UNIVERSIDAD DE CONCEPCIÓN

FACULTAD DE INGENIERÍA

DEPARTAMENTO INGENIERÍA INFORMATICA

Desarrollo e Implementación de arquitectura para producto IOT visión artificial, caso de estudio Alma Ltda.

POR

Álvaro Matamala Gutiérrez

Memoria de Título presentada a la Facultad de Ingeniería de la Universidad de Concepción para optar al título profesional de Ingeniero Civil Informático

Profesor Guía:
Geoffrey Hecht
Ingeniero Guía:
Miguel Chandía

Agosto 2024
Concepción (Chile)

Resumen

En el marco de la implementación de una nueva línea de negocios para la empresa Proyectos de Ingeniería Alma Ltda (Alma), se continuó con el trabajo previo de Alma para diseñar e implementar una arquitectura de servicios que conecte un software IOT y un frontend mediante servicios API. Con el objetivo principal de permitir tanto a usuario como administrador revisar reportes de operación y resultados entregados por el dispositivo IOT en un contexto logístico. Se presenta el desarrollo, implementación y las decisiones tomadas tanto para la escalabilidad y rendimiento del producto junto con la herramientas y documentos utilizados.

Palabras clave: escalabilidad, rendimiento, operación, administrar, seguimiento, dispositivo

Abstract

As part of the implementation of a new business line for the company *Proyectos de Ingeniería Alma Ltda* (Alma), the previous work by Alma continued to design and implement a service architecture that connects IoT software and a frontend through API services. The main objective is to allow both users and administrators to review operational reports and results provided by the IoT device in a logistics context. This document presents the development, implementation, and decisions made for the scalability and performance of the product, along with the tools and documents used.

Keywords: scalability, performance, operation, management, tracking, device

Tabla de Contenidos

Lista de Tablas

Lista de Figuras

Glosario

IOT	:	Internet of things
API	:	Application Programming Interface

Framework: También conocido como marco de trabajo, es una estructura conceptual y tecnológica que proporciona una base para el desarrollo de aplicaciones, sistemas o proyectos. Incluye herramientas, bibliotecas y convenciones las cuales facilitan y agilizan el proceso de desarrollo al ofrecer soluciones predefinidas para problemas comunes.

Interfaz: Se refiere al medio por el cual uno o más usuarios interactúan con un sistema, software o dispositivo, y diseñada para facilitar la comunicación y la interacción entre un usuario y el sistema.

Plataforma: Entorno o conjunto de herramientas que permite el desarrollo y ejecución de aplicaciones, servicios o sistemas.

Software: Es un conjunto de programas, datos y procesos que instruyen un dispositivo electrónico a realizar tareas específicas. Este incluye sistemas operativos, aplicaciones, utilidades que permiten a los usuarios interactuar con el software.

Commodities: Son bienes básicos que se utilizan como insumos en la producción de otros bienes y que pueden ser usados en el comercio o en el sector financiero, como objetos de adquisición.

Backend: Son todos los códigos ocultos que sirven para que una página web o aplicación funcione correctamente, optimizando elementos y recursos como la seguridad y privacidad en un sitio web o aplicación.

Frontend: También conocido como «desarrollo del lado del cliente» se refiere a la práctica de producir la parte frontal de un sitio web o aplicación. Esto incluye los fondos, colores, texto, animaciones o efectos.

CAPÍTULO 1: Introducción

En el marco de la implementación de una nueva línea de negocios para Alma Ltda, de ahora en adelante Alma, se continuó el trabajo de práctica para diseñar e implementar una arquitectura de servicios que permita conectar mediante algún servicio API a un software IOT, otro que permita reportar los resultados al cliente y un tercer servicio que permita llevar la trazabilidad de todos los clientes y sus procesos.

Actualmente, se encuentra en desarrollo el prototipo de software que se ejecuta de manera embebida en plataformas IOT, este software se basa en visión artificial y el prototipo va integrado a un medio de transporte durante un periodo logístico, este se encarga del reconocimiento de determinados objetos, el registro de eventos o alertas y la trazabilidad geográfica en el tiempo. Para este software se implementa un servicio que permita comunicar toda la recolección de datos de campo en las bases de datos de Alma, de ahora en adelante conocido como *alma track*.

Dentro de los desafíos a abordar desde el diseño en esta memoria, se encuentra la construcción de un servicio que permita al cliente observar los datos entregados por el software IOT durante procesos logísticos. Así, podrá monitorizar sus eventos logísticos y gestionar de alertas para la toma de decisiones oportunas, que permitan entregar valor maximizando el cuidado de su patrimonio.

Finalmente, diseñar e implementar un API que junte la comunicación de todos y proponga un diccionario de datos para seguir acoplando más servicios a todo los que se construya.

1.1 Objetivos

En este capítulo se ven los objetivos principales del desarrollo de *alma track*, propuesto y discutido entre empresa y memorista.

1.1.1 Objetivos Generales

El propósito del proyecto consiste en crear una base de datos relacional, los modelos Rest API y una interfaz que permita al usuario hacer un seguimiento de los datos entregados por el dispositivo IOT, el cual sigue en su fase de prototipo.

1.1.2 Objetivos específicos

1. Diseñar arquitectura para los tres servicios de la solución propuesta (software IOT ya existente, software Front y software Back para desarrollar en esta memoria).
2. Diseñar modelo de datos con enfoque relacional para el almacenamiento y registro de las entidades recolectadas en el software IOT que responda a la arquitectura del objetivo 1.
3. Diseñar e implementar un sistema de gestión de procesos donde cada cliente pueda listar, monitorear sus procesos y definir alertas con su respectiva distribución de contactos, que responda con la arquitectura del objetivo 1.
4. Diseñar e implementar un API que comunique con el sistema del punto 3 y el software IOT.
5. Diseñar un plan de pruebas para cada sistema de los puntos anteriores.
6. Diseñar un plan de pruebas de integración para la operación de todos los sistemas juntos.
7. Proponer e implementar un sistema de documentación para la API que permita comprender la manera correcta de consumirla y escalarla.
8. Implementar un sistema de gestión de procesos donde Alma lmtd pueda monitorear sus procesos y gestionar el flujo de usuarios, que responda como un añadido a la arquitectura del objetivo.

CAPÍTULO 2: Análisis

Esta capitulo tiene como objetivo analizar los sistemas, tanto en el ámbito de sus requerimientos funcionales como en los no funcionales. Para una mejor comprensión en las características que debe tener el software y en general una visión menos técnica.

2.1 Descripción del software

Alma track es una solución de múltiples servicios que en su conjunto tienen como objetivo la protección de activos de alto valor durante procesos logísticos para los clientes de alma, dentro de los cuales se destacan como clientes principales aquellos en rubros que producen commodities de sector primario o desarrollan productos que puedan o estén sensibles al daño o siniestro en la etapa de transporte impactando de forma negativa generando pérdidas y mermas de la producción.

La lógica de la primera versión de *Alma track*, esta ópera mediante el monitoreo continuo de uno o más dispositivos que van integrados a los vehículos de transporte de carga durante un periodo

logístico, el seguimiento de alertas generadas por el prototipo y procesadas en API's que orquestan una interfaz web donde los clientes pueden observar para tomar acciones que permitan mitigar o minimizar riesgos de su operación.

Dentro de las condiciones que maneja, esta posee una gestión manual, lo cual conlleva que un administrador por parte de alma debe ingresar a la base de datos, la empresa y los usuarios que estarán vinculados a ella, debido a la necesidad de hacer el vínculo entre el cliente y los dispositivos IOT que le serán correspondidos e instalados para el uso de los servicios de *alma track.*

Cumpliendo con la condición previa, los usuarios ingresan a la plataforma web de *alma track* y disponen de funciones como, el inicio de sesión con una contraseña automática la cual puede ser cambiada posteriormente a gusto mientras maneje sus credenciales, la visualización de los datos obtenidos de los dispositivos IOT, dentro de los cuales entran los reconocimientos, que son lecturas generadas por la inteligencia artificial del dispositivo, las coordenadas en la cual se generó y una visualización desde Google maps, las alertas y los comentarios, que pueden ser ingresados por los mismos usuarios. Además, en caso de cualquier duda o problema, tienen acceso de una opción de ayuda y soporte.

La primera versión de *alma track* puede ser accedida desde cualquier dispositivo con un navegador web, pero se recomienda fuertemente que solo se utilice en computadora.

2.2 Requerimientos

Para la descripción de los requerimientos del software es necesario especificar los tipos de usuarios y entidades con los que cuenta la plataforma:

- **Usuario cliente:** Es el actor principal al cual está enfocado el software, el cual debe ser registrado por el administrador y el cual, al momento del registro se vinculará a una empresa. Tiene la facultad de monitorear los dispositivos IOT vinculados al viaje de esa empresa.
- **Administrador:** Es el administrador del flujo de usuarios y dispositivos, se encarga de mantener la constancia de estos.
- **Dispositivo IOT:** Prototipo IOT basado en inteligencia artificial que se encarga del reconocimiento de objetos, gestión de alertas y trazabilidad geográfica para posterior envió al API.

Cabe destacar que el o los viajes representan un periodo de tiempo en el cual el dispositivo es utilizado y todos sus datos generados guardados para ser monitorizados. El inicio y el final de este periodo es determinado por el administrador.

2.2.1 Requerimientos funcionales

Para una comprensión más profunda de los requisitos del software y de los distintos tipos de entidades que lo componen, se presenta en la ***Tabla 1*** un conjunto de funciones o capacidades esenciales que deben cumplir.

