ULPGC
Universidad de
Las Palmas de
Gran Canaria

Escuela de
Ingeniería Informática

Infraestructura web para el seguimiento, monitorización y control de misiones para vehículos marinos autónomos a vela

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: Jon Lander Torres Lombraña

TUTORIZADO POR:
Antonio Carlos Domínguez Brito
Jorge Cabrera Gámez

Octubre 2021

Resumen

En el presente Trabajo Final de Grado se plantea el desarrollo y diseño de una infraestructura web para la visualización de misiones de navegación de los distintos vehículos de la División de Robótica y Oceanografía Computacional del Instituto Universitario SIANI [1]. Con esta solución se persigue el objetivo de que usuarios externos a la entidad puedan consultar la información sobre las distintas misiones. Asimismo, la División podrá gestionar distintos aspectos relacionados con las misiones, tales como: editar la información, incorporar nuevos datos y misiones, asignar vehículos a misiones y viceversa, crear nuevas misiones, crear nuevos vehículos, entre otras. Debe permitir seguir las misiones que se estén llevando en curso, así como reproducir y analizar las misiones almacenadas ya realizadas.

Abstract

In this Final Degree Project, the development and design of a web infrastructure for the visualization of navigation missions of the different vehicles of the Division of Robotics and Computational Oceanography of the SIANI University Institute [1] is proposed. This solution pursues the objective that users external to the entity can consult the information on the different missions. Likewise, the Division will be able to manage different aspects related to missions, such as: editing the information, incorporating new data and missions, assigning vehicles to missions and vice versa, creating new missions, creating new vehicles, among others. It must allow to follow the missions that are being carried out, as well as reproduce and analyze the stored missions already carried out.

Índice.

Índice de figuras.

1. Introducción.

Figura 1: Barco A-Tirma navegando.

En las siguientes páginas se presenta al lector un análisis detallado del desarrollo del software para la visualización de datos relacionados con el seguimiento, monitorización y control de misiones para vehículos marinos autónomos.

En primer lugar en los capítulos 2, 3 y 4 se presentan los detalles más académicos del proyecto, que versan sobre los objetivos, la competencias desarrolladas y las aportaciones del mismo. Seguidamente en el capítulo 5 y el 6 se exponen las herramientas empleadas así como los requisitos de software y hardware del trabajo llevado a cabo.

Los detalles acerca de la metodología seguida y el plan de trabajo aplicado se encuentran expuestos en el capítulo 7. En estos apartados el lector podrá tener una información más exhaustiva acerca de la forma en la que las metodologías ágiles se han implementado en el proceso de desarrollo de la aplicación.

Se llega a continuación al capítulo 8 y 9, los capítulos más importantes del presente documento académico: el análisis del problema y diseño e implementación. En estos capítulos se encuentran recogidas las principales barreras a las que la aplicación web pretende dar una solución eficiente y cómo se han implementado las distintas herramientas en el proyecto.

El diseño y la implementación de las distintas herramientas para solventar los problemas expuestos en los capítulos anteriores constituirán el contenido de las conclusiones, capítulo 10, exponiendo algunas consideraciones finales y recomendaciones acerca del uso presente y futuro del software.

Asimismo se incluyen en el apéndice las instrucciones para llevar a cabo el despliegue de la herramienta teniendo en cuenta tres escenarios con servidores con distintas características y especificaciones.

2. Objetivos.

El proyecto que se recoge en este documento tiene como principal objetivo el poner al alcance de las personas ajenas a la organización SIANI [1] información de utilidad con el fin de que puedan seguir el trabajo que esta organización realiza con respecto a los vehículos marinos autónomos de que dispone.

A continuación se detallan específicamente el resto de objetivos de este proyecto:

- Desarrollo, diseño y puesta en marcha de una infraestructura web para el seguimiento y análisis de misiones para el barco autónomo a vela A-Tirma.

- Generalización del diseño para su uso para múltiples vehículos marinos.

Por otro lado desde el punto de vista académico, en relación a los objetivos que se presentan en el proyecto docente de la asignatura "40866 - Trabajo fin de grado" [2] se persiguen los siguientes:

- Elaborar un trabajo en el que el estudiante universitario desarrolle las competencias y los conocimientos adquiridos, teóricos y prácticos como culminación de sus estudios y como preparación para el desempeño futuro de actividades profesionales en el ámbito correspondiente a la titulación obtenida.

- Adquirir conocimiento y habilidades para la elaboración, redacción, presentación y defensa del Trabajo Fin de Grado.

- Concebir, diseñar, planificar e implementar un TFG.

- Adquirir conocimiento de las normas de la ULPGC y de la EII para la realización de un TFG.

3. Competencias.

A continuación se desglosarán las competencias que se han desarrollado durante este proyecto, detallando en que parte del trabajo se han usado cada una de ellas.

- **3.1. CII01.** Capacidad para diseñar, desarrollar, seleccionar y evaluar aplicaciones y sistemas informáticos, asegurando su fiabilidad, seguridad y calidad, conforme a principios éticos y a la legislación y normativa vigente.

  Esta competencia ha sido desarrollada a la hora de diseñar y construir la arquitectura para la creación de la aplicación, podemos ver más información en el capítulo 9 de "Diseño e implantación".

- **3.2. CII02.** Capacidad para planificar, concebir, desplegar y dirigir proyectos, servicios y sistemas informáticos en todos los ámbitos, liderando su puesta en marcha y su mejora continua y valorando su impacto económico y social.

  Esta competencia se ve reflejada a la hora de elegir una metodología de trabajo y planificar dicho trabajo, a su vez el impacto social se ve reflejado en el capítulo 9 de aportaciones.

- **3.3. CII08.** Capacidad para analizar, diseñar, construir y mantener aplicaciones de forma robusta, segura y eficiente, eligiendo el paradigma y los lenguajes de programación más adecuados.

  Esta competencia ha sido desarrollada sobre todo a la hora de pensar y diseñar la aplicación creada, se ve reflejado sobre todo en el capítulo 8 y 9 de "Análisis del problema" y "Diseño e implantación".

- **3.4. CII012.** Conocimiento y aplicación de las características, funcionalidades y estructura de las bases de datos, que permitan su adecuado uso, y el diseño y el análisis e implementación de aplicaciones basadas en ellos.

  Esta competencia se vio reflejada sobre todo a la hora de diseñar la base de datos. Se puede ver más en detalle en el capítulo 9 "Diseño e implantación".

- **3.5. CII013.** Conocimiento y aplicación de las herramientas necesarias para el almacenamiento, procesamiento y acceso a los Sistemas de información, incluidos los basados en web.

  Esta competencia se vio reflejada a la hora de realizar todo el manejo de información a través de controladores en la aplicación, se puede ver más detalladamente en el capítulo 9 de "Diseño e implantación".

- **3.6. CII017.** Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas.

  Esta competencia ha sido muy importante a la hora del desarrollo del proyecto, se ve claramente reflejada en la interfaz de usuario elaborada para la aplicación. Se desarrolla en el capítulo 9 de "Diseño e implantación".

4. Aportaciones.

Las principales aportaciones que este proyecto aporta a la División de Robótica y Oceanografía Computacional de SIANI [1] son las siguientes:

- Infraestructura web con diseño escalable basada en herramientas de uso público.

- Posibilidad no sólo de hacer accesibles las actividades de los vehículos autónomos marinos de la división ROC del SIANI, sino de constituir una herramienta de almacenamiento a largo plazo y análisis de misiones a posteriori para las investigaciones de la división.

- Acercar la actividad de la división ROC del SIANI a los usuarios externos, proporcionando una interfaz donde podrán seguir con facilidad desde cualquier dispositivo todas las misiones que se realicen.

- Aplicación web con un diseño que se adapte a cualquier dispositivo y con ello se pueda incrustar en otras plataformas fácilmente a través de un iframe [3]. Con esto se consigue que si algún organismo desea vincular el mapa con el seguimiento de misiones en su página web, sea posible.

5. Análisis del problema.

Durante la reunión inicial del proyecto, los tutores transmitieron el principal problema u objetivo que el desarrollo debía abordar. El problema se resume de la siguiente manera: La División de Robótica y Oceanografía Computacional del Instituto Universitario SIANI [1] quiere llevar la información que registra con sus vehículos autónomos a cualquier usuario interesado de manera pública a través de una infraestructura web.

Transmitieron que como solución para este problema se buscaba una aplicación web con la capacidad de mostrar los datos relevantes a cualquier usuario que la visitase.

La aplicación desarrollada debía cumplir una serie de requisitos, que se traducen en las siguientes historias de usuario:

- Los visitantes de la aplicación deberán poder seguir misiones pasadas almacenadas en el sistema.
- los visitantes de la aplicación deberán poder ver la ruta que sigue el vehículo en una navegación en directo.
- Los visitantes de la aplicación deberán poder ver la información que el vehículo esta recogiendo sobre el viento.
- La aplicación debe soportar varios vehículos.
- Debe haber perfiles de usuarios con distintos permisos.

A estos requerimientos solicitados por el departamento, se añadieron las siguientes aportaciones realizadas de forma proactiva por Jon Lander Torres:

- La aplicación debe ser fácilmente incrustable en otra página web a través de un iframe [3]. Este elemento permite introducir otra página web en la página que estemos viendo, simplemente sabiendo la dirección de la página que desea introducir.
- La aplicación debía tener una aplicación móvil para que cualquiera pueda descargarla en su teléfono.

Sabiendo estas condiciones da comienzo la búsqueda de soluciones, considerando en primer lugar las distintas opciones de infraestructuras de desarrollo web más óptimas.

En primer lugar, para que la aplicación fuese totalmente compatible con cualquier dispositivo inclusive la inserción con un iframe [3], la aplicación debe ser compatible con cualquier resolución de pantalla.

Por el lado de la aplicación móvil la forma más eficiente de conseguirlo es con una PWA [27] (Progressive Web App), con los únicos requisitos de tener un diseño compatible con cualquier dispositivo y la instalación de un Service Worker [28], que se trata de un intermediario entre la página web y el usuario que controla el uso de caché de la aplicación.

En el frontend era necesario un framework que fuese reactivo, y que no necesitase configuraciones complejas, para que la vista se adaptase a los datos que le llegasen. Esta reactividad en la actualidad se puede conseguir de 3 frameworks distintos todos basados en JavaScript [29], React JS [30], Angular [31] y Vue JS [7]. Siendo el último la opción elegida por su sencillez, su facilidad de uso y la cantidad de documentación disponible para resolver cualquier problema potencial durante el desarrollo.

Por otro lado para el backend se buscaba desarrollar una API sencilla, y después de valorar opciones como usar PHP [32] para crearla, se optó por una librería de Node JS [8] llamada express [15]. Esta librería facilita la creación de APIs haciendo que su desarrollo sea accesible contando con los conocimientos necesarios de JavaScript.

