Trabajo de Fin de Grado

A-Tirma WebWare v.2: Infraestructura web para el seguimiento, monitorización, control y análisis de misiones para vehículos autónomos marinos

TITULACIÓN: Grado en Ingeniería Informática

AUTOR: González Ortega, Kenai Jesús

TUTORIZADO POR:
Domínguez Brito, Antonio Carlos
Cabrera Gámez, Jorge

Fecha [Junio/2024]

Agradecimientos

Quiero expresar mi más profundo agradecimiento a todas aquellas personas que han hecho posible no solo la realización de este Trabajo Final de Grado, sino también el éxito de esta etapa académica.

En primer lugar, agradezco al Profesor Antonio Carlos Domínguez Brito por su constante apoyo, valiosa orientación y paciencia a lo largo de todo este proyecto.

Al Profesor Jorge Cabrera Gamez, por su inestimable ayuda y sus sugerencias precisas que contribuyeron a mejorar significativamente la calidad de este trabajo.

A mi pareja, Jacqueline Quevedo, por su amor, comprensión y apoyo incondicional durante toda esta etapa. Su presencia ha sido una fuente constante de motivación y fortaleza para mí. A mi madre, por su inagotable apoyo y confianza en mis capacidades. Sin su sacrificio y dedicación, no habría sido posible llegar hasta aquí.

A toda mi familia por su aliento y soporte en los momentos más difíciles. Su fe en mí me ha impulsado a seguir adelante y superar los desafíos que se presentaron en el camino. Finalmente, a todas las personas que de una u otra manera han contribuido a mi crecimiento personal y académico, les agradezco de corazón. Este logro es el resultado del esfuerzo colectivo y el apoyo recibido de todos ustedes.

Resumen

El Trabajo Final de Grado se centra en el desarrollo y mejora de una infraestructura web para visualizar las misiones de navegación de vehículos autónomos marinos de la División de Robótica y Oceanografía Computacional del Instituto Universitario SIANI. Su objetivo es brindar acceso a la información de las misiones para usuarios externos y miembros de la División, permitiendo consulta y gestión de datos, así como la asignación de vehículos. Se destaca la implementación de nuevas funcionalidades, como la reproducción en tiempo real de las misiones, la optimización del rendimiento del sistema y la mejora de la visualización de datos. Estas mejoras amplían la utilidad de la infraestructura web, facilitando el seguimiento y análisis detallado de las misiones almacenadas.

Abstract

The Final Degree Project focuses on the development and improvement of a web infrastructure to visualize the navigation missions of marine autonomous vehicles of the Division of Robotics and Computational Oceanography of the SIANI University Institute. Its objective is to provide access to mission information for external users and members of the Division, allowing data consultation and management, as well as vehicle assignment. The implementation of new functionalities must be highligthed, such as real-time playback of missions, optimization of system performance, and improved data visualization. These enhancements extend the usefulness of the web infrastructure, making it easier to track and analyze stored missions in detail.

Índice general

Índice de figuras

Capítulo 1

Introducción, Estado actual y Objetivos iniciales

Este capítulo está diseñado para proporcionar una visión detallada y estructurada del proyecto, comenzando con una introducción al contexto y la importancia de la gestión de misiones de navegación. A continuación, se presenta un análisis del estado actual de la infraestructura web, identificando áreas que requieren mejoras. Esta evaluación inicial es crucial para establecer una línea base desde la cual se pueden medir los avances y las mejoras implementadas durante el proyecto.

El capítulo continua con la exposición de los objetivos iniciales del proyecto, delineando las metas específicas que se pretenden alcanzar. Estos objetivos están orientados a optimizar la funcionalidad, la accesibilidad y la seguridad del sistema, garantizando que cumpla con las necesidades tanto de los miembros de la División como de usuarios externos interesados en el seguimiento y análisis de las misiones de navegación. Esta estructura lógica y progresiva asegura una comprensión clara del proyecto y prepara el terreno para las secciones posteriores que detallarán las metodologías y resultados alcanzados.

Por ultimo, el capítulo concluye con la estructura que presenta este documento, especificando el contenido de cada capítulo.

1.1. Introducción

En el ámbito de la División de Robótica y Oceanografía Computacional del Instituto Universitario SIANI, la gestión eficiente de las misiones de navegación de vehículos es crucial para la realización exitosa de proyectos de investigación y exploración. La complejidad y la cantidad de datos generados durante estas misiones requieren herramientas avanzadas para su seguimiento y gestión. En este contexto, la implementación y mejora de una infraestructura web que permita la visualización y gestión de estas misiones se convierte en un aspecto fundamental.

El presente Trabajo Final de Grado se centra en la mejora de dicha infraestructura web, con el objetivo de proporcionar un acceso intuitivo y eficaz a la información relacionada con las misiones de navegación. Este sistema no solo servirá para satisfacer las necesidades de los miembros de la División, sino que también estará disponible para usuarios externos interesados en seguir de cerca el progreso y los resultados de las misiones.

Mediante este sistema, se pretende facilitar el acceso a datos críticos de las misiones, mejorar la colaboración entre investigadores y asegurar que la información sea gestionada de manera eficiente y segura. Esto permitirá a los investigadores centrarse en el análisis y la interpretación de los datos, potenciando así los avances en los proyectos de exploración y estudio del entorno.

1.2. Estado actual

Este Trabajo Final de Grado se enfoca en la mejora de una infraestructura web ya existente. En esta sección se explicará y mostrará el estado actual de dicha infraestructura antes del inicio del proyecto. Se describirán las funcionalidades previamente implementadas, las herramientas y tecnologías utilizadas, así como los desafíos y limitaciones identificadas en el sistema actual. Esta evaluación inicial es muy importante para entender las áreas que requieren mejoras y establecer una base sólida sobre la cual se desarrollarán las nuevas funcionalidades y optimizaciones.

Cabe destacar que la información mostrada en esta sección ha sido consultada en la Memoria del Trabajo Final de Grado de la primera versión de la infraestructura web [42].

1.2.1. Frameworks y Herramientas

En la versión original del sistema se emplea un framework (*Vue2 [22]*) y una herramienta (*Vuex [30]*) que serán reemplazados por sus equivalentes oficiales y actualizados y las correspondientes librerías.

- **Vuejs:** En la versión original se utiliza como framework Vuejs (*"Vuejs es un framework de JavaScript para crear interfaces de usuario. Se basa en HTML [47], CSS [49]*

y JavaScript [48] estándar y proporciona un modelo de programación declarativo basado en componentes que le ayuda a desarrollar eficientemente interfaces de usuario de cualquier complejidad." [24]), pero en su versión Vue2 [22], la cual es una versión anterior de la actual. Para profundizar en la razón del cambio véase la sección 3.2.

- **Vuex :** Vuex (*"Vuex es una biblioteca y patrón de gestión de estado para aplicaciones Vue.js. Sirve como un almacén centralizado para todos los componentes de una aplicación, con reglas que garantizan que el estado sólo pueda modificarse de forma predecible."[30]*) es un gestor de estados utilizado en la versión original del proyecto. Para profundizar en la razón de su remplazo por su equivalente véase la sección 3.3.
- **vue2-leaflet-rotatedmarker :** Además de los elementos mencionados, también hay que destacar la librería ***vue2-leaflet-rotatedmarker [13]***. Es un componente desarrollado por mudin (nombre de usuario de GitHub [34](*"GitHub es una plataforma donde puedes almacenar, compartir y trabajar junto con otros usuarios para escribir código."[35]*) 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 iconos de los vehículos en la dirección de la navegación.

1.2.2. Funciones

Las funciones más importantes implementadas inicialmente son las siguientes :

- **Reproducción de misiones :** El sistema es capaz de reproducir misiones que previamente se han subido a la web mediante un archivo CSV (*"Los archivos CSV (del inglés comma-separated values) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas"[28]*). Como se muestra en la figura 1.1, la reproducción muestra el icono del vehículo con su respectiva ruta, además de los controles de reproducción en la esquina inferior izquierda.
- **Administrar elementos :** La aplicación web permite la administración de elementos como las misiones, vehículos y usuarios. Para ello, el usuario tiene un panel general en el que aparecen todos los elementos editables como se muestra en la *Figura 1.2*

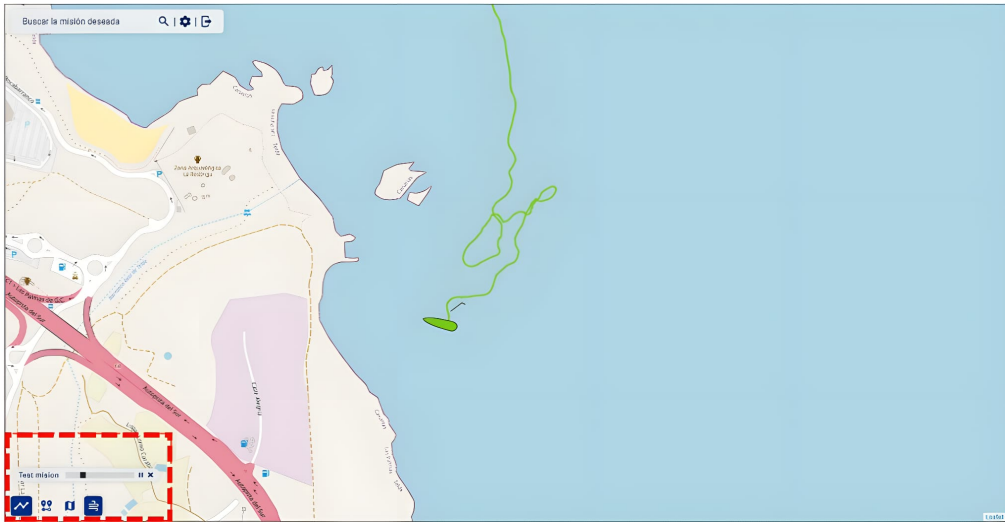


Figura 1.1: Misión reproduciéndose en la versión original. Se resaltan los controles de reproducción en la esquina inferior izquierda.

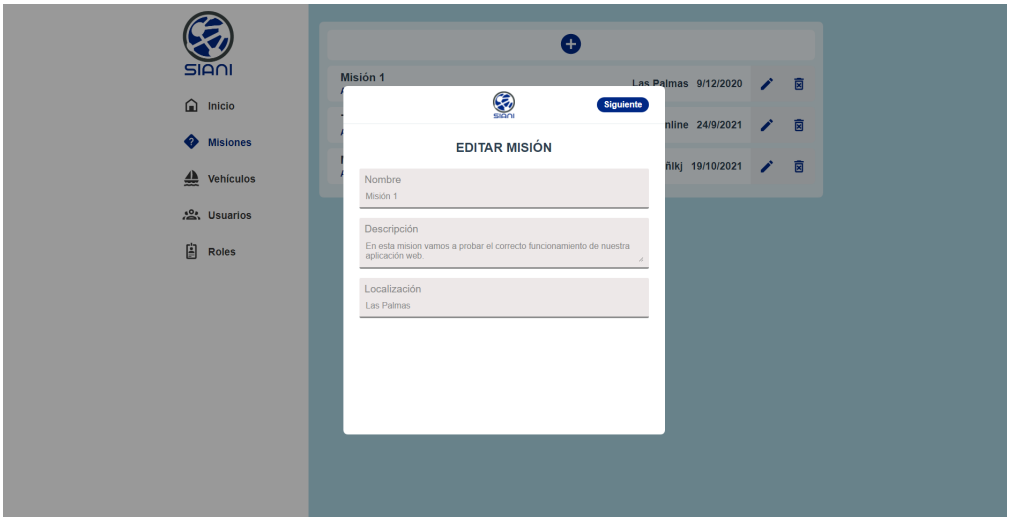


Figura 1.2: Panel general de la versión original

1.2.3. Evaluación del estado actual

Al analizar detenidamente el estado actual del proyecto, se ha identificado que el principal desafío radica en la refactorización y mejora del código existente. El sistema actualmente está basado en una versión obsoleta del framework Vue.js, lo que limita su rendimiento y capacidad de mantenimiento. Por lo tanto, actualizar a una versión más reciente de Vue.js es una prioridad, ya que esto no solo optimizará el rendimiento, sino que también facilitará futuras mejoras y ampliaciones.

Además de la actualización del framework, es crucial mejorar la interfaz de usuario. Esto incluye no solo un rediseño visual para hacerla más moderna y atractiva, sino también la adición y perfeccionamiento de funcionalidades clave. Dos de los elementos más importantes a mejorar son el panel de edición de elementos y la reproducción de misiones.

El panel de edición de elementos debe ser más intuitivo y eficiente para los usuarios, permitiendo una gestión más fácil y rápida de las misiones de navegación.

La reproducción de misiones, por otro lado, debe ser más clara y precisa, proporcionando a los usuarios una visualización más comprensible y detallada del progreso y los resultados de las misiones. Además, se buscará optimizar la carga y visualización de los datos, garantizando así la mejor experiencia de usuario posible .

1.3. Objetivos iniciales

Los objetivos planteados en la propuesta de TFG son los siguientes:

- **Refactorización de la infraestructura web de monitorización, control y seguimiento de misiones para vehículos autónomos marinos actualmente operativos en la división ROC del SIANI.**
  Respecto a este objetivo, como se mencionó en la sub-sección 1.2.3, los esfuerzos se centrarán en dos áreas principales: la actualización tecnológica del framework subyacente y la mejora de la experiencia del usuario mediante una interfaz más funcional y amigable. Estas mejoras son fundamentales para asegurar que la plataforma cumpla con los estándares actuales y sea capaz de soportar las necesidades futuras de la División de Robótica y Oceanografía Computacional del Instituto Universitario SIANI.
- **Mejora de la capacidad de visualización de la información recopilada y registrada durante una misión de navegación con vistas a facilitar su análisis, tanto durante, como a posteriori de haberse llevado a cabo.**
  Dentro de este objetivo se incluye la función de reproducción de misiones en tiempo real, que se desarrolla en la sección 3.8, además de la mejora de visualización de estas misiones.

1.4. Estructura del Documento

El presente documento se estructura de la siguiente manera:

- **Capítulo 1 Introducción, estado actual y objetivos iniciales:** Se presenta el contexto, justificación, estado actual, objetivos y estructura del documento.
- **Capítulo 2 Competencias específicas y aportaciones del trabajo** : Se nombran y justifican las competencias específicas cubiertas además de explicar la aportación de este proyecto.
- **Capítulo 3 Desarrollo:** Se describe detalladamente el proyecto realizado, incluyendo las decisiones técnicas y las tecnologías utilizadas.
- **Capítulo 4 Conclusiones y Trabajo Futuro:** Se resumen las principales conclusiones del proyecto y se proponen posibles líneas de trabajo futuro para mejorar y expandir el sistema.
- **Capítulo 5 Apéndice:** Se incluye la información del despliegue de la aplicación web en un servidor y un glosario de términos técnicos utilizados a lo largo del documento.
- **Capítulo 6 Bibliografía:** Se listan las referencias bibliográficas consultadas para el desarrollo del proyecto.

Con este documento se espera proporcionar una visión completa y detallada del proyecto A-Tirma WebWare v2, destacando su relevancia y las aportaciones realizadas al campo de la gestión y visualización de misiones.

Capítulo 2

Competencias específicas y aportaciones del trabajo

En este capítulo se detallan las competencias específicas cubiertas durante el proyecto. Cada sub-sección de la sección 2.1 de este capítulo se centra en una competencia particular, describiendo cómo se ha aplicado en el contexto del proyecto . Este enfoque permite una comprensión clara y detallada del impacto del proyecto en el desarrollo profesional y técnico del autor, así como en el campo de la ingeniería informática. Para terminar el capítulo se especificará lo que este proyecto aporta a la División de Robótica y Oceanografía Computacional del Instituto Universitario SIANI.

2.1. Competencias

2.1.1. CI01: 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.

Durante el desarrollo de A-Tirma WebWare v.2, se llevaron a cabo varias iniciativas para asegurar la fiabilidad, seguridad y calidad del sistema informático. Se implementaron protocolos de seguridad para proteger los datos sensibles de las misiones de vehículos autónomos marinos (Como la gestión de permisos de misiones explicado en la Sección 3.9, o la identificación de vehículos, mencionado en la Sección 3.8). Además, se realizaron pruebas de calidad y rendimiento para garantizar que el sistema funcionara de manera eficiente.

2.1.2. CI02: 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.

El proyecto A-Tirma WebWare v.2 destacó por la necesidad de planificar, desarrollar y dirigir un sistema web avanzado dedicado al seguimiento y análisis de misiones de vehículos autónomos marinos. Desde la fase inicial de planificación, se establecieron objetivos claros para guiar la implementación eficiente del proyecto. Una de las principales contribuciones técnicas fue la migración exitosa de Vue2 a Vue3, desarrollado en la sección 3.2, lo cual permitió la incorporación de nuevas funcionalidades, como la reproducción en tiempo real de misiones, véase sección 3.8, mejorando significativamente la utilidad del sistema, permitiendo visualizar datos en vivo.

2.1.3. CI12: 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.

En el marco de este competencia, se llevó a cabo un trabajo significativo enfocado en el diseño y gestión de la base de datos SQL (*"SQL es un lenguaje específico de dominio, diseñado para administrar, y recuperar información de sistemas de gestión de bases de datos relacionales."[29]*) para almacenar y administrar datos cruciales de misiones. Este proyecto destacó por su integración estratégica con XAMPP [33], una herramienta que facilitó la administración eficiente y el óptimo rendimiento de estas bases de datos.

El rediseño de la estructura de la base de datos SQL fue crucial para cumplir con los requisitos específicos del sistema. Además, la implementación cuidadosa garantizó no solo la integridad y la accesibilidad de los datos, sino también una operación eficiente y efectiva del sistema en su conjunto. La integración con XAMPP permitió no solo la gestión centralizada y segura de las bases de datos, sino también una optimización continua del rendimiento, asegurando que el sistema pudiera manejar de manera efectiva la carga de trabajo y responder de manera rápida a las demandas operativas.

Este enfoque avanzado no solo puso de manifiesto habilidades técnicas en la implementación práctica de sistemas de gestión de bases de datos, sino que también subrayó la capacidad para utilizar herramientas adecuadas para mejorar la eficiencia y la funcionalidad de las aplicaciones basadas en datos.

2.1.4. CI13: Conocimiento y aplicación de las herramientas necesarias para el almacenamiento, procesamiento y acceso a los sistemas de información, incluidos los basados en web.

En el proyecto A-Tirma WebWare v.2, se utilizaron herramientas y tecnologías como Node.js [32] y Lighttpd [41] para asegurar un almacenamiento, procesamiento y acceso eficientes a los sistemas de información. Node.js permitió crear un servidor web robusto y escalable, mientras que Lighttpd ofreció una gestión eficiente de las conexiones concurrentes, optimizando el rendimiento del sistema. Se integraron librerías específicas para el manejo y procesamiento de datos, como *mysql2* explicada en la sub-sección 3.1.6, lo que facilitó el almacenamiento seguro y la recuperación rápida de información, así como la implementación de funcionalidades avanzadas como la búsqueda y filtrado de datos.

Además, se implementaron tecnologías de procesamiento en tiempo real tanto en el lado del cliente como del servidor para presentar datos de manera intuitiva y accesible. El uso de frameworks y librerías de front-end como Vue.js y Vuetify [27] mejoró significativamente la interfaz de usuario, haciendo la navegación y la interacción con los datos más fluidas. Para proteger la integridad y confidencialidad de los datos, se utilizó la librería de cifrado *bcrypt* [11], la cual se explica en la sub-sección 3.1.6, y políticas de acceso controlado, asegurando que solo el personal autorizado pudiera acceder a información sensible.

2.1.5. CI17: Capacidad para diseñar y evaluar interfaces persona computador que garanticen la accesibilidad y usabilidad a los sistemas, servicios y aplicaciones informáticas.

En el contexto de esta competencia, se realizó un trabajo enfocado en el diseño avanzado y la evaluación de interfaces de usuario para garantizar la accesibilidad y mejorar la usabilidad del sistema. El proyecto se destacó por la creación de interfaces intuitivas y accesibles utilizando tecnologías modernas como Vue.js y Vuetify.

El diseño de interfaces de usuario fue meticuloso, asegurando que fueran intuitivas y fácilmente navegables para los usuarios finales. La implementación de Vue.js permitió una experiencia interactiva y dinámica, mientras que Vuetify facilitó la integración de componentes estandarizados y visualmente atractivos. Estas decisiones de diseño no solo mejoraron la estética del sistema, sino que también optimizaron la experiencia del usuario al hacerla más accesible y funcional.

2.2. Aportaciones

En esta sección se enumerarán las aportaciones que este Trabajo Final de Grado proporciona tanto a la infraestructura web ya existente como a la propia División de Robótica y Oceanografía Computacional del Instituto Universitario SIANI.

