
Trabajo de Fin de Grado

# Desarrollo de proyecto fullstack: planificador de camiones

*Fullstack project development: truck planner*

Jorge Hernández Batista

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:

”*Desarrollo de proyecto fullstack: planificador de camiones*”

ha sido realizada bajo su dirección por Don **Jorge Hernández Batista**, con N.I.F. 51.150.899-A.

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

A mis tutores, Christopher e Israel, por guiar y encaminarme de la mejor manera a la hora de tutorizar este trabajo.

A Luis Cutillas y Leonardo Ruiz, por introducirme en el mundo de las matemáticas y la informática, sin ustedes no estaría dónde estoy hoy.

A mis padres, hermana, abuelas y amigos, por apoyarme de manera incondicional, darme todas las oportunidades posibles y confiar en mí siempre, les quiero mucho.

A todas las personas que he conocido a lo largo de la carrera, que de una forma u otra me han empujado hacia adelante para finalizar esta etapa de mi vida que ya llega a su final.

**Resumen**

*El proyecto "Planificador de Camiones" se ha desarrollado como una aplicación para la gestión eficiente de camiones en un puerto, con el objetivo de mejorar los procesos de intercambio de mercancías para camioneros y gestores portuarios. La aplicación permite a los usuarios programar y coordinar la llegada y salida de los diferentes camiones en el puerto, para optimizar los tiempos de carga y descarga, y evitar retrasos y congestiones en las zonas de carga. Además, la herramienta también brinda información en tiempo real sobre el estado y ubicación de los camiones, lo que permite a los usuarios planificar sus operaciones de manera más efectiva. En cuanto a la implementación, se ha utilizado una combinación de lenguajes de programación y tecnologías de desarrollo, como Java y Timescale para el back-end así como Vue3 y JavaScript para el front-end. Con esta herramienta, se busca facilitar la gestión de los procesos logísticos del puerto, reducir los tiempos de espera y mejorar la experiencia de los usuarios involucrados en el intercambio de mercancías en el puerto.*

**Palabras clave:** Aplicación, Java, Timescale, back-end, Vue3, JavaScript, front-end.

**Abstract**

The ”Truck Planner” project has been developed as an application for the efficient management of trucks in a port, with the goal of improving the exchange of goods processes for truck drivers and port managers. The application allows users to schedule and coordinate the arrival and departure of different trucks in the port, to optimize loading and unloading times, and to avoid delays and congestion in loading areas. Additionally, the tool also provides real-time information about the status and location of the trucks, which allows users to plan their operations more effectively. In terms of implementation, a combination of programming languages and development technologies such as Java and Timescale for the back-end, as well as Vue3 and JavaScript for the front-end, have been used. With this tool, the aim is to facilitate the management of the port’s logistics processes, reduce waiting times, and improve the experience of users involved in the exchange of goods in the port.

***Keywords:*** *Application, Java, Timescale, back-end, Vue3, JavaScript, front-end.*

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción

## 1.1. Definición del problema

En los últimos años, en la sociedad actual, se ha podido observar claramente el impacto significativo de la globalización en todos los aspectos de nuestras vidas. Uno de los ámbitos en los que se ha experimentado un crecimiento considerable es el transporte de mercancías, como indica el Ministerio de Transportes, Movilidad y Agenda Urbana de España, "La variación [...] fue positiva desde 2015. A 1 de enero de 2022 el incremento interanual [...] fue del 4 %."[1]. Cada vez más, nos encontramos consumiendo productos que provienen de lugares muy alejados de nuestro punto de compra, lo que ha generado un aumento considerable en los envíos y la logística de transporte.

Este fenómeno ha llevado consigo una serie de inconvenientes que debemos enfrentar. Uno de los principales desafíos que surgen es la congestión de vehículos pesados. Debido a la gran demanda existente y la necesidad de cubrirla, se ha generado un incremento en el número de camiones y otros medios de transporte utilizados para llevar las mercancías de un lugar a otro. Esto se puede apreciar en la Figura 1.1, la cuál refleja el incremento del tránsito de vehículos pesados en el puerto de Algeciras, con un aumento de de 10.000 operaciones de carga y descarga entre los años 2020 y 2023.[1]. La congestión vehicular resultante ha generado una serie de problemas adicionales. Por un lado, el tráfico lento y atascado ocasiona retrasos en las entregas y afecta la eficiencia del sistema logístico en su conjunto. Además, el aumento de la circulación de camiones ha contribuido al deterioro de la infraestructura vial, generando una necesidad de inversiones y mantenimiento más frecuentes.

Otro aspecto negativo que se deriva de esta situación es el impacto ambiental. El incremento en el transporte de mercancías ha llevado a un aumento en las emisiones de gases de efecto invernadero y la contaminación del aire, ya que en 2022 las emisiones de gases de efecto invernadero aumentaron un 5,7 % respecto al año anterior. El transporte por carretera, en este año, también lo hizo en un 3,3 %[2], lo que afecta directamente la calidad de vida de las personas y contribuye al cambio climático.

En conclusión, la globalización ha traído consigo un aumento significativo en el trans-

[1] https://www.cadenadesuministro.es/noticias/el-trafico-ro-ro-de-camiones-en-el-puerto-de-algeciras-aumento-un-12-en-marzo/

[2] https://www.soziable.es/espana-incremento-emisiones-gases-de-efecto-invernadero

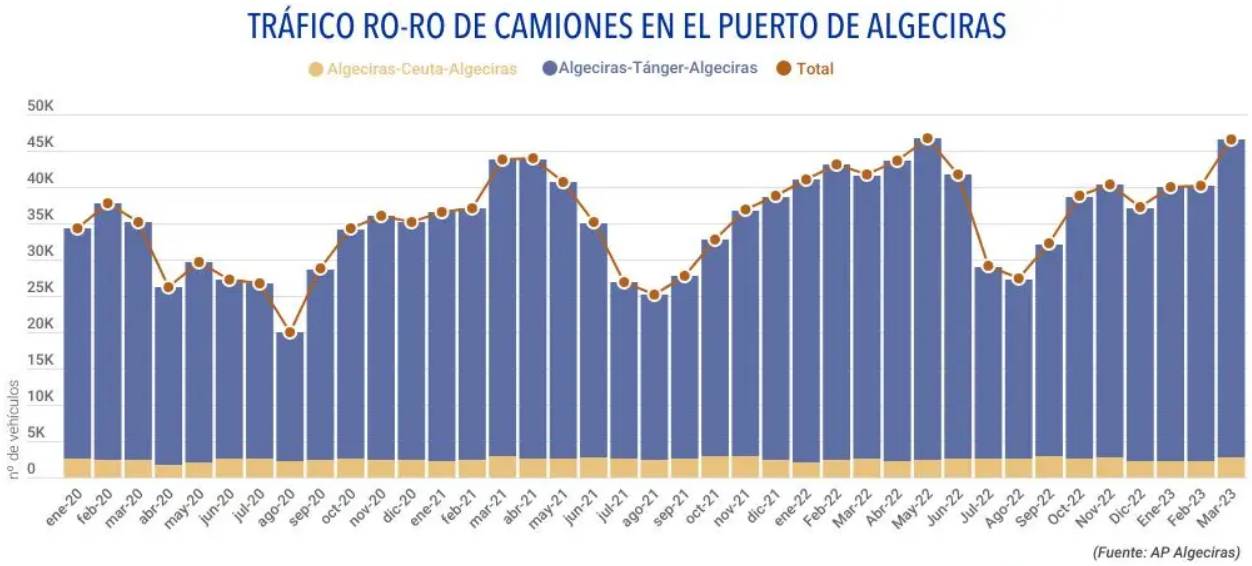


Figura 1.1: Gráfico aumento del tráfico de operaciones de carga y descarga rodadas

porte de mercancías, lo que generado una serie de problemas y desafíos. La congestión vehicular debido a la alta demanda, los retrasos en las entregas, el deterioro de la infraestructura vial y el impacto ambiental son solo algunos de los inconvenientes que debemos abordar y buscar soluciones efectivas. La repercusión de este factor se puede apreciar de manera notable en las diferentes zonas donde se realizan las actividades de recepción o entrega de servicios portuarios. Estos lugares adquieren una importancia clave, ya que su funcionamiento adecuado afecta a todos los eslabones de la cadena de suministro. Las consecuencias derivadas de esta situación se pueden dividir en función del tipo de operador al que afecten:

- **Operadores terrestres:** Se produce un incremento en tiempos de rotación, generando retrasos y disminuyendo la eficiencia, incapacitando la realización de más viajes y por ende, afectando a más cadenas de suministro, más allá de la propiamente ya demorada.
- **Operadores marítimos:** Pueden generar demoras para los trayectos planificados, afectando a mercancías que en un primer momento no deberían verse afectadas.
- **Operadores portuarios:** Se produce una reducción de la eficiencia, y con ello también se reduce la competitividad, generando pérdidas para este sector.

Finalmente, esta problemática acaba repercutiendo en toda la sociedad, pues al ser cada vez más dependiente de productos que se producen, generan o extraen de puntos lejanos al centro de adquisición, se produce un sobrecoste que acaba afectando al precio final que los clientes acaban pagando.

Algunos casos de ejemplo que pueden servir para percibir mejor la problemática que esto supone pueden ser el atasco del Canal de Suez[2], ruta marítima que supone el 12 % del comercio mundial, que se vivió entre el 23 de marzo de 2021 y el 29 de marzo del mismo mes, que provocó pérdidas de 364 millones de euros. Y aunque esta problemática se perciba a nivel portuario y no marítimo, un conjunto de pequeños retrasos puede provocar un mayor sobrecoste en diversas rutas y por ello, en los productos.

## 1.2. Justificación

Tal y como se expone en la definición de la problemática desarrollada en el apartado anterior, existe una oportunidad de mercado basada en la creación y promoción de una aplicación que permita agilizar todo el proceso de entrega y recogida de mercancías en las instancias portuarias, y así disminuir el problema de los retrasos en las cadenas de suministros. Como se expone en la Figura 1.2, los proveedores, con el paso de los años, se retrasan más respecto al mes anterior en realizar sus envíos[3].

Debido a las posibles pérdidas en eficiencia, competitividad y necesidad de incremento de precios en los que pueden incurrir los diferentes operadores, encontramos la oportunidad de lanzar la aplicación y que sea acogida con gran interés, ya que no solo ofrece un muy útil servicio, sino que no hay una gran competencia en el sector, lo que permitiría explotar un nicho dónde obtener un gran rédito.

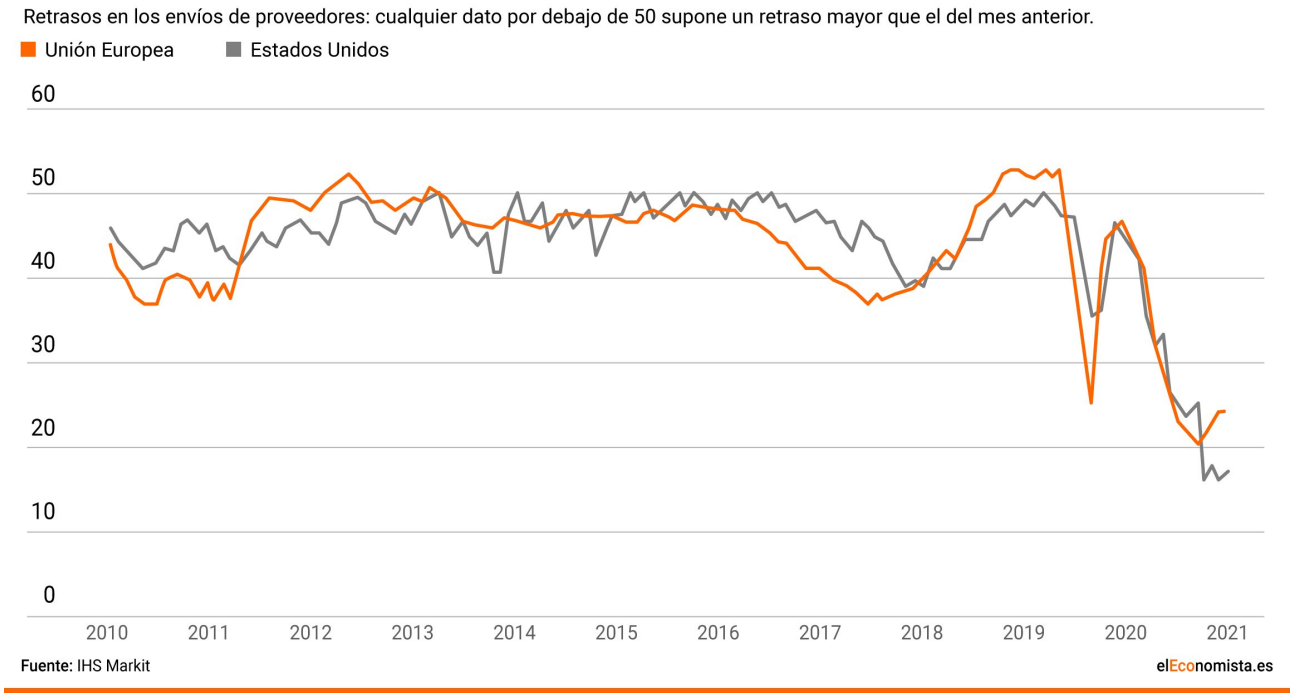


Figura 1.2: Gráfico retrasos cadena de suministros

## 1.3. Objetivos

El objetivo de este trabajo es desarrollar una aplicación Full Stack que permita comunicar y visualizar los datos relativos a la entrega o recogida de mercancías en estancias portuarias. Se desea que sea de utilidad bien por parte tanto de operadores marítimos como terrestres, y siendo los operadores portuarios los encargados de organizar los turnos y horarios de los diferentes solicitantes. De esta forma se pretende disminuir los tiempos de espera y con ello maximizar los beneficios de todos los operadores, incentivando el uso de esta aplicación y obteniendo beneficios en base a su uso.

Los diferentes objetivos del proyecto se pueden presentar de la siguiente manera:

[3]https://www.eleconomista.es/economia/noticias/11450909/10/21/El-grafico-del-FMI-que-muestra-el-atasco-global-en-puertos-y-cadenas-de-suministro.html

- Desarrollar un back-end utilizando una arquitectura de microservicios. El objetivo principal es implementar una arquitectura de microservicios para el back-end, lo cual permitirá una mayor modularidad, escalabilidad y flexibilidad en el desarrollo de futuras funcionalidades. Hacer uso de una arquitectura de microservicios permitirá una mejor separación de responsabilidades y facilitará el desarrollo ágil de nuevas funcionalidades. Además, al ser independientes entre sí, los microservicios pueden ser escalados de manera individual, lo que mejora la eficiencia del sistema y garantiza una mayor disponibilidad.
- Implementar una base de datos escalable y segura. El objetivo es diseñar e implementar una base de datos robusta, capaz de gestionar volúmenes de datos de forma eficiente. Además, se deben aplicar medidas de seguridad adecuadas para proteger la integridad y confidencialidad de la información almacenada. Con estos estándares nos aseguramos de que el crecimiento de la cantidad de datos no compromete el rendimiento.
- Diseñar y desarrollar un front-end intuitivo y de fácil uso. El objetivo es desarrollar una interfaz de usuario atractiva y amigable, que brinde una experiencia fluida a los usuarios. Esto se logrará a través de un diseño cuidado, una navegación intuitiva y una disposición adecuada de los elementos visuales. El front-end debe ser fácil de entender y utilizar, independientemente del nivel de conocimientos técnicos del usuario, obteniendo de esta forma una mejor experiencia de usuario y aumentando la satisfacción y fidelidad de los usuarios.