Por último en la base de datos la tecnología elegida fue MySQL [20] por el gran soporte que existe por la comunidad en la red y por que consiste en una base de datos relacional para relacionar los distintos datos con los que trabajaría la aplicación web.

En los capítulos siguientes se detallan las herramientas elegidas y con ellas los requisitos que la aplicación necesita para su funcionamiento.

6. Herramientas.

En este apartado se presentan las principales herramientas empleadas en el proceso de desarrollo. Se exponen a continuación las consideraciones más relevantes acerca del servidor empleado, los framework, las librerías externas utilizadas, así como las herramientas de edición y tratamiento de imágenes.

6.1. Servidor.

- **6.1.1. AWS [4]:** Amazon Web Services es un compendio de servicios de computación en la nube ofrecido por Amazon desde 18 regiones geográficas, entre las que se encuentran Europa (Fráncfort, Irlanda, Londres, París), EEUU Este (Norte de Virginia, Ohio), EEUU Oeste (Norte de California, Oregón), China (Pekín, Ningxia).

  Entre estos servicios que ofrece Amazon se encuentran RDS [5] (Relational Database Service) y EC2 [6] (Elastic Compute Cloud) que son los que se utilizarán en el presente proyecto.

  Antes de adentrarnos en estos dos servicios se debe destacar que Amazon funciona a través de entornos informáticos virtuales, conocidos como instancias. Entre otros atributos, las instancias cuentan con unas configuraciones preestablecidas de CPU, memoria, almacenamiento y capacidad de redes. Las configuraciones tienen unos nombres establecidos para referirse a ellas dependiendo del tipo y el rendimiento que ofrecen siendo *t2* el tipo recomendado para su uso habitual y *micro* una de las de rendimiento más bajo (t2.micro). Estas configuraciones se detallan en la Figura 2.

	Family	Type	vCPUs	Memory (GiB)
☐	t2	t2.nano	1	0.5
☑	t2	t2.micro Free tier eligible	1	1
☐	t2	t2.small	1	2
☐	t2	t2.medium	2	4
⊘	t2	t2.large	2	8
⊘	t2	t2.xlarge	4	16
⊘	t2	t2.2xlarge	8	32

Figura 2: Esquema de distribuciones de Amazon EC2.

Amazon RDS es un servicio que facilita la escalabilidad, configuración y funcionamiento de una base de datos relacional en la nube. Una vez contratado este servicio debe elegir la instancia que quiere utilizar dependiendo del uso que le vaya a dar. Seleccionando para el presente caso una instancia con poco rendimiento, ya que solo se utilizará para probar la aplicación en un entorno real.

El servicio Amazon EC2 brinda un entorno de computación escalable en la nube con el que se elimina la necesidad de contar con hardware a la hora de desarrollar una aplicación. Facilitando en gran medida el despliegue de instancias con los requisitos necesarios de la aplicación.

6.2. Frameworks.

- **6.2.1. Vue JS [7]:** Vue JS es un framework basado en JavaScript [18] diseñado para crear interfaces de usuario de forma incremental. La base de este framework está enfocada en la facilidad de uso y el desarrollo de la capa visual de las aplicaciones. Está preparado para facilitar la integración de librerías externas.

  Una de las características más relevante de Vue JS es la presencia de componentes. Los componentes son estructuras independientes unas de otras las cuales se pueden emplear en cualquier parte de la aplicación. Estos componentes pueden ser tanto simples listas de elementos, como un menú u otras estructuras más complejas. Una vez desarrollado

un componente puede ser utilizado en la parte que se necesite de la aplicación. Para ello únicamente es necesario hacer referencia a él como si de una llamada a una función se tratase.

```
<template>
    <div>

    </div>
</template>

<script>
export default {
    name: 'ComponentName'
}
</script>

<style>

</style>
```

Figura 3: Plantilla básica de componente en Vue JS.

- **6.2.2. Node JS [8]:** Node JS es un entorno de ejecución de JavaScript [18] diseñado para crear aplicaciones network escalables. En este caso se usa para crear el backend de la aplicación, que consiste en una API para la obtención de los datos de los vehículos autónomos a través de unas rutas establecidas.

  Con la instalación de Node JS viene asociado npm [9], su gestor de paquetes, es la herramienta que se utilizará para instalar, actualizar y borrar cualquier librería en este proyecto.

6.3. Librerías externas.

- **6.3.1. axios [10]:** Es una librería desarrollada por Matt Zabriskie que ayuda a la comunicación entre Node JS y el navegador, con la particularidad de que el mismo código base sirve tanto para el navegador como para node.js.

- **6.3.2. vue-leaflet [11]:** Se trata de una librería que se encarga de la integración del frontend con OpenStreetMap [12]. Ofrece funcionalidades como ubicar iconos y dibujar líneas en el mapa entre otros.

- **6.3.3. vue2-leaflet-rotatedmarker [13]:** Es un componente desarrollado por mudin (nombre de usuario de github) que se encarga de posicionar una marca en el mapa que fácilmente se puede rotar en la dirección que se quiera, con esto se consigue rotar los barcos en la dirección de la navegación.

- **6.3.4. stylus [14]:** Consiste en una librería CSS desarrollada por TJ Holowaychuk (nombre de usuario de gitbhub) que ayuda a escribir cómodamente el código relacionado con el estilo de la aplicación web, ademas permite declarar variables, que se pueden emplear en el código posteriormente.

- **6.3.5. express [15]:** Es el framework más utilizado para la creación de una infraestructura de una API en Node JS. Sus puntos fuertes son la sencillez y la flexibilidad. Este framework esta desarrollado por la Fundación OpenJS.

- **6.3.6. cors [16]:** Es un paquete de Node JS que provee de un middleware que se puede utilizar para habilitar CORS con varias opciones. Este paquete esta desarrollado por Troy Goode.

- **6.3.7. bcrypt [17]:** Esta librería que facilita la encriptación de las contraseñas, brindando funciones tanto para encriptar o desencriptar contraseñas. Librería desarrollada por Kelektiv (nombre de usuario de github).

- **6.3.8. jsonwebtoken [18]:** Es una librería que ayuda a encriptar en un token los datos que queramos para asi poder identificar los usuarios con sesión abierta en la aplicación. Desarrollado por Auth0 (nombre de usuario de github).

- **6.3.9. mysql2 [19]:** Es un paquete de Node JS que brinda las funciones necesarias para conectar una aplicación con una base de datos MySQL [20]. Desarrollado por sidorares (nombre de usuario de github).

6.4. Tratamiento de imágenes.

- **6.4.1. Illustrator [21]**: Illustrator es una aplicación para editar gráficos vectoriales del paquete ADOBE [22]. Esta aplicación se ha utilizado durante la realización de este trabajo para el diseño de los vectores correspondientes a la marca de viento en los barcos.

6.5. Software documental.

- **6.5.1. InDesign [23]**: InDesign es una potente herramienta de software del sector del diseño editorial, perteneciente al paquete ADOBE, la cual se posiciona actualmente como el software de referencia dentro de la industria para realizar proyectos sencillos.

  Se ha empleado principalmente para la maquetación de la memoria del presente proyecto.

7. Requisitos.

En este apartado se van a desglosar los requisitos tanto de hardware como de software necesarios para la correcta utilización de la aplicación en las distintas etapas en las que se dividió el proyecto.

7.1. Hardware.

Se han utilizado dos servidores durante el transcurso de este proyecto, uno durante la fase de desarrollo, con el fin de minimizar los posibles problemas al instalar la aplicación en su entorno real, y otro para el despliegue final.

El servidor que se utilizó para el desarrollo fue un servidor basado en AWS [4], utilizando dos de sus servicios más comunes Amazon EC2 [6] para la instancia que portaría la aplicación (1GiB RAM y 1 CPUs) y Amazon RDS [5] para la instancia de base de datos (1GiB RAM y 1 CPUs).

Para su posterior despliegue se utilizó un contenedor de Linux LXC [37] en el servidor del SIANI [1] con unas prestaciones de 2 CPUs y 1GB de RAM.

7.2. Software.

Para la correcta instalación de la aplicación web se debe instalar en primer lugar las siguientes aplicaciones:

Inicialmente tenemos que instalar git [24], un sistema de control de versiones necesario para poder clonar el repositorio donde están alojados los proyectos.

En segundo lugar se debe tener instalado Node JS [8], dado que el backend de esta aplicación funciona con esta tecnología.

En tercer lugar Vue JS [7], todo el front end esta desarrollado con este framework.

Por último es necesario contar con una base de datos MySQL [20], necesaria para almacenar, visualizar y modificar los datos de los vehículos autónomos, rutas, condiciones climáticas, misiones y usuarios, entre otros.

8. Metodología y plan de trabajo.

La metodología ágil [25] para el desarrollo de software consiste en un enfoque que afecta a la toma de decisiones de los proyectos de desarrollo. Esta se basa en el desarrollo iterativo e incremental, es decir, los requisitos y soluciones evolucionan en función de las necesidades del proyecto.

Se trata de una metodología que aporta entre otras ventajas la reducción de tiempos en la toma de decisiones y una mayor flexibilidad en la gestión de proyectos.

En el presente proyecto, la metodología utilizada se ha basado en la realización de reuniones periódicas y sprints divididos según las distintas fases del proyecto.

Los sprints son segmentos de tiempo en los que se realiza una serie de tareas. Antes de cada sprint y al finalizar los mismo se ha realizado una reunión con dos objetivos principales: En primer lugar revisar el avance del desarrollo y posibles problemas que hayan podido surgir. En segundo lugar definir las tareas del siguiente sprint.

Las fases que ha seguido la metodología de trabajo han sido Análisis, Desarrollo e implantación y Prueba final. Todas y cada una de estas fases son necesarias para el correcto desarrollo de la aplicación.

Tal como se ha mencionado anteriormente, al finalizar cada uno de los sprints se ha realizado una reunión para verificar y evaluar el resultado de dicho sprint y a su vez establecer objetivos para el siguiente.

Al iniciar el proyecto en primer lugar se planteó una fase de análisis, en la que se investigaron las distintas tecnologías del mercado al igual que se definieron las necesidades que debía cubrir la aplicación.

Posteriormente se comenzó con las tareas de desarrollo.

8.1. Fase inicial - Análisis.

Antes de comenzar el desarrollo de la plataforma digital se ha llevado a cabo una fase de análisis previa. El objetivo de esta fase ha sido detectar las necesidades que la plataforma digital debía solucionar.

Esta información se ha obtenido de forma cualitativa mediante una entrevista personal con los profesores Antonio Carlos Domínguez Brito y Jorge Cabrera Gámez, responsables del proyecto y tutores de este trabajo. En estas entrevistas se han expresado las necesidades de la división, así como la información disponible para el desarrollo de la plataforma.

8.1.1. Sprint 1.

En base a dicha información se ha establecido el primer sprint, cuyo objetivo era investigar sobre las tecnologías a utilizar, establecer un primer esquema de la arquitectura de la solución y crear el boceto inicial para el diseño de la plataforma.