Funciones de usuario	Usuario cliente	Administrador	Sistema IOT
Registrar empresa	✗	✓	✗
Registrar usuario	✗	✓	✗
Acceder a la cuenta de usuario	✓	✗	✗
Acceder a la cuenta de super usuario	✗	✓	✗
Cambiar la contraseña de usuario	✓	✗	✗
Cambiar la contraseña de admin	✗	✓	✗
Añadir dispositivos a la base de datos	✗	✓	✗
Empezar viaje	✗	✓	✗
Visualizar los detalles de un viaje activo	✓	✗	✗
Visualizar los detalles de un viaje terminado	✓	✗	✗
Añadir reconocimientos a un viaje	✗	✗	✓
Añadir alertas a un viaje	✗	✗	✓
Añadir comentarios a un viaje	✓	✗	✗

Visualizar historial de alertas	✔	✖	✖
Terminar viaje	✖	✔	✖

Tabla 1 Requerimientos funcionales y usos correspondientes.

Casos de uso:

Para representar los requerimientos funcionales y una ilustración representativa de la perspectiva de los usuarios, se puede utilizar un diagrama de casos de uso, el cual provee resumen del sistema con fácil comprensión.

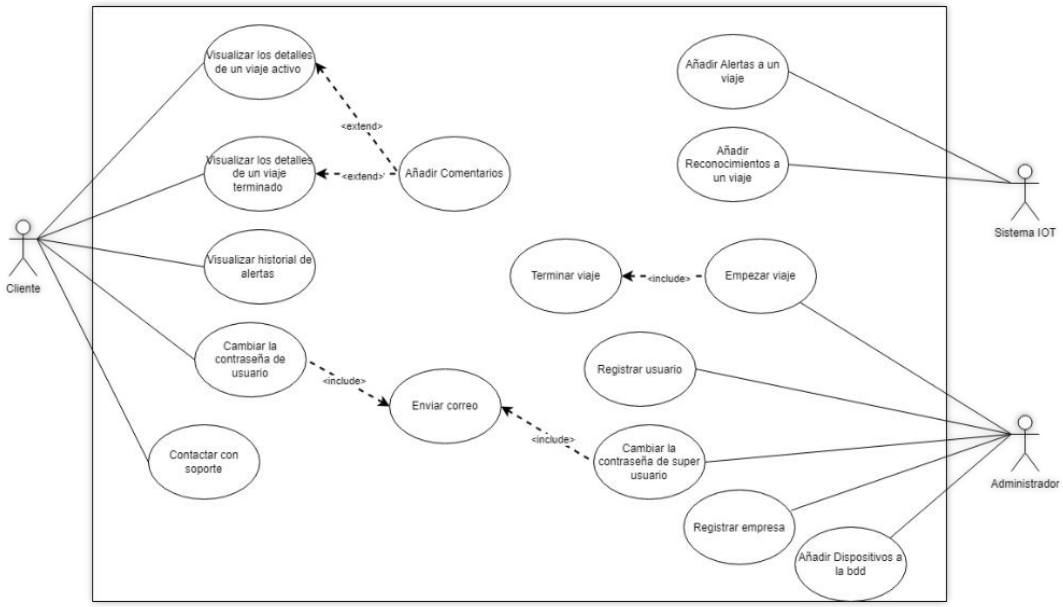

Figura 1 Casos de uso.

2.2.2 Requerimientos no funcionales

- **Rendimiento:** El sistema debe ser capaz de soportar una carga de usuarios en simultáneo sin sacrificar aspectos como el tiempo de respuesta de los sistemas. Con un tiempo de respuesta estimado de 500 milisegundos
- **Seguridad:** El sistema debe cumplir una cuota mínima de seguridad en su primera versión a través de la autenticación con credenciales únicas por token, con el objetivo de proteger los datos de los usuarios.

- **Usabilidad:** El sistema debe estar enfocado en la experiencia de usuario objetivo, con el fin de brindar una navegación intuitiva y sencilla.
- **Consistencia:** El sistema debe mantener la integridad de la información de los usuarios.
- **Escalabilidad:** El sistema debe sostener una capacidad de adaptarse a futuros cambios o integración de funciones de manera sencilla, ya que, se estima que este reciba actualizaciones y mejoras durante el tiempo.
- **Portabilidad:** El sistema debe ser utilizable por cualquier dispositivo con un navegador web

CAPÍTULO 3: Diseño y arquitectura

A la hora de decidir las herramientas para este proyecto, se discutió con la empresa cuales eran los puntos fuertes que necesitaba el desarrollo de la API, los cuales eran:

- **Rendimiento:** La herramienta utilizada debe rápida en tiempo de respuesta, ya que se estima un flujo irregular de solicitudes y debe estar preparado para estas.
- **Mantenibilidad:** Debe ser fácil de aprender y utilizar, ya que, se espera que la misma empresa se haga cargo de actualizaciones posteriores.
- **Robustez:** Debe ser seguro y constante a la hora de enviar y recibir datos sin que estos se vean afectados o sufran algún tipo de daño.

Por los puntos anteriores, se tomó como herramienta para el desarrollo de la arquitectura del proyecto a FastAPI [1].

FastAPI posee un patrón de estructura MVC (“Model-View-Controller”), el cual posee las componentes clave modelo, vista y controlador que coordinan toda la interacción y flujo de información entre con la interfaz web y la base de datos:

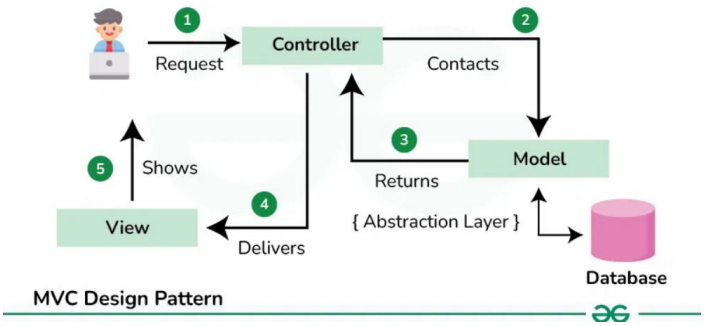

Figura 2 Diagrama de arquitectura MVC [2].

A continuación, una explicación de la función de cada componente clave del patrón de arquitectura que se puede apreciar en la ***Figura 2***:

- **Modelo:** Se utilizan modelos para definir la estructura y lógica de los datos y usuarios. Estas entidades se separan en modelos y esquemas dentro de fastAPI y están definidos como tablas en la base de datos.
- **Controlador:** Se encarga de dirigir las solicitudes HTTP y dirigirlas hacia los servicios o modelos pertinentes. En fastAPI están representados por los endpoints definidos con decoradores como "@app.get()" o "@app.post", los cuales hacen de intermediarios entre las rutas y la lógica de negocio, orquestando el flujo de datos desde y hacia el cliente
- **Vista:** Esta permite definir el diseño y estructura de la página web, se enfoca principalmente con la presentación de los datos entregados por el controlador. Utilizando herramientas como *React* y *TailwindCSS* [4] nos permite una fácil personalización del interfaz de usuario, además, también ayuda a la mantenibilidad de un código limpio y de fácil uso.

Para comprender mejor cómo interactúa FastAPI con otros sistemas y los roles que desempeña en la estructura general, en las ***Figuras 3*** y ***4*** se presentan los modelos C4 en los niveles de Contenedores y Componentes. Estos diagramas permiten visualizar tanto las interacciones de FastAPI con aplicaciones, bases de datos y servicios externos, como los componentes internos clave de cada contenedor, facilitando así una comprensión clara de la integración y funcionamiento del sistema.

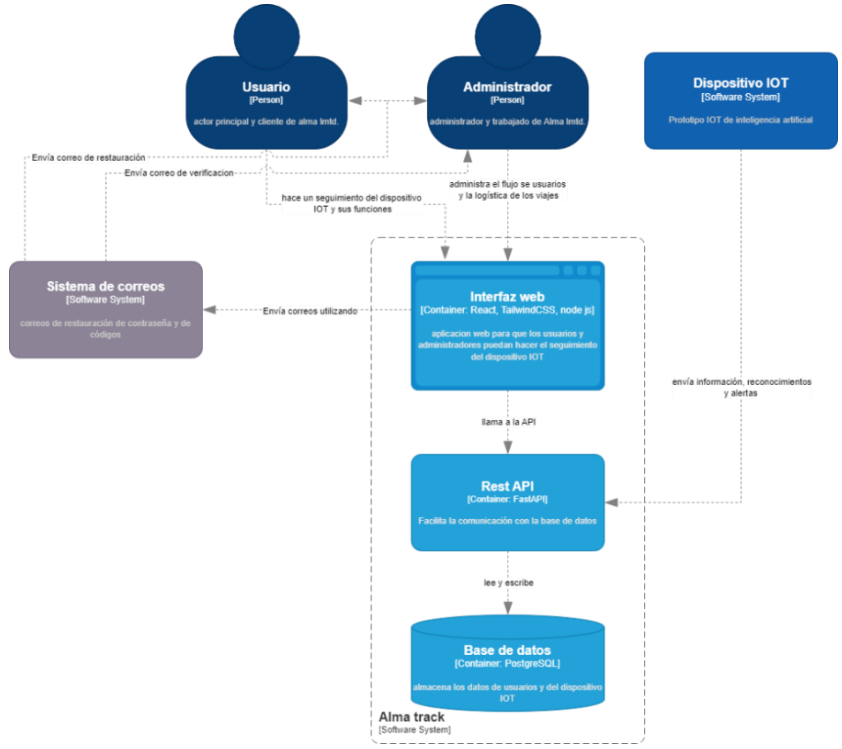

Figura 3 Modelo C4 nivel de contenedores.

