granelero*: Se utilizan para transportar mercancía sin empaquetar. Transporta grandes cantidades de un solo producto, como grano, minerales, madera, fertilizantes, etc.
- *Buque portacontenedores*: Estos buques representan gran parte de la carga no granel a nivel mundial. Su nombre proviene del hecho de que la carga viene transportada en contenedores de diversa índole. Este tipo de embarcaciones se utilizan tanto en el comercio internacional que suelen estar automatizados.
- *Buques frigoríficos*: Utilizados para transportar carga perecedera, almacenan la mercancía refrigerada o congelada.
- *Buques roll-on/roll-off*: Este tipo de buque tiene el objetivo de transportar mercancía con ruedas como automóviles, camiones o maquinaria pesada.
- *Buques petroleros*: Están destinados al transporte de petróleo siendo capaces de atracar en alta mar.

---

<sup>1</sup><https://www.suisagroup.com/en/noticias/types-of-cargo-ships-according-to-the-load-they-carry/>

- *Buques metaneros*: Se encargan del transporte de gas licuado o gas natural, utilizando maquinaria de alto rendimiento. La carga se almacena en tanques esféricos ubicados en la cubierta.
- *Buque de carga química*: Utilizados para transportar productos químicos. Normalmente los productos se guardan en tanques especiales para evitar la corrosión.
- *Buques ganaderos*: Diseñados para transportar gran cantidad de animales vivos. Los animales pueden estar ubicados en corrales en la cubierta, o en el interior con temperatura climatizada.
- *Buques para grandes cargas*: Utilizados para transportar cargas muy pesadas o de gran tamaño como maquinaria industrial.

Las infraestructuras portuarias son instalaciones ubicadas en la costa o en alta mar utilizados para el comercio e intercambio de mercancías o pasajeros entre diferentes países. Independientemente del tamaño del puerto, estos son diseñados para proporcionar un entorno a los buques, ya sea en alta mar o en interior, así como ofrecer diferentes clases de servicios. Estos servicios pueden ir desde el almacenamiento, manipulación o distribución de mercancía, hasta servicios de reparación, mantenimiento y suministro de combustible o proveer servicios logísticos.

Al ser infraestructuras logísticas y financieras fundamentales para la economía global, las autoridades portuarias están a cargo de albergar y asegurar los buques, al tiempo que garantizan que las operaciones de fondeo, atraque/desatraque, amarre/desamarre, descarga y carga, etc. transcurran sin problemas. Se puede definir que la gestión portuaria es el proceso de organizar, monitorizar y controlar las actividades ocurridas en las infraestructuras portuarias [3].

## 1.2. Antecedentes

En la actualidad, debido a la globalización económica, el transporte marítimo es la solución predilecta para la transferencia de mercancías de diversa índole, desde tecnología hasta graneles sólidos o productos industriales. Según la Conferencia de las Naciones Unidas sobre Comercio y Transporte (UNCTAD), actualmente el transporte marítimo transporta más del 80 % del volumen de mercancías a nivel mundial <sup>2</sup>.

A lo largo de la historia el transporte marítimo también ha jugado un papel fundamental en el desarrollo del comercio, tanto internacional como interregional. El transporte por rutas oceánicas y costeras nos permiten posteriormente conectar las carreteras, los ferrocarriles y el interior de vías fluviales. Con esto se ha logrado que los países puedan abrir sus fronteras y experimentar un crecimiento económico sin precedentes gracias al mayor flujo de conocimiento, servicios y bienes [4].

Hoy en día la economía global y marítima se enfrentan a la situación post-COVID-19 y los conflictos geopolíticos. Según las previsiones, a lo largo del 2022 el comercio marítimo

---

<sup>2</sup><https://unctad.org/rmt2022>

se moderaría un 1,4 %, mientras que a lo largo del periodo 2023-2027 crecerá un 2,1 % anual, distando del promedio del 3,3 % de crecimiento de los tres decenios anteriores<sup>3</sup>.

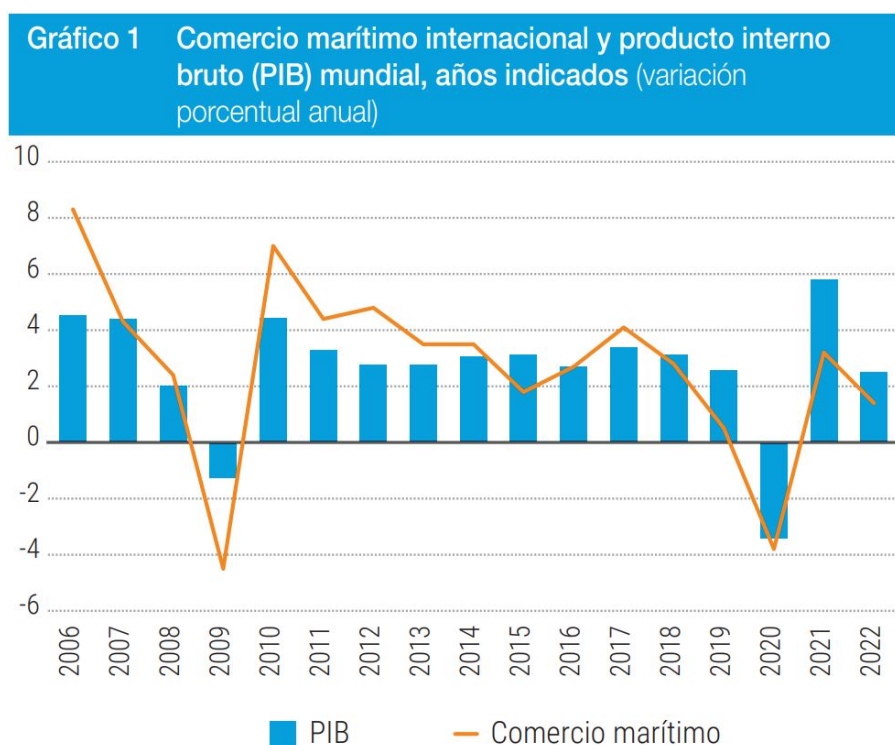


Figura 1.1: Crecimiento marítimo internacional y PIB

Como se puede ver en la Figura 1.1, con la situación de la pandemia el comercio marítimo se resintió con una contracción del 3,8 % en 2020, recuperándose con un crecimiento del 3,2 % en 2021, siendo éste un crecimiento que no dista mucho de los niveles de los años anteriores al COVID-19. Esto último es debido a que el comercio aún se hallaba recuperándose de la pandemia y, consecuentemente, del bloqueo del sistema logístico mundial al existir una gran demanda pero grave escasez de oferta.

En el ámbito local, la economía canaria se enfrenta a varias dificultades adicionales por su situación de archipiélago ultraperiférico, que conlleva un aumento en los costes de transporte, que a su vez repercuten en un menor desarrollo. Es fundamental que los puertos canarios tengan su operativa optimizada, ya que sus infraestructuras portuarias son clave para su economía y evolución socioeconómica.

Canarias cuenta con diversos enclaves portuarios, este conjunto de puertos es administrado tanto por la Autoridad Portuaria de Santa Cruz de Tenerife<sup>4</sup>, como por la Autoridad Portuaria de Las Palmas<sup>5</sup>, encargándose cada uno de sus respectivas provincias. Es importante remarcar que el comercio marítimo en Canarias cerró el 2021 con un volumen de 8,4 millones de toneladas en mercancías procedentes de la Península, lo que implica que el 91 % de las importaciones mercantiles se produjeron por vía marítima<sup>6</sup>.

<sup>3</sup>[https://unctad.org/system/files/official-document/rmt2022overview\\_es.pdf](https://unctad.org/system/files/official-document/rmt2022overview_es.pdf)

<sup>4</sup><https://www.puertosdetenerife.org>

<sup>5</sup><https://www.palmasport.es/es/>

<sup>6</sup><https://www.rhenus.group/es/es/rhenus-group/rhenus-en-espana/blog/blog-detail/como-se-realiza-el-transporte-de-mercancias-de-la-peninsula-a-canarias/>

### 1.3. La congestión marítima

Una de las principales problemáticas en el comercio marítimo es la congestión portuaria en todos sus ámbitos.

Cuando los distintos buques mercantes no atracan en el instante esperado en las estructuras portuarias se empieza a generar la congestión marítima. Se considera que un puerto se encuentra congestionado cuando distintos usuarios interfieren entre ellos en el uso de los recursos portuarios, lo que aumenta el tiempo de estos usuarios en puerto.

La congestión portuaria resulta en un mayor tiempo de espera antes del atraque y menores niveles de calidad en los servicios dedicados a los buques. Este problema a largo plazo contribuye también a que exista una menor competitividad entre los puertos y una disminución de demanda. Como norma general, se produce congestión en un puerto cuando la demanda de recursos portuarios excede a la oferta disponible. Si se consigue reducir la congestión se genera una mayor oferta de servicios de transporte y ayuda a eliminar la crisis de las cadenas de suministro y los bloqueos logísticos. Para conseguir reducir la congestión es necesario enfocarse en integrar soluciones en toda la cadena de suministros y no solo en el cuello de botella. [7]

**Cuadro 1** Tiempo en puerto, edad y tamaño de los buques, por tipo de buque, 2021 (total mundial)

Tipo de buque	Mediana de tiempo en puerto (días)	Mediana de tiempo en puerto, variación anual (%)	Tamaño medio de los buques (TB)	Tamaño máximo de los buques (TB)	Capacidad de carga media por buque (TPM)	Capacidad de carga máxima de los buques (TPM)	Capacidad de carga media por portacontenedor (TEU)
Portacontenedores	0,80	13,7	37 223	237 200			3 431
Buques de carga seca fraccionada	1,17	2,1	5 463	91 784	7 427	116 173	
Graneleros de carga seca	2,11	2,3	32 011	204 014	57 268	404 389	
Buques para el transporte de GNL	1,13	0,9	95 356	168 189	74 522	155 159	
Buques para el transporte de GPL	1,03	-1,5	10 541	61 000	11 799	64 220	
Graneleros de carga líquida	0,98	1,3	15 739	170 618	27 275	323 183	
<b>Todos los buques</b>	<b>1,05</b>	<b>4,8</b>	<b>21 732</b>	<b>237 200</b>	<b>26 997</b>	<b>404 389</b>	<b>3 431</b>

Figura 1.2: Estadía en puerto

En 2021 la congestión portuaria se concentró en tres zonas críticas: China, Europa del Norte y la Coste Oeste de los Estados Unidos. Los puertos, al hallarse congestionados por no disponer de equipo suficiente, personal y espacios de almacenamiento, tenían dificultades para responder al aumento de demanda. Como consecuencia, el retraso promedio en el transporte de contenedores se duplicó, los retrasos pasaron de 2 a 12 días y el tiempo medio de estadía en puerto de los portacontenedores creció un 13,7% (Figura 1.2). Para evitar la congestión portuaria en Europa del Norte se limitaron el número de escalas por rotación, lo que conllevó a un incremento de carga en cada esca-

la, más horas de trabajo en las terminales y un aumento de presión en los grandes puertos.

Por otro lado, los puertos de Estados Unidos tuvieron que manejar cerca de 28 millones de contenedores entrantes. Estos contenedores llegaron a abrumar algunos puertos provocando un cuello de botella sin precedentes en la cadena de suministros, que hizo más lucrativo para las empresas devolver contenedores vacíos a Asia. Por ejemplo, en Noviembre de 2021, enviar un contenedor de veinte pies de Shanghái a Los Ángeles costaba 5.000\$, mientras que el recorrido de vuelta solo 650\$ [8].

En 2022, muchos buques se encontraban bloqueados en puerto, haciendo que se encontraran varados en las infraestructuras portuarias un 37 % de la capacidad mundial de portacontenedores. Este aspecto, junto con la política de cero COVID de China y el consiguiente confinamiento de Shenzhen y Shanghái, obligaron a las empresas a redirigir sus buques a puertos alternativos. De esta forma, para que Asia pudiera responder al problema de la congestión marítima se tuvieron que reorganizar las rutas comerciales y crear nuevos servicios regionales.

## 1.4. Objetivos

El objetivo principal de este proyecto es el desarrollo de una aplicación software que permita la gestión y planificación, proponiendo una solución eficiente, de la llegada de los buques a las infraestructuras portuarias.

Sea Calm, a diferencia de otras aplicaciones web que ofrecen servicios parciales de forma gratuita o tienen planes de pago, se ofrece de forma totalmente gratuita. De esta forma cualquier persona se puede beneficiar de los servicios ofrecidos.

Los objetivos específicos del proyecto son los siguientes:

- Permitir la visualización en tiempo real de la ubicación de buques mercantes así como de infraestructuras portuarias.
- Reflejar el estado de los puertos y buques a lo largo del tiempo.
- Incluir diferentes indicadores estadísticos de utilidad para la gestión portuaria.
- Realizar una implementación limpia y mantenible de una aplicación *full stack*. La aplicación resultante debe ser escalable, con un código fácil de mantener y comprender, además de poseer test de diferente índole.
- Internacionalizar la aplicación, incluyendo el idioma Inglés.

## 1.5. Planificación

El desarrollo del proyecto se plantea de una forma autónoma e iterativa, marcando los diferentes hitos a cumplir por la aplicación. El plan de trabajo se divide en la ejecución de las siguientes tareas:



- *Tarea 1:* Documentación e investigación de la problemática y las herramientas a utilizar.
- *Tarea 2:* Diseño de prototipo de la aplicación web.
- *Tarea 3:* Diseño de las entidades.
- *Tarea 4:* Diseño y creación del back-end
  - Implementación de la capa de dominio
  - Creación de los servicios
  - Creación de los controladores
  - Recepción de señales mediante websocket
- *Tarea 5:* Diseño y creación del front-end
  - Visualización de buques y puertos en tiempo real
  - Mostrar datos de los buques
  - Mostrar datos de los puertos
  - Visualización de planificación
  - Panel de control con indicadores
- *Tarea 6:* Redacción de la memoria
- *Tarea 7:* Presentación final

La Figura 1.3 muestra un diagrama de Gantt con la planificación inicial planteada para el proyecto.



Figura 1.3: Diagrama de Gantt

## 1.6. Estructura del documento

El resto de la presente memoria se divide en los siguientes capítulos:

- *Capítulo 2 - Estado del arte.* Revisión de las herramientas y tecnologías que se han desarrollado en este ámbito, así como las aplicaciones que las utilizan.
- *Capítulo 3 - Diseño e implementación.* Explicación del diseño de la aplicación realizada, así como diversos detalles en su implementación.
- *Capítulo 4 - Desarrollo.* Muestra cada una de las funcionalidades disponibles en la aplicación. Además, explica en detalle como se realiza el testing, documentación y despliegue de la misma.
- *Capítulo 5 - Experimentación.* Prueba de funcionamiento de la aplicación a través de un caso realista. Obtiene resultados a través de diferentes indicadores.
- *Capítulo 6 - Presupuesto.* Presupuesto para la realización del proyecto.
- *Capítulo 7 - Conclusiones y líneas futuras.* Principales conclusiones del trabajo, limitaciones del mismo y mejoras que se pueden implementar en el futuro.
- *Capítulo 8 - Conclusions and Future Work.* Capitulo anterior escrito en inglés.

# Capítulo 2

## Estado del arte

### 2.1. Tecnologías desarrolladas

En los últimos años, el desarrollo de tecnologías para el seguimiento y rastreo del tráfico marítimo ha sido necesario, si no fundamental, para mejorar la seguridad y eficiencia del transporte de mercancías.

#### 2.1.1. Sistemas de Identificación Automática (AIS)

AIS es una solución tecnológica importante y ampliamente utilizada, dado que es un sistema de uso obligatorio en la mayoría de buques comerciales. Su utilización abarca supervisión del tráfico marítimo, prevención de accidentes, operaciones de búsqueda y rescate e investigaciones de accidentes. La información aportada por AIS en rescate e investigación es altamente valiosa, dado que proporciona coordenadas GPS, rumbo, velocidad e información adicional. En comparación con la tecnología de radar, ampliamente usada en la actualidad, se consiguen datos mucho más precisos.

AIS funciona adquiriendo coordenadas GPS e intercambiando información actual y actualizada a través de transmisiones de radio a otros barcos y autoridades marítimas, es decir, servicios de tráfico marítimo ubicados en tierra. Esto permite que las autoridades portuarias puedan ayudar en la navegación de los buques o advertir sobre peligros como las mareas bajas, afloramientos rocosos o bancos de arena.

Los datos de AIS se recopilan e intercambian entre los proveedores de AIS que operan a través de Internet, estos proveedores ofrecen visualización, monitorización e informes de datos AIS de forma gratuita o comercial [1].

La tecnología AIS es un buen complemento para las tecnologías que se explicarán a continuación, sobre todo en combinación con las tecnologías de radar y ARPA, sin ser AIS reemplazo de estos. La combinación de AIS y ARPA para fines anticolidión simplifica la identificación de los barcos y ayuda a la toma de decisiones. Por ejemplo, con la capacidad de predicción de trayecto que proporciona AIS en situaciones de encuentro, los buques pueden saber con exactitud la forma en la que maniobran otros barcos [6].

### **2.1.2. Radar de Puntero Automático (ARPA)**

Los sistemas ARPA utilizan tecnología de radar avanzado que permiten visualizar en pantalla de forma automática los objetos en el agua, incluidos los barcos y el propio buque. Estos sistemas permiten calcular el rumbo del resto de buques, su velocidad y el punto más cercano de aproximación (CPA), pudiendo avisar de un posible riesgo de colisión. Este tipo de radares son comunes en barcos mercantiles y de gran tamaño.

ARPA ha sido uno de los primeros dispositivos reconocidos por la Organización Marítima Internacional y gradualmente se fue convirtiendo en un requisito para barcos de gran envergadura. Para los barcos de menor tamaño se desarrolló el dispositivo de Ayuda de seguimiento automático (ATA), que tiene casi todas las características de ARPA.