Como resultado de dicha investigación, se decidió que las tecnologías a utilizar serían Vue JS [7] en el front-end y Node JS [8] en el back-end, sobre una base de datos MySQL [20].

No se presentaron problemas ni incidencias a lo largo de este sprint.

8.2. Fase 2 - Desarrollo, prueba e implantación.

8.2.1. Sprint 2.

Una vez llevado a cabo el análisis inicial y definida la arquitectura de la solución se inicia el segundo sprint del proyecto en el que se persiguen los siguientes objetivos pertenecientes a la fase de desarrollo del mapa donde se podrá:

- Visualizar los distintos vehículos.
- Buscar vehículos y misiones.
- Reproducir las misiones.
- Acceder a una barra de herramientas.
- Visualizar los datos de los vehículos en su última interacción.
- Entrar al panel de acceso de usuarios.

En la Figura 4 se puede observar como se reproduce una misión en la aplicación.

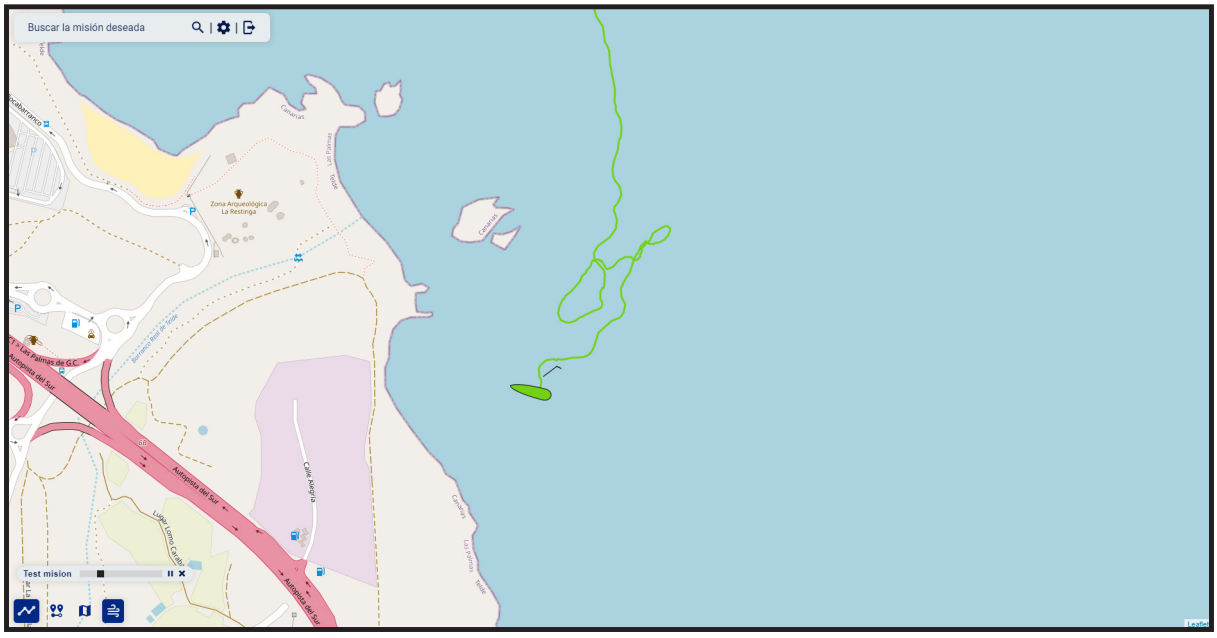


Figura 4: Vista del mapa de la aplicación web.

8.2.2. Sprint 3.

En el sprint 3, se ha construido el panel para el control de la información, donde se puede:

- Crear y administrar usuarios.
- Crear y administrar misiones, en la Figura 5 podemos observar un ejemplo de la administración de una misión.
- Crear y administrar vehículos.
- Crear y administrar los permisos de los distintos roles de usuarios.

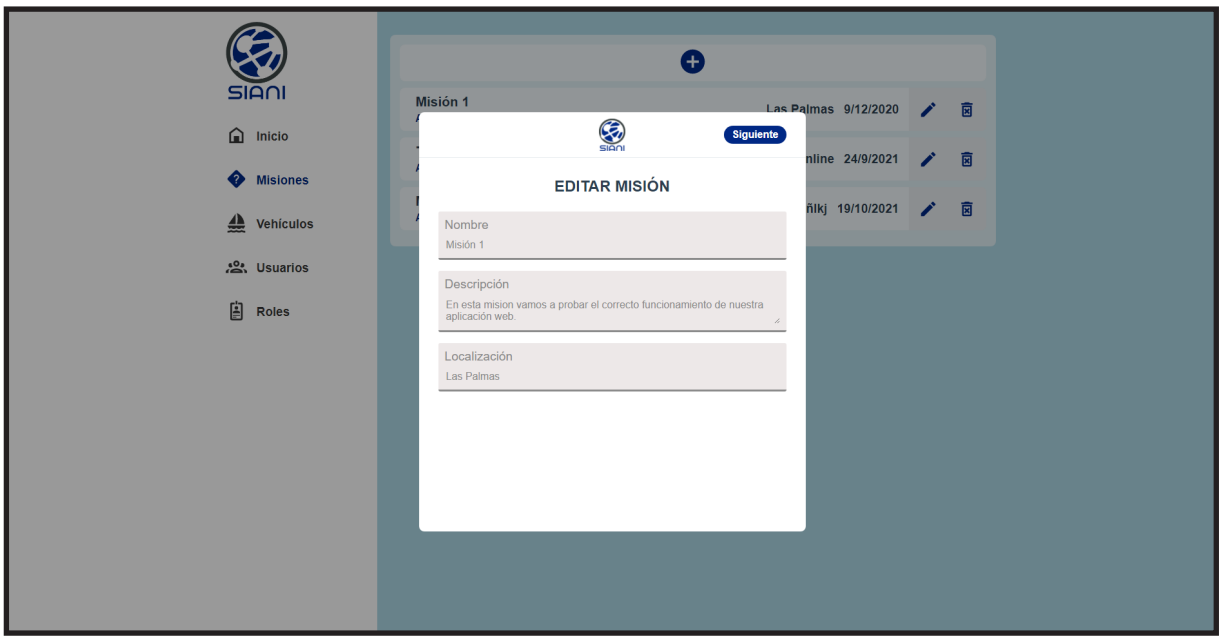


Figura 5: Editar misión en el panel de administrador.

8.2.3. Sprint 4.

Los puntos realizados en este sprint corresponden a las herramientas de la parte inferior izquierda las cuales te permiten realizar acciones como ver información del viento que está afectando al barco o ver hacia qué coordenadas se dirige el barco en cada instante.

- Función de mostrar ruta seguida.
- Función de mostrar siguiente punto.
- Función de mostrar todos los puntos por donde va a pasar el vehículo.
- Función de mostrar los datos del viento.

8.2.4. Sprint 5.

A lo largo de este sprint, correspondiente al último sprint de la fase de desarrollo, prueba e implantación, se han abordado las siguientes tareas:

Insertar datos referentes a un vehículo en una misión en la base de datos MySQL [20] a través de un fichero de extensión .csv .

Implantación de un sistema de notificaciones que permita a los usuarios conocer si las acciones realizadas en la aplicación se han realizado correctamente o si por el contrario presentan algúna incidencia.

Ejemplo de ello son las notificaciones que se muestran al crear, editar, borrar misiones, usuarios, roles o vehículos, entre otros.

8.3. Fase 3 - Prueba final.

A lo largo del desarrollo se han realizado las comprobaciones pertinentes para conocer que las tareas desarrolladas en los distintos sprints fueron resueltas de forma exitosa.

Una vez desarrollado el software en su totalidad se procedió a su instalación en los servidores proporcionados por SIANI [1] para comprobar el correcto funcionamiento conjunto de las distintas funcionalidades desarrolladas, también con éxito.

9. Diseño e implementación.

La solución ha sido una aplicación basada en JavaScript, la cual permite por su reactividad, eliminar del proceso de desarrollo todo el código destinado a renovar variables que se están mostrando.

En el Frontend se ha elegido Vue JS [7], este framework permite generar una aplicación de una sola página. Esto significa que la solución web solo va a tener tiempo de carga la primera vez que un usuario accede a ella, el resto de movimientos que el usuario realice dentro de la página web cambiará la información que se muestra pero no tomará tiempo de carga.

Esta aplicación de una sola página a su vez permite separar en componentes todo lo necesario para que la aplicación funcione, y estos componentes reaccionan a unos datos que le llegan, como por ejemplo los vehículos que se están siguiendo ahora mismo. Si estos vehículos cambian a otros, los componentes reaccionan automáticamente y mostrarían los nuevos vehículos.

También en el frontend hemos instalado un compilador de CSS [33] para que el estilo sea más fácil de entender y desarrollar. Este compilador se llama Stylus [14], que entre otras acciones permite declarar unas variables como los colores principales de la aplicación.

No se debe olvidar uno de los elementos más importantes de la aplicación, el mapa. Para la representación vía OpenStreetMaps [12] del mapa y pintar los barcos en las posiciones del mapa, se ha utilizado vue2-leaflet [11]. Esta herramienta brinda una gran variedad de funcionalidades con respecto al uso del mapa, como dibujar en él de una forma sencilla e intuitiva.

En el backend se ha elegido Node JS [8], utilizando un framework llamado express [15], tal como se ha mencionado en apartados anteriores, el cual tiene como primera característica la minimización de sus componentes, por lo que hace que sea una herramienta que facilita su aprendizaje y posterior uso e implementación como solución eficiente en el desarrollo.

También en el backend como conexión a la base de datos se utiliza la librería mysql2 [19]. Esta librería ofrece unas funciones y estructura para conectarnos con facilidad desde Node JS a una base de datos MySQL [20] que como se ha mencionado previamente es la base de datos por la que se ha optado en este proyecto.

A continuación se desarrollará en detalle todo lo descrito en este apartado.

9.1. Base de datos

Para el desarrollo de la base de datos, como se ha mencionado anteriormente se elige MySQL [20]. La estructura utilizada para la creación de esta base de datos está desglosada en la Figura 6. A continuación se va a desglosar el contenido y propósito de cada tabla de la base de datos.