2.2.1. Aportaciones a la infraestructura web

1. **Actualización Tecnológica:**

   - Modernización del framework Vue.js, lo que asegura un código más eficiente, mantenible y compatible con las tecnologías web actuales.

2. **Mejora de la Interfaz de Usuario:**

   - Rediseño y optimización de la interfaz para hacerla más intuitiva, moderna y atractiva. Esto incluye la mejora del panel de gestión de los diferentes elementos (misiones, vehículos y usuarios) y la claridad en la reproducción de misiones.

3. **Optimización de Rendimiento:**

   - Implementación de técnicas para la gestión de datos y visualización, garantizando una experiencia de usuario fluida y sin interrupciones, incluso con grandes volúmenes de datos.

4. **Aumento de la Funcionalidad:**

   - Incorporación de nuevas funcionalidades y perfeccionamiento de las existentes, lo que permite a los usuarios realizar tareas de manera más eficiente y efectiva.

5. **Mejora en la Reproducción de Misiones:**

   - Optimización de la funcionalidad de reproducción de misiones, proporcionando una representación más clara y precisa de los datos, lo que facilita el análisis y la interpretación de los resultados por parte de los científicos.

6. **Gestión de Permisos:**

   - Introducción de un sistema de gestión de permisos que permite un control granular sobre quién puede ver y editar las misiones, mejorando la seguridad y la colaboración entre usuarios.

7. **Acceso Externo:**

   - Extensión del acceso a usuarios externos interesados, permitiendo una mayor difusión y colaboración en los proyectos de investigación y exploración.

2.2.2. Aportaciones a la División de Robótica y Oceanografía Computacional del Instituto Universitario SIANI

1. **Facilitación del Análisis de Datos:**

    - Mejora en la visualización y manejo de grandes volúmenes de datos generados durante las misiones de navegación, permitiendo a los investigadores realizar análisis más profundos y precisos.

2. **Fomento de la Colaboración Científica:**

    - Herramientas que promueven una colaboración más efectiva entre los miembros de la División de Robótica y Oceanografía Computacional, así como con investigadores externos.

Capítulo 3

Desarrollo

En este capítulo, se detalla el proceso de desarrollo del proyecto, destacando las herramientas y tecnologías clave utilizadas. Se abordan los frameworks seleccionados, como XAMPP [33] para la gestión de bases de datos, Vue.js [25] y Vuetify [27] para la interfaz de usuario, Node.js [32] para el backend (El backend es la parte de una aplicación que gestiona la lógica del negocio, las bases de datos y la comunicación entre el servidor y el frontend, que es la parte con la cual interactúa el usuario.), y Lighttpd [41] como servidor web . Cada herramienta se eligió por sus ventajas específicas y su contribución a un desarrollo eficiente y eficaz.

El capítulo también cubre la migración de Vue2 a Vue3 y el cambio de gestor de estados, explicando las razones detrás de estas decisiones y los beneficios obtenidos. Estas migraciones fueron esenciales para mejorar el rendimiento y la capacidad de mantenimiento del proyecto.

Además, se describen las mejoras implementadas, tales como optimizaciones en la experiencia de usuario, mejora de inserción de datos de misiones a través de archivos CSV [28], y mejoras en la base de datos y en la reproducción de misiones. Se profundiza en la funcionalidad de reproducción, explicando cómo los diferentes componentes, como la barra de reproducción y la trayectoria del vehículo, se combinan para proporcionar una experiencia de usuario coherente y eficiente.

También se cubre la arquitectura de la base de datos y la funcionalidad de la reproducción en tiempo real, incluyendo las restricciones de versión del servidor y el sistema operativo, y el sistema de suscripción y petición de datos en tiempo real.

Por último, se aborda la gestión de permisos respecto a las misiones, donde cada usuario tiene la capacidad de gestionar qué otros usuarios pueden ver o editar sus misiones.

3.1. Frameworks y herramientas Utilizadas

3.1.1. XAMPP para la gestión de bases de datos SQL

XAMPP [33] es una distribución de software libre y de código abierto que facilita la creación de un servidor web local, integrando componentes esenciales para el desarrollo y pruebas de aplicaciones web. Una de sus funcionalidades más destacadas es la gestión de bases de datos SQL [29] a través de MySQL [52] o MariaDB [31], junto con la herramienta phpMyAdmin [53] para su administración. Esta herramienta se utiliza en el proyecto para la gestión de la base de datos en el entorno de desarrollo, para poder realizar las pruebas y cambios necesarios en un entorno seguro. Según [33, 51], sus ventajas son:

Fácil instalación y configuración: XAMPP permite instalar y configurar MariaDB/MySQL junto con phpMyAdmin de manera rápida y sencilla, sin necesidad de realizar configuraciones manuales complejas. Esto agiliza el proceso de creación de un entorno de desarrollo local.

Entorno de pruebas Seguro: Utilizar XAMPP para gestionar bases de datos SQL en un entorno local permite a los desarrolladores realizar pruebas y depurar aplicaciones sin riesgo de afectar los datos en un entorno de producción.

Interfaz intuitiva con phpMyAdmin: phpMyAdmin proporciona una interfaz gráfica intuitiva que facilita la gestión de bases de datos, permitiendo a los desarrolladores realizar operaciones complejas de manera visual sin necesidad de utilizar comandos SQL en línea de comandos. Esto agiliza el desarrollo en el entorno local para poder desplegarlo en el servidor sin errores.

3.1.2. Uso de Vue.js para la construcción de interfaces de usuario

Vue.js [25] es un *framework* progresivo de JavaScript [48] utilizado para construir interfaces de usuario interactivas y dinámicas. Este framework se caracteriza por su flexibilidad y su capacidad para integrarse de manera incremental en un proyecto, lo que permite adoptar solo las funcionalidades necesarias sin imponer una estructura rígida. [24]

Arquitectura reactiva: La arquitectura reactiva de Vue.js permite que la interfaz de usuario se actualice automáticamente en respuesta a los cambios en los datos subyacentes. Esta característica facilita la creación de aplicaciones web interactivas y responsivas, ya que el DOM [46] (El DOM *Document Object Model* es una interfaz de programación que representa la estructura de un documento HTML o XML como un árbol de nodos, permitiendo su manipulación dinámica.) se actualiza automáticamente sin necesidad de intervención manual por parte del desarrollador.

Composition API: La Composition API [23], introducida en Vue 3, proporciona una forma más flexible y modular de organizar y reutilizar el código. A diferencia del Options API [23], la Composition API permite definir la lógica del componente de manera más clara

y cohesiva, mejorando la mantenibilidad y escalabilidad del código. Esta API [45] es especialmente útil para gestionar componentes complejos, facilitando su descomposición en funciones reutilizables.

3.1.3. Integración de Vuetify para componentes de UI mejorados

Vuetify [27] es un marco de componentes de Material Design [37] para Vue.js. Proporciona un conjunto de componentes predefinidos siguiendo las directrices de Material Design de Google [37], lo que permite a los desarrolladores crear interfaces de usuario visualmente atractivas y consistentes con facilidad. Vuetify ofrece una amplia gama de componentes personalizables, incluyendo botones, tarjetas, cajones de navegación, y más, que pueden integrarse fácilmente en aplicaciones Vue.js para mejorar su funcionalidad y estética.

Al integrar Vuetify en el proyecto, se puede aprovechar su amplia colección de componentes para construir rápidamente interfaces de usuario receptivas y atractivas sin necesidad de un extenso trabajo de estilismo o diseño personalizado. Esto no solo acelera el proceso de desarrollo, sino que también garantiza que la aplicación resultante cumpla con los estándares de diseño modernos y proporcione una experiencia de usuario cohesiva en diferentes dispositivos y tamaños de pantalla.

Además, la documentación exhaustiva de Vuetify y el apoyo activo de la comunidad facilitan a los desarrolladores comenzar y solucionar cualquier problema que puedan encontrar durante el desarrollo. Esta accesibilidad y soporte contribuyen a un proceso de desarrollo más fluido y, en última instancia, resultan en una aplicación más pulida y profesional.

En resumen, la integración de Vuetify en el proyecto mejora el proceso de desarrollo de la interfaz de usuario al proporcionar un conjunto sólido de componentes y recursos de Material Design. [26]

3.1.4. Node.js para el desarrollo de backend

Node.js [32] es un entorno de tiempo de ejecución de JavaScript de código abierto que permite ejecutar JavaScript [48] en el servidor. Es ampliamente utilizado para el desarrollo de aplicaciones de backend debido a su naturaleza basada en eventos y su capacidad para manejar grandes cantidades de conexiones simultáneas de forma eficiente. Se puede encontrar mas informació acerca de NodeJS en [18].

Características Principales:

- **Event-Driven:** Node.js utiliza un modelo de E/S sin bloqueo y orientado a eventos, lo que lo hace adecuado para aplicaciones en tiempo real que requieren una alta capacidad de respuesta.
- **JavaScript en ambos lados:** Permite a los desarrolladores utilizar JavaScript tanto en el lado del cliente como en el lado del servidor, lo que facilita la creación de aplicaciones de pila completa con un lenguaje de programación coherente. Una aplicación de pila

completa (*full-stack*) abarca tanto el *frontend* (interfaz de usuario) como el *backend* (lógica del servidor y bases de datos), proporcionando una solución completa desde la presentación hasta la gestión de datos.

- **Módulos nativos:** Node.js cuenta con un ecosistema de módulos nativos y de terceros que facilita la construcción de aplicaciones escalables y modulares.

Node.js se utiliza comúnmente en el proyecto para implementar la lógica del servidor y proporcionar servicios de backend para la aplicación web. Su facilidad de uso y su capacidad para escalar hacen que sea una opción popular para el desarrollo de sistemas web modernos.

3.1.5. Lighttpd como Servidor Web

Lighttpd (o "Lighty") [41] es un servidor web de código abierto conocido por su velocidad y eficiencia. Es una opción popular para implementaciones que requieren un consumo reducido de recursos y un alto rendimiento en servidores con capacidades limitadas. La información acerca de Lighttpd se ha consultado en [50].

Características Principales:

- **Rendimiento Optimizado:** Lighttpd está diseñado para ser rápido y eficiente en el manejo de solicitudes HTTP, ideal para servidores con grandes volúmenes de tráfico.
- **Uso Eficiente de Recursos:** Consumo bajo de memoria y capacidad de procesamiento, adecuado para entornos con limitaciones de recursos hardware.
- **Configuración Flexible:** Ofrece una configuración flexible a través de archivos de configuración simples y directos, permitiendo ajustes específicos según las necesidades del proyecto.
- **Soporte para Múltiples Protocolos:** Además de HTTP, Lighttpd puede manejar protocolos como FastCGI y SCGI, ampliando su versatilidad en entornos de desarrollo web.

Lighttpd se utiliza comúnmente para servir contenido estático y dinámico en proyectos que requieren un servidor web rápido y eficiente. Su capacidad para escalar y su configuración flexible lo convierten en una opción popular para aplicaciones web que buscan optimizar el rendimiento del servidor.

3.1.6. Librerías Utilizadas

Backend :

- **bcrypt [11]**: Esta librería es utilizada para el hashing de contraseñas en el backend. Proporciona funciones para el cifrado seguro de contraseñas antes de almacenarlas en la base de datos, lo que garantiza que las contraseñas permanezcan seguras incluso si la base de datos es comprometida.

- **bcryptjs** [**4**]: Se trata de una alternativa a bcrypt para el hashing de contraseñas en JavaScript. Ofrece una implementación puramente en JavaScript de la función de hashing bcrypt, lo que permite su uso en entornos donde no se pueden utilizar módulos nativos.
- **express** [**39**]: Framework web de Node.js utilizado para la construcción de aplicaciones RESTful. Express simplifica el desarrollo de servidores web en Node.js al proporcionar una capa de abstracción sobre el protocolo HTTP, facilitando la creación de rutas, middleware y gestión de peticiones.
- **jsonwebtoken** [**1**]: Esta librería se utiliza para la implementación de JSON Web Tokens (JWT) en el backend. Los JWT son una forma segura de transmitir información entre partes como un objeto JSON, y son ampliamente utilizados para la autenticación y la autorización en aplicaciones web modernas.
- **mysql2** [**5**]: Cliente MySQL utilizado para interactuar con bases de datos MySQL desde aplicaciones Node.js. mysql2 es una librería rápida y segura que proporciona una interfaz para realizar consultas y transacciones de manera eficiente.
- **winston** [**54**]: Herramienta de registro utilizada para el registro de mensajes en el backend. Winston es una librería flexible y extensible que permite registrar mensajes de registro en varios niveles de severidad y en diferentes formatos, facilitando la depuración y el seguimiento del estado de la aplicación.

Frontend :

- **mdi/font** [**7**]: Esta librería proporciona iconos de Material Design para Vue.js. Ofrece una amplia variedad de iconos que pueden ser fácilmente integrados en aplicaciones Vue.js para mejorar la experiencia de usuario y la estética del diseño.
- **vue-leaflet** [**16**]: Componente de Vue.js para Leaflet, una biblioteca de mapeo interactiva. Permite integrar mapas interactivos en aplicaciones Vue.js para visualizar y manipular datos geoespaciales de manera efectiva.
- **vueuse** [**17**]: Conjunto de utilidades Vue.js. VueUse/Core proporciona una colección de utilidades esenciales para el desarrollo de aplicaciones Vue.js, incluyendo funciones para la gestión del estado, la manipulación de datos y la interacción con el DOM.
- **axios** [**2**]: Cliente HTTP para realizar solicitudes AJAX [43]. Axios es una librería de cliente HTTP basada en promesas que permite realizar solicitudes HTTP desde el navegador o desde Node.js de forma sencilla y eficiente.
- **core-js** [**10**]: Biblioteca que proporciona *polyfills* para características de ECMAScript que no están disponibles en todos los navegadores, permitiendo a los desarrolladores utilizar las últimas características de JavaScript de forma consistente en todos los entornos.
- **leaflet-rotate** [**14**]: Plugin para Leaflet que permite rotar los marcadores en el mapa. Leaflet-Rotate es un plugin para la biblioteca de mapeo Leaflet que añade la capacidad de rotar los marcadores en el mapa, lo que permite representar direcciones y

orientaciones de manera más precisa.

- **leaflet-rotatedmarker** [**8**]: Plugin adicional para Leaflet que proporciona funcionalidades adicionales para rotar los marcadores en el mapa. Leaflet-RotatedMarker complementa las características de Leaflet-Rotate y permite una mayor personalización en la representación de los marcadores en el mapa.
- **leaflet-textpath** [**12**]: Plugin para Leaflet que permite representar texto a lo largo de un camino en el mapa. Leaflet-TextPath facilita la creación de visualizaciones geoespaciales más detalladas y expresivas al permitir la representación de texto en formas personalizadas en el mapa.
- **pinia** [**19**]: Librería para la gestión del estado global en aplicaciones Vue.js. Pinia proporciona una API simple y reactiva para gestionar el estado compartido entre componentes de manera eficiente y escalable, facilitando el desarrollo de aplicaciones Vue.js con una arquitectura de estado robusta.
- **vue-leaflet-rotatedmarker** [**6**]: Plugin para Vue.js que permite rotar los marcadores en el mapa. Este plugin proporciona una integración directa entre Vue.js y Leaflet-RotatedMarker, permitiendo a los desarrolladores utilizar la funcionalidad de rotación de marcadores en sus aplicaciones Vue.js de manera sencilla. Esta librería es sustituta de la librería *vue2-leaflet-rotatedmarker*, mencionada en la sub-sección 1.2.1, la cual es cambiada por razones de compatibilidad con Vue3. Este plugin fue desarrollado por *AntoninRousset*, cuyo repositorio en GitHub se encuentra en https://github.com/AntoninRousset. Dado que no existe un plugin oficial en este momento, se recomendará migrar a uno oficial tan pronto como esté disponible.
- **vue-material-design-icons** [**15**]: Iconos de Material Design para Vue.js. Esta librería proporciona una amplia variedad de iconos de Material Design que pueden ser fácilmente integrados en aplicaciones Vue.js para mejorar la experiencia de usuario y la estética del diseño.
- **vue-router** [**20**]: Enrutador oficial para Vue.js. Vue Router es una herramienta que facilita la navegación en una aplicación Vue.js al permitir la definición de rutas y la navegación entre diferentes vistas de manera intuitiva y eficiente.
- **vuetify** [**21**]: Marco de componentes de Material Design para Vue.js. Vuetify es una biblioteca de componentes de interfaz de usuario basada en Material Design que proporciona una amplia gama de componentes predefinidos para la construcción rápida y fácil de aplicaciones Vue.js con un diseño moderno y atractivo.
- **dompurify** [**9**]: DOMPurify es una biblioteca de JavaScript que se utiliza para sanitizar (limpiar) el contenido HTML y evitar ataques XSS (Cross-Site Scripting). Básicamente, toma un fragmento de HTML, lo analiza y elimina cualquier contenido potencialmente dañino, asegurándose de que solo se permita HTML seguro.

3.2. Migración de Vue2 a Vue3

3.2.1. Introducción

La decisión de migrar de Vue.js 2 a Vue.js 3 fue fundamentada en una serie de consideraciones técnicas y prácticas que buscaban mejorar diversos aspectos del desarrollo y rendimiento de la aplicación. A continuación, se detallan las principales razones detrás de esta migración teniendo en cuenta a [36] y [38]:

3.2.2. Razones para el cambio

- **Mejoras en el rendimiento:** Vue.js 3 introduce un nuevo Motor de Renderizado denominado Reactivity, el cual se caracteriza por su eficiencia en el uso de memoria y su velocidad de actualización, atributos que superan significativamente a su predecesor, Vue.js 2.
- **Más modularidad y escalabilidad:** La versión 3 de Vue.js ha sido rediseñada para ser más modular, lo que facilita la integración de nuevas funcionalidades y mejora la escalabilidad del código base. La introducción de los Composables y el sistema de Composición API permiten una organización más eficiente del código y una reutilización más efectiva de la lógica entre componentes.
- **Optimizaciones de tamaño y carga:** Vue.js 3 presenta un tamaño de biblioteca reducido y una mejor gestión del Árbol de Componentes, lo que se traduce en una carga más rápida de las aplicaciones y una experiencia de usuario más fluida.En Vue.js, un componente es una instancia reutilizable y encapsulada que combina HTML, CSS y JavaScript para definir un segmento de la interfaz de usuario, facilitando la creación y gestión de aplicaciones web modulares y dinámicas.
- **Soporte continuo y comunidad activa:** La migración a Vue.js 3 garantiza un soporte continuo y una comunidad activa que proporciona recursos, herramientas y ayuda en línea. Esto asegura que la aplicación esté alineada con las últimas prácticas y tecnologías, y que pueda beneficiarse de futuras actualizaciones y mejoras en el ecosistema de Vue.js.

En resumen, la migración de Vue.js 2 a Vue.js 3 se realizó con el objetivo de aprovechar las mejoras en rendimiento, modularidad y herramientas de desarrollo que ofrece la última versión del framework, lo que contribuirá a una experiencia de usuario más eficiente y satisfactoria, así como a un desarrollo más ágil y escalable de la aplicación.

3.3. Cambio de gestor de estados

3.3.1. Introducción

En esta sección se presenta la justificación para el cambio de la biblioteca Vuex a Pinia en el contexto de la gestión de estado de la aplicación web. Se exploran las razones detrás de esta decisión y se establece el fundamento para la migración. Todas las afirmaciones se apoyan en [40].

Figura 3.1: Vuex vs Pinia. Fuente: [40].

3.3.2. Razones para el cambio

- **Rendimiento y escalabilidad:** Pinia ofrece una Arquitectura Reactiva optimizada que puede mejorar el rendimiento y la escalabilidad en comparación con Vuex, especialmente en proyectos de mayor envergadura o complejidad.
- **Integración con Vue 3 y Composition API:** Pinia se integra de manera más natural con Vue 3 y la Composition API, lo que permite aprovechar al máximo las características y mejoras introducidas en estas versiones más recientes de Vue.js.
- **Modularidad y mantenibilidad:** Pinia promueve una estructura más modular y desacoplada para la gestión del estado, lo que facilita la mantenibilidad y el escalado del código en el tiempo.

- **Comunidad:** Pinia cuenta con una comunidad activa y soporte continuo, lo que garantiza que la biblioteca esté actualizada y en constante evolución para satisfacer las necesidades de los desarrolladores de JavaScript en el ecosistema de Vue.js.

El cambio de Vuex a Pinia se fundamenta en la necesidad de mejorar el rendimiento, la escalabilidad y la mantenibilidad de la aplicación, aunque la razón de peso que sostiene este cambio es que el propio equipo de Vue.js afirma que Pinia es la nueva biblioteca oficial de gestor de estados para Vue, como se puede observar en la *Figura 3.2*. Estas razones respaldan la decisión de migrar la gestión de estado a Pinia, proporcionando así una base sólida y moderna para el desarrollo del proyecto.

