UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

# Desarrollo de un sistema de geolocalización de aviones en tiempo real

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Forment Reina, Óscar

Tutor/a: Hernández Orallo, Enrique

CURSO ACADÉMICO: 2023/2024

# Resum

Aquest Treball de Fi de Grau aborda el desenvolupament d'una aplicació web destinada a la visualització i la gestió d'informació d'aeronaus en temps real. L'objectiu principal del projecte és proporcionar una eina que permeti als usuaris accedir a dades detallades sobre vols i avions, com el nom de l'aerolínia, el tipus d'aeronau, el nombre de motors i una imatge de l'avió, tot això una interfície web intuïtiva i fàcil dutilitzar.

Per resoldre el problema plantejat, es va utilitzar una arquitectura basada en un backend desenvolupat a Flask i un frontend construït amb React. El backend s'encarrega de gestionar les sol·licituds a una API externa, processar les dades rebudes i emmagatzemar-les en una base de dades SQLite per facilitar l'accés ràpid a la informació. El frontend s'encarrega de presentar aquestes dades a l'usuari de manera clara i accessible, cosa que permet la interacció en temps real amb la informació disponible.

El projecte va incloure una fase de disseny on es van definir els requisits del sistema, es va escollir l'arquitectura més adequada i es va dissenyar una base de dades eficient per manejar la informació de les aeronaus. Posteriorment, es van implementar les funcionalitats necessàries per obtenir, emmagatzemar i mostrar les dades seguides d'una sèrie de proves exhaustives per assegurar la qualitat i el rendiment del sistema.

Les proves realitzades van confirmar que l'aplicació compleix els requisits funcionals, ofereix un rendiment adequat sota càrrega i proporciona una experiència d'usuari satisfactòria. A més, l'ús d'eines com ara GitHub per a la gestió de versions i Ngrok per a la implementació en un domini públic va facilitar el desenvolupament i la distribució de l'aplicació.

Aquest projecte no només va complir els objectius inicials, sinó que també va demostrar ser una solució robusta i escalable per visualitzar informació d'aeronaus. L'experiència adquirida en el desenvolupament web i la gestió de dades serà valuosa per a projectes futurs, tant acadèmics com professionals.

**Paraules clau:** avió, aeronau, pàgina web, ads-b

---

# Resumen

Este Trabajo de Fin de Grado aborda el desarrollo de una aplicación web destinada a la visualización y gestión de información de aeronaves en tiempo real. El objetivo principal del proyecto es proporcionar una herramienta que permita a los usuarios acceder a datos detallados sobre vuelos y aviones, como el nombre de la aerolínea, el tipo de aeronave, el número de motores y una imagen del avión, todo ello a través de una interfaz web intuitiva y fácil de usar.

Para resolver el problema planteado, se utilizó una arquitectura basada en un backend desarrollado en Flask y un frontend construido con React. El backend se encarga de gestionar las solicitudes a una API externa, procesar los datos recibidos, y almacenarlos en una base de datos SQLite para facilitar el acceso rápido a la información. El frontend se encarga de presentar estos datos al usuario de

manera clara y accesible, permitiendo la interacción en tiempo real con la información disponible.

El proyecto incluyó una fase de diseño en la que se definieron los requisitos del sistema, se escogió la arquitectura más adecuada y se diseñó una base de datos eficiente para manejar la información de las aeronaves. Posteriormente, se implementaron las funcionalidades necesarias para obtener, almacenar y mostrar los datos, seguidas de una serie de pruebas exhaustivas para asegurar la calidad y el rendimiento del sistema.

Las pruebas realizadas confirmaron que la aplicación cumple con los requisitos funcionales, ofrece un rendimiento adecuado bajo carga y proporciona una experiencia de usuario satisfactoria. Además, el uso de herramientas como GitHub para la gestión de versiones y Ngrok para la implementación en un dominio público facilitó el desarrollo y la distribución de la aplicación.

Este proyecto no solo cumplió con los objetivos iniciales, sino que también demostró ser una solución robusta y escalable para la visualización de información de aeronaves. La experiencia adquirida en el desarrollo web y la gestión de datos será valiosa para futuros proyectos, tanto académicos como profesionales.

**Palabras clave:** avión, aeronave, página web, ads-b

# Abstract

This Final Degree Project addresses the development of a web application for the visualization and management of aircraft information in real time. The main objective of the project is to provide a tool that allows users to access detailed data about flights and aircraft, such as the name of the airline, the type of aircraft, the number of engines and an image of the aircraft, all through an intuitive and easy-to-use web interface.

To solve the problem posed, an architecture based on a backend developed in Flask and a frontend built with React was used. The backend is responsible for managing requests to an external API, processing the data received, and storing it in a SQLite database to facilitate quick access to the information. The frontend is responsible for presenting this data to the user in a clear and accessible way, allowing real-time interaction with the available information.

The project included a design phase in which the system requirements were defined, the most appropriate architecture was chosen and an efficient database was designed to manage aircraft information. Subsequently, the necessary functionalities to obtain, store and display the data were implemented, followed by a series of exhaustive tests to ensure the quality and performance of the system.

The tests carried out confirmed that the application meets the functional requirements, offers adequate performance under load and provides a satisfactory user experience. In addition, the use of tools such as GitHub for version management and Ngrok for deployment in a public domain facilitated the development and distribution of the application.

This project not only met the initial objectives, but also proved to be a robust and scalable solution for the visualization of aircraft information. The experience

gained in web development and data management will be valuable for future projects, both academic and professional.

**Key words:** plane, aircraft, web page, ads-b

# Índice general

# Índice de figuras

# CAPÍTULO 1
# Introducción

En los tiempos actuales, el transporte aéreo se ha vuelto indispensable. Sectores importantes en nuestras vidas como el turismo o la logística se encuentran en estos momentos en completa dependencia del uso de aeronaves. La aviación comercial permite una conexión rápida y cómoda entre los distintos continentes, permitiendo desde que familias puedan explorar el mundo a un módico precio hasta que grandes empresas puedan transportar mercancías en grandes cantidades y a gran velocidad. Sin embargo, el crecimiento exponencial del tráfico aéreo ha generado nuevos desafíos que demandan soluciones innovadoras para garantizar la eficiencia, la seguridad y la sostenibilidad de este sector.

Detrás de cada vuelo que ves llegar y salir de los aeropuertos, detrás de cada avión que ves al alzar la mirada al cielo hay toda una compleja red de información compartida entre aerolíneas, controladores aéreos y sistemas de navegación. Toda esa red es la encargada de lograr que esas aeronaves sean capaces de llegar a su destino de manera segura y en el tiempo planificado. Sin embargo, la gestión de esta red es una tarea titánica que necesita de datos precisos y en tiempo real sobre todos y cada uno de los vuelos.

Es aquí de donde surge el problema: acceder a información precisa y actual sobre los aviones. Esa información es de gran importancia para el correcto funcionamiento de la aviación y es completamente indispensable para muchos actores que forman parte, desde controladores aéreos hasta gestores de aeropuertos o aerolíneas. Sin embargo, obtener este tipo de información puede requerir de altas tecnologías difíciles de encontrar o con un coste muy elevado de adquirir.

Este proyecto nace con el objetivo de permitir el acceso a información en tiempo real centrándose en mantener un coste mínimo. Con el único gasto de una antena y el acceso a un ordenador este proyecto pretende permitir que cualquier persona, incluso un simple fanático de la aviación, pueda tener acceso a información en tiempo real y que ésta sea mostrada de una manera amable y fácil de comprender. Todo esto con el fin de dar una mayor visibilidad a este apartado un poco más oculto y atraer a posibles mentes a mejorar la aviación con mejores técnicas y tecnologías que mantengas un bajo coste pero garanticen una buena relación con el medio ambiente.

Este proyecto, por tanto, no es solo una solución técnica; es una herramienta diseñada para mejorar un sector que afecta a millones de personas todos los días,

contribuyendo a la eficiencia, seguridad, y sostenibilidad del transporte aéreo en un mundo cada vez más interconectado.

## 1.1 Motivación

### 1.1.1. Motivación personal

Al final de mi paso por la universidad tuve la oportunidad de participar en unas prácticas de empresa como analista empresarial en una empresa que ofrece software como servicio (SaaS) basado en una página web y me encontré con el desarrollo web. Esta experiencia me abrió al mundo del desarrollo web y me permitió ver de primera mano la facilidad de hacer uso de un software desde cualquier máquina sin necesidad de tener nada instalado más que un navegador. Este interés por el desarrollo web me impulsó a querer explorar a fondo esta nueva área de la informática a la que nunca le había dedicado esfuerzo.

Al decidir sobre el tema de mi Trabajo de Fin de Grado (TFG), tenía claro que no quería que fuese relacionado con la IA que había estudiado durante mi camino por la rama de Computación puesto que no iba a ser una experiencia que fuera a disfrutar realizando, porque, para mí, mi principal consideración a la hora de realizar el TFG era hacer un proyecto con el que podía disfrutar. Entonces, cuando vi la oportunidad de elegir un proyecto sobre desarrollar un sistema de geolocalización de aviones no lo dudé, y elegí realizar una página web que permitiera observar los aviones que volaban sobre mi cabeza. Este proyecto me permite introducirme en este mundo del desarrollo web que había descubierto recientemente, alineando mi proyecto académico con mi actual trayectoria laboral.

Este proyecto también representa una oportunidad para expandir mis habilidades técnicas más allá de lo que aprendí en la rama de Computación, permitiéndome consolidar mi experiencia en desarrollo web y un poco de bases de datos. El enfoque en la creación de una aplicación robusta y escalable, con un backend eficiente y una interfaz de usuario intuitiva, está directamente relacionado con las necesidades actuales del mercado tecnológico.

### 1.1.2. Motivación del proyecto

El desarrollo de este proyecto surge de la necesidad de mejorar la accesibilidad y eficiencia en la gestión y visualización de datos relacionados con el seguimiento y monitoreo de vuelos en tiempo real. En un mundo cada vez más globalizado y conectado, el tráfico aéreo juega un papel crucial en la economía y en la vida diaria de millones de personas. Sin embargo, la información sobre vuelos, aviones y rutas no siempre es fácilmente accesible o se presenta de una manera que sea intuitiva y útil para el usuario final.

Además, la creciente importancia de las tecnologías web y el acceso a datos en tiempo real han creado una oportunidad para innovar en la forma en que se presenta y se interactúa con esta información. El proyecto, por lo tanto, se centra en crear una aplicación web que no solo agregue y muestre datos sobre vuelos y ae-

ronaves de manera eficiente, sino que también lo haga de una forma visualmente atractiva y fácil de usar.

La motivación del proyecto radica en la creación de una herramienta que facilite el acceso a información relevante sobre el tráfico aéreo, mejorando así la experiencia del usuario y aportando una solución a la fragmentación y dificultad de acceso a datos actualmente existentes en este campo. Esta solución no solo tiene el potencial de beneficiar a la industria aeronáutica, sino también de servir como recurso educativo y de interés para el público general.

## 1.2 Objetivos

El principal objetivo de este proyecto es desarrollar una aplicación web accesible y económica que permita mostrar información precisa y actualizada sobre aviones, con el propósito de inspirar a otros profesionales de la informática a explorar y contribuir al campo de la aviación. Este proyecto busca demostrar que es posible crear herramientas avanzadas en este ámbito sin necesidad de contar con grandes recursos financieros o conocimientos especializados en desarrollo web.

### 1.2.1. Objetivos Generales

1. Desarrollar una aplicación web funcional que permita visualizar datos detallados sobre aviones en tiempo real, utilizando un decodificador con antena como única inversión adicional, sin depender de tecnologías complejas o costosas.

2. Promover la accesibilidad a la tecnología en la aviación al demostrar que la creación de aplicaciones útiles en este sector está al alcance de personas con recursos limitados y conocimientos básicos en programación web.

### 1.2.2. Objetivos Específicos

1. Implementar una base de datos para almacenar y gestionar la información recibida sobre los aviones, permitiendo así reducir las consultas repetitivas a la API externa y mejorar la eficiencia de la aplicación.

2. Configurar una infraestructura mínima que permita la publicación y acceso público de la aplicación haciendo uso de la herramientas necesarias, con el objetivo de mostrar que es posible compartir proyectos de este tipo sin necesidad de servidores dedicados.