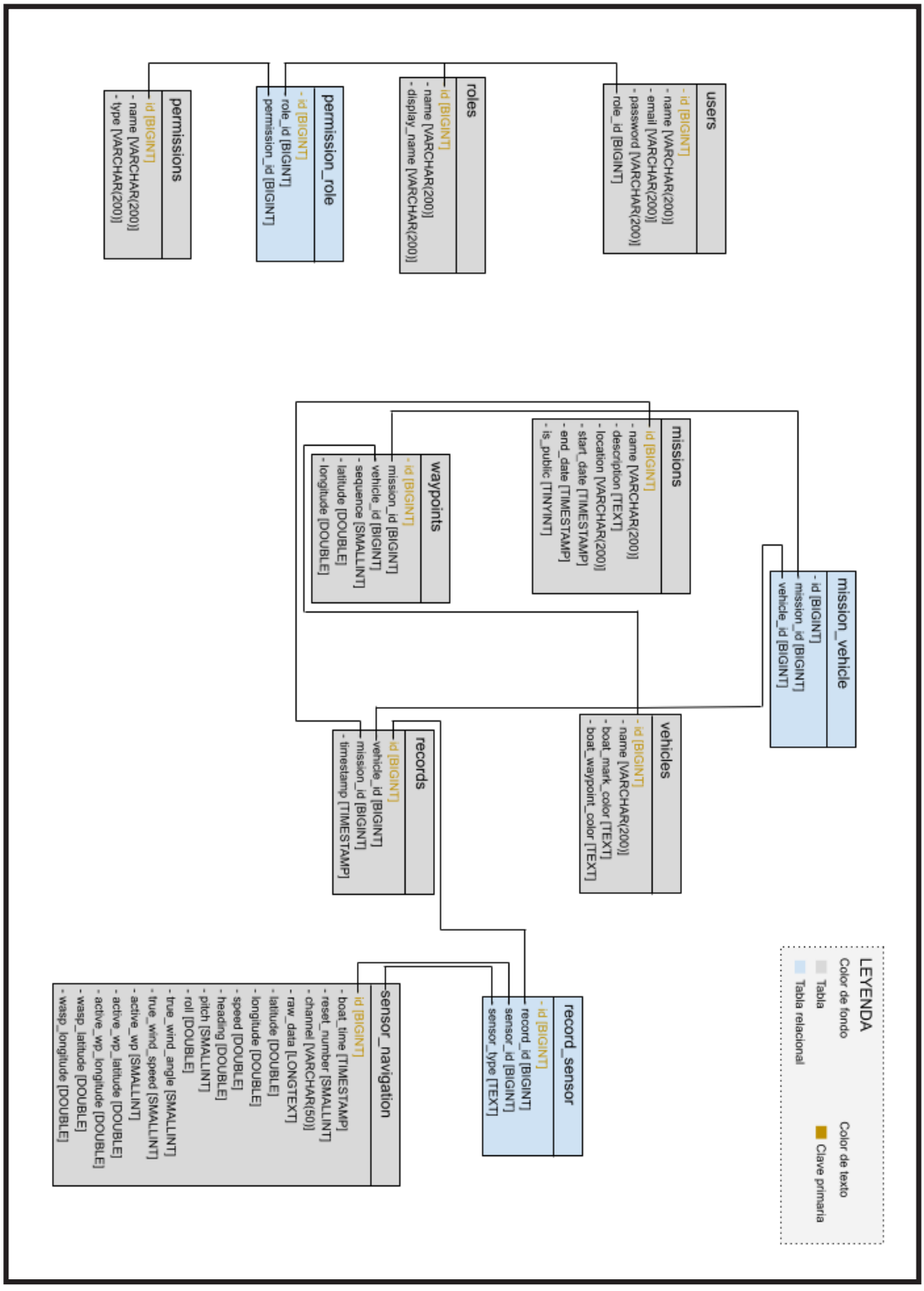


Figura 6: Esquema de la base de datos.

Figura 7: Tabla missions.

9.1.1. Tabla missions

Esta tabla está destinada a contener toda la información relativa a las misiones, como su nombre, descripción, localización, fecha de inicio y fin.

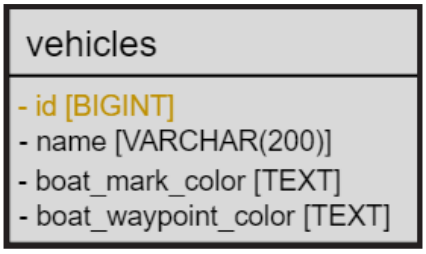


Figura 8: Tabla vehicles.

9.1.2. Tabla vehicles

Esta tabla está destinada a contener toda la información relativa a los vehículos, como su nombre, color del icono, y color de sus waypoints.

En una primera instancia los vehículos son barcos, pero podrían también ser vehículos terrestres.

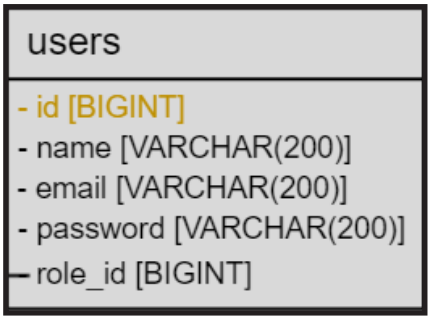


Figura 9: Tabla users.

9.1.3. Tabla users

Esta tabla está destinada a contener toda la información relativa a los usuarios, como su nombre, email, contraseña y rol.

Usuario se considera a cualquier persona con la que se deba tener un trato especial en la aplicación, puede ser un cliente, un administrador, encargado de mantenimiento, estas funciones se controlan con el rol.

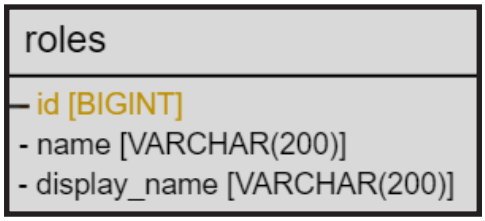


Figura 10: Tabla roles.

9.1.4. Tabla roles

Esta tabla está destinada a contener toda la información relativa a los roles. Sólo se requiere de un nombre para identificar al rol.

Los roles sirven para delimitar las funciones que tiene un usuario, por ejemplo, el rol de administrador tiene el máximo nivel de permisos en la aplicación pero el rol de cliente no vé el panel de administrador.

9.1.5. Tabla permissions

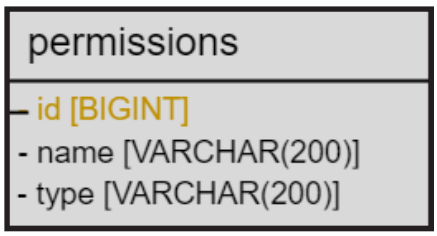


Figura 11: Tabla permissions.

Esta tabla está destinada a contener toda la información relativa a los los permisos de usuarios, como nombre y tipo que sirve para categorizar.

Los permisos habilitan a hacer acciones dentro de la aplicación, por ejemplo el permiso de "show_mission" permite ver las misiones en el panel de administrador.

9.1.6. Tabla records

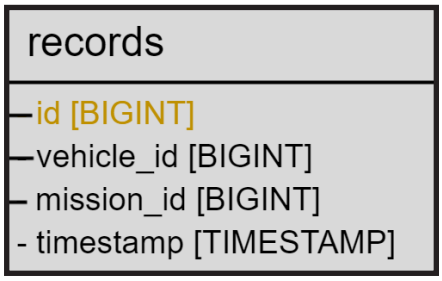


Figura 12: Tabla records.

Esta tabla está destinada a contener toda la información general relativa a las tomas de datos del vehículo en una misión, como fecha de la toma.

9.1.7. Tabla waypoints

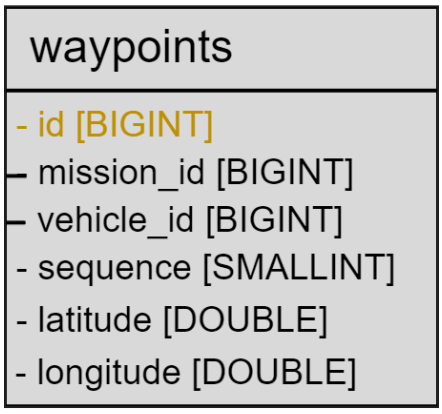


Figura 13: Tabla waypoints.

Esta tabla está destinada a contener toda la información relativa a los waypoints de un vehículo en una misión, como su latitud y longitud y el orden de los mismos.

Los waypoints sirven para saber la ruta por donde tiene que ir el barco, y así poder mostrar esa información.

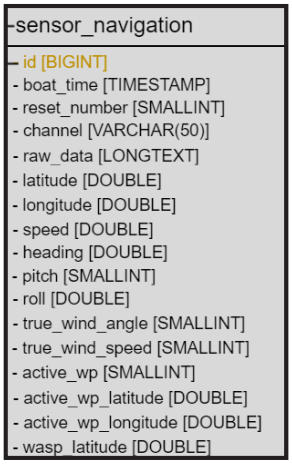
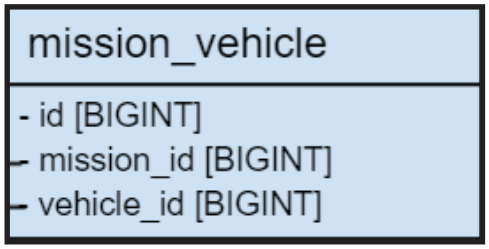


Figura 14: Tabla sensor_navigation.

9.1.8. Tabla sensor_navigation

Esta tabla está destinada a contener toda la información relativa al sensor de navegación que lleva el vehículo, como el posicionamiento del mismo, velocidad, velocidad del viento, o hacia donde se dirige.

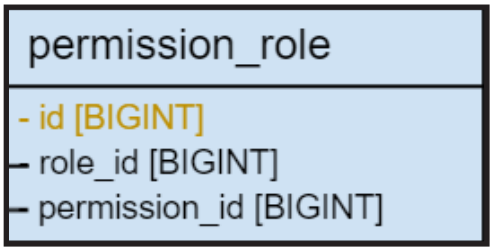
9.1.9. Tabla relacional mission_vehicle



Esta tabla sirve para relacionar una misión con los distintos vehículos que la realizan.

Figura 15: Tabla mission_vehicle.

9.1.10. Tabla relacional permission_role



Esta tabla sirve para relacionar un rol con los distintos permisos que tiene.

Figura 16: Tabla permission_role.

9.1.11. Tabla relacional polimórfica record_sensor

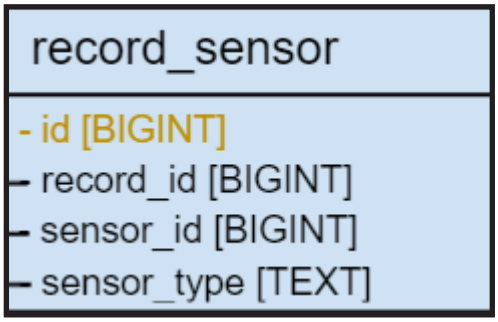


Figura 17: Tabla record_sensor.

Esta tabla sirve para relacionar una toma de datos con distintos sensores que lleva el barco.