## 1.4. Estructura del documento

El resto del presente documento se estructura de la siguiente manera:

- **Capítulo 2 - Estado del arte.** En este capítulo se examina el estado actual del tránsito de vehículos pesados en relación a la carga y descarga de mercancías en los puertos. Se analizan investigaciones recientes y aplicaciones existentes en el mercado que abordan los desafíos asociados con la congestión de vehículos pesados en áreas portuarias y ofrecen soluciones para optimizar la logística de transporte y mejorar la eficiencia en estas operaciones críticas.
- **Capítulo 3 - Desarrollo.** En este capítulo se explora el desarrollo tanto del back-end como del front-end de la aplicación y se examinará cómo se conectan entre sí para lograr un funcionamiento eficiente y cohesionado. Finalmente se analizan las tecnologías y herramientas utilizadas en cada uno de estos componentes.
- **Capítulo 4 - Funcionamiento de la aplicación.** Este capítulo describe el funcionamiento de la aplicación, los posibles casos de uso que pueden llevarse a cabo en ella y como se ejecutan en la propia aplicación.
- **Capítulo 5 - Testing y despliegue.** Este apartado trata como se ha comprobado el correcto funcionamiento de las diferentes implementaciones y como se levanta la aplicación para ser lanzada y ejecutada en los diferentes dispositivos.
- **Capítulo 6 - Presupuesto.** En este apartado se busca calcular los gastos asociados a una posible implementación futura de la plataforma, así como el tiempo aproximado necesario para recuperar la inversión.

- **Capítulo 7 - Conclusiones y líneas futuras.** Este capítulo cierra el proyecto con una serie de conclusiones, así cómo observaciones sobre funcionalidades a implementar a futuro.

# Capítulo 2

# Motivación y estado del arte

## 2.1. Estado del arte

El tránsito de vehículos pesados en puertos es un tema de gran importancia y relevancia en el ámbito del transporte y la logística. La operación eficiente de estos vehículos es crucial para asegurar el flujo constante de mercancías y mantener la cadena de suministro en funcionamiento. Sin embargo, este proceso enfrenta una serie de desafíos complejos y multifacéticos que requieren atención y soluciones adecuadas.

Uno de los desafíos más prominentes es la congestión vial. Los puertos suelen ser puntos clave de entrada y salida de mercancías, lo que genera una gran demanda de transporte de carga. Esta alta demanda puede resultar en un gran número de vehículos pesados circulando por las áreas portuarias, lo que a su vez puede conducir a atascos de tráfico significativos. Estos atascos no solo causan retrasos en las entregas, sino que también tienen un impacto negativo en la productividad general de las operaciones portuarias. Esto puede apreciarse en el Puerto de Valencia, el más grande de España y el quinto de Europa[1], que sufre retenciones dentro y fuera de las estancias portuarias constantemente, como se puede apreciar en la Figura 2.1. Aunque se valoran diferentes opciones para solventar esta problemática[2], disponer de puertos dónde diariamente pasan más de 3000 camiones va a generar, en unas vías o en otras, una gran congestión y otra problemática derivada como la contaminación por el emisión de gases de efecto invernadero derivado del tiempo que un vehículo espera en una retención.

Así mismo, se han introducido soluciones tecnológicas específicas para los vehículos pesados en los puertos. Por ejemplo, se han desarrollado sistemas de seguimiento y gestión de flotas que permiten a las empresas monitorear la ubicación y el rendimiento de sus vehículos en tiempo real. Un ejemplo, que se desarrollará más a continuación, es Transportic, un sistema empleado para comunicar a los diferentes agentes en el transporte de mercancías y tener un seguimiento de los contenedores y agentes participantes. Estos sistemas proporcionan información valiosa sobre la eficiencia de las rutas, el consumo de combustible y el mantenimiento de los vehículos, lo que ayuda a optimizar las operaciones y reducir los tiempos de espera en los puertos.

[1] https://www.diariodetransporte.com/articulo/transporte-maritimo-y-puertos/valenciaport-situo-2022-cabeza-puertos-espana-trafico-vehiculos/20230320223053082199.html

[2] https://www.diariodetransporte.com/opinion/diariodetransporte-com-redaccion/las-retenciones-de-camiones-en-el-puerto-de-valencia-opinion-de-miguel-a-mata-pardo/20200731194245035392.html


Figura 2.1: Casos de retenciones a lo largo de las zonas colindantes al Puerto de Valencia

Además de las soluciones mencionadas, también se han implementado medidas para fomentar la colaboración y la coordinación entre los actores involucrados en el tránsito de vehículos pesados en puertos. Esto incluye la creación de plataformas de intercambio de información, donde las empresas y las instituciones pueden compartir datos y buenas prácticas para mejorar la eficiencia y reducir la congestión en las áreas portuarias. Esta colaboración también puede extenderse a la planificación conjunta de horarios de llegada y salida de vehículos, con el objetivo de distribuir de manera más equitativa la demanda de transporte de mercancías a lo largo del día.

En resumen, el tránsito de vehículos pesados en puertos es un desafío complejo que requiere soluciones integrales. La congestión vial, los retrasos en las entregas y la disminución de la productividad son algunos de los problemas a los que se enfrenta este sector. Sin embargo, gracias a la implementación de sistemas de gestión del tráfico, soluciones tecnológicas para la gestión de flotas y una mayor colaboración entre los actores involucrados, se están logrando avances significativos para mejorar la eficiencia y reducir la congestión en los puertos. Estas soluciones no solo benefician a las empresas y a las instituciones portuarias, sino que también tienen un impacto positivo en la economía en general, al garantizar un flujo eficiente de mercancías y mantener las operaciones comerciales en marcha.

A continuación se describen algunas de las soluciones ya existentes en el mercado, dónde algunas llevan más de 15 años de vigencia.

### 2.1.1. Estado del arte a nivel nacional

- **ValenciaPortPCS:** La aplicación, lanzada por la Autoridad Portuaria de Valencia, cuenta con funcionalidades tales como dar a conocer a los profesionales del transporte a que hora deben estar en el puerto para descargar un contenedor, detectar incidencias en las órdenes de entrega o modificar datos tales como la matrícula (Figura 2.2). También permite visionar en directo, a través de las cámaras del puerto,

la situación actual del tráfico en el recinto portuario, con el fin de reducir los tiempos de espera, las emisiones de CO2 y agilizar los procesos de carga y descarga en las terminales. Se ha comenzado a utilizar recientemente, en Enero de 2023.[3].

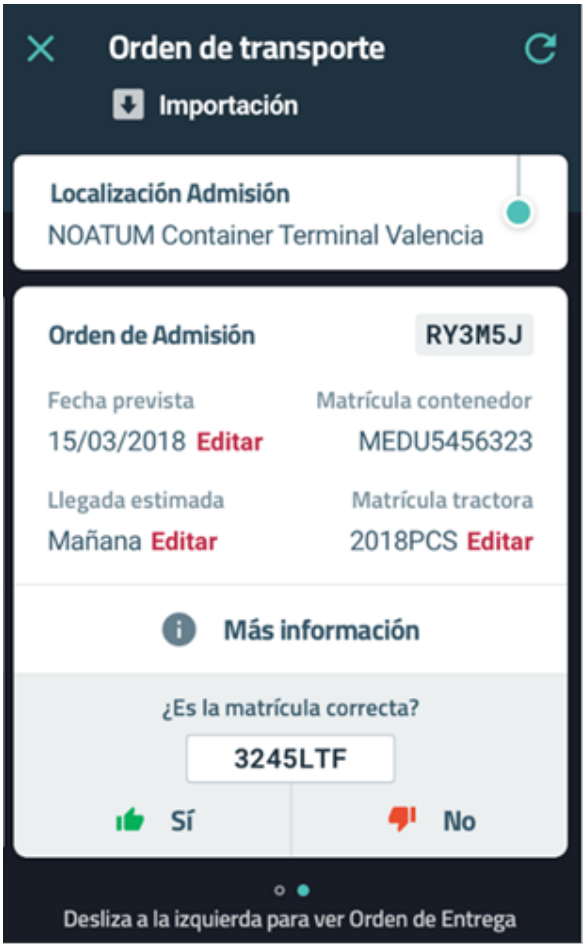


Figura 2.2: Caso de uso de la aplicación del Puerto de Valencia

- **Transportic:** Más allá de ser una aplicación, es un sistema empleado por el Puerto de Barcelona que permite la comunicación entre la empresa de transporte y el chófer del camión, pero que no comunica con la estancia portuaria. En cambio, es la estancia portuaria la que se comunica con la empresa de transporte y esta envía los datos al camionero (Figura 2.3). Aunque más alejado del objetivo final de la aplicación de este proyecto, refleja el estado del campo en el territorio nacional, siendo el Puerto de Barcelona uno de lo más relevantes del territorio español. Su uso comenzó a darse a inicios del 2000 y se mantiene vigente hasta hoy.[4]

[3]https://www.valenciaport.com/el-puerto-de-valencia-inicia-la-formacion-de-5-000-camioneros-en-la-app-para-movil-de-valenciaportpcs/

[4]http://www.portic.net/bcnelearningport/transportic/transportic.html


Figura 2.3: Caso de uso del sistema Transportic

### 2.1.2. Estado del arte a nivel internacional

- **Circular:** Esta aplicación, usada en en la ciudad de Rosario (Argentina), en uno de los mayores centros de exportación de bienes del país, fue impulsada por un problema de cuellos de botella, pues el 70 % de los camiones acudían a realizar las operaciones entre las 6:00pm y las 2:00am. Soluciona el problema haciendo uso de un turno previamente indicado, y estableciendo una franja y horario específico (Figura 2.4), pudiendo llevar la tarea en solo cuatro horas y no demorando al transportista. El uso de esta aplicación consta desde 2019.[5].


Figura 2.4: Caso de uso de la aplicación Circular

[5]https://www.iproup.com/innovacion/4299-cargill-nueva-aplicacion-para-mejorar-la-logistica-de-su-flota-de-camiones

- **PORTail:** La aplicación "Trucking PORTail", usada en la isla de Montréal (Canadá), transmite el tráfico de camiones y tiempos de espera en tiempo real a los puertos, permite reducir la congestión por medio de alertas y perfiles personalizados (Figura 2.5), mejorando el tráfico, la organización y la eficiencia. El proyecto se lanzó en 2016 y continúa vigente hasta hoy.[6]

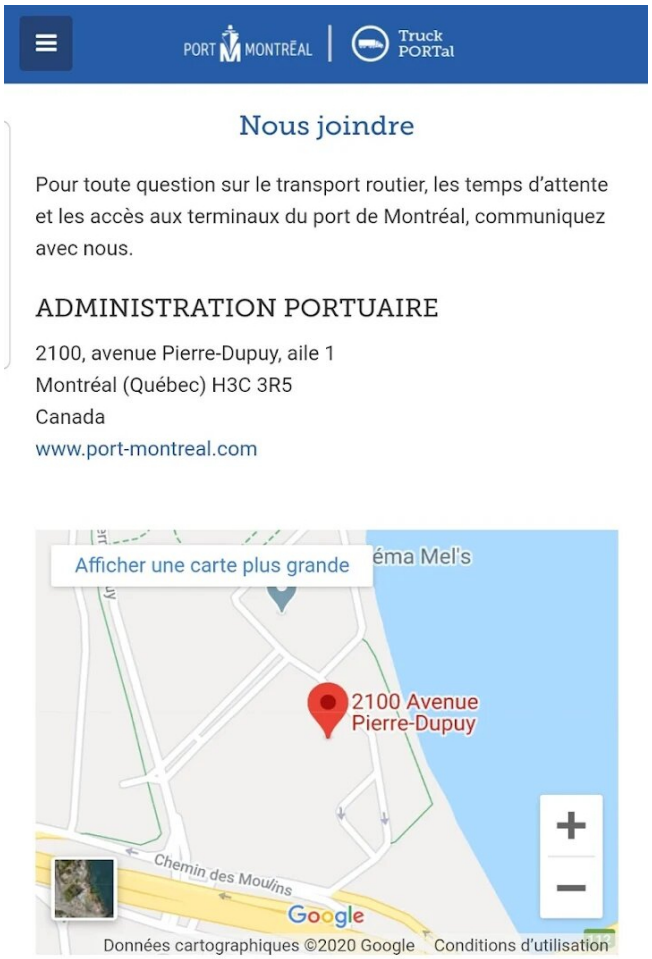


Figura 2.5: Caso de uso de la aplicación TruckPortail

## 2.2. Motivación

Debido a la gran importancia que tiene para la economía y bienestar de las Islas Canarias el acceso a mercancías externas, y ya que esta se hace en casi su totalidad por medio de transportes marítimos, es que este Trabajo de Fin de Grado despertó en mi un gran interés por llevarlo a cabo. Factores como la gran relevancia que podría tener para el desarrollo de los canarios un aumento de la eficiencia en el intercambio de mercancías en las instancias portuarias, o ser consciente de que apenas existen tecnologías implementadas en el campo, ya no solo a nivel autonómico, sino nacional o internacional me motivan especialmente para llevar este trabajo a cabo.

Otro aspecto que me ha motivado a llevar a cabo este Trabajo de Fin de Grado es el desarrollo de una aplicación de inicio a fin. Este proyecto me permitirá adquirir

[6] https://play.google.com/store/apps/details?id=com.portmontreal.truckportal

conocimientos en nuevas tecnologías y combinarlas entre sí, lo que a su vez fortalecerá los conceptos y habilidades que he adquirido a lo largo de mi carrera universitaria, enriqueciendo así mi formación académica.

# Capítulo 3

# Truck-planner

## 3.1. Descripción general

El aplicativo implementado en este Trabajo de Fin de Grado es una aplicación Full Stack consistente en organizar y planificar de forma óptima y ágil los procesos de carga y descarga de mercancías en estancias portuarias, bien entre camioneros y gestores portuarios, o bien entre los propios gestores y los estibadores, por medio de un sistema de reservas tanto de fechas y horarios como de espacios.

Para llevar a cabo esta tarea es necesario que cuando los usuarios usen la aplicación introduzcan los datos para hacer uso del sistema de reservas, y esta dependerá del tipo de usuario que se trate, por lo se identifiquen para llevar a cabo este proceso, y por ello podemos definir diferentes tipos de usuario, que tendrán que aportar diferente tipo de información:

### 3.1.1. Usuarios

#### Camioneros

Los camioneros deben informar varios elementos en la aplicación:

- **El camión:** Se debe aportar la información de la matrícula, la marca, el nombre del conductor, y si es una operación de carga o descarga.
- **La operación a realizar:** También debe ser informada la mercancía de esta operación, la cantidad de la misma, y nuevamente si es de carga o descarga.
- **El acceso del camión al puerto:** A pesar de que el gestor portuario es quién fijará los horarios y puntos de entrada y salida, una vez el camionero selecciona entre las diferentes posibilidades, debe ser este quién cumplimente su forma de identificación en el puerto (haciendo uso de una tarjeta, de un código qr..., etc).

#### Gestores portuarios

Los gestores portuarios deben informar los siguientes elementos en la aplicación:

- **El acceso del camión al puerto:** Aunque es el camión el que hace la reserva, previamente el gestor portuario debe fijar que reservas están disponibles para hacer,

y no exclusivamente en términos de tiempo, sino también de espacio, fijando el punto de entrada y salida además de la fecha y hora tanto de entrada como salida.

- **La escala del buque en el puerto:** A la vez que las reservas de espacios de camiones son creadas por los gestores portuarios, también se hace lo propio con los buques, indicando este usuario las fechas y horarios en los que se llevará a cabo la escala.

**Estibadores**

Los estibadores deben comunicar los siguientes datos a la aplicación:

- **El buque:** Deben aportar información relativa al buque que se dispone a atracar, entre ella el identificador del barco.
- **Las escalas:** También se deben informar las últimas escalas que haya llevado a cabo el buque, informando con ellas las diferentes operaciones que hayan realizado en dichos puertos y las fechas en las que se llevaron a cabo los atraques y los embarques.
- **La escala del buque en el puerto:** Si bien las fechas de entrada y salida son establecidas por los gestores, son los estibadores los encargados de informar el tipo de operaciones que se llevarán a cabo.

## 3.2. Tecnología y arquitectura del software

Toda aplicación Full Stack se compone de varias componentes, en esta destacan el back-end, el front-end y la base de datos, y debido a la complejidad que cada uno presenta, es importante elegir una tecnología adecuada para cada uno y establecer una forma de relacionar las componentes entre sí, esta puede apreciarse en la figura 3.1.