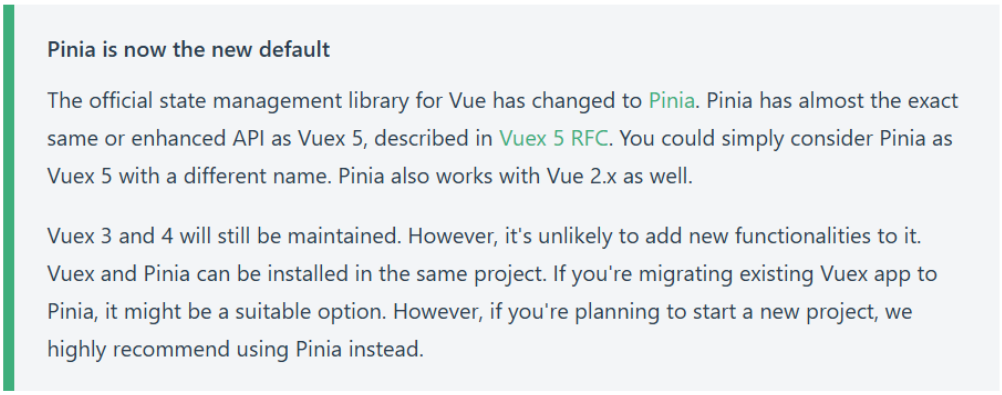


Figura 3.2: Recomendación para el uso de Pinia en sistemas que vengan usando Vuex. Fuente: [30]

3.4. Mejoras

Uno de los objetivos del proyecto era mejorar tanto el aspecto visual como el rendimiento de la aplicación web. Para ello se han realizado varios cambios.

3.4.1. Mejoras en la experiencia de usuario (UX)

Con el fin de mejorar la experiencia visual y la usabilidad de la aplicación, se han implementado diversas mejoras en el diseño de la interfaz de usuario. Una de las principales adiciones ha sido la introducción de menús contextuales, diseñados para guiar al usuario de manera más efectiva a través de las diversas funcionalidades de la aplicación. Estos menús están estratégicamente ubicados en puntos clave de la Interfaz de usuario (UI), ofreciendo acceso rápido y claro a las diferentes secciones y características de la aplicación. Estas diferencias se pueden notar en la Figura 3.3

Además de los menús, se han realizado ajustes en la disposición y presentación de elementos visuales para mejorar la coherencia y la legibilidad en toda la aplicación. Se ha prestado especial atención a la selección de colores, tipografías y espaciado, buscando crear una experiencia visual ordenada y atractiva para el usuario.

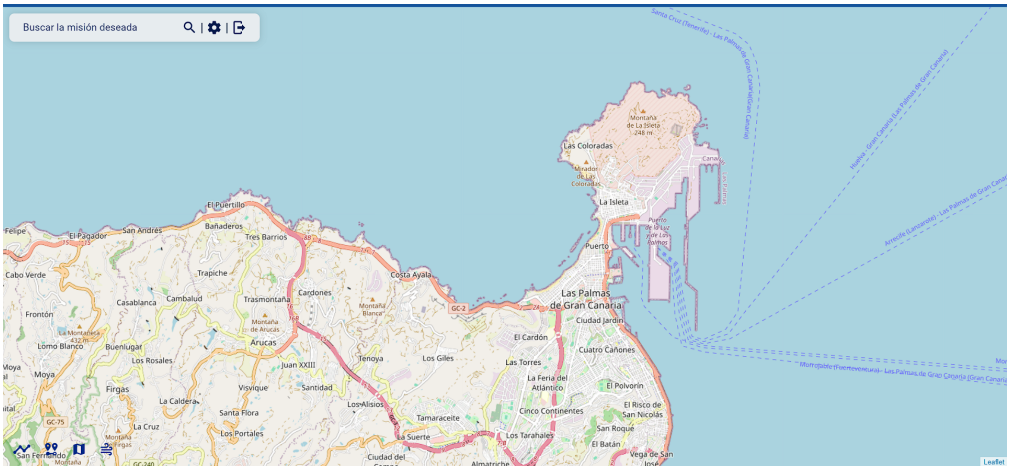
Otra mejora notable ha sido la optimización de la navegación, con la introducción de elementos visuales como botones de navegación intuitivos y marcadores visuales, que facilitan la orientación y la exploración del contenido para el usuario.

En conjunto, estas mejoras en la experiencia visual no solo contribuyen a una apariencia más atractiva y profesional de la aplicación, sino que también tienen como objetivo aumentar la usabilidad y la satisfacción del usuario al interactuar con la plataforma.

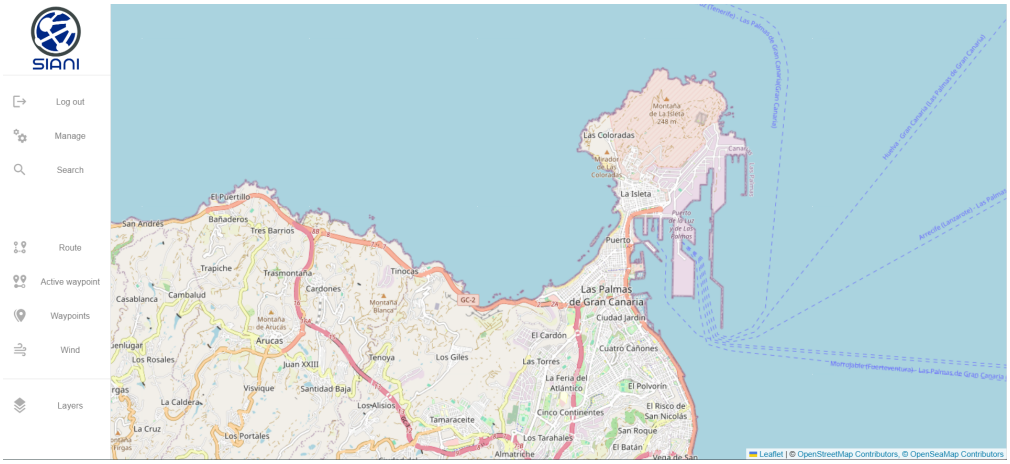
3.4.2. Mejora al agregar misiones a través de archivos CSV

La mejora más significativa con respecto a la inserción de misiones ha sido la reducción del tiempo de respuesta. Como se puede observar en la Figura 3.4a, la eficiencia en la inserción de datos era bastante costosa en tiempo, lo que llevó a una mejora significativa tras la refactorización del código y la optimización de las consultas SQL. Como se evidencia en la Figura 3.4b, se logró reducir el tiempo de inserción de datos mediante el CSV en aproximadamente un 96 % con respecto al tiempo original.

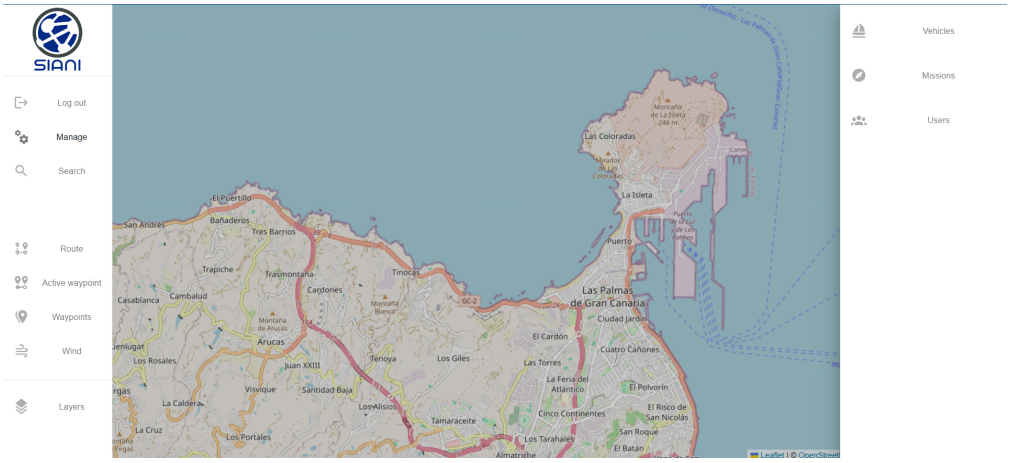
Inicialmente, cuando se trataban los datos de una misión obtenidos mediante archivo CSV, se insertaba cada registro individualmente, lo que resultaba en una carga computacional grande, ya que se debía realizar la conexión con la base de datos cada vez. Como se muestra en la Figura 3.5, la mejora ha consistido principalmente en realizar la inserción por bloques, lo que disminuye la carga computacional.



(a) Pagina principal original.



(b) Nueva pagina principal, con menú desplegado



(c) Nueva pagina principal, con ambos menús desplegados

Figura 3.3: Muestra de mejora visual

```
[nodemon] starting `node index.js`
Example app listening at http://localhost:3000
19542 inserciones en 46.61segundos
```

(a) Muestra de inserción de la versión original del proyecto

```
[nodemon] starting `node index.js`
Example app listening at http://localhost:3000
19542 inserciones en 1.45segundos
```

(b) Muestra de inserción de la versión corregida

Figura 3.4: Muestra de mejora de inserción de datos por CSV

```
async function insertMissionRecords(sensor_nav_data, records_data){
    const batchSize = 3000; // Tamaño del lote

    for (let i = 0; i < sensor_nav_data.length; i += batchSize) {

        const sensor_nav_batch = sensor_nav_data.slice(i, i + batchSize);
        let sensor_nav_placeholders = sensor_nav_batch.map(() => '(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)').join(',');
        let sensor_nav_values = sensor_nav_batch.flatMap(item => item);
        let query = `INSERT INTO sensor_navigation
                    (boat_time, reset_number, channel, raw_data, latitude, longitude, speed, heading, pitch, roll, true_wind_angle,
                    VALUES ${sensor_nav_placeholders}`;
        let sensor_nav = await db.query(query, sensor_nav_values);
        const sensor_nav_ids = helper.generateArray(sensor_nav.insertId, sensor_nav_batch.length);

        const records_batch = records_data.slice(i,i+batchSize);
        let records_placeholders = records_batch.map(() => '(?, ?, ?, ?)').join(',');
        let records_values = records_batch.flatMap((item, index) => {
            item[0] = sensor_nav_ids[index];
            return item;
        });
        query = `INSERT INTO records
                (sensor_id, mission_id, timestamp, data_source)
                VALUES ${records_placeholders}`;
        await db.query(query,records_values);
    }

}
```

Figura 3.5: Muestra del método utilizado para insertar grandes cantidades de datos en la base de datos.

3.4.3. Base de datos

Se han realizado modificaciones significativas en la estructura de la base de datos para abordar problemas de redundancia de tablas y mejorar el rendimiento del sistema. Como parte de este proceso, se han revisado y reorganizado las tablas existentes para eliminar duplicidades y optimizar la eficiencia en el almacenamiento y acceso a los datos. Además, se han añadido nuevas tablas para dar soporte a las funcionalidades adicionales y las nuevas implementaciones requeridas por los últimos desarrollos del proyecto. Estas modificaciones se abordarán en mayor detalle en el apartado 3.5. Estos cambios han sido implementados con el objetivo de garantizar una base de datos más eficiente, escalable y preparada para futuras expansiones y mejoras en la aplicación.

Una de las mejoras más notables ha sido la reducción significativa del tiempo de recuperación de datos. Esto ha sido posible gracias a la mejora de las consultas SQL, las cuales han sido optimizadas utilizando consultas avanzadas, como se muestra en la Figura 3.6, para agilizar el proceso de recuperación de datos, mejorando así la eficiencia y la capacidad de respuesta del sistema ante las solicitudes de los usuarios.

Además, se ha logrado una notable mejora en el tiempo de inserción de datos mediante la utilización de archivos CSV, como se detalla en la sección anterior. La combinación de estas mejoras ha contribuido a un rendimiento general mejorado del sistema, proporcionando una experiencia más fluida y eficiente para los usuarios al interactuar con la aplicación.

```
WITH
record_interval AS(
SELECT
    FROM_UNIXTIME(
        FLOOR(UNIX_TIMESTAMP(TIMESTAMP) / ?) * ?
    ) AS interval_start,
    MIN(id) AS id
FROM
    records
WHERE
    mission_id = ? AND data_source = ?
GROUP BY
    interval_start
)
SELECT
    sensor_navigation.*
FROM
    records
INNER JOIN record_interval ON records.id = record_interval.id
INNER JOIN sensor_navigation ON sensor_navigation.id = records.sensor_id
LIMIT ? OFFSET ?
`;
```

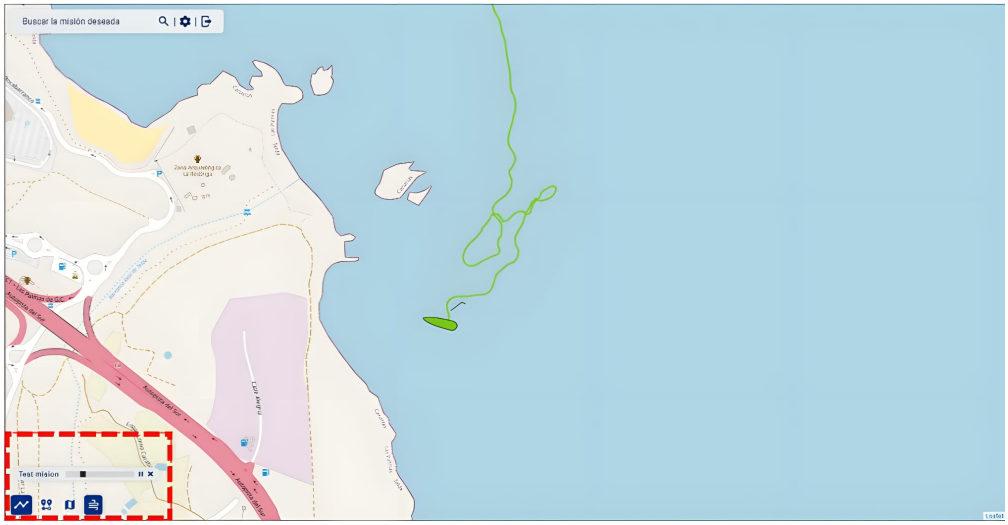
Figura 3.6: Ejemplo de una consulta SQL avanzada utilizada para la recuperación de datos de las misiones

3.4.4. Mejora en la Reproducción de Misiones

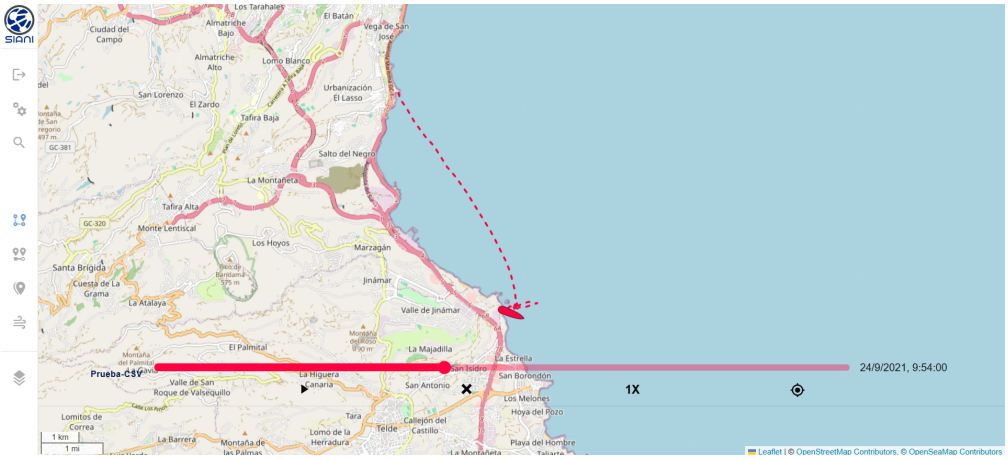
Se ha realizado una mejora significativa en la reproducción de las misiones, abordando tanto aspectos visuales como de eficiencia. En términos visuales, se ha optimizado la representación gráfica de las misiones para proporcionar una experiencia más agradable y detallada. Esto incluye mejoras en la barra de reproducción 3.7 y la precisión en la visualización de datos, lo que permite una representación más precisa de las rutas y ubicaciones de los vehículos en la aplicación.

Además de las mejoras visuales, se ha trabajado en mejorar la eficiencia del proceso de reproducción de las misiones. Se han implementado técnicas de optimización de código y de acceso a datos para reducir los tiempos de carga y procesamiento, lo que se traduce en una reproducción más fluida y rápida de las misiones almacenadas. Estas mejoras no solo benefician la experiencia del usuario al proporcionar una reproducción más ágil y precisa de las misiones, sino que también optimizan el rendimiento general del sistema, garantizando un uso eficiente de los recursos y una mayor escalabilidad para futuras expansiones del proyecto.

.



(a) Misión reproduciéndose en la versión original. Se resaltan los controles de reproducción en la esquina inferior izquierda.



(b) Misión reproduciéndose en la versión actual

Figura 3.7: Muestra de mejora de reproducción de misiones

3.5. Base de datos

La estructura de la base de datos y las relaciones entre sus tablas constituyen el esqueleto fundamental sobre el cual se construye todo el sistema de almacenamiento y gestión de datos. En esta sección, nos centraremos en el diseño detallado de la base de datos utilizada para almacenar la información relacionada con las misiones de navegación de vehículos de la División de Robótica y Oceanografía Computacional del Instituto Universitario SIANI.

Exploraremos en detalle las diversas tablas que componen la base de datos, identificando sus campos clave, su propósito y su relación con otras tablas en el esquema.

Es importante destacar que para la implementación de esta base de datos, se ha optado por utilizar MariaDB [31], una poderosa y versátil base de datos relacional de código abierto. Esta elección se basa en la compatibilidad con MySQL [52], su capacidad para manejar grandes volúmenes de datos y su robusta seguridad y escalabilidad, lo que la convierte en una opción ideal para el almacenamiento de datos críticos en este proyecto.

3.5.1. Tablas

A continuación se explican algunas de las tablas más importantes dentro del proyecto:

- **Tabla *missions***

  La tabla *missions*(Figura 3.8) almacena información sobre las misiones en el sistema. Sus campos principales son:

  id: Identificador único de cada misión.
  owner: Identificador del propietario de la misión.
  name: Nombre de la misión.
  description: Descripción de la misión.
  location: Ubicación de la misión.
  start_date: Fecha y hora de inicio de la misión.
  end_date: Fecha y hora de finalización de la misión.
  is_public: Indicador de si la misión es pública o no.
  active_mission: Indicador de si la misión está activa o no.

id	owner	name	description	location	start_date	end_date	is_public	active_mission
5	1	Mision	Probando	Las Palmas	2021-09-24 07:38:33	2021-09-24 13:04:15	1	0

Figura 3.8: Ejemplo de la tabla *missions*

- **Tabla *mission_permission***

  La tabla *mission_permissions* (Figura 3.9) establece las relaciones entre las misiones y los permisos asignados a los usuarios para esas misiones. Sus campos principales son:

id : Identificador único de cada relación.
mission_id : Identificador de la misión asociada.
user_id: Identificador del usuario asociado.
permission: Permiso asignado al usuario para la misión.

id	mission_id	user_id	permission
15	5	7	Full access

Figura 3.9: Ejemplo de la tabla *mission_permission*

- **Tabla *mission_vehicle***

  La tabla *mission_vehicle* (Figura 3.10) establece las relaciones entre las misiones y los vehículos asignados a esas misiones. Sus campos principales son:

  id : Identificador único de cada relación.
  mission_id : Identificador de la misión asociada.
  vehicle_id: Identificador del vehículo asociado.

id	mission_id	vehicle_id
30	7	9

Figura 3.10: Ejemplo de la tabla *mission_vehicle*

- **Tabla *permissions***

  La tabla *permissions* (Figura 3.11) almacena los permisos asociados a diferentes funciones en el sistema. Sus campos principales son:

  id : Identificador único de cada permiso.
  name : Nombre descriptivo del permiso.
  type: Tipo de permiso, que categoriza la función asociada al permiso.

id	name	type
1	show_missions	missions

Figura 3.11: Ejemplo de la tabla *permissions*

- **Tabla *permission_role***

La tabla *permission_role* establece las relaciones entre los permisos y los roles en el sistema. Sus campos principales son:

id : Identificador único de cada relación.
permission_id : Identificador del permiso asociado.
role_id: Identificador del rol asociado.