El funcionamiento de ARPA puede ser de forma manual o automática, en ambos casos se centra en la adquisición y seguimiento de objetivos. En la forma manual el operador indica al ordenador los objetivos que se van a rastrear, mientras que de la forma automática el propio dispositivo está programado para detectar los objetivos que corresponden con las características asignadas. Cuando se adquiere un objetivo, el ordenador comienza a recopilar datos relacionados [2].

### **2.1.3. Sistemas de imágenes de satélites**

La utilización de imágenes de satélites ha incrementado la eficacia del seguimiento del tráfico marítimo a nivel global. Los satélites con sensores ópticos y radares permiten detectar barcos incluso en zonas remotas. Este tipo de sistemas de seguimiento pueden detectar aquellos buques que no dispongan de dispositivo de rastreo o que éste se encuentre apagado, lo que permite obtener una visión más completa del tráfico marítimo.

### **2.1.4. Radares terrestres**

Además de utilizar los radares de los barcos, también se han implementado radares terrestres para el seguimiento del tráfico marítimo. Los radares terrestres, ubicados en las estaciones costeras o faros, son utilizados para detectar y rastrear los buques dentro de su alcance. Son muy útiles en combinación con datos extraídos de otras fuentes, como AIS, para obtener una información más completa del tráfico marítimo en una región determinada

## **2.2. Aplicaciones existentes**

En el mercado también se han desarrollado diversas aplicaciones y plataformas en línea que han empezado a utilizar datos provenientes de AIS u otras fuentes para realizar el seguimiento del tráfico marítimo en tiempo real. En el presente Trabajo de Fin de Grado se comentan tres aplicaciones existentes: Maritime Portal, Vesselfinder y MarineTraffic.

## 2.2.1. Maritime Portal

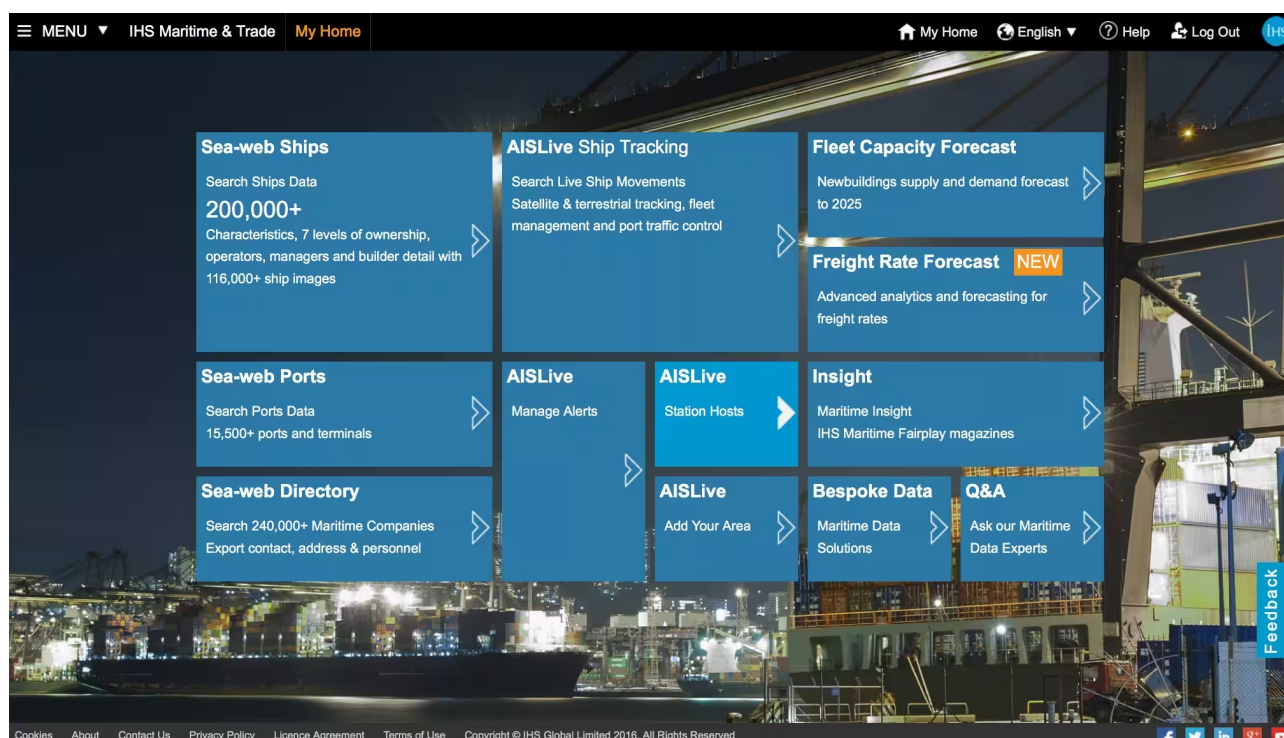


Figura 2.1: Página de inicio de Maritime Portal

Maritime Portal<sup>1</sup> es una solución software de S&P Global. Esta herramienta es una plataforma que a su vez combina los datos de otros dos productos de la empresa, Sea-web<sup>2</sup> y AISLive<sup>3</sup>.

Por un lado Sea-web se describe como una de las bases de datos marítimas más grande de la industria. Cuenta con múltiples módulos separados donde cada uno de ellos integra información detallada sobre puertos, buques y su movimiento, propietarios o compañías, así como noticias y análisis marítimos. Adicionalmente te permite exportar datos como archivos de texto para realizar análisis estadísticos, realizar búsquedas simples o complejas seleccionando los campos de datos deseados y contando con un historial de búsquedas, además de permitir a los usuarios tener notas personales o compartidas de los registros de la base de datos.

Por otro lado, AISLive es un software en línea que permite rastrear el tráfico marítimo en tiempo real de forma actualizada. Ofrece la visualización del movimiento de más de 130.000 buques y embarcaciones con una actualización cada 60 segundos. Incluye adicionalmente la actividad de más de 15.500 puertos de todo el mundo. Para realizar este rastreo en tiempo real se utiliza cobertura satélite y cobertura marítima gracias al equipamiento de la tecnología AIS en cada buque.

<sup>1</sup><https://www.spglobal.com/marketintelligence/en/mi/products/maritime-ship-tracker-ais-live-ship-data-seaweb.html>

<sup>2</sup><https://www.spglobal.com/marketintelligence/en/mi/products/sea-web-maritime-reference.html>

<sup>3</sup><https://www.spglobal.com/marketintelligence/en/mi/products/ais-live-ship-tracker.html>

Maritime Portal se ofrece mediante planes de suscripción, contando con los planes: bronze, silver y gold. Las funcionalidades que ofrece son:

- Permite proteger los activos de los clientes, supervisando automáticamente cualquier barco que disminuya la velocidad o atraque en una zona o puerto personalizado.
- Envía avisos cuando un barco cambie su estado AIS a amarrado o encallado.
- Permite conocer los detalles críticos sobre los barcos que ingresan en los puertos propios y recibir alertas cuando un barco cambie su destino o llegada estimada.
- Ofrece imágenes holísticas de la flota mundial, las empresas que las gestionan, los puertos a los que hacen escala, sus movimientos y su historial comercial.

## 2.2.2. Vesselfinder

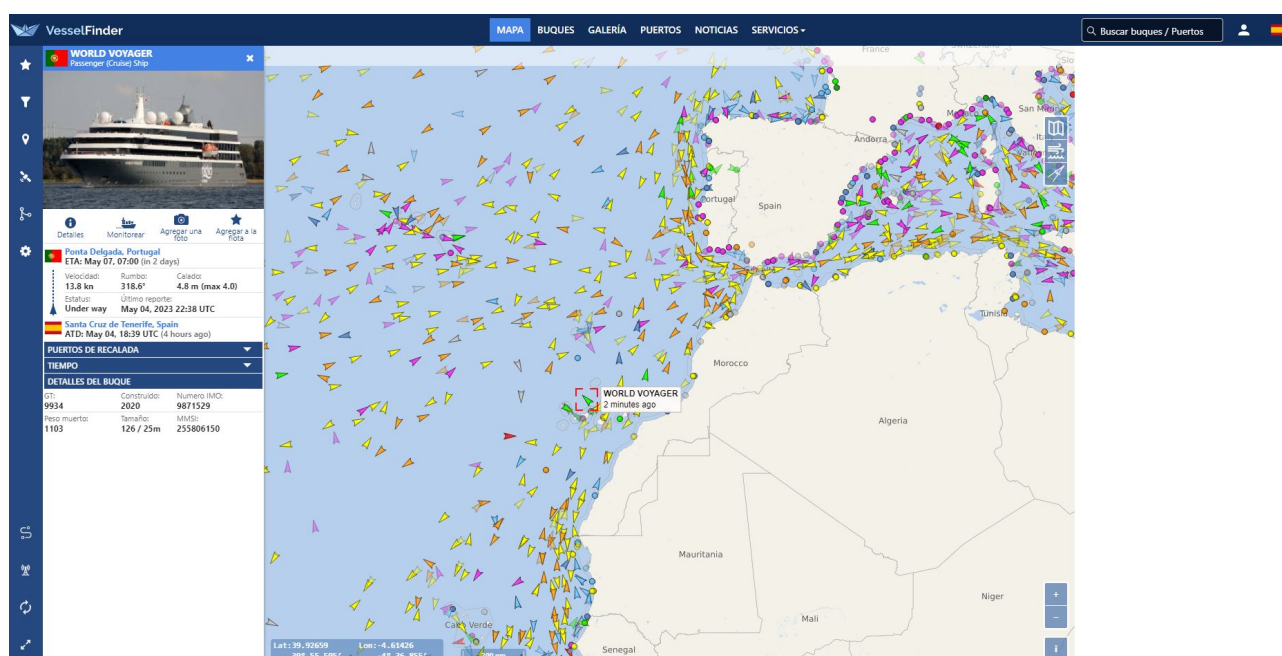


Figura 2.2: Interfaz de Vesselfinder

Vesselfinder<sup>4</sup> es un servicio en línea que permite a los usuarios rastrear la ubicación y movimiento de barcos en tiempo real utilizando tecnología AIS. Permite recibir información sobre la ubicación de los barcos, su rumbo y velocidad para mostrar esta información en un mapa interactivo.

Los usuarios pueden buscar los barcos por su nombre, número IMO (Organización Marítima Internacional)<sup>5</sup>, MMSI (Identificador de Sistema Mundial de Socorro y Seguridad Marítima)<sup>6</sup> o por la ubicación geográfica.

<sup>4</sup><https://www.vesselfinder.com/es>

<sup>5</sup><https://www.imo.org/en/ourwork/msas/pages/imo-identification-number-scheme.aspx>

<sup>6</sup><https://www.mitma.gob.es/marina-mercante/radiocomunicaciones/solicitud-mmsi/identificacion-del-servicio-movil-maritimo-mmsi>

También ofrece herramientas y funciones para ayudar a los usuarios a seguir y analizar la actividad marítima con alertas de llegada y salida, seguimiento de la flota e historial de viajes.

La utilización de VesselFinder puede ser gratuita, pero limitada a un historial de seguimiento de 1 día y una flota de 10 buques, ofreciendo mayor cobertura con los planes Premium y Satélite.

### 2.2.3. MarineTraffic

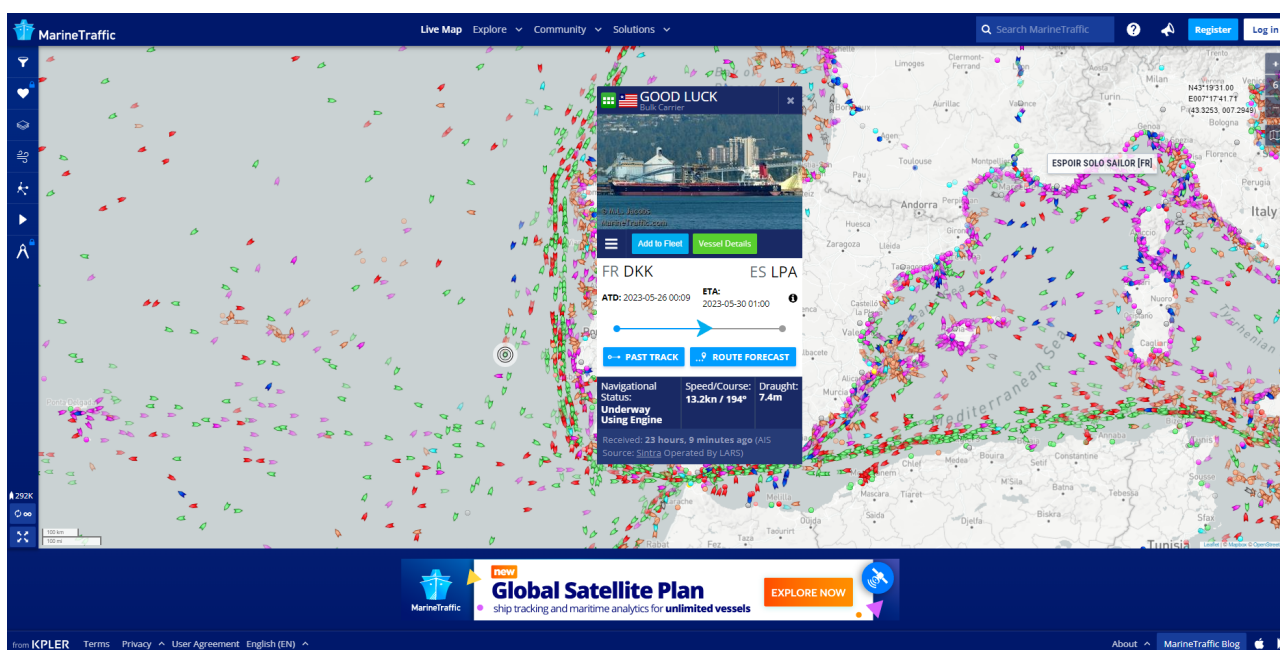


Figura 2.3: Interfaz de MarineTraffic

La aplicación MarineTraffic<sup>7</sup> es una plataforma en línea que ofrece datos en tiempo real sobre el tráfico marítimo en todo el mundo. Al igual que las aplicaciones Maritime Portal y VesselFinder utiliza el sistema AIS para obtener y mostrar la ubicación e información de los buques en un mapa interactivo.

Marine Traffic destaca por su accesibilidad, no solo ofreciendo información de buques, también de puertos y terminales, de forma gratuita. Se ha convertido en una de las herramientas más utilizadas para el seguimiento de barcos, creciendo considerablemente desde que surgió en 2007.

Cuenta a su vez con una amplia base de datos que contiene información de barcos, incluyendo buques mercantes, barcos pesqueros, de recreo o buques militares. Entre la información ofrecida se incluye el nombre del barco, la posición en tiempo real, velocidad y rumbo, su destino y que tipo de barco es. Adicionalmente, muestra también el histórico de ruta de los buques, pudiendo hacer un seguimiento de su actividad.

Además de tener una red de estaciones receptoras de AIS costeras para conseguir la información de los buques, complementan este aspecto con rastreo mediante satélite.

<sup>7</sup><https://www.marinetraffic.com>

# Capítulo 3

## Diseño e implementación

### 3.1. Arquitectura Hexagonal

Una de las bases fundamentales del diseño de la aplicación es la utilización de la arquitectura hexagonal, por tanto es importante conocer y entender que implica su utilización.

La arquitectura hexagonal, también conocida como arquitectura de puertos y adaptadores, fue enunciada por primera vez por Alistair Cockburn en 2005 en su artículo "The Pattern: Ports and Adapters (Object Structural)"<sup>1</sup>. Como expone Alistair en su artículo, el propósito de la arquitectura hexagonal es el siguiente:

*"Permitir que una aplicación sea ejecutada indistintamente por usuarios, programas, pruebas automatizadas, o archivos batch; y que sea desarrollado y probado por separado, sin los posibles dispositivos y bases de datos de los que dependa en tiempo de ejecución."*

Esta premisa permite comprender las bases de lo que se requiere con la arquitectura hexagonal. Una de las ideas principales al utilizar esta arquitectura es el concepto de separar el código de negocio del código tecnológico, cumpliendo los siguientes principios:

- Asegurarse de que el lado tecnológico no dependa del comercial, para que este último consiga evolucionar sin preocuparse de las tecnologías a usar para cumplir con los objetivos comerciales.
- Ser capaz de cambiar el código tecnológico sin causar ningún problema en su contraparte comercial.

Para cumplir estos dos principios hay que determinar un lugar donde exista el código comercial de forma aislada y protegida de cualquier problema externo. Esto da pie, como se puede observar en la Figura 3.1, a la creación de tres capas o hexágonos, los cuales además de cumplir con los requerimientos ya comentados permiten establecer el propósito de la arquitectura hexagonal.