3. Garantizar el cumplimiento de las licencias al integrar imágenes y otros recursos multimedia en la aplicación, respetando los derechos de autor y fomentando prácticas responsables en el desarrollo web.

4. Facilitar la reutilización y la extensión del proyecto mediante una estructura de código clara y modular, para que otros desarrolladores puedan adaptar o mejorar la aplicación según sus necesidades.

5. Fomentar la innovación y la participación en la aviación a través de la divulgación de este proyecto, mostrando que cualquier persona interesada, independientemente de su nivel económico o técnico, puede contribuir al avance de la tecnología en este campo.

# CAPÍTULO 2
# Contexto

El campo de la aviación ha experimentado una evolución tecnológica significativa en las últimas décadas, con avances notables en la recopilación, procesamiento y visualización de datos de vuelo en tiempo real. Este progreso ha sido facilitado por una serie de aplicaciones y herramientas que han permitido tanto a profesionales como a entusiastas del sector obtener información precisa y detallada sobre el tráfico aéreo.

Este proyecto no solo busca replicar funcionalidades existentes, sino también explorar nuevas formas de hacer que la tecnología de monitorización de vuelos sea más accesible y asequible para todos, sin sacrificar la precisión y la utilidad de la información proporcionada.

Antes de empezar es necesario comprender que son los mensajes ADS-B que se van a utilizar en el proyecto y que utilizan el resto de aplicaciones existentes.

### ADS-B

Los mensajes ADS-B (Automatic Dependent Surveillance-Broadcast) son parte de un sistema de vigilancia basado en satélites que permite a las aeronaves transmitir información como su posición, velocidad e identificación a través del protocolo Mode S Extended Squitter (1090 MHz). La mayoría de las aeronaves modernas emiten constantemente estos mensajes, y desde 2020, en Europa y Estados Unidos, es obligatorio que las aeronaves civiles cumplan con este estándar, siendo necesario modificar o retirar del servicio, después de un plazo determinado, aquellas aeronaves más antiguas que no cumplan con este requisito.

Un mensaje ADS-B tiene una estructura de 112 bits y se divide en cinco partes principales:

1. DF (Downlink Format, 5 bits): Indica el formato del mensaje, siendo el formato 17 el más común para los mensajes ADS-B.

2. CA (Transponder Capability, 3 bits): Indica el nivel de capacidad del transpondedor, con valores que varían entre 0 y 7.

3. ICAO (ICAO Aircraft Address, 24 bits): Es la dirección única asignada a cada aeronave según las regulaciones de la OACI.

4. ME (Message, 56 bits): Contiene la información específica, como identificación de la aeronave, posición, velocidad, etc.

5. PI (Parity/Interrogator ID, 24 bits): Sirve para verificar la integridad del mensaje.

El código ICAO en el mensaje identifica de manera única a la aeronave durante toda su vida útil, aunque en casos específicos, como en aviones militares, este código puede ser reprogramado.

Los tipos de mensajes ADS-B varían según el código de tipo (Type Code) dentro del mensaje, que determina la información que se transmite, como la identificación de la aeronave, posición en superficie o aire, velocidad, estado operativo, entre otros.

## 2.1 Aplicaciones Existentes

Uno de los referentes más conocidos en la monitorización de vuelos es Flightradar24, una plataforma que permite rastrear aviones en tiempo real utilizando datos de transpondedores ADS-B (Automatic Dependent Surveillance-Broadcast). Flightradar24 se ha consolidado como una herramienta fundamental tanto para el público en general como para profesionales de la aviación, proporcionando información detallada sobre el estado de los vuelos, la ubicación de los aviones, y datos técnicos como altitud, velocidad y ruta. La precisión y accesibilidad de Flightradar24 han establecido un estándar en la industria, aunque su modelo de negocio incluye planes de suscripción para acceder a funcionalidades avanzadas.

Otra aplicación destacada es FlightAware, que también ofrece seguimiento de vuelos en tiempo real, pero con un enfoque en la integración de múltiples fuentes de datos, incluidos radares, satélites y redes ADS-B, para proporcionar una cobertura global. FlightAware ofrece servicios a aerolíneas, aeropuertos y usuarios individuales, destacándose por su capacidad para proporcionar análisis de datos históricos y predicciones de vuelo. Al igual que Flightradar24, FlightAware opera bajo un modelo freemium, lo que limita el acceso gratuito a ciertas funcionalidades avanzadas .

En el ámbito de las soluciones de código abierto, OpenSky Network es un proyecto colaborativo que permite a los usuarios contribuir con datos ADS-B y, a cambio, acceder a una base de datos pública de vuelos en tiempo real. Esta plataforma se distingue por su enfoque en la investigación y el desarrollo, proporcionando datos sin restricciones para el análisis y la mejora de los sistemas de tráfico aéreo .

## 2.2 Alternativas y Justificación del Camino Elegido

El desarrollo de este proyecto se ha basado en la premisa de que es posible crear una aplicación web funcional que proporcione datos de vuelos sin incurrir en altos costos o depender de tecnologías propietarias. Si bien Flightradar24 y

FlightAware ofrecen soluciones robustas y ampliamente utilizadas, su enfoque en modelos de suscripción y la dependencia de infraestructuras avanzadas puede limitar el acceso a usuarios con menos recursos económicos o técnicos.

Por otro lado, OpenSky Network, aunque accesible y basado en la comunidad, se orienta más hacia la investigación y puede no ser tan amigable para usuarios que buscan una solución sencilla y directa para visualizar datos de vuelos.

Este proyecto opta por una solución intermedia: utilizar un decodificador con antena para capturar datos ADS-B y procesarlos a través de una aplicación web diseñada con herramientas de código abierto, minimizando costos y complejidad técnica. La elección de SQLite como base de datos responde a la necesidad de una solución ligera y fácil de implementar, ideal para proyectos con limitaciones de recursos. Además, la publicación de la aplicación utilizando ngrok permite demostrar que es posible compartir los resultados sin necesidad de infraestructuras costosas como servidores dedicados.

## 2.3 Evolución Tecnológica y Perspectivas

El uso de datos ADS-B para la monitorización de vuelos ha evolucionado considerablemente desde su introducción. Inicialmente, estos datos estaban limitados a aplicaciones profesionales y gubernamentales, pero con el tiempo, han sido democratizados gracias a tecnologías accesibles como los decodificadores de bajo costo y plataformas colaborativas. Este proyecto busca contribuir a esta evolución al mostrar que es posible desarrollar herramientas de monitorización avanzadas con recursos mínimos, abriendo nuevas oportunidades para la innovación en la aviación, especialmente para aquellos con recursos limitados.

# CAPÍTULO 3
# Análisis del problema

El análisis del problema y la identificación de oportunidades en este proyecto permiten establecer un camino claro para el desarrollo de una aplicación accesible y eficiente para la visualización de datos de aviones en tiempo real. Al atender cuidadosamente los requisitos técnicos, de seguridad, legales y de eficiencia, el proyecto no solo logrará sus objetivos funcionales, sino que también servirá como una herramienta inspiradora y accesible para otros interesados en la intersección de la informática y la aviación.

## 3.1 Especificación de Requisitos

El proyecto propuesto tiene como objetivo el desarrollo de una aplicación web para la visualización de datos en tiempo real de aviones utilizando tecnología accesible y de bajo costo. Para asegurar el éxito del proyecto, es fundamental establecer requisitos claros y específicos que guíen el desarrollo del sistema.

### 3.1.1. Requisitos Funcionales

Captura de Datos ADS-B:

- La aplicación debe ser capaz de recibir y procesar datos ADS-B utilizando un decodificador con antena.
- Los datos deben incluir información clave como la altitud, velocidad, posición, tipo de avión, compañía aérea, y otros detalles técnicos del vuelo.

Almacenamiento de Datos:

- Los datos recibidos deben ser almacenados en una base de datos local (SQLite) para su posterior análisis y consulta.
- La base de datos debe ser capaz de manejar un volumen significativo de datos con eficiencia.

Visualización en Tiempo Real:

- La aplicación debe permitir a los usuarios visualizar los aviones en tiempo real en un mapa interactivo.
- Los usuarios deben poder acceder a información detallada sobre cada avión seleccionado, incluyendo su modelo y detalles técnicos.

Interfaz de Usuario:

- La aplicación debe ser accesible a través de un navegador web, con una interfaz intuitiva y fácil de usar.
- La interfaz debe permitir a los usuarios filtrar y buscar vuelos específicos, así como personalizar la visualización de los datos.

Publicación Web:

- La aplicación debe ser accesible públicamente a través de la web utilizando herramientas como ngrok para compartir la aplicación sin necesidad de un servidor dedicado.

### 3.1.2. Requisitos No Funcionales

Rendimiento:

- La aplicación debe ser capaz de procesar y visualizar los datos en tiempo real sin retrasos significativos.
- El sistema debe ser optimizado para minimizar el uso de recursos del servidor y el tiempo de respuesta.

Seguridad:

- Los datos almacenados deben ser protegidos contra accesos no autorizados, y las comunicaciones deben ser seguras.

Escalabilidad:

- Aunque el sistema está diseñado para un uso personal o de pequeña escala, debe poder escalarse para manejar un mayor volumen de datos si es necesario.

Portabilidad:

- La aplicación debe ser fácilmente desplegable en cualquier entorno que soporte Python y SQLite, sin necesidad de configuraciones complejas.

## 3.2 Análisis de Seguridad

La seguridad es un aspecto crucial en este proyecto debido a la naturaleza de los datos que se manejan, aunque no sean de carácter personal, la integridad y la disponibilidad de los datos son esenciales para el correcto funcionamiento de la aplicación.

- Protección de Datos: Aunque los datos no son personales, es importante asegurar que no se manipulen o se pierdan durante la transmisión o almacenamiento. Se deben implementar medidas para evitar inyecciones SQL y asegurar que la aplicación no sea vulnerable a ataques comunes.
- Acceso Restringido: El acceso a funciones críticas de la aplicación, como la modificación de la configuración del sistema, debe estar protegido mediante autenticación y autorización adecuadas.

## 3.3 Análisis de Eficiencia

Dado que la aplicación está diseñada para ejecutarse en un entorno de hardware limitado (como una Raspberry Pi), la eficiencia algorítmica y energética es crucial.

- Optimización del Procesamiento de Datos: El procesamiento de datos ADS-B debe ser eficiente para no sobrecargar el sistema. Esto implica el uso de algoritmos optimizados para la decodificación y almacenamiento de datos.
- Uso de Recursos: Se debe minimizar el uso de CPU y memoria, especialmente durante el procesamiento en tiempo real, para garantizar que la aplicación pueda ejecutarse de manera sostenible durante períodos prolongados.

## 3.4 Análisis del Marco Legal y Ético

El proyecto también debe considerar los aspectos legales y éticos relacionados con la recolección, almacenamiento y publicación de datos.

- Protección de Datos: Aunque los datos ADS-B no contienen información personal, es importante cumplir con regulaciones locales sobre la publicación de datos en tiempo real, especialmente si se comparte públicamente en la web.
- Propiedad Intelectual: El desarrollo de la aplicación debe respetar las licencias de software de terceros utilizados, como bibliotecas y frameworks. El proyecto en sí puede ser distribuido bajo una licencia de código abierto para fomentar su adopción y modificación por parte de la comunidad.

- Ética: Publicar información sobre aviones en tiempo real implica ciertas responsabilidades éticas. Se debe garantizar que la aplicación no facilite usos indebidos de la información, como su empleo para fines malintencionados.

## 3.5 Análisis de Riesgos

El proyecto también enfrenta ciertos riesgos que deben ser gestionados adecuadamente:

Riesgos Tecnológicos:

- Integración de Hardware y Software: Existe el riesgo de que el decodificador ADS-B no funcione correctamente con la aplicación, lo que podría afectar la captura de datos.
- Compatibilidad: La aplicación debe ser compatible con diferentes versiones de sistemas operativos y navegadores.

Riesgos de Seguridad:

- Ataques Externos: Si la aplicación es accesible públicamente, puede ser susceptible a ataques externos como DDoS o intentos de hackeo.

Riesgos de Escalabilidad:

- Crecimiento de Usuarios: Si la aplicación se vuelve popular, el aumento en el número de usuarios y de datos procesados podría sobrecargar el sistema.

Riesgos de Aceptación:

- Adopción por la Comunidad: Existe el riesgo de que la comunidad objetivo no adopte la aplicación, lo que reduciría su impacto potencial.