- **Tabla *records***

  La tabla *records* (Figura 3.12) almacena los datos recopilados por los sensores durante una misión. Sus campos principales son:

  id : Identificador único de cada registro de datos.
  sensor_id : Identificador del sensor que recopiló los datos.
  mission_id : Identificador de la misión asociada a los datos.
  timestamp: Marca de tiempo que indica cuándo se recopilaron los datos.
  data_source: Fuente de los datos, por ejemplo, si fueron recopilados desde un archivo CSV o fueron recibidos por telemetría.

id	sensor_id	mission_id	timestamp	data_source
39089	39090	5	2021-09-24 07:38:33	csv

Figura 3.12: Ejemplo de la tabla *records*

- **Tabla *roles***

  La tabla *roles* (Figura 3.13) almacena los diferentes roles que pueden tener los usuarios en el sistema. Sus campos principales son:

  id: Identificador único de cada rol.
  name: Nombre del rol, usado internamente en el sistema.
  display_name : Nombre del rol mostrado a los usuarios.

id	name	display_name
1	admin	Administrador

Figura 3.13: Ejemplo de la tabla *roles*

- **Tabla *sensor_navigation***

  La tabla *sensor_navigation* almacena datos de navegación recopilados por sensores. Sus campos principales son:

  id : Identificador único de cada registro de datos.
  boat_time : Marca de tiempo específica del barco en la que se recopilaron los datos.

reset_number : Número de reinicio del sistema.
channel : Canal de comunicación utilizado para recopilar los datos.
raw_data : Datos sin procesar recopilados por el sensor.
latitude : Latitud de la ubicación registrada.
longitude : Longitud de la ubicación registrada.
speed : Velocidad del barco en el momento de la medición.
heading : Dirección en la que se dirige el barco.
pitch : Ángulo de inclinación del barco en el eje longitudinal.
roll : Ángulo de inclinación del barco en el eje transversal.
true_wind_angle : Ángulo del viento real.
true_wind_speed : Velocidad del viento real.
active_wp : Punto de referencia activo actualmente.
active_wp_latitude : Latitud del punto de referencia activo.
active_wp_longitude : Longitud del punto de referencia activo.
wasp_latitude : Latitud de la ubicación WASP.
wasp_longitude : Longitud de la ubicación WASP.

- **Tabla *users*** La tabla *users* (Figura 3.14) almacena la información esencial de los usuarios del sistema. Sus campos principales son:

  id: Identificador único de cada usuario.
  name: Nombre del usuario, con índice para búsquedas rápidas.
  email : Dirección de correo electrónico del usuario, también indexada.
  password: Contraseña encriptada del usuario.
  role_id : Identificador del rol asociado al usuario, indexado para consultas eficientes.

id	name	email	password	role_id
1	usuario 1	test@test.com	$2b$10$CfxtwEqgP1CAV5lBvE6/UeYGxtuPD62v7jWUWqsF.hl...	1

Figura 3.14: Ejemplo de la tabla *users*

- **Tabla *vehicles***

  La tabla *vehicles* almacena información sobre los vehículos utilizados en el sistema. Sus campos principales son:

  id : Identificador único de cada vehículo.
  name : Nombre del vehículo.
  boat_mark_color : Color de marcador del barco.
  boat_waypoint_color : Color de los waypoints del barco.
  mac : Dirección MAC del vehículo.

- **Tabla *waypoints***

  La tabla *waypoints* (Figura 3.15) almacena información sobre los puntos de ruta definidos en las misiones. Sus campos principales son:

  id : Identificador único de cada paquete de puntos de ruta.

mission_id : Identificador de la misión asociada al paquete de puntos de ruta.
nodes : Lista de nodos que componen el paquete de puntos de ruta.
latitudes : Lista de latitudes de los nodos del paquete de puntos de ruta.
longitudes : Lista de longitudes de los nodos del paquete de puntos de ruta.
is_cyclic : Indicador de si el paquete de puntos de ruta es cíclico o no.
timestamp : Marca de tiempo de cuando se creó el paquete de puntos de ruta.

id	mission_id	nodes	latitudes	longitudes	is_cyclic	timestamp
10	7	[0,1,2,3,4,5]	[27.986999512,28.085300446,28.009000778,27.8633995...	[-15.369999886,-15.36439991,-15.32320118,-15.31630...	1	2024-05-28 19:45:50

Figura 3.15: Ejemplo de la tabla *waypoints*

3.5.2. Relaciones de tablas

Este esquema de base de datos está diseñado para gestionar y monitorear misiones de vehículos, incluyendo la asignación de permisos a usuarios, la gestión de roles y permisos, y el seguimiento detallado de los datos de navegación. La flexibilidad en las relaciones permite un manejo eficiente y seguro de la información. En la Figura 3.16 se muestra la estructura de la base de datos, y a continuación se explicará cada una de las relaciones existentes de esta estructura.

1. **Usuarios y roles**
- **Tablas involucradas**: ‘users‘, ‘roles‘, ‘permission_role‘
- **Descripción**: Esta relación define el rol que tiene cada usuario en el sistema. Los roles están definidos en la tabla ‘roles‘, y cada usuario tiene asignado un rol específico.

2. **Roles y permisos**
- **Tablas involucradas**: ‘roles‘, ‘permissions‘, ‘permission_role‘
- **Descripción**: Los permisos definen las acciones que pueden realizar los roles. La tabla ‘permission_role‘ conecta roles con permisos, permitiendo una configuración flexible de los derechos de acceso.

3. **Misiones y vehículos**
- **Tablas involucradas**: ‘missions‘, ‘vehicles‘, ‘mission_vehicle‘
- **Descripción**: Esta relación permite asignar varios vehículos a una misión específica, lo que es útil para coordinar operaciones donde se utilizan múltiples vehículos.

4. **Misiones y usuarios**
- **Tablas involucradas**: ‘missions‘, ‘users‘, ‘missions_permissions‘
- **Descripción**: Esta relación permite asignar permisos específicos a los usuarios sobre misiones particulares. Por ejemplo, un usuario puede tener permisos de ”vista.º .ªcceso completo.ª una misión.

5. **Misiones y puntos de paso**
- **Tablas involucradas**: ‘missions‘, ‘waypoints‘
- **Descripción**: Los puntos de paso (‘waypoints‘) definen el camino o trayectoria que deben seguir los vehículos durante una misión. Cada misión puede tener un conjunto de puntos de ruta asociados.

6. **Sensores y navegación**
- **Tablas involucradas**: ‘sensor_navigation‘, ‘records‘
- **Descripción**: Esta relación registra los datos de navegación recopilados por los sensores durante una misión. Cada registro contiene información detallada sobre la posición y movimiento del sensor.

7. **Misiones y registros de navegación**
- **Tablas involucradas**: ‘missions‘, ‘records‘
- **Descripción**: Esta relación documenta los datos de navegación específicos de cada misión. Permite almacenar y analizar el comportamiento de los vehículos durante la misión.

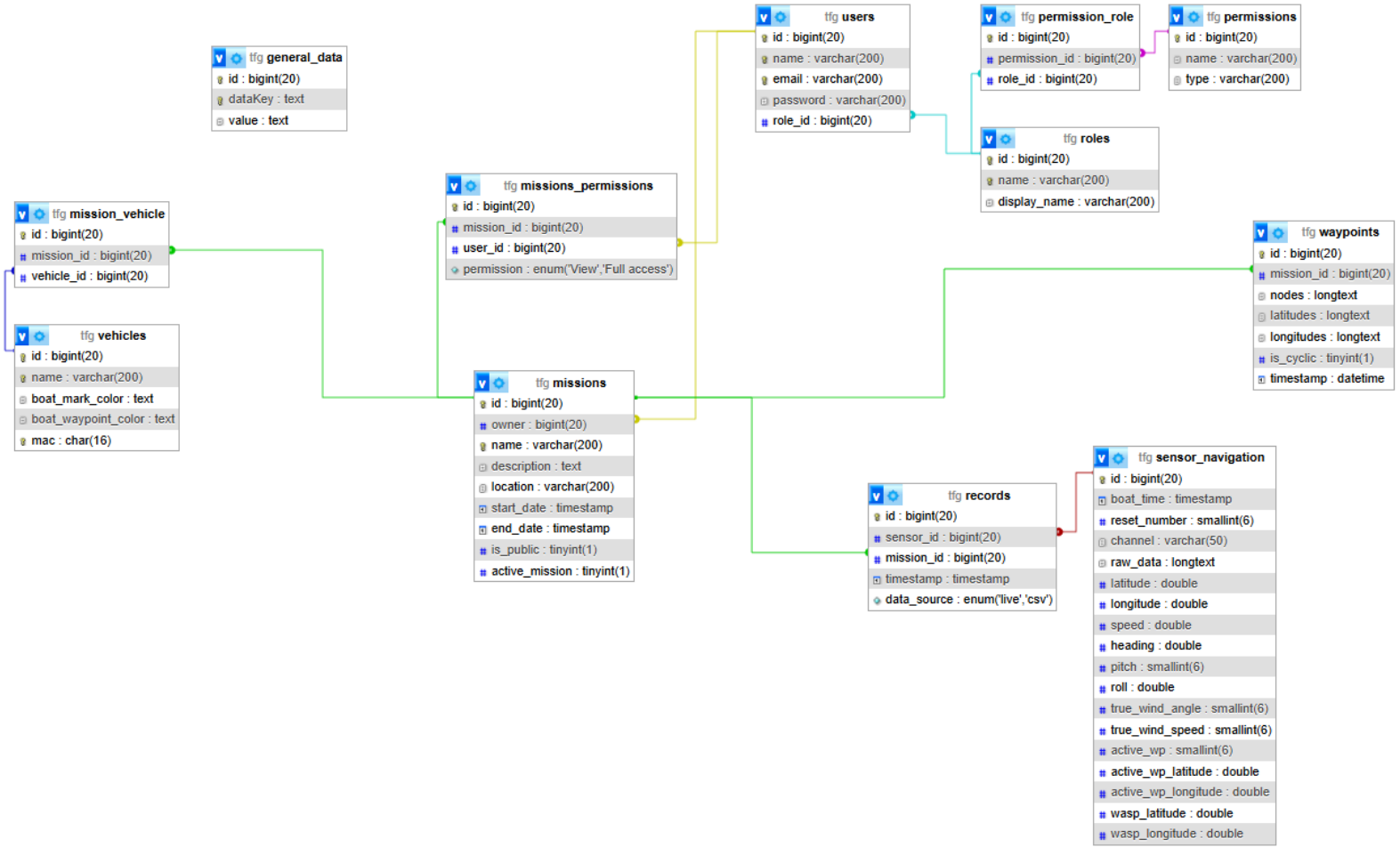


Figura 3.16: Estructura de la base de datos

3.6. Reproducción de misiones.

La función principal de esta aplicación web es la reproducción de misiones, ya sean misiones en tiempo real o misiones guardadas, a partir de registros de tiempo real o mediante un archivo *CSV*. En esta sección, se describirá esta funcionalidad y los componentes que la integran.

3.6.1. Funcionalidad de la aplicación web de reproducción de misiones

Reproducción de misiones en tiempo real:

La aplicación web permite la reproducción de misiones en tiempo real, lo que implica que los datos de la misión se transmiten y se muestran al usuario a medida que se generan. Esta funcionalidad es esencial para la monitorización de misiones en vivo, permitiendo a los usuarios observar y reaccionar a los eventos a medida que ocurren. (Para mayor detalle véase la sección 3.8)

- **Captura y transmisión de datos en tiempo real**: La información de telemetría de los vehículos durante una misión de navegación se reciben mediante una solocitud POST utilizando el protocolo HTTP. Esta se detalla a continuación.
  - **Solicitud POST para datos de telemetría**
    - **Endpoint :**
      POST /live/telemetry
    - **Cabeceras (Headers):**
      1. **Content-Type:** application/json
      2. **mac:** Dirección MAC del vehículo
    - **Cuerpo de la Solicitud (Request Body):**
      El cuerpo de la solicitud debe contener los datos de telemetría en formato JSON. En la Figura 3.17 se muestra un ejemplo de estos datos.
  - **Solicitud POST para datos de waypoints**
    - **Endpoint :**
      POST /live/waypoints
    - **Cabeceras (Headers):**
      1. **Content-Type:** application/json
      2. **mac:** Dirección MAC del vehículo

```
{
    "boat_time": "2024-06-29T12:34:56Z",
    "reset_number": 1,
    "channel": "channel_1",
    "raw_data": "example_raw_data",
    "latitude": 34.123456,
    "longitude": -118.123456,
    "speed": 12.34,
    "heading": 123.45,
    "pitch": 10,
    "roll": 1.23,
    "true_wind_angle": 45,
    "true_wind_speed": 15,
    "active_wp": 1,
    "active_wp_latitude": 34.654321,
    "active_wp_longitude": -118.654321,
    "wasp_latitude": 34.789012,
    "wasp_longitude": -118.789012
}
```


Figura 3.17: Ejemplo de la estructura de los datos de telemetría

- **Cuerpo de la Solicitud (Request Body):**
  El cuerpo de la solicitud debe contener el paquete con los datos de waypoints en formato JSON. En la Figura 3.18 se muestra un ejemplo de estos datos.


```
{
    "mission_id": 123,
    "nodes": "1,2,3,4,5",
    "latitudes": "34.123456,34.223456,34.323456,34.423456,34.523456",
    "longitudes": "-118.123456,-118.223456,-118.323456,-118.423456,-118.523456",
    "is_cyclic": 1,
    "timestamp": "2024-06-29T12:34:56Z"
}
```


Figura 3.18: Ejemplo de la estructura de los datos de waypoints

- **Visualización en Tiempo Real**: La interfaz de usuario del frontend muestra los datos recibidos casi instantáneamente, proporcionando una vista actualizada del estado de la misión.
- **Polling**: Para mantener los datos actualizados, la aplicación usa la técnica polling, donde el cliente envía solicitudes periódicas al servidor para obtener los datos más recientes.

Reproducción de misiones guardadas:

Además de la reproducción en tiempo real, la aplicación también permite a los usuarios reproducir misiones guardadas. Estas misiones pueden haberse registrado durante una misión en tiempo real o importado mediante un archivo CSV.

- **Registros de Tiempo Real:** Cuando una misión en tiempo real termina, los datos recopilados se almacenan en una base de datos. Estos datos pueden ser reproducidos posteriormente para análisis o revisión.
- **Importación de Archivos CSV:** Los usuarios pueden añadir registros a una misión utilizando archivos CSV. Este formato de archivo permite a los usuarios cargar grandes cantidades de datos estructurados de manera eficiente. Estos archivos se generan a partir de los registros de navegación de los vehículos que se almacenan localmente en memoria secundaria a bordo del propio vehículo durante el desarrollo de la misión. Estos ficheros de registro tienen una mayor densidad temporal de información, y a partir de ellos se generan ficheros en formato CSV para importar estos datos de registro a bordo en la infraestructura web.
- **Selección de Datos:** Los usuarios pueden seleccionar qué tipo de datos desean reproducir: datos históricos o datos en tiempo real, permitiendo flexibilidad en el análisis y la visualización de las misiones.

3.6.2. Buffering

Debido a la prolongada duración y la voluminosa cantidad de datos que algunas misiones generan, resulta fundamental implementar estrategias eficientes para su gestión dentro de la aplicación web. Para abordar este desafío, el sistema utiliza un enfoque de buffering. Esto implica que la aplicación solicita y almacena localmente secciones específicas de la misión conforme son necesarias para su visualización o análisis inmediato. Esta técnica no solo optimiza el rendimiento al minimizar la carga de datos, sino que también asegura que los recursos del sistema se utilicen de manera efectiva, facilitando una experiencia fluida y eficiente para los usuarios durante toda la duración de la misión.

3.7. Componentes de la reproducción de misiones

En la reproducción de misiones, diversos componentes trabajan en conjunto para proporcionar una experiencia completa y funcional al usuario. A continuación, se exploran algunos de los componentes clave que componen esta herramienta esencial para el monitoreo y análisis de misiones:

3.7.1. Barra de reproducción:

La barra de reproducción (Figura 3.19) desempeña un papel crucial al proporcionar el control sobre la reproducción de la misión. Permite a los usuarios navegar de manera intuitiva a través del tiempo y revisar eventos específicos o períodos de interés dentro de la misión. Las distintas partes de la barra de reproducción son las siguientes:

Figura 3.19: Barra de reproducción de una misión

- **Barra de progreso:** La barra de progreso es fundamental en cualquier aplicación que implique la reproducción secuencial de datos, especialmente cuando implica una representación visual, como es el caso de esta aplicación. Proporciona una clara indicación visual del avance en el proceso de reproducción, permitiendo a los usuarios controlar y navegar intuitivamente a través del tiempo.

  Para la implementación de esta se ha utilizado el componente ***v-slider*** de ***Vuetify*** como barra de progreso debido a varias razones clave:

  1. **Representación visual clara:** Era necesario mostrar de manera efectiva el progreso de las misiones dentro de la interfaz de usuario. El v-slider proporciona una representación visual clara e intuitiva del avance mediante su posición y diseño visual.

2. **Interacción intuitiva:** Para mejorar la experiencia del usuario, era importante que se pudiera interpretar fácilmente el tiempo actual de las misiones. El v-slider facilita la interacción, ofreciendo un feedback visual inmediato sobre el progreso de la misión en curso.

3. **Personalización y estilo:** Vuetify ofrece amplias opciones para personalizar el v-slider, lo que permitió ajustarlo al diseño y estilo visual específico de la aplicación. Se pudieron modificar colores, tamaño y otros aspectos visuales para integrarlo de manera coherente con la interfaz general.

4. **Integración con Vue.js:** Dado que la aplicación está basada en Vue.js, el v-slider se integra naturalmente con el ecosistema de Vue y Vuetify. Esto facilitó aprovechar características como la vinculación de datos y la gestión de eventos para actualizar dinámicamente el progreso conforme avanzaban los procesos en segundo plano.

- **Botones de Play/Pause y Stop :** Los botones de Play/Pause y Stop (Figura 3.20) son esenciales en la reproducción de misiones dentro de una aplicación web :

Figura 3.20: Botones de Play/Pause y Stop

  - **Botón de Play/Pause:** Este botón permite al usuario iniciar la reproducción de una misión o pausar la misión temporalmente. Proporciona control preciso sobre el flujo de la misión, permitiendo que los usuarios detengan y reanuden la reproducción sin perder su posición actual.
  - **Botón de Stop:** Permite al usuario detener completamente la reproducción y salir de la misión en curso. Este botón es crucial para finalizar la reproducción y liberar recursos.

- **Botón de control de velocidad :** El botón de control de velocidad permite a los usuarios ajustar la velocidad de reproducción de acuerdo a sus necesidades y preferencias. Por ejemplo, en casos donde se requiere una revisión rápida, la velocidad puede incrementarse, mientras que para un análisis detallado, la velocidad puede disminuirse.

  La aplicación permite valores de x1 (velocidad normal), x2, x4, x8, x16, x32 y x64. El control de velocidad está diseñado de manera eficiente para mantener un rendimiento óptimo incluso a velocidades elevadas. El sistema garantiza que no se perderá rendimiento sin importar la velocidad seleccionada. Es posible agregar más opciones de velocidad desde el código (Figura 3.22) sin pérdida de rendimiento, asegurando que la aplicación siga funcionando de manera eficiente.

- **Botón de centrar y seguimiento del vehículo:** El botón de centrar/seguimiento (Figura: 3.23) al vehículo es una característica esencial en la reproducción de misiones

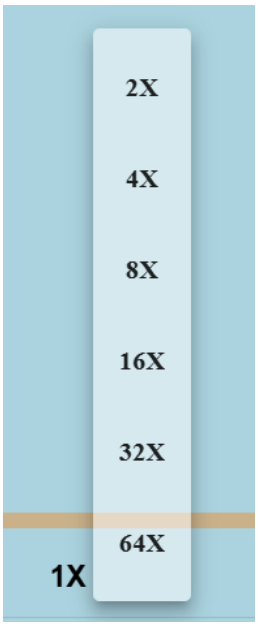


Figura 3.21: Botón y opciones de control de velocidad de reproducción de misión

```
const speedOptions = ref([1, 2, 4, 8, 16, 32,64]);
```

Figura 3.22: Constante que almacena los posibles valores de velocidad

dentro de esta aplicación web. Su implementación eficiente garantiza que los usuarios puedan mantener un enfoque constante en el vehículo durante la misión, mejorando significativamente la experiencia y usabilidad de la aplicación.

- **Centrar el vehículo:** Permite a los usuarios centrar rápidamente la vista en el vehículo, asegurando que siempre puedan visualizar su ubicación actual sin necesidad de desplazarse manualmente por la interfaz.
- **Seguimiento continuo:** Habilita el seguimiento continuo del vehículo durante la misión, manteniendo automáticamente la vista centrada en el mismo a medida que se mueve. Los usuarios pueden seguir la misión sin interrupciones, ya que el botón asegura que el vehículo siempre esté visible en la pantalla eliminando la necesidad de ajustes manuales de la vista y permitiendo a los usuarios concentrarse en el análisis de datos y la toma de decisiones.

3.7.2. Trayectoria del vehículo

La trayectoria del vehículo (Figura 3.24) es un componente esencial en la reproducción de misiones dentro de esta aplicación web. Su representación visual proporciona una serie de beneficios clave que mejoran significativamente la comprensión y el análisis de las misiones.