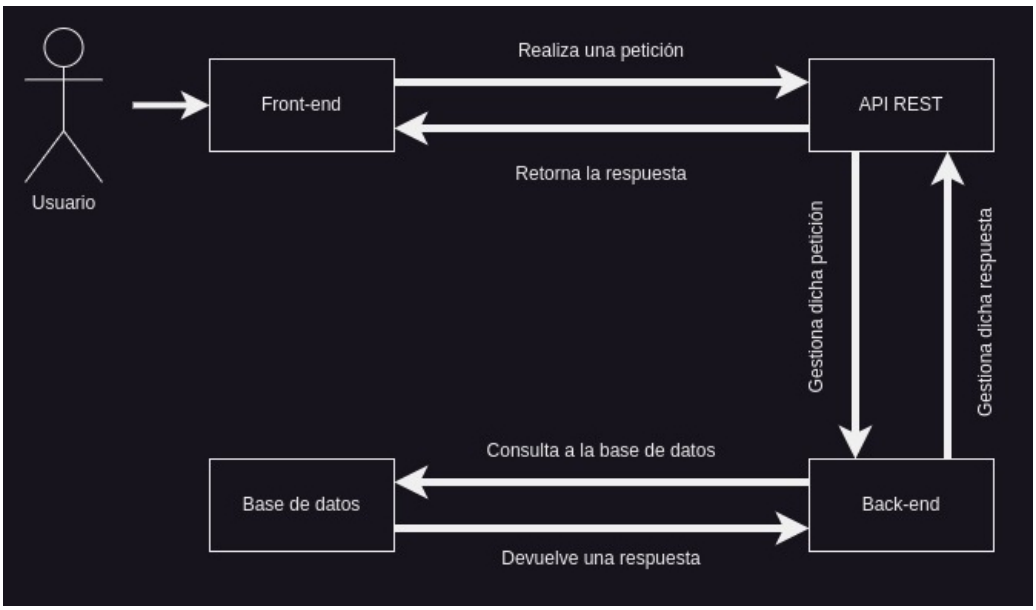


Figura 3.1: Relación entre el usuario, la aplicación, y las componentes de esta.

Para llevar a cabo la aplicación, y haciendo uso de las componentes mencionadas anteriormente, se ha seguido el desarrollo de una arquitectura hexagonal para las dos componentes principales de la aplicación, el front-end y el back-end. Esta arquitectura consiste en aplicar un enfoque de diseño de software que busca separar claramente las diferentes capas y responsabilidades de una aplicación, como se puede apreciar en la 3.2.

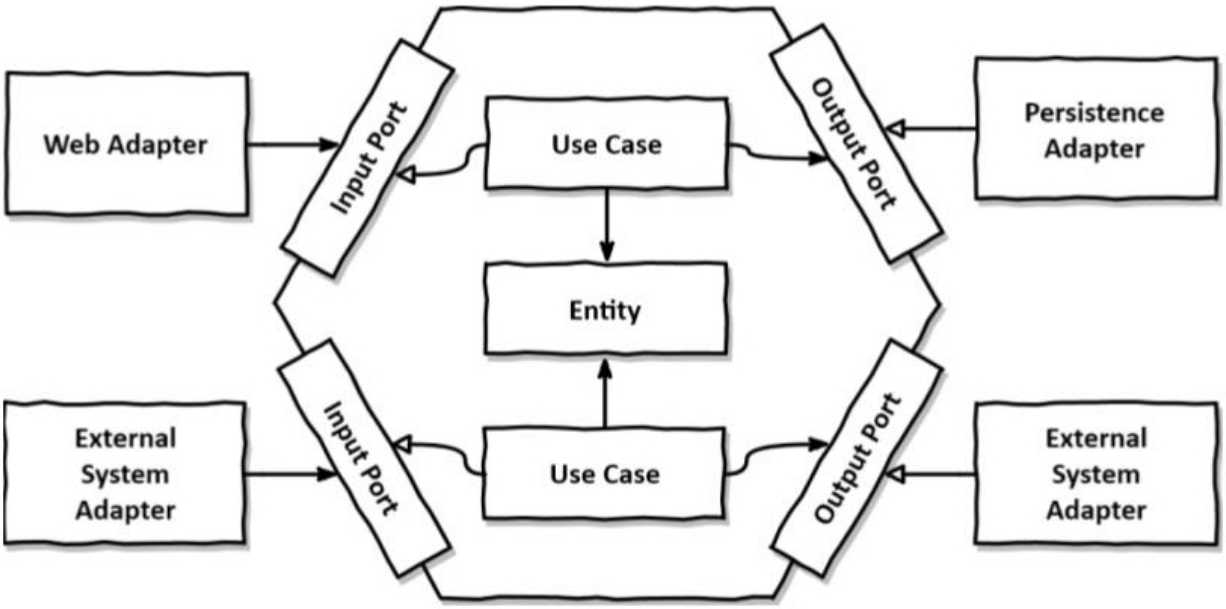


Figura 3.2: Funcionamiento de una arquitectura hexagonal.

En una arquitectura hexagonal, el núcleo de la aplicación, también conocido como dominio, se encuentra en el centro del hexágono. Esta capa contiene la lógica de negocio y las reglas específicas de la aplicación. El núcleo es independiente de cualquier framework o tecnología externa y define una serie de puertos o interfaces que describen cómo interactúa con el mundo exterior. Alrededor del núcleo se encuentran los adaptadores, que son los encargados de conectar el núcleo con los componentes externos, como la interfaz de usuario, las bases de datos, servicios web u otros sistemas. Estos adaptadores implementan los puertos definidos por el núcleo y adaptan las solicitudes y respuestas para que sean comprensibles para cada componente externo. El objetivo principal de esta arquitectura es permitir que el núcleo de la aplicación sea independiente de los detalles de implementación externos y facilitar la prueba y sustitución de componentes. Además, promueve la reutilización de código, ya que los adaptadores pueden ser intercambiados fácilmente sin afectar el núcleo de la aplicación.. A la vez, se ha desarrollado una API que permite gestionar las peticiones que se le realicen a la aplicación, por lo que en total poseemos cuatro componentes diferenciadas.

### 3.2.1. Back-end

Las tecnologías utilizadas en el desarrollo del Back-end se puede resumir en Java, apoyado principalmente en Spring[1], que es un framework de desarrollo de aplicaciones Java que ofrece un amplio conjunto de herramientas y funcionalidades para simplificar y

[1] https://spring.io/projects/spring-boot

agilizar el proceso de desarrollo como la inyección de dependencias o la integración de otras tecnologías como JDBC[2]. La elección de Spring como tecnología para llevar a cabo el Back-end obedece a que esta permite una fácil y sencilla configuración, inicio e integración, además de ser capaz de manejar numerosas peticiones al mismo tiempo y tiene un alto nivel de seguridad. Todos estos puntos hacen que el Back-end que se desarrolle sea muy robusto, y haciendo una gran demanda de la tecnología, siendo involucrada en la producción de numerosas empresas de gran calibre como Google, Microsoft o Amazon entre otras[3].

Dado que Spring es un marco de trabajo que emplea Java como lenguaje de programación, al iniciar el proyecto desde cero se requiere una herramienta apropiada. Es posible optar por Maven, la cual es una herramienta destinada a la gestión de dependencias y comprensión de proyectos, y se ha optado por su elección para mantener la misma estructura de archivos a lo largo del Trabajo de Final de Grado.

El Back-end posee una estructura de ficheros muy amplia, ya que cuenta con diferentes componentes importantes, dónde destacan el dominio, los serializadores y deserializadores, los repositorios, servicios y controladores. En primer lugar se encuentra el dominio, uno de los componentes más relevantes en un proyecto en Spring, y es que el dominio contienen las clases que representan a los objetos a almacenar, con sus respectivos atributos y métodos. En el proyecto el dominio consta de 7 entidades: Camión, Puerto, Buque, Escala, Atraque, Acceso y Operación, las relaciones y atributos de esta puede apreciarse en la figura 3.3.

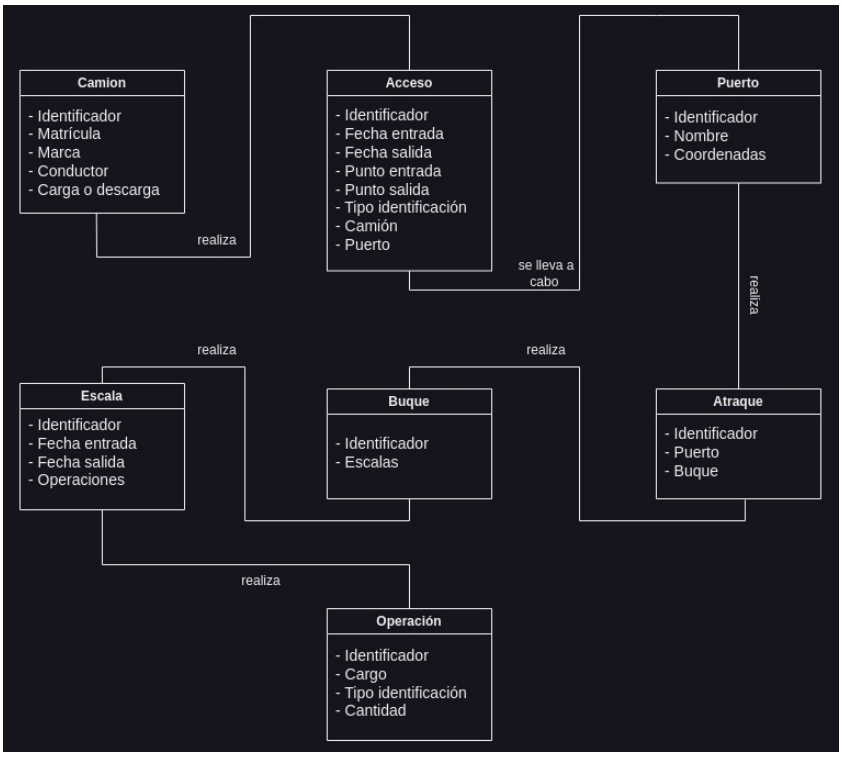


Figura 3.3: Diagrama de las entidades, sus atributos y relaciones.

[2]JDBC: API de Java que permite establecer conexiones con bases de datos, enviar consultas SQL, recuperar y manipular datos

[3]https://medium.com/@ktufernando/node-js-vs-spring-boot-cuál-elegir-5a687cd1abae

Las entidades son las siguientes:

- Camión: Almacenará la información relativa a los camiones y camioneros que deseen acceder a los puertos a realizar operaciones de carga o descarga de mercancías. Almacenará datos relativos al camión como la matrícula, constada de siete caracteres -cuatro números y tres letras- y marca, nombre del conductor, y si se dispone a realizar operaciones de carga o descarga.
- Puerto: Respecto al puerto este contendrá datos relativos al mismo, un identificador, su nombre y sus coordenadas geográficas, partiendo de la latitud y la longitud, pero representándolas en formato decimal.
- Buque: Del buque solo se almacena un identificador y una lista de las últimas escalas que haya realizado.
- Entidad Escala: Respecto a las escalas se almacena, un identificador, además del buque que la ha llevado a cabo y el atraque que se realizó, la fecha y hora tanto de embarque como de desembarque, y las operaciones que se llevaron a cabo en dicho puerto.
- Atraque: En términos del atraque se guardará un identificador único, además del puerto dónde se realizó y las coordenadas geográficas de este.
- Acceso: De cara a los accesos a los puertos, uno de los puntos más relevantes, se guardará su identificador, y las fechas y horas de entrada y salida, además de los puntos por los que se accederá y saldrá del recinto. Por último también se añadirá el puerto y el camión involucrados en este acceso, y la forma en la que el camionero se identifica en la instancia portuaria.
- Operación: Respecto a la operaciones se almacena el identificador único, además del tipo de operación (si consiste en carga o descarga de mercancía), además de la materia que se dispone y la cantidad de la misma.

Además, a fin de garantizar un funcionamiento óptimo de estos modelos, resulta imprescindible establecer un repositorio individual para cada uno. Dichos repositorios constituyen elementos vitales en la capa de persistencia de la aplicación, responsables de almacenar y recuperar las entidades requeridas de manera eficiente. Por ello, los repositorios extienden la interfaz de TimescaleRepository, y a su vez están vinculados de forma directa con la base de datos TimeScaleDB. Una de las mayores ventajas que ofrece SpringBoot es el poder usar la clase JdbcTemplate, y con la interfaz TimescaleRepository, de este modo, la integración del marco de trabajo con la base de datos se realiza de manera automática, sin necesidad de realizar configuraciones adicionales.

Para ser capaz de modificar los datos de los repositorios se dispone de un servicio exclusivo para cada uno en concreto. Como se puede apreciar por su nombre, estos pertenecen a la capa de servicios, y se encargan de resolver la lógica del negocio aplicativo.

Una vez los servicios han sido creados, estos son accedidos por los controladores, pertenecientes a la capa con el mismo nombre. El proyecto contiene tantos controladores como entidades, es decir, siete en total, y pertenencen a la API REST que se ha desarrollado.

### 3.2.2. API REST

La API REST la conforman dos componentes principales, el primero de ellos, los controladores son los encargados de recibir y devolver información frente a las peticiones HTTP. Una petición HTTP es un mensaje enviado por un cliente para añadir, editar, actualizar o borrar información del servidor. Los diferentes métodos HTTP implementados son los siguientes:

- **GET:** Solicita información al back-end de la aplicación.
- **POST:** Envía una petición de inserción al al back-end.
- **PUT:** Comunica al back-end la intencionalidad de modificar datos.
- **DELETE:** Envía una solicitud al back-end para eliminar información.

Para realizar estás peticiones se debe hacer desde una dirección concreta, que dependerá de la entidad que se quiera acceder, aunque todas radican en un punto de origen, la dirección siempre comenzará por *http://localhost:8080/*, dirección dónde se encuentra desplegado el Back-end y desde dónde se atenderán las diferentes peticiones.

A partir de aquí se añaden diferentes terminaciones o *"endpoints"* a la dirección anterior para acceder a las diferentes entidades:

- ***/trucks*** Acepta las siguientes peticiones:
  - GET: Aportando al endpoint el identificador del camión a consultar *(/trucks/id")* se obtendrá el camión de vuelta si existe con código *HTTP 200*, indicando que ha sido satisfactoria la búsqueda y obtención del ítem.
  - POST: En este caso la operación es algo más compleja, pues requiere de una estructura conformada por un encabezado (*H 'Content-Type: application/json' -d"*), seguida de un objeto de tipo json, -estructura de datos que se utiliza para almacenar información de manera organizada-, si el cuerpo es correcto y no se encuentra ya definido en la base de datos, se devuelve el código *HTTP 200*. En cambio, si el ítem ya existe en la base de datos este no se insertará, y se retornará el código *HTTP 500 Internal Server Error*. También se puede producir otro caso, y es que el cuerpo del objeto JSON sea incorrecto, y entonces se obtenga el código *HTTP 400 Bad Request*, informando que tampoco se ha insertado el objeto.
  - PUT: Bastante similar a la operación POST, requiere del encabezado y del cuerpo conformado por el objeto json, pero a su vez al final del *Endpoint*, y así como en la petición GET, requiere del id del camión a modificar. Nuevamente si es correcto se obtendrá el código *HTTP 200*, pero así como en la operación POST, si el camión no existe se retornará el código *HTTP 500 Internal Server Error*, y si el cuerpo es incorrecto *HTTP 400 Bad Request*.
  - DELETE: Esta petición requiere exclusivamente de la adición del id del camión al término del endpoint, con ello ya es suficiente para llevar a cabo la acción de borrado, y nuevamente, mostrará código *HTTP 200* si el borrado es exitoso, y *HTTP 500 Internal Server Error* si no encuentra el camión a borrar.
- ***/ports*** Acepta las siguientes peticiones:

- GET: Añadiendo al endpoint el identificador del puerto a consultar se retornará, si existe, el puerto con código *HTTP 200*.
- POST: Añadiendo un encabezado correcto y un cuerpo con un objeto json para el puerto correctamente formateado, este será añadido a la base de datos.
- PUT: Nuevamente, adjuntando al final del *endpoint* el id del puerto en a modificar y su correspondiente encabezado y cuerpo se actualizaría el puerto indicado por el identificador con el nuevo adjunto en el cuerpo.
- DELETE: En este caso, añadiendo el identificador del puerto a eliminar al término del *endpoint*, este sería borrado si existía previamente en la base de datos.