Para mitigar estos riesgos, se deben implementar medidas como pruebas exhaustivas, implementación de mecanismos de seguridad robustos, y planificación para la escalabilidad desde la fase inicial de desarrollo.

## 3.6 Identificación y análisis de soluciones posibles

En este proyecto, cuyo objetivo es desarrollar una aplicación web accesible que muestre información precisa sobre aviones en tiempo real, se consideraron varias soluciones posibles para los componentes clave del sistema: el backend, la base de datos, el frontend y la publicación del sitio web. A continuación, se presentan las opciones evaluadas, con sus respectivos pros y contras, y se justifica la elección final para cada uno de estos componentes.

### 3.6.1. Soluciones para el Backend

**Solución 1: Flask + SQLite**

Flask es un microframework de Python, conocido por su simplicidad y flexibilidad. SQLite es una base de datos ligera que se integra bien con Flask, no requiere configuración de servidor y almacena la base de datos en un archivo local.

Pros:

- Simplicidad: Ideal para proyectos pequeños, permite un desarrollo rápido y sencillo.
- Portabilidad: SQLite es altamente portátil, lo que facilita la migración del proyecto.
- Curva de Aprendizaje: Flask y SQLite son accesibles para desarrolladores con conocimientos básicos de Python.

Contras:

- Escalabilidad Limitada: No es ideal para aplicaciones con alta concurrencia o grandes volúmenes de datos.
- Limitaciones de SQLite: Carece de algunas funcionalidades avanzadas que ofrecen otros sistemas de bases de datos.

**Solución 2: Django + PostgreSQL**

Django es un framework web completo que incluye muchas funcionalidades listas para usar, como autenticación y administración. PostgreSQL es una base de datos relacional robusta y escalable, adecuada para manejar aplicaciones más complejas.

Pros:

- Escalabilidad: Django y PostgreSQL manejan bien el crecimiento del tráfico y la base de datos.
- Funcionalidades Avanzadas: Django ofrece características de seguridad, administración y autenticación listas para usar.
- Comunidad y Soporte: Amplia comunidad y documentación, lo que facilita la resolución de problemas.

Contras:

- Curva de Aprendizaje: Más complejo que Flask, lo que podría ser un desafío para principiantes.
- Requisitos de Infraestructura: Necesita más recursos y una configuración más compleja que Flask y SQLite.

#### Solución 3: Node.js + MongoDB

Node.js es una plataforma basada en JavaScript que permite desarrollar tanto el frontend como el backend con el mismo lenguaje. MongoDB es una base de datos NoSQL que almacena datos en formato JSON, lo que lo hace ideal para aplicaciones que manejan grandes cantidades de datos no estructurados.

Pros:

- Unificación del Stack: Permite usar JavaScript tanto en el frontend como en el backend.
- Escalabilidad: MongoDB maneja grandes volúmenes de datos y puede escalar horizontalmente.
- Rapidez de Desarrollo: Node.js permite un desarrollo ágil y eficiente, especialmente en aplicaciones con alta interacción en tiempo real.

Contras:

- Complejidad en Modelado de Datos: MongoDB puede ser menos intuitivo para modelar datos relacionales.
- Requiere Más Conocimiento de JavaScript: Si el desarrollador no está familiarizado con JavaScript, puede haber una curva de aprendizaje considerable.

### 3.6.2. Soluciones para el Frontend

#### Solución 1: React

React es una biblioteca de JavaScript desarrollada por Facebook para construir interfaces de usuario. Es conocida por su enfoque basado en componentes, lo que permite crear interfaces dinámicas y reutilizables.

Pros:

- Interactividad: Permite crear interfaces de usuario dinámicas y altamente interactivas.
- Reutilización de Componentes: Fomenta la modularidad y el mantenimiento del código.
- Amplio Ecosistema: Soporte de una comunidad activa y numerosas bibliotecas adicionales.

Contras:

- Curva de Aprendizaje: Puede ser más complejo que otros frameworks si se incluyen características avanzadas.
- Sobrecarga para Proyectos Pequeños: React puede ser excesivo si la aplicación no requiere una interfaz de usuario muy dinámica.

**Solución 2: Vue.js**

Vue.js es un framework progresivo de JavaScript que se enfoca en la creación de interfaces de usuario y aplicaciones de una sola página. Es más fácil de integrar en proyectos existentes que React o Angular.

Pros:

- Facilidad de Integración: Vue.js es sencillo de integrar en aplicaciones existentes o nuevas.
- Documentación y Comunidad: Excelente documentación y una comunidad activa.
- Curva de Aprendizaje Suave: Más accesible para principiantes en comparación con otros frameworks como Angular.

Contras:

- Menos Adoptado en la Industria: Aunque popular, Vue.js tiene menos adopción en grandes empresas en comparación con React o Angular.
- Ecosistema Más Pequeño: Menor cantidad de herramientas y complementos en comparación con React.

**Solución 3: Angular**

Angular es un framework completo de desarrollo frontend mantenido por Google, que permite la creación de aplicaciones web complejas.

Pros:

- Arquitectura Completa: Ofrece una solución integral con todo lo necesario para desarrollar aplicaciones frontend complejas.
- Soporte Empresarial: Amplio soporte y uso en aplicaciones de nivel empresarial.
- Rich Features: Ofrece características como enrutamiento avanzado, servicios de inyección de dependencias y formularios reactivos.

Contras:

- Complejidad y Curva de Aprendizaje: Es más complejo que React o Vue.js, lo que podría ser desalentador para proyectos pequeños o para desarrolladores principiantes.
- Carga Inicial: Requiere más configuración y tiene un tamaño mayor que otras soluciones como Vue.js.

### 3.6.3. Soluciones para la Publicación del Sitio Web

**Solución 1: Ngrok**

Ngrok es una herramienta que permite exponer un servidor local a la web mediante la creación de un túnel seguro, haciendo que el sitio web sea accesible a través de una URL pública temporal.

Pros:

- Facilidad de Uso: Configuración sencilla y rápida, ideal para pruebas y demostraciones.
- Portabilidad: Funciona en cualquier máquina local sin necesidad de una infraestructura compleja.
- Acceso Público Temporal: Ideal para compartir el desarrollo en curso sin necesidad de un dominio permanente.

Contras:

- Seguridad: El uso de Ngrok para exponer servidores locales puede ser un riesgo si no se toman medidas de seguridad adecuadas.
- Desconexión y Límites de la Versión Gratuita: La versión gratuita de Ngrok tiene limitaciones en la duración de las conexiones y en el ancho de banda.
- No Adecuado para Producción: No es una solución viable para producción a largo plazo.

**Solución 2: AWS EC2 + Nginx**

AWS EC2 permite desplegar instancias de servidores en la nube, y Nginx puede ser utilizado como servidor web y proxy inverso para gestionar el tráfico entrante.

Pros:

- Escalabilidad y Flexibilidad: AWS ofrece recursos elásticos que pueden escalar según la demanda.
- Seguridad y Confiabilidad: AWS es una plataforma segura y confiable para el despliegue de aplicaciones en producción.
- Personalización: Nginx permite una configuración personalizada para optimizar el rendimiento y la seguridad.

Contras:

- Costos: Aunque escalable, AWS EC2 puede ser costoso en comparación con soluciones más sencillas como Ngrok.

- Curva de Aprendizaje: Configurar correctamente EC2 y Nginx puede ser complejo, especialmente para principiantes.
- Requiere Mantenimiento: Necesita una gestión continua para asegurar la disponibilidad y la seguridad.

**Solución 3: GitHub Pages + Netlify**

Para proyectos estáticos o con backend separado, GitHub Pages permite publicar directamente desde un repositorio de GitHub, y Netlify ofrece una plataforma de despliegue continua que maneja automáticamente el frontend.

Pros:

- Gratuito y Fácil de Usar: GitHub Pages es gratuito y fácil de configurar, ideal para proyectos estáticos.
- Despliegue Continuo: Netlify facilita la automatización del despliegue con cada cambio en el repositorio.
- Dominio Personalizado: Ambas plataformas permiten configurar dominios personalizados sin complicaciones.

Contras:

- Limitado a Sitios Estáticos: GitHub Pages está limitado a sitios estáticos, por lo que es necesario un backend separado.
- Dependencia de Servicios Externos: Ambas soluciones dependen de la infraestructura de GitHub y Netlify.
- Integración Compleja con Backend Dinámico: Requiere configuración adicional para trabajar con un backend que no sea estático.

## 3.7 Criterio de Selección

La selección de las soluciones finales se basó en los siguientes criterios:

- Simplicidad y Facilidad de Implementación: El objetivo es hacer que la aplicación sea accesible y replicable por personas con conocimientos básicos, por lo que se priorizan herramientas que sean fáciles de usar e implementar.
- Costo y Accesibilidad: Se buscó minimizar los costos, utilizando herramientas gratuitas o de bajo costo que no requieran infraestructura costosa.
- Escalabilidad y Flexibilidad: Aunque el proyecto es pequeño, se consideraron soluciones que puedan escalar si el tráfico o los datos aumentan en el futuro.

- Adecuación al Proyecto: Las soluciones deben alinearse con los requisitos específicos del proyecto, permitiendo una experiencia de usuario fluida y una implementación sencilla.

## 3.8 Solución propuesta

Backend con Flask y SQLite: Se eligió Flask por su simplicidad y facilidad de uso, y SQLite por su portabilidad. Esta combinación es suficiente para el nivel de tráfico y volumen de datos esperado en este proyecto.

Frontend con React: React fue seleccionado debido a su capacidad para crear interfaces de usuario dinámicas y su amplia comunidad, lo que facilita encontrar soluciones y recursos adicionales.

Publicación con Ngrok: Ngrok fue elegido por su facilidad de configuración y su utilidad para pruebas y demostraciones rápidas, a pesar de sus limitaciones en seguridad y conectividad.

La combinación de Flask, React y Ngrok se ha seleccionado como la solución óptima para este proyecto. Estas herramientas permiten desarrollar una aplicación web completa y funcional que cumple con los objetivos de accesibilidad, bajo costo y facilidad de uso. Además, ofrecen suficiente flexibilidad para que el proyecto pueda ser replicado y adaptado por otros desarrolladores, contribuyendo a la difusión del conocimiento y la inspiración en el campo de la aviación.

# CAPÍTULO 4
# Diseño de la solución

## 4.1 Arquitectura del Software

El diseño de la solución para este proyecto se estructura en varias capas, siguiendo una arquitectura típica de aplicaciones web basada en el patrón Model-View-Controller (MVC). Este enfoque permite una separación clara de responsabilidades, lo que facilita el mantenimiento, la escalabilidad y la reutilización de componentes.

La arquitectura propuesta combina tecnologías modernas y patrones de diseño probados para crear una solución robusta y eficiente. La clara separación de responsabilidades entre el frontend, el backend y la base de datos asegura que el sistema sea fácil de mantener y expandir. Esta arquitectura no solo satisface los requisitos actuales del proyecto, sino que también proporciona una base sólida para futuros desarrollos y mejoras.

El diagrama representa los principales componentes del sistema y cómo interactúan entre sí. Los componentes esenciales son:

- Frontend: El cliente, desarrollado en React.js, que se encarga de la interfaz de usuario y la interacción con los usuarios. Este componente consume las API del backend para obtener y mostrar datos.
- Backend: Un servidor desarrollado en Flask (Python) que expone una serie de APIs para el manejo de la lógica de negocio y la interacción con la base de datos. Este componente se encarga de recibir peticiones del frontend, comprobar si la información necesaria se encuentra en la base de datos, si es así se procesa y se devuelve la información al frontend, si la información no se encuentra en la base de datos se hace una solicitud a la API externa de AeroDataBox, se procesa y se guarda la información en la base de datos para luego enviársela al frontend.
- Base de Datos: Una base de datos SQLite que almacena información relevante sobre los aviones, incluyendo detalles de cada aeronave y sus imágenes asociadas. Este componente se comunica directamente con el backend.
- API Externa: El sistema interactúa con la API de AeroDataBox que proporciona datos en tiempo real sobre vuelos y aeronaves. El backend se encarga

de gestionar las llamadas a esta API, almacenando la información necesaria en la base de datos y retornándola al frontend según sea requerido.