---

<sup>1</sup><https://alistair.cockburn.us/hexagonal-architecture/>



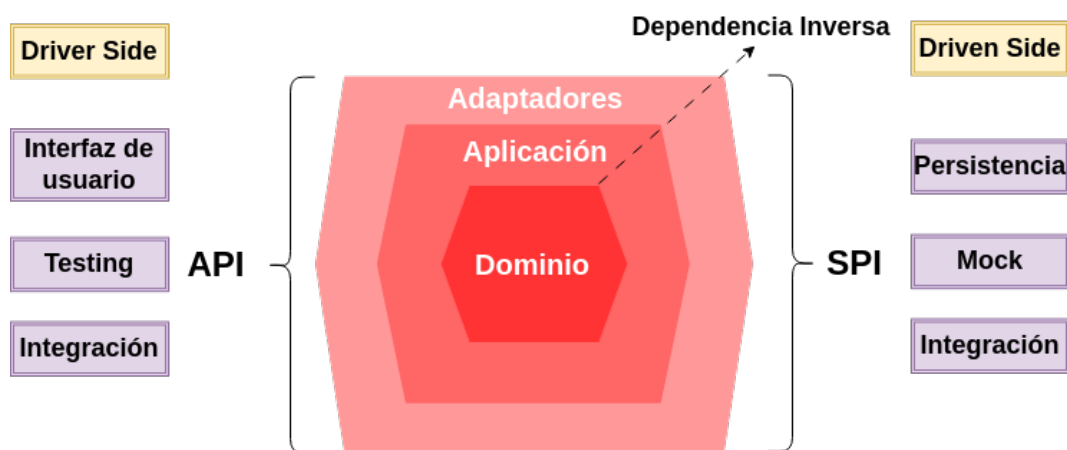


Figura 3.1: Diseño básico arquitectura hexagonal

La capa más interna, conocida como la capa de dominio, aloja el código comercial y es donde se construyen los elementos necesarios para describir los problemas que se quieren resolver con la aplicación. Dentro de esta capa se definen las entidades correspondientes a reglas y datos de negocio críticos, ya que representan un modelo de un problema real, tal y como se muestra en la Figura 3.2. Esta capa es básica para la construcción de la aplicación, por tanto, a la hora de diseñarla se debe tener un conocimiento profundo del dominio del problema. En caso contrario, el resto de capas estarán basadas en entidades incorrectas con los problemas que esto ocasiona.

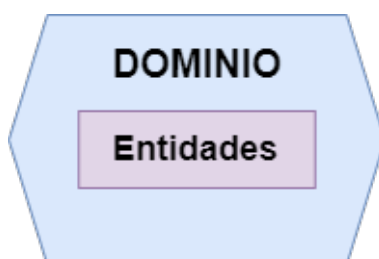


Figura 3.2: Capa de dominio

La siguiente capa es la de aplicación (Figura 3.3), que permite usar, procesar y manejar la lógica de negocio proveniente de la capa de dominio, tratando de forma abstracta las tareas específicas de la aplicación. La capa de aplicación se encuentra entre el código comercial y el código tecnológico, sirviendo de intermediario entre ambos.

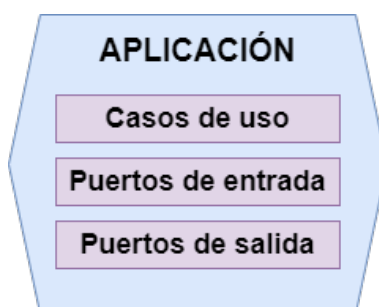


Figura 3.3: Capa de aplicación

Esta capa utiliza casos de uso y puertos de entrada y salida para generar las funcionalidades:

- **Casos de uso:** Los casos de uso representan el comportamiento del sistema a través de operaciones específicas de la aplicación. Pueden interactuar directamente con entidades y otros casos de uso, lo que los convierten en componentes flexibles.
- **Puertos de entrada:** Son componentes que se adjuntan directamente a los casos de uso en el nivel de la aplicación. Los puertos de entrada permiten implementar la intención del software en los términos del dominio.
- **Puertos de salida:** Al igual que los puertos de entrada, son componentes adjuntados directamente a los casos de uso, con la diferencia que tienen el papel de obtener datos de recursos externos y satisfacer el caso de uso.

La capa más externa es la capa de adaptadores (Figura 3.4), la cual proporciona la interfaz de comunicación. En la capa de adaptadores se definen de qué manera se exponen las funciones de la aplicación y qué tecnologías van a conectarse con la aplicación adaptándose a los puertos de entrada y salida. Esta conexión se puede realizar de dos formas, mediante los adaptadores de entrada (*driver side*) o a través de los adaptadores de salida (*driven side*)[9].

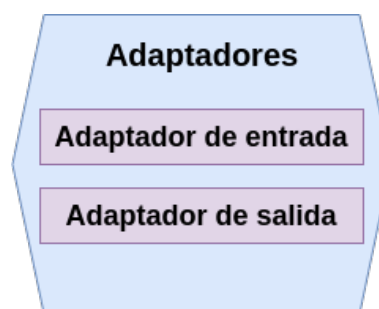


Figura 3.4: Capa de adaptadores

- Los adaptadores de entrada son los que solicitan acciones al software. Estos adaptadores son usados para permitir la comunicación, por ejemplo, con un front-end que utilice una interfaz de programación de aplicaciones (API). Esta API define como las entidades externas se comunican con el sistema y posteriormente traduce la petición para el dominio de la aplicación. Estos adaptadores de entrada tienen que ajustarse a los puertos de entrada.
- Los adaptadores de salida son activados desde la propia aplicación y salen al exterior para obtener datos que satisfagan las necesidades del software. Normalmente un adaptador de salida es utilizado como respuesta a un adaptador de entrada. Estos adaptadores tienen que ajustarse a los puertos de salida.

Los beneficios que se persiguen al utilizar la arquitectura hexagonal son los siguientes [5]:

- **Independiente de frameworks.** La arquitectura no depende de la existencia de alguna librería software o framework. Esto permite utilizar dichos frameworks como herramientas, en lugar de tener que forzar su utilización.

- Testeable. La lógica de negocio o funcionalidades pueden ser testeadas sin la interfaz de usuario, la base de datos, el servidor web o cualquier otro elemento externo.
- Independiente de la interfaz de usuario. La UI puede ser sustituida con facilidad sin cambiar el resto del sistema. Una interfaz de usuario web podría ser reemplazada con una interfaz de usuario de consola y no habría necesidad de cambiar la lógica de negocio.
- Independiente de la base de datos. Permite cambiar MongoDB por Oracle, SQL Server, BigTable, CouchDB o cualquier otra base de datos. La lógica de negocio no es dependiente o se encuentra vinculada a la base de datos.
- Independiente de cualquier agente externo. De hecho, las funcionalidades y entidades desconocen las interfaces que existen en el exterior.

## 3.2. Diseño e implementación de la aplicación

El diseño de la aplicación se organiza en cuatro grandes componentes intercomunicados: front-end, back-end, base de datos y productor de barcos, este último explicado en detalle en el capítulo 5. La Figura 3.5 muestra como se organiza cada uno de los componentes en la aplicación. Cabe destacar que cada uno de ellos se encuentra alojado en un contenedor docker, lo que proporciona un entorno aislado de cara al despliegue de la aplicación. Adicionalmente, se observa que el front-end y el back-end se conectan mediante una API REST. El back-end también tiene dos tipos de conexiones adicionales, una conexión WebSocket con el productor de barcos y otra conexión MongoDB con la base de datos.

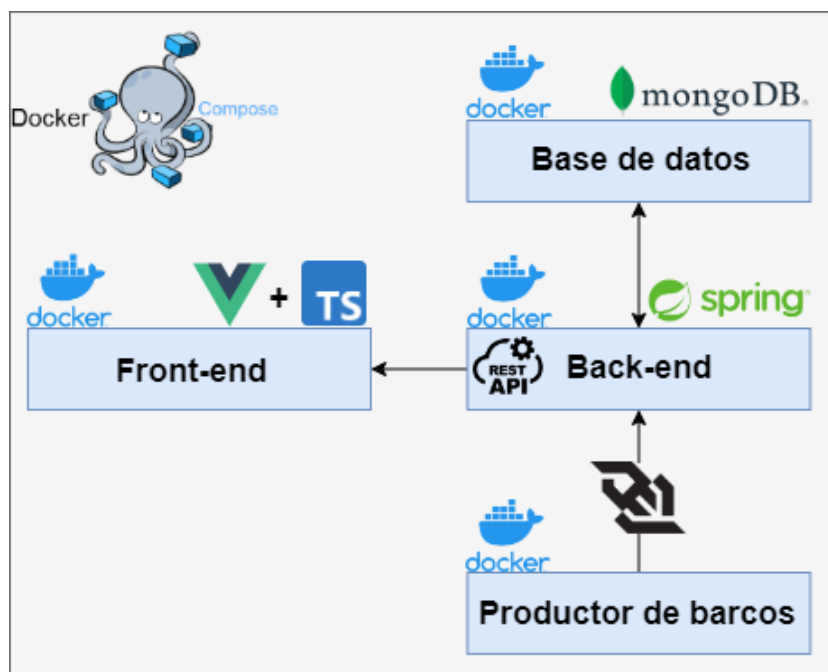


Figura 3.5: Diagrama de la aplicación

### 3.3. Back-end, diseño e implementación

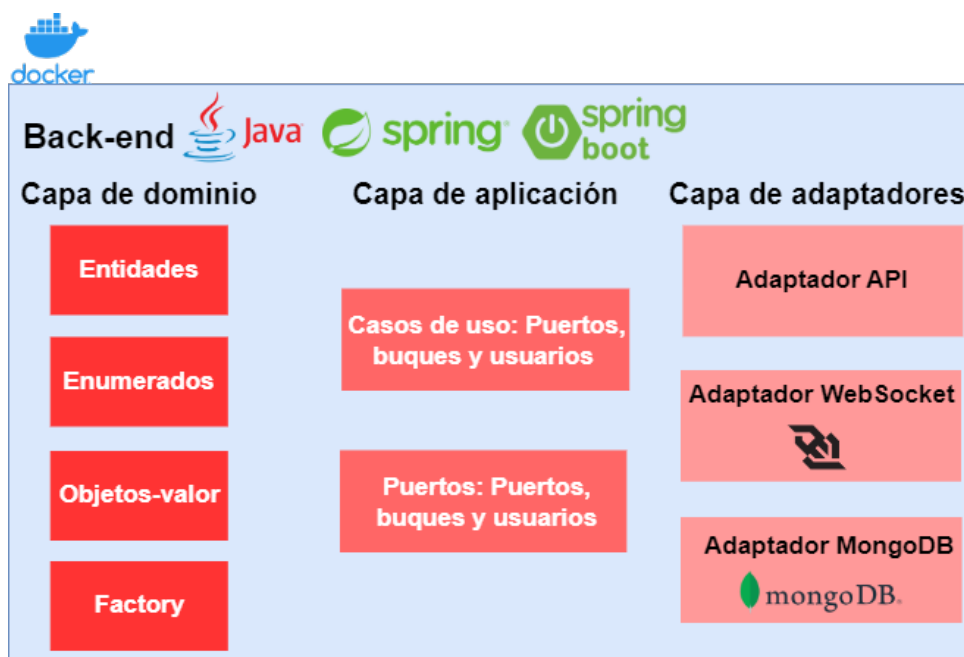


Figura 3.6: Diagrama del back-end

Una de las decisiones fundamentales en cuestión de diseño del back-end es el lenguaje de programación utilizado para la implementación. El lenguaje escogido es Java<sup>2</sup>, ya que nos permite realizar, gracias a los frameworks disponibles, el diseño de la arquitectura hexagonal con mayor facilidad.

Para ello, el principal framework de desarrollo utilizado es Spring<sup>3</sup>. Spring se trata de un framework open source que permite programar en Java de forma más rápida y segura. Cuenta con un gran número de características, de entre las cuales la más utilizada en este proyecto es la inyección de dependencias.

Con el fin de facilitar la configuración del proyecto en Spring, se ha utilizado Spring Boot<sup>4</sup>. Entre las características de esta herramienta se encuentran:

- Permitir hacer una configuración automática para que Spring Framework funcione.
- Ofrecer métricas de rendimiento.
- Ofrecer un servidor incorporado para evitar la complejidad en el despliegue de una aplicación Spring.
- Contar con dependencias iniciales que simplifican la compilación y configuración de una aplicación Spring.

<sup>2</sup><https://www.java.com/es/>

<sup>3</sup><https://spring.io>

<sup>4</sup><https://spring.io/projects/spring-boot>

### 3.3.1. Capa de dominio

En la capa de dominio del back-end se implementan las entidades que se utilizan para la aplicación. Es importante que las entidades se encuentren bien definidas ya que van a representar los datos que necesitan persistencia y con los que se construye la lógica de la aplicación. Por otra parte, estas entidades son las que el cliente, nuestro front-end, va a solicitar mediante peticiones al API REST.

Esta capa se encuentra en la carpeta *back-end/src/main/java/com/vessels/domain*. Dentro del dominio se tienen las carpetas:

- *./entity*: Almacena las entidades.
- *./enums*: Guarda los enumerados utilizados para construir las entidades
- *./valueobject*: Guarda los objeto-valor<sup>5</sup> utilizados para construir las entidades.
- *./factory*: Guarda una clase factory de cada entidad, permitiendo establecer el patrón de diseño Factory<sup>6</sup>.

#### Entidades

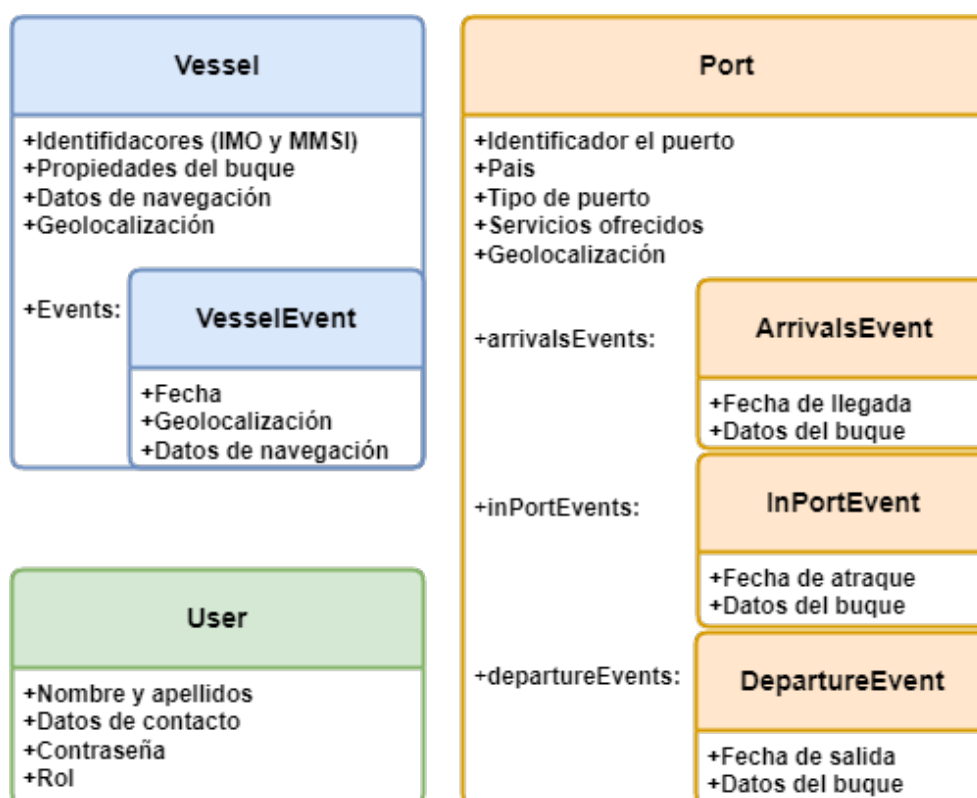


Figura 3.7: Entidades

Como muestra la Figura 3.7, las entidades definidas así como sus atributos más importantes son:

<sup>5</sup><https://leanmind.es/es/blog/primitive-obsesion-value-objects/>

<sup>6</sup><https://refactoring.guru/es/design-patterns/factory-method/java/example>

- *Vessel*: Entidad fundamental de la aplicación, la cual se corresponde a un buque mercante.
- *Port*: Entidad cuyo cometido es la representación de un puerto.
- *User*: Entidad utilizada para registrar los usuarios de la aplicación.
- *VesselEvent*: Entidad que representa los eventos que tendría un barco. Cuando el barco cambia de posición o estado, el evento se actualizaría. Al tener la lista de eventos de un barco se puede tener un historial y representar su tránsito u observar el estado del barco en un momento determinado.
- *ArrivalsEvent*: Entidad que representa el evento correspondiente a la llegada de un barco a puerto.
- *InPortEvent*: Entidad que representa el evento correspondiente a cuando un barco se encuentra en puerto.
- *DepartureEvent*: Entidad que representa el evento correspondiente a cuando un barco sale de un puerto.

Las entidades implementadas siguen un esquema parecido en donde se simplifica al máximo sus funcionalidades, son clases de "datos" o "modelos de datos". Se encuentran conformadas por los atributos privados, métodos setters y getters, dos métodos adicionales para obtener los datos en formato JSON o String, y en el caso de las entidades *Port* y *Vessel*, métodos adicionales para insertar nuevos eventos.

Los atributos de la entidad están conformados por atributos simples, objeto-valor y enumerados. Los atributos simples cuentan con un solo par de getter y setter para obtener o establecer los valores. Los atributos basados en objetos-valor tienen un getter simple y cuentan con tres setters, uno para realizar la introducción mediante valores simples, otro mediante un objeto-valor del mismo tipo y el último con un objeto `JsonNode`. Para los atributos de tipo enumerado se tiene un getter sencillo y dos setters, un setter para establecer los valores mediante un objeto del mismo tipo que el enumerado y otro para hacerlo mediante tipos de datos simples.

Además, para los atributos que se construyen a partir de las entidades que representan un *evento* se tienen un getter y un setter sencillos, acompañado métodos para incrementar o decrementar la lista que conforma el atributo *events*.

Por último, se tienen dos métodos `toJson` y `toString` para formatear los atributos de la clase en formato `Json` o formato `String` respectivamente.

## Enumerados

Los enumerados creados representan: estado de navegación del buque, tipo de buque, tipo de origen de las señales AIS, tipo de puerto, servicios que ofrece un puerto y roles que puede tener un usuario.

Los enumerados son sencillos, implementado los valores que pueden tener junto con un método para obtener el valor de forma aleatoria.

## Objetos-valor

La utilización de objetos-valor es bastante importante en el proyecto, ya que nos permiten definir nuestros propios tipos, lo que nos proporciona una capa adicional de seguridad al poder utilizar validaciones propias.