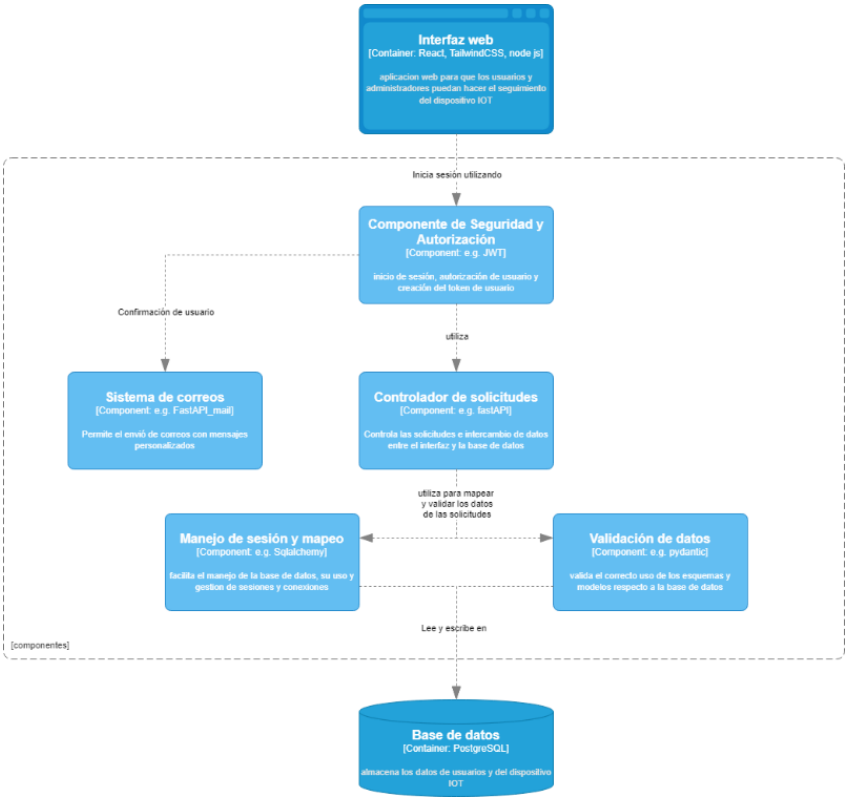

Figura 4 Modelo C4 nivel de componentes.

CAPÍTULO 4: Desarrollo

Para el desarrollo del proyecto y organización, se aplicó la metodología ágil con sprints de dos a tres semanas dependiendo del grado de dificultad y/o conocimiento del contenido del sprint, en las cuales se fueron anotando las tareas como tablones en Trello [15] dependiendo en la fase de desarrollo que se encontraban, las cuales se clasificaban en pendientes, en desarrollo, en pruebas y terminado, para de esa forma ir manteniendo constancia del progreso. En cada sprint se realizaban de una a dos reuniones con el profesor encargado *Geoffrey Hecht,* esto variando, dependiendo de las necesidades y dudas que iban surgiendo durante el desarrollo, más una reunión de retroalimentación y aprobación por el ingeniero encargado de Alma *Miguel Chandia*, sino también para discutir y analizar posibles cambios y descubrimientos, como lo pueden ser el enfoque de la base de datos y sus cambios por la escalabilidad del producto.

Para mantener el código actualizado en un repositorio se utilizó *GitHub*, debido a su facilidad a la hora de aplicar actualizaciones y evitar errores. De igual forma el código fuente viene incluido con comentarios y documentación del funcionamiento de este, además de un plan de integración a modo de tutorial en *Notion* exclusivo para los trabajadores de alma. En conformidad con los términos establecidos en el contrato con la empresa, el repositorio asociado y el tutorial de integración de este proyecto no puede ser compartido públicamente.

4.1 Tecnologías Utilizadas

Con el fin de detallar y explicar la relevancia de cada componente, tecnología o herramienta utilizada, se explica de forma breve la relevancia y desempeño de cada una de ellas:

- **Docker:**
  - ¿Qué es?: Es una plataforma que permite crear, distribuir y ejecutar aplicaciones en entornos aislados mediante contenedores.
  - Relevancia: Fue clave para garantizar la portabilidad y consistencia de la base de datos en PostgreSQL, simplificando el desarrollo y testeo de la aplicación.
- **FastAPI:**
  - ¿Qué es?: Es un framework web moderno y de alto rendimiento para la construcción de APIs en Python. Este destaca en por su facilidad de uso, eficiencia, además de tener soporte nativo para la validación de datos.
  - Relevancia: Se utilizó como base para el desarrollo de la API del proyecto, permitiendo construir endpoints rápidos y estables, además de facilitar la integración de servicios como los de autentificación y correo.

- **Uvicorn:**
  - ¿Qué es?: Es un servidor ASGI (Asynchronous Server Gateway Interface) [5] de alto rendimiento que se utiliza para ejecutar aplicaciones basadas en Python, como las desarrolladas con FastAPI.
  - Relevancia: Se empleó como servidor de producción para la API desarrollada con fastAPI, proporcionando un rendimiento óptimo y un soporte para el manejo de endpoints.
- **Node.JS:**
  - ¿Qué es?: Es un entorno de ejecución para JavaScript en el lado del servidor. Permite construir aplicaciones escalables y rápidas.
  - Relevancia: Se utilizó como entorno principal para el desarrollo del frontend, sosteniendo herramientas clave como Vite y TailwindCSS, asegurando un flujo de desarrollo eficiente y una integración fluida del frontend.
- **Vite:**
  - ¿Qué es?: Es una herramienta de desarrollo y build para proyectos de frontend, enfocada en la rapidez. Proporciona un servidor de desarrollo con recarga instantánea y una optimización avanzada para la compilación en producción.
  - Relevancia: Se utilizó para mejorar la experiencia de desarrollo del proyecto, permitiendo una mayor velocidad en la recarga de cambios durante el desarrollo y una optimización eficiente en la compilación del código.
- **TailwindCSS:**
  - ¿Qué es?: Es un framework de CSS basado en utilidades que permite diseñar interfaces de manera rápida y flexible, proporcionando clases predefinidas que se aplican directamente en el HTML.
  - Relevancia: Facilitó la personalización y consistencia en el diseño de la interfaz, permitiendo un desarrollo ágil con sus clases predefinidas.
- **GitHub:**
  - ¿Qué es?: Es una plataforma de control de versiones y colaboración basada en Git, utilizada para almacenar y gestionar repositorios de código fuente.
  - Relevancia: Se utilizó para gestionar el código fuente del proyecto y control de versiones, manteniendo un orden durante el desarrollo y asegurando un historial claro de cambios.
- **Trello:**

  - ¿Qué es?: Es una herramienta de gestión de proyectos basada en tableros, que permite organizar tareas y proyectos de manera visual mediante tarjetas.
  - Relevancia: Se utilizó para la planificación y seguimiento de tareas durante el desarrollo del proyecto, organizando los sprints.
- **Postman:**
  - ¿Qué es?: Es una herramienta para probar y documentar APIs. Permite realizar solicitudes HTTP de manera sencilla, facilitando la validación de endpoints y la automatización de pruebas.
  - Relevancia: Se utilizó para realizar pruebas de la API durante el desarrollo y la automatización de pruebas para los endpoints, asegurando su correcto funcionamiento.
- **VS Code:**
  - ¿Qué es?: Editor de código fuente que incluye depuración, finalización inteligente de código que permite múltiples expansiones y modificaciones.
  - Relevancia: Se utilizó como editor de código durante el desarrollo y su expansión "port forwarding" [14] que permitió el testeo con usuarios por medio de puertos que podían ser compartidos por internet.
- **DBeaver:**
  - ¿Qué es?: Es una herramienta universal para la administración y diseño de bases de datos. Proporciona una interfaz gráfica para conectarse y gestionar diversas bases de datos como MySQL, PostgreSQL, entre otras.
  - Relevancia: Se utilizó para gestionar, visualizar y comprobar el correcto funcionamiento de la base de datos, además de la creación de un esquema relacional de la misma.
- **Notion:**
  - ¿Qué es?: Es una herramienta de organización y gestión de información que permite la creación de documentos y tableros para la colaboración en equipo.
  - Relevancia: Se utilizó para la creación de un tutorial de implementación e integración entre todos los sistemas creados, enfocado a los empleados de Alma.

4.2 Base de Datos

Con FastAPI como framework de desarrollo, este es capaz de soportar distintas bases de datos relacionales, pero en la cual PostgreSQL destacaba por sobre las demás por su facilidad de uso

con Docker y su compatibilidad con SQLAlchemy, además de que opción sugerida dentro de la documentación de FastAPI.

A través de las reuniones con Alma y para cuando había un entendimiento mayor de la visión de la empresa, se establecieron las entidades principales de la base de datos. Posteriormente, se modificó para conveniencia de las necesidades de rendimiento y de escalabilidad [3]. El diagrama de la ***Figura 5*** fue realizado con la herramienta DBeaver de forma automática utilizando la base de datos creada en PostgreSQL.