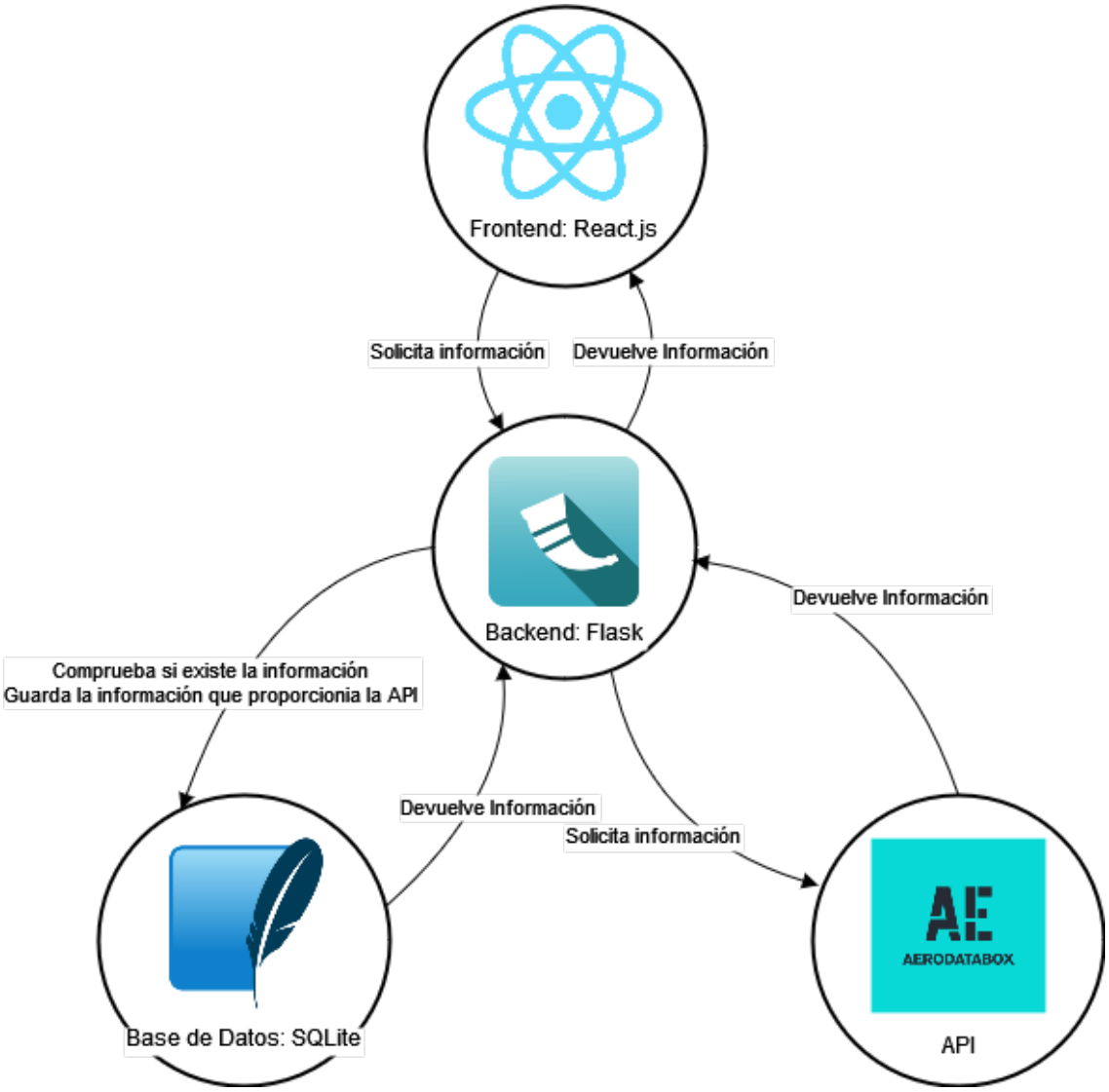



**Figura 4.1:** Diagrama Arquitectura del Software

## 4.2 Arquitectura del sistema

La arquitectura del sistema desarrollado para este proyecto se basa en una configuración que integra hardware especializado con componentes de software diseñados para capturar, procesar y presentar información de vuelos en tiempo real. A continuación, se describen los principales componentes de esta arquitectura:

**Antena ADS-B**

La antena especializada para ADS-B es crítica para recibir señales claras y estables de las aeronaves en la región. La calidad de la antena y su colocación afectan directamente la cantidad y la precisión de los datos capturados. La ante-

na está conectada al dongle USB RTL-SDR, amplificando y enviando las señales recibidas al dongle para su decodificación.

**Dongle USB RTL-SDR**

Se utiliza un dongle USB RTL-SDR (Software Defined Radio) conectado a la Raspberry Pi. Este dispositivo se encarga de capturar las señales de radiofrecuencia en la banda de 1090 MHz, que es la frecuencia utilizada para la transmisión de mensajes ADS-B por las aeronaves. El dongle, junto con la antena, permite que el sistema reciba estas señales y las transmita a la Raspberry Pi para su procesamiento.

**Raspberry Pi**

La Raspberry Pi actúa como el núcleo del sistema, proporcionando una plataforma de bajo costo y bajo consumo para la ejecución de los procesos necesarios. Esta mini computadora es responsable de la ejecución del software que decodifica las señales ADS-B y gestiona las solicitudes a la API externa.

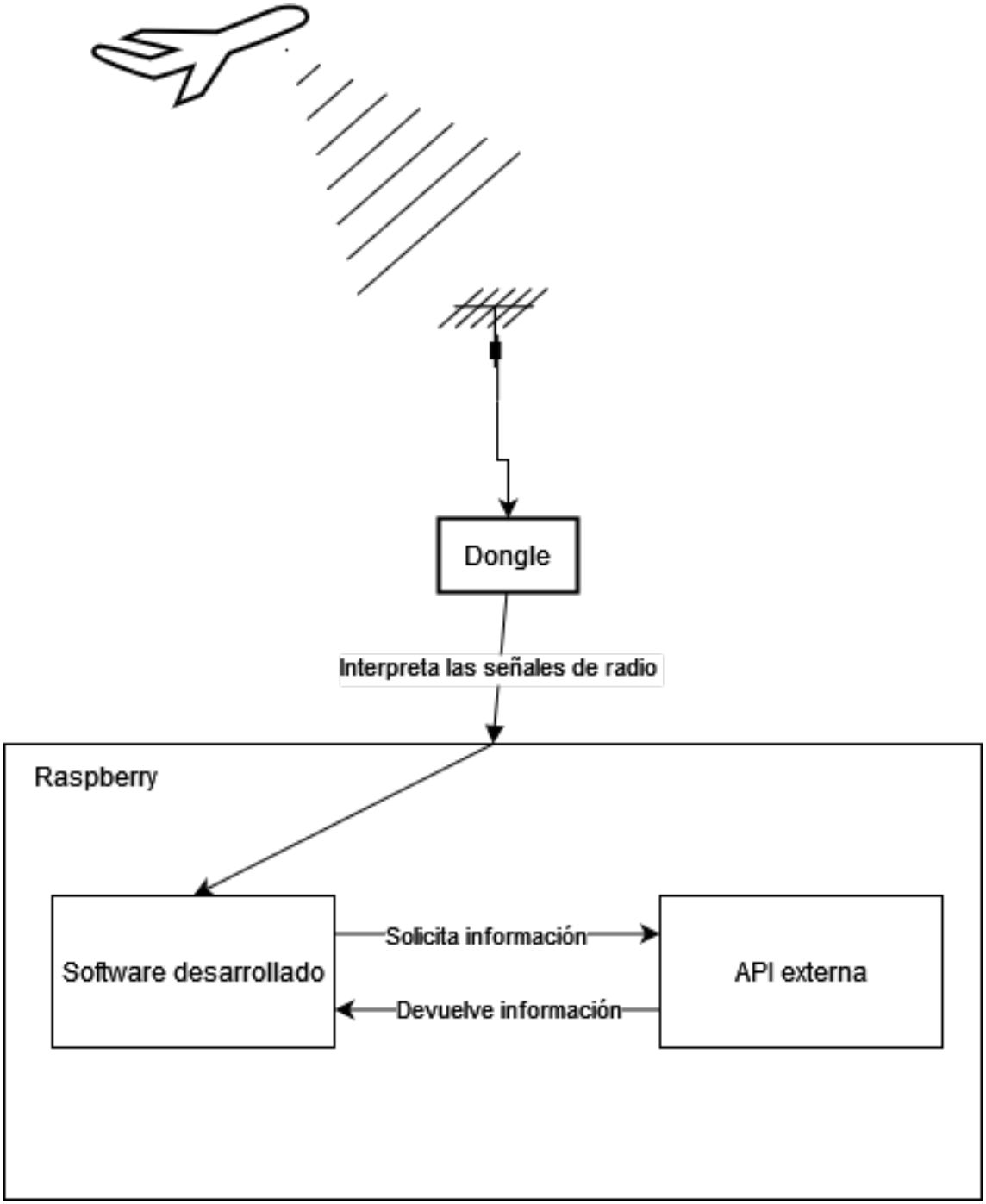



**Figura 4.2:** Diagrama Arquitectura del Sistema

## 4.3 Diseño Detallado

### 4.3.1. Diseño de la Base de Datos

El diseño de la base de datos se centra en la estructura de las tablas necesarias para almacenar la información de los aviones. El esquema de la base de datos incluye una única tabla denominada Aircraft, que contiene campos como id, flight_hex, airline_name, icao_code, type_name, model_code, num_seats,

num_engines, engine_type, age_years, y image. Sin embargo, esta base de datos puede crecer conforme crezca la aplicación y se podrían sumar otras tablas como una de aerolíneas que haga que el campo airline_name se vuelve una clave ajena.

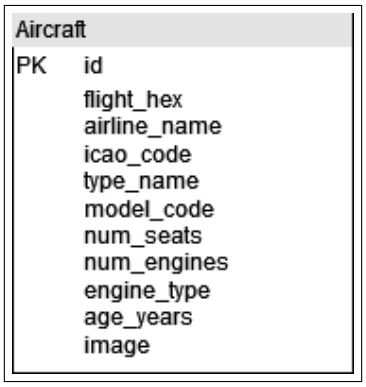



**Figura 4.3:** Diagrama Base de Datos

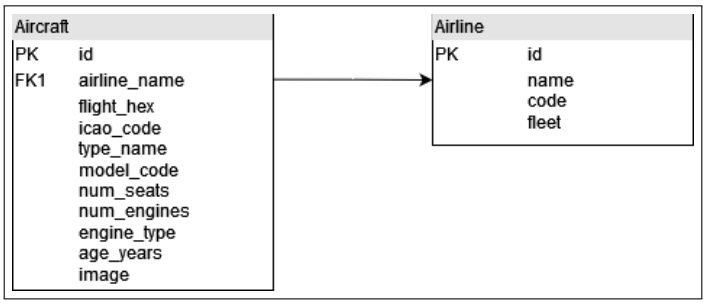



**Figura 4.4:** Diagrama Posible Futura Base de Datos

### 4.3.2. Diseño de Clases y Componentes

El diseño detallado también incluye la definición de las clases y módulos que constituyen la aplicación. En el backend, las clases principales incluyen:

Aircraft: Una clase que representa el modelo de datos de un avión, con atributos que corresponden a las columnas de la tabla Aircraft en la base de datos.

Controllers: Módulos que manejan las diferentes rutas de la aplicación, gestionando la lógica de negocio y sirviendo como intermediarios entre el frontend y la base de datos. Además también actúa de APIHandler encargándose de realizar las llamadas a la API externa, procesar las respuestas y almacenar la información relevante en la base de datos.

En el front actualmente hay un componente principal:

Map: Componente que se encarga de toda la interacción del usuario con el mapa interactivo, es el componente que se encarga de realizar las llamadas al backend y mostrar por pantalla la información obtenida

## 4.4 Tecnología utilizada

El desarrollo de este proyecto ha implicado la selección y utilización de diversas tecnologías, herramientas y frameworks que han facilitado la implementación y asegurado la calidad y eficiencia del producto final. La elección de estas tecnologías se ha basado en criterios como la compatibilidad, el coste de aprendizaje, la viabilidad del proyecto, el coste económico y el conocimiento previo. A continuación, se describen las principales tecnologías utilizadas.