Esta relación tiene una particularidad y es que la tabla con la que se relaciona está en un campo que se llama sensor_type. Con esto conseguimos que si se quiere añadir un sensor nuevo al vehículo y que éste se muestre en los datos solo tenemos que crear una nueva tabla en la base de datos con el nombre "sensor_[nombre del sensor]"

9.2. Frontend.

A la hora del desarrollo del frontend se ha optado por la utilización de Vue JS [7] debido a que es un framework de JavaScript reactivo, esto significa que va a reaccionar a cualquier cambio en los datos que le lleguen a cada uno de sus componentes.

9.2.1. Arquitectura de ficheros

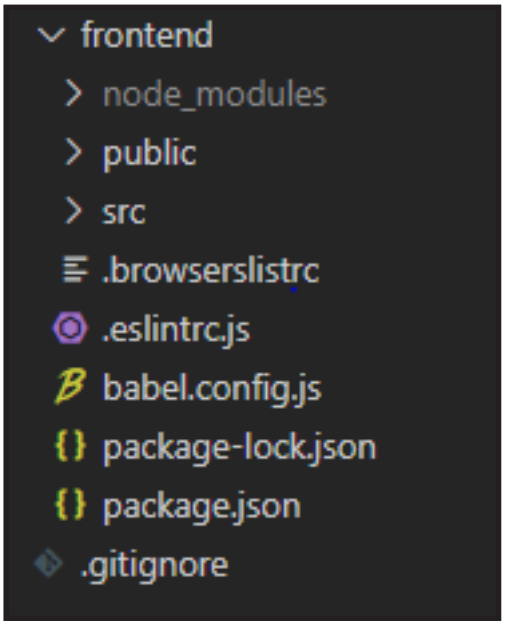


Figura 18: Arq. ficheros del frontend.

En un primer nivel la arquitectura de ficheros está organizada por el creador de proyectos de Vue JS, contando con 3 carpetas y una serie de ficheros necesarios para la configuración del proyecto. Estas carpetas son node_modules, public y src.

En la carpeta node_modules se encuentran todos los ficheros necesarios para las librerías instaladas. Esta carpeta se genera automáticamente al ejecutar "npm install".

En la carpeta public se encuentran todos los ficheros de acceso publico, tales como fotografías, tipografías y ficheros, entre otros.

En la carpeta src se encuentra la mayor parte de los elementos del proyecto de frontend, en ella se encuentra el código desarrollado. Dentro de esta carpeta se localizan otras carpetas como, assests, components, router y views.

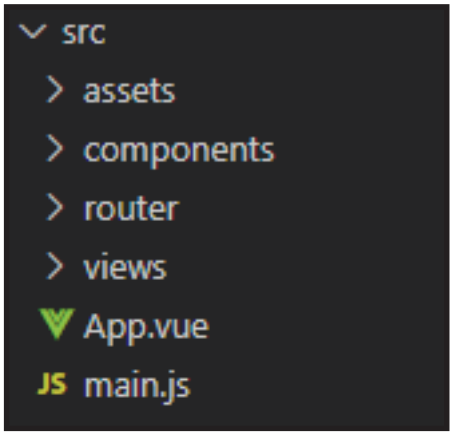


Figura 19: Contenido carpeta src.

En el segundo nivel dentro de la carpeta src como se muestra en la figura 19, nos encontramos con 4 carpetas y los ficheros "main.js" y "App.vue", estos ficheros se encargan respectivamente de iniciar la aplicación y montarla en el html así como de enlazarla con el resto de componentes.

En la carpeta assets se encuentran los ficheros de imágenes que no se quiere que sean de acceso publico. Aquí se encuentra un fichero css llamado "generalColors.styl", encargado de definir los colores utilizados en la aplicación.

En la carpeta router se encuentra un fichero llamado "index.js", encargado de definir qué componentes hay que cargar en cada ruta.

En la carpeta views se encuentran todas las vistas de la aplicación, estando en la misma carpeta las publicas, y en una subcarpeta llamada "admin" las del panel de administración.

En la carpeta components como se muestra en la figura 20, se encuentran todos los componentes desarrollados. Dentro de esta carpeta se encuentran 2 ficheros y 4 carpetas. Estos ficheros son importantes, sobre todo el fichero "Map.vue". Este componente es el que muestra todo el mapa apoyándose en la librería correspondiente.

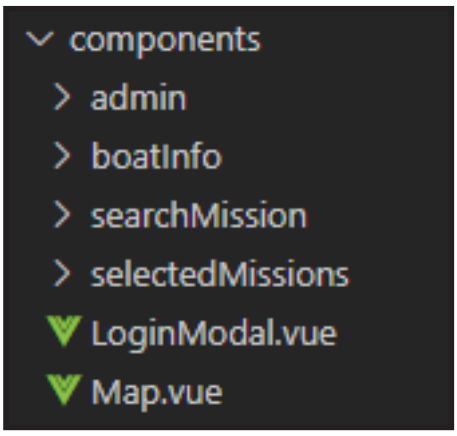


Figura 20: Contenido carpeta components.

En un tercer nivel dentro de la carpeta components se pueden observar 4 carpetas, admin, boatInfo, searchMission y selectedMissions. Estas carpetas contienen los componentes necesarios para realizar su cometido.

En la carpeta boatInfo se encuentra un componente que es el encargado de mostrar la información del barco cuando se selecciona mediante el ratón.

En la carpeta searchMission se encuentran los componentes necesarios para el funcionamiento del buscador de misiones sumado a la recomendación cuando el usuario no ha introducido caracteres para la búsqueda.

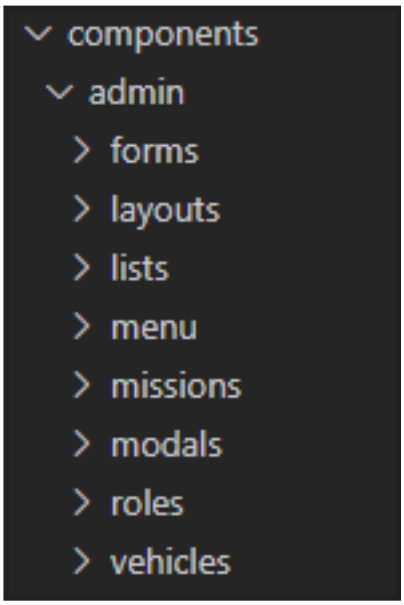


Figura 21: Contenido carpeta admin.

En la carpeta admin se encuentran los componentes necesarios para renderizar el panel de administración, dentro de la misma se encuentra otra separación en carpetas según su funcionalidad.

Como se observa en la figura 21 en un cuarto nivel dentro de la carpeta de admin se localizan 8 carpetas, cada una de ellas contiene los componentes necesarios para realizar sus funciones.

En la carpeta forms, los componentes necesarios para crear los formularios. Hay 2 tipos de formularios; uno de ellos solo permite enviar información cuando el valor introducido es igual a uno dado, y otro formulario estándar en el que se requieren unos campos.

En la carpeta layouts se puede encontrar los componentes necesarios para la renderización de unas plantillas utilizadas para el panel de administración.

En la carpeta lists se puede encontrar una plantilla para las listas de cualquier cosa. En este proyecto, esta lista se utiliza para mostrar los usuarios, las misiones, los vehículos, roles, permisos, entre otros.

En la carpeta menú se puede encontrar los componentes que renderizan el menú. Están diferenciados el menú para móvil con respecto al menú general para dispositivos de mayor tamaño como tabletas gráficas y ordenadores.

En la carpeta modals se puede encontrar la plantilla para los modales que se ven. Los modales son “ventanas emergentes” que se utilizan principalmente para los formularios.

En las carpetas missions, roles y vehicles se puede encontrar el componente que se encarga de renderizar una pantalla al ver una misión, un rol o un vehículo.

9.2.2. Librerías utilizadas

Para la realización del frontend nos hemos apoyado en una serie de librerías externas, a continuación se expone el uso que se les da en el frontend.

- **axios [10]:** Se utiliza como canal de comunicación entre el frontend y backend.

- **vue-leaflet [11]:** Esta librería permite la visualización y la colocación de marcas en un mapa.

- **vue2-leaflet-rotatedmarker [13]:** Nos permite que el elemento marker de vue-leaflet pueda tener una rotación para simular la dirección a la que navega el barco.

- **stylus [14]:** Nos ayuda con la generación del CSS [33] haciendo que sea mucho más fácil, cómodo y rápido su uso.

9.2.3. Componentes

En esta sección se tratarán los detalles acerca de los componentes que forman la aplicación. Estos componentes se pueden agrupar en 2 diferentes tipos: vistas y componentes.

Antes de profundizar en estos dos tipos de componentes es necesario señalar el VueRouter [34]. VueRouter es una herramienta que facilita el enrutamiento de las páginas dentro de la aplicación. Genera un fichero bajo la ruta *"/src/router/index.js"* en el cual se puede crear rutas nuevas fácilmente. Las rutas generadas en este fichero tendrán que hacer referencia a componentes del tipo vista.

Los componentes de tipo vista son los encargados de generar la página y a su vez llamar a los componentes necesarios para su visualización. Dentro de este grupo destacan dos subgrupos: la vista del mapa, que es la que se muestra a cualquier usuario que entre a la aplicación y la parte de administración, la cual solo la pueden ver los usuarios registrados.

- **Vista Map:** Esta vista contiene los componentes más importantes de la aplicación, comenzando por un componente llamado "Map" que es el encargado de generar el mapa. A este componente le llega información sobre las misiones seleccionadas y qué elementos debe mostrar. Reacciona automáticamente a cualquier cambio en esta información.

```
<Map
    :missions="missions"
    :show_boats_route="show_boats_route"
    :show_next_waypoint="show_next_waypoint"
    :show_waypoints="show_waypoints"
    :show_wind="show_wind"
    :zoomControl="false"
    @showInfo="showBoatInfo($event)"
    @mapFocused="show_recomend_search = false"
    @mapUnfocused="show_recomend_search = false"
/>
```

Figura 22: Llamada al componente Map.

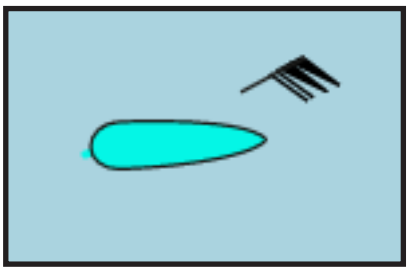


Figura 23: Marcador del barco.

A la vez que genera el mapa también se encarga de dibujar los barcos seleccionados, como se observa en la figura 23, este dibujo se genera a través de posicionar un SVG [35] diseñado con Illustrator [21] en la posición del barco. Al elegir un SVG como elemento con el que visualizar el barco se consiguen varias cosas: una imagen escalable sin perder calidad, y por otro lado la posibilidad modificar el color.

A su vez esta vista contiene otros elementos. A través de una barra de control como se observa en la figura 24, en la esquina inferior izquierda se puede seleccionar qué elementos se quiere visualizar:

- Visualizar ruta que ha hecho el vehículo.
- Visualizar punto al que se dirige el vehículo.
- Visualizar datos del viento que capta el vehículo.
- Visualizar ruta a realizar por el vehículo.