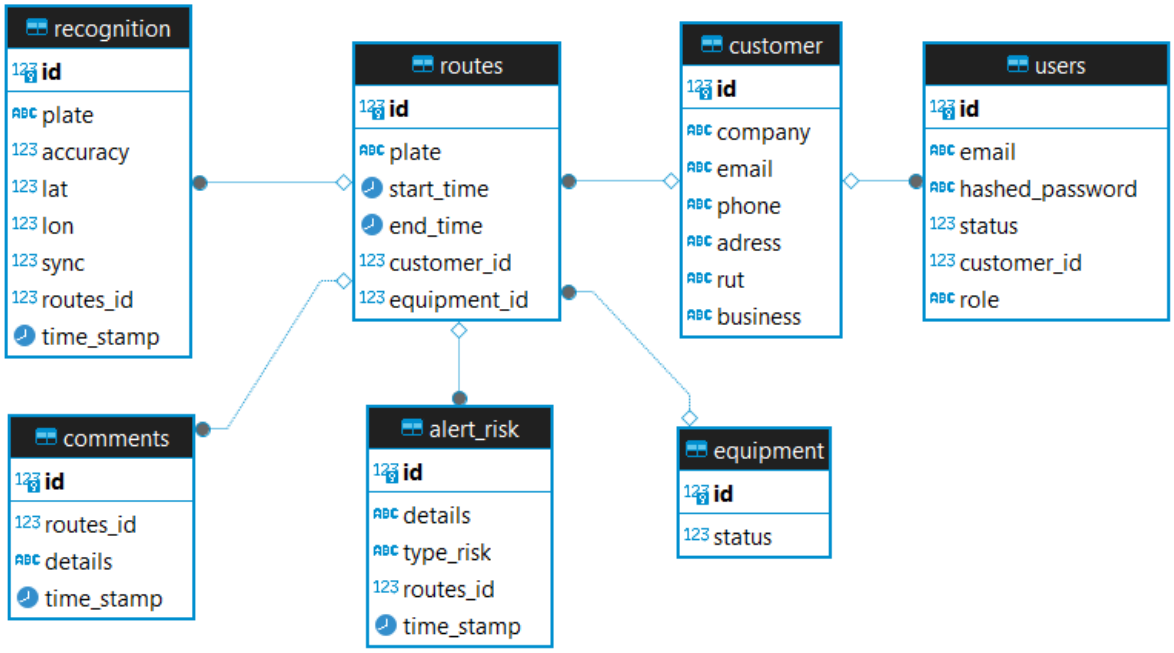

Figura 5 Diagrama ER final de la base de datos.

A continuación, una explicación más detallada de cada entidad y las relaciones visibles en la ***Figura 5***:

- **Customer:** Entidad encargada de representar a la empresa cliente, sus datos principales y de contacto.
- **Users:** Entidad encargada de representar a los usuarios, con la facultad de hacer seguimiento de los viajes asociados a su empresa. Los roles pueden ser user y admin, que representan al usuario cliente y administrador respectivamente.
    - Relaciones:
        - Customer_id: Empresa a la que está asociado el usuario. Es nulo si el rol del usuario es admin.
- **Routes:** Entidad encargada de representar a los viajes asociados a una empresa.

    - Relaciones:
        - Customer_id: Empresa a la que está asociado el viaje y el cual todos los usuarios con este id pueden hacer seguimiento.
        - Equipment_id: Dispositivo IOT el cual es utilizado durante el viaje para generar reconocimientos y alertas.
- **Recognition:** Entidad encargada de representar los reconocimientos generados por un dispositivo IOT durante un viaje.
    - Relaciones:
        - Routes_id: Viaje en el cual se registraron los reconocimientos.
- **Alert_risk:** Entidad encargada de representar las alertas generadas por el dispositivo IOT durante un viaje.
    - Relaciones:
        - Routes_id: Viaje en el cual se registraron las alertas.
- **Comments:** Entidad encargada de representar los comentarios asociados a un viaje.
    - Relaciones:
        - Routes_id: Viaje en el cual se ingresaron los comentarios.
- **Equipment:** Entidad encargada de representar el dispositivo IOT, esta entidad es representativa, ya que, para el momento del desarrollo de proyecto, este solo es un prototipo que todavía está en fase de mejora.

4.3 Estructura de Archivos

Para permitir ver todo el panorama y los detalles del proyecto, en la ***Figura 6*** se muestra y describe la estructura principal de los archivos, en los cuales está la App de FastAPI (backend) y la App de React (frontend), con el fin de facilitar la compresión de la aplicación y sus respectivas funcionalidades.

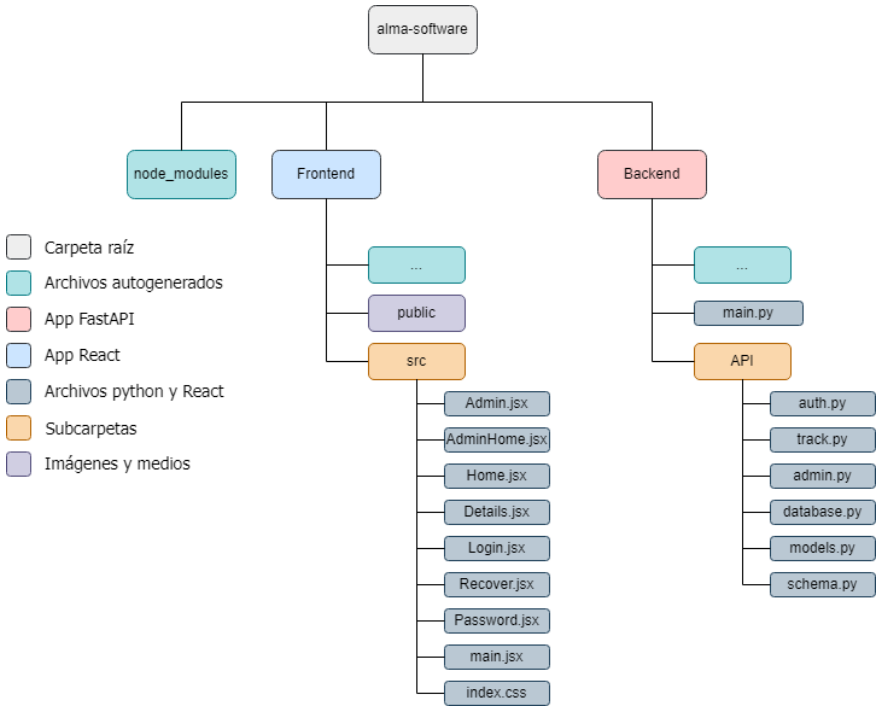

Figura 6 Estructura de archivos del proyecto "Alma track".

Debido a la facilidad de uso de los archivos en FastAPI, dentro de la subcarpeta API cada archivo es utilizado para una función, tanto para la logística y coordinación del framework, como para funciones específicas de la aplicación Alma track.

Dicho esto, estos son los archivos más importantes de la App FastAPI y su subcarpeta:

- **Main.py:** Este archivo es el núcleo de la API, encargándose de inicializar FastAPI, gestionar las rutas de las funciones de la App, y manejar las políticas de CORS [8], permitiendo que solo las IP autorizadas puedan acceder a los endpoints. Además, incluye funciones básicas de tipo POST.
- **Auth.py:** Este módulo contiene los endpoints relacionados con la autenticación de usuarios, incluyendo funcionalidades para el inicio de sesión, cambio de contraseñas, y envío de correos electrónicos.
- **Track.py:** Este módulo contiene los endpoints de las funciones pertenecientes al usuario para hacer el seguimiento de los viajes, inspección de reconocimientos, alertas y comentarios.
- **Admin.py:** En este módulo se encuentran las funciones dedicadas a la administración, tales como la incorporación de empresas y usuarios en la base de datos, así como la gestión de equipos y viajes.

- **Database.py:** Este archivo alberga la URL de la base de datos y gestiona la inicialización y mantenimiento de la sesión activa, permitiendo que los demás módulos que contienen los endpoints puedan acceder de manera eficiente a los recursos de la base de datos.
- **Models.py:** Este archivo define las clases de modelos que representan las tablas de la base de datos, utilizando SQLAlchemy como ORM (Object-Relational Mapping) [9]. Los modelos especifican la estructura de las tablas, incluyendo los tipos de datos, las relaciones entre las entidades, y cualquier restricción o clave foránea necesaria para mantener la integridad referencial en la base de datos.
- **Schema.py:** Este archivo contiene las clases de esquemas que describen la estructura de los datos que se intercambian a través de la API, utilizando Pydantic [10] para la validación y serialización de datos. Los esquemas se utilizan para validar los datos de entrada y salida en los endpoints, garantizando que la información cumpla con los requisitos definidos antes de ser procesada o enviada al cliente.

Retomando el patrón de arquitectura MVC (Model-View-Controller), se observa en la ***Figura 6*** que los componentes están representados en los siguientes archivos:

- **Model:** representado en los archivos *models.py* y *schema.py* de la subcarpeta *API* de *Backend*, contienen los modelos y los esquemas que representan la estructura de las tablas de la base de datos.
- **View:** Representado en todos los archivos *jsx* de la subcarpeta *src* de *Frontend*, estas tienen las vistas del usuario y la información que reciben de los controladores utilizando las funciones *fetch* [6] las cuales son autorizadas y manejadas en los archivos *main.py* y *database.py.*
- **Controller:** representado en los archivos *main.py*, *auth.py*, *admin.py* y *track.py* de la subcarpeta *API* de *Backend*, los cuales contienen los endpoints que manejan las funciones básicas de GET, POST y PATCH [8] de la base de datos y las funciones de registro, inicio de sesión, tokens y envío de correos respectivamente.