### 4.4.1. Entorno de Desarrollo

Para llevar a cabo el desarrollo del proyecto, se utilizó una combinación de herramientas y entornos que proporcionan un ambiente eficiente y productivo:

#### Sistema operativo

El desarrollo se realizó en un entorno Linux, específicamente en una Raspberry Pi con Raspberry Pi OS Lite. Este entorno es liviano, eficiente, y compatible con una amplia gama de herramientas de desarrollo, lo que permite un desarrollo fluido y sin complicaciones. Además se utilizo una conexión ssh y samba para poder acceder a la raspberry y sus archivos desde otros disposivos conectados a la red.

#### Editor de texto

Se empleó Visual Studio Code, un editor de código multiplataforma que ofrece una experiencia de desarrollo rica en funciones gracias a su amplia gama de extensiones. VS Code permite depurar, gestionar versiones con Git, y utilizar terminales integrados, lo que facilita la codificación y prueba en tiempo real.

#### Decodificador de mensajes

Readsb fue elegido para este proyecto debido a su capacidad robusta y eficiente para decodificar las señales ADS-B (Automatic Dependent Surveillance-Broadcast) captadas por la antena y procesadas por el decodificador, que es esencial para obtener información precisa y en tiempo real sobre aviones. Es una herramienta de código abierto que ofrece un rendimiento excelente al manejar grandes volúmenes de datos de aviación, facilitando la recopilación y procesamiento de datos críticos para la visualización y análisis de vuelos en tiempo real.

### 4.4.2. Backend

El backend del proyecto se desarrolló utilizando tecnologías robustas y ampliamente adoptadas en la industria para garantizar una lógica de negocio sólida y escalable:

#### Lenguaje de Programación

Python fue elegido como el lenguaje principal para el desarrollo del backend debido a su simplicidad, amplia comunidad y la disponibilidad de numerosas bibliotecas que aceleran el desarrollo. Python es conocido por ser fácil de aprender y utilizar, lo que reduce el tiempo de desarrollo y depuración.

#### Framework Web

Flask se seleccionó como el framework para el desarrollo del backend. Flask es un microframework ligero para Python que proporciona la flexibilidad necesaria para construir aplicaciones web desde cero, sin imponer una estructura rígida. Esta flexibilidad fue clave para adaptar el backend a las necesidades específicas del proyecto.

#### Base de Datos

Se utilizó SQLite como sistema de gestión de base de datos (DBMS). SQLite es una base de datos relacional ligera, ideal para aplicaciones de tamaño medio que no requieren un servidor de base de datos completo. Su integración sencilla con Python mediante la biblioteca SQLAlchemy permitió manejar eficientemente el almacenamiento de datos sin necesidad de configuraciones complejas.

#### ORM (Object-Relational Mapping)

SQLAlchemy se utilizó como ORM para interactuar con SQLite. SQLAlchemy simplifica la manipulación de la base de datos al permitir que las operaciones con los datos se realicen mediante objetos de Python, facilitando la escritura y lectura de datos en la base de datos.

### 4.4.3. Frontend

El frontend del proyecto se construyó utilizando tecnologías modernas que permiten la creación de interfaces de usuario interactivas y responsivas:

#### Lenguaje de Programación

JavaScript fue seleccionado como el lenguaje para el frontend debido a su posición como el lenguaje nativo de los navegadores, su extensa compatibilidad con bibliotecas y frameworks como React.js, y su capacidad para crear interfaces de usuario interactivas y dinámicas.

#### Framework de Frontend

React.js fue elegido para el desarrollo del frontend. React.js es una biblioteca de JavaScript para construir interfaces de usuario. Su enfoque basado en compo-

nentes facilita la creación de interfaces modulares y reutilizables, lo que mejora la mantenibilidad del código y la experiencia del usuario.

**Biblioteca de Mapeo**

Leaflet fue seleccionado para este proyecto debido a su capacidad para ofrecer mapas interactivos y personalizables de manera eficiente y ligera. Es una biblioteca de código abierto ampliamente utilizada que se integra fácilmente con otras tecnologías web y ofrece una rica variedad de plugins y extensiones. Además, su simplicidad y flexibilidad permiten una rápida implementación de funcionalidades avanzadas, como la visualización de datos geoespaciales en tiempo real, lo cual es crucial para los objetivos del proyecto.

### 4.4.4. API Externa

El proyecto se apoya en la API externa de AeroDataBox para obtener datos sobre vuelos y aeronaves. Esta API proporciona información detallada que es procesada y almacenada en la base de datos para ser utilizada por la aplicación. La elección de esta API se basó en su fiabilidad, amplitud de datos y facilidad de integración gracias a permitir llamadas con información obtenida gracias al decodificador de mensajes.

### 4.4.5. Control de Versiones

Para el control de versiones y la futura gestión colaborativa del código, se utilizó Git junto con GitHub como plataforma de alojamiento de repositorios. Git es una herramienta esencial para mantener un historial de cambios, colaborar con otros desarrolladores y gestionar el ciclo de vida del desarrollo de software.

### 4.4.6. Herramientas de Implementación y Publicación

Para hacer pública la aplicación, se utilizó ngrok, una herramienta que permite exponer localmente el servidor de desarrollo a la web mediante la creación de túneles seguros y de manera completamente gratuita. Ngrok es especialmente útil para la etapa de pruebas y demostraciones, permitiendo que el proyecto sea accesible temporalmente desde cualquier lugar.

# CAPÍTULO 5
# Desarrollo

El desarrollo de la solución propuesta ha sido un proceso iterativo que implicó diversas fases, desde la conceptualización inicial hasta la implementación final. El proceso de desarrollo se basaba en una metodología Agile centrada en la realización de pequeños entregables funcionales en vez de un gran desarrollo seguido según la planificación.

## 5.1 Fase Inicial: Conceptualización y Planificación

El proyecto comenzó con una fase de conceptualización en la que se definieron los objetivos principales y se identificaron los requisitos funcionales y no funcionales de la aplicación. Se decidió utilizar una arquitectura basada en el patrón Model-View-Controller (MVC) para separar las responsabilidades del sistema, lo que facilita el mantenimiento y la escalabilidad.

Fueron en estas primeras fases del proyecto donde hubieron más problemas debido a un conocimiento no tan extenso sobre la materia. Al principio se investigó y se planificó el uso de herramientas que proporcionaban otras webs del mismo campo como flightaware o ADSBexchange. Estas herramientas consistían en unos sistemas operativos configurados específicamente para transformar la raspberry pi en un 'feeder', es decir, una máquina hecha para detectar los mensajes emitidos por las aeronaves y usarlos para 'alimentar' a sus webs con la información obtenida con el fin de mejorar sus servicios. Sin embargo, estas soluciones no eran útiles para el proyecto puesto que, a parte de tener todo hecho, no había documentación para que el usuario que utilizase tales herramientas pudiera hacer uso de la información. Por lo tanto, después de investigar en profundidad estos sistemas operativos personalizados se descubrió que todos estaban basados en el software de decodificación 'readsb', un software de decodificación de mensajes ADS-B que sí que posee una gran documentación sobre su funcionalidad pero sobre todo sobre los mensajes decodificados.

Una vez cerrado el problema con el decodificacor solo quedaba por planificar la selección de las tecnologías a utilizar. Después de evaluar varias opciones, en un principio se optó por:

Frontend: React.js, debido a su capacidad para construir interfaces de usuario dinámicas y su popularidad en el desarrollo web moderno. Backend: Flask, un microframework en Python, elegido por su simplicidad y flexibilidad.

En este momento se realizaron bocetos sobre como debería ser la interfaz de usuario

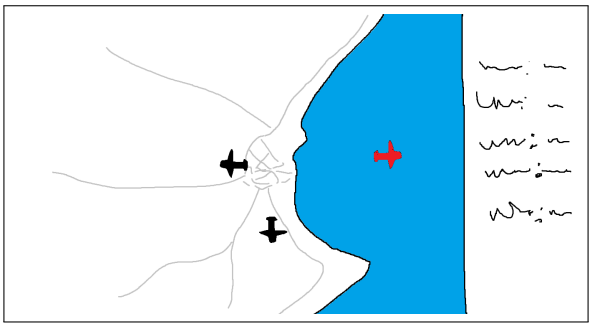


**Figura 5.1:** Boceto Inicial

Más tarde, conforme fue avanzando el proyecto se fueron sumando el resto de agentes como la base de datos o la API externa y se modificó el boceto inicial para incluir la información adicional.

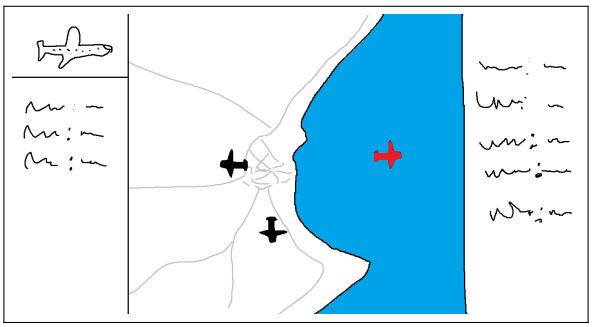


**Figura 5.2:** Boceto Intermedio

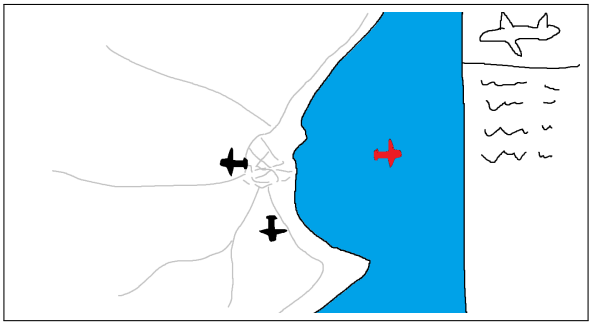


**Figura 5.3:** Boceto Final

## 5.2 Fase de Desarrollo: Implementación y Prototipado

La fase de desarrollo comenzó con la creación de un prototipo básico que integraba el frontend con el backend. Esta etapa fue fundamental para validar la arquitectura propuesta y asegurarse de que los componentes principales funcionaran correctamente.

### 5.2.1. Desarrollo del Backend

El backend fue desarrollado utilizando Flask, y comenzó con la implementación de rutas para manejar las solicitudes HTTP que se realizarían desde el frontend. Al seguir la metodología Agile comenzó con simples solicitudes, más tarde al observar que la información obtenida por los mensajes ADS-B era muy técnica y poco amigable para usuarios sin grandes conocimientos de aviación se optó por utilizar una API externa.

Con la elección de utilizar una API externa para obtener información extra sobre las aeronaves surgió otro proceso de investigación y prueba y error para escoger la API adecuada. Entre las APIs candidatas se encontraban la API ofrecida por ADSBexchange, la API de FlightAware 'AeroAPI' y 'Aviationstack'. Centrado en el objetivo de mantener el proyecto con un presupuesto mínimo se observaron los planes gratuitos de cada API y se decidió en un principio por 'Aviationstack'. Sin embargo, después de probar la API descubrí que había cometido un error, había malinterpretado la información necesaria para poder filtrar la solicitud, entre la información obtenida por los mensajes ADS-B no se encontraban los parámetros necesarios para poder utilizar la API. Sin embargo, se descubrió la API 'AeroDataBox', con la que se podía hacer solicitudes filtrando por el número ICAO codificado en hexadecimal de 24 bits, parámetro que era capaz de obtener la antena.

AeroDataBox es una API RESTful que ofrece unos planes limitados por 'puntos', el plan gratuito posee un límite de mil doscientos puntos al mes y te permite realizar cualquier consulta que ofrece la API. Sin embargo, según la complejidad de la consulta, el coste de 'puntos' varía. La consulta al endpoint de aeronaves utilizando el filtrado por número ICAO tiene un coste asociado de un punto por lo que la aplicación, de manera gratuita, podría realizar hasta mil doscientas consultas. Este endpoint es capaz de otorgar información sobre las aeronaves como el nombre de la aerolínea, el modelo de la aeronave, una imagen del avion o hasta la edad del avión.

Una vez solucionado el problema con la API surgió otra necesidad en el código. Debido a la limitada cantidad de posibles peticiones había que pensar en soluciones que permitiesen reducir el número de solicitudes. Se aplicó que las solicitudes solo tuviesen lugar en el momento de seleccionar un avión y se utilizó un sistema de cache en memoria para no repetir solicitudes. Sin embargo, la solución de la caché no fue muy convincente y se decidió cambiarla por una base de datos en SQLite puesto que ofrecía una solución permanente con mucha más capacidad que la caché, y que, llegado a un punto más futuro del proyecto se po-

dría utilizar para poder ofrecer un servicio de API capaz de ofrecer los mismos servicios que una API externa como la de 'AeroDataBox'.