La visualización de la trayectoria permite a los usuarios observar el camino recorrido por

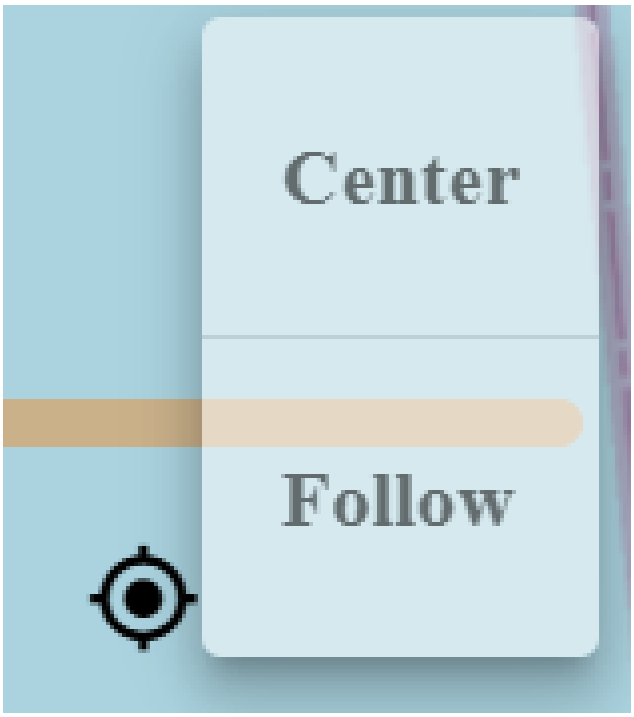


Figura 3.23: Botón de Centrar/Seguir vehículo

el vehículo tanto en tiempo real como a partir de misiones guardadas. Esto facilita el seguimiento del vehículo en el mapa, asegurando que los usuarios puedan monitorizar su progreso y posición actual en todo momento. Además, la representación de la trayectoria ayuda a identificar patrones de movimiento, comportamientos anómalos o desviaciones del curso planificado, permitiendo un análisis detallado del rendimiento del vehículo y su comportamiento en diferentes condiciones.

Para maximizar estos beneficios, se ha optimizado la representación de la trayectoria, asegurando que su cálculo sea eficiente y evitando cualquier pérdida de rendimiento. Esto garantiza una experiencia fluida y precisa para los usuarios al interactuar con la visualización de las misiones.

Debido a que el cálculo de la ruta del vehículo supone un costo computacional elevado, se ha optado por utilizar *workers*. Los workers son un concepto en programación que permite ejecutar *scripts* en segundo plano, independientemente de otros *scripts*, sin interferir con la interfaz de usuario. Se utilizan para realizar cálculos intensivos de manera eficiente y paralela, asegurando que no haya pérdidas de rendimiento y proporcionando una experiencia fluida y precisa para los usuarios al interactuar con la visualización de las misiones. El *script* que se utiliza para calcular la ruta del vehículo está representado en la *Figura 3.25*

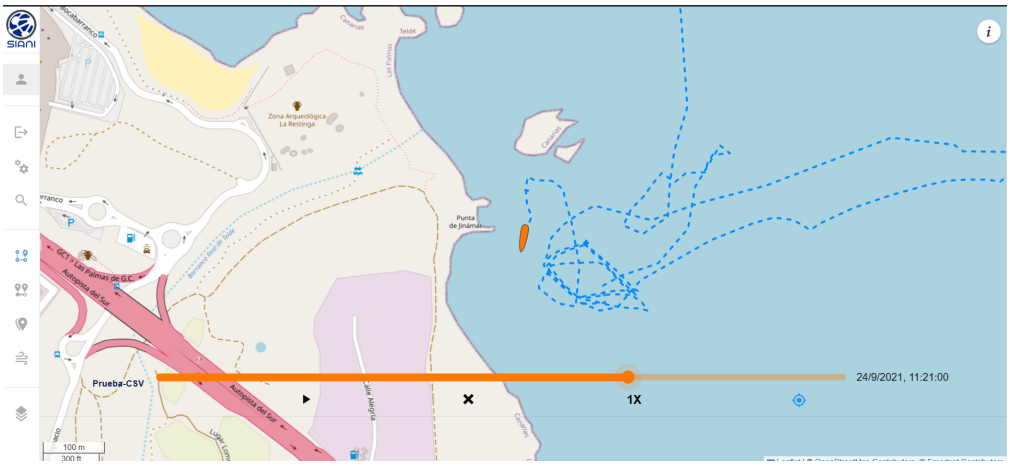


Figura 3.24: Muestra de ruta del vehículo

3.7.3. Datos recopilados de la misión

Los datos proporcionados son los típicos de la telemetría de un vehículo. Estos datos incluyen información crucial para monitorear y gestionar el estado y el rendimiento del vehículo durante una misión o actividad. Cada campo de la figura 3.26 proporciona detalles específicos:

- **Name:** Nombre del vehículo.
- **Latitude:** Coordenada de latitud actual del vehículo.
- **Longitude:** Coordenada de longitud actual del vehículo.
- **Speed:** Velocidad actual del vehículo.
- **Course:** Rumbo actual del vehículo.
- **Wind Direction:** Dirección actual del viento.
- **Wind Speed:** Velocidad actual del viento.
- **Pitch:** Ángulo de inclinación longitudinal del vehículo.
- **Roll:** Ángulo de inclinación lateral del vehículo.
- **Date/Time:** Fecha y hora en que se registraron estos datos.

Estos datos son esenciales para evaluar la situación actual del vehículo, su entorno y condiciones operativas, permitiendo a los operadores y gestores tomar decisiones informadas basadas en la información en tiempo real recopilada.

```
export const dividePolyline = (records, interval) => {
  records = JSON.parse(records);


  const max_distance = interval * 20;


  const calculateDistance = (lat1, lon1, lat2, lon2) => {
    const R = 6371e3; // Radio de la Tierra en metros
    const φ1 = (lat1 * Math.PI) / 180; // φ, latitud en radianes
    const φ2 = (lat2 * Math.PI) / 180;
    const Δφ = ((lat2 - lat1) * Math.PI) / 180;
    const Δλ = ((lon2 - lon1) * Math.PI) / 180;

    const a =
      Math.sin(Δφ / 2) * Math.sin(Δφ / 2) +
      Math.cos(φ1) * Math.cos(φ2) * Math.sin(Δλ / 2) * Math.sin(Δλ / 2);
    const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

    const distance = R * c; // Distancia en metros
    return distance;
  };
  const linesArray = [];
  let lineArray = [];
  let lastPoint = null;
  records.forEach((record) => {
    if (record === 0) {
      linesArray.push([...lineArray]);
      lineArray = [];
      return;
    }

    if (lastPoint === null) {
      lastPoint = [record.latitude, record.longitude];
      lineArray.push([...lastPoint]);
      return;
    }

    const distance = calculateDistance(
      lastPoint[0],
      lastPoint[1],
      record.latitude,
      record.longitude
    );
    if (distance > max_distance) {
      linesArray.push([...lineArray]);
      lineArray = [];
    }
    lastPoint = [record.latitude, record.longitude];
    lineArray.push([...lastPoint]);
  });

  if (lineArray.length > 0) {
    linesArray.push([...lineArray]);
  }
  return JSON.stringify(linesArray);
};
```

Figura 3.25: Script que se encarga de calcular correctamente la ruta del vehículo

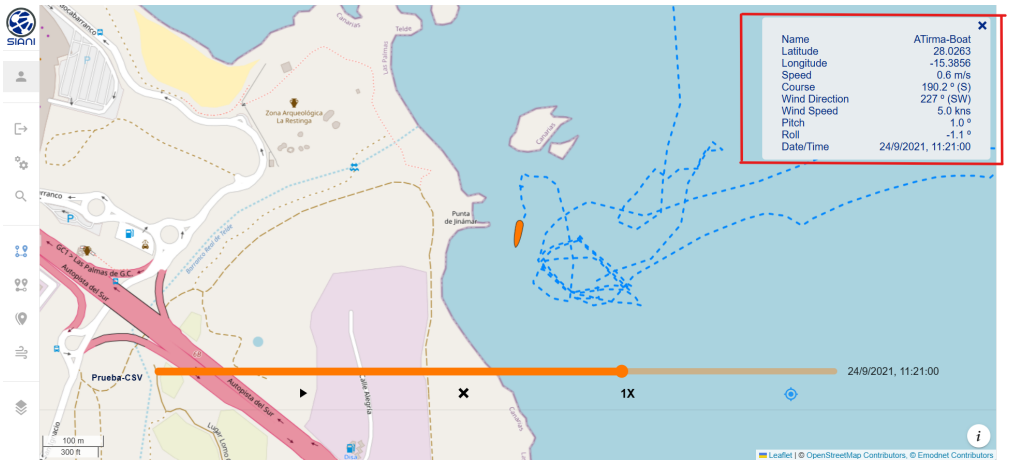


Figura 3.26: Muestra de la tabla de información que el vehículo esta recopilando datos del entorno. Se muestra haciendo click en el icono del vehículo.

3.7.4. Waypoints

Los waypoints 3.27 juegan un papel fundamental en la visualización y seguimiento de las misiones. Su representación se muestra según el momento de la reproducción, lo que ayuda a los usuarios a visualizar la ruta inmediata que el vehículo debe seguir. Esto permite un seguimiento más preciso y en tiempo real del progreso del vehículo.

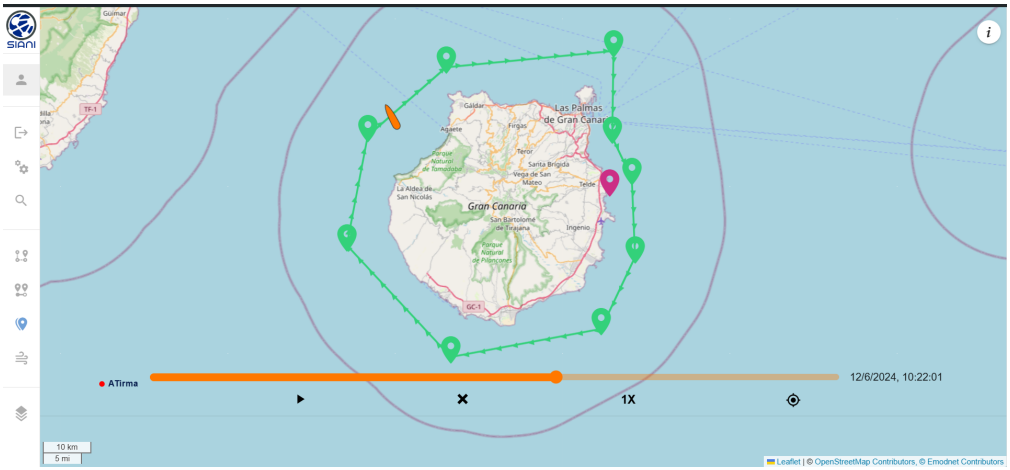


Figura 3.27: Muestra de los waypoints de una misión

Los datos de waypoints vienen en paquetes, por lo que estos paquetes deben ser procesados adecuadamente para poder mostrarlos claramente. Para lograr esto, se han utilizado workers, que permiten tratar los datos de cada paquete sin afectar el rendimiento de la aplicación. Esta técnica asegura que la visualización de los waypoints sea fluida y eficiente, proporcionando una experiencia de usuario óptima. El script utilizado para ello se muestra en la ilustración 3.28

```
export const checkWaypoints = (waypoints, current_time) => {
    const waypointsArray = JSON.parse(waypoints);
    const current_time_number = new Date(current_time).getTime();
    let resultWaypoint = waypointsArray[0];

    waypointsArray.forEach(waypointsObject => {
        const last_time = new Date(resultWaypoint.timestamp).getTime();
        const object_time = new Date(waypointsObject.timestamp).getTime();

        if(object_time <= current_time_number && object_time > last_time) {
            resultWaypoint = waypointsObject;
        }

    });

    const resultObject = {
        is_cyclic : resultWaypoint.is_cyclic,
        latLngs : []
    }

    resultWaypoint.nodes.forEach(index => {
        resultObject.latLngs.push([resultWaypoint.latitudes[index], resultWaypoint.longitudes[index]]);
    });

    if(resultObject.is_cyclic) {
        resultObject.latLngs.push(
            [resultWaypoint.latitudes[resultWaypoint.nodes[1]],
            resultWaypoint.longitudes[resultWaypoint.nodes[1]]]);
    }

    return JSON.stringify(resultObject);
}
```

Figura 3.28: Script encargado de calcular los waypoints del momento específico de la reproducción a partir del paquete de waypoints.

Además, al pasar el ratón sobre un waypoint, se despliega información detallada del mismo. Esta información incluye la posición precisa del waypoint, especificada por su latitud y longitud, y, cuando corresponde, el tipo de waypoint. Esto proporciona a los usuarios una visión más detallada y contextualizada, mejorando la comprensión y análisis de la misión. Esta funcionalidad permite una interacción intuitiva y enriquecedora, facilitando la identificación y evaluación de puntos clave en la ruta del vehículo.

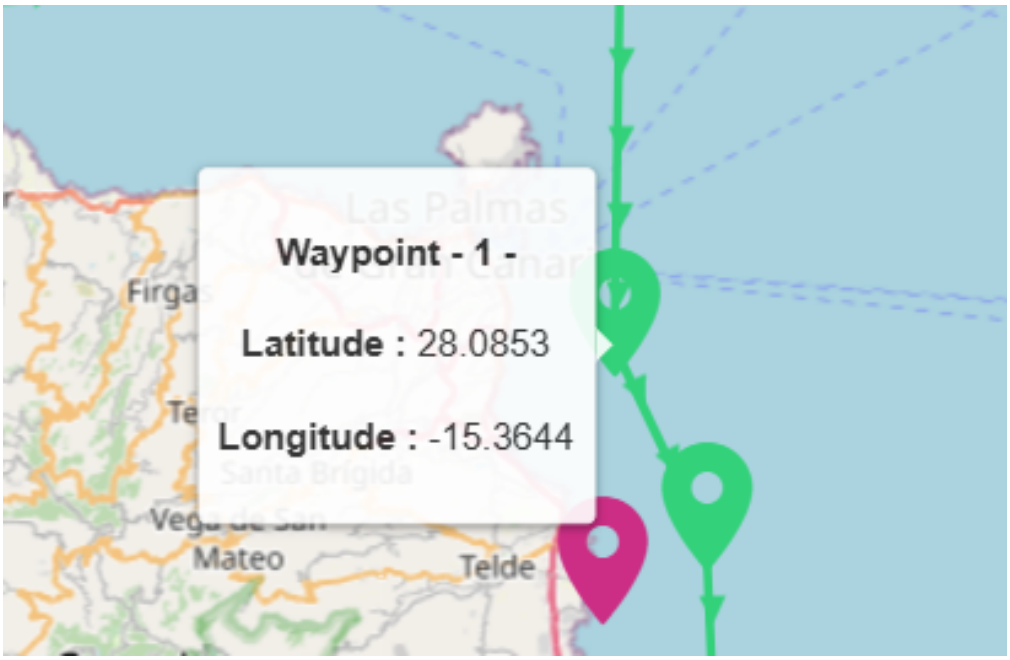


Figura 3.29: Información del waypoint 1

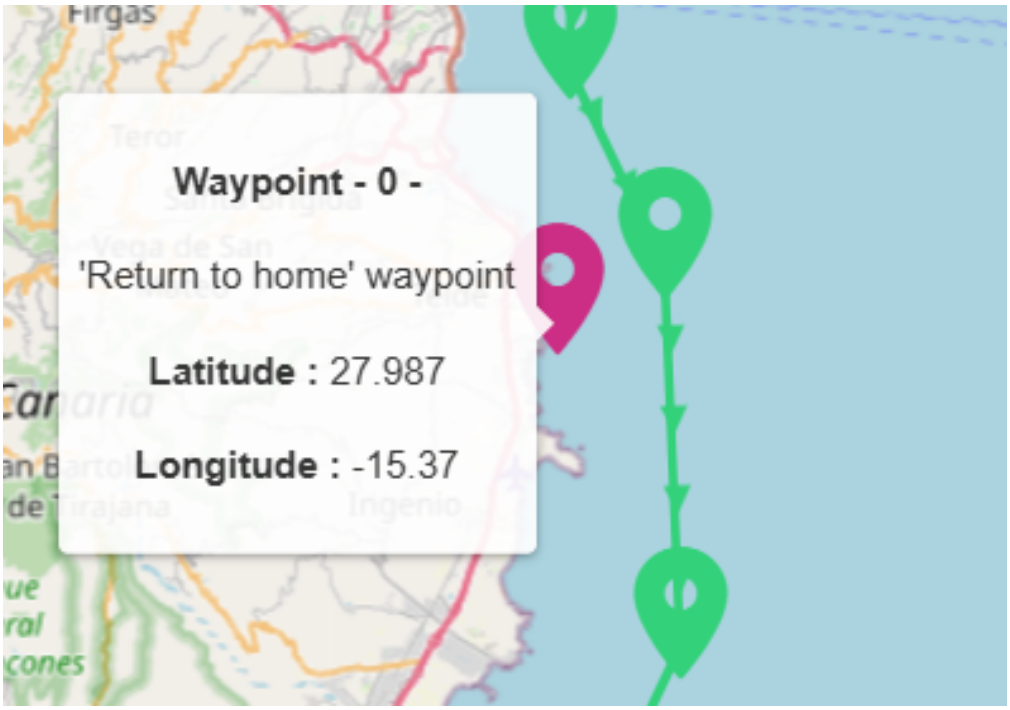


Figura 3.30: Información del waypoint 2

3.7.5. Mapa batimétrico

Como cita [3], *"La batimetría es el equivalente marino a la altimetría, es el estudio de las profundidades marinas, de la tercera dimensión de los fondos lacustres o marinos. Un mapa o carta batimétricos normalmente muestra el relieve del fondo o terreno como isogramas (curvas de nivel), y puede también dar información adicional de navegación en superficie. Su medición implica la obtención de datos con los valores de la profundidad y la posición de cada uno de los puntos muestreados. Estos puntos de posición, al igual que ocurre con la altimetría, están formados por coordenadas de puntos X, Y, Z."*

Dada la importancia de la batimetría para una misión marina, se ha incluido varias capas que ayudan al usuario a obtener información adicional sobre la ruta que sigue el vehículo:

- **Capa de isóbatas (Figura 3.31):**

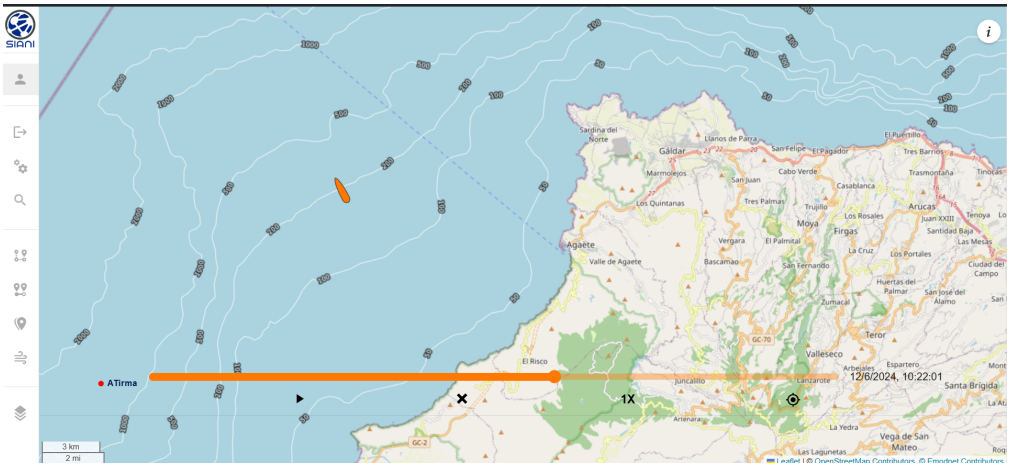


Figura 3.31: Muestra de la capa de isóbatas

- **Capa de batimetría de alta definición (Figura 3.32):**

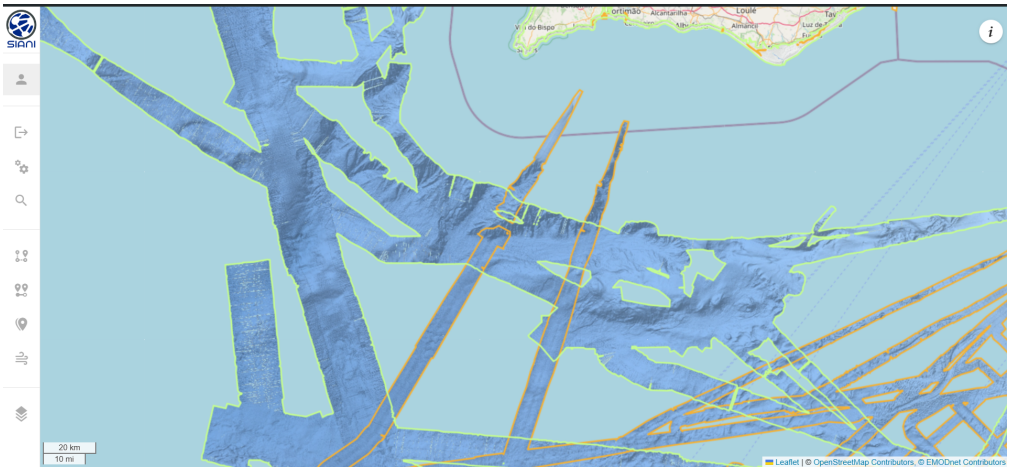


Figura 3.32: Muestra de la capa de batimetría de alta definición

- **Profundidad del fondo marino por colores (Figura 3.33):**

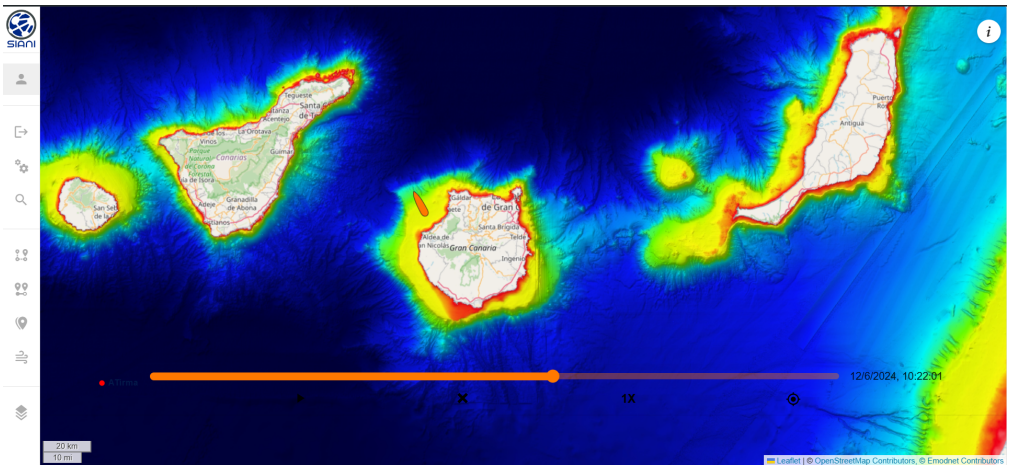


Figura 3.33: Muestra de la capa de profundidad del fondo marino por colores

- **Profundidad del fondo marino por colores del arco iris (Figura 3.34:)**

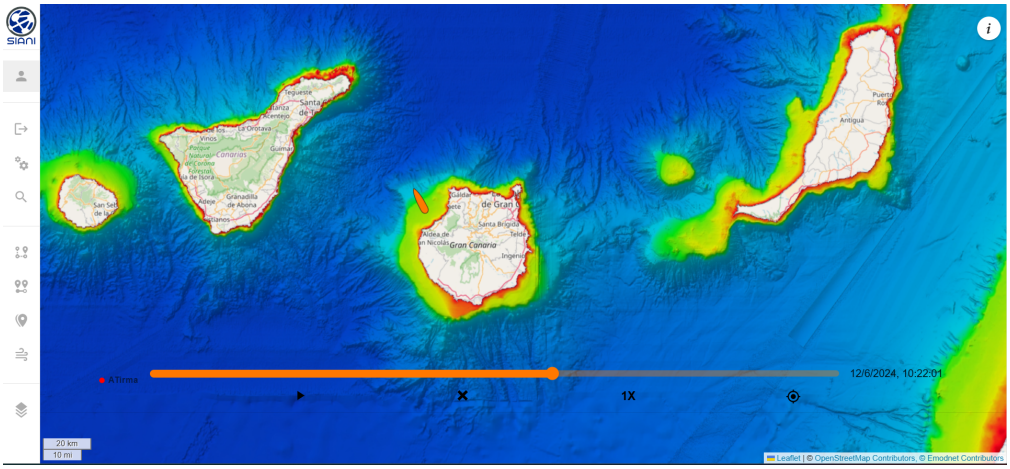


Figura 3.34: Muestra de la capa de profundidad del fondo marino por colores del arco iris

Todas estas capas se pueden combinar de manera flexible para proporcionar la información específica que el usuario necesita. Estos datos han sido recopilados y procesados por EMODnet [44], garantizando su calidad y precisión. Esta integración de capas permite a los usuarios obtener una visión completa y detallada del entorno marino, facilitando análisis más profundos y decisiones informadas basadas en datos confiables y actualizados.

3.8. Reproducción en tiempo real

La característica principal de la aplicación web del proyecto es la capacidad de monitorizar en tiempo real el seguimiento de un vehículo durante una misión en curso. Esta funcionalidad ha incrementado significativamente la utilidad de la aplicación, ya que permite a cualquier persona autorizada seguir la misión a través de la web sin necesidad de estar presente en el lugar donde se lleva a cabo. Esto no solo mejora la eficiencia y la seguridad, sino que también facilita la coordinación y la toma de decisiones, ya que los datos en tiempo real están disponibles para todos los miembros del equipo en cualquier momento y desde cualquier lugar.

Existen dos tipos de datos que cada cliente web recibe en vivo durante cada misión: datos telemétricos y datos de waypoints.

3.8.1. Restricciones de Versión de Lighttpd y Sistema Operativo: Impacto en la Implementación de WebSocket.

Durante la implementación del sistema de reproducción de misiones en tiempo real, nos enfrentamos a desafíos significativos debido a la versión del sistema operativo del servidor, la cual es de 2019, y debido a que el despliegue del servidor se realiza sobre instalaciones institucionales no fue posible la actualización de dicho sistema operativo a la versión deseada. La incapacidad de actualizar Lighttpd 3.1.5 a una versión compatible con WebSocket fue un obstáculo significativo. Esta versión específica de Lighttpd, necesaria para habilitar WebSocket, no era compatible con la versión del sistema operativo existente en el servidor. Esta limitación técnica impidió establecer la comunicación bidireccional requerida entre el servidor y los clientes de manera directa a través de WebSocket. Como resultado, nos vimos obligados a explorar soluciones alternativas que permitieran mantener la funcionalidad de reproducción en tiempo real dentro de las restricciones impuestas por el entorno de implementación. Dichas soluciones se comentan en la subsección 3.8.2.

3.8.2. Sistema de subscripción y petición de datos a misión de tiempo real.

Como se mencionó anteriormente (ver Sección 3.8.1), debido a las limitaciones con la versión de Lighttpd mencionadas, fue necesario encontrar una alternativa al uso de WebSocket para facilitar la comunicación en tiempo real entre el servidor y los clientes. La solución más prometedora y viable fue la implementación de un sistema de suscripción, que se basa en el uso del polling . En este sistema, los clientes solicitan al servidor la suscripción a misiones específicas, permitiéndoles recibir datos actualizados de manera continua y en tiempo real durante la ejecución de las misiones. Este enfoque no solo resolvió los desafíos técnicos presentes, sino que también ofreció una solución eficiente y escalable para cumplir con los requisitos de interactividad y actualización continua de datos en el contexto del sistema de

reproducción de misiones en tiempo real.

3.8.3. Funcionamiento.

Backend:

Suscripción:

Cuando un cliente quiere reproducir una misión en tiempo real, le hace una solicitud al servidor para suscribirse a dicha misión. En este proceso el cliente le proporciona al servidor el nombre de usuario y la misión a la que quiere suscribirse.

El servidor lo gestiona de la siguiente manera (Figura 3.35):

```
function subscribe(req,res,next) {
    const {user,mission_id} = req.body;
    const existingUser = telemetryData.find((value) => value.user === user);
    if(existingUser != undefined){
        if(existingUser.data.has(mission_id)){
            res.status(409).json({message : `Existing user ${user} are alredy listening mission ${mission_id}`});
            return;
        }

        existingUser.data.set(mission_id,{records : null, waypoints : null});

    }else{

        const userData = {
            user,
            data:new Map(),
            disconnectTimeout : setSubscriptionTimeout(user) };

        userData.data.set(mission_id,{records : null, waypoints : null});
        telemetryData.push(userData);
    }
    res.status(200).json('ok');
}
```

Figura 3.35: Muestra de código de la implementación de suscribirse a una misión.

1. Se desestructura la solicitud (`req.body`) para obtener `user` y `mission_id`.
2. Se busca si ya existe un usuario (`existingUser`) con el mismo nombre en `telemetryData`.
3. Si `existingUser` está definido (es decir, el usuario ya existe):
   - Se verifica si ya está suscrito a la misión (`mission_id`).
   - Si está suscrito, se devuelve un código de estado 409 Conflict.
   - Si no está suscrito, se añade la misión al mapa de datos del usuario. (Figura 3.36)
4. Si `existingUser` es `undefined` (el usuario no existe):
   - Se crea un nuevo objeto `userData` con el nombre de usuario, un nuevo mapa para almacenar datos de misiones y se establece un tiempo de desconexión.

- Se añade la misión al mapa de datos del usuario y se agrega `userData` a `telemetryData`.

5. Se responde con un código de estado 200 OK y un mensaje JSON 'ok'.

Estructura de datos:

En cuanto a la estructura para manejar las suscripciones, se utiliza un Array de objetos, en el que cada objeto pertenece a un usuario y tiene la siguiente estructura (Figura 3.36):

```
const userData = {
    user,
    data:new Map(),
    disconnectTimeout : setSubscriptionTimeout(user) };
```

Figura 3.36: Estructura de datos de usuario en el sistema de subscripción.

- El usuario
- El Mapa, el cual contiene la información nueva tanto de los datos de telemetría como de los datos de Waypoints que recibe el vehiculo en tiempo real.
- Y un Timeout, el cual está para desubscribir al usuario en caso de que el cliente no pida datos en un tiempo determinado.Esto se hace principalmente por si dicho cliente pierde la conexión antes de desubscribirse de la misión por si mismo.

Obtener datos:

Como se mencionó anteriormente, se está utilizando el método de `polling` para la obtención de datos. Esto implica que el cliente solicita los datos al servidor de manera periódica.

Por ello, el servidor gestiona la petición de datos de la siguiente manera (Figura 3.37) :

1. Se desestructura la solicitud (`req.query`) para obtener `user` y `mission_id`.
2. Se convierte `mission_id` a un número y se guarda en la variable `mission`.
3. Se busca si ya existe un usuario (`existingUser`) con el mismo nombre en `telemetryData`.
4. Si `existingUser` está definido (es decir, el usuario ya existe):
    - Se busca la misión (`mission`) en el mapa de datos del usuario (`existingUser.data`).
    - Si `data` no existe (es decir, el usuario no está suscrito a la misión):
        - Se devuelve un código de estado 404 Not Found con un mensaje JSON indicando que el usuario no está escuchando la misión.
    - Si `data` existe:

```
async function getData(req,res,next) {

    const {user,mission_id} = req.query;
    const mission = Number(mission_id)
    const existingUser = telemetryData.find((value) => value.user === user);
    if(existingUser != undefined){
        const data = existingUser.data.get(mission);
        if(!data) {
            res.status(404).json({message : `User ${user} are not listening mission ${mission}`});
            return;
        }
        clearTimeout(existingUser.disconnectTimeout);
        existingUser.disconnectTimeout = setSubscriptionTimeout(existingUser.user);
        res.status(200).json({mission_id: mission, records : data.records, waypoints : data.waypoints});
        existingUser.data.set(mission, {records : null, waypoints : null});
        return;
    }

    res.status(404).json({message : `User ${user} are not listening any mission`});
}
```

Figura 3.37: Muestra de código de la implementación de obtener datos en tiempo real de una misión

- Se limpia el temporizador de desconexión (`disconnectTimeout`) del usuario.
- Se establece un nuevo temporizador de desconexión usando `setSubscriptionTimeout` y se actualiza `existingUser.disconnectTimeout`.
- Se devuelve un código de estado 200 OK con un mensaje JSON que contiene `mission_id`, `records` y `waypoints`.
- Se actualiza la misión en el mapa de datos del usuario con `records` y `waypoints` establecidos a `null`.

5. Si `existingUser` es `undefined` (el usuario no existe):
    - Se devuelve un código de estado 404 Not Found con un mensaje JSON indicando que el usuario no está escuchando ninguna misión.

Recepción de datos de Telemetría:

Si hay una misión en curso, el servidor recibirá los datos de telemetría siempre y cuando algún usuario la haya registrado previamente. El servidor identifica al vehículo y acepta o no sus datos una vez se identifique. Los datos se reciben mediante solicitudes POST HTTP a /live/telemetry enviadas por el vehículo durante el desarrollo de una misión.

El servidor gestiona la recepción de datos de telemetría de la siguiente manera (Figura 3.38) :

1. Primero, se realiza una validación del MAC del barco llamando a una función específica para este propósito. Esta función verifica la validez del barco que envía los datos de telemetría.

```
async function telemetry(req, res, next) {
    try {
        let data = await boatMACValidation(req, res, next);
        if(!data) return;
        let result = await organizeAndInsertRecordsIntoDB({data : req.body, ...data});
        addDataIntoTelemetryDataArray(result.data, data, 'records');
        res.status(200).json(result.message);

    } catch (error) {
        console.error("Error while receive telemetry data", error);
        res.status(400).json({"message" : error});
        next(error);
    }
}
```

Figura 3.38: Muestra de código de la implementación de la recepción de datos de telemetría por parte del servidor. Es un callback vinculado a la solicitud HTTP POST /live/telemetry.

2. Si la validación del MAC del vehículo no devuelve datos válidos, la función termina inmediatamente, deteniendo el proceso.
3. Si la validación es exitosa, los datos de la solicitud se organizan y se insertan en la base de datos. Este paso asegura que los datos de telemetría recibidos estén estructurados correctamente y almacenados en la base de datos correspondiente.
4. Luego, se añade la nueva información de telemetría al Array de datos de telemetría existente. Esto mantiene actualizado el Array de datos con la información más reciente.
5. Finalmente, se envía una respuesta al cliente (El vehículo que envía los datos de telemetría) con un código de estado 200 OK y un mensaje indicando que la operación fue exitosa. Este mensaje proporciona una confirmación al cliente de que los datos de telemetría se han recibido y procesado correctamente.
6. Si ocurre un error en cualquier paso del proceso, el error es capturado en el bloque `catch`. Esto garantiza que cualquier problema durante el procesamiento sea manejado adecuadamente.
7. Se envía una respuesta al cliente con un código de estado 400 Bad Request y un mensaje de error, informando al cliente de que hubo un problema con la solicitud.
8. El error se pasa al siguiente middleware para su manejo adicional, asegurando que el flujo de errores se gestione de manera consistente en toda la aplicación.

Recepción de datos de Waypoints:

Al igual que con los datos de telemetría, el servidor recibe simultáneamente los datos de los waypoints que el vehículo transmite en tiempo real mediante solicitudes POST HTTP a /live/waypoints realizadas por el vehículo durante el desarrollo de una misión, pero estos se reciben con menos frecuencia, ya que, a diferencia de los datos de telemetría que se reciben de uno en uno, estos se reciben en paquetes, los cuales definen la ruta que debe seguir el

vehículo en un periodo determinado.

El servidor maneja la recepción de datos de Waypoints de la siguiente manera 3.39 :

```
async function waypoints (req, res, next) {
    try {
        let data = await boatMACValidation(req, res, next);
        if(!data) return;
        const result = await organizeAndInsertWaypointsIntoDB({data: req.body, ...data });
        addDataIntoTelemetryDataArray(result.data, data, 'waypoints');
        res.status(200).json(result.message);
    } catch (error) {
        console.error("Error while receive waypoints data", error);
        res.status(400).json({"message" : error});
        next(error);
    }
}
```

Figura 3.39: Muestra de código de la implementación de la recepción de datos de Waypoints por parte del servidor. Es un callback vinculado a la solicitud HTTP POST /live/waypoints

.

1. Se realiza la validación del MAC del barco mediante la función `boatMACValidation`, la cual espera obtener datos válidos.
2. Si la validación del MAC del barco no produce datos válidos, la función termina prematuramente, deteniendo cualquier procesamiento adicional.
3. Si la validación es exitosa, los puntos de ruta (*waypoints*) proporcionados en el cuerpo de la solicitud se organizan y se insertan en la base de datos mediante la función `organizeAndInsertWaypointsIntoDB`.
4. Los nuevos datos de puntos de ruta (*waypoints*) se agregan al Array de datos de telemetría existente utilizando la función `addDataIntoTelemetryDataArray`.
5. Se responde al cliente (el vehículo que realiza una petición POST HTTP) con un código de estado 200 OK y un mensaje JSON que contiene la confirmación del éxito de la operación.
6. En caso de que ocurra un error en cualquiera de los pasos anteriores, el bloque `catch` captura la excepción y maneja el error.
7. Se responde al cliente con un código de estado 400 Bad Request y un mensaje JSON que contiene detalles sobre el error que ocurrió.
8. El error capturado se pasa al siguiente middleware (`next(error)`) para un manejo adicional si es necesario, asegurando una gestión coherente de errores en la aplicación.

Frontend:

Para reproducir una misión en tiempo real, hay que seguir una serie de pasos en la aplicación :

1. Seleccionar en el menú lateral izquierdo la opción *Search*. (Figura 3.40)

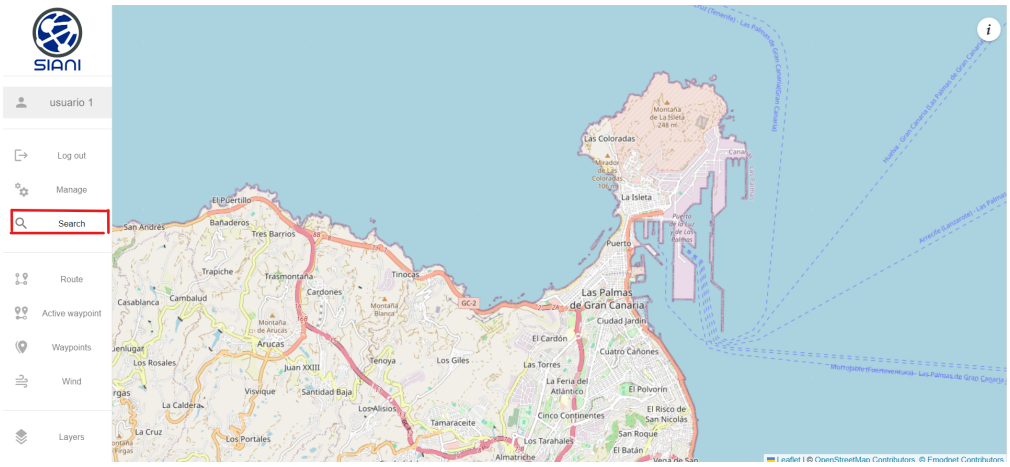


Figura 3.40: Primer paso para reproducir una misión

2. A continuación, aparecerá uno de los menús laterales derechos de la aplicación, en el que nos aparecerá una barra de búsqueda. Si le damos click, aparecerán dos opciones: ***Vehicles*** y ***Missions***. En este caso hay que seleccionar *Missions*. (Figura 3.41)

Figura 3.41: Segundo paso para reproducir una misión

3. Se mostrará la opción para elegir entre dos pestañas, ***History***, las cuales son misiones que ya han terminado y poseen datos, ya sean grabados en tiempo real cuando se reprodujo la misión, o subidos mediante un archivo *CSV*, y ***Actives***, que son las misiones

Figura 3.42: Tercer paso para reproducir una misión

que se están llevando a cabo en tiempo real. Para reproducir la misión deseada, hay que darle click al boton de ***add*** que se encuentra justo al lado de la misión.

4. Cuando se seleccione una misión, aparecerá una ventana en la cual habrá que seleccionar dos datos para la reproducción (Figura 3.43) :

   - **Intervalo en segundos**:Dado que las misiones pueden durar días, cada una puede generar miles de registros. Dependiendo del propósito en la aplicación, puede ser más eficiente para el usuario reproducir solo un porcentaje de estos registros en lugar de todos. Por ejemplo, si el usuario selecciona un intervalo de 60 segundos, la reproducción incluirá registros de la misión obtenidos cada 60 segundos, resultando en menos registros que el conjunto original. En cambio, si se selecciona un intervalo de 1 segundo (el tiempo mínimo por registro), se reproducirán todos los registros de la misión.

   - **Data type**: Este es el tipo de dato que se desea reproducir: ***History records*** o ***Live records***. Durante una misión en vivo, solo se permiten los registros provenientes de la reproducción en vivo (***Live records***). Una vez que la misión termina, y un usuario añade registros a la misión mediante un archivo *CSV*, se podrá seleccionar entre ambos tipos de registros.