Los objeto-valor creados son para: países, nombres propios, correos electrónicos, contraseñas, teléfonos, geolocalización (latitud y longitud), conjunto de tiempo estimado de llegada (estimado, AIS, y predicción), código LOCODE, capacidad de un puerto (atraques), identificadores de un buque (IMO y MMSI), velocidad de un buque, medidas de un buque (eslora, ancho y calado), datos de navegación de un buque (curso y rumbo).

Todos los objetos-valor siguen el mismo esquema. Primero se definen las constantes usadas para realizar las validaciones y los atributos que conforma el objeto-valor. Luego se crea dos constructores, uno que recibe tipos de datos sencillo y otro constructor que recibe objetos de tipo JSON. En ambos casos primero se llaman a las validaciones y si todo ha ido bien se termina de construir el objeto-valor.

También se tienen los métodos más importantes, que son las funciones requeridas para hacer las validaciones necesarias. En caso de utilizar el constructor mediante JSON, primero se comprueba que valores recibidos están definidos o se encuentran en el formato correcto, posteriormente se utilizan validaciones más específicas.

Adicionalmente, cada objeto-valor tienen también un método para realizar la sobrecarga del método equals y una función para obtener un objeto aleatorio de la entidad objeto-valor.

Por último, tienen métodos getter para cada atributo definido y un método toString. Es importante que las clases que definen los objeto-valor no tengan ningún setter ya que la asignación de los valores se tienen que realizar mediante los constructores, con el fin de obligar a pasar por las validaciones impuestas y ser inmutables para evitar inconsistencias.

## Factory

Las entidades que utilizan el patrón de diseño Factory son *Port* y *Vessel*. Las clases factory son sencillas debido a que su principal objetivo es ofrecer objetos aleatorios de una misma entidad. Es por este motivo que solo tienen implementado un método *random* que construye un objeto de la entidad correspondiente mediante atributos aleatorios.

### 3.3.2. Capa de aplicación

En la capa de aplicación se encuentra implementado los casos de uso y los puertos. Cada entidad en el dominio posee una serie de casos de uso así como un puerto definido.

## Casos de uso

Los casos de uso definen las acciones que los usuarios pueden realizar en el sistema, definiéndose éstos mediante servicios. Los ficheros de servicios tiene como nombre

`<source>Service.java`, teniendo un servicio para cada entidad principal. Existen `PortService.java`, `VesselService.java` y `UserService.java`. Estos servicios generalmente son utilizados por los adaptadores y tienen los métodos básicos de las entidades: eliminar, encontrar todos, encontrar por ID, encontrar por nombre, guardar o actualizar.

Cada uno de estos métodos se encarga de llamar al método correspondiente implementado en las interfaces `<source>Repository.java` y devolver el resultado obtenido. Adicionalmente, también alojan métodos para realizar filtrados en el caso de buscar todos los elementos de una entidad mediante el método `findAll()`. En el caso de `VesselService.java` tiene un método adicional para mapear un objeto JSON a un objeto `Vessel` y `PortService.java` cuenta con métodos para generar la información de atraque de un buque.

Se tiene un fichero `WebSocketService.java` que sirve para realizar el handshake mediante una conexión `WebSocket` con el productor de buques. Este servicio se ejecuta automáticamente al inicializar el servidor back-end.

## Puertos

Se crea un puerto por cada entidad principal, teniendo `PortRepository.java`, `VesselRepository.java` y `UserRepository.java`. Cada fichero implementa una interfaz correspondiente a cada entidad, teniendo todas las interfaces los mismos métodos: borrar, guardar, encontrar todos, encontrar por ID y encontrar por nombre. Estos métodos definidos por las interfaces posteriormente son implementados por los adaptadores.

### 3.3.3. Capa de adaptadores

En esta capa se implementan los siguientes adaptadores: *API REST*, *MongoDB* y *WebSocket*.

#### Adaptador API REST

Adaptador que permite ofrecer los servicios del back-end a través de una API REST <sup>7</sup>. Con ello se consigue una abstracción total entre el back-end y los posibles clientes del mismo, ya que toda comunicación se hace mediante peticiones HTTP. Este esquema de comunicación se puede observar en la Figura 3.8.

En el adaptador cada entidad tiene implementado su respectivo ENDPOINT, de tal forma que el cliente pueda consultar o guardar nuevos datos mediante peticiones HTTP. En concreto se han creado controladores para *Vessels*, *Ports* y *Users*. Se tiene en cuenta que cada controlador, como mínimo, tiene que: obtener todos los objetos de una entidad, obtener un objeto concreto por su ID o por su nombre (peticiones GET), insertar nuevos objetos (peticiones POST) y modificar los objetos existentes (peticiones PUT). Adicionalmente, el controlador `PortHandler.java` tiene un endpoint para acceder a los métodos que generan la información de atraque.

Los controladores se encuentran en los ficheros `PortHandler.java`, `UserHandler.java` y `VesselHandler.java`. Para poder realizar los métodos ya comentados se utilizan las fun-

<sup>7</sup><https://www.redhat.com/es/topics/api/what-is-a-rest-api>



ciones implementadas en los servicios. Por ejemplo, el back-end al recibir una petición de obtener todos los buques almacenados debe ejecutar el método GET correspondiente al endpoint de *VesselHandler.java*. y ejecutar el servicio *this.vesselService.findAll()* para obtener la lista de barcos y responder a la petición.

El adaptador API REST también incluye una serie de ficheros para el manejo de JSON, como por ejemplo:

- *JsonFields.java*. Guarda las constantes de los campos a utilizar para mapear JSON.
- *<source>Deserializer.java*. Clases para realizar deserializaciones.
- *<source>Validator.java*. Clases para realizar validaciones en los controladores.

## Adaptador WebSocket

Un aspecto fundamental de la aplicación es como se consiguen los datos, en concreto los datos de los buques mercantes, ya que estos se encuentran en continua actualización. Para esto se ha decidido contar con un productor de barcos que genera buques aleatorios, los cuales realizan un recorrido desde un puerto de salida hasta un puerto de destino. Destacar que en cualquier momento es posible sustituir el productor de barcos actual y cambiarlo por uno real basado en las tecnologías nombradas en el capítulo 2, ya sea por mensajes AIS o por satélite.

Al igual que en un caso real, los buques cambian constantemente su geolocalización, su velocidad o su estado. Si la comunicación con el productor de barcos se hiciera mediante API REST, en cada intervalo de tiempo el back-end tendría que realizar peticiones al productor, con la posible pérdida de información en el proceso.

Por este motivo la comunicación con el productor se realiza mediante una conexión WebSocket<sup>8</sup>, implementando el adaptador correspondiente. Las conexiones WebSocket permiten mantener abierto un canal de comunicación de forma indefinida entre cliente y servidor, siendo en este caso back-end y productor de barcos respectivamente.

Como se aprecia en la Figura 3.8, al contrario que las conexiones API REST, con la conexión WebSocket se realiza una primera llamada para establecer el canal bidireccional de comunicación para, posteriormente, mantenerse escuchando de forma permanente.

Para implementar la conexión WebSocket se desarrolla un adaptador WebSocket, en el cual se implementa un controlador/manejador propio que hereda del *WebSocketHandler* que sería el por defecto para realizar este tipo de conexiones.

El controlador WebSocket tiene los siguientes métodos:

- *afterConnectionEstablished()*: Que se ejecuta automáticamente al establecer una conexión y comunica que la conexión se ha realizado con éxito.
- *handleMessage()*: Se ejecuta cuando recibe un mensaje, de esta forma se pueden realizar el procedimiento adecuado para procesar el mensaje recibido. En este caso,

---

<sup>8</sup><https://spring.io/guides/gs/messaging-stomp-websocket/>

al recibir un buque mercante comprueba si existe ya en la base de datos, si no existe utiliza el método `vesselService.saveVessel()` y en caso contrario `vesselService.updateVessel()`. Se aprovecha también que los barcos guardan los datos de los puertos para registrar nuevos puertos y actualizar los eventos de los mismos.

- `handleTransportError()`: Se ejecuta cuando existe un error en la comunicación.
- `afterConnectionClosed()`: Se invoca después de que cualquiera de los lados haya cerrado la conexión WebSocket o después de que se haya producido un error de transporte.
- `supportsPartialMessages()`: Se ejecuta si el WebSocket tiene que manejar mensajes parciales.

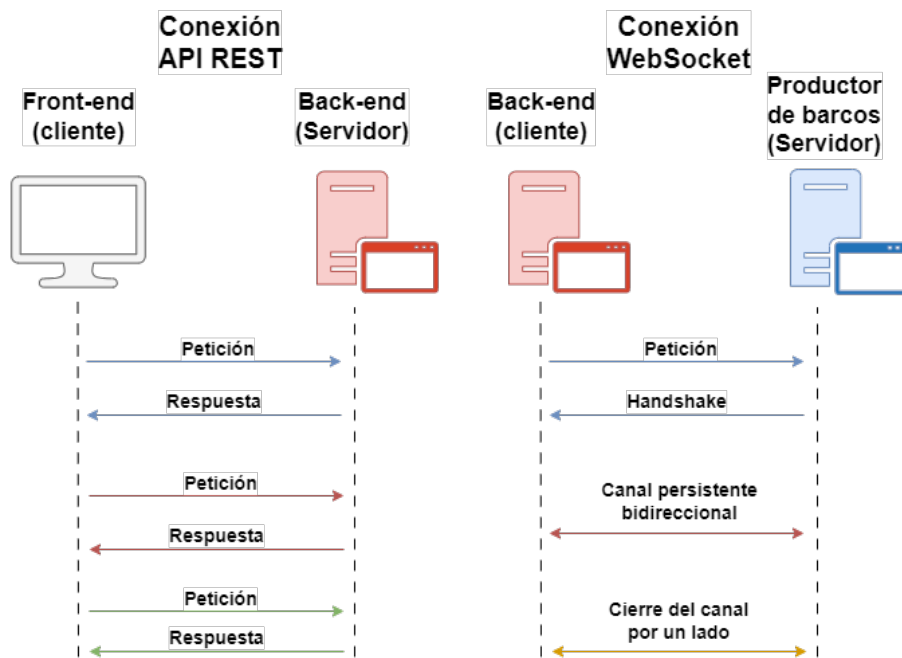


Figura 3.8: Esquemas conexión API y WebSocket

## Adaptador MongoDB

El adaptador de MongoDB permite conectar y realizar las operaciones de persistencia y manejo de datos en base a las entidades creadas. De la misma forma que el adaptador API REST, se crea un fichero individual para cada entidad. Las operaciones permitidas para cada entidad son búsquedas, inserciones, eliminaciones y actualizaciones.

Se ha elegido MongoDB ya que ofrece gran flexibilidad y escalabilidad al ser una base de datos NoSQL no tiene un esquema rígido predefinido, pudiendo almacenar diferentes tipos de datos y estructuras en un mismo documento. Además permite realizar consultas complejas y hacer inserciones o actualizaciones en tiempo real, fundamental para los objetivos que se requieren para el proyecto.

Para implementarlo se tiene el adaptador MongoDB, con un fichero para cada entidad principal, coincidiendo también con los puertos creados, teniendo `MongoDbPortRepository.java`, `MongoDbUserRepository.java` y `MongoDbVesselRepository.java`. Cada uno

de estos ficheros implementa los métodos de las interfaces de sus correspondientes `<source>Repository.java` y, por tanto, desarrollan las funciones de borrado, encontrar todos, encontrar por ID, encontrar por nombre y guardado. Cada uno de estos métodos utilizan los métodos de MongoDB para así poder acceder y usar la base de datos.

A nivel global, para realizar operaciones el back-end hace el siguiente recorrido: En el momento de recibir una petición que llega a los endpoints implementado en los `<resource>Handler.java` del adaptador de salida de API REST, estos métodos utilizan los servicios `<source>Service.java` de la capa de aplicación. Estos servicios a su vez llaman a los métodos definidos por las interfaces `<source>Repository.java` que son implementados por el adaptador de entrada de MongoDB para así guardar, extraer, modificar o borrar los datos en la base de datos. Una vez terminado de ejecutarse la función del adaptador MongoDB, el método del controlador que ha recibido la petición retorna su respuesta al cliente.

### 3.4. Front-end, diseño e implementación



Figura 3.9: Diagrama Front-end

Como se comenta en los apartados anteriores, el front-end se encuentra diseñado siguiendo la implementación de una arquitectura hexagonal. Como se aprecia en la Figura 3.9, a diferencia del diseño del back-end, en el front-end no se implementa el patrón factory en la capa de dominio. Por otro lado, en la capa de adaptadores solo hay dos: el adaptador API REST para realizar las peticiones al back-end y el adaptador Vue para implementar la interfaz de usuario.

#### 3.4.1. Capa de dominio

En la capa de dominio del front-end se implementan las entidades que se van a visualizar y las cuales son pedidas al back-end mediante peticiones API REST. Destacar que estas entidades son equivalentes a las entidades back-end, con el fin de facilitar la integridad de la aplicación permitiendo un mapeo de objetos más sencillo, eficaz y reutilizable. La

diferencia entre esta capa de dominio y la existente en el back-end es que en el front-end no se encuentra implementado el patrón Factory, pero se sigue teniendo las entidades, enumerados y objetos-valor.

### 3.4.2. Capa de aplicación

En la capa de aplicación se alojan los puertos y servicios necesarios para implementar la lógica de las entidades. En concreto se incluyen los servicios (casos de uso) que va a utilizar posteriormente el adaptador VueJs con Pinia para solicitar los recursos, y por otro lado, los puertos implementados por el adaptador API REST para efectuar las peticiones HTTP al back-end.

#### Casos de uso

Los casos de uso se implementan mediante servicios en los ficheros *<source>Service.ts*. Estos servicios implementados mediante una clase homónima para cada entidad principal, siendo estas *Vessel*, *Port* y *User*.

Cada servicio posee un constructor, un método para obtener todos los elementos de una entidad, un método para obtener un elemento por su ID o por su nombre, así como los métodos correspondientes para poder eliminar un elemento o guardarlo. Cada método se encarga de llamar al método correspondiente definido en las interfaces *<source>Repository*.

#### Puertos

Los puertos se implementan en los ficheros *<source>Repository.ts*. Estos ficheros implementan una interfaz para definir los métodos que van a tener que ser desarrollados por los adaptadores. Para *PortRepository.ts*, *VesselRepository.ts* y *UserRepository.ts* los métodos definidos son los mismos.

### 3.4.3. Capa de adaptadores

En esta capa se incluyen los siguientes adaptadores: VueJs y API REST.

#### Adaptador API REST

Al igual que en el back-end, se tienen un adaptador API REST, pero con la diferencia que será utilizado para realizar las peticiones HTTP al servidor. Para formalizar la realización de peticiones se utiliza la API Fetch. API Fetch es una interfaz de programación que proporciona una forma de realizar solicitudes de red mediante un conjunto de métodos a través de HTTP, como GET, POST, PUT y DELETE. Estos métodos permiten realizar peticiones y recuperar respuestas en formato JSON, XML o texto plano.

Propiamente, la implementación de las peticiones GET, POST, PUT y DELETE con API Fetch consiste en desarrollar los métodos definidos en las interfaces *<source>Repository.ts* y se encargan de utilizar las peticiones implementadas en *ull-https.ts*. Adicionalmente, en

los métodos encargados de recibir objetos del back-end: `findAll()`, `findById()` y `findByName()` es necesario mapear los datos recibidos en sus entidades correspondientes.

Para el mapeado se usan las clases `<source>Response`. Estos ficheros son los encargados, con el método `to<source>()` desarrollado, de mapear los objetos recibidos en respuesta del back-end y devolver directamente las entidades.

## Adaptador VueJs

Para la implementación del front-end se ha escogido como framework principal VueJs 3<sup>9</sup>, ya que permite la creación de aplicaciones dinámicas de forma rápida y práctica. Entre sus características nos encontramos que Vue es completamente modular y se encuentra basado en componentes web, además de proporcionar reactividad y reaccionar a los eventos que se producen en el DOM. Dos aspectos destacados de Vue son la posibilidad de crear aplicaciones de una sola página, teniendo una sola página con varias vistas, así como la capacidad de controlar todo el ciclo de vida de los componentes.

Con estos aspectos Vue permite transmisiones rápidas y una comunicación entre cliente y servidor más fluida, así como liberar recursos de forma sencilla al desmontar un componente. Cabe destacar que el framework, al estar implementado en la capa de adaptadores, se puede sustituir fácilmente por otra alternativa como Angular<sup>10</sup> o React<sup>11</sup>.

Para facilitar el diseño y construcción de la interfaz de usuario se utiliza Vuetify<sup>12</sup>. Vuetify es un marco de interfaz de usuario totalmente construido sobre VueJS, el cual permite crear experiencias de usuario ricas y atractivas. Por otro lado, se encuentra diseñado desde cero para que sea fácil de aprender mediante sus componentes, que siguen las especificaciones *Material desing*<sup>13</sup>

Para implementar en su totalidad la interfaz con Vue, además de utilizar Vuetify, se emplean librerías o módulos para obtener funcionalidades adicionales y se tienen en cuenta tener una modularización completa de los componentes para que cada uno se encargue de su propia funcionalidad. De esta forma el adaptador VueJs implementa los siguientes elementos: *I18n*, *router*, *store*, *layouts*, *views* y *components*.

## I18n

Para la internacionalización de la aplicación se ha decidido utilizar Vue I18n<sup>14</sup>. Esta librería otorga un soporte completo para la pluralización, lo que permite manejar de forma efectiva diferencias gramaticales y definir reglas de pluralización para cada idioma.