Una de las dificultades encontradas fue la gestión de los datos recibidos de la API externa, que en ocasiones eran incompletos o contenían información inesperada. Para resolver este problema, se implementaron verificaciones y filtros que aseguran que solo se almacene información válida en la base de datos. Un ejemplo de código relevante fue la función add_plane_to_db, que maneja la inserción de nuevos registros, incluyendo mecanismos de control para evitar errores cuando ciertos datos están ausentes.

### 5.2.2. Desarrollo del Frontend

El frontend fue construido con React.js, y se desarrollaró un componente reutilizable para la interfaz de usuario. La comunicación con el backend se implementó utilizando fetch API para realizar solicitudes HTTP y actualizar dinámicamente la interfaz en función de las respuestas recibidas.

En el frontend, al seguir una metodología centrada en pequeños entregables, también fue creciendo desde una simple solución como era imprimir por pantalla el resultado de la solicitud al backend.

El primer paso después de comprobar la relación con el backend fue hacer uso de leaflet, una biblioteca de JavaScript que permité mostrar un mapa he interactuar con él. El siguiente paso fue hacer que los aviones obtenidos fueran marcados en el mapa y con un Popup apareciese su información. Ese Popup acabo evolucionando a un icono de un avión que rota siguiendo la dirección de la aeronave a la que hace referencia y que cuando se selecciona se abre un panel con la información obtenida en el mensaje ADS-B. Por último, junto a la información obtenida por la API externa se hizo una selección de la información más importante para no abrumar al usuario.

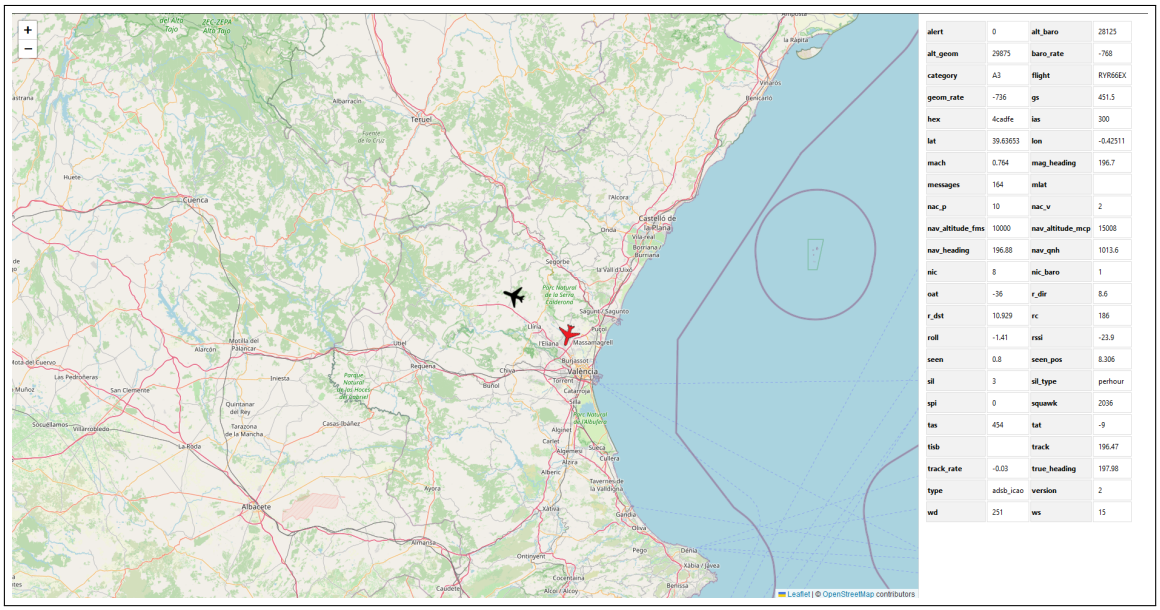



**Figura 5.4:** Solución Inicial

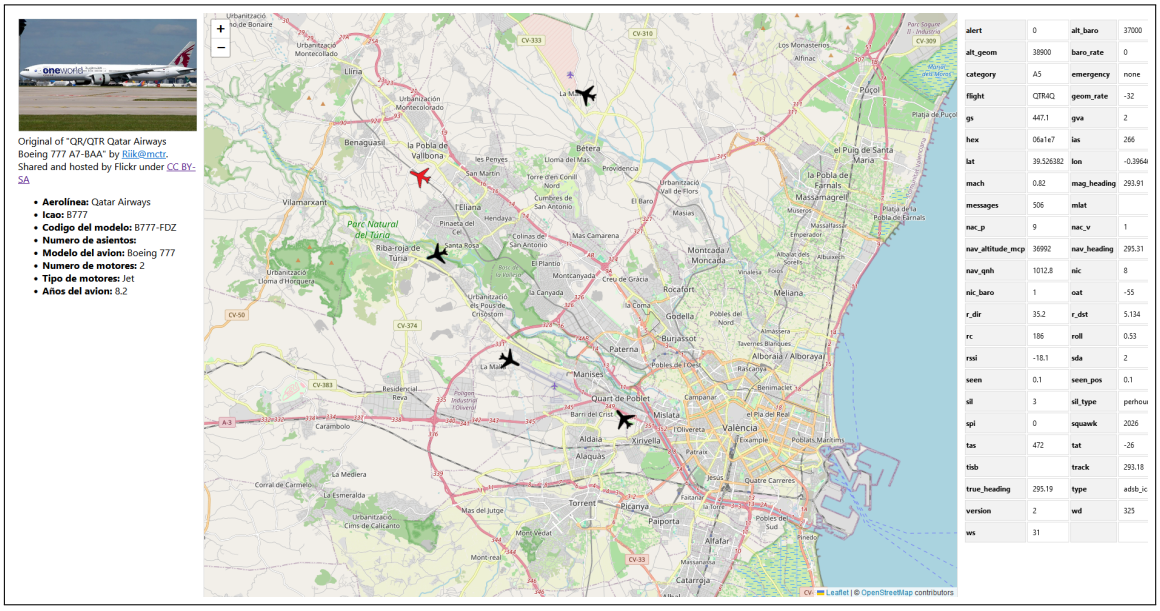



**Figura 5.5:** Solución Intermedia

Además, gracias a las pruebas de usuario que se realizaron, se observó que la información no era legible desde un dispositivo móvil como un teléfono por lo que se cambio la forma de mostrar esa información basándose en el ancho de la pantalla desde la que se accede a la web.

Un desafío particular fue la representación gráfica de los datos obtenidos de la API externa, especialmente para mostrar la imagen de la aeronave y respetar las licencias asociadas a cada recurso. Para esto, se diseñó una lógica que verifica la licencia antes de mostrar las imágenes, asegurando que se cumplan las restricciones de uso.

## 5.3 Fase de Integración y Pruebas

Una vez que los componentes individuales estuvieron implementados, se procedió a la integración de los mismos. Esta fase incluyó la conexión del frontend con el backend y la base de datos, así como la integración de la API externa.

Durante la integración, se realizaron varias pruebas para asegurar la robustez del sistema. Se identificaron y solucionaron problemas de rendimiento, especialmente en relación con la cantidad de datos recibidos de la API externa y la capacidad del sistema para manejarlos eficientemente. Esto llevó a la implementación de un sistema de almacenamiento en caché utilizando una base de datos SQLite, lo que redujo la cantidad de llamadas a la API externa y mejoró la velocidad de respuesta.

## 5.4 Problemas y Dificultades

Durante el desarrollo del proyecto, se enfrentaron varias dificultades técnicas y decisiones críticas:

- Manejo de Datos Incompletos: La API externa a menudo devolvía datos incompletos, lo que generó la necesidad de implementar verificaciones adicionales y lógica de manejo de errores en el backend.

- Optimización de la Interfaz de Usuario: Inicialmente, la interfaz presentaba problemas de rendimiento debido a la carga de datos en tiempo real. Se realizaron optimizaciones en React.js para mejorar la experiencia del usuario. Además, se mejoró la interfaz para permitir un uso más amable desde dispositivos menos anchos que una pantalla de ordenador.

- Licencias de Imágenes: Respetar las licencias de las imágenes obtenidas de la API externa fue un reto. Se implementó una lógica específica para asegurarse de que solo se mostraran imágenes cuya licencia permitiera su uso en la aplicación.

## 5.5 Solución Final

El resultado final es una aplicación web robusta que permite a los usuarios acceder a información en tiempo real sobre aeronaves, mostrando datos detallados y respetando las licencias de las imágenes asociadas. La solución final no solo cumple con los requisitos establecidos al inicio del proyecto, sino que también ofrece una base sólida para futuras expansiones o mejoras.

Este proceso de desarrollo ha permitido abordar y resolver una variedad de desafíos técnicos, lo que ha resultado en un producto funcional y eficiente que refleja las decisiones cuidadosas y el trabajo iterativo realizado durante el proyecto.

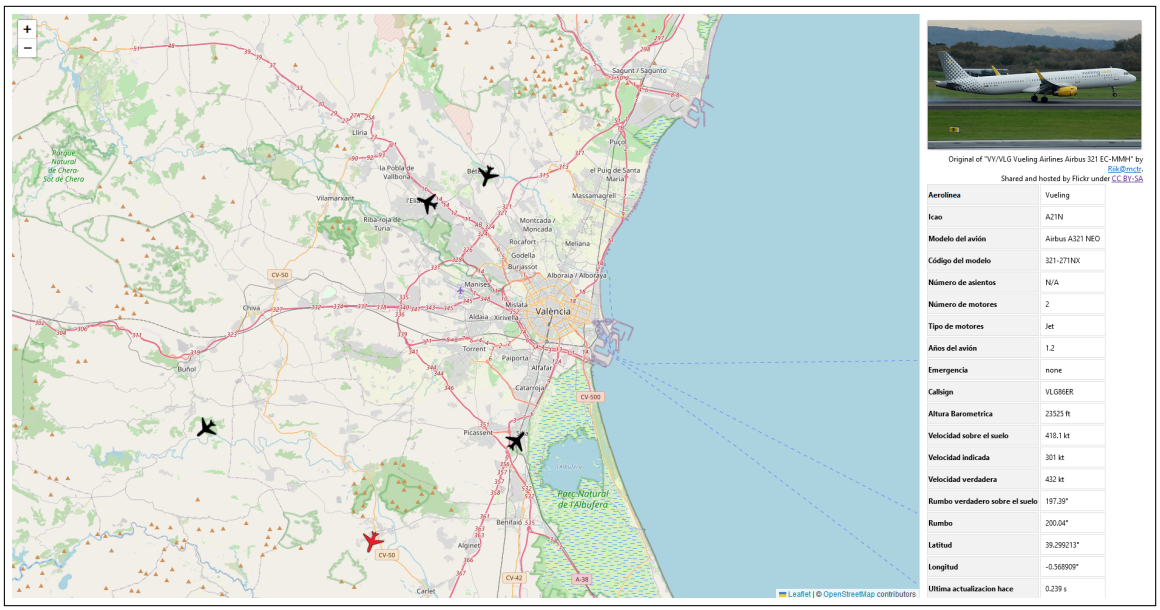
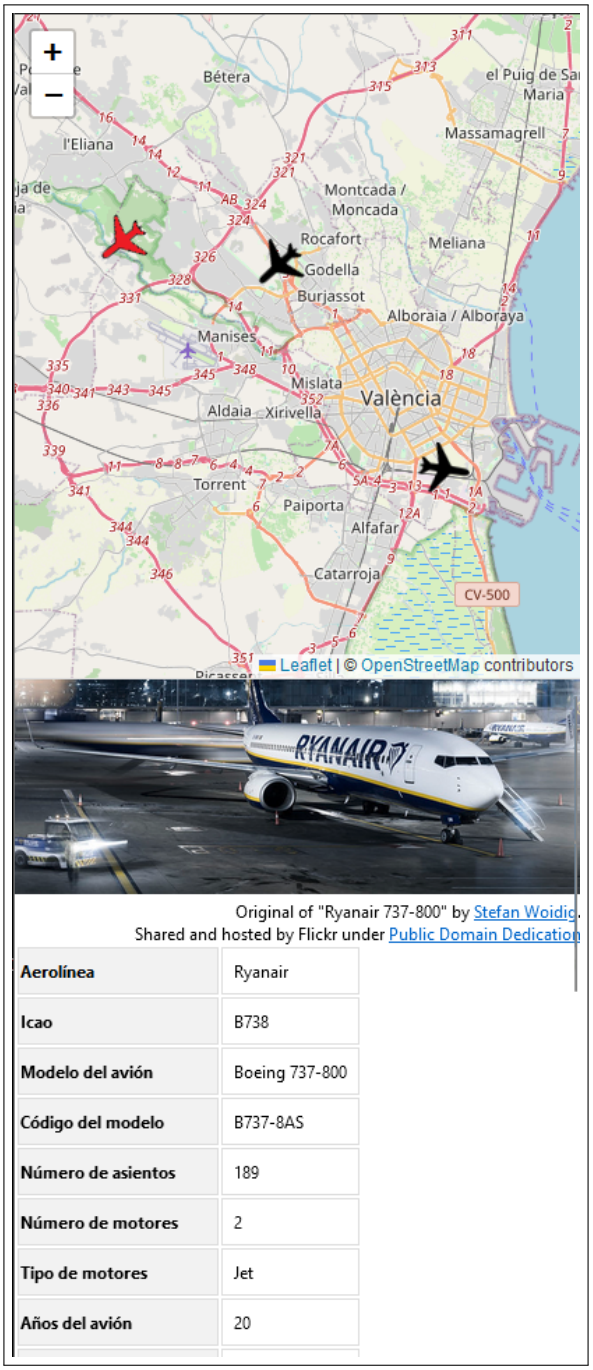