- ***/vessels*** Acepta las siguientes peticiones:
  - GET: Proporcionando el identificador del buque al endpoint correspondiente, se obtendrá el buque de vuelta si existe, con código *HTTP 200*.
  - POST: Mediante la inclusión de un encabezado correcto y un cuerpo con un objeto JSON que contenga la información del buque correctamente formateada, este se añadirá a la base de datos.
  - PUT: Una vez más, al agregar al final del endpoint el ID del buque que se desea modificar, junto con el encabezado y cuerpo correspondientes, se actualizará el buque indicado por el identificador con los nuevos datos proporcionados en el cuerpo.
  - DELETE: En este caso, al añadir el identificador del buque que se desea eliminar al final del endpoint, dicho buque será eliminado de la base de datos si existía previamente.

- ***/scales*** Acepta las siguientes peticiones:
  - GET: Aportando al endpoint el identificador de la escala a consultar se obtendrá la escala de vuelta si existe, con código *HTTP 200*.
  - POST: Usado para crear escalas, aportando un cuerpo con un json correctamente formateado para una escala, además de su correspondiente encabezado, se insertará sin fallo la escala en la base de datos.
  - DELETE: En este caso, sin requerir de un cuerpo ni encabezado, siendo necesario solamente añadir el identificador de la escala a eliminar al final del *endpoint* se llevará correctamente a cabo esta petición, retornando código *HTTP 200*.

- ***/berths*** Acepta las siguientes peticiones:
  - GET: Al proporcionar el identificador del atraque al endpoint correspondiente, se obtendrá el atraque de vuelta si existe con código *HTTP 200*.
  - POST: Mediante la inclusión de un encabezado correcto y un cuerpo con un objeto JSON que contenga la información del atraque correctamente formateada, este será añadido a la base de datos.
  - PUT: Nuevamente, al adjuntar al final del endpoint el ID del atraque que se desea modificar, junto con el encabezado y cuerpo correspondientes, se actualizará el atraque indicado por el identificador con los nuevos datos proporcionados en el cuerpo.

    - DELETE: En este caso, al agregar el identificador del atraque que se desea eliminar al final del endpoint, dicho atraque será eliminado de la base de datos si existía previamente.

- ***/access*** Acepta las siguientes peticiones:

    - GET: Añadiendo al endpoint el identificador del acceso a consultar se obtendrá con código *HTTP 200* dicho acceso, si se encuentra en la base de datos.
    - POST: Se usa para añadir accesos a la base de datos, pero requiere de cuerpos con encabezados y objetos json correctos o retornará *HTTP 400 Bad Request*, o si la petición es correcta pero ya existe dicho acceso en la base de datos se devolverá el código *HTTP 500 Internal Server Error*.
    - DELETE: Permite borrar accesos de la base de datos si previamente ya existen y el identificador adjuntado al final del *endpoint* es correcto.

- ***/operations*** Acepta las siguientes peticiones:

    - GET: Proporcionando al endpoint el identificador de la operación a consultar, se obtendrá la operación correspondiente si existe, con código *HTTP 200*.
    - POST: Utilizado para crear operaciones, al proporcionar un cuerpo con un JSON correctamente formateado para una operación, junto con su correspondiente encabezado, se insertará la operación en la base de datos sin errores.
    - DELETE: En este caso, sin requerir un cuerpo ni encabezado, solo es necesario agregar el identificador de la operación a eliminar al final del endpoint. Al hacer esto, se llevará a cabo correctamente la solicitud y se devolverá el código *HTTP 200*.

Antes de que las peticiones sean tratadas, estás pasan por los otros adaptadores del API REST, los serializadores y deserializadores, especialmente estos últimos son los que se encargan de, cuando se recibe una petición, convertir los los objetos JSON en sus correspondientes entidades, para que luego los controladores de la propia API REST puedan tratarlos en términos de entidades.

Una vez se obtienen los resultados de la consulta, y de cara a dar una respuesta a la petición del usuario, si se va a retornar algún elemento de la base de datos, este pasará por los serializadores para ser convertido al formato JSON con el que se hizo la petición.

### 3.2.3. Base de datos

Debido a que la aplicación se basa fundamentalmente en la inserción y consulta de datos, es de vital importancia un buen desarrollo de esta. En este caso se ha usado una base de datos SQL TimescaleDB. Esta base de datos relacional está diseñada para manejar datos de series temporales a gran escala. Combina las capacidades de una base de datos relacional con las optimizaciones y funcionalidades específicas para datos de series temporales, y debido al enfoque temporal que poseen los datos primordiales (especialmente los de reserva de horarios), cobra una gran relevancia este punto. Algunas características a resaltar de TimeScaleDB:

- Utiliza algoritmos de compresión específicos para datos de series temporales, lo que reduce significativamente el espacio de almacenamiento requerido.

- Ofrece escalabilidad horizontal y vertical. Puede manejar grandes volúmenes de datos y escalar a través de múltiples nodos para soportar cargas de trabajo intensivas.
- Utiliza una arquitectura de almacenamiento optimizada para consultas rápidas y eficientes en datos de series temporales. Emplea técnicas como particionamiento automático, compresión de datos, indexación y agregaciones rápidas para mejorar el rendimiento de las consultas.
- Se distribuye bajo la licencia de código abierto Apache 2.0, lo que significa que es gratuito para su uso en proyectos comerciales y de código abierto.
- Al ser de código abierto posee una gran comunidad de desarrolladores, ofreciendo un buen soporte en base a cuestiones de los usuarios que hayan podido tener previamente.

### 3.2.4. Front-end

Para el desarrollo del front-end se ha hecho uso de dos tecnologías principales, la primera de ellas es Typescript[4], un lenguaje de programación que se basa en JavaScript. Es conocido como un superconjunto de JavaScript, lo que significa que cualquier programa JavaScript válido también es un programa válido en TypeScript, heredando todas las características de este otro lenguaje e incorporando características adicionales, como un sistema de tipos estático que permite detectar errores y mejorar la calidad del código. Además, se ha hecho uso también de Vue.js[5] para el desarrollo de la interfaz de usuario. Este es un *framework* progresivo con una curva de aprendizaje bastante accesible. A la hora de elegir tecnologías, existen otros *frameworks* populares como pueden ser *React*[6] o *Angular*[7], aunque de cara a desarrollar este poyecto se ha optado por la elección de Vue, que en comparacińn con los dos previamente mencionados, este tiene una mejor curva de aprendizaje, debido a aspectos como su baja comlejidad al momento de crear un proyecto, o el uso de una sintaxis sencilla, ya que mantiene separado en el fichero diferentes secciones, siendo una para HTML, otra para TypeScript, y otra para los estilos a aplicar. Vue.js posee múltiples librerías, y algunas de ellas se han utilizado en el desarrollo del proyecto:

- Vuetify[8]: Una librería de componentes, en esta aplicación se usa esta librería exclusivamente en la página principal, pues una componente como son las tarjetas de la página de inicio son hechas en base a esta librería, una vez se ha añade e implementa, se le incluyen adaptaciones para que esta cumpla con los requisitos de la interfaz de usuario deseada.
- Vue Router[9]: Una librería de enrutamiento, que permite asignar a cada una de las páginas de la aplicación su dirección URL correspondiente, aunque no ha recibido configuraciones adicionales más allá de las que incorpora, es utilizado en las diversas componentes que redirigen a otras páginas, tales como la barra superior o las

[4]Typescript: https://www.typescriptlang.org
[5]Vue.js: https://vuejs.org
[6]React: https://es.react.dev
[7]Angular: https://angular.io
[8]Vuetify: https://vuetifyjs.com/en/
[9]Vue Router: https://router.vuejs.org

tarjetas de la página de inicio, aunque también es recurrente su uso en la consulta de los registros individuales, accesibles desde los registros principales de los accesos y los atraques.

- Pinia[10]: Facilita la administración de estado para aplicaciones Vue. Se ha utilizado con el objetivo de proporcionar una solución simple y eficiente para gestionar el estado de la aplicación, posee una sintaxis intuitiva y ayuda a mantener una arquitectura hexagonal en el desarrollo del front-end. Su uso es primordial en el aplicativo, pues permite mantener la arquitectura hexagonal de desarrollo, al hacer uso de pinia se mantienen claramente diferenciados los componentes de los servicios y repositorios que requiere el front-end para su correcto funcionamiento, pues pinia hace de intermediario entre los dos, y de querer cambiar los servicios y repositorios no sería requerido modificar las componentes, así como si se quisiera cambiar de tecnología en las componentes, no sería necesario actualizar los servicios y repositorios, tan solo el gestor de estados que hace uso de esta tecnología.

A su vez, también se aplica el uso de *Leaflet*[11], una librería de código abierto en Javascript que permite visualizar mapas y personalizarlos. Existen otras tecnologías que se pueden utilizar como la de Google Maps[12] pero *Leaflet* al ser de código abierto ofrece una gran ventaja sobre esta, y es que la API de Google Maps requiere de insertar métodos de pago y es más compleja de utilizar. Principalmente por esta causa, y por la mayor facilidad que ofrece *Leaflet* al tener una mayor comunidad de uso por ser de código abierto, se opta por el uso de esta.

Otras tecnologías que se aplican en el desarrollo del front-end son *jQuery*[13], una biblioteca que facilita la interacción con elementos HTML. Y también se utiliza *Bootstrap*[14], que es un framework utilizado para crear sitios web responsivos, debido a que proporciona una serie de estilos CSS predefinidos, componentes y scripts de JavaScript que facilitan la creación de interfaces de usuario.

### Estructura de directorios

La estructura del front-end también sigue un esquema hexagonal, por lo que, de forma similar a como se desarrolla el back-end, está dividida en diferentes directorios y ficheros relevantes:

- */public*: En este directorio podemos encontrar los archivos estáticos, que al ser colocados en esta ruta se asegura que sean servidos directamente por el servidor web sin ningún tipo de procesamiento adicional, lo que mejora la eficiencia y el rendimiento.

- */src/adapter/rest*: En esta ruta encontramos los adaptadores que implementan los respositorios del front-end, con el objetivo de recuperar cada uno de los recursos de la API del back-end.

[10] Pinia: https://pinia.vuejs.org
[11] Leaflet: https://leafletjs.com
[12] Google Maps: https://developers.google.com/maps
[13] jQuery: https://jquery.com
[14] Boostrap: https://getbootstrap.com

- */src/adapter/vuejs*: En esta ruta encontramos todos los componentes definidos de la apliación, algunos de ellos reutilizables como es el caso de las tablas, y otros que consisten en las componentes de cada ruta de la interfaz de usuario.
- */src/adapter/vuejs/stateManager*: Con el fin de mantener un código mantenible, es necesario que los componentes de vue no hagan uso de los servicios y repositorios, es por ello que se dispone de los gestores de estados, que conectan a los diferentes componentes con los servicios y repositorios que requieran usar
- */src/application/repository*: Este directorio contiene interfaces que definen cómo se recuperan o almacenan los datos desde el back-end.
- */src/application/service*: En esta ruta encontramos los diferentes servicios de los que hace uso la interfaz de usuario, cada uno se encarga de satisfacer los casos de uso de la aplicación, de forma similar a como se hace en el back-end.
- */src/domain/entity*: Este directorio contiene una entidad representando a su correspondiente del back-end
- */src/domain/valueobject*: A su vez, este directorio contiene los tipos de datos que son utilizados por las diferentes entidades.
- */src/router/index.ts*: Este fichero contiene la configuración de las rutas de la aplicación y como van a ser utilizadas.
- */src/utils*: Esta ruta contiene diferentes clases e interfaces que ayudan a tener un código más limpio
- */src/main.ts*: El fihero *main.ts* es el principal del proyecto, pues se encarga de arrancarlo y es insertado en el *index.html* principal.

# Capítulo 4

# Funcionamiento de la aplicación

El funcionamiento del aplicativo se puede dividir en diferentes etapas, las **vistas**, las **operaciones** y la **resolución** de estas.

## 4.1. Vistas

Uno de los principales aspectos de este Trabajo de Fin de Grado es la implementación de diferentes vistas que cumplan con los casos de uso de la aplicación, esta es la presentación de la aplicación al usuario y la forma que dispone de interactuar con ella. La primera vista que se encuentra el usuario es la página principal (Figura 4.1).

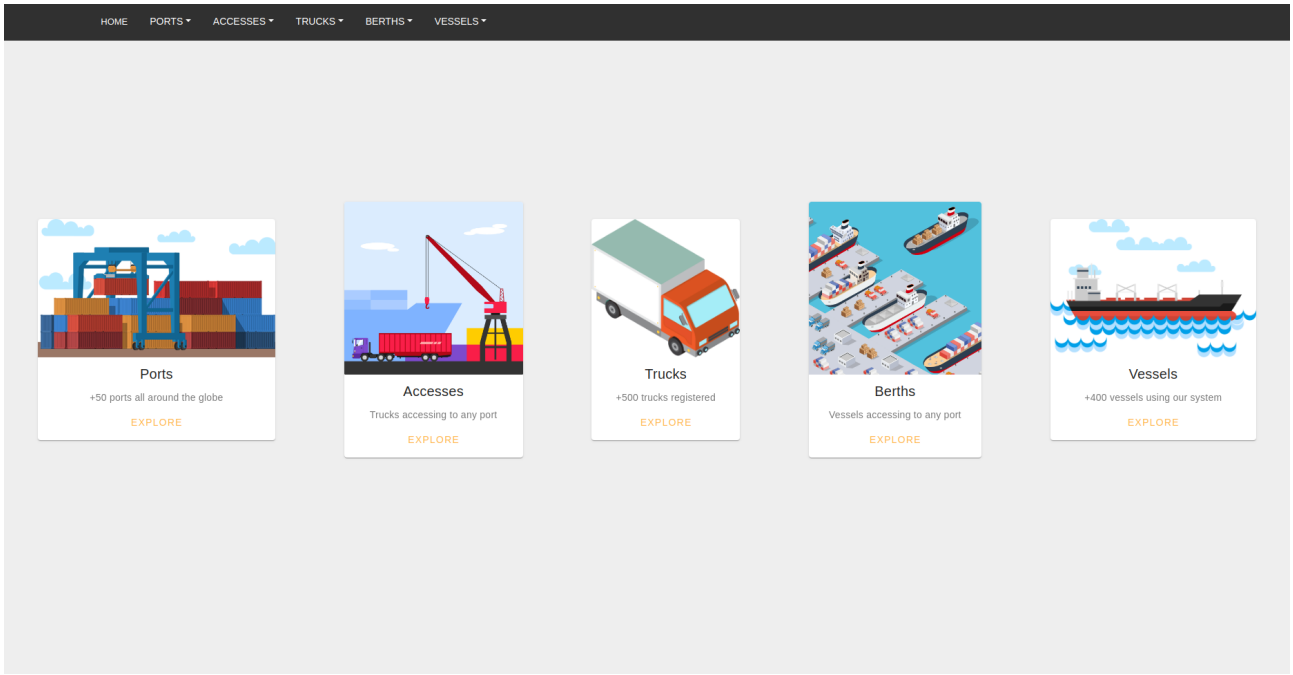


Figura 4.1: Vista de la página de inicio.

Esta vista presenta al usuario la aplicación, pudiendo moverse por los diferentes registros por medio de las tarjetas que se presentan en la propia pantalla, o bien accediendo a los diferentes paneles desde la barra superior, que permite acceder a estos registros o bien a los formularios de inserción de datos.