En la carpeta *I18n* se implementa dos ficheros JSON: *en.json* y *es.json*. Ambos ficheros implementan el texto estático en inglés y español respectivamente que es utilizado en la aplicación. De esta forma si queremos implementar un nuevo idioma solo tendríamos

---

<sup>9</sup><https://vuejs.org>

<sup>10</sup><https://angular.io>

<sup>11</sup><https://es.react.dev>

<sup>12</sup><https://vuetifyjs.com/en/>

<sup>13</sup><https://m3.material.io/>

<sup>14</sup><https://vue-i18n.intlify.dev/>

que crear un nuevo fichero y que los campos tengan como valor la traducción al lenguaje. Con esto, los usuarios dentro de la propia aplicación ya son capaces de escoger el idioma.

## Router

En la carpeta *router* se aloja el fichero *index.ts* encargado de implementar el funcionamiento del vue-router. Esto nos permite que nueva aplicación de Vue sea de una sola página, de tal forma que en el fichero alojamos la ruta y componente a visualizar y posteriormente Vue se encarga de que todas las vistas alojadas sean dinámicas gracias a la etiqueta `<router-view>`.

Por ejemplo, tenemos el componente *PortsList* y *VesselList*, al navegar por la aplicación, se monta un componente, y al pasar al otro componente, desmontará en el que se encontraba y montará el nuevo, pero sin recargar la página entera, solo cambiará dinámicamente esa porción de la aplicación.

## Store

Se utiliza como librería de gestión de estado Pinia<sup>15</sup>. Pinia sigue el patrón Flux proporcionando una escalabilidad mayor a la hora de administrar el estado de la aplicación. Se basa en los modelos de tienda (store) centralizados que almacenan las variables y entidades, que luego se comparten entre los componentes. A su vez utiliza los principios de la programación reactiva, esto significa que cualquier cambio del estado de una entidad guardada en una store de Pinia se verá reflejado en los componentes que utilicen dicha entidad.

En la carpeta *store* se alojan dos ficheros *portStore.ts* y *vesselStore.ts*. Cada fichero implementa un store diferente, en donde se tiene un state para guardar *portArray* `Array<Port>` y *vesselArray* `Array<Vessel>` respectivamente. También tienen las acciones necesarias para solicitar todos los elementos de las entidades, obtener solo un elemento o aplicar filtro y ordenaciones.

De esta forma Pinia es el encargado de solicitar todos los buques y puertos al back-end al inicializar el front-end mediante la utilización de los servicios implementados en la capa de aplicación. Una vez obtenido todos los elementos se almacena en los state y los componentes de Vue solo tienen que solicitar los datos a Pinia mediante las acciones o accediendo directamente a *portArray* o *vesselArray*. Con esto logramos seguir el patrón flux centralizando la información y teniendo un flujo de comunicación más limpio, ya que no es necesario que cada componente solicite los recursos al back-end.

## Layouts

Implementan la visualización global de la aplicación, así como la visualización e implementación de los elementos estáticos. Existen cuatro componentes *AppBar.vue* *Footer.vue*, *View.vue* y *Default.vue*.

---

<sup>15</sup><https://pinia.vuejs.org/>

El componente *AppBar.vue* es el encargado de construir y visualizar barra de navegación, se aprovecha la utilización de la etiqueta `<v-app-bar>` de Vuetify para crear este elemento estático.

El componente *Footer.vue* construye y visualiza el pie de página de forma estática. Se utiliza la etiqueta de vuetify `<v-footer>` para construir el footer e incorpora botones para poder cambiar el idioma mediante Vue I18n.

El componente *View.vue* solo tiene la etiqueta `<v-main>` de vuetify y anidada la etiqueta `<router-view>` para poder mostrar las visualizaciones dinámicas mediante vue-router. Posteriormente, el componente *Default.vue* importa los demás componentes y los visualiza en el siguiente orden: barra de navegación, contenido dinámico y pie de página.

## Views

Son componentes cuya funcionalidad es importar y visualizar otros componentes. Estos componentes usados para mostrar vistas son los que se incorporan luego en el fichero *router/index.ts* para realizar los enrutamientos y tener una aplicación de una sola página.

Al realizarlo de esta forma podemos desacoplar la construcción de los componentes de su propia visualización.

## Components

Son los componentes que construyen la interfaz de usuario. Tenemos los siguientes grupos de componentes:

### *Filters*

En la carpeta *filters* se implementan componentes para realizar filtrados, hay un componente por cada filtro que se requiera y todos se implementan de la misma forma. Cada uno construye su propio campo para insertar o escoger valores y son utilizados como componentes hijos desde otros componentes. Para esto último se usa *defineProps* y *defineEmits*, de forma que los componentes padres le pasan una variable mediante el *defineProps* y el componente hijo emite un evento con *defineEmits* si se modifica el valor de su campo.

### *Pagination*

Se tiene un componente *PaginationComponent.vue* para realizar paginaciones, así como permitir el cambio de tamaño en los resultados a mostrar. Se implementa de forma independiente como un componente hijo mediante *defineProps* y *defineEmits* para poder ser reutilizado en aquellos componentes que lo necesiten.

### *Home*

Para construir la página de inicio de la aplicación se utiliza dos componentes. El componente *HomeComponent.vue* que es el utilizado por la vista *HomeView.vue* y que a su vez

implementa *CardsComponent.vue*. Este último componente es el encargado de construir tres cards mediante la etiqueta `<v-card>` de vuetify, y los cuales son usados para mostrar información de las funcionalidades de la página.

### *Tracking*

En el apartado de tracking es donde se implementa la funcionalidad de Live Tracker. El componente principal es *TackingComponent.vue*, y al construirse con *onMounted()* realiza una petición a Pinia para obtener todos los buques y puertos registrados cada cierto tiempo. Con los buques y puertos obtenidos estos se muestran en un mapa interactivo.

Para realizar la visualización en tiempo real del movimiento de los barcos y la ubicación de los puertos se utiliza la librería de javascript Leaflet<sup>16</sup>. Leaflet es una librería open source que permite crear y visualizar mapas interactivos para aplicaciones web de forma sencilla, liviana y eficiente.

Implementa también como componentes hijos varios filtros de *filters*, el *PaginationComponent.vue* y *TrackingTableComponent.vue*. Este último componente es utilizado para visualizar en una tabla los datos de los buques, ya que únicamente se utiliza para la visualización de datos solo tiene definido *defineProps*.

### *Ports*

El grupo de componentes de *ports* se divide en dos apartados, uno para mostrar la lista de puertos y otro para mostrar un puerto individual.

Para mostrar la lista de puertos se utiliza el componente *PortListComponent.vue*, este componente es el que se utiliza en la vista *PortListView.vue*, y al montarse con *onMounted()* solicita la lista de puertos a Pinia. Implementa a su vez varios filtros de *filters*, *PaginationComponent.vue* y un componente hijo propio *PortTableComponent.vue*. *PortListComponent.vue* pasa mediante *defineProps* a *PortTableComponent.vue* la lista de puertos y el componente hijo es el encargado de mostrar en una tabla los datos de cada puerto.

Para mostrar un puerto de forma individual se utiliza *PortComponent.vue*, que es usado por *PortView.vue* para mostrarlo. Al construirse solicita el puerto en concreto a Pinia, los buques que tienen pendiente llegar o que han salido y la información de atraque de los buques que tienen como destino o se encuentran en el puerto. Con estos datos muestra en el mapa interactivo la ubicación del puerto y buques correspondientes.

Adicionalmente *PortComponent.vue* utiliza varios componentes hijos para mostrar toda la información, gran parte de estos mediante una `<v-tab>` de Vuetify. Estos componente son:

- *PortDataComponent.vue*: Recibe un objeto *Port* para luego mostrar la información del mismo en una `<v-card>` de Vuetify.

---

<sup>16</sup>leafletjs.com



- *PortState*: Utilizado en el `<v-tab>`, implementa un diagrama de gantt usando `vue-ganttastic` para mostrar las próximas llegadas, que atraque van a usar y durante cuanto tiempo.
- *PortHistory*: Utilizado en el `<v-tab>`, implementa una `v-table` para representar todo el histórico de llegadas, en puerto y salidas que ha tenido el puerto.
- *PortArrivalHistory.vue*: Utilizado en el `<v-tab>` y recibiendo el array de eventos de llegada de un puerto, implementa una `v-table` para las próximas llegadas que tiene el puerto.
- *PortInPortHistory.vue*: Utilizado en el `<v-tab>` y recibiendo el array de eventos de buques en puerto, implementa una `v-table` para mostrar los buques que se encuentra actualmente en puerto.
- *PortDepartureHistory.vue*: Utilizado en el `<v-tab>` y recibiendo el array de eventos de salidas, implementa una `v-table` para mostrar todas las salidas que ha tenido el puerto.

## Vessels

El grupo de componentes de *vessels* se divide en dos apartados, uno para mostrar la lista de buques y otro para mostrar un buque de forma individual.

Para mostrar la lista de buques se utiliza el componente *VesselListComponent.vue*, siendo este el que se utiliza en la vista *VesselListView.vue* y solicita la lista de barcos a Pinia al montarse con `onMounted()`. Implementa a su vez varios filtros de *filters*, *PaginationComponent.vue* y un componente hijo propio *VesselTableComponent.vue*. *VesselListComponent.vue* pasa mediante `defineProps` a *VesselTableComponent.vue* la lista de buques y el componente hijo es el encargado de mostrar en una tabla los datos de cada buque.

Para mostrar un buque de forma individual se utiliza *VesselComponent.vue*, que es usado por *VesselView.vue* para mostrarlo. Al construirse solicita el buque en concreto a Pinia, en caso de no encontrarlo Pinia solicita el buque al back-end. Con el buque a disposición del componente, se aprovecha para mostrarlo en el mapa interactivo y transmitir la información a los componentes hijos. Estos componentes hijos son:

- *VesselDataComponent.vue*: Recibe un objeto *Vessel* para luego mostrar la información de buque en una `<v-card>` y los datos de navegación en otra `<v-card>`.
- *PortDataComponent.vue*: Utilizado en el `<v-tab>` de *VesselComponent.vue*, recibe un objeto *Port* para luego mostrar la información del puerto en una `<v-card>`. Se utiliza para representar tanto el puerto de salida como el puerto de destino.
- *VesselHistoryComponent.vue*: Utilizado también en el `<v-tab>`, recibe el array de eventos del buque para implementar una `<v-table>` que muestra toda la información y así representar el histórico de eventos.

# Capítulo 4

## Desarrollo

### 4.1. Pruebas

En esta sección se organizan las diferentes pruebas para testear ambas partes de la aplicación, back-end y front-end. Las pruebas de software son fundamentales ya que nos permiten identificar errores, mejorar la calidad del software y garantizar una funcionalidad adecuada. A nivel general se tienen pruebas unitarias, de integración y end-to-end.

#### 4.1.1. Pruebas del back-end

Para realizar las pruebas en el back-end se cuenta con dos tipos de pruebas, pruebas unitarias para probar de forma aislada clases definidas en la capa de dominio y pruebas de integración para comprobar que funcionan de forma correcta diferentes componentes en conjunto.

##### **Pruebas unitarias**

Se tienen pruebas unitarias para todos los objetos-valor definidos en la capa de dominio. Estos test se realizaron con JUnit, marco de pruebas unitarias para el lenguaje de programación Java que permite escribir y ejecutar pruebas automatizadas para garantizar que las funciones y clases tengan un comportamiento correcto. Un ejemplo de ejecución de estas pruebas se muestra en la Figura 4.1.

```

[INFO] -----
[INFO] TESTS
[INFO] -----
[INFO] Running com.vessels.domain.valueobject.VesselMeasurementTests
[INFO] Tests run: 13, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.034 s - in com.vessels.domain.valueobject.VesselMeasurementTests
[INFO] Running com.vessels.domain.valueobject.LocodeTests
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s - in com.vessels.domain.valueobject.LocodeTests
[INFO] Running com.vessels.domain.valueobject.GeoLocationTests
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 s - in com.vessels.domain.valueobject.GeoLocationTests
[INFO] Running com.vessels.domain.valueobject.NameTests
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.011 s - in com.vessels.domain.valueobject.NameTests
[INFO] Running com.vessels.domain.valueobject.CountryTests
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.009 s - in com.vessels.domain.valueobject.CountryTests
[INFO] Running com.vessels.domain.valueobject.PhoneTests
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.004 s - in com.vessels.domain.valueobject.PhoneTests
[INFO] Running com.vessels.domain.valueobject.EmailTests
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.007 s - in com.vessels.domain.valueobject.EmailTests
[INFO] Running com.vessels.domain.valueobject.PortCapacityTests
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.005 s - in com.vessels.domain.valueobject.PortCapacityTests
[INFO] Running com.vessels.domain.valueobject.PasswordTests
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.004 s - in com.vessels.domain.valueobject.PasswordTests
[INFO] Running com.vessels.domain.valueobject.EToATests
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.013 s - in com.vessels.domain.valueobject.EToATests
[INFO] Running com.vessels.domain.valueobject.VesselIdentifierTests
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.014 s - in com.vessels.domain.valueobject.VesselIdentifierTests
[INFO] Running com.vessels.domain.valueobject.VesselNavigationTests
[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.003 s - in com.vessels.domain.valueobject.VesselNavigationTests
[INFO] Running com.vessels.domain.valueobject.SpeedTests
[INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.004 s - in com.vessels.domain.valueobject.SpeedTests
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 113, Failures: 0, Errors: 0, Skipped: 0
[INFO]

```

Figura 4.1: Resultados test unitarios

Los test creados para cada objeto-valor comprueban que se crean correctamente y que fallen según las validaciones puestas, de esta forma nos cercioramos que las validaciones funcionan. Además de comprobar las validaciones, se testea los métodos para igualar, obtener los códigos hash o para convertir a String.

## Pruebas de integración

Para realizar las pruebas de integración se utiliza JUnit y así garantizar la funcionalidad y calidad de la API REST. Se cubren las principales funcionalidades de la API con test, incluyendo:

- Verificación de los endpoints y su correcta respuesta a las peticiones HTTP (códigos de respuesta, headers y cuerpo).
- Prueba de los casos de éxito y los casos de error para cada endpoint.

Se tienen pruebas de integración para los siguientes controladores:

- PortHandler

```

[INFO] Tests run: 11, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.287 s - in com.vessels.adapter.rest.controller.PortHandlerIntegrationTest
[INFO]

```

Figura 4.2: Resultados PortHandlerIntegrationTest

- VesselHandler

```

[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 3.78 s - in com.vessels.adapter.rest.controller.VesselHandlerIntegrationTest
[INFO]

```

Figura 4.3: Resultados VesselHandlerIntegrationTest

- UserHandler

```

[INFO] Tests run: 9, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.14 s - in com.vessels.adapter.rest.controller.UserHandlerIntegrationTest
[INFO]

```

Figura 4.4: Resultados UserHandlerIntegrationTest

### 4.1.2. Pruebas del front-end

En el front-end se tienen pruebas end-to-end para evaluar el comportamiento del sistema en su totalidad, desde el inicio hasta el final de un proceso.

#### Pruebas end-to-end

Para crear las pruebas end-to-end se utiliza Cypress, la cual es una herramienta en JavaScript open source que permite configurar, escribir y ejecutar pruebas, así como pruebas de depuración.

Se tienen dos ficheros para comprobar el funcionamiento de la aplicación en varios aspectos.

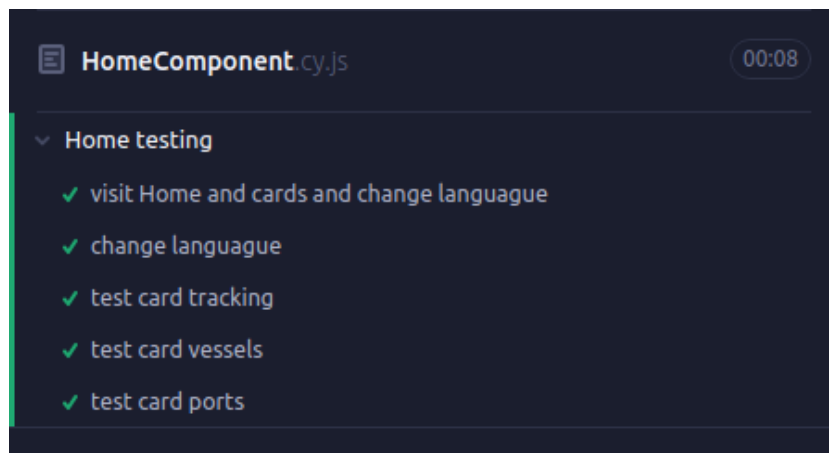


Figura 4.5: Resultados test end-to-end HomeComponent

Por un lado, se tiene *HomeComponent.cy.js*, cuyos resultados se muestran en la Figura 4.5 y que posee el siguiente procedimiento:

- Acceder a la aplicación en en la página de inicio.
- Comprobar los títulos de los títulos de la página de inicio en español.
- Cambiar el idioma a inglés.
- Comprobar los títulos de la página de inicio en inglés.
- Cambiar el idioma a español.
- Comprobar que las cards implementadas llevan a la opción correspondiente.
- Revisar el contenido de cada opción.
- Volver a la página de inicio mediante los breadcrumb-trail y comprobar que se vuelve a la página de inicio de forma satisfactoria.