**Figura 5.6:** Solución Final




| | |
|---|---|
| **Aerolínea** | Ryanair |
| **Icao** | B738 |
| **Modelo del avión** | Boeing 737-800 |
| **Código del modelo** | B737-8AS |
| **Número de asientos** | 189 |
| **Número de motores** | 2 |
| **Tipo de motores** | Jet |
| **Años del avión** | 20 |

**Figura 5.7:** Solución Final en dispositivo móvil

## 5.6 Gestión de versiones

La gestión de versiones es una práctica esencial en el desarrollo de software que permite controlar y documentar los cambios realizados en el código fuente a lo largo del tiempo. En este proyecto, que se desarrolló de manera individual, pero con la posibilidad de expandirse como proyecto de código abierto, se ha implementado utilizando Git y GitHub.

### 5.6.1. Uso de Git y GitHub

#### Git

Control Local de Versiones: Git es una herramienta de control de versiones distribuido que permite gestionar los cambios en el código fuente de manera local. Durante el desarrollo del proyecto, se ha utilizado Git para registrar todos los cambios realizados en el código. Cada cambio se documenta mediante commits, que incluyen un mensaje descriptivo que explica el propósito del cambio.

Commits: Los commits se han realizado con frecuencia para asegurar que cada etapa del desarrollo esté bien documentada y sea reversible en caso de problemas. Cada commit incluye un mensaje claro que describe el propósito del cambio.

#### GitHub

Repositorio Remoto: GitHub se ha utilizado como plataforma de alojamiento para el repositorio del proyecto. Ofrece una vista centralizada del historial de versiones y facilita la colaboración, aunque en este caso fue un desarrollo individual.

Pull Requests: Aunque el proyecto fue desarrollado de manera individual, GitHub permite la creación de pull requests para revisar cambios antes de fusionarlos con la rama principal. Esta funcionalidad es útil para proyectos de código abierto y para garantizar la calidad del código antes de integrarlo en la versión estable.

Issues: GitHub Issues se puede utilizar en un futuro para registrar y seguir el progreso de errores y tareas pendientes. Permite mantener un registro de los problemas encontrados y las mejoras deseadas.

### 5.6.2. Consideraciones para el Futuro

#### Código Abierto y Colaboración

Expansión del Proyecto: Dado que el proyecto está diseñado para ser de código abierto, la gestión de versiones es crucial para coordinar la colaboración con otros desarrolladores. En el futuro, se podrían incorporar colaboradores externos que contribuyan al desarrollo del proyecto. Dando lugar al uso de ramas (branches) para organizar el desarrollo de nuevas características y correcciones de errores. La rama principal main o master contiene la versión estable del código, mientras que las ramas secundarias, como feature/new-feature o bugfix/fix-bug, se utilizan para el desarrollo y pruebas de nuevas funcionalidades o correcciones.

Contribuciones: Los colaboradores pueden hacer fork del repositorio y enviar pull requests para proponer cambios. La gestión de estas contribuciones se realizará mediante revisiones de código y la integración de mejoras en el proyecto principal.

### Versionado del Software

Etiquetas (Tags): Para indicar versiones estables del software, se utilizarán etiquetas en Git. Esto facilita la referencia a versiones específicas del proyecto y es útil para la distribución de lanzamientos.

Historial de Versiones: El historial de versiones documentará los cambios significativos en cada versión del proyecto, incluyendo nuevas características, mejoras y correcciones de errores.

### Documentación y Comunicación

Historial de Cambios: Se mantendrá un registro detallado de los cambios en el archivo CHANGELOG.md, que documentará todas las versiones del software y los cambios incluidos en cada una.

Documentación de Código: La documentación del código y los comentarios serán fundamentales para facilitar la comprensión del proyecto a nuevos colaboradores y para mantener la calidad del software.

# CAPÍTULO 6
# Implantación

La implantación del proyecto marca la transición desde el desarrollo y las pruebas hacia el entorno de producción, donde la solución se pone en funcionamiento y se pone a disposición de los usuarios finales. En este proyecto, la implantación se ha realizado utilizando ngrok, una herramienta que permite exponer aplicaciones locales a la web a través de direcciones temporales pero que ofrece un dominio persistente gratuito.

Gracias al uso de herramientas gratuitas y fáciles de usar cualquier persona es capaz de reproducir los mismos pasos para hacer funcionar el proyecto desde sus ordenadores, con el único requisito de obtener un decodificador y una antena.

A continuación, se describe el proceso de implantación, la configuración de ngrok, y los pasos realizados para llevar el proyecto a producción.

## 6.1 Exposición de la Aplicación con ngrok

Para hacer accesible la aplicación web desarrollada desde cualquier parte del mundo, se ha utilizado ngrok, una solución de túnel que expone servicios locales a través de un dominio público. La URL utilizada por ngrok gracias al dominio gratuito, https://liberal-cockatoo-moderately.ngrok-free.app/, proporciona un punto de acceso a la aplicación en un entorno de pruebas, permitiendo que la solución sea accesible fuera del entorno local.

La implantación del proyecto mediante ngrok ha permitido exponer la aplicación web desarrollada a un entorno global, facilitando la prueba y la demostración de la solución. Aunque ngrok es una herramienta útil para la exposición temporal de aplicaciones, para un despliegue a largo plazo es recomendable considerar opciones más robustas de alojamiento y dominio para un futuro. La implementación exitosa de la solución y la validación exhaustiva garantizan que la aplicación esté lista para su uso y accesible a los usuarios finales, todo ello manteniendo los costes al mínimo.

### 6.1.1. Instalación y Configuración de ngrok

Descarga e Instalación: Ngrok se descargó e instaló en el sistema donde se ejecuta el proyecto. La instalación puede realizarse siguiendo las instrucciones

disponibles en la documentación oficial de ngrok. Autenticación: Se realizó la autenticación con la cuenta de ngrok para generar un token de autenticación, lo cual permite la creación de túneles personalizados y el acceso a funciones adicionales. Creación del Túnel: Se ejecutó el comando de ngrok para crear un túnel que expone el puerto en el que se ejecuta la aplicación web. En este caso, gracias a Flask y a React, el backend es capaz de servir el frontend por lo que solo hace falta exponer el puerto del backend, el 5000. El comando utilizado fue:

```
ngrok http --domain=liberal-cockatoo-moderately.ngrok-free.app 5000
```

Obtención de la URL: Ngrok proporcionó una URL pública, https://liberal-cockatoo-moderately.ngrok-free.app/, que permite acceder a la aplicación desde cualquier navegador.

## 6.2 Proceso de Despliegue y Configuración

Durante el proceso de despliegue, se llevaron a cabo varias actividades clave para garantizar que la aplicación esté correctamente configurada y lista para su uso:

### 6.2.1. Configuración del Entorno

Se instalaron todas las dependencias y paquetes necesarios utilizando pip para el backend y npm para el frontend.

Las dependencias del backend se encuentran dentro de un archivo 'requirements.txt', mientras que las del frontend se encuentran en un archivo 'package.json', de esta manera cualquier persona que quiera replicar la aplicación será capaz de conocer todas las dependencias necesarias.

### 6.2.2. Ejecutar el Backend y Frontend

Backend: Se inició el servidor del backend utilizando el siguiente comando dentro de la carpeta 'backend':

```
python app.py
```

Frontend: Si se utiliza un frontend independiente en el puerto 3000, se inició con el comando correspondiente, por ejemplo:

```
npm start
```

Esta parte no es necesaria porque se configuró para que el backend fuera capaz de servir al frontend gracias a construirlo utilizando el siguiente comando:

```
npm run build
```

Monitoreo: Se realizó un monitoreo continuo del rendimiento y la estabilidad de la aplicación a través de ngrok para identificar y solucionar problemas potenciales en tiempo real.

Mantenimiento: Se implementaron procedimientos para la actualización y mantenimiento de la aplicación, incluyendo la gestión de actualizaciones y corrección de errores.

## 6.3 Limitaciones y Consideraciones

Seguridad: La exposición de la aplicación a través de ngrok puede presentar riesgos de seguridad. Se recomienda implementar medidas adicionales de seguridad, como autenticación y cifrado, para proteger los datos y las funcionalidades de la aplicación.

Escalabilidad: Ngrok es adecuado para pruebas y desarrollo, pero para aplicaciones en producción se debe considerar el uso de servicios de alojamiento en la nube o servidores dedicados que puedan manejar un mayor volumen de tráfico y ofrecer mayor fiabilidad.

## 6.4 Pruebas

En esta sección se detalla el proceso de verificación de la solución desarrollada, incluyendo pruebas funcionales, pruebas de validación, y pruebas de carga. El objetivo de estas pruebas es asegurar que la solución cumple con los requisitos especificados y funciona de manera eficiente en condiciones de uso real.

### 6.4.1. Pruebas Funcionales

Las pruebas funcionales se realizaron para asegurar que la aplicación cumple con las funcionalidades esperadas y que cada componente opera correctamente. Las pruebas se realizaron en diferentes fases del desarrollo, tanto en el backend como en el frontend. A continuación se detallan las principales pruebas funcionales:

#### Pruebas de recepción

Para garantizar la eficacia del sistema de recepción de señales ADS-B, se realizaron pruebas específicas para verificar la capacidad de la antena de captar información emitida por los aviones. Estas pruebas se llevaron a cabo en mi residencia, situada en un séptimo piso en Alaquàs, uno de los edificios más altos de las cercanías, lo que proporcionó una ventaja en términos de altura para la recepción de señales.

La antena fue instalada en un estante de mi habitación, estratégicamente ubicada al lado de una ventana que da a un patio interior. Aunque la posición de la antena no es la ideal en términos de exposición y recepción—ya que no está

orientada directamente hacia el exterior en un punto elevado y despejado—se optó por esta ubicación debido a la comodidad y la facilidad de acceso en el entorno doméstico.

A pesar de la limitación en la ubicación de la antena, los resultados de las pruebas fueron satisfactorios. La antena demostró ser capaz de recibir señales de aviones en un radio promedio de aproximadamente 20 kilómetros llegando a recibir mensajes de aeronaves sobrevolando Castelló de la Plana o Ayora. Este alcance es notable, considerando la ubicación interna de la antena, lo que sugiere que la instalación actual es adecuada para las necesidades del proyecto, aunque se podría mejorar la recepción con una mejor orientación o ubicación de la antena.

**Figura 6.1:** Antena


**Figura 6.2:** Raspberry Pi con el dongle conectado

### Pruebas de API

Se verificó que las endpoints del backend respondan correctamente a las solicitudes utilizando la herramienta Postman. Esto incluyó pruebas para endpoints que manejan la obtención de información de aviones, almacenamiento en base de datos, y recuperación de datos.

### Pruebas de Interfaz de Usuario (UI)

Se probó la interfaz de usuario para verificar que los elementos visuales se cargan correctamente y que las interacciones del usuario, como la selección de aviones en el mapa y la visualización de detalles, funcionan sin errores.