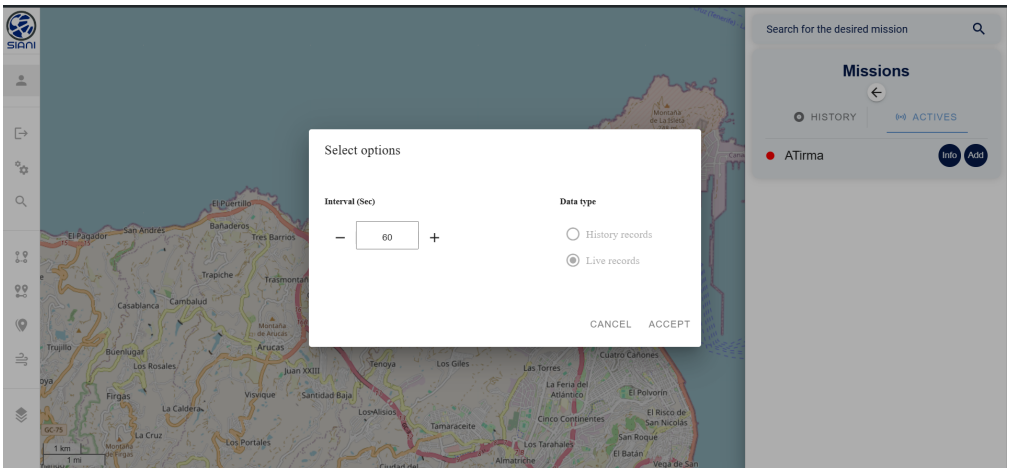


Figura 3.43: Cuarto paso para reproducir una misión

Una vez que la misión está en reproducción, el cliente y el servidor inician un proceso de polling, en el cual el cliente envía una petición al servidor a intervalos de tiempo de 3 segundos, dado que el vehículo envía datos telemétricos cada 6 segundos aproximadamente, asegurando que no se acumulan datos sin ser mostrados, hasta que uno de los dos termine la conexión.

3.9. Gestión de permisos de misiones de usuarios

En el contexto de aplicaciones web colaborativas y científicas, la gestión de permisos juega un papel fundamental para asegurar la seguridad, integridad y accesibilidad de los datos entre distintos usuarios. En el ámbito específico de la aplicación web, la gestión eficiente de permisos de misiones se convierte en un componente esencial para facilitar la colaboración y proteger la información sensible.

Esta sección aborda cómo se ha implementado un sistema de gestión de permisos dentro de la plataforma, permitiendo a los usuarios definir quién puede acceder y editar sus misiones.

3.9.1. Niveles de usuarios

Dentro del sistema de gestión de permisos de misiones de usuarios, se han establecido tres niveles distintos de acceso para garantizar la adecuada administración y seguridad de la plataforma:

1. **Usuario Público:**

   - **Descripción:** Este nivel de usuario representa a los visitantes no autenticados que acceden a la plataforma web. Tienen acceso limitado y solo pueden visualizar información general y la reproducción de misiones declaradas públicas por sus propietarios.
   - **Permisos:** Solo pueden ver misiones y datos públicos. No tienen la capacidad de editar o modificar ninguna información.

2. **Usuario Normal:**

   - **Descripción:** Los usuarios normales son aquellos que han recibido una cuenta creada por los administradores de la plataforma y tienen acceso autenticado. Generalmente, son investigadores y colaboradores internos.
   - **Permisos:** Pueden acceder y editar las misiones que ellos mismos han creado, así como aquellas a las que otros usuarios les han otorgado permisos específicos. También pueden compartir acceso a sus misiones con otros usuarios según sea necesario, como se explica en la sub-sección 3.9.2.

3. **Usuario Administrador:**

   - **Descripción:** Este nivel de usuario posee los máximos privilegios dentro de la plataforma. Son responsables de la administración y configuración global de las misiones y usuarios.

- **Permisos:** Tienen control total sobre todas las misiones y usuarios en la plataforma. Pueden crear, editar y eliminar misiones, así como gestionar cuentas de usuarios, asignar roles y definir permisos para otros usuarios.

Estos niveles de usuario aseguran que cada usuario tenga el acceso adecuado a la información y funciones necesarias para llevar a cabo sus tareas sin comprometer información.

3.9.2. Asignación de permisos a otros usuarios

Dentro del sistema de gestión de misiones, los usuarios normales tienen la capacidad de asignar permisos específicos a otros usuarios según sea necesario. Esta funcionalidad permite una colaboración efectiva y controlada entre diferentes investigadores y equipos.

A continuación se detalla cómo los usuarios pueden asignar permisos a otros usuarios dentro del sistema:

1. **Acceso a la configuración de misiones:** Los usuarios con permisos adecuados pueden acceder a la configuración de cada misión que han creado o que tienen acceso total a la misión. Para poder administrar un misión hay que seguir los siguientes pasos.

   a) En el menú lateral izquierdo seleccionar ***Manage*** (Figura 3.44), lo que desplegara el menú lateral derecho correspondiente.

Figura 3.44: Selección de administrar (*Manage*) desde el menú

   b) En el menú lateral derecho seleccionar ***Missions*** (Figura 3.45 lo que abrirá todas las misiones que el usuario puede gestionar.

   c) Seleccionar la misión se desea gestionar (Figura 3.46, lo que abrirá el panel de control de dicha misión.

2. **Gestión de permisos:** Dentro del panel de control de la misión 3.47, el usuario puede encontrar la opción para gestionar los permisos de acceso.

Figura 3.45: Selección de misiones (*Missions*) desde el menú de administrar (*Manage*)

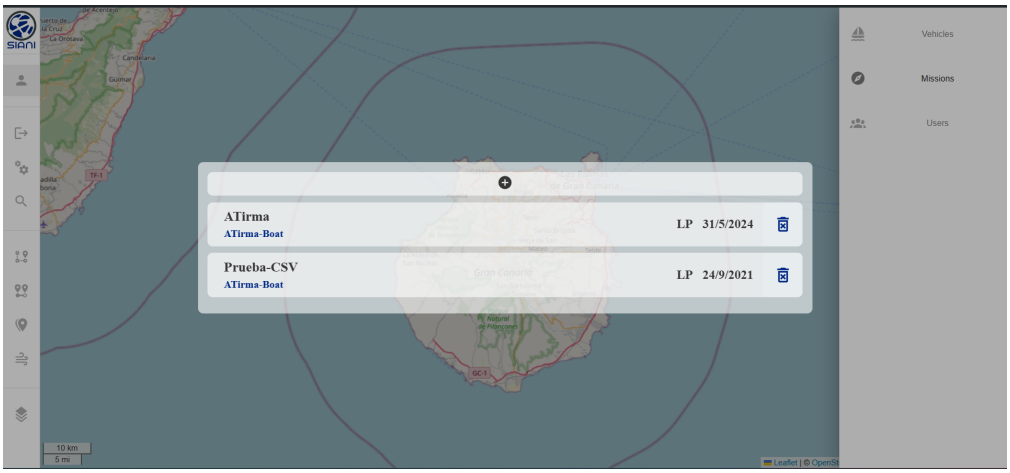


Figura 3.46: Panel de misiones

3. **Tipos de permisos disponibles:** Se pueden asignar dos niveles de permisos (Figura 3.48), según las necesidades específicas de colaboración.

   - ***View Permission :*** Este permiso permite solamente la reproducción de la misión.
   - ***Full Access :*** Este permiso permite tanto reproducir la misión como cambiar y/o añadir datos a la misión. Lo único que no permite este permiso es cambiar los permisos de la misión, ya que esta función solo está disponible para el propietario de dicha misión.

4. **Confirmación y aplicación de cambios:** Una vez seleccionados los usuarios y definidos los permisos específicos, se confirman y aplican los cambios. Los usuarios afectados podrán entonces acceder y realizar las acciones permitidas según los permisos asignados. Si no se confirman los cambios y se cambia de pantalla, aparecerá una ventana de aviso (Figura 3.49).

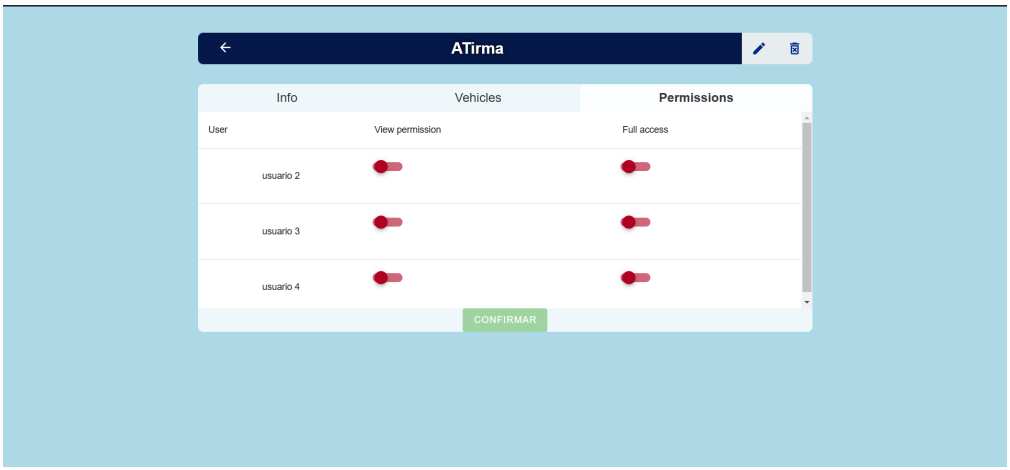


Figura 3.47: Panel de control de la misión

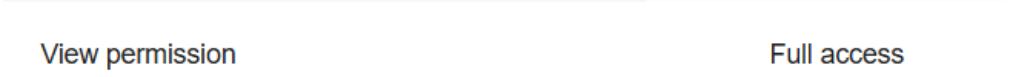


Figura 3.48: Tipos de permisos en una misión

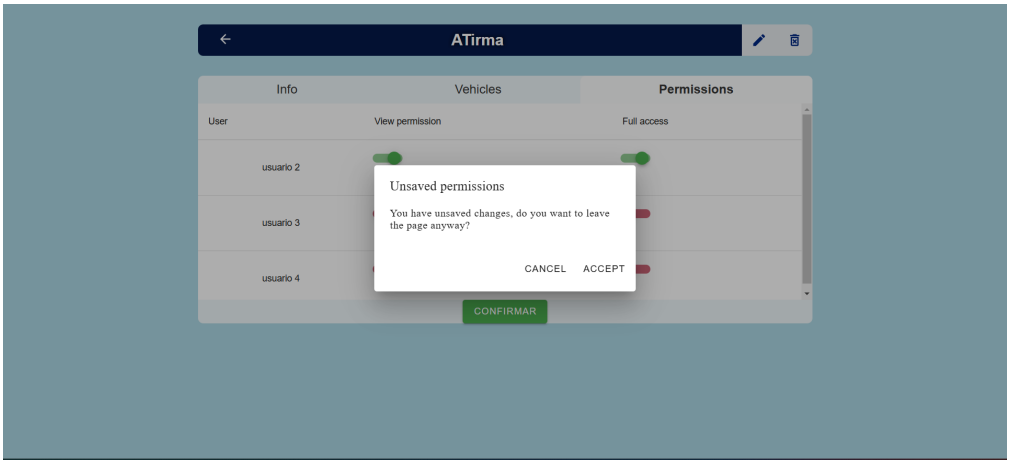


Figura 3.49: Aviso de cambios no guardados en la asignación de permisos de una misión

Capítulo 4

Conclusiones y trabajo futuro

El proyecto A-Tirma WebWare v2 ha resultado en una serie de avances significativos en diversas áreas clave, fortaleciendo tanto la funcionalidad como la usabilidad de la plataforma. En este capítulo se detallan las conclusiones específicas relacionadas con cada aspecto del proyecto, destacando las mejoras implementadas y su impacto en el rendimiento y la experiencia del usuario. Además, se presentarán las oportunidades de mejora y los trabajos futuros que podrían potenciar aún más el sistema, asegurando su adaptabilidad y eficacia en un entorno en constante evolución. Esta sección proporciona una visión global de los logros alcanzados

4.1. Conclusiones

4.1.1. Mejoras en la Interfaz de Usuario

La migración de Vue2 a Vue3, junto con la implementación de Vuetify, ha mejorado notablemente la experiencia del usuario. La interfaz ahora es más intuitiva y eficiente, facilitando la navegación y el acceso a las funcionalidades del sistema. Esto ha resultado en una mayor satisfacción del usuario y ha simplificado la interacción con los datos de las misiones de vehículos autónomos marinos.

A continuación se enumerarán algunos de los detalles mas importantes:

1. **Rediseño visual:**
   La actualización a Vue3 ha permitido aprovechar nuevas capacidades de renderizado y optimización, haciendo que la interfaz sea más dinámica y visualmente atractiva. Vuetify ha proporcionado un conjunto de componentes modernos y estilizados que se integran perfectamente, mejorando la cohesión y el diseño visual del sistema.
2. **Eficiencia en la navegación:**
   La reestructuración del flujo de navegación ha sido clave. Se han implementado rutas más claras y accesos directos a las funcionalidades , lo que reduce el tiempo y el esfuerzo

necesarios para realizar tareas comunes. Los menús y paneles ahora son más coherentes y están mejor organizados, lo que facilita la localización de funciones y herramientas.

3. **Interacción con datos en tiempo real:**
   Con las mejoras en la interfaz, la visualización y el manejo de los datos en tiempo real han sido optimizados. Los usuarios pueden ver actualizaciones instantáneas y precisas sobre el estado de las misiones, lo que es crucial para la toma de decisiones rápidas y efectivas en el ámbito de la exploración y monitoreo marino.

Estas mejoras han resultado en una interfaz de usuario que no solo es más agradable a la vista sino también más funcional y adaptada a las necesidades de los usuarios.

4.1.2. Optimización del rendimiento del sistema

La optimización del rendimiento ha sido una prioridad central en el proyecto A-Tirma WebWare v2, resultando en mejoras sustanciales en la velocidad y eficiencia del sistema. A continuación, se describen las principales optimizaciones implementadas y sus impactos.

1. **Migración a Vue3:**

   La actualización a Vue3 no solo mejoró la interfaz de usuario, sino que también optimizó el rendimiento del sistema. Vue3 incluye un nuevo motor de reactividad y mejoras en la virtualización del DOM, lo que reduce significativamente el tiempo de renderizado y mejora la capacidad de respuesta de la aplicación lo que fue esencial en esta aplicación web.

2. **Uso de Web Workers:**

   Para manejar tareas intensivas en cálculo, como el procesamiento de rutas y la gestión de datos de misiones en tiempo real, se han implementado Web Workers. Esto permite ejecutar scripts en segundo plano, distribuyendo la carga de trabajo y evitando bloqueos en la interfaz de usuario. Los Web Workers aseguran que las operaciones complejas no afecten la fluidez de la navegación ni la interacción con el sistema.

3. **Mejoras en la gestión de la base de datos:**
   Se han revisado y optimizado las consultas SQL para mejorar su eficiencia. Esto incluye la refactorización de consultas complejas, la eliminación de subconsultas innecesarias y el uso de uniones eficientes. Estas optimizaciones han resultado en una notable reducción de los tiempos de respuesta, permitiendo una interacción más rápida y eficiente con los datos. Los usuarios ahora pueden acceder a la información necesaria con mayor velocidad, lo que mejora la experiencia general del usuario y aumenta la productividad en la gestión y análisis de misiones.

4.1.3. Mejora en la colaboración

La implementación de un sistema robusto de gestión de permisos en las misiones de usuarios ha fomentado una colaboración más efectiva. A continuación, se describen los logros alcanzados y su impacto en el entorno de investigación.

1. **Niveles de usuario:**
   La introducción de diferentes niveles de usuario, como usuarios públicos, normales y administradores, permite una gestión granular del acceso. Los usuarios públicos pueden visualizar misiones públicas, los usuarios normales pueden gestionar sus propias misiones y colaborar en misiones compartidas, y los administradores tienen control total sobre todas las misiones y usuarios.
2. **Facilitación de la colaboración:** La posibilidad de compartir misiones con permisos específicos ha mejorado la colaboración entre investigadores. Los usuarios pueden trabajar conjuntamente en misiones compartidas, aportando diferentes perspectivas y conocimientos sin comprometer la seguridad de los datos.
3. **Seguridad y confidencialidad:** Al proporcionar un sistema de permisos detallado, se garantiza que solo las personas autorizadas puedan acceder a la información crítica de las misiones. Esto es esencial para mantener la confidencialidad y la integridad de los datos, especialmente en proyectos de investigación sensibles.

4.1.4. Desafíos técnicos

El proyecto A-Tirma WebWare v2 ha enfrentado y superado una serie de desafíos técnicos a lo largo de su desarrollo. La capacidad de abordar estos problemas de manera efectiva ha sido crucial para el éxito del proyecto y ha llevado a mejoras sustanciales en la funcionalidad y el rendimiento del sistema. A continuación, se describen algunos de los desafíos técnicos más significativos y, en su caso, las soluciones implementadas para superarlos.

1. **Importante proceso de refactorización:**
   - **Desafío:** Actualizar la infraestructura web existente a las versiones más recientes, optimizar la estructura de tablas de la base de datos y mejorar las consultas SQL, así como reorganizar y mejorar la interfaz de usuario para optimizar los tiempos de procesamiento de algunas operaciones.
   - **Solución:** Se realizó una actualización integral que incluyó la optimización de la estructura de tablas de la base de datos, la regranularización y reorganización de las consultas SQL, y la mejora de la interfaz de usuario. Esto implicó una refactorización del código tanto en el backend como en el frontend, siguiendo mejores prácticas de desarrollo y aprovechando las últimas características de los frameworks utilizados. Gracias a esto, se logró una mejora significativa en el rendimiento y la eficiencia del sistema.
2. **Migración de Vue2 a Vue3:**

- **Desafío:** La actualización del framework Vue.js de la versión 2 a la versión 3 presentaba varios retos, incluyendo la compatibilidad de componentes y la adaptación de la nueva sintaxis.
- **Solución:** Se llevó a cabo una planificación meticulosa para la migración, asegurando que todos los componentes fueran compatibles con Vue3. Se aprovechó la nueva API de composición para mejorar la modularidad y la reutilización del código. Además, se realizaron pruebas exhaustivas para garantizar que la funcionalidad existente no se viera afectada durante el proceso de migración.

3. **Optimización de Consultas SQL:**
   - **Desafío:** Las consultas a la base de datos SQL presentaban problemas de rendimiento debido a la gran cantidad de datos generados y almacenados durante las misiones de navegación.
   - **Solución:** Se implementaron optimizaciones en las consultas SQL para reducir el tiempo de respuesta y mejorar la eficiencia del sistema. Esto incluyó la creación de índices adecuados, la reestructuración de algunas consultas complejas y la implementación de mecanismos de buffering en reproducciones en tiempo real para reducir la carga en la base de datos. Estas optimizaciones resultaron en una mejora significativa en el rendimiento del sistema y una experiencia de usuario más fluida.
4. **Tratamiento de datos en tiempo real:**
   - **Desafío:** La necesidad de procesar y mostrar datos de misiones en tiempo real requería soluciones eficientes para manejar grandes volúmenes de datos sin comprometer el rendimiento del sistema.
   - **Solución:** Se implementó un sistema de polling en lugar de WebSockets, que hubiera sido lo óptimo, para la transmisión de datos en tiempo real, debido a la incompatibilidad con el servidor. Además, se utilizaron workers para el procesamiento paralelo de datos intensivos. Estas tecnologías aseguraron una actualización rápida y eficiente de la información mostrada, garantizando que los usuarios tuvieran acceso a datos precisos y actualizados en todo momento.

En conclusión, la superación de estos desafíos técnicos ha sido fundamental para el éxito del proyecto A-Tirma WebWare v2. Las soluciones implementadas no solo han mejorado el rendimiento y la funcionalidad del sistema, sino que también han sentado las bases para futuras mejoras y expansiones del proyecto.

4.2. Trabajo futuro

Con respecto al trabajo futuro en el proyecto A-Tirma WebWare v2, hay algunas ideas que podrían incrementar su valor y mejorar la utilidad de esta para investigaciones:

- **Actualización del sistema operativo del servidor o cambio de servidor:** Se recomienda actualizar el sistema operativo del servidor o considerar un cambio de servidor para facilitar la implementación adecuada de WebSockets. Esto permitirá una transmisión más eficiente de datos en tiempo real y mejorará la experiencia de usuario en la plataforma.
- **Ampliación de tipos de vehículos y sensores:** Integrar la capacidad de añadir más tipos de vehículos con diferentes configuraciones de sensores y datos. Esto ampliará las posibilidades de investigación y exploración submarina, permitiendo adaptar la plataforma a una variedad más amplia de proyectos y necesidades específicas.
- **Soporte para múltiples vehículos por misión:**Implementar la funcionalidad para que cada misión pueda involucrar múltiples vehículos, es decir, que exista la posibilidad de añadir vehículos extra a la misión (en el estado actual de la aplicación solo se permite añadir un vehículo por misión). Esto mejorará la capacidad de realizar operaciones coordinadas y complejas durante las misiones, facilitando un análisis más detallado y exhaustivo del entorno marino.
- **Mostrar datos mediante gráficos:** Desarrollar la posibilidad de visualizar datos recopilados durante las misiones mediante gráficos interactivos.