Por otro lado, se cuenta con *PortComponent.cy.js*, cuyo resultado se muestra en la Figura 4.6 y que realiza los siguientes pasos:

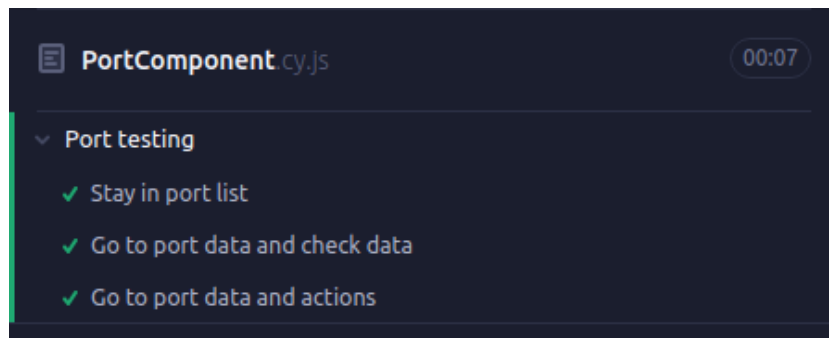


Figura 4.6: Resultados test end-to-end PortComponent

- Acceder a la aplicación directamente en la lista de puertos.
- Probar que el filtro por nombre funciona escribiendo el nombre del puerto.
- Acceder al puerto en concreto desde la tabla y comprobar que se encuentran los mismos del puertos.
- Repetir el procedimiento pero ahora comprobar que las diversas acciones funcionen. Utilizar la opción de localización del puerto y que las ventanas implementadas muestren el contenido.

## 4.2. Documentación

Para documentar la API REST se utiliza OpenAPI. OpenAPI es una especificación de API que describe cómo interactuar con una API RESTful de forma programática. En formato JSON o YAML describe todos los endpoints, parámetros, tipos de datos, métodos de solicitud (como GET, POST, PUT, DELETE) y respuestas de una API.

Después de implementar las librerías y comentar los endpoints del back-end con el formato adecuado, de forma automática se genera la documentación. Esto permite ver la documentación en JSON de forma directa o ver la documentación de la API mediante Swagger. La Figura 4.7 muestra un extracto de la documentación generada.

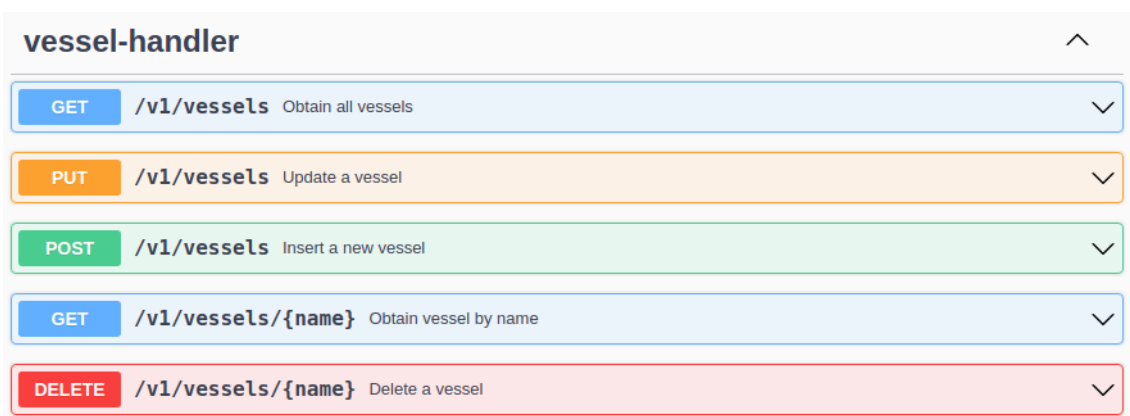


Figura 4.7: Documentación en Swagger

### 4.3. Despliegue

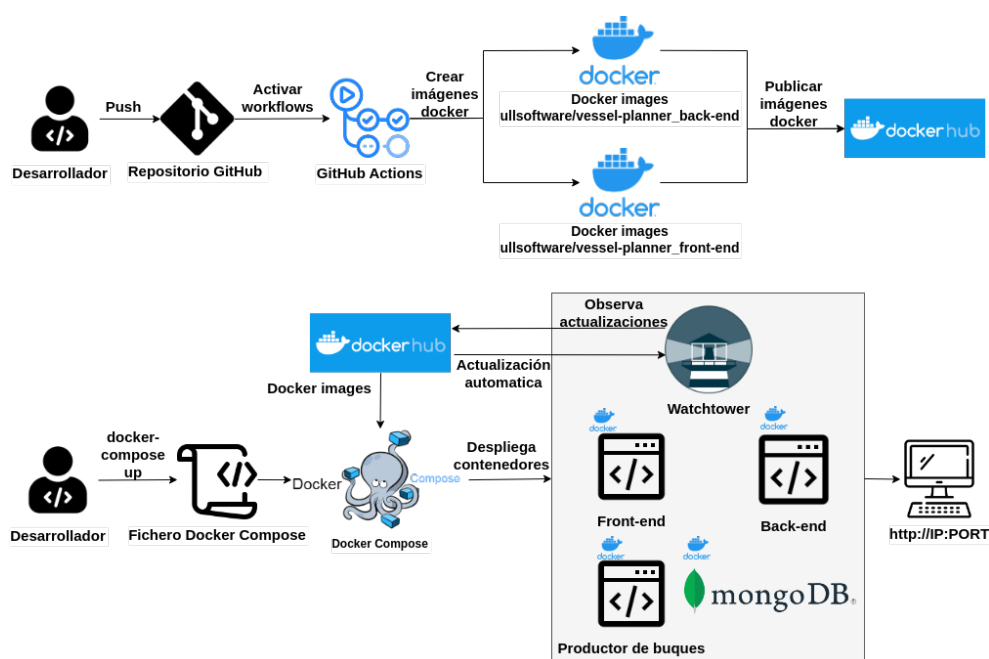


Figura 4.8: Diagrama de despliegue

El despliegue gira en torno a la utilización de Docker, teniendo que todos los componentes de la aplicación son contenedores docker. Docker permite eliminar las tareas de configuración repetitivas utilizadas en el ciclo de vida de desarrollo de un proyecto para conseguir que sea más rápido, sencillo, escalable y portátil.

Como se aprecia en la Figura 4.8 el despliegue de la aplicación consiste en dos partes. En primera instancia, cada vez que se realiza un commit en el repositorio de trabajo se activan las GitHub Actions incorporadas. Las GitHub Actions permiten realizar una integración continua en el proyecto, así como una entrega continua (CI/CD). Las GitHub Actions implementadas son:

- *front-end\_publish-docker-image.yml*: Crea una imagen Docker basada en el front-end y la publica en Docker Hub<sup>1</sup>.
- *back-end\_publish-docker-image.yml*: Crea una imagen Docker basada en el back-end y la publica en Docker Hub<sup>2</sup>.

Con ambas imágenes publicadas en Docker Hub cualquier usuario puede descargarlas y utilizarlas con un despliegue sencillo. Además, se cuenta con un fichero *docker-compose.yml* que permite desplegar la aplicación. Docker Compose es una herramienta que nos permite definir y ejecutar aplicaciones Docker multicontenedores. Con las opciones del *docker-compose*, en caso de no tener descargadas las imágenes las descarga de Docker Hub, permite desplegar por separado front-end y back-end o a la vez. Cuenta también con Watchtower para actualizar en caliente las imágenes y contenedores mientras se encuentran desplegados. Adicionalmente el fichero *docker-compose.yml* permite descargar y desplegar un contenedor de MongoDB y un productor de buques.

<sup>1</sup>[https://hub.docker.com/r/ullsoftware/vessel-planner\\_front-end](https://hub.docker.com/r/ullsoftware/vessel-planner_front-end)

<sup>2</sup>[https://hub.docker.com/r/ullsoftware/vessel-planner\\_back-end](https://hub.docker.com/r/ullsoftware/vessel-planner_back-end)

## 4.4. Producto final

Los usuarios tienen acceso a diferentes funcionalidades o casos de usos. Para poder acceder a estas funcionalidades tienen a su disposición la barra de navegación.



Figura 4.9: Barra de navegación

La barra de navegación aparece en todo momento durante la utilización de la aplicación siendo un elemento estático. Como se observa en la Figura 4.9 se tienen los siguientes apartados:

- *Inicio*. Permite acceder al homepage.
- *Live tracker*. Permite acceder a la funcionalidad de monitorizar el tráfico marítimo.
- *Buques*. Permite acceder al caso de uso de comprobar el estado de un buque.
- *Puertos*. Permite acceder al caso de uso de comprobar el estado de un puerto.
- *Panel de control*. Permite acceder al caso de uso de visualización de estadísticas globales.
- *Perfil de usuario*. Al seleccionar esta opción se despliega las opciones de usuario. Se observa en la Figura 4.10 que el desplegable se encuentra conformado por:
  - *Acceder al perfil*. Muestra el perfil de usuario.
  - *Cerrar sesión*. Cierra la sesión activa.
  - *Iniciar sesión*. Inicia una sesión nueva.
  - *Registrarse*. Permite registrar un nuevo usuario.

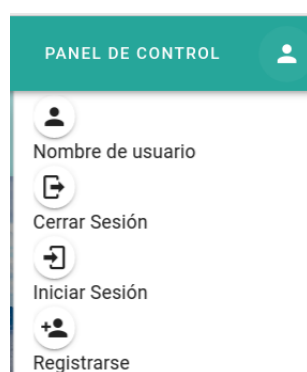


Figura 4.10: Opciones de usuario

### 4.4.1. Cambiar el idioma

Los usuarios pueden encontrar las opciones para cambiar el idioma de la aplicación en el pie de página de la web.



Figura 4.11: Pie de página

El pie de página, de la misma forma que la barra de navegación es un elemento estático que puede ser accedido desde cualquier punto de la aplicación web. Como se muestra en la Figura 4.11 los usuarios pueden elegir el idioma, actualmente pudiendo escoger entre español e inglés. En el pie de página también tenemos dos apartados adicionales, una sección donde se reflejan los datos de contacto y otra sección para alojar las redes sociales asociadas al proyecto.

#### 4.4.2. Monitorizar el tráfico marítimo (Live tracker)

Permite ver en tiempo real los barcos reflejados en un mapa interactivo. Como muestra la Figura 4.12 se tiene el mapa interactivo en el cual se pueden visualizar los buques y puertos registrados, estos buques se pueden filtrar para visualizar solo los barcos que deseados. Los campos de filtrado son para el nombre del buque, IMO o MMSI, puerto de destino y estado de navegación. Se tienen dos opciones adicionales para elegir si mostrar o no los puertos y la ruta de navegación. Además, al hacer click encima del marcador de un puerto o buque se muestra parte de su información. De esta forma se consigue de forma rápida conocer la ubicación de cada buque y comprender el panorama general.

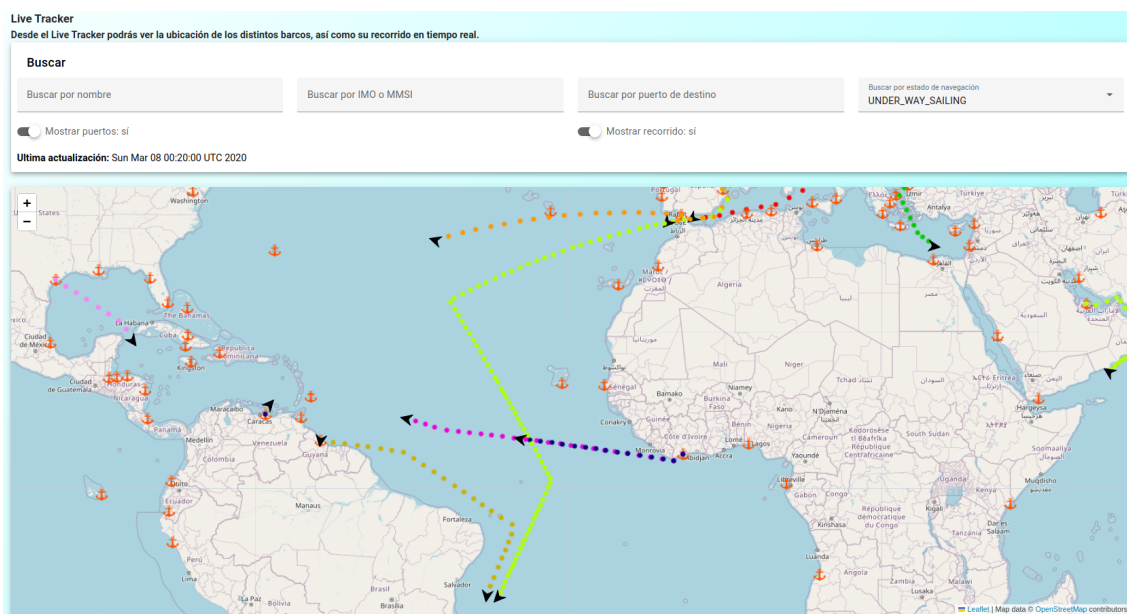


Figura 4.12: Live tracker mapa

Acompañando al mapa tenemos la tabla mostrada en la Figura 4.13, que muestra los detalles de cada uno de los barcos bajo seguimiento. Esta tabla muestra el nombre del barco, desde el cual es posible acceder a toda la información del buque en concreto, los identificadores IMO y MMSI, el puerto de destino, la fecha de llegada estimada y el



tiempo restante para que se efectuó la llegada. Adicionalmente se cuenta con un botón para ver en el mapa solo el buque escogido.

Nombre	IMO	MMSI	Puerto de llegada	ETA predecido	Tiempo hasta la llegada	Ver en el mapa
<a href="#">MSC_VIRTUOSA_RDFLd</a>	82540	581729027	URUGUAY MONTEVIDEO	24/8/2023 1:15:37	51d 9h 29m 19s	<a href="#">VER</a>
<a href="#">COLUMBUS_sJSpR</a>	717063	655519261	UNITED STATES CRYSTAL RIVER	17/8/2023 1:47:56	44d 10h 1m 38s	<a href="#">VER</a>
<a href="#">COSTA_FASCINOSA_IKDPk</a>	630304	403802220	URUGUAY MONTEVIDEO	24/8/2023 16:39:27	52d 53m 9s	<a href="#">VER</a>
<a href="#">OPEN_ARMS_lwORO</a>	301875	763469878	MOROCCO AGADIR	2/10/2023 9:3:56	90d 17h 17m 38s	<a href="#">VER</a>
<a href="#">JOHN_GMUNSON_AXmtX</a>	821158	641066556	BULGARIA VARNA	9/8/2023 15:57:48	37d 11m 30s	<a href="#">VER</a>
<a href="#">ECLIPSE_RZMsl</a>	105030	464054816	ISRAEL ACRE	10/8/2023 8:57:30	37d 17h 11m 12s	<a href="#">VER</a>
<a href="#">NORWEGIAN_SPIRIT_CBVHb</a>	552140	893226104	EGYPT EL ISKANDARIYA ALEXANDRIA	12/8/2023 8:10:53	39d 16h 24m 35s	<a href="#">VER</a>
<a href="#">AIDASOL_pCCK</a>	681768	303810106	PORTUGAL VILA DO PORTO	16/8/2023 22:59:13	44d 7h 12m 55s	<a href="#">VER</a>
<a href="#">PELORUS_pgNCZ</a>	399064	706382980	MALAYSIA PENANG GEORGETOWN	28/8/2023 11:21:32	55d 19h 35m 14s	<a href="#">VER</a>
<a href="#">NORRONA_DHNhY</a>	867805	709938133	INDIA DHAMRA	18/8/2023 18:55:1	46d 3h 8m 43s	<a href="#">VER</a>

< 1 2 3 4 5 >  
 Resultados totales: 414

Resultados por página: 100      Ir a la página: 1

Figura 4.13: Live tracker tabla

Por último tenemos opciones para cambiar la página de la tabla, y de la misma forma, cambiar la cantidad de buques.

#### 4.4.3. Comprobar el estado de un buque

Caso de uso accesible desde la opción *Buques* del menú principal. De forma parecida a otros casos de uso, muestra en una tabla todos los buques registrados en la aplicación. En la Figura 4.14 podemos ver filtros para realizar búsquedas sobre el nombre del buque, sus identificadores IMO o MMSI, por el tipo de buque mercante o por su estado de navegación actual. También tenemos un botón para acceder a la opción de live tracker con la visualización del barco en concreto. Por último, en cada registro tenemos un enlace para ver toda la información de un buque.

**Buques**  
Desde aquí podrás ver los distintos barcos, el tipo de buque y su estado actual, así cómo acceder a cada buque individualmente.

**Buscar**

Buscar por nombre     Buscar por IMO o MMSI     Buscar por tipo de barco     Buscar por estado de navegación

Última actualización: Thu Jan 16 23:20:00 UTC 2020

Nombre	IMO	MMSI	Tipo	Estado	Ver en el mapa
<a href="#">ABEILLE_BOURBON_IASK</a>	857283	870382017	HIGH_SPEED_CRAFT_HSC_RESERVED_FOR_FUTURE_USE_46	MOORED	<a href="#">VER</a>
<a href="#">ABEILLE_BOURBON_TIRet</a>	461374	426074477	SPARE_LOCAL_VESSEL_56	MOORED	<a href="#">VER</a>
<a href="#">ABEILLE_BOURBON_xSTvd</a>	914179	174559960	RESERVED_39	MOORED	<a href="#">VER</a>
<a href="#">ADVENTURE_OF_THE_SEAS_SaFFO</a>	715555	244178144	WING_IN_GROUND_WIG_HAZARDOUS_CATEGORY_A	MOORED	<a href="#">VER</a>
<a href="#">AEGEAN_GODESS_pmfmc</a>	328173	167082786	HIGH_SPEED_CRAFT_HSC_RESERVED_FOR_FUTURE_USE_48	MOORED	<a href="#">VER</a>
<a href="#">AGE_OF_UNION_kVvW</a>	702649	632915055	PASSENGER_RESERVED_FOR_FUTURE_USE_66	MOORED	<a href="#">VER</a>
<a href="#">AIDABELLA_VvixC</a>	790578	591436440	PASSENGER_HAZARDOUS_CATEGORY_B	MOORED	<a href="#">VER</a>
<a href="#">AIDABLU_RMyQd</a>	305000	818103808	PASSENGER_HAZARDOUS_CATEGORY_C	MOORED	<a href="#">VER</a>
<a href="#">AIDABLU_RXgga</a>	993950	281776811	DREDGING_OR_UNDERWATER_OPS	MOORED	<a href="#">VER</a>
<a href="#">AIDABLU_TalBi</a>	905840	605912972	WING_IN_GROUND_WIG_RESERVED_FOR_FUTURE_USE_29	MOORED	<a href="#">VER</a>

Resultados totales: 578

Resultados por página: 10    Ir a la página: 1

Figura 4.14: Lista de buques

En el apartado de cada barco individual tenemos todos los datos referente al buque mercante escogido, mostrados en la Figura 4.15. Esta pantalla muestra por un lado un carrusel con las imágenes del buque y la información del propio buque. Por otro lado, muestra toda la información referente a los datos de navegación y la información AIS recibida.