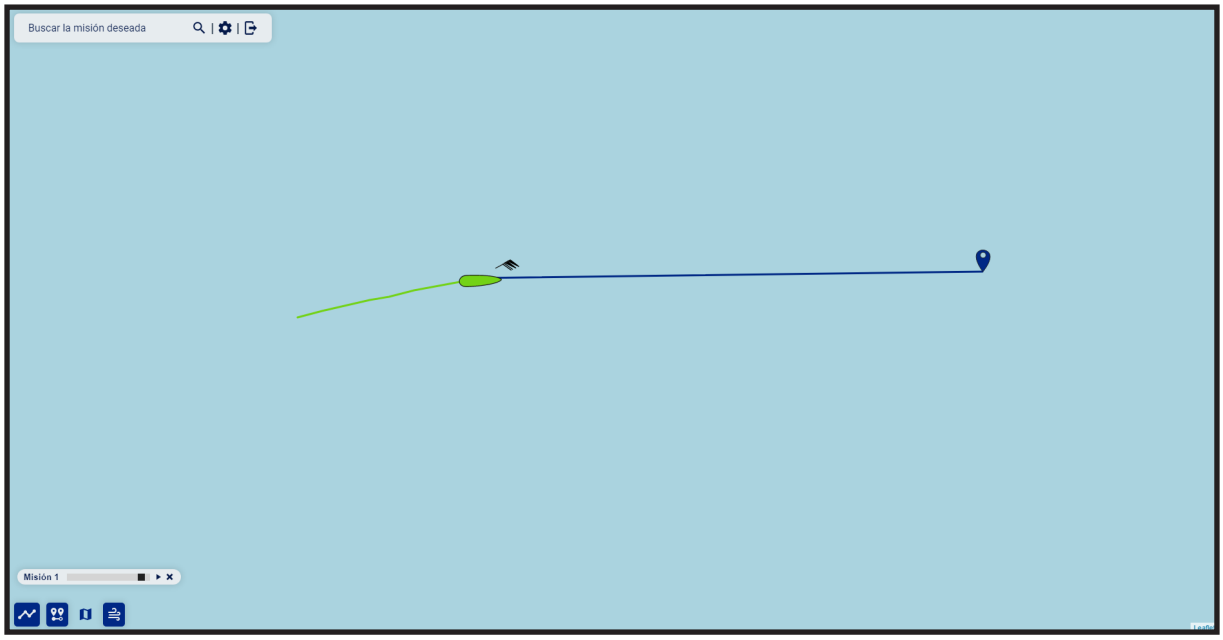


Figura 24: Visualización de una misión del vehículo autónomo.

Es de destacar el sistema de visualización del viento basado en Wind Barbs [38]. A través de un sistema ideado a base de SVG que dinámicamente se sitúa en relación a la posición del barco para indicar el angulo de incidencia del viento sobre el barco y la velocidad del mismo. Cada triangulo significa que el viento va a 50 nudos, cada raya a 10 nudos y cada raya pequeña a 5 nudos.

Figura 25: Seleccionar mision.

En la misma vista podemos encontrar otro componente que sirve para buscar la misión que un usuario desee visualizar, este componente permite buscar directamente la misión escribiendo su nombre en el buscador, o navegar por las distintas misiones que se le permite visualizar y elegir la que quiera (Figura 25).

Figura 26: Reproductor de misión.

Por último se puede encontrar un reproductor de misión en el cual se puede ver de manera rápida que misiones están seleccionadas y avanzar, pausar o reanudar su reproducción (Figura 26).

- **Vista Admin:** Para la elaboración de todo el apartado de administración se ha tenido en cuenta principalmente la escalabilidad. Este panel se ha desarrollado completamente con componentes reutilizables, por ejemplo, existe un componente de lista al cual le llega la información que mostrar y este componente se repite en todas las listas a lo largo del panel.

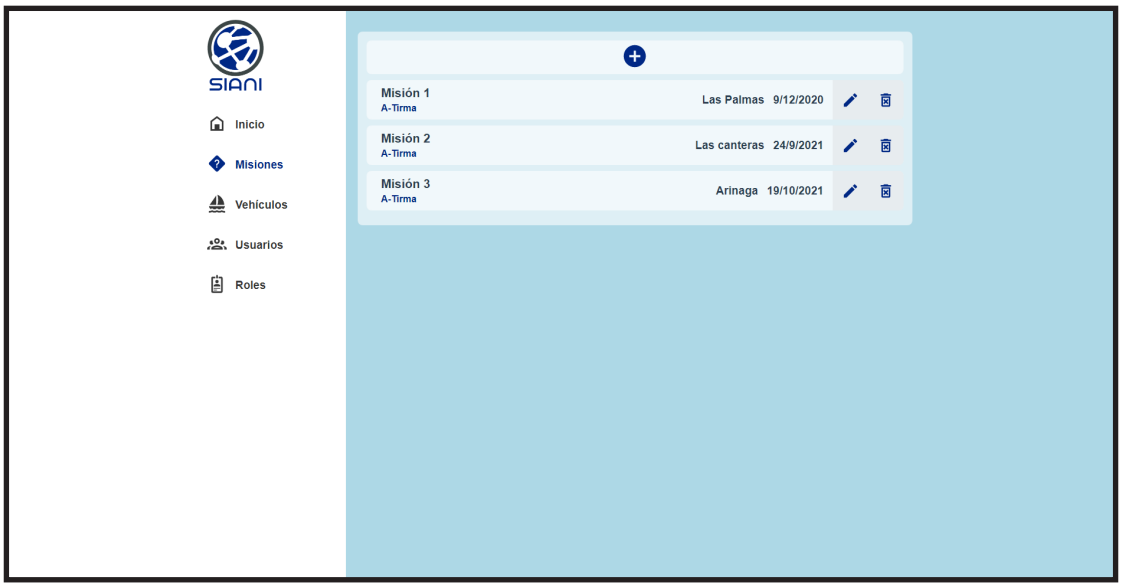


Figura 27: Vista del panel de administrador de la aplicación web.

A continuación se exponen todos estos componentes reutilizables utilizados en el panel de administración comentando sus características más importantes.

El primero de estos componentes son las listas de elementos, a este componente aparte de la información a mostrar, dependiendo de la sección en la que se use, se le adjuntan unos elementos de control que en conjunción con los permisos que tenga el usuario le permitirá eliminar o editar elementos de la lista.

Figura 28: Vista de componente de Lista general.

Al hacer click en algunos elementos de las listas del administrador se puede visualizar una vista más detallada de dicho elemento, esta vista se compone de una cabecera donde se encuentran los botones de control sobre el elemento en cuestión y de un espacio destinado a la información necesaria del elemento.

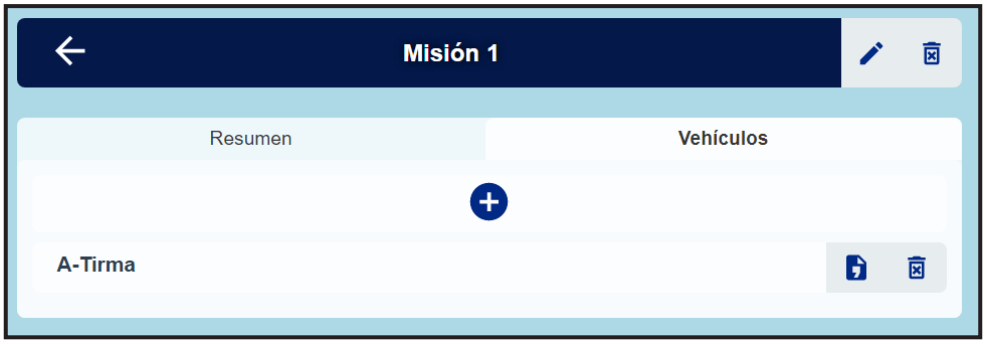


Figura 29: Vista de información general sobre un elemento.

En algunos elementos, como las misiones, se puede encontrar unas pestañas de resumen y vehículos, especialmente en la de vehículos se puede encontrar una tabla con los vehículos relacionados con ella, tal como se muestra en la figura 29. En esa misma tabla se pueden administrar las relaciones de esa misión.

Figura 30: Vista de formulario.

Al añadir, editar o borrar elementos en el panel de administrador aparece una ventana, este es el componente de formulario (Figura 30).

Este componente crea una ventana con un formulario que varía según los parámetros que reciba, siendo *"type"* el parámetro más importante, con el cual se identifica qué formulario se está mostrando.

Dentro de este componente se observa que dependiendo del tipo de campo a mostrar, se llama a otro componente que lo renderiza y así poder tener distintos tipos como:

- Texto.
- Fecha.
- Contraseña.
- Área de texto.
- Selector entre opciones.
- Selector de color.

```
<template>
    <GeneralModal
        @close="$emit('close')"
        @nextPage="nextPage()"
        @previousPage="previousPage()"
        :showBackArrow="showBackArrow"
        :isShowNextButtonDisabled="isShowNextButtonDisabled"
        :next_button_text="next_button_text"
    >
        <form action="">
            <div v-for="page in formData" :key="page.id" class="modal-content" :class="{showing: formData.indexOf(page) == 0}">
                <h2 v-if="page.title != ''">
                    {{page.title}}
                </h2>
                <div v-for="field in page.fields" :key="field.id">
                    <TextField v-if="field.type == 'text'" :field="field" @changingValue="showNextPageButton()"/>
                    <DateTimeField v-else-if="field.type == 'datetime'" :field="field" @changingValue="showNextPageButton()"/>
                    <TextAreaField v-else-if="field.type == 'textarea'" :field="field" @changingValue="showNextPageButton()"/>
                    <SelectField v-else-if="field.type == 'select'" :field="field" @changingValue="showNextPageButton()"/>
                    <ColorField v-else-if="field.type == 'color'" :field="field" @changingValue="showNextPageButton()"/>
                    <PasswordField v-else-if="field.type == 'password'" :field="field" @changingValue="showNextPageButton()"/>
                </div>
            </div>
        </form>
    </GeneralModal>
</template>
```

Figura 31: Código general de un formulario.

Figura 32: Notificación.

Al realizar cambios en cualquier registro de la aplicación hay un elemento que da una respuesta instantánea informando de la correcta realización de algunas tareas como añadir, editar o borrar cualquier registro. Este elemento se ha denominado "notificaciones".

Toda comunicación entre frontend y backend está gestionada por la vista en la que se produce esta comunicación, por ejemplo, si se está en la vista de vehículos y se edita un vehículo, el propio componente de vista vehículos gestionará esa petición a través de axios [10].

```
addVehicle: function(input_vehicle){
    axios.post(`https://atirma.iusiani.ulpgc.es/api/admin/vehicle`, input_vehicle)
        .then(response => {
            this.vehicles = response.data.data
            this.$store.commit('addNotification', {
                type: 'success',
                title: 'Vehículo añadido'
            })
        }).catch(e => {
            console.log(e);
            this.$store.commit('addNotification', {
                type: 'error',
                title: `Error añadiendo vehículo`
            })
        });
},
```

Figura 33: Código de petición post a la API.