Si desde la página principal se selecciona la tarjeta de los puertos y se clica en el botón "*Explore*", se accede al panel que muestra toda la información contenida en la base de datos de los puertos, con los diferentes datos relativos a las entidades que previamente se han explicado, y con la posibilidad de adaptar los resultados según convenga al usuario, ya que se puede filtrar por el campo de texto superior derecho, así como seleccionar cuantos resultados desea por página. Para cambiar de página, se puede hacer uso de los botones que se encuentran debajo de la tabla, estos permiten desplazarte (en orden de izquierda a derecha, a la primera, a la inmediatamente inferior, a la página superior, o a la última página de resultados). Inmediatamente debajo de la tabla encontramos un mapa interactivo, en el cual el usuario puede desplazarse, hacer zoom o alejarse y clickar en él. Precisamente al hacer click aparece un marcador que indica las coordenadas seleccionadas, siendo así posible elegir un punto concreto del mapa a la hora de querer editar las coordenadas de un puerto ya existente. Finalmente se encuentran botones que permiten borrar, por medio del botón *DELETE* o actualizar los resultados -usando el botón *UPDATE*-, introduciendo el identificador del puerto en cuestión, o borrar todos los resultados de la base de datos clickando en el botón *DELETE ALL*. (Figura 4.2).

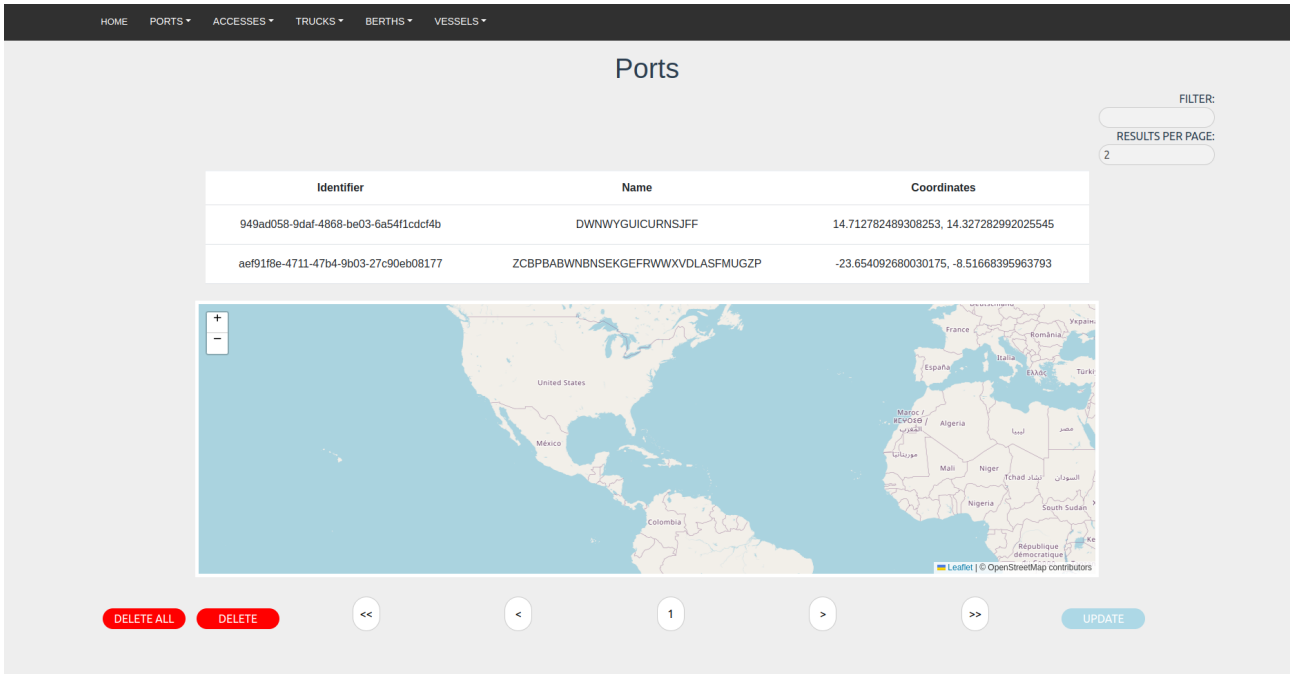


Figura 4.2: Vista del registro de puertos.

También podemos acceder, al formulario de inserción de nuevos puertos. Para ello se debe clickar en la barra superior en el apartado *"PORTS"*, y en la opción desplegable seleccionar *"INSERT PORT"*, una vez ahí nos llevará al formulario en cuestión (Figura 4.3). En él se puede introducir un nombre y elegir una coordenadas en el mapa que aparece a la derecha. Como en el mapa presente en el registro de puertos, este también es interactivo, es decir, al clickar aparece un marcador y el campo de coordenadas automáticamente es actualizado con el valor seleccionado desde el mapa. De esta forma el usuario puede elegir las coordenadas del puerto sin requerir hacer uso de una herramienta externa, tan solo debe conocer la ubicación aproximada del puerto y seleccionarlo en el mapa. Finalmente, cuando se pulsa el botón *"Submit port"* el puerto se envía a la base de datos para ser almacenado, si el proceso y los datos son correctos, aparecerá una notificación en la esquina superior derecha indicando la correcta inserción del puerto, en caso contrario,

aparecerá una notificación de fallo con información relativa al mismo.

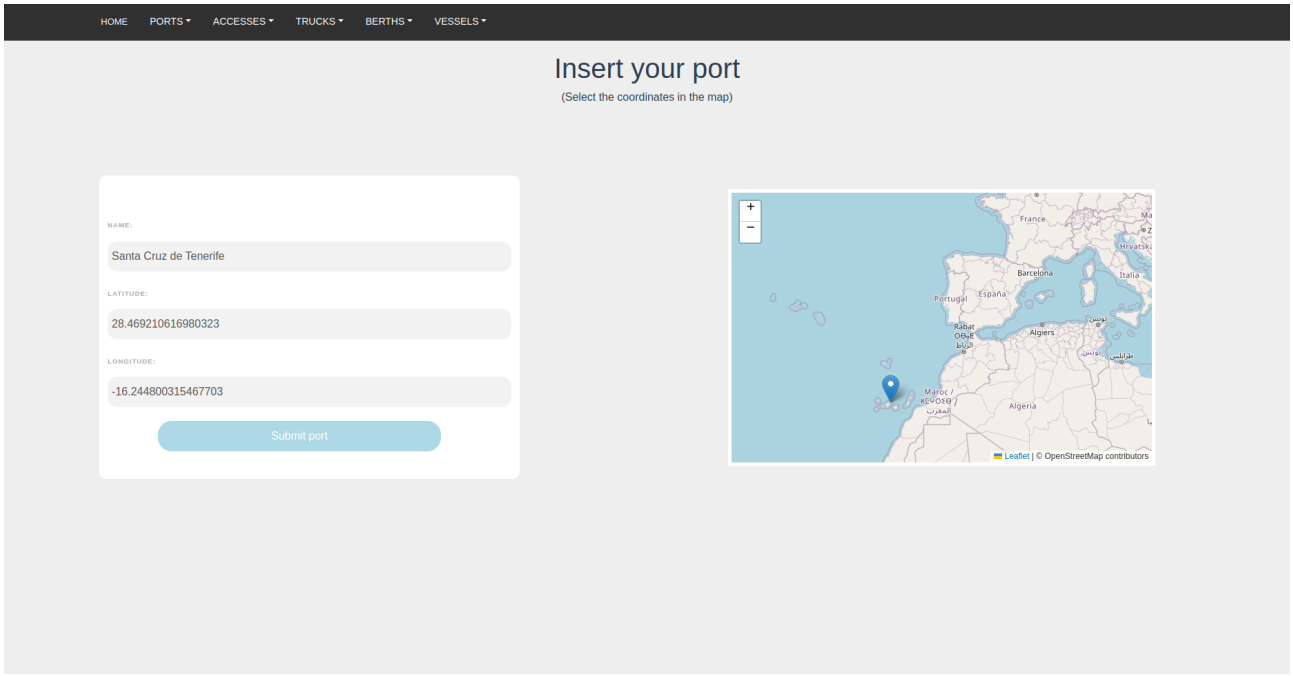


Figura 4.3: Vista del formulario de registro de puertos.

Si en lugar de los puertos, se opta por consultar el registro de camiones, bien desde la página principal en la tarjeta *"Trucks"* y pulsando sobre el botón *"Explore"* , o desde la barra superior en la opción *"TRUCK REGISTER"* se llegará a la página que muestra el registro de camiones. Esta presenta atributos relativos a los mismos tales como la matrícula o el conductor autorizado, y nuevamente presenta las opciones de filtrar resultados y consultar diferentes páginas, así como actualizar o borrar camiones concretos, o bien eliminar todos los camiones de la base de datos. (Figura 4.4)

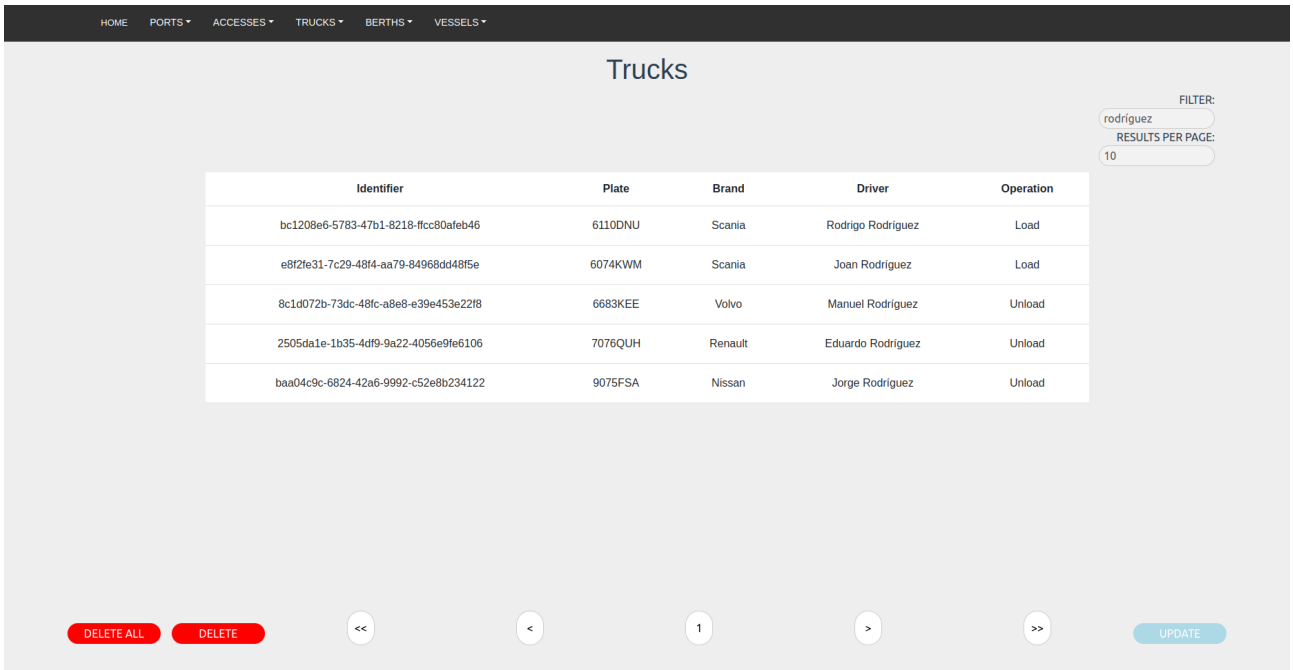


Figura 4.4: Vista del registro de camiones filtrando los resultados por el contenido de la palabra ’rodríguez’.

Además del registro, también se pueden insertar camiones, para ello se debe acudir a la barra superior, y en la opción desplegable *“TRUCKS"* se selecciona posteriormente *“INSERT TRUCKS"* y se accede a dicho formulario. EN este se deben introducir una matrícula correctamente formateada, una marca, y el nombre completo del conductor, así como si las operaciones que se dispone a hacer son de carga o de descarga. (Figura 4.5)

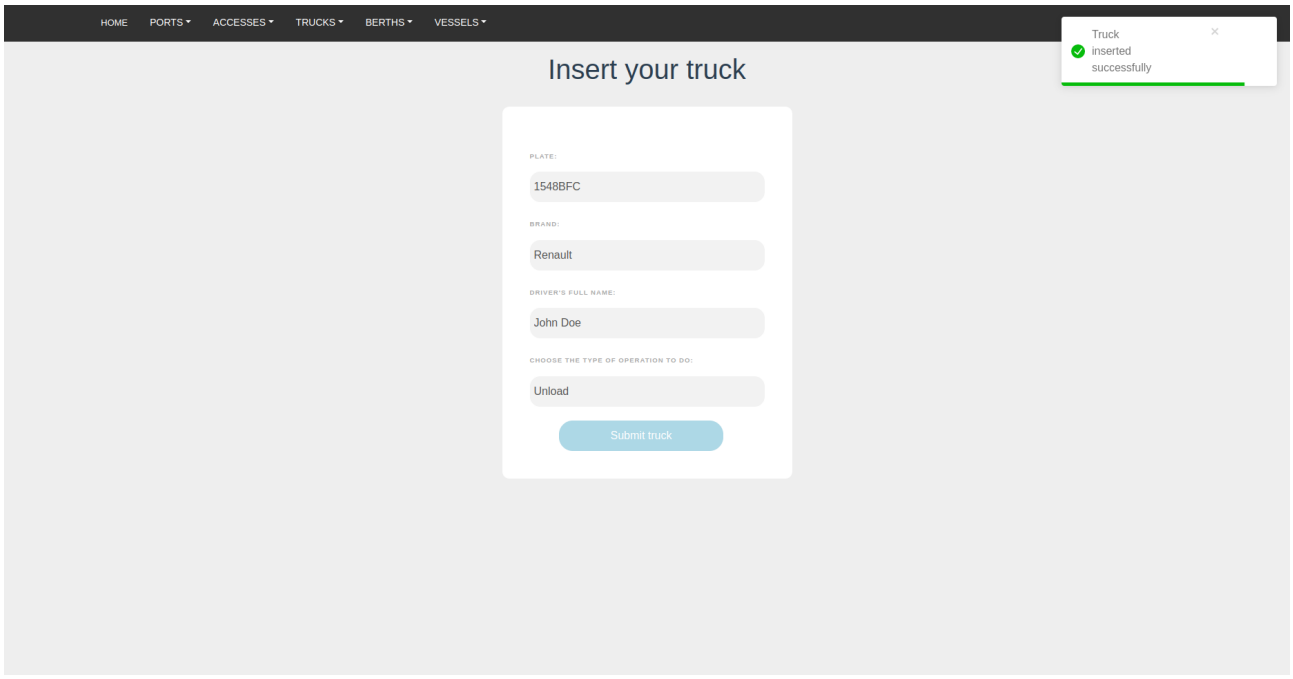


Figura 4.5: Vista del formulario de registro de camiones.

La entidad que relaciona a los camiones y puertos es el acceso, y estos se pueden insertar accediendo desde la barra superior, a la opción desplegable *“ACCESSES"*, seleccionando luego la opción *"INSERT ACCESS"*, una vez en el formulario, introduciendo identificadores de camiones y puertos, fechas y horas de entrada y salida, así como puntos de acceso y salida del puerto, y forma de identificarse en el acceso, siempre y cuando los datos aportados sean correctos y el camión y puerto indicado exista previamente en la base de datos, se registrará correctamente el acceso (Figura 4.6).