Al ser Frontend la carpeta encargada de la interfaz de la aplicación web, esta se explica de forma más detallada en el siguiente punto, ahondando en las vistas de cada archivo y el recorrido URL que les permite al usuario manejarse en el interfaz web.

4.4 Interfaz

Con el objetivo de proporcionar una experiencia agradable y accesible para el usuario final, se tomó como principios para el enfoque del desarrollo el que esta interfaz fuese consistente, flexible, eficiente, intuitivo y de fácil uso. Además, para la selección de colores e identidad artística de la interfaz se tomó como base el logo proporcionado por la empresa.

A continuación, se detallarán las vistas principales de la aplicación web, así como las funcionalidades que gestionan y los archivos JSX que las contienen.

- **Inicio de sesión**

  El archivo *Login* contiene esta vista, que incluye los campos necesarios para que el usuario cliente introduzca su correo electrónico y contraseña. En caso de estar registrado, se procederá a la creación del token de autenticación. La vista se ilustra en la ***Figura 7***.

  - Navegación:
    - **Página principal: Viajes y alertas**: Tras la generación exitosa del token, el usuario será redirigido a la vista mostrada en la ***Figura 8***.
    - **Enviar código de recuperación:** Al seleccionar el hipervínculo "¿Olvidaste la contraseña?" el usuario será redirigido a la vista de la ***Figura 15***.

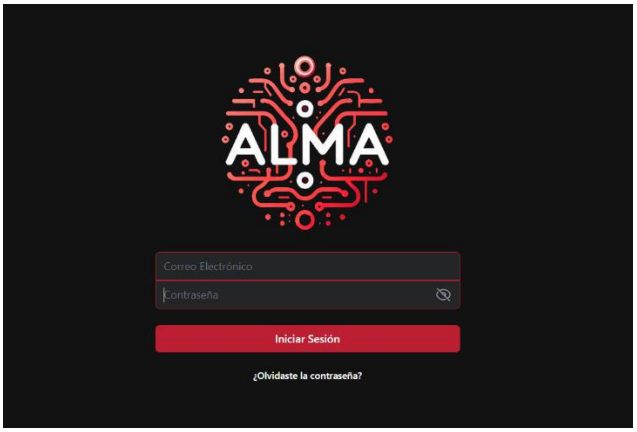

Figura 7 Inicio de sesión de Usuario.

- **Página principal: Viajes y alertas**

  La vista principal, contenida en el archivo *Home*, incluye las funciones clave que los usuarios pueden utilizar. Estas funcionalidades son accesibles a través del menú lateral, visible en las ***Figuras 8*** y ***9***.

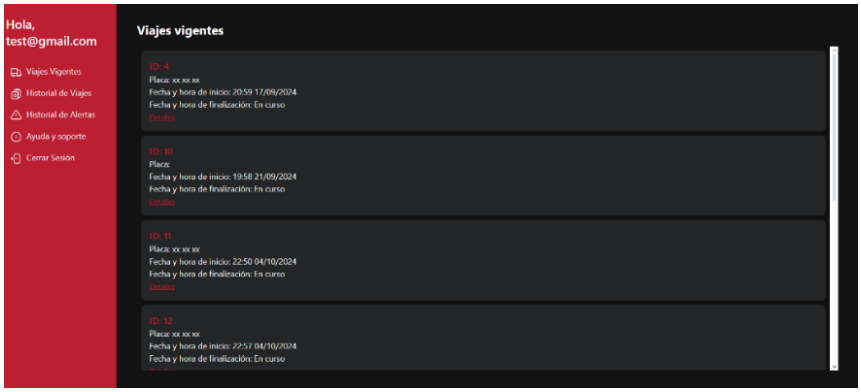

Figura 8 Vista principal de Alma Track.

Figura 9 Menú lateral de la vista principal.

Dentro de las vistas disponibles en el menú, se encuentran:

Viajes vigentes: Tal como indica su nombre, esta sección permite el seguimiento de los viajes. Los viajes se presentan en contenedores organizados que muestran sus detalles más relevantes, junto con un hipervínculo para más información. La vista se ilustra en la ***Figura 10***.

- Navegación:
  - Detalles del viaje: Cada viaje cuenta con un hipervínculo titulado “seguimiento”, a través del cual se puede acceder a información más detallada, visible en la ***Figura 11***.

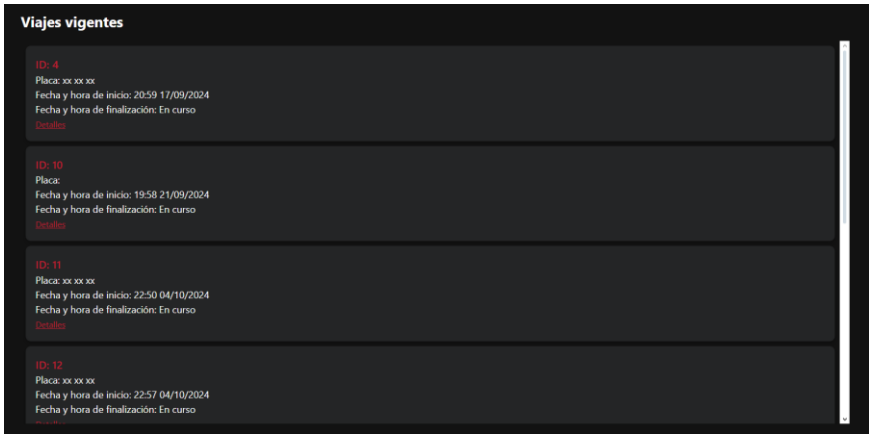

Figura 10 Vista de viajes vigentes.

Figura 11 Vista de los detalles de un viaje.

Historial de viajes: Al momento de finalizar un viaje, deja de ser visible en la vista Viajes vigentes y se traslada al historial. Esta vista cuenta con filtros para seleccionar la fecha de inicio y de término. La visualización se presenta en la ***Figura 12***.

- Navegación:
  - Detalles del viaje: Cada viaje en el historial incluye un hipervínculo titulado “Detalles”, a través del cual se puede acceder a información detallada, visible en la ***Figura 11***.

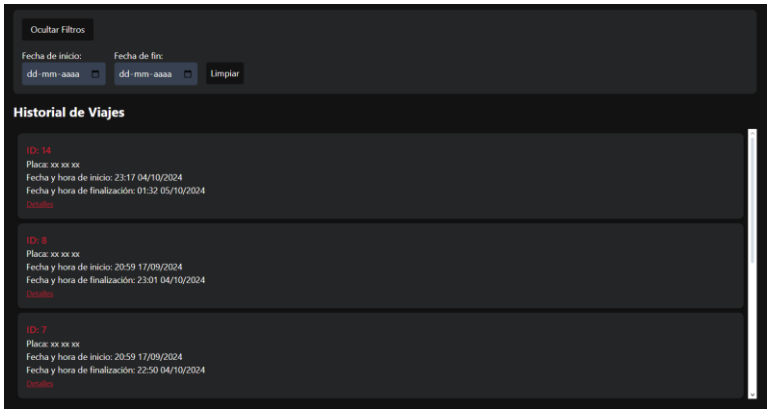

Figura 12 Vista de historial de viajes.

Historial de alertas: Manejando un formato similar al de los viajes, el historial de alertas utiliza un filtro fecha de inicio, de término y por tipo de riesgo. La vista está disponible en la ***Figura 13.***

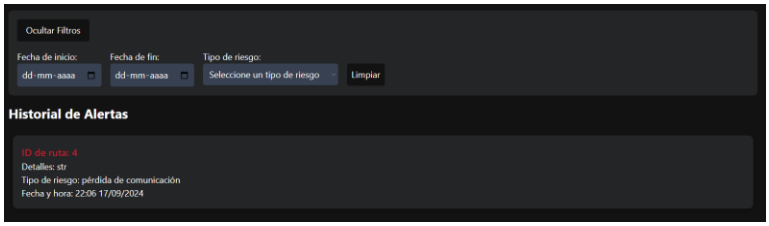

Figura 13 Vista de historial de alertas.

Ayuda y soporte: En esta sección se proporciona acceso a la información de contacto de Alma Ltda. La representación visual de esta vista se encuentra en la ***Figura 14***.

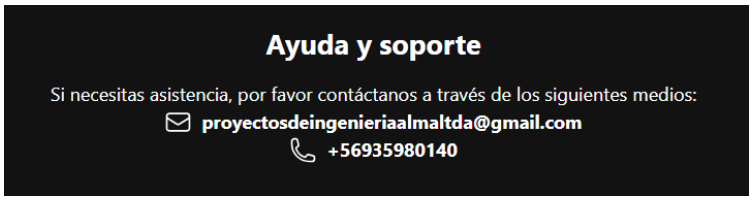

Figura 14 Vista de ayuda y soporte.

- **Detalles**

Contenido dentro del archivo *Details*. Esta vista contiene la información relacionada con el viaje, la cual se muestra en la parte superior de la pantalla, mientras que los reconocimientos, alertas y comentarios están claramente diferenciados entre sí.
Entre las funciones disponibles dentro de la sección de reconocimientos es posible visualizar la ubicación en la que ocurrió el reconocimiento pulsando el botón que está a la derecha de

cada reconocimiento, el cual, utilizando un hipervínculo, el navegador abre una nueva pestaña y redirige al usuario a la ubicación en Google Maps [17], la cual es estática e independiente. Además, en la sección de comentarios, es posible ingresar comentarios en tiempo real, y el sistema registra y notifica la hora en que fueron realizados. Esta vista se puede observar en la ***Figura 11***.