9.3. Backend.

Para el desarrollo de el backend del proyecto se seleccionó la tecnología Node JS [8], por la facilidad que ofrece para la creación de una API. Asimismo, desde un punto de vista más personal, era una tecnología que me permitía aprender sobre ella.

Todo el proyecto se apoya en un framework que se llama express [15], este framework ayuda a montar APIs, y hace que el código de Node JS sea mucho más sencillo y fácil de seguir.

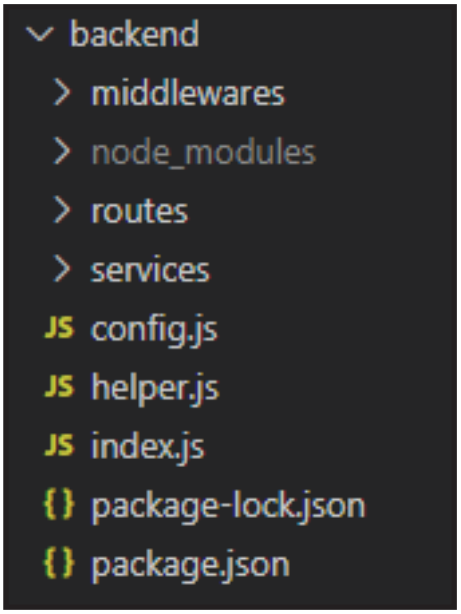


Figura 34: Arq. ficheros del backend.

9.3.1. Arquitectura de ficheros.

En este caso, al contrario que en el Frontend, la arquitectura inicial no la crea ningún asistente para la creación de proyectos, sino que debía ser desarrollada, por lo que se optó por una arquitectura que en un primer nivel tendría 3 carpetas y los ficheros de configuración necesarios para el funcionamiento de la API.

En la carpeta middlewares se encuentran los elementos que actúan como filtros para otros elementos, como por ejemplo la autentificación para acceder a una ruta.

En la carpeta routes podemos encontrar todos los manejadores de rutas del proyecto, para hacer más sencilla la API se han creado unos ficheros encargados de manejar las rutas y así el "fichero maestro" no es demasiado extenso y es más fácil de entender.

En la carpeta services se pueden encontrar todos los controladores con las funciones que se llaman desde los propios manejadores de rutas, estos controladores suelen tener el mismo nombre que el manejador de ruta para facilitar su uso.

En la raíz del proyecto se encuentran los ficheros encargados de la configuración del proyecto y de sus librerías.

9.3.2. Librerías utilizadas.

Para la realización del backend nos hemos apoyado en una serie de librerías externas las cuales voy a desglosar aquí.

- **express [15]:** Este framework nos facilita en la creación de nuestra api, aportandonos una base con la que trabajar.

- **cors [16]:** Se utiliza para generar de manera automática las cabeceras de las llamadas a la API.

- **bcrypt [17]:** Se utiliza para encriptar las contraseñas de los usuarios.

- **jsonwebtoken [18]:** Brinda una interfaz con la que generar tokens con la información que queramos de los usuarios registrados. Se utiliza para la autentificación de usuarios.

- **mysql2 [19]:** Permite la conexión con nuestra base de datos. Proporciona unas funciones necesarias para la utilización de la misma.

9.3.3. Funciones importantes.

A continuación se muestran una serie de funciones importantes en el código de la API, como por ejemplo el fichero de inicialización de la misma que está en la carpeta raíz, llamado "index.js".

- **index.js:** Este es el fichero encargado de la inicialización de la API, y conecta todo. Es la columna vertebral de la API. Analizando el fichero, en una primera instancia se inicializa la aplicación express, se le añade cors [16] para que se permita las llamadas a la base de datos y respetar la cabecera de las mismas.

```
app.use('/', AdminLoginRouter);
app.use('/', MissionSearch);
app.use('/admin/user', AdminUsersRouter);
app.use('/admin/mission', AdminMissionsRouter);
app.use('/admin/vehicle', AdminVehiclesRouter);
app.use('/admin/role', AdminRolesRouter);
```

Figura 35: Implementación de rutas en el backend.

En este momento se le añaden las rutas apuntando hacia los ficheros correspondientes en su carpeta de rutas y bajo el enlace que se determine.

- **Rutas:** En este apartado se tomará como ejemplo el fichero "adminMission.js". Dentro de este tipo de ficheros, lo primero que se hace es importar todos los recursos necesarios, como el middleware de autentificación o el controlador oportuno.

  Seguidamente, siguiendo una estructura muy similar hay un bloque por cada petición que se puede recibir en este apartado. Cada una de estas peticiones llaman a una función del controlador pertinente.

```
router.get('/:id', authentication, async function(req, res, next) {
    try {
        res.json(await adminMission.get(req.params.id));
    } catch (err) {
        console.error(`Error while getting mission with id: ${req.params.id}`, err.message);
        next(err);
    }
});
```

Figura 36: Gestión de rutas en el backend.

- **Controladores:** Este tipo de fichero es el encargado de definir las funciones que usan las rutas para conseguir sus datos, y a la vez hacen de puente entre la ruta y la base de datos.

```
async function get(id){
    let data = false
    if(id){
        const mission = await db.query(
            `SELECT *
            FROM missions
            WHERE id=?`,
            [id]
        );
        data = helper.emptyOrRows(mission)[0];
    }else{···
    }
    const meta = {};

    return {
        data,
        meta
    }
}
```

Figura 37: Función de recopilación de datos.

En primera instancia se importan los archivos de configuración de la base de datos donde figuran todos los datos necesarios para la conexión a la misma.

Posteriormente se declara cada función para las distintas rutas de los controladores.

- **auth.js:** Este fichero es el único middleware que tiene el proyecto. Un middleware es una función que hace de intermediario para filtrar conexiones, en este caso detecta si el usuario está conectado a la aplicación.

  La función se llama en el momento de activar la ruta seleccionada, y si se detecta que el usuario está autentificado continuará con su ejecución pero en el caso de que no lo esté la ejecución parará instantáneamente.

```
let authentication = (req, res, next) => {
    // Leer headers
    let token = req.headers.authorization;
    jwt.verify(token, 'secret', (err, decoded) => {
        if(err) {
            next('route')
        }else{
            // Creamos una nueva propiedad con la información del usuario
            if(decoded) req.user = decoded.data //data viene al generar el token en login.js
            next()
        }
    });
}
```

Figura 38: Función de autentificación de usuarios.

10. Conclusiones.

El desarrollo de esta solución digital para SIANI [1], así como el proceso de planificación y documentación de la misma han sido un proceso complejo. Un proceso del que valorar los conocimientos que he podido aportar, así como aquellos que he aprendido, y podré incorporar a proyectos futuros. Por su relevancia, me gustaría destacar a continuación, a modo de conclusiones del presente documento académico, cuáles han sido las principales:

- Aprender Node JS [8] y sus librerías, así como su utilidad para el desarrollo de proyectos dada la versatilidad de esta tecnología, y la facilidad que ofrece para el desarrollo de APIs.

- Modular en Vue JS [7]. Haciendo componentes más independientes y que se pudiesen reutilizar, lo que resulta especialmente útil para mejorar la eficiencia en el proceso de desarrollo.

- Uso de OpenStreetMap [12]. Dibujar y hacer marcas en el mapa, tales como rutas o marcadores, un aprendizaje que puede implementarse en futuros proyectos con componentes de geolocalización.

- Funcionamiento del SIANI, con proyectos como el actual velero A Tirma, en cuyos datos disponibles se ha basado la presente solución digital.

- Documentación de proyectos. Las bases necesarias no solo a lo largo del proyecto de desarrollo con una metodología de trabajo ágil, sino la documentación del mismo de modo que esta información resulte útil y accesible para futuras consultas.

- Detectar la importancia de componentes tales como las notificaciones, que inciden directamente en la experiencia de los usuarios al utilizar la herramienta.

Con respecto a los objetivos definidos en el inicio del documento, se han alcanzado los siguientes: Se ha diseñado y desarrollado con éxito la infraestructura digital necesaria para analizar y realizar el seguimiento de las misiones del barco autónomo a vela A-Tirma, con el procedimiento y especificaciones que se han detallado a lo largo de todo el documento.

Se ha dotado a esta herramienta con las funcionalidades necesarias para utilizarla en el seguimiento y análisis de las misiones de otros vehículos marinos.

En cuanto a los objetivos académicos, todos han sido alcanzados. Destacando especialmente el desarrollo de los conocimientos y competencias teóricas y prácticas aplicadas al desarrollo de esta aplicación, así como las habilidades para elaborar la documentación de un proyecto.

Asimismo, me gustaría destacar las siguientes recomendaciones futuras relacionadas con la herramienta de software y su uso:

1. Si bien este software se ha basado en los datos disponibles del velero autónomo A Tirma, ha sido preparado para posibilitar su uso no solamente con otros vehículos marinos, sino terrestres.

2. Este detalle recoge el mayor potencial de la herramienta, capaz de convertirse en un hub de información accesible, tanto para usuarios profesionales y conocedores del sector, como para usuarios no vinculados con el SIANI pero que estén interesados en conocer los diferentes desarrollos e investigaciones llevadas a cabo.

3. Esta accesibilidad es posible gracias las vistas intuitivas de la aplicación web, que permiten a cualquier usuario aprender su funcionamiento de forma fácil tanto desde dispositivos móviles como ordenadores o tabletas gráficas.

Apéndice

1. Instrucciones para el despliegue.

A continuación se detallan tres ejemplos de despliegue de la aplicación: un despliegue local, uno en Amazon AWS [4] y el último en el servidor proporcionado por el SIANI [1].

1.1. Despliegue en Local.

Para desplegar este proyecto en local se debe, en primer lugar, clonar el repositorio de git [24] en la carpeta deseada. Para ello utilizamos el siguiente comando:

"cd [Ruta]"

"git clone https://github.com/Pendregon/TFG.git"

Para el correcto despliegue de la aplicación en local se debe instalar MySQL [20] y una herramienta externa como XAMPP [36], que sirve para facilitar la gestión bases de datos MySQL. Una vez se tiene un servidor de base de datos funcionando en local, añadimos a éste el documento que se encuentra en la raíz del proyecto, llamado *"default_database.sql"*.

Seguidamente se debe instalar el backend de la aplicación, para ello nos trasladaremos a la carpeta clonada del repositorio llamada *"backend"*. Una vez en la carpeta se usará la herramienta npm [9] para instalar todas las librerías que se requieran. Para finalizar con el backend, se ejecutará el servicio con la herramienta Node JS [8].