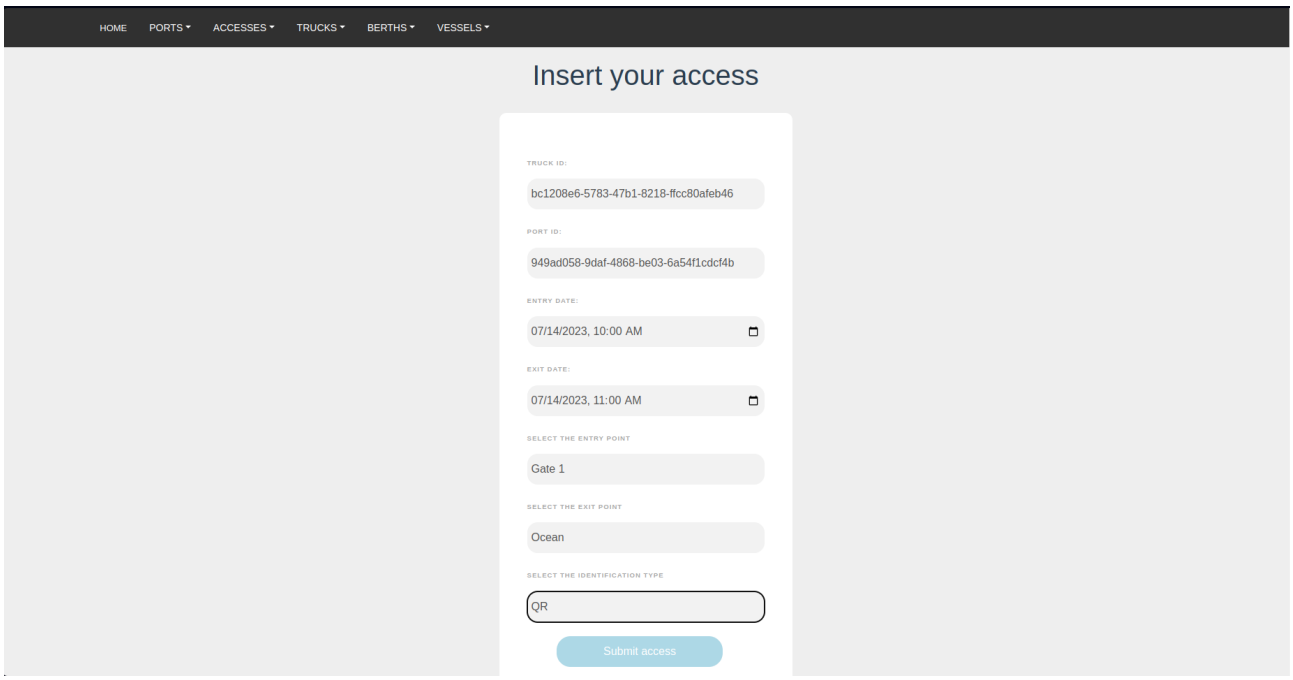


Figura 4.6: Vista del formulario de registro de accesos.

A la hora de querer consultar el registro de accesos este podrá ser accedido desde la página principal, en su correspondiente tarjeta, o bien desde la barra superior, en la opción desplegable *"ACCESSES"*, y posteriormente elegir *"ACCESS REGISTER"*, una vez en el formulario (figura 4.7), si el usuario selecciona los diferentes identificadores de camiones o puertos, podrá acceder a una tabla con el registro de dicho puerto o camión, como se puede apreciar en la Figura 4.8.

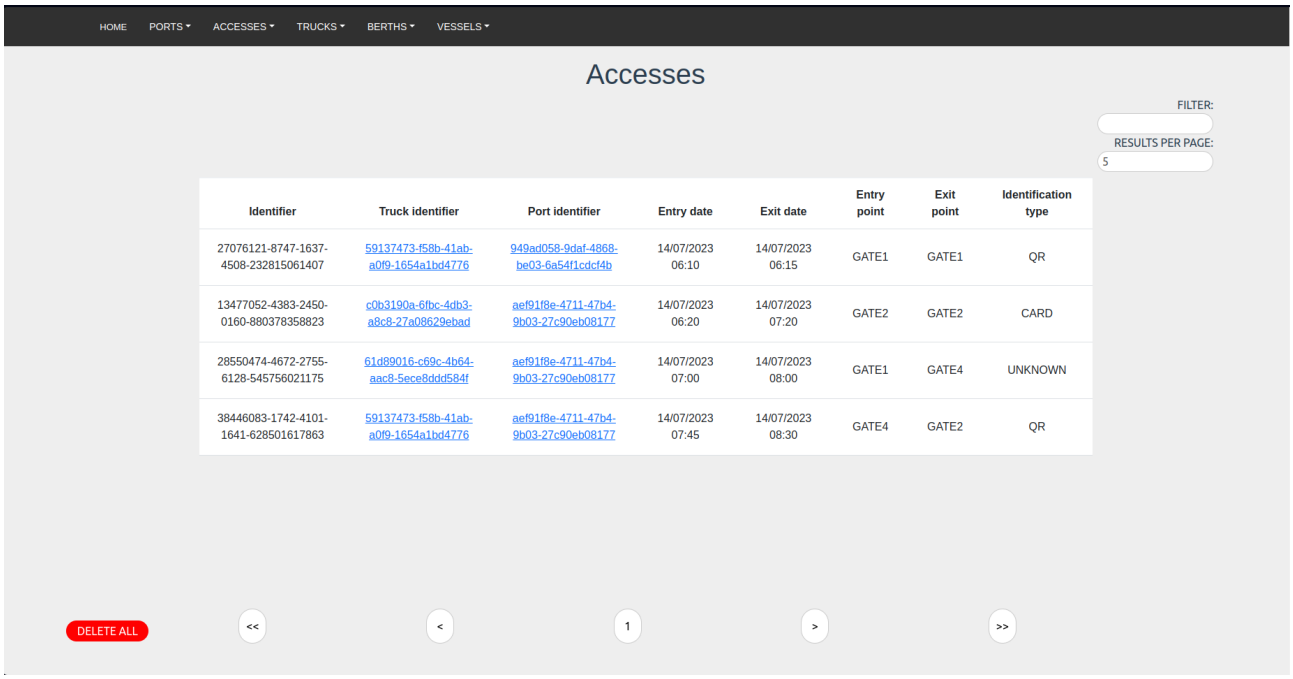


Figura 4.7: Vista del registro de accesos.

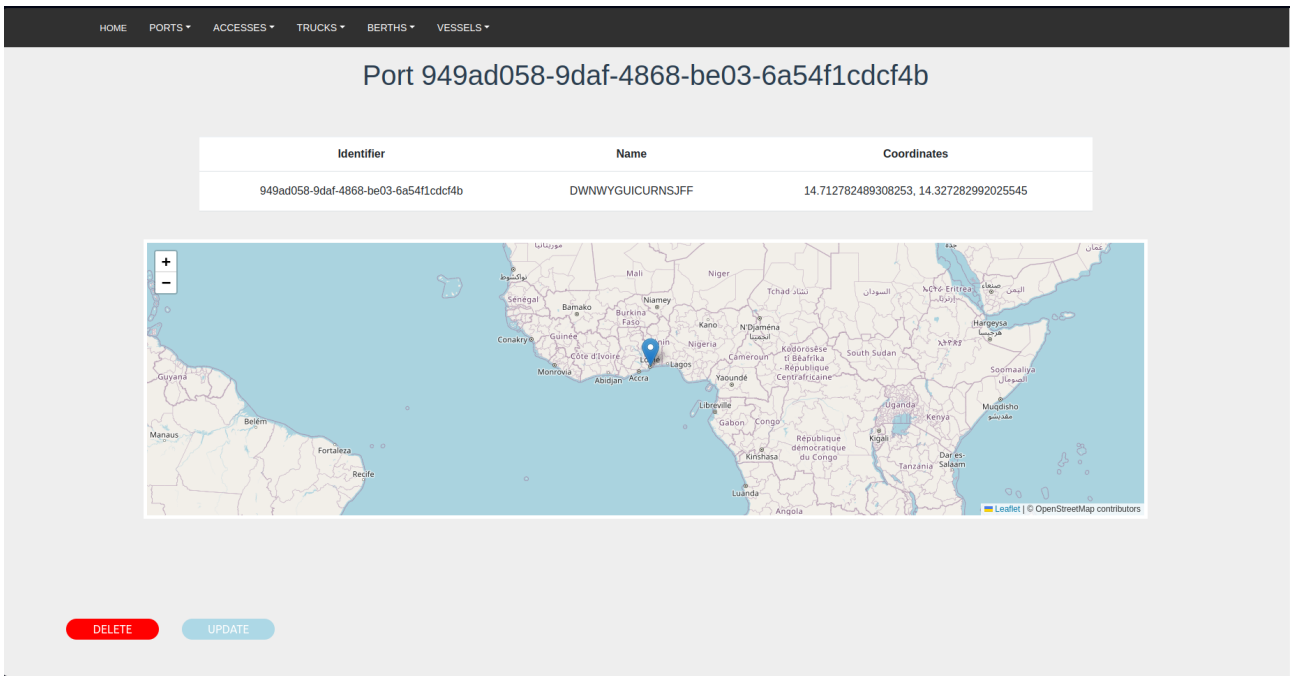


Figura 4.8: Vista del registro de un puerto específico.

Desde la barra superior, también se puede acceder a la opción desplegable *"VESSELS"*, en la cuál, al seleccionar la opción *"INSERT VESSEL"*, permite añadir un nuevo buque a la base de datos. Este formulario incluye tres campos diferentes, siendo el primero de ellos las operaciones, este contiene un menú desplegable con los diferentes materiales que se vayan a tratar en la operación, a su vez, el siguiente campo perite introducir entre si es una operación de carga o descarga. Finalmente el último es un campo numérico para indicar la cantidad de material de la operación. El formulario que se encuentra inmediatamente a la derecha contiene las escalas a insertar, en las cuáles solo se debe indicar la fecha y hora de entrada y de salida, además de las operaciones que esta contenga. Por último encontramos el formulario de los buques, el cuál contiene las escalas insertadas y el botón de confirmar buque. Para que las oepraciones sean efectivas en las escalas, o bien las escalas en los barcos, se debe pulsar sobre el botón azul *.Append"* para añadirlas. Una vez se desee confirmar el buque, se pulsa el botón azul *"Submit"* y este es enviado a la base de datos para ser almacenado (Figura 4.9).

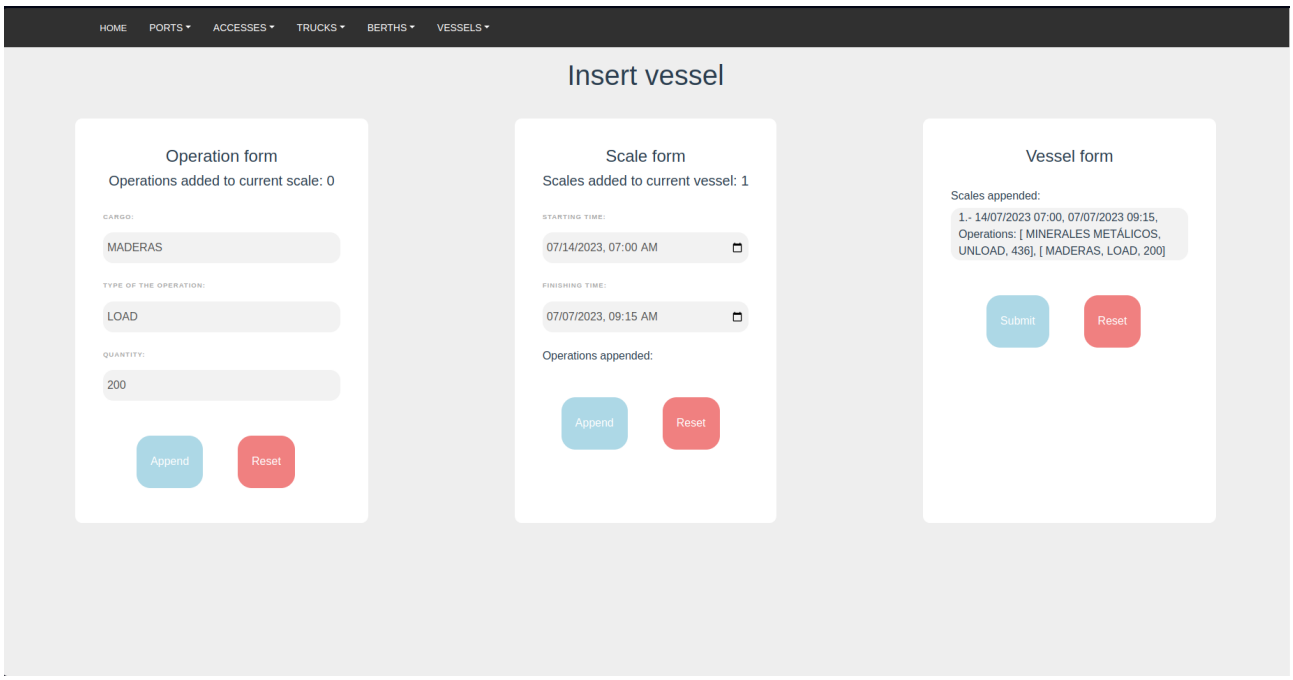


Figura 4.9: Vista del formulario de registro de buques.

Además del formulario de inserción, también se puede consultar el registro de buques (Figura 4.10), el cuál difiere de los previos en que posee un botón para consultar todas las escalas registradas en dicho barco (Figura 4.11), redirigiendo al registro de dicho buque exclusivamente. A su vez desde esta vista se podrá consultar las diferentes operaciones de cada escala con un botón propio que permite consultarlas, accediendo a una vista nueva que contiene todas las operaciones realizadas en una escala. (Figura 4.12)

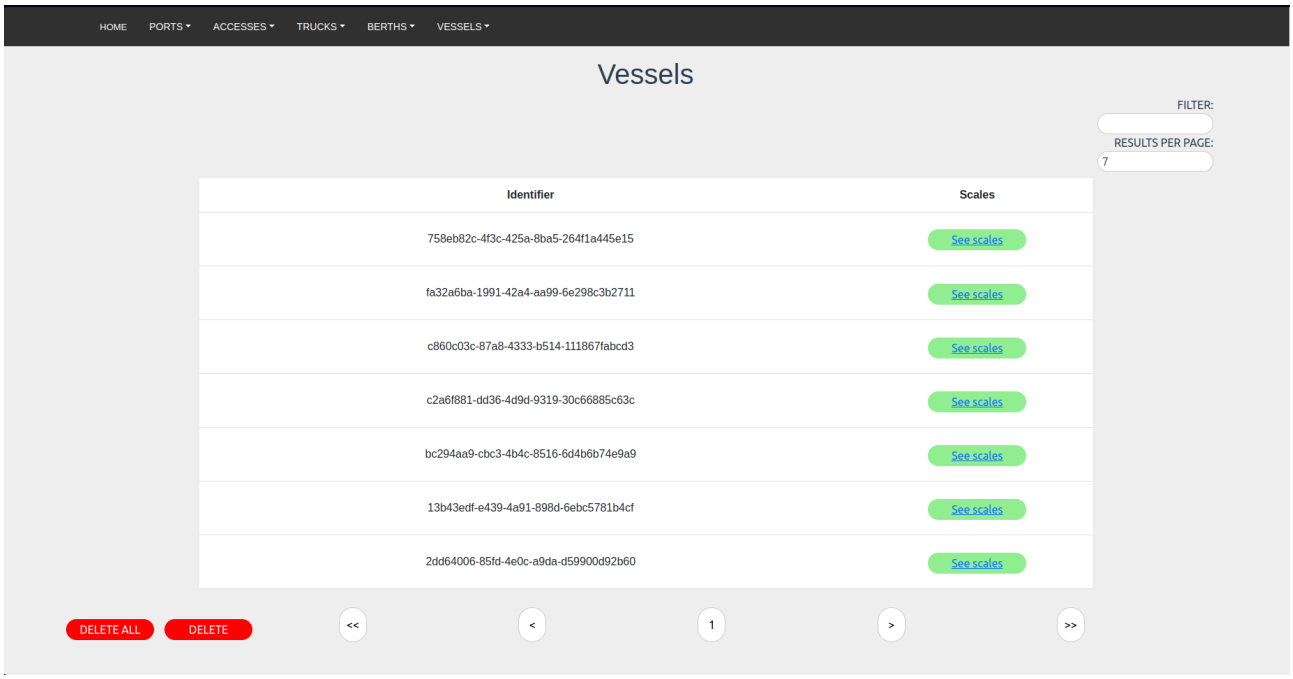


Figura 4.10: Vista del registro de buques.