- **Recuperación de contraseña**

  El archivo "Recover" contiene el sistema implementado para la recuperación de contraseñas. El proceso comienza con la confirmación del usuario, quien debe ingresar su correo electrónico para recibir un código de recuperación, necesario para proceder con el cambio de contraseña. El campo para ingresar el código permanecerá bloqueado hasta que el código haya sido enviado. Esta vista se muestra en la ***Figura 15***.

  - Navegación:
    - Cambiar contraseña: Si el código de recuperación es correcto, el usuario será redirigido a la página para cambiar su contraseña, como se puede ver en la ***Figura 16***.

Figura 15 Vista para recuperar contraseña.

- **Cambiar contraseña**

  Contenido en el archivo *Password*, para realizar el cambio de contraseña, se implementó un contenedor que solicita al usuario ingresar la nueva contraseña dos veces, a fin de confirmar que ambas coincidan. Esta vista se puede observar en la ***Figura 16***.

  - Navegación:
    - Inicio de sesión: En caso de que la nueva contraseña sea válida, el cambio será efectuado y el usuario redirigido al inicio de sesión.

Figura 16 Vista para confirmar el cambio de contraseña.

- **Página principal: administrador**

  Contenido en el archivo *AdminHome*, Al ingresar a través del inicio de sesión siendo administrador, Mantiene el mismo esquema de menú con las vistas ubicadas en el lado izquierdo de la pantalla, como se muestra en la ***Figura 17***. Además, incluye mensajes y alertas que notifican sobre el correcto funcionamiento de las funciones, así como advertencias sobre errores de tipeo, problemas de conexión o fallos del sistema, los cuales se pueden apreciar ejemplos de estos en la ***Figura 18*** y ***Figura 19***. Las vistas disponibles para el administrador son:

Figura 17 Menú de página principal: administrador.

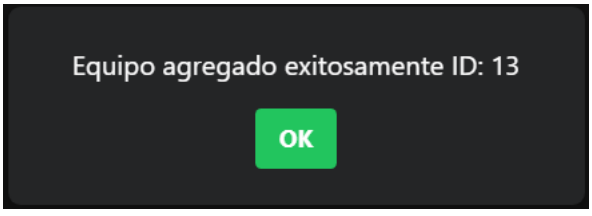

Figura 18 Mensaje de confirmación.

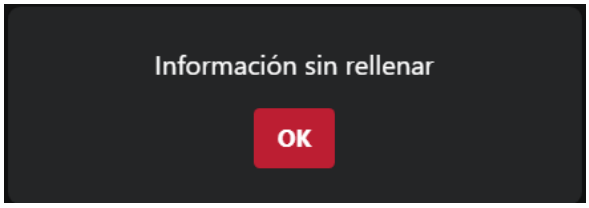

Figura 19 Mensaje de Error.

Añadir empresa: Esta funcionalidad consiste en un formulario con campos de entrada para completar todos los datos necesarios para registrar una empresa correctamente. En caso de que falte información, se muestra un mensaje personalizado indicando los datos faltantes. La vista correspondiente se puede observar en la ***Figura 20*.**

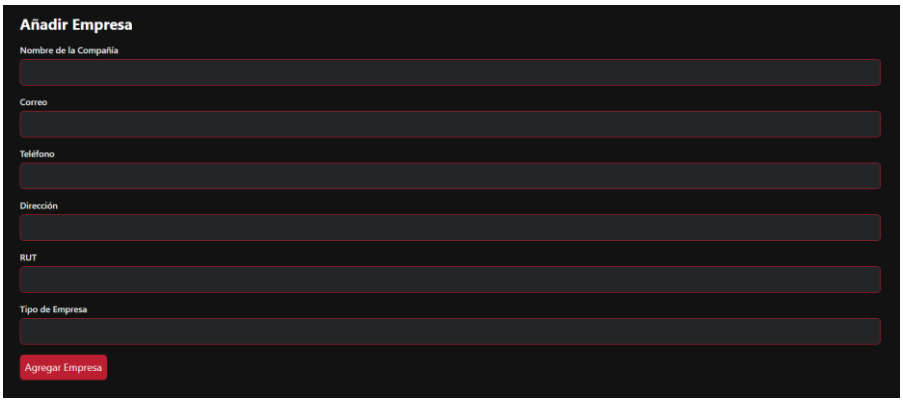

Figura 20 Vista de añadir empresa.

Añadir usuario: Esta funcionalidad incluye una barra de selección para elegir la empresa, que permite filtrar las opciones mediante una búsqueda, y un campo de entrada para ingresar el correo electrónico del usuario. Según las políticas de la empresa, el usuario será creado por el administrador con una contraseña generada aleatoriamente, la cual será enviada directamente al correo del usuario. La vista correspondiente se puede observar en la ***Figura 21***.

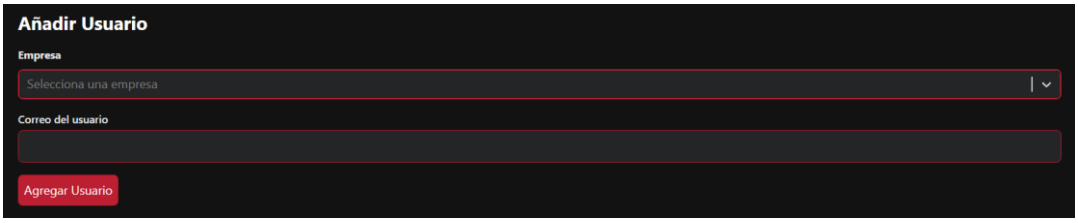

Figura 21 Vista de añadir usuario cliente.

Añadir equipos: Debido a la ausencia del prototipo IOT durante el desarrollo, y sujeto a posibles modificaciones futuras, esta vista actualmente solo cuenta con un botón que permite agregar un equipo a la base de datos.

Añadir viajes: Esta funcionalidad incluye una barra de selección para elegir la empresa y el ID del equipo, con la opción de filtrar las opciones mediante una búsqueda, así como un campo de entrada para ingresar la patente del vehículo. La vista correspondiente se puede observar en la ***Figura 22***.

Figura 22 Vista de añadir viajes.

Terminar viajes: Para un funcionamiento más ordenado de esta vista, primero se solicita al usuario que ingrese la empresa afiliada al viaje que desea finalizar. A continuación, se muestran todos los viajes junto con parte de sus datos, y a la derecha de cada uno hay un botón que permite culminar el viaje. Tras la confirmación del navegador, se muestra un mensaje en pantalla indicando si el viaje fue finalizado correctamente. La vista correspondiente se puede observar en la ***Figura 23***.

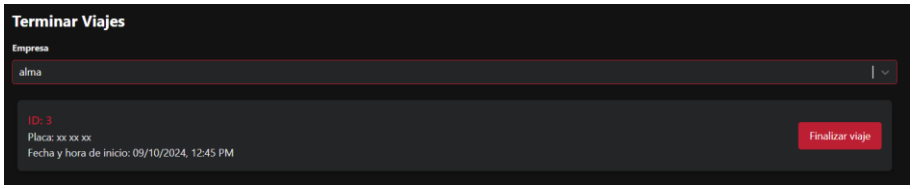

Figura 23 Vista de terminar viaje.

CAPITULO 5: Test y Validación

En este capítulo se darán a conocer los métodos y resultados de las pruebas de software, con tal de asegurar un nivel de calidad del producto final.

5.1 Pruebas con Postman

Con el objetivo de validar los endpoints de la API, se creó un espacio de trabajo en Postman, el cual permite el uso de Scripts Post-Response [11] para la realización de pruebas automatizadas. Este espacio fue organizado en cuatro subcarpetas, correspondientes a los endpoints definidos en los archivos *main.py*, *track.py*, *admin.py* y *auth.py*.

Es importante mencionar que las pruebas se llevaron a cabo con la base de datos vacía, siguiendo un orden estratégico, donde cada prueba se ejecutaba de acuerdo con los prerrequisitos necesarios para obtener las respuestas esperadas. Asimismo, se omitieron algunas pruebas debido al nivel de seguridad integrado que proporciona Pydantic en la validación de datos, categoría en las que entran las pruebas de formularios vacíos, datos incorrectos y no existentes.

Fueron cinco los tipos de pruebas para todos los endpoints, con ligeras variaciones las cuales son especificadas en explicaciones posteriores.

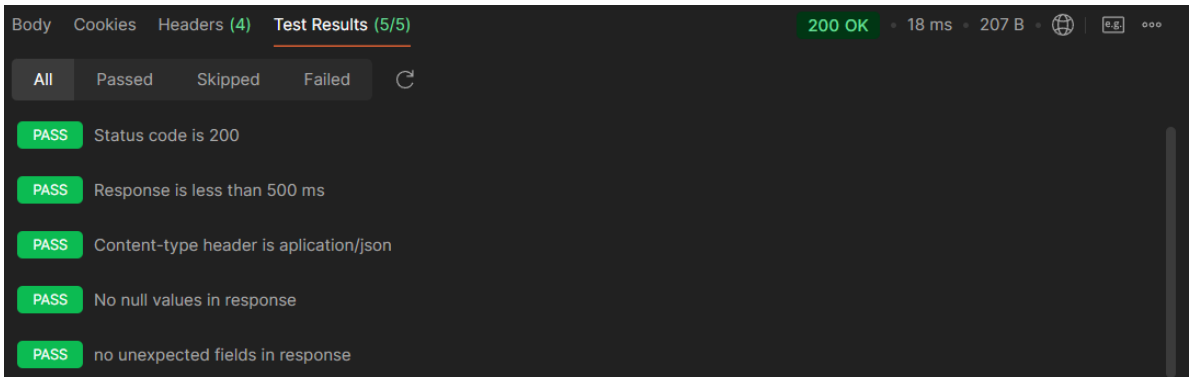

Figura 24 Test realizados en Postman.