"cd backend"

"npm install"

"node index.js"

Se debe hacer lo mismo en el frontend con la diferencia de que al ejecutar el servicio se utilizará la herramienta npm.

"cd ../frontend"

"npm install"

"npm run dev"

Una vez finalizado se debe tener en cuenta para futuros usos que para el funcionamiento de la aplicación se debe activar el servidor de base de datos a través de XAMPP y posteriormente ejecutar en consola los 2 servicios, frontend y backend.

1.2. Despliegue en servidor del SIANI.

Para desplegar la aplicación en el servidor del SIANI [1], tal como se detalló en el anterior apartado, se debe clonar el repositorio de git en la ruta deseada y posteriormente instalar las dependencias de cada uno de los proyectos.

"cd /usr/local/share/atirma/"

"git clone https://github.com/Pendregon/TFG.git"

"cd backend"

"npm install"

"cd ../frontend"

"npm install"

En este punto se debe transformar la aplicación de Vue JS a una versión para web. Con ello se consigue minimizar todo el código haciendo que su rendimiento aumente.

"npm run build"

La aplicación ya está instalada, ahora se debe cargar la base de datos en el servidor. Para ello hay que conectarse a la base de datos como usuario administrador y una vez dentro utilizar el fichero llamado "default_database.sql" situado en la raíz del proyecto para cargar los datos iniciales. A su vez se debe crear un usuario y una contraseña para la conexión de la aplicación a esta base de datos.

"mysql -u root -p"

"CREATE DATABASE IF NOT EXISTS `tfg`;"

"CREATE USER username IDENTIFIED BY password;"

Salir de la consola MySQL[20] para cargar el fichero inicial con el siguiente comando:

"mysql -u root -p tfg < /usr/local/share/atirma/default_database.sql"

A continuación se debe cambiar el fichero de configuración de lighttpd [39] para que la aplicación funcione correctamente. En él se debe redirigir el tráfico de la API a la IP y el puerto de la misma.

```
server.modules   += ( "mod_openssl",
                      "mod_proxy",
)

$SERVER["socket"] == "192.168.53.9:80" {
    $HTTP["host"] == "atirma.iusiani.ulpgc.es" {
        url.redirect = ("^/.*" => "https://atirma.iusiani.ulpgc.es/%0$0")
    }
}
else $SERVER["socket"] == "192.168.53.9:443" {
    ssl.engine  = "enable"
    ssl.ca-file = "/etc/ssl/certs/ca-certificates.crt"
    ssl.pemfile = "/etc/lighttpd/ssl/atirma_web06.pem"
    ssl.cipher-list = "ECDHE-RSA-AES256-SHA384:AES256-SHA256:RC4:HIGH:!MD5:!aNULL:!EDH:!AESGCM"
    ssl.honor-cipher-order = "enable"
    $HTTP["host"] == "atirma.iusiani.ulpgc.es" {
        $HTTP["url"] =~ "^/api" {
            proxy.server = (
                "" => (( "host" => "127.0.0.1",
                         "port" => 3000)
                )
            )
        }
        else {
            server.document-root = "/usr/local/share/atirma/frontend/dist"
        }
    }
    url.rewrite-if-not-file = (
        "/.*" => "/index.html"
    )
}
index-file.names := ( "index.html" )
```

Figura 39: Ejemplo de configuración Lighttpd.

Por último se debe configurar el servicio de la api para que siempre esté funcionando. Para ello se utilizará el systemd [40] para crear un nuevo servicio llamado atirma, tal como se muestra en la figura 40.

```
[Unit]
Description=A-Tirma Backend
After=network.target

[Service]
WorkingDirectory=/usr/local/share/atirma-app
User=atirma
Group=www-data
ExecStart=/usr/bin/node /usr/local/share/atirma/backend/index.js
Restart=on-failure
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=AtirmaApp-0.1

[Install]
WantedBy=multi-user.target
```

Figura 40: Ejemplo de configuración del servicio.

Para finalizar se debe inicializar los servicios creados y reiniciar el lighttpd.

1.3. Despliegue en Amazon AWS.

En primer lugar se debe lanzar 2 instancias. Por un lado una instancia EC2 [6] del tipo t2.micro como mínimo, este tipo de instancia es la más económica. En el lado de la base de datos se necesitará una instancia RDS [5] db.t2.micro.

Una vez lanzadas las 2 instancias necesarias, se comienza con la base de datos. Para ello lo único que es necesario es habilitar el acceso para poderse conectar desde un equipo fuera de la red, y ejecutar el fichero llamado "default_database.sql". Con esto se consigue tener los datos necesarios para que la aplicación funcione.

Por la parte de la instancia EC2 donde irá tanto el frontend como el backend, hay que seguir los siguientes pasos:

- Instalar git. Esta parte es muy simple, con el comando "yum install git".

- Ir a la carpeta donde queremos tener el proyecto, en este caso ha sido la carpeta opt en la raíz, por lo que se ejecuta el comando "cd /opt".

- Instalar el reporsitorio en la máquina. En este paso se debe usar el comando "sudo git clone https://github.com/Pendregon/TFG.git". Este paso debe realizarse como super usuario.

- Cambiar el propietario de los ficheros clonados del git. En este paso se debe usar el comando "sudo chown ec2-user /TFG -R ".

- Ahora se debe instalar Node JS en la máquina. Para ello se debe descargar nvm como gestor de descarga y con ello instalar Node. En este paso se ejecutará los siguientes comandos.
    - "curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash"
    - . ~/.nvm/nvm.sh"
    - "nvm install node"

- Instalar pm2, este paquete de Node JS sirve para crear un daemon con la ejecución del backend, así no habrá que preocuparse por dicha ejecución. "npm install pm2 -g".

- Instalar nginx como controlador http. Para instalarlo en un EC2 hay que acceder a las librerías de Amazon con este comando "sudo amazon-linux-extras install nginx1".

- En cuanto al frontend, para su correcta instalación hay que moverse a la carpeta del frontend, instalar npm para que se descarguen todas las librerías necesarias y construir el proyecto. "cd /opt/TFG/frontend", "npm install" y "npm run build".

- Por último se debe instalar el backend desde su carpeta y actualizar el proyecto. “cd /opt/TFG/backend” y “npm install”.

Finalmente para poder utilizar la aplicación hay que modificar el fichero de configuración de nginx con el comando “sudo nano /etc/nginx/nginx.conf”, tal como se muestra en la figura 41.

```
server {
    listen       80;
    listen       [::]:80;
    server_name  _;
    root         /usr/share/nginx/html;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;
    location / {
            root /opt/TFG/frontend/dist;
            try_files $uri /index.html;
    }
    location /api/ {
            proxy_pass http://localhost:3000/;
    }
    error_page 404 /404.html;
    location = /404.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

Figura 41: Ejemplo de configuración Nginx.

Por último ejecutar el daemon con el pm2 del backend, esto se consigue con el siguiente comando “pm2 start /opt/TFG/backend/index.js”.

Bibliografía.

- [1] SIANI, página web oficial (inglés). https://www.siani.es
- [2] 40866 - Trabajo fin de grado, página web de la asignatura (español). https://www2.ulpgc.es/aplicaciones/proyectosdocentes/pdf.php?id_proyecto=63324&NUEVA=1
- [3] iframe, página web MDN (español). https://developer.mozilla.org/es/docs/Web/HTML/Element/iframe
- [4] AWS, página web oficial (inglés). https://vuejs.org
- [5] Amazon RDS, guía de usuario (español). https://docs.aws.amazon.com/es_es/AmazonRDS/latest/UserGuide/Welcome.html
- [6] Amazon EC2, guía de usuario (español). https://docs.aws.amazon.com/es_es/AWSEC2/latest/UserGuide/concepts.html
- [7] Vue JS, página web oficial (inglés). https://vuejs.org
- [8] Node JS, página web oficial (inglés). https://nodejs.org/
- [9] Npm, página web oficial (inglés). https://www.npmjs.com
- [10] axios, página web oficial (inglés). https://axios-http.com
- [11] vue-leaflet, página web oficial (inglés). https://vue2-leaflet.netlify.app
- [12] OpenStreetMap, página de artículo de wikipedia (español). https://es.wikipedia.org/wiki/OpenStreetMap
- [13] vue2-leaflet-rotatedmarker, repositorio de github (inglés). https://github.com/mudin/vue2-leaflet-rotatedmarker

- [14] stylus, página web oficial (inglés). https://stylus-lang.com
- [15] express, página web oficial (inglés). https://expressjs.com/
- [16] cors, repositorio de github (inglés). https://github.com/expressjs/cors
- [17] bcrypt, página web referente al paquete de npm (inglés). https://www.npmjs.com/package/bcrypt
- [18] jsonwebtoken, repositorio de github (inglés). https://github.com/auth0/node-jsonwebtoken
- [19] mysql2, página web referente al paquete de npm (inglés). https://www.npmjs.com/package/mysql2
- [20] MySQL, página web oficial (inglés). https://www.mysql.com
- [21] Illustrator, página web oficial (español). https://www.adobe.com/es/products/illustrator.html
- [22] ADOBE, página web oficial (español). https://www.adobe.com/es/about-adobe/fast-facts.html
- [23] InDesign, página web oficial (español). https://www.adobe.com/es/products/indesign.html
- [24] git, página web oficial (inglés). https://git-scm.com
- [25] metodología agil, página de artículo de wikipedia (español). https://es.wikipedia.org/wiki/Desarrollo_ágil_de_software
- [26] Google maps, página de artículo de wikipedia (español). https://es.wikipedia.org/wiki/Google_Maps
- [27] PWA, página web MDN (español). https://developer.mozilla.org/es/docs/Web/Progressive_web_apps

- [28] Service Workers, página de Google hablando sobre ellos (español). https://developers.google.com/web/fundamentals/primers/service-workers?hl=es

- [29] JavaScript, página de articulo de wikipedia (español). https://es.wikipedia.org/wiki/JavaScript

- [30] React JS, página web oficial (inglés). https://reactjs.org

- [31] Angular, página web oficial (inglés). https://angular.io

- [32] PHP, página web oficial (inglés). https://www.php.net

- [33] CSS, página de artículo de wikipedia (español). https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada

- [34] VueRouter, página web oficial (ingles). https://router.vuejs.org

- [35] SVG, página de artículo de wikipedia (español). https://es.wikipedia.org/wiki/Gráficos_vectoriales_escalables

- [36] XAMPP, página de artículo de wikipedia (español). https://es.wikipedia.org/wiki/XAMPP

- [37] Linux LXC, página de artículo de wikipedia (español).´ https://es.wikipedia.org/wiki/LXC

- [38] Wind Barbs, página web donde se explica en qué consiste (ingles). https://www.weather.gov/hfo/windbarbinfo

- [39] Lighttpd, página web oficial (ingles). https://www.lighttpd.net

- [40] systemd página de artículo de wikipedia (español). https://es.wikipedia.org/wiki/Systemd