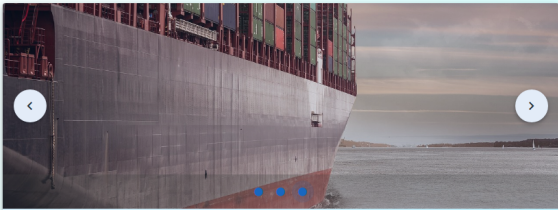
		<p><b>Datos de navegación</b></p> <p>Salida: <a href="#">GREECE PIRAEUS</a>      Destino: <a href="#">ALGERIA COLLO</a></p> <p>LOCODE: GRPIR      LOCODE: DZCOL</p> <p>Estado de navegación: MOORED</p> <p>Fecha de llegada estimada: 2023-08-03T17:29:28Z      Tiempo hasta la llegada: 31d 4h 58m 51s</p> <p>Distancia hasta el destino (metros): 34      Velocidad del buque (nudos): 0.000</p> <p>Latitud actual: 37.003      Longitud actual: 6.572</p> <p>Rumbo (grados): 0.861      Rumbo de proa (grados): -88.697</p> <p>Zona del buque:      Zona ECA: false</p> <p><b>Información AIS</b></p> <p>Timestamp del mensaje AIS: Wed Aug 02 20:26:40 UTC 2023      Fuente del mensaje AIS: SATELLITE</p> <p>Tiempo estimado de llegada del buque (AIS): 2023-08-03T17:29:28Z      Tiempo estimado de llegada del buque (Predicción): 2023-08-03T17:29:28Z</p>	
<p><b>Datos del buque</b></p> <p>Nombre: ROYAL_PRINCESS_QwSAz      Callsign del buque: ROYAL_PRINCESS_QwSAz_568536</p> <p>Número MMSI: 716336968      Número IMO: 568536</p> <p>Tipo de buque: HIGH_SPEED_CRAFT_HSC_NO_ADDITIONAL_INFORMATION</p> <p>Eslora total (metros): 180.000      Manga total (metros): 101.000</p> <p>Calado (metros): 0.007</p>			

Figura 4.15: Datos de un buque

Inmediatamente debajo se encuentra el mapa (Figura 4.16), que permite hacer el seguimiento en tiempo real del buque y ver el recorrido realido por el mismo. Este recorrido se puede visualizar gracias al histórico de eventos que tiene el barco asociado. También se tienen dos opciones:

- **Localizar:** Permite localizar y centrar en el mapa la posición actual del buque mercante.
- **Mapa global:** Permite visualizar en el mapa global del live tracker el buque mercante.



Figura 4.16: Seguimiento del buque

Debajo del mapa hay varias pestañas con diferentes opciones:

- **Historial:** Desde esta opción, mostrada en la Figura 4.17, se observa el histórico de eventos que ha tenido el buque. Podemos ver su puerto de salida, la fecha del evento, la geolocalización en ese momento, su velocidad, su estado de navegación y el puerto que tenía de destino.

HISTORIAL							PUERTO DE SALIDA	PUERTO DE DESTINO	ESTADISTICAS
Salida	Timestamp	Latitud	Longitud	Velocidad	Estado de navegación	Destino			
<a href="#">GREECE PIRAEUS</a>	16/1/2020 20:33:20	37.003	6.572	0.000	MOORED	<a href="#">ALGERIA COLLO</a>			
<a href="#">GREECE PIRAEUS</a>	16/1/2020 17:46:40	37.488	7.812	17.240	UNDER_WAY_SAILING	<a href="#">ALGERIA COLLO</a>			
<a href="#">GREECE PIRAEUS</a>	16/1/2020 15:0:0	38.010	9.202	18.402	UNDER_WAY_SAILING	<a href="#">ALGERIA COLLO</a>			
<a href="#">GREECE PIRAEUS</a>	16/1/2020 12:13:20	37.611	10.887	20.333	UNDER_WAY_SAILING	<a href="#">ALGERIA COLLO</a>			
<a href="#">GREECE PIRAEUS</a>	16/1/2020 9:26:40	36.397	12.272	19.429	UNDER_WAY_SAILING	<a href="#">ALGERIA COLLO</a>			
<a href="#">GREECE PIRAEUS</a>	16/1/2020 6:40:0	36.131	14.065	17.314	UNDER_WAY_SAILING	<a href="#">ALGERIA COLLO</a>			
<a href="#">GREECE PIRAEUS</a>	16/1/2020 3:53:20	35.831	15.883	17.668	UNDER_WAY_SAILING	<a href="#">ALGERIA COLLO</a>			
<a href="#">GREECE PIRAEUS</a>	16/1/2020 1:6:40	35.449	17.979	20.380	UNDER_WAY_SAILING	<a href="#">ALGERIA COLLO</a>			
<a href="#">GREECE PIRAEUS</a>	15/1/2020 22:20:0	35.100	20.021	19.904	UNDER_WAY_SAILING	<a href="#">ALGERIA COLLO</a>			
<a href="#">GREECE PIRAEUS</a>	15/1/2020 19:33:20	35.630	21.796	17.905	UNDER_WAY_SAILING	<a href="#">ALGERIA COLLO</a>			

Figura 4.17: Historial del buque

- **Puerto de salida:** Desde esta opción (figura 4.18), se visualizan todos los datos relevantes del puerto de salida actual.

HISTORIAL		PUERTO DE SALIDA	PUERTO DE DESTINO	ESTADISTICAS
Nombre:	<a href="#">GREECE PIRAEUS</a>	Pais:	GREECE	
Tipo de terminal de carga:	TANKER	Servicios ofrecidos:	UNMOORING, TOWING	
Capacidad total de barcos:	12.000	Capacidad actual:		
Latitud:	37.933	Longitud:	23.617	
LOCODE:	GRPIR			

Figura 4.18: Puerto de salida

- **Puerto de destino:** Desde esta opción (Figura 4.19) podemos visualizar todos los datos relevantes del puerto de destino actual.

HISTORIAL	PUERTO DE SALIDA	PUERTO DE DESTINO	ESTADISTICAS
Nombre: <a href="#">ALGERIA COLLO</a>		Pais: ALGERIA	
Tipo de terminal de carga: CONTAINER		Servicios ofrecidos: REPAIR, DRY_DOCKING	
Capacidad total de barcos: 17.000		Capacidad actual:	
Latitud: 37.003		Longitud: 6.572	
LOCODE: DZCOL			

Figura 4.19: Puerto de destino

- **Estadísticas:** Desde esta opción (Figura 4.20) se puede apreciar una gráfica que muestra como ha variado la velocidad del buque a lo largo del tiempo y su velocidad media.



Figura 4.20: Gráfica de velocidades del buque

#### 4.4.4. Comprobar el estado de un puerto

En la opción de puertos tenemos la tabla utilizada para visualizar todos los puertos registrados en la aplicación. En la Figura 4.21 podemos ver filtros para realizar búsquedas por el nombre del puerto, por el país en el que se encuentra, por el tipo de puerto y por su capacidad total de buques atracados.

**Puertos**  
Desde aquí podrás ver los distintos puertos, el tipo de puerto y su estado actual, así cómo acceder a cada puerto individualmente.

**Buscar**

Buscar por nombre     Buscar por locode     Buscar por país     Buscar por tipo de puerto     Buscar por servicios ofrecidos     Capacidad total: 30 Barcos

Ultima actualización: Thu Jan 16 23:20:00 UTC 2020

Nombre	LOCODE	País	Tipo de puerto	Servicios ofrecidos	Capacidad total
<a href="#">ALBANIA DURRES</a>	ALDRZ	ALBANIA	BULK	DRY_DOCKING, BERTHING	12
<a href="#">ALGERIA ALGER ALGHIERS</a>	DZALG	ALGERIA	OTHER	BUNKERING, SHIP_MANAGEMENT	12
<a href="#">ALGERIA COLLO</a>	DZCOL	ALGERIA	RORO	BUNKERING, UNBERTHING	5
<a href="#">ANGOLA LOBITO</a>	AOLOB	ANGOLA	FISHING	SHIP_CHANDLING, BERTHING	8
<a href="#">ARGENTINA BAHIA BLANCA</a>	ARBHI	ARGENTINA	PASSENGER	TOWING, BUNKERING	8
<a href="#">ARGENTINA BUENOS AIRES</a>	ARBUE	ARGENTINA	RORO	TOWING, BERTHING	0
<a href="#">ARGENTINA PUERTO SANTA CRUZ</a>	ARRZA	ARGENTINA	FISHING	TOWING, DRY_DOCKING	17
<a href="#">AUSTRALIA ALBANY</a>	AUALH	AUSTRALIA	RORO	DRY_DOCKING, SHIP_CHANDLING	12
<a href="#">AUSTRALIA FREMANTLE</a>	AUFRE	AUSTRALIA	CONTAINER	REPAIR, MOORING	7
<a href="#">AUSTRALIA KARUBA</a>	AUKRB	AUSTRALIA	RORO	UNMOORING, SHIP_MANAGEMENT	9

Resultados totales: 155


Resultados por página: 10    Ir a la página: 1

Figura 4.21: Lista de puertos

La información que muestra la tabla es el nombre del puerto, desde el cual es posible acceder a la página de cada puerto y filtrar por nombre, LOCODE, el país en el que se encuentra, el tipo de puerto, servicios ofrecidos y su capacidad total. Por último, tenemos las opciones de paginación y tamaño de elementos a mostrar en la tabla.

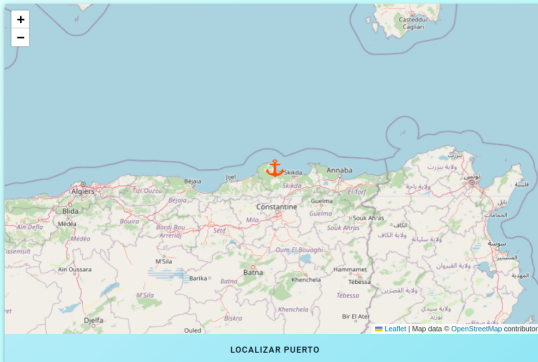
En el apartado de cada puerto individual tenemos todos los datos referente al puerto escogido. Lo primero que se visualiza es un mapa que muestra la ubicación del puerto, como se ve en la Figura 4.22. Por otro lado, el mapa cuenta con una opción para localizar y centrar la ubicación del puerto.

Home > Ports > ALGERIA COLLO



**Datos del puerto**

Nombre: ALGERIA COLLO	País: ALGERIA
LOCODE: DZCOL	Tipo de terminal de carga: CONTAINER
Capacidad total de barcos: 17	Capacidad actual: 15
Latitud: 37.003	Longitud: 6.572
Servicios ofrecidos: REPAIR, DRY_DOCKING	



LOCALIZAR PUERTO

Figura 4.22: Vista de un puerto con llegadas planificadas

Debajo del apartado anterior tenemos varias opciones para obtener información adicional:

- **Historial:** Desde esta opción (Figura 4.23) se visualiza de forma cronológica las próximas llegadas, buques en puerto y las salidas del puerto.

HISTORIAL					
Estado	Fecha	Nombre	MMSI	IMO	
NEXT ARRIVAL	8/2/2020 11:6:25	<a href="#">NORRONA_KuUNU</a>	192885913	678964	
IN PORT	7/2/2020 17:33:20	<a href="#">SEA_WATCH_3_XPaxy</a>	658011357	662338	

Figura 4.23: Historial del puerto

- **Planificación:** Desde esta opción (Figura 4.24) se muestra la planificación que tiene el puerto respecto a los buques que están por llegar y los que se encuentran en puerto.

HISTORIAL						
PLANIFICACIÓN						
February 2020						
10.Feb	11.Feb	12.Feb	13.Feb	14.Feb	15.Feb	16.Feb
Atraque 1						
Atraque 2						
Atraque 3		SEA_WATCH_3_XPaxy				
Atraque 4						
Atraque 5						
Atraque 6						
Atraque 7						
Atraque 8						
Atraque 9						
Atraque 10						
Atraque 11				NORRONA_KuUNU		

Figura 4.24: Planificación del puerto

- **Próximas llegadas:** Desde esta opción (Figura 4.25), se observan los datos de los próximos buques mercantes que tienen pendiente realizar una llegada.

HISTORIAL						
PLANIFICACIÓN						
PRÓXIMAS LLEGADAS						
Nombre	MMSI	IMO	Estado	ETA predecido	Puerto anterior	
<a href="#">NORRONA_KuUNU</a>	192885913	678964	UNDER_WAY_SAILING	8/2/2020 11:6:25	<a href="#">GUYANA GEORGETOWN</a>	

Figura 4.25: Próximas llegadas

- **Buques en puerto:** Desde esta opción (Figura 4.26) se muestran los datos de los buques que actualmente se encuentran en puerto.

HISTORIAL						
PLANIFICACIÓN						
PRÓXIMAS LLEGADAS						
BUQUES EN PUERTO						
Fecha	Nombre	MMSI	IMO	Estado	Puerto anterior	
7/2/2020 17:33:20	<a href="#">SEA_WATCH_3_XPaxy</a>	658011357	662338	MOORED	<a href="#">GUYANA GEORGETOWN</a>	

Figura 4.26: Buques en puerto

- **Salidas:** Desde esta opción (Figura 4.27) se visualizan los datos de los buques que han salido del puerto, incluyendo también su siguiente destino.

	HISTORIAL	PLANIFICACIÓN	PRÓXIMAS LLEGADAS	BUQUES EN PUERTO	SALIDAS	
Fecha	Nombre	MMSI	IMO	Estado	Destino	
2/1/2020 12:6:40	<a href="#">ALFA_NERO_sTeVN</a>	673790494	757078	UNDER_WAY_SAILING	<a href="#">CHILE SAN ANTONIO</a>	
27/1/2020 17:40:0	<a href="#">DISNEY_MAGIC_JOZec</a>	283157228	267123	UNDER_WAY_SAILING	<a href="#">CHILE SAN ANTONIO</a>	
30/1/2020 9:33:20	<a href="#">CELEBRITY_CONSTELLATION_AWVUD</a>	299059256	807207	UNDER_WAY_SAILING	<a href="#">CHILE SAN ANTONIO</a>	

Figura 4.27: Salidas

#### 4.4.5. Visualizar estadísticas globales

En la opción de panel de control se tiene acceso a diversas estadísticas que ofrece la aplicación. De forma general (Figura 4.28) podemos ver la cantidad de puertos, países y buques registrados, de estos últimos se puede ver la cantidad actualmente en puerto o navegando.



Figura 4.28: Estadísticas generales

Además de las estadísticas generales, se tiene acceso a varias estadísticas más específicas:

- *Distribución de puertos por tipo y servicios ofrecidos:* Se muestran dos gráficos circulares que distribuyen los puertos por tipo de puerto y por los servicios que ofrecen.
- *Distribución de puertos por países:* Se muestra una gráfica de tipo donut donde se refleja la distribución de puerto mediante países.
- *Top de puertos con mayor capacidad:* Se muestra un top 10 de los puertos con mayor capacidad.
- *Distribución de buques por tipos:* Se muestra la distribución de buques según el tipo en un diagrama de tipo donut.

En el Capítulo 5, al describir los resultados de la experimentación se muestran ejemplos de las estadísticas y diagramas descritos.

# Capítulo 5

## Experimentación

### 5.1. Preparación

Para la experimentación se ha utilizado un productor de buques mercantes para realizar una simulación con datos parecidos a los que se obtendría en un entorno real. Su objetivo es generar barcos con una ruta predefinida para simular la salida, recorrido y entrada de los buques en las infraestructuras portuarias.

El productor de buques va generando datos de forma continua, de forma que cada buque se va actualizando y alcanzando el destino asignado. Al productor se le puede configurar variables de entorno para modificar su comportamiento, de esta forma se puede escoger el periodo de tiempo en el que se generan los buques y cuanto es el máximo de barcos que pueden estar en ruta en un mismo instante de tiempo.

También se han preparado 83 rutas de navegación entre pares de puertos diferentes. Cada buque realiza el recorrido marcado por puntos intermedios de la ruta que le corresponde.

Con ambas herramientas se puede simular un entorno con una carga de datos similar a la que se tendría en un entorno real

### 5.2. Resultados

Para tener un entorno de simulación controlado se ha trabajado con los siguientes parámetros:

- El tiempo simulado ha sido desde el día 1 de agosto de 2023 hasta el día 1 de octubre de 2023.
- El máximo de buques navegando a la vez se ha establecido en 50.
- Se ha establecido que el front-end solicite al back-end cada 2 segundos los datos de los buques y cada 10 segundos los datos de los puertos para la actualización de ambos.