A continuación, una explicación de los tipos de pruebas realizadas en Postman, visibles en las ***Figura 24:***

- **Status Code is 200/400**: Esta prueba verifica si la respuesta retornada contiene en sus headers un código de estado 200 o 400, que indican solicitud correcta (*OK*) o solicitud incorrecta (*BAD REQUEST*) respectivamente [12].

- **Response is less than 500 ms**: Esta prueba evalúa si la respuesta se generaba en menos de 500 ms, con el fin de asegurar un tiempo de respuesta adecuado y dentro de los límites aceptables para la aplicación.
- **Content-type header is application/json**: Esta prueba validaba que el header Content-Type de la respuesta es application/json, asegurando que el formato de la respuesta es el correcto y compatible con las especificaciones de la API.
- **No null values in response**: En esta prueba se verifica que la respuesta no incluya valores nulos o que incluye un valor específico en ciertos casos, lo que garantiza la consistencia y completitud de los datos retornados por la API.
- **No unexpected fields in response**: Esta prueba valida que la respuesta no contenga campos no esperados, asegurando que solo se devuelvan los datos especificados en la documentación de la API y evitando posibles inconsistencias o fallos de seguridad.

A continuación, se describen de manera detallada los endpoints, el cómo funcionan y cambios que se le pudieron haber aplicado a las pruebas anteriormente mencionadas con tal de abarcar más margen de error:

- Main: En este entorno se ponen a prueba las funciones relacionadas con el sistema IOT.
    - Add Recognition: Se rellenan los campos del formulario de registro con los datos correctos, con el fin de comprobar su funcionamiento.
    - Add Alert risk: Se llenan los campos del formulario de registro con información válida para comprobar su funcionalidad.
- Track: En este entorno se ponen a prueba las funciones de seguimiento del usuario cliente.
    - Add Comments: Se rellenan los campos del formulario de registro con los datos correctos, con el fin de comprobar su funcionamiento, de igual forma, verifica que exista un *Route* asociado.
    - Get Routes Recognition: Tomando como input el ID de la *Route*, esta verifica que exista tal viaje dentro de la base de datos y que reciba los *Recognition* con su mismo ID de viaje.
    - Get Routes Alerts: Tomando como input el ID de la *Route*, esta verifica que exista tal viaje dentro de la base de datos y que reciba los *Alert risk* con su mismo ID de viaje.

  - Get Routes Comments: Tomando como Input el ID de *Route,* esta verifica que exista tal viaje dentro de la base de datos y que reciba los *Comments* con su mismo ID de viaje.
  - Get User Alerts: Tomando como input el ID de *User*, esta verifica que exista tal usuario y pasa la llamada por la tabla *Customer*, busca las *Routes* relacionadas y de estas extrae todas las Alert risk.
  - Get Active User Routes: Tomando como input el ID de *User*, esta verifica que exista el usuario y las *Routes* activas. En la prueba "*No null values in response"* se cambiaron los requisitos, ya que, los viajes activos poseen el campo end_time nulo.
  - Get Clossed User Routes: Tomando como input el ID de *User*, esta verifica que exista el usuario y las *Routes* terminadas. En la prueba "*No null values in response"* no se hicieron cambios, ya que, los viajes terminados poseen el campo *end_time* distinto de nulo.
- Admin: En este entorno se ponen a prueba las funciones del administrador.
  - Add Equipment: Se rellenan los campos del formulario de registro con los datos correctos, con el fin de comprobar su funcionamiento.
  - Add Customer: Similar a la anterior, pero con la información de un cliente.
  - Add Route: Verifica que el *Customer* y el *Equipment* asociados existan en la base de datos. En la prueba "No null values in response", se ajustaron los requisitos debido a que las rutas activas tienen el campo end_time nulo. Además, se cambia el estado del equipo de 1 a 0.
  - Add Route / equip not aviable: Tomando el caso anterior, en caso de que el *Equipment* asociado no esté disponible, el *status* del equipo utilizado es igual a 0, no se ejecutara el endpoint.
  - End Route: Tomando como input el ID de Route, verifica que el campo *end_time* se rellene con la fecha y hora de la base de datos. También se realiza un cambio en el *status* del equipo utilizado de 0 a 1.
  - Get All Customer: Sin tomar ningún input, verifica que retorne todas las empresas ingresadas dentro de la base de datos.
  - Get Aviable Equipment: Sin tomar ningún input, verifica que retorne todos los equipos disponibles. En la prueba "*No null values in response"* se verifica que la respuesta del campo *status* de todos los equipos recibidos sea igual a 1.

    - Get Active Customer Routes: Tomando como input el ID de un Customer, verifica si este posee Routes activos. En la prueba "*No null values in response"* se verifica que todas las respuestas obtenidas tengan nulo en su campo *end_time*.
- Authentification: En este entorno se ponen a prueba las funciones de autentificación, token y envió de correos.
    - Get Current User: Usando el token de usuario como input, se verifica que coincida con los datos utilizados en JWT.
    - Invalid Current User: Similar a la prueba anterior, pero con un token inválido.
      Add Admin: Agrega un administrador a la base de datos, asegurando que no falten datos.
    - Already Registered Admin: Mismo endpoint que el anterior, pero en el caso de que el administrador ya esté registrado en la base de datos.
    - Add User: Agrega un usuario a la base de datos, verificando que no falten datos.
    - Already Registered User: Similar a la prueba anterior, pero en el caso de que el usuario ya esté registrado.
    - Create Token: Usando JWT, se genera un token cifrado con el correo y el ID del usuario, verificando que los campos retornados sean correctos.
    - Create Admin Token: Igual que el anterior, pero para un administrador.
    - Send Email Code: Envía un código al correo del usuario.
    - Send Password to Email: Envía la contraseña al correo del usuario.

5.2 Pruebas con usuarios

Con el objetivo de evaluar la experiencia de usuarios al utilizar la plataforma, se creó una encuesta basada en la escala de Likert [13], para evaluar de forma lineal la experiencia del usuario, el cual permite evaluar de forma fácil y confiable la usabilidad del software. Este está compuesto por diez preguntas con respuestas que van del uno al cinco, siendo "Totalmente en desacuerdo" y "Totalmente de acuerdo", respectivamente. Además de las preguntas base, también se agregaron cuatro preguntas adicionales de desarrollo con el objetivo que los usuarios puedan aportar con opiniones, comentarios o ideas que pudiesen mejorar el producto a futuro.

Las preguntas de la encuesta se pueden apreciar en la Tabla:

	Totalmente en desacuerdo				Totalmente de acuerdo
1. Usaría esta plataforma con frecuencia	1	2	3	4	5
2. La plataforma es innecesariamente compleja	1	2	3	4	5
3. La plataforma era fácil de usar	1	2	3	4	5
4. Se requiere de un guía o manual para utilizar la plataforma	1	2	3	4	5
5. Las diversas funciones de esta plataforma estaban bien integradas	1	2	3	4	5
6. La plataforma era muy inconsistente	1	2	3	4	5
7. La mayoría aprendería a utilizar este sistema rápidamente	1	2	3	4	5
8. La plataforma era muy incómoda de usar	1	2	3	4	5
9. Me sentí muy confiado/a usando la plataforma	1	2	3	4	5
10. Es necesario aprender muchas cosas antes de utilizar la plataforma	1	2	3	4	5
Otros:					
11. ¿Qué funcionalidad de la plataforma ha sido la que más le ha gustado?					
12. ¿Qué funcionalidad de la plataforma usted eliminaría?					
13. ¿Qué otras funcionalidades extras agregarías a la plataforma?					
14. Comentario/opinión en general:					

Tabla 2 Encuesta a usuarios.

Utilizando como medio “port forwarding” de VS Code, el cual, al ejecutar el servicio de forma local, esta generaba una vista del puerto local que permitió a los usuarios interactuar con una versión de prueba del interfaz web.

La encuesta se aplicó a un grupo de diez usuarios, un número que, aunque reducido, abarca una diversidad de perfiles con el fin de analizar la usabilidad de la plataforma más allá del público objetivo. Este grupo incluyó estudiantes de ingeniería, estudiantes de animación digital, ingenieros con más de cuatro años de experiencia laboral e incluso miembros de distintas destrezas de la empresa homónima.

Es importante destacar que la versión de prueba contenía datos ya ingresados en su base de datos para que pudiesen interactuar con todas las funcionalidades. Al grupo de usuarios encuestados se les otorgo acceso a la vista de usuarios mientras que los miembros de Alma tuvieron acceso a la vista de administrador y de usuario.