### Pruebas de Integración

Se realizó una prueba de integración para garantizar que el frontend y el backend funcionen correctamente juntos. Esto incluyó verificar que los datos del backend se muestren adecuadamente en el frontend y que la comunicación entre ambos sistemas sea fluida.

### Pruebas de Manejo de Errores

Se probó cómo la aplicación maneja errores, como solicitudes inválidas o respuestas de la API externas con datos faltantes o incorrectos. Se verificó que la

aplicación maneje estos errores de manera adecuada, mostrando mensajes informativos al usuario y no bloqueando el funcionamiento de la aplicación.

### 6.4.2. Pruebas de Validación

Las pruebas de validación se llevaron a cabo para asegurarse de que la solución cumple con las expectativas del usuario final y con los requisitos del proyecto. Estas pruebas incluyeron:

#### Pruebas con el Usuario Final

Se realizó una sesión de pruebas con usuarios representativos para validar que la aplicación cumpla con sus necesidades y expectativas. Los usuarios probaron diferentes funcionalidades, como la visualización de aviones, la visualización de detalles y la navegación por la interfaz. Se recopilaron comentarios para ajustar y mejorar la experiencia de usuario.

#### Validación de Requisitos

Se revisaron los requisitos especificados para verificar que todas las funcionalidades requeridas se hayan implementado correctamente. Se comprobó que la aplicación permita gestionar información de aviones, mostrar imágenes y manejar datos de manera eficiente.

### 6.4.3. Pruebas de Carga

Se evaluó la capacidad de respuesta tanto del backend como del frontend con múltiples usuarios simultáneos. Se comprobó que la interfaz sigue siendo funcional y accesible bajo condiciones de alta carga, asegurando una experiencia de usuario fluida.

### 6.4.4. Resultados de las Pruebas

Las pruebas realizadas confirmaron que la solución desarrollada cumple con los requisitos funcionales y de rendimiento establecidos. Se identificaron y corrigieron algunos errores menores durante el proceso de pruebas, lo que resultó en una aplicación robusta y confiable. Las pruebas de validación con usuarios finales mostraron una alta satisfacción con la interfaz y las funcionalidades de la aplicación.

Las pruebas de carga indicaron que la aplicación puede manejar el tráfico y el volumen de datos esperado sin problemas significativos de rendimiento. Se realizaron ajustes en la configuración del servidor y en el código para optimizar el rendimiento y garantizar que la aplicación sea escalable y eficiente.

La fase de pruebas aseguró que la solución cumple con los estándares de calidad y funcionamiento requeridos, ofreciendo una experiencia de usuario satisfactoria y un rendimiento adecuado.

# CAPÍTULO 7
# Conclusiones

Este proyecto ha sido una experiencia de aprendizaje significativa que me ha permitido integrar conocimientos adquiridos durante mi formación académica con nuevas tecnologías y herramientas que no había explorado en profundidad previamente. Desde el inicio, me propuse desarrollar una aplicación web capaz de gestionar y mostrar información detallada sobre aeronaves, utilizando para ello tecnologías modernas y accesibles. A lo largo del proceso, he alcanzado satisfactoriamente los objetivos planteados, aunque no sin enfrentar desafíos que me han permitido crecer tanto profesional como personalmente.

Uno de los objetivos principales era la creación de un sistema eficiente y funcional que pudiera manejar y almacenar datos sobre aeronaves de manera robusta. Para ello, decidí emplear una arquitectura basada en Flask para el backend, con una base de datos SQLite, y React para el frontend. Estos componentes, aunque familiares en un sentido básico, requerían un aprendizaje adicional para poder ser utilizados de manera óptima en un entorno real. La integración de estas tecnologías ha demostrado ser efectiva, logrando un sistema que cumple con los requisitos funcionales y de rendimiento.

En términos de dificultades, uno de los problemas más notables fue la gestión de datos faltantes o inconsistentes provenientes de la API externa. Este reto me obligó a implementar soluciones creativas para asegurar que la aplicación se mantuviera estable y funcional incluso cuando los datos no eran ideales. Además, la implementación de un sistema de pruebas para validar la funcionalidad y la eficiencia de la aplicación resultó ser un proceso complejo, pero esencial para asegurar la calidad del producto final.

En cuanto a errores, la falta de experiencia previa con algunas de las tecnologías utilizadas llevó a ciertos tropiezos iniciales, como elegir una API que filtraba por parámetros que no tenía o como los muchos errores que han ido surgiendo durante el desarrollo del frontend. Sin embargo, estos errores me proporcionaron valiosas lecciones sobre la importancia de una investigación cuidadosa y la necesidad de familiarizarse en profundidad con las herramientas antes de su implementación.

El desarrollo de este proyecto también ha sido una oportunidad para aprender nuevas tecnologías y mejorar mi dominio sobre ellas. A pesar de mi experiencia en el área de inteligencia artificial, este proyecto se enfocó en el desarrollo web, lo cual fue una desviación significativa de mi trayectoria académica. El aprendi-

zaje de Flask, React, herramientas de despliegue como ngrok, y la integración de frontend y backend en un entorno real de producción, son habilidades que he adquirido y que ahora domino a un nivel que me permitirá hacer uso de estos conocimientos para proyectos personales o profesionales futuros.

Este proyecto ha cumplido con los objetivos propuestos, ha facilitado el aprendizaje de nuevas tecnologías y ha sido una experiencia enriquecedora tanto a nivel personal como profesional. Los resultados obtenidos reflejan no solo la capacidad de resolver los problemas planteados, sino también la habilidad para adaptarse y aprender en un entorno técnico en constante evolución. Como conclusión, se abre la puerta a futuras mejoras y expansiones del sistema, particularmente en aspectos relacionados con la escalabilidad y la seguridad, áreas en las que este proyecto podría seguir evolucionando.

## 7.1 Relación del trabajo desarrollado con los estudios cursados

El desarrollo de este proyecto ha permitido poner en práctica y coordinar los conocimientos adquiridos a lo largo de mis estudios en el grado, especialmente en áreas como la programación, desarrollo de software, y gestión de bases de datos. A través de la implementación de un sistema web que integra un frontend en React.js y un backend en Flask, he podido aplicar conceptos fundamentales de ingeniería de software, demostrando la capacidad de resolver problemas reales utilizando tecnologías actuales.

Durante el grado, el enfoque en la programación ha sido clave para poder diseñar y desarrollar tanto el frontend como el backend de esta aplicación. La comprensión de la lógica de programación y las estructuras de datos ha facilitado la creación de un backend robusto, capaz de gestionar y procesar datos provenientes de una API externa, así como de interactuar con una base de datos SQLite. Asimismo, los conocimientos en diseño de interfaces de usuario han sido esenciales para crear un frontend eficiente que garantiza una buena experiencia de usuario.

Aunque mi especialización en Computación se centró en la inteligencia artificial, la cual no está directamente relacionada con el proyecto desarrollado, los fundamentos técnicos adquiridos durante el grado han sido invaluables. Estos conocimientos me han proporcionado las herramientas necesarias para comprender y aplicar nuevas tecnologías, como React y Flask, que no fueron cubiertas en profundidad durante mis estudios. Esto pone de manifiesto mi capacidad para aprender de forma autónoma y adaptarme a nuevas tecnologías, una competencia crucial en el entorno laboral actual.

Además, el proyecto ha requerido la utilización de competencias transversales como la gestión del tiempo, la organización, y la capacidad de resolver problemas de manera efectiva. Estas habilidades han sido esenciales para poder abordar un proyecto de esta magnitud de manera individual, asegurando tanto su funcionalidad como su eficiencia. La combinación de estas competencias y conocimientos ha demostrado que, a pesar de haber tomado un camino diferente al de mi espe-

cialización, he podido desarrollar una solución tecnológica completa que aborda un problema real, mostrando un dominio amplio y versátil de las tecnologías aplicadas en el proyecto.

# CAPÍTULO 8
# Trabajos futuros

En el desarrollo de este proyecto, han surgido diversas ideas y oportunidades para futuras mejoras y expansiones que no pudieron ser implementadas en el tiempo disponible para la realización del TFG. A continuación, se presentan algunas de las principales líneas de trabajo futuro que podrían enriquecer la aplicación y aumentar su valor tanto para los usuarios como para la comunidad de desarrollo.

## 8.1 Ampliación de la Información Obtenida a Través de APIs

Una de las mejoras más significativas que se podrían implementar en el futuro es la ampliación de la cantidad y variedad de datos obtenidos a través de APIs externas. Esto permitiría ofrecer a los usuarios un mayor conocimiento sobre las aeronaves, como detalles más específicos de los vuelos, historial de mantenimiento, estadísticas de rendimiento, y otros datos relevantes. Este aumento en la información requeriría, a su vez, una expansión de la base de datos para incorporar nuevas entidades de datos, lo cual permitiría organizar y gestionar de manera eficiente esta nueva información. Con la correcta planificación, la integración de estos nuevos datos no solo aumentaría la funcionalidad de la aplicación, sino también su atractivo para un público más amplio de entusiastas y profesionales de la aviación.

## 8.2 Adquisición de un Dominio Propio

Actualmente, la aplicación se encuentra publicada utilizando la herramienta Ngrok con un dominio gratuito. Una posible mejora para el futuro sería la adquisición de un dominio propio, lo cual proporcionaría una imagen más profesional y mayor estabilidad en el acceso a la aplicación. Sin embargo, esta opción conlleva un costo económico, lo que contrasta con el enfoque actual de mantener el proyecto sin gastos. Es importante evaluar si los beneficios de adquirir un dominio propio justifican el gasto, especialmente si el proyecto continúa siendo una plataforma sin fines de lucro.

## 8.3 Crecimiento de la Comunidad de Desarrolladores

El mejor plan de futuro para esta aplicación es fomentar la participación de nuevos desarrolladores que compartan un interés por la aviación y deseen contribuir al proyecto. La idea es que esta aplicación pueda servir como una plataforma colaborativa donde desarrolladores de diferentes partes del mundo puedan mejorarla o incluso usarla como base para nuevos desarrollos. Al ser un proyecto de código abierto, la participación de la comunidad es crucial para su crecimiento y evolución. Esta colaboración podría llevar a la implementación de nuevas funcionalidades, mejoras de rendimiento, y la incorporación de tecnologías emergentes que mantengan la aplicación a la vanguardia.

## 8.4 Áreas y Caminos a Evitar

Es importante también reconocer algunas direcciones que, aunque inicialmente puedan parecer atractivas, pueden no ser viables o recomendables para el desarrollo futuro del proyecto. Por ejemplo, la integración de sistemas de pago o funcionalidades premium podría desviar el enfoque del proyecto, que se basa en ofrecer una herramienta gratuita y accesible. Además, es crucial evitar una sobrecarga de datos o funcionalidades que puedan complicar la usabilidad de la aplicación o aumentar excesivamente los costos de mantenimiento. La clave es mantener un balance entre expansión y simplicidad, priorizando siempre la experiencia del usuario y la sostenibilidad del proyecto.

# Bibliografía


[1] Junzi Sun. *The 1090 Megahertz Riddle: A Guide to Decoding Mode S and ADS-B Signals*. TU Delft OPEN Publishing, segunda edición, 2021.

[2] Flightradar24. Aplicación web de seguimiento de aviones. Obtenido en https://www.flightradar24.com/.

[3] FlightAware. Aplicación web de seguimiento de aviones. Obtenido en https://flightaware.com/.

[4] OpenSky Network. Monitoreo de trafico aereo de codigo abierto. Obtenido en https://opensky-network.org/.

[5] SQLite. Base de datos ligera y auto-contenida. Obtenido en https://www.sqlite.org/.

[6] Ngrok. Herramienta para exponer servidores locales a internet. Obtenido en https://ngrok.com/.

[7] Flask. Microframework web para Python. Obtenido en https://flask.palletsprojects.com/.

[8] Django. Framework web de alto nivel en Python. Obtenido en https://www.djangoproject.com/.

[9] PostgreSQL. Sistema de gestión de bases de datos relacional. Obtenido en https://www.postgresql.org/.

[10] Node.js. Entorno de ejecución para JavaScript en el servidor. Obtenido en https://nodejs.org/.

[11] MongoDB. Base de datos NoSQL orientada a documentos. Obtenido en https://www.mongodb.com/