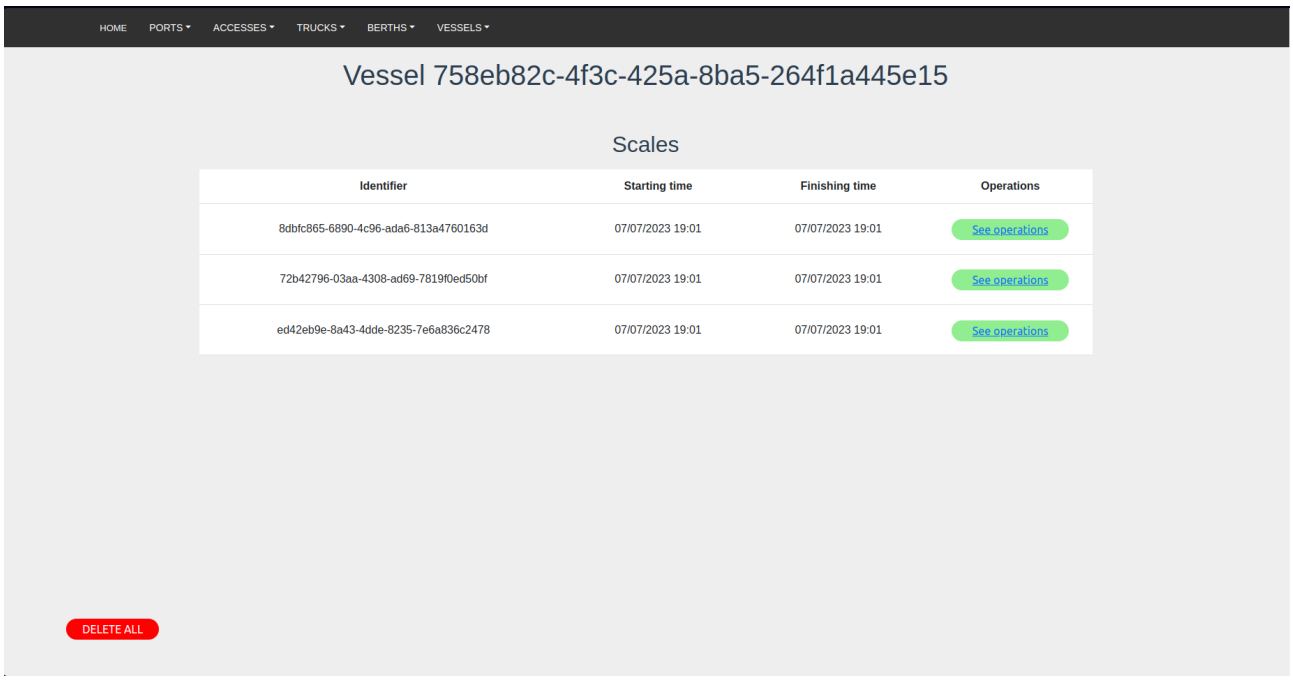


Figura 4.11: Vista del registro de un buque concreto.

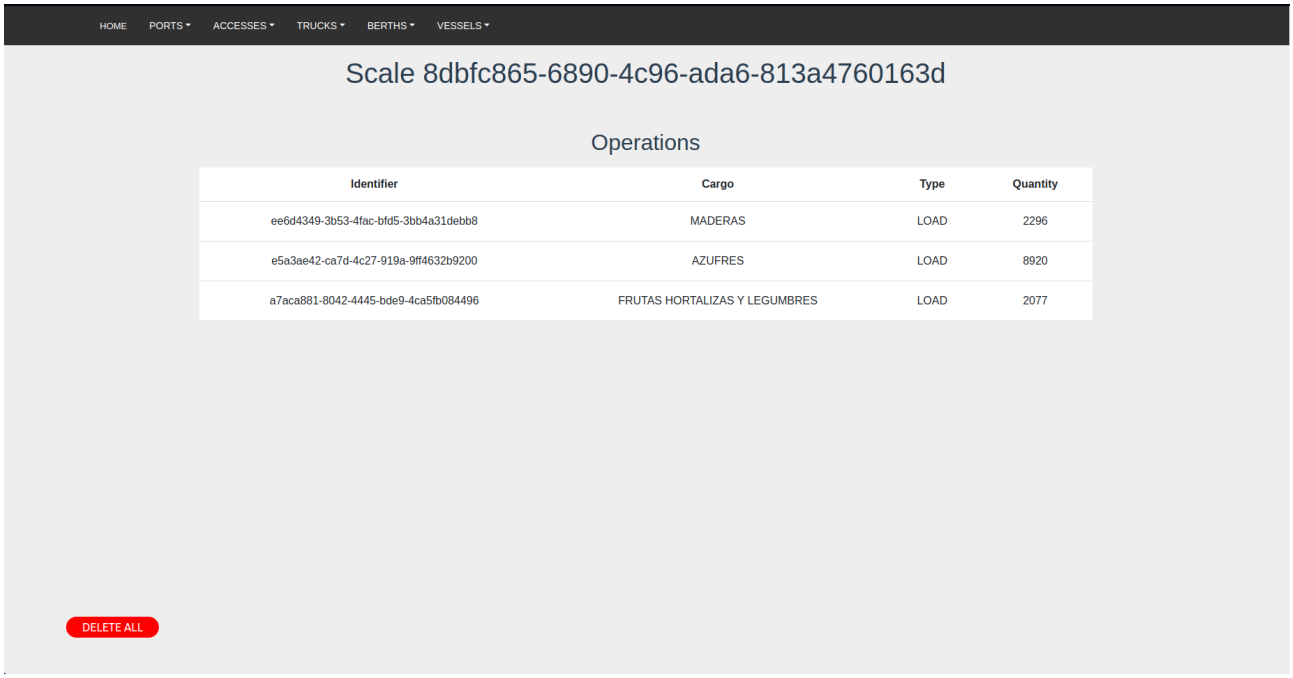


Figura 4.12: Vista del registro de una escala concreta.

El último formulario que se encuentra en el front-end es el correpondiente a los atraques, accesible desde la barra superior en el menú depslegable *'BERTHS"*, y seleccionando la opción *'INSERT BERTH"*. Y del mismo modo que se realiza en los accesos, este solamente requiere de introducir dos identificadore,s en este caso eld el puerto y el del buque, si ambos existen en la base de datos, se guardará el atraque correctamente. (Figura 4.13).

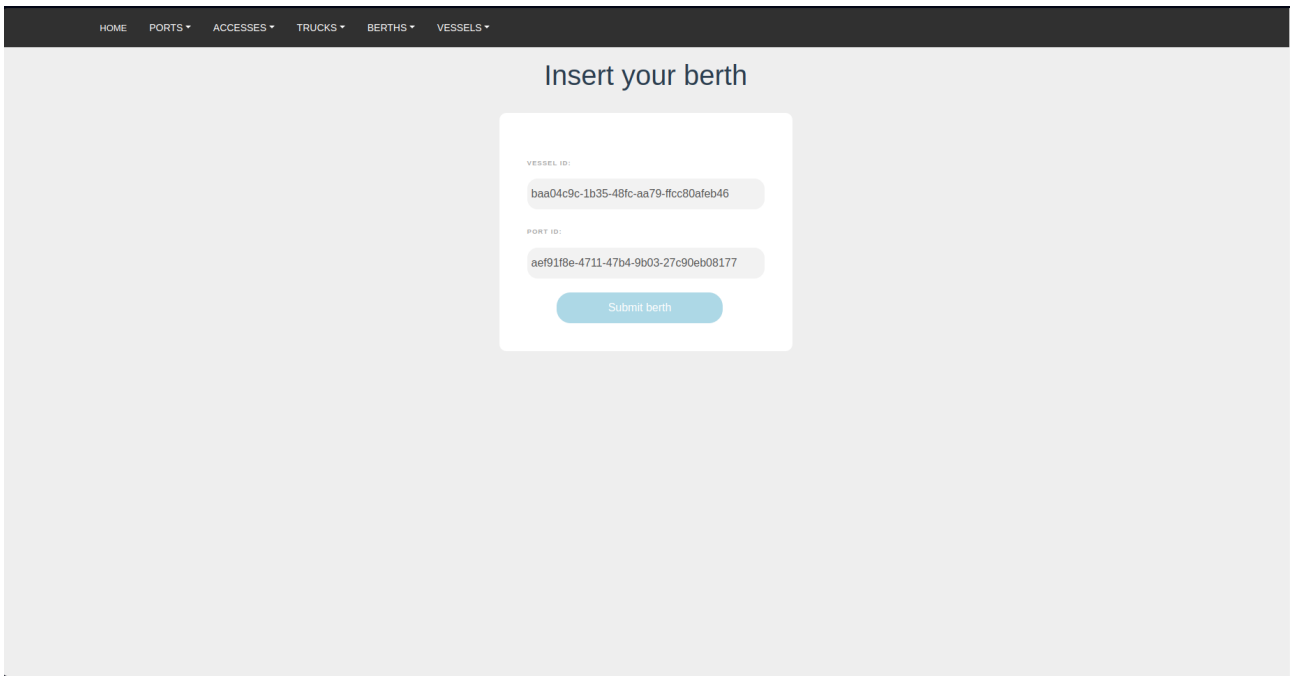


Figura 4.13: Vista del formulario de registro de atraques.

El último registro que posee la aplicación es el propio de los atraques, que como el resto, se puede consultar desde la página principal, en su tarjeta correspondiente *"BERTHS"*, haciendo click en la opción *"EXPLORE"*, o bien desde la barra superior, en el menú desplegable *"Berths"*, seleccionando *"BERTH REGISTER"*. En este caso, desde los atraques se puede acceder a los buques o puertos individualmente, en las vistas previas que ya se han mostrado, pues también son accesibles desde otros registros. Finalmente cuenta con un campo de filtrado y opción de paginado, así como un botón que permite borrar todos los atraques registrados en la base de datos.

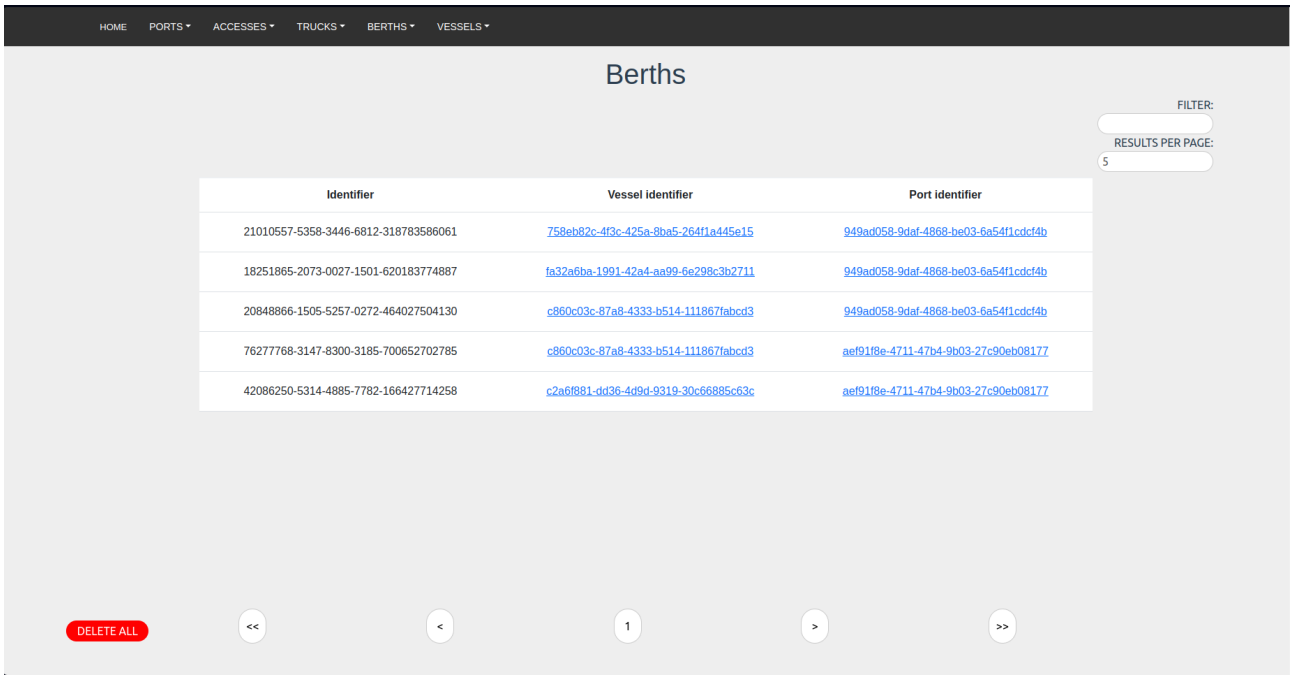


Figura 4.14: Vista del registro de atraques.

## 4.2. Operaciones y resolución

El usuario puede generar diferentes operaciones en función de la vista en la que se encuentre, estas se pueden dividir en:

- **Inserción de datos:** El usuario puede añadir información a la base de datos por medio de las vistas que contienen formularios sobre las diferentes entidades. Para esto se hace uso de los métodos GET que están presente en los adaptadores del API REST del back-end, el cuál a su vez, haciendo uso de los adaptadores de la base de datos, insertará la información en esta. En este caso se pueden producir diversos casos.
  - ***"Success":*** Si el mensaje retornado es de éxito, significa que la petición ha funcionado correctamente, y que los datos han sido insertados en la base de datos.
  - ***"Error 404":*** Hay múltiples casos, por ejemplo en los que se requiere insertar datos que previamente existen en la base de datos, como pueden ser los camiones, puertos o buques en el caso de insertar accesos o atraques, si los identificadores de estos datos no se encuentran en la base de datos retornará este error e informará al usuario de que no se encuentra dicho identificador en al base de datos.
  - ***"Error 500":*** Este tipo de errores significan error interno del servidor, puede deberse a que el servidor no esté operativo, o haya ocurrido un problema inesperado, así que es usado para tratar los errores por defecto que no se hubiesen incluido.
- **Recuperación de información:**
  - ***"Success":*** Si el mensaje retornado es de éxito, significa que la petición ha funcionado correctamente, y que los datos han sido recuperados correctamente, normalmente viene acompañado de un código 200.
  - ***"Error 404":*** Como en el caso de la inserción, para comodidad del usuario, este solo requiere introducir el identificador de una entidad en lugar de todos sus atributos a la hora de ingresar accesos, puertos o buques. Para que esto pueda darse, es necesario recuperar toda la información de dicha entidad. Si este dato no se encontrara en la base de datos, devolvería un código de error 404, indicando que no se ha encontrado.
  - ***"Error 500":*** Se incluye para el resto de errores por defecto que pudieran ocasionarse, manejando este código de error se puede lanzar una notificación al usuario de forma que ante el error sepa como actuar.
- **Actualización de datos:**
  - ***"Success":*** Si los datos a modificar son correctos, y no se presenta ningún fallo durante el proceso de modificación, se lanzará una notificación al usuario avisándole de que los datos han sido modificados correctamente.
  - ***"Bad Request":*** Si el usuario no inserta datos para actualizar los campos requeridos, o estos no son aptos, se le notificará que revise el formato del campo incorrecto, ya que para campos omo las fechas, las matrículas o el tipo

de operaciones que se realizan, las inserciones deben seguir ciertos formatos, y de no cumplirse la información a actualizar no es válida.

- ***"Error 404":*** Si el usuario desea modificar una entidad la cuál no se encuentra en la base de datos, el código de error retornado será este, pues no puede modificar dichos datos.
- ***"Error 500":*** En caso de que algún error no manejable pudiese ocurrir, se lanza este erro al usuario, que por medio de una notificación se le informa del error pertinente sin ocasionar fallos.