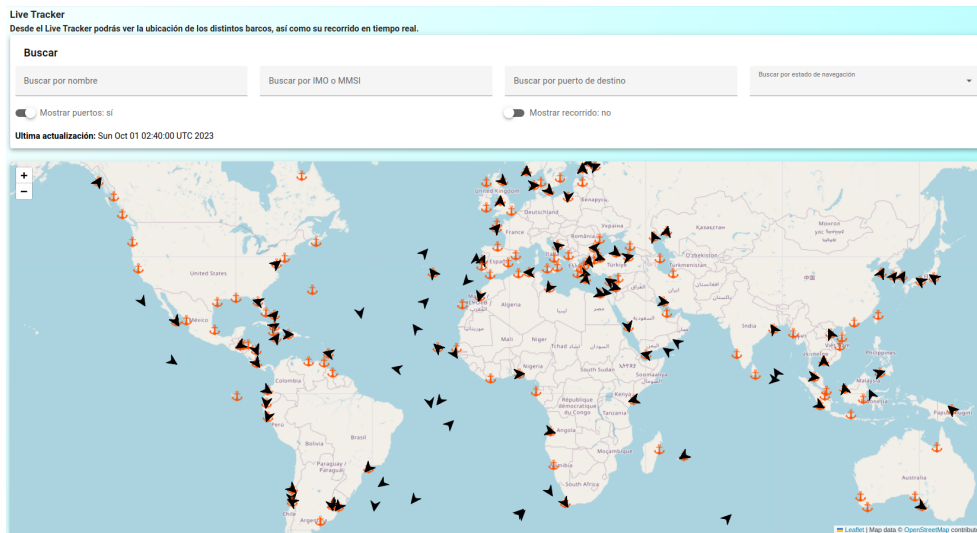


Figura 5.1: Live Tracker al terminar la simulación

Al terminar la simulación se ha llegado a registrar 155 puertos diferentes en 85 países. Unos 414 buques mercantes, de los cuales 363 llegaron a puerto y 50 en ruta. En la Figura 5.2 se muestra un resumen de las estadísticas simuladas.

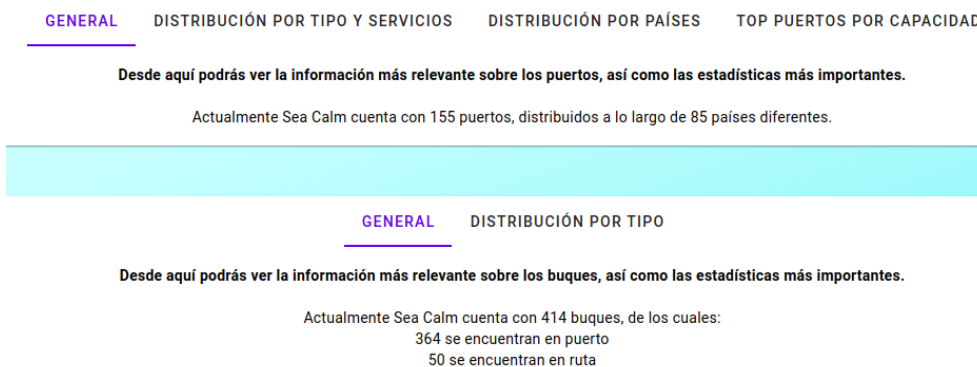


Figura 5.2: Estadísticas generales simulación

En cuanto a las distribuciones por tipo de puerto, los puertos con la categoría *passenger* son los mayoritarios. Por otro lado, por servicios ofrecidos, los puertos que ofrecen *bunkering* son los más abundantes, como se ve en la Figura 5.3.

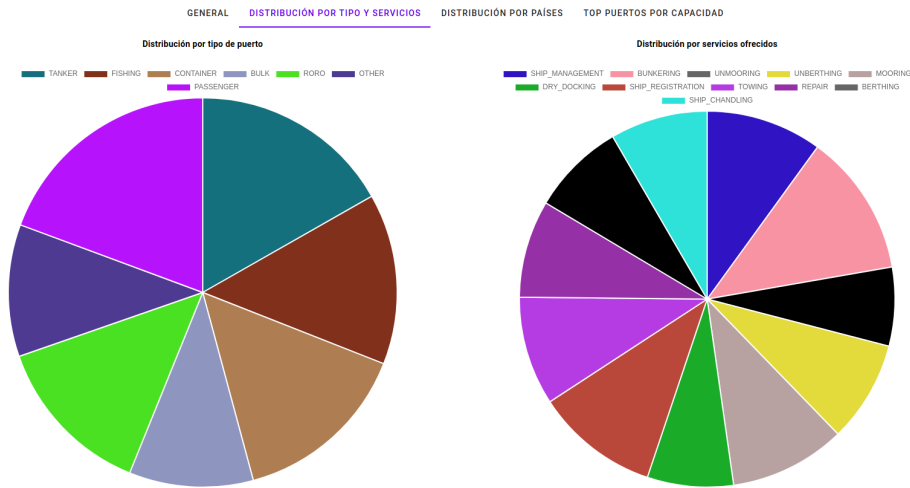


Figura 5.3: Puertos distribución por tipo y servicios ofrecidos

El país con mayor cantidad de puertos diferentes ha sido Estados Unidos, con con 14 puertos, seguido de España y Australia con 5 puertos cada uno, como se aprecia en la Figura 5.4.

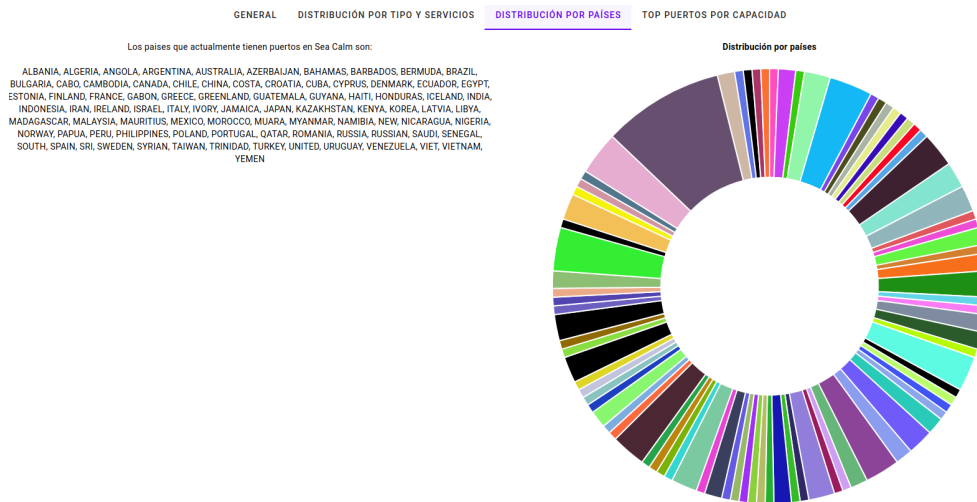


Figura 5.4: Puertos distribuidos por países

Los puertos con mayor capacidad, como refleja la Figura 5.5, han sido: San Antonio(Chile), Hals(Dinámica), Skyros(Grecia), Puerto Santo Tomas de Castilla(Guatemala), Trapani(Italia), Agadir(Marruecos), Lagos(Nigeria), Albany(Australia), Karuba(Australia) y Quintero(Chile).

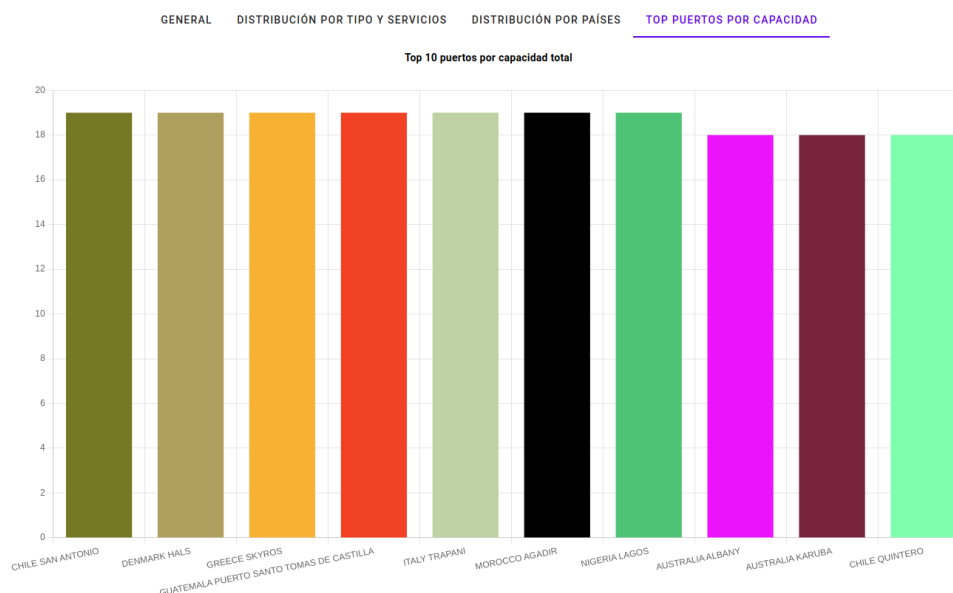


Figura 5.5: Top puertos con mayor capacidad

Por último, como se ve en la Figura 5.6, los buques auxiliares son el tipo de buque mayoritario (11 buques), seguido de buques de carga (10 buques) y barcos de ocio o recreativos (8 buques).

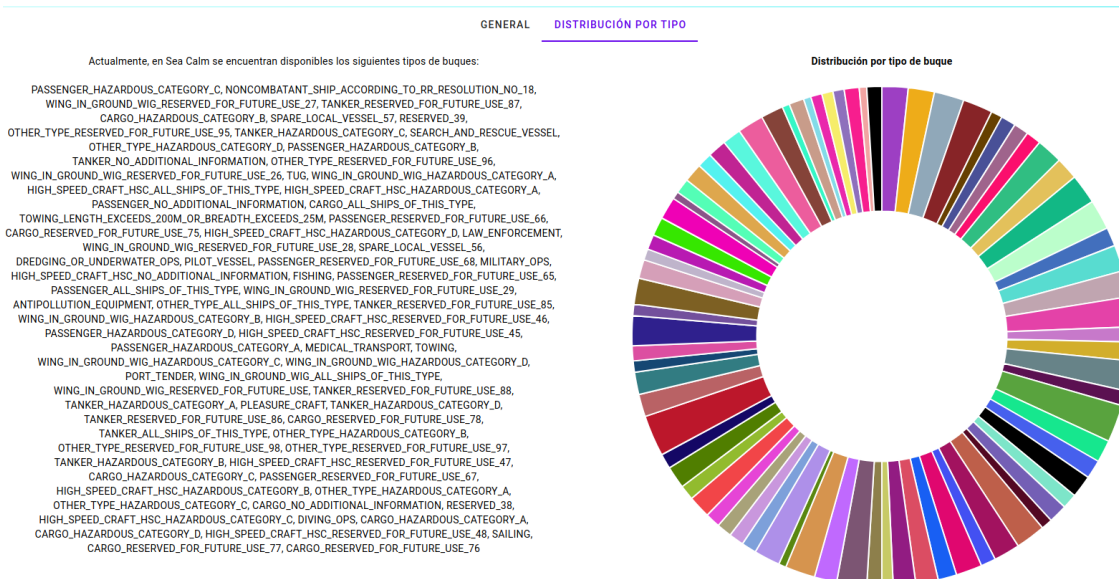


Figura 5.6: Buques distribuidos por tipos

Se ha realizado un segundo experimento, en esta ocasión con un intervalo de tiempo de dos años, desde el 1 de enero de 2020 hasta el 1 de enero de 2022 para comprobar con distintos tiempos de actualización de los buques desde el front-end el rendimiento de la aplicación. Los valores obtenidos son la media de buques que puede recargar el front-end de forma simultanea sin presentar fallos en su rendimiento.

<b>Tiempo de refresco</b>	<b>Capacidad máxima de buques simultáneos</b>
1 segundo	250
2 segundos	400
5 segundos	600
10 segundos	900
30 segundos	1600
60 segundos	2200

Tabla 5.1: Rendimiento de la aplicación

Dado que se han realizado en un entorno simulado donde se tiene que representar un periodo de tiempo extenso en una corta duración, los tiempos de actualización de los datos tienen que ser mucho menores para apreciar los cambios en tiempo real. En una situación real no es necesaria una actualización tan frecuente, por lo que 60 segundos de tasa de refresco podría considerarse el mejor de los casos posibles.

Con esto en mente, viendo los resultados obtenidos y teniendo en cuenta que 2200 buques simultáneos es una cantidad aceptable, podemos considerar que aún queda mucho margen para optimizar el rendimiento de la aplicación, ya que en situaciones reales se espera una mayor carga de datos. Para mejorar el rendimiento se pueden optimizar las peticiones al back-end para recuperar los datos siendo más selectivos, por ejemplo, recuperando los datos utilizando las fechas de actualización.

# Capítulo 6

## Presupuesto

En la sección de presupuesto se realiza una estimación de los costes para realizar el proyecto. En este apartado se detallan los costos asociados a la planificación, creación del prototipo, estudio de herramientas, investigación del ámbito, implementación front-end y back-end de la solución propuesta y realización de testing, documentación y experimentación.

Para establecer el precio por hora de lo que costaría el proyecto se tiene en cuenta el salario medio que obtiene un desarrollador full-stack junior en España. De esta forma se han extraído los datos de diversos portales web que publican esta información, estos portales han sido: Glassdoor<sup>1</sup>, Talent<sup>2</sup> e Indeed<sup>3</sup>. Tras revisar los datos se ha establecido un precio por hora de 17,00€.

En la tabla 6.1 se especifican las tareas y las horas invertidas, así como el precio final de cada tarea, el tiempo total y el precio final.

<b>Tarea</b>	<b>Nº Horas</b>	<b>Precio final de la tarea (€)</b>
Planificación	6	102 €
Prototipado	12	204 €
Estudio de tecnologías	48	816 €
Investigación	18	306 €
Frontend	130	2.210 €
Backend	80	1.360 €
Testing	12	204 €
Documentación	66	1.122 €
Experimentación	18	306 €
<b>TOTAL</b>	<b>390</b>	<b>6.630 €</b>

Tabla 6.1: Presupuesto

---

<sup>1</sup><https://www.glassdoor.es>

<sup>2</sup><https://es.talent.com>

<sup>3</sup><https://es.indeed.com/career/desarrollador-junior/salaries>

# Capítulo 7

## Conclusiones y líneas futuras

### 7.1. Conclusiones

A lo largo del proyecto se ha desarrollado una aplicación que sirve para realizar el seguimiento de los buques mercantes, así como obtener la planificación que tendría un puerto respecto a sus próximas llegadas.

En los momentos iniciales del proyecto hubo un periodo de formación para la utilización del framework de Vue con Vuetify, pero el momento en el que más se ha aprendido ha sido en el desarrollo en sí del proyecto. A rasgos generales se han conseguido bastantes progresos y objetivos:

- Para la visualización en tiempo real de buques y puertos se uso Leaflet.
- Se ha conseguido implementar los históricos de puertos y buques mediante las estructuras de datos que representan eventos y visualización en el front-end de los mismos.
- Para los indicadores estadísticos se tiene el panel de control con estadísticas de diversas índole. Los buques y puertos tienen sus gráficas e indicadores de planificación.
- Para tener una implementación limpia y mantenible se ha construido tanto el back-end como el front-end mediante arquitectura hexagonal. Se tienen tests unitarios, de integración y end-to-end.
- La aplicación tiene múltiples idiomas con Vue I18n, estando en su totalidad en Español e Inglés.

### 7.2. Líneas futuras

Para futuras mejoras del proyecto se propone lo siguiente:

- Implementar servicios para autenticar y registrar usuarios.
- Contar con envío y recogida de señales AIS para una futura implementación real.
- Diseñar e implementar la comunicación con los buques mercantes para permitir la comunicación entre buques y gestores portuarios.

# Capítulo 8

## Conclusions and Future Work

### 8.1. Conclusions

Throughout the project, an application has been developed to track merchant ships and obtain the planning for upcoming arrivals at a port.

During the initial stages of the project, there was a learning period to use the Vue framework with Vuetify. However, the most significant learning took place during the development of the project. In general, significant progress and objectives have been achieved:

- Leaflet was used for real-time visualization of ships and ports.
- Historical data for ports and ships has been implemented using data structures that represent events and their front-end visualization.
- The control panel provides various statistical indicators. Ships and ports have their own charts and planning indicators.
- To ensure a clean and maintainable implementation, both the back-end and the front-end have been built using a hexagonal architecture. There are unit tests, integration tests, and end-to-end tests.
- The application supports multiple languages using Vue I18n, currently available in Spanish and English.

### 8.2. Future Lines

For future project improvements, the following have been taken into account:

- Implement services for user authentication and registration..
- Incorporate the sending and receiving of AIS signals for future real-world implementation.
- Design and implement the communication with merchant ships to allow communication between ships and port managers.

# Bibliografía

- [1] Marco Balduzzi, Alessandro Pasta, and Kyle Wilhoit. A security evaluation of ais automated identification system. *Proceedings of the 30th Annual Computer Security Applications Conference*, 2014.
- [2] A.G. et al. Bole. Radar and arpa manual : Radar and target tracking for professional mariners, yachtsmen and users of marine radar. 2005.
- [3] M.G. Burns. Port management and operations. 2015.
- [4] James Corbett and James Winebrake. *The impact of globalisation on international maritime transport activity: Past trends and future perspectives*. 01 2008.
- [5] Robert C. Martin. *Clean architecture : a craftsman's guide to software structure and design*. 2017.
- [6] Abbas Harati Mokhtari. Impact of automatic identification system (ais) on safety of marine navigation. page 19, 2007.
- [7] Naima Saeed, Dong-Wook Song, and Otto Andersen. Governance mode for port congestion mitigation: A transaction cost perspective. *Netnomics*, 19(3):159–178, 2018.
- [8] Sandro Steinbach. Port congestion, container shortages, and u.s. foreign trade. *Economics Letters*, 213, 2022.
- [9] Davi Vieira. *Designing Hexagonal Architecture with Java*. 2022.