De esta forma, durante el desarrollo del proyecto y al final de este, se obtuvo una retroalimentación por parte de los encuestados, representado en la ***Tabla 3*** a través de la media de puntaje:

Pregunta de la encuesta	Media de puntaje
1. Usaría esta plataforma con frecuencia	4.6
2. La plataforma es innecesariamente compleja	1.4
3. La plataforma era fácil de usar	4.8
4. Se requiere de un guía o manual para utilizar la plataforma	1.9
5. Las diversas funciones de esta plataforma estaban bien integradas	4.7
6. La plataforma era muy inconsistente	2.1
7. La mayoría aprendería a utilizar este sistema rápidamente	3.5
8. La plataforma era muy incómoda de usar	1.3
9. Me sentí muy confiado/a usando la plataforma	4.5
10. Es necesario aprender muchas cosas antes de utilizar la plataforma	1.3

Tabla 3 Resultados de la encuesta de usuarios

Para poder interpretar y cuantificar el desempeño de la plataforma durante las pruebas se realizó un cálculo del método SUS (System Usability Scale) [16] para evaluar la usabilidad del sistema en base a una escala de Likert. a partir de estos resultados y utilizando la fórmula de Spreadsheets,

Formula de Spreadsheets	
Pasos de la formula	Puntajes obtenidos
Suma de las preguntas impares y resta 5 a ese total	17.1
Suma las respuestas de los enunciados pares y resta ese total a 25	17
Suma ambos resultados y multiplícalo por 2,5	85.25

Tabla 4 Resultados de la fórmula de Spreadsheets de SUS

De esta forma se calculó que el puntaje obtenido fue de 85.25 sobre 100, lo cual, según el criterio de la fórmula, se considera dentro del rango aceptable. Asimismo, en la ***Figura 25*** se presenta la escala de puntaje SUS, que permite visualizar y evaluar el desempeño de la usabilidad del sistema y es el mismo que se utilizó para llegar a las conclusiones de esta.

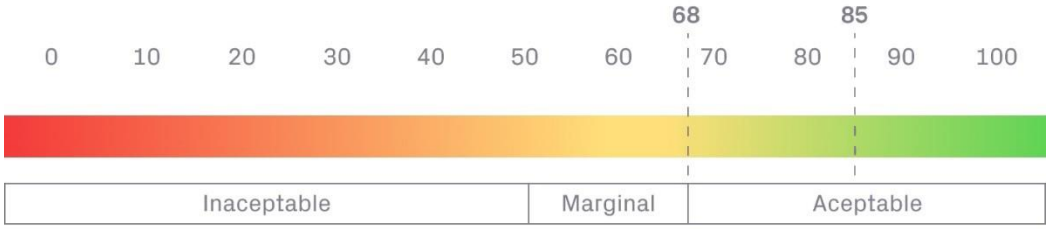

Figura 25 Escala de puntaje SUS

Podemos concluir que el desempeño de la plataforma durante pruebas de usuarios fue aceptable, pero que de igual forma hay un margen de mejora, ya que, a través de las preguntas y comentarios de los encuestados, se extrajeron funciones e ideas que pueden brindar una mejoría en la usabilidad dentro del contexto de la aplicación y se explora a detalle en el apartado de planes futuros.

CAPÍTULO 6: Conclusiones

El propósito del proyecto era crear desde cero una arquitectura para el producto IOT basado en inteligencia artificial también conocido como "*alma track"* para el contexto de la empresa de Proyectos de ingeniería Alma ltda. A lo largo de este informe, se han descrito las características del sistema, la base de datos, la API y el frontend, así como las mejoras y enfoques implementados en términos de escalabilidad, rendimiento y trazabilidad de procesos. En esta sección, se discuten los resultados y conclusiones obtenidos durante el desarrollo del proyecto.

Dentro de las implementaciones, si bien es no es una versión definitiva, sienta una base sólida para futuras expansiones una vez que se cumplan los requisitos previos, como la finalización del prototipo. A pesar de ello, las funcionalidades actuales ya disponibles proporcionan un marco robusto que permite gestionar y visualizar los procesos de trazabilidad de manera efectiva, lo que facilita la operación del sistema para los usuarios a través de una interfaz simple e intuitiva.

Con la solución propuesta, se ha logrado cumplir con los objetivos definidos al inicio del proyecto. Las nuevas funcionalidades propuestas durante las discusiones con la organización aportaron un valor agregado a la gestión de los procesos de la arquitectura. Se llevaron a cabo pruebas exhaustivas en *postman* y a través de encuestas las cuales indicaron una buena aceptación en términos de usabilidad y rendimiento. De igual forma, a lo largo de las encuestas fue notable el que todavía existe un margen de mejora en cuanto a la experiencia de usuario a través de futuras funciones e integraciones.

En conclusión, el proyecto ha representado una oportunidad para aplicar conocimientos técnicos en un entorno práctico y desafiante, permitiendo al autor obtener lecciones sobre la importancia de la planificación arquitectónica, integración de tecnologías y de un correcto flujo de ideas. Siendo este último lo que permitió adaptar el desarrollo a de las necesidades de la organización, maximizando así el valor del producto final. Además, el proyecto proporcionó una experiencia enriquecedora en cuanto a la colaboración, la capacidad de adaptación y el enfoque en el desarrollo de software, fortaleciendo habilidades esenciales para futuros desafíos profesionales.

Planes futuros

Entre los planes futuros, se contempla el despliegue completo de la plataforma web y la base de datos, pasos originalmente previstos en el proyecto, pero pospuestos a solicitud de la empresa debido a circunstancias imprevistas.

Uno de los principales enfoques de desarrollo a futuro es la mejora de la funcionalidad GPS. Aunque la sincronización actual con Google Maps es funcional, se espera que esta herramienta se convierta en una de las más utilizadas por los clientes de Alma Ltda., por lo que su optimización será clave para ofrecer un servicio más robusto.

Asimismo, se planea mejorar la interfaz con el objetivo de incrementar el dinamismo y reducir los tiempos de respuesta durante las monitorizaciones operacionales. Esto se logrará mediante la implementación de WebSocket, lo que permitirá alertar al usuario en tiempo real sobre las notificaciones emitidas por el sistema IOT. Además, se integrará un sistema de correos más robusto que, podrá seguir enviando alertas a usuarios que no se encuentren conectados a la plataforma.

En cuanto a las propuestas sugeridas por los usuarios durante la encuesta, aunque muchas estuvieron orientadas hacia mejoras estéticas, también se identificaron varias funcionalidades que podrían agregar valor significativo al producto final. Entre las recomendaciones seleccionadas se destacan:

- Función de exportación de registros de viajes en un archivo de texto, proporcionando un reporte final detallado.
- Filtros por placas en la vista de viajes vigentes y en el historial de viajes para facilitar la búsqueda y gestión.
- Implementación de criterios más estrictos para el cambio de contraseña, alineados con los estándares actuales de seguridad.

Estas mejoras y adiciones se estima que fortalecerán el producto, haciéndolo más eficiente, seguro y alineados con los estándares actuales de seguridad.

Referencias

[1] Learn - FastAPI. (s. f.). https://fastapi.tiangolo.com/learn/

[2] GeeksforGeeks. (2024, 19 febrero). *MVC Design pattern*. GeeksforGeeks. https://www.geeksforgeeks.org/mvc-design-pattern/

[3] Silverschatz A., Korth H.,Sudarshan S., "Fundamentos de Bases de Datos", 5tha edición. McGraw Hill, 2006.

[4] *Install Tailwind CSS with Create React App - Tailwind CSS*. (s. f.). Tailwind CSS. https://tailwindcss.com/docs/guides/create-react-app

[5] InfSeg. (2021, 2 noviembre). ASGI el futuro de Python en los servidores web - InfSeg. *InfSeg*. https://infseg.com/informatica/asgi-el-futuro-de-python/

[6] *Fetch API - referencia de la API Web | MDN*. (2024, 28 julio). MDN Web Docs. https://developer.mozilla.org/es/docs/Web/API/Fetch_API

[7] *Request Body - FastAPI*. (s. f.). https://fastapi.tiangolo.com/tutorial/body/

[8] *CORS (Cross-Origin Resource Sharing) - FastAPI*. (s. f.). https://fastapi.tiangolo.com/tutorial/cors/#wildcards

[9] *SQL (Relational) Databases - FastAPI*. (s. f.). https://fastapi.tiangolo.com/tutorial/sql-databases/

[10] *Models - Pydantic*. (s. f.). https://docs.pydantic.dev/latest/concepts/models/#basic-model-usage

[11] *Use scripts to add logic and tests to Postman requests | Postman Learning Center*. (2024, 21 junio). Postman Learning Center. https://learning.postman.com/docs/tests-and-scripts/write-scripts/intro-to-scripts/

[12] *HTTP response status codes - HTTP | MDN*. (2024, 25 julio). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

[13] Muguira, A. (2024, 6 junio). *¿Qué es la escala de Likert y cómo utilizarla?* QuestionPro. https://www.questionpro.com/blog/es/que-es-la-escala-de-likert-y-como-utilizarla/#:~:text=La%20escala%20de%20Likert%20asume,las%20actitudes%20pueden%20ser%20medidas.

[14] *Port forwarding local services with VS Code*. (2021, 3 noviembre). https://code.visualstudio.com/docs/editor/port-forwarding

[15] *What is Trello: Learn Features, Uses & More | Trello*. (s. f.). https://trello.com/tour

[16] Busquets, C., & Busquets, C. (2021, 24 diciembre). *Medir la usabilidad con el Sistema de Escalas de Usabilidad (SUS)*. uiFromMars. https://www.uifrommars.com/como-medir-usabilidad-que-es-sus/

[17] Localizar el mapa. (s. f.). Google For Developers. https://developers.google.com/maps/documentation/javascript/localization?hl=es-419