- **Borrado de información:**
  - ***"Success":*** Si el conjunto de elementos a eliminar es apto, y el proceso es llevado a cabo correctamente, se le informa al usuario que los datos seleccionados han sido eliminados.
  - ***"Error 404":*** Si el usuario desea eliminar datos que no se encuentran en la base de datos, es informado que no puede llevar dicha operación a cabo.
  - ***"Error 500":*** EN caso de que pudieran suceder errores no manejados se utilizará este código por defecto.

# Capítulo 5

# Testing y despliegue

## 5.1. Testing

Para comprobar que la aplicación funciona correctamente se han llevado a cabo diferentes pruebas para comprobar el correcto funcionamiento de los componentes y las conexiones de estos entre sí.

Así por ejemplo se hacen pruebas de inserción de datos manuales en la base de datos cuando esta se crea para comprobar si el uso de claves foráneas, primarias y restricciones en el borrado o actualizado es correcto.

Para llevar a cabo este proceso de hace uso de PSQL, como se puede ver en la Figura 5.1, herramienta que proporciona una interfaz de línea de comandos que permite a los usuarios interactuar con bases de datos PostgreSQL, llevando así a cabo operaciones de inserción, actualización, borrado y comprobación de los datos.

```
jorge@jorge-acer:~/Escritorio/TFG/truck-planner/data-loader$ psql -h localhost -p 5432 --username postgres
Password for user postgres:
postgres=# \c truckplanner
You are now connected to database "truckplanner" as user "postgres".
truckplanner=# \d
                  List of relations
 Schema |             Name              | Type  |  Owner
--------+-------------------------------+-------+----------
 public | operation                     | table | postgres
 public | port                          | table | postgres
 public | scale                         | table | postgres
 public | scale_do_operation            | table | postgres
 public | truck                         | table | postgres
 public | truck_access_port             | table | postgres
 public | vessel                        | table | postgres
 public | vessel_berth_in_port_with_scale | table | postgres
 public | vessel_do_scale               | table | postgres
(9 rows)

truckplanner=# select * from truck;
                  id                  |  plate  |  brand   |      driver       | is_unload
--------------------------------------+---------+----------+-------------------+-----------
 c057252a-ca69-4486-a526-9cf40d88b337 | 4182XJD | V        | UM YRPUFK         | f
 8b51baa3-69d4-4648-9030-ef355724d9eb | 4252IFK | VZ       | NWT V             | t
 cac591d4-4df7-4e16-99ce-ec86e4f13331 | 2520FDY | BRRJ     | ZHLPM CLQ         | f
 b3600281-611c-4213-901c-8b86c82e9a7a | 3084LXC | JRUZ     | MFQM LUNIZCCG     | f
 ff50e48d-1705-40e5-a151-1000fa5e7816 | 5792TQX | RERZ     | QPYUQE DFNFMVVXYO | f
 349f0806-3445-40e9-8dcd-d52c1a42c4ac | 3058OOX | IWLWARMH | EIGZVL XDZW       | t
 331d17f3-46b6-4d5e-8239-bfc153833e61 | 7835IAG | UXKWC    | X FUP             | t
 0a6bd829-7f20-480b-8129-b365a7e62cba | 1317GCB | QFSIERJE | DX TNUGKWDHKF     | t
 8e2c8204-d3dc-4b0b-a272-9611360442e1 | 9706VPL | VVOTCB   | P LWBTHDR         | t
 729ff4b4-009c-497a-8af0-4447baf67d07 | 1939XRK | PQUTM    | TXFTCAVD CCVVME   | t
 77590762-c63f-4076-88ce-a796bd9fc9a4 | 5729NUZ | Z        | OWB UERJLE        | f
(11 rows)
```

Figura 5.1: Caso de uso de la herramienta PSQL

A su vez, para comprobar el correcto funcionamiento del back-end se lleva a cabo

testing sobre el las diferentes entidades, respecto a la validación de datos y de métodos que son necesarios en el resto de componentes de la aplicación. En este caso se llevan a cabo con el mismo desarrollo del back-end, siguiendo una metodología de pruebas unitarias, y permitiendo así comprobar individualmente cada aspecto de estas. Estas son ejecutadas por Maven (Figura 5.2) una vez el código es compilado, de esta forma, si se implementa una entidad, se comprueba su correcto funcionamiento, y posteriormente esta falla debido a un cambio que se ha llevado a cabo en el código, se notificará que ciertas funcionalidades se han perdido y que se requiere de una adaptación del cambio que se quiere llevar a cabo.

```
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 33, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:3.2.2:jar (default-jar) @ turnaround ---
[INFO] Building jar: /home/jorge/Escritorio/TFG/truck-planner/back-end/target/turnaround-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:2.7.3:repackage (repackage) @ turnaround ---
[INFO] Replacing main artifact with repackaged archive
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  32.521 s
[INFO] Finished at: 2023-07-04T11:19:01+01:00
[INFO] ------------------------------------------------------------------------
```

Figura 5.2: Ejemplo del testing al compilar el código del back-end

Por otra parte, para comprobar el correcto funcionamiento del API REST y los adaptadores del back-end que interaccionan con esta, se llevó a cabo un programa de manipulación masiva de datos, permitiendo realizar múltiples operaciones de una sola vez. Llevándolo a un caso práctico, esta aplicación permite insertar un número determinado de camiones, posteriormente recupera dichos camiones para mostrarlos, y es capaz tanto de actualizarlos como de borrarlos. En la práctica, esta aplicación permitió asegurar el correcto funcionamiento de estas tres componentes y las conexiones entre sí, ya que permitía ejecutar múltiples operaciones en un instante. Para llevar a cabo esta tarea se ha hecho uso de CURL, lo que permitía llevar a cabo operaciones de forma individual o masiva. En el caso de las individuales, estas requieren de un mayor tiempo y aumentan la posibilidad de dejar errores en el código, por lo que se optó por declinar su uso. En cambio, haciendo uso de CURL desde un script (como se puede ver en la figura **??**) si se pueden llevar a cabo operaciones masivas, y al comprobar todos los posibles casos se puede asegurar el perfecto funcionamiento de estas componentes.


Figura 5.3: Ejemplo de uso del script de manipulación masiva de datos

Respecto al front-end y su correcto funcionamiento se han probado, manualmente todas las posibles operaciones que se pueden llevar a cabo, comprobando que la respuesta retornada a la propia interfaz es correcta.

## 5.2. Despliege

El software puede ser implementado en un entorno local utilizando Docker[1]. Esta es una plataforma de código abierto que facilita la ejecución de aplicaciones en contenedores. Estos contenedores contienen todas las dependencias necesarias para asegurar que, sin importar en qué máquina se ejecuten, la configuración sea consistente. El proceso de creación de un contenedor implica lo siguiente:

- Se crea un Dockerfile en el cuál se especifican las dependencias y la configuración del proyecto.
- Se crea una imagen Docker que contiene tanto el código fuente como las librerías.
- Se crea el contenedor dónde se está ejecutando la aplicación.

Este contenedor permite crear un entorno allá dónde se ejecute el archivo Dockerfile, es por ello que es tan ampliamente usado, por su capacidad para desplegar aplicaciones sin apenas complicaciones.

En este caso, el proyecto consta de tres contenedores, uno para tanto el front-end como el back-end, y otro para la base de datos. Para poder ejecutar estos tres contenedores simultáneamente se debe especificar en un fichero denominado Docker Compose.

Para que las imágenes sean eficaces tienen que actualizarse a medida que se realicen cambios en la aplicación, y llevar esto a cabo manualmente puede resultar en una tarea

[1]Docker: https://www.docker.com

tediosa. Para ello este proceso se ha automatizado con la ayuda de Github Actions[2], ya que esta plataforma, al registrar los cambios haciendo uso de git es ideal para llevar esta tarea. De esta forma github se vuelve una herramienta vital en el proyecto, no solo por su faceta de gestr de versiones, sino también por su labor de compilación, pruebas y despliegue dentro del mismo.

Todo este proceso sigue un flujo de trabajo que se puede ver reflejado en la figura 5.4.

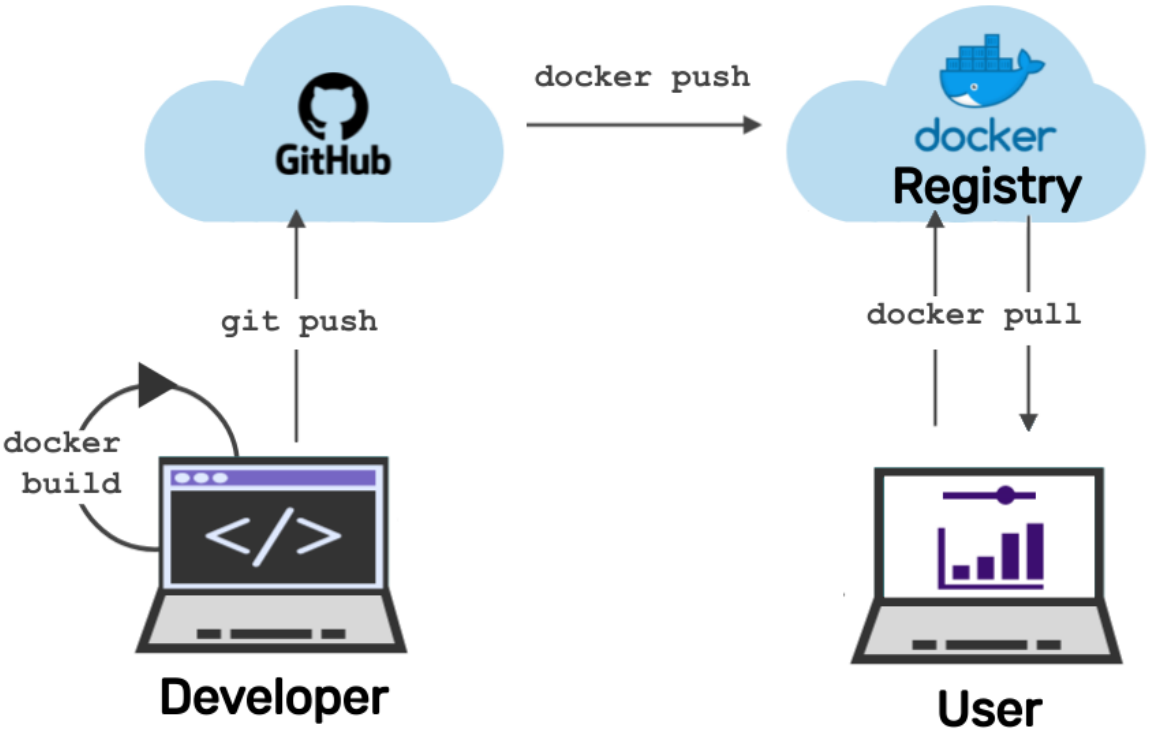


Figura 5.4: Diagrama de flujo de trabajo entre docker y la aplicación.

El flujo de trabajo es el siguiente:

- El flujo se inicia cuando se empujan los cambios confirmados al repositorio.
- En función de dónde se hayan detectado los cambios se crean las diferentes imágenes, o bien para el back-end o para el front-end.
- Estas se publican en Docker Hub[3], un repositorio público para distribuir imágenes Docker. Finalmente, desde el fichero de configuración de Docker Compose, se descargan estas imágenes (back-end, front-end o timescaledb), para poder ejecutarse conjuntamente.

[2]GitHub Actions: https://github.com/features/actions
[3]Docker Hub: https://hub.docker.com

# Capítulo 6

# Presupuesto

El presupuesto para lo implementado en este Trabajo de Fin de Grado ha sido calculado dividiendo el trabajo en conceptos, especificando las horas empleadas para cada uno de ellos y un precio por hora fijo. El resultado del mismo puede ser consultado en la siguiente tabla 6.1.

| **Concepto** | **Coste por hora** | **Horas** | **Coste total** |
|---|---|---|---|
| Estudio de las tecnologías | 10€ | 10 | 100€ |
| Creación de la estructura del proyecto | 20€ | 20 | 400€ |
| Desarrollo del back-end | 20€ | 50 | 1000€ |
| Creación de la base de datos | 20€ | 30 | 600€ |
| Creación de la API REST | 20€ | 40 | 800€ |
| Desarrollo del front-end | 20€ | 50 | 1000€ |
| Conexión entre los componentes | 20€ | 40 | 800€ |
| Mejoras y revisión de la aplicación | 20€ | 20 | 400€ |
| Redacción de la memoria | 15€ | 60 | 900€ |
| | *Total* | *320* | *6000€* |

Tabla 6.1: Presupuesto del trabajo realizado.

# Capítulo 7

# Conclusiones y líneas futuras

Como se ha podido apreciar en este Trabajo de Fin de Grado, cada día encontramos aplicaciones especializadas para cada ámbito concreto de nuestra vida. En este caso, se proporciona una oportunidad para los involucrados en el tráfico de mercancías en instancias portuarias, ahorrando tiempo, costes, y facilitando el trabajo de los involucrados. Los aspectos que más dificultades han presentado se pueden concretar en el uso de la arquitectura hexagonal, que, al ser desconocida me ha supuesto numerosos cambios a lo largo del proyecto para que este se adaptara a ella, además de la conexión entre las componentes y el correcto funcionamiento final desde el punto de vista del usuario, ya que todas las conexiones debían funcionar sin fallo y a pesar de que algunas conexiones no suponían una mayor dificultad, el tratamiento de peticiones entre API y tanto front-end como back-end, al tenerse que llevar de forma asíncrona me supuso también algunos problemas que se han conseguido solucionar. Otro aspecto a destacar es el uso de identificadores para el tratamiento de datos a nivel interno, que si bien funciona correctamente, este no es un dato a introducir por el usuario, y sin embargo debe introducirlo a la hora de actualizar o borrar información, esto nos lleva a plantear las siguientes líneas futuras:

- Modificación de identificadores en campos no necesarios como los camiones, que bien podrían identificarse exclusivamente con otros atributos como la matrícula y el conductor.
- Integrar este proyecto con otro que permita hacer un seguimiento de barcos o tráfico, permitiendo a los usuarios acceder a toda esta información desde un solo punto.
- Mejoras en la interfaz de usuario, requerir de un registro e inicio de sesión, dónde se permitan guardar datos para los usuarios, y de esta forma hacer consultas de forma más rápida en la aplicación.
- De mano con el registro de usuarios, permitir tener usuarios administradores y uso de permisos a la hora de actualizar o borrar información, de forma que no se pierdan datos relevantes o a los que no deberían poder acceder ciertos usuarios.

# Capítulo 8

# Summary and Conclusions

As has been seen in this Final Degree Project, every day we find specialized applications for each specific area of our lives. In this case, it provides an opportunity for those involved in the traffic of goods in port instances, saving time, costs, and facilitating the work of those involved. The aspects that have presented more difficulties can be specified in the use of hexagonal architecture, which, being unknown to me has meant numerous changes throughout the project for it to adapt to it, in addition to the connection between the components and the correct final operation from the user's point of view, since all connections should work without failure and although some connections did not pose a major difficulty, the processing of requests between API and both front-end and back-end, having to be carried asynchronously also meant some problems that I have managed to solve. Another aspect to highlight is the use of identifiers for data processing internally, which although it works correctly, this is not a data to be entered by the user, and yet it must be entered when updating or deleting information, this leads us to consider the following future lines:

- Modification of identifiers in fields which are not necessary, such as trucks, which could be identified exclusively with other attributes such as plates and driver's name.
- Integrate this project with another project to track ships or traffic, allowing users to access all this information from a single app.
- Improvements in the user interface, requiring a registration and login, where data can be saved for users, and thus make faster queries in the application.
- Hand in hand with user registration, allow for administrator users and use of permissions when updating or deleting information, so that relevant data or data that should not be accessible to certain users is not lost.

# Bibliografía

[1] Gobierno de España. Observatorio de mercado del transporte de mercancías por carreteras. *Ministerio de Transportes, Movilidad y Agencia Urbana*, 31:5–5, 2022.

[2] Fernando Morales Rodríguez. Ever given: un año del gran atasco del canal de suez. *Las provincias*, 2022.

[3] R. Pell T. L. Saaty, P. Rogers. Portfolio selection through hierarchies. *Journal of Portafolio Management*, 6(3):16–21, 1988.