Estas mejoras no solo expandirán las capacidades operativas de la plataforma, sino que también asegurarán su relevancia y utilidad continua en el contexto de la investigación oceanográfica y robótica submarina.

Apéndice A

Despliegue de la infraestructura web

En este apéndice se proporcionan detalles técnicos adicionales que complementan el contenido principal del documento. La información aquí presentada se basa en el despliegue de la infraestructura web realizada en el servidor

A.1. Despliegue en el Servidor

Esta sección describe el proceso detallado para desplegar la infraestructura web desarrollada en un servidor. Se incluyen pasos específicos para la copia del repositorio, el despliegue del front con Lighttpd, y la creación del servicio del backend. Además, se abordan consideraciones sobre la configuración del entorno y la optimización del rendimiento.

A.1.1. Copia del Repositorio

Para comenzar con el despliegue, primero se debe realizar la copia del repositorio que contiene el código fuente del proyecto. El código del proyecto está disponible en el repositorio git.iusiani.ulpgc.es del Instituto Universitario SIANI. Aunque se trata de un repositorio privado, se puede solicitar acceso enviando una solicitud a Antonio Carlos Domínguez Brito, cuyo correo electronico es antonio.dominguez@ulpgc.es.

- Clonar el repositorio:

  ```
  git clone https://repogit.iusiani.ulpgc.es/git/kgonzalez/A-Tirma-WebWare.git
  ```

- Navegar al directorio del proyecto:

  ```
  cd A-Tirma-WebWare
  ```

A.1.2. Despliegue del Frontend con Lighttpd

El siguiente paso es configurar el servidor web Lighttpd para servir el frontend de la aplicación. La aplicación web se puede acceder a través del enlace https://atirma.iusiani.ulpgc.es/.

- Instalar Lighttpd:

  `sudo apt-get install lighttpd`

- Crear los certificados correspondientes y crear archivo *.conf para añadir la configuración del servidor web:

  `sudo nano /etc/lighttpd/conf-enabled/99-atirma_proxy.conf`

  Añadir las siguientes líneas para configurar el servidor:

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

$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 = "/ruta/al/certificado/*.crt"
    ssl.pemfile = "/ruta/al/certificado/*.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 $HTTP["url"] =~ "^/live" {
            proxy.server = (
                "" => (( "host" => "127.0.0.1",
                         "port" => 3000)
```

```
                )
            )
        }
        else {
            server.document-root = "/ruta/al/repositorio/frontend/dist"
        }
    }


    url.rewrite-if-not-found = (
        "/.*" => "/index.html"
    )
}
index-file.names := ( "index.html" )
```

- Reiniciar el servicio Lighttpd para aplicar los cambios:

  ```
  sudo systemctl restart lighttpd.service
  ```

A.1.3. Creación del Servicio del Backend

Finalmente, se debe configurar el servicio del backend para que se ejecute continuamente en el servidor.

- Crear un archivo de servicio en systemd:

  ```
  sudo nano /etc/systemd/system/backend.service
  ```

- Añadir la siguiente configuración al archivo de servicio:

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

[Service]
WorkingDirectory=/usr/local/share/atirma-app
User=root
Group=www-data
ExecStart=/usr/local/bin/node /ruta/al/repositorio/backend/index.js
EnvironmentFile=/ruta/al/repositorio/backend/.env
Restart=on-failure
StandardOutput=syslog
```

```
StandardError=syslog
SyslogIdentifier=AtirmaApp-0.1
[Install]
WantedBy=multi-user.target
```

- Recargar el daemon de systemd para aplicar los cambios:

  `sudo systemctl daemon-reload`

- Iniciar el servicio del backend:

  `sudo systemctl start backend.service`

- Habilitar el servicio para que se inicie automáticamente al arrancar el sistema:

  `sudo systemctl enable backend.service`

A.1.4. Despliegue de la base de datos

En la raíz del repositorio del proyecto se incluye el archivo 'default_database.sql', el cual contiene la configuración necesaria para desplegar la base de datos.

Para desplegarla en el servidor, hay que hacer lo siguiente :

1. **Instalación de MySQL:** Para instalar MySQL, primero actualiza el sistema y luego instala el paquete 'mysql-server'.

   ```
   sudo apt-get update
   sudo apt-get install mysql-server
   ```

2. **Seguridad inicial de MySQL:** Después de la instalación, se recomienda ejecutar el script de seguridad de MySQL para ajustar algunas configuraciones y mejorar la seguridad inicialmente:

   ```
   sudo mysql_secure_installation
   ```

3. **Creación de Usuario en MySQL:** A continuación, crea un usuario específico con los privilegios necesarios para administrar la base de datos:

   ```
   sudo mysql
   CREATE USER 'nombre_usuario'@'localhost' IDENTIFIED BY 'contraseña';
   GRANT ALL PRIVILEGES ON *.* TO 'nombre_usuario'@'localhost' WITH GRANT OPTION;
   ```

```
FLUSH PRIVILEGES;
```

4. **Importación del Archivo ‘.sql‘:** Para inicializar la base de datos con la estructura y los datos necesarios, importa el archivo ‘default_database.sql‘.:

```
mysql -u nombre_usuario -p nombre_base_datos < /ruta/al/repositorio/
default_database.sql
```

Glosario

Arquitectura Reactiva Un enfoque de diseño de software que se centra en la creación de sistemas reactivos, los cuales son capaces de responder a cambios en el entorno de manera rápida y eficiente, manteniendo una alta capacidad de respuesta y una arquitectura resiliente.. 19

backend la parte de un sistema informático que gestiona y procesa datos, no visible para el usuario. 48

clientes En el contexto de las aplicaciones cliente-servidor, se refiere a un dispositivo o programa informático que solicita y consume servicios, recursos o datos ofrecidos por un servidor.. 47

Composition API La API de composición es una característica de Vue.js que permite organizar la lógica y el estado de los componentes de manera más estructurada y reutilizable. Introduce un nuevo enfoque para definir componentes que facilita la gestión de la lógica compleja y la compartimentación del estado en aplicaciones Vue.js.. 13, 19

Cross-Site Scripting Un tipo de vulnerabilidad de seguridad que se encuentra típicamente en aplicaciones web, que permite a los atacantes inyectar scripts maliciosos en contenido de sitios web de confianza. 17

CSV Acrónimo de *Comma-Separated Values* (valores separados por comas). Es un formato de archivo de texto que se utiliza para almacenar datos tabulares, como una hoja de cálculo o una base de datos. Cada línea del archivo es un registro de datos, y los campos dentro de cada registro están separados por comas.. 3, 33

DOM El **Modelo de Objetos del Documento** (*Document Object Model, DOM*) es una interfaz de programación para los documentos HTML y XML. Representa la estructura del documento como un árbol de nodos, donde cada nodo corresponde a una parte del documento, como un elemento, un atributo o un texto. Esto permite a los lenguajes de programación manipular, estructurar y dar estilo a los documentos de manera dinámica.. 13

ECMAScript ECMAScript es el estándar sobre el cual se basa JavaScript. Define la sintaxis y las reglas para el lenguaje de programación JavaScript, especificando cómo deben

interpretarse y ejecutarse los programas escritos en JavaScript. ECMAScript es mantenido por ECMA International, una organización sin ánimo de lucro que desarrolla estándares para tecnologías de la información y la comunicación.. 16

framework Una estructura de soporte en la que se puede construir software, proporcionando una base sobre la cual se pueden desarrollar aplicaciones. 2, 12, 13

frontend Parte de una aplicación o sistema con la que los usuarios interactúan directamente. El frontend se refiere típicamente a la interfaz de usuario y la experiencia del usuario, abarcando todo lo que el usuario ve y utiliza en una aplicación o sitio web.. 53

Interfaz de usuario (UI) El medio a través del cual un usuario interactúa con un sistema informático, que puede incluir elementos visuales como botones, menús y formularios, así como interacciones táctiles o gestuales en dispositivos táctiles.. 21

Material Design Un sistema de diseño desarrollado por Google que combina principios de diseño clásico con innovaciones tecnológicas y científicas, con el objetivo de crear una experiencia de usuario visualmente unificada y coherente en las plataformas y dispositivos modernos. 14, 16

Motor de Renderizado Un motor de renderizado es un componente de software que se encarga de procesar y mostrar gráficamente elementos visuales en una aplicación o sistema. Este motor interpreta y ejecuta instrucciones relacionadas con la representación de gráficos, como renderizar imágenes, aplicar efectos visuales, y gestionar la composición de elementos en una pantalla o ventana. Los motores de renderizado son fundamentales en aplicaciones gráficas, juegos, navegadores web, y otros tipos de software que requieren una representación visual eficiente y de alta calidad.. 18

Plugin Un plugin es un módulo de software que se utiliza para extender las funcionalidades de una aplicación o sistema. Los plugins permiten añadir nuevas características o modificar el comportamiento existente de una aplicación sin necesidad de modificar su código base. Los plugins son comúnmente utilizados en sistemas como navegadores web, editores de texto, gestores de contenido y aplicaciones de diseño para personalizar la experiencia del usuario y adaptar la aplicación a necesidades específicas.. 16

polling Polling o sondeo en computación hace referencia a una operación de consulta constante, generalmente hacia un dispositivo de hardware, para crear una actividad sincrónica sin el uso de interrupciones, aunque también puede suceder lo mismo para recursos de software.. 47, 55

polyfill Un polyfill es un fragmento de código JavaScript que proporciona funcionalidades modernas a navegadores que no las admiten nativamente. Los polyfills se utilizan para asegurar que una aplicación web funcione correctamente en una amplia gama de navegadores, incluso en aquellos que no admiten las últimas características de JavaScript o HTML5.. 16

script Conjunto de instrucciones o código que se ejecuta dentro de un programa o aplicación para automatizar tareas, manipular datos, o controlar la ejecución de procesos. Los scripts pueden ser ejecutados tanto en el lado del cliente (frontend) como en el lado del servidor (backend).. 39

SQL SQL (Structured Query Language) es un lenguaje de programación diseñado para administrar y manipular bases de datos relacionales. Permite realizar consultas, actualizaciones, inserciones, eliminaciones y otros tipos de operaciones en los datos almacenados en una base de datos. Es un estándar ampliamente utilizado en el desarrollo y gestión de sistemas de bases de datos relacionales.. 13

telemetría Proceso de medición y transmisión remota de datos a través de sistemas de telecomunicaciones. En el contexto de vehículos y sistemas autónomos, la telemetría se refiere a la recolección y transmisión de datos operativos y de rendimiento en tiempo real.. 50

Vuetify Vuetify es un framework de componentes para Vue.js que permite desarrollar interfaces de usuario ricas y atractivas de manera rápida y eficiente. Proporciona una biblioteca de componentes predefinidos que siguen las directrices de Material Design, facilitando la creación de aplicaciones web modernas y responsivas.. 9, 36

waypoints Puntos de referencia específicos utilizados en sistemas de navegación para definir una ruta o trayectoria, especialmente en el contexto de vehículos autónomos o sistemas de navegación GPS. Los waypoints indican ubicaciones clave a las que el vehículo debe dirigirse o seguir durante su trayecto.. 51

WebSocket Protocolo de comunicación bidireccional y full-duplex a través de un único socket TCP, diseñado para ser implementado en navegadores y servidores web. Permite una comunicación interactiva en tiempo real entre el cliente y el servidor.. 47

workers Un concepto en programación que permite ejecutar scripts en segundo plano, independientemente de otros scripts, sin interferir con la interfaz de usuario. Los workers son utilizados para realizar cálculos intensivos de manera eficiente, mejorando el rendimiento de la aplicación al realizar tareas en paralelo.. 39

Árbol de Componentes Un árbol de componentes es una estructura jerárquica utilizada en el desarrollo de software para representar la relación entre los diferentes elementos o partes de un sistema de software basado en componentes. En el contexto de las aplicaciones web y los frameworks como Vue.js o React, un árbol de componentes describe la estructura y la organización de los componentes de la interfaz de usuario (UI) de la aplicación. Cada nodo del árbol representa un componente, y los nodos hijos representan los componentes anidados dentro de otros componentes. El árbol de componentes facilita la comprensión, el diseño y la gestión de la interfaz de usuario de la aplicación, al proporcionar una vista estructurada de cómo se organizan y se relacionan los diferentes elementos de la interfaz de usuario.. 18

Bibliografía

[1] Auth0. node-jsonwebtoken. https://github.com/auth0/node-jsonwebtoken, 2023. Repositorio de GitHub.

[2] axios (Equipo de GitHub). axios. https://github.com/axios/axios, 2024. Repositorio de GitHub.

[3] Cima Canarias. Las batimetrías y su importancia. Consulta realizada el 20 jun 2024, https://cimacanarias.com/2019/02/01/las-batimetrias-y-su-importancia/, 2019. Blog.

[4] dcode (Usuario de GitHub). bcrypt.js. https://github.com/dcodeIO/bcrypt.js, 2020. Repositorio de GitHub.

[5] Andrey Sidorov (Usuario de GitHub). node-mysql2. https://github.com/sidorares/node-mysql2, 2024. Repositorio de GitHub.

[6] Antonin Rousset (Usuario de GitHub). vue-leaflet-rotatedmarker. https://github.com/AntoninRousset/vue-leaflet-rotatedmarker, 2021. Repositorio de GitHub.

[7] Austin Andrews (Usuario de GitHub). mdi/font. https://www.npmjs.com/package/@mdi/font, 2024. Repositorio de GitHub.

[8] Benjamin Becquet (Usuario de GitHub). Leaflet.rotatedmarker. https://github.com/bbecquet/Leaflet.RotatedMarker, 2018. Repositorio de GitHub.

[9] Cure53 (Usuario de GitHub). Dompurify. https://github.com/cure53/DOMPurify, 2024. Repositorio de GitHub.

[10] Denis Pushkarev (Usuario de GitHub). core-js. https://github.com/zloirock/core-js, 2024. Repositorio de GitHub.

[11] Kelektiv (Equipo de GitHub). node.bcrypt.js. https://github.com/kelektiv/node.bcrypt.js, 2023. Repositorio de GitHub.

[12] Makina Corpus (Equipo de GitHub). Leaflet.textpath. https://github.com/makinacorpus/Leaflet.TextPath, 2024. Repositorio de GitHub.

[13] Mudin Ibrahim (Usuario de GitHub). vue2-leaflet-rotatedmarker. https://github.com/mudin/vue2-leaflet-rotatedmarker, 2019. Repositorio de GitHub.

[14] Raruto (Usuario de GitHub). leaflet-rotate. https://github.com/Raruto/leaflet-rotate, 2024. Repositorio de GitHub.

[15] Rob Cresswell (Usuario de GitHub). vue-material-design-icons. https://github.com/robcresswell/vue-material-design-icons, 2024. Repositorio de GitHub.

[16] Vue Leaflet (Equipo de GitHub). vue-leaflet. https://github.com/vue-leaflet/vue-leaflet, 2023. Repositorio de GitHub.

[17] VueUse (Equipo de GitHub). vueuse. https://github.com/vueuse/vueuse, 2024. Repositorio de GitHub.

[18] Equipo de Nodejs. Introduction to node.js. Consulta realizada el 16 feb 2024, https://nodejs.org/en/learn/getting-started/introduction-to-nodejs, 2024. Document.

[19] Equipo de Vue. pinia. https://github.com/vuejs/pinia, 2024. Repositorio de GitHub.

[20] Equipo de Vue. vue-router. https://github.com/vuejs/router, 2024. Repositorio de GitHub.

[21] Equipo de Vue. Vuetify. https://github.com/vuetifyjs/vuetify, 2024. Repositorio de GitHub.

[22] Equipo de Vue.js. Página web oficial de vue2. https://v2.vuejs.org/, 2023. Sitio Web.

[23] Equipo de Vue.js. Composition api faq. https://vuejs.org/guide/extras/composition-api-faq.html, 2024. Documentación.

[24] Equipo de Vue.js. Introduction to vuejs. Consulta realizada el 15 feb 2024, https://vuejs.org/guide/introduction.html, 2024. Document.

[25] Equipo de Vue.js. Página web oficial de vuejs. https://vuejs.org/, 2024. Sitio Web.

[26] Equipo de Vue.js. Why vuetify? Consulta realizada el 15 feb 2024, https://vuetifyjs.com/en/introduction/why-vuetify/#getting-started, 2024. Document.

[27] Equipo de Vuetify. Página web oficial de vuetify. https://vuetifyjs.com/en/, 2024. Sitio Web.

[28] Colaboradores de Wikipedia. Comma-separated values. Consulta realizada el 20 jun 2024, https://en.wikipedia.org/wiki/Comma-separated_values, 2024. Articulo de Wikipedia.

[29] Colaboradores de Wikipedia. Sql. Consulta realizada el 20 jun 2024, https://es.wikipedia.org/wiki/SQL, 2024. Articulo de Wikipedia.

[30] Equipo de Vue.js. Vuex. Consulta realizada el 15 feb 2024, https://vuex.vuejs.org/„ 2023. WebSite.

[31] MariaDB Foundation. Página web oficial de mariadb. https://mariadb.org/, 2024. Sitio Web.

[32] OpenJS Foundation. Página web oficial de nodejs. https://nodejs.org/en, 2024. Sitio Web.

[33] Apache Friends. Página web oficial de xampp. Consulta realizada el 8 mar 2024, https://www.apachefriends.org/es/index.html, 2023. Sitio web.

[34] Inc. GitHub. Página web oficial de github. https://github.com/, 2024. Sitio Web.

[35] Inc. GitHub. Acerca de github y git. https://docs.github.com/es/get-started/start-your-journey/about-github-and-git, 2024. Documentación.

[36] Julio Varela Gomez. Vue 3 vs vue 2: Las nuevas características y mejoras que debes conocer. Consulta realizada el 21 feb 2024, https://es.linkedin.com/pulse/vue-3-vs-2-las-nuevas-caracter%C3%ADsticas-y-mejoras-que-varela-g%C3%B3mez, 2023. Blog.

[37] Google. Página web oficial de material design. https://m3.material.io/, 2024. Sitio Web.

[38] Antoine Goret. Vue.js: Cómo migrar un proyecto grande de vue 2 a vue 3. Consulta realizada el 21 feb 2024, https://crisp.chat/es/blog/vue-js-migrar/, 2023. Blog.

[39] StrongLoop IBM and other expressjs.com contributors. Página web oficial de express. https://expressjs.com/, 2017. Sitio Web.

[40] Emmanuel John. Pinia vs. vuex: Which state management library is best for vue? Consulta realizada el 15 feb 2024, https://blog.logrocket.com/pinia-vs-vuex/, 2022. Blog.

[41] Jan Kneschke. Página web oficial de lighttpd. https://www.lighttpd.net/, 2024. Sitio Web.

[42] Jon Lander Torres Lombraña. Infraestructura web para el seguimiento, monitorización y control de misiones para vehículos marinos autónomos a vela. Memoria de Trabajo Final de Titulo, Escuela de Ingeniería Informática, 2021.

[43] Manz. Ajax: Peticiones http. https://lenguajejs.com/javascript/peticiones-http/ajax/, 2024. Artículo.

[44] DG MARE. Página web oficial de emodnet. https://emodnet.ec.europa.eu/en, 2024. Sitio Web.

[45] MDN. Api. https://developer.mozilla.org/es/docs/Glossary/API, 2023. Documentación.

[46] MDN. Dom. https://developer.mozilla.org/es/docs/Glossary/DOM, 2023. Documentación.

[47] MDN. Html. https://developer.mozilla.org/es/docs/Web/HTML, 2023. Documentación.

[48] MDN. Javascript. https://developer.mozilla.org/es/docs/Web/JavaScript, 2023. Documentación.

[49] MDN. Css. https://developer.mozilla.org/es/docs/Web/CSS, 2024. Documentación.

[50] Dimitri Nek. What is lighttpd web server and how does it work? Consulta realizada el 8 mar 2024, https://webhostinggeeks.com/blog/what-is-lighttpd-web-server-and-how-does-it-work/, 2023. Blog.

[51] OpenAI. Chatgpt response on xampp for sql database management. Consulta realizada el 8 mar 2024, https://www.openai.com/chatgpt, 2024. ChatGPT.

[52] Oracle. Página web oficial de mysql. https://www.mysql.com/, 2024. Sitio Web.

[53] phpMyAdmin contributors. Página web oficial de phpmyadmin. https://www.phpmyadmin.net/, 2024. Sitio Web.

[54] winstonjs (Equipo de GitHub). winston. https://github.com/winstonjs/winston, 2024. Repositorio de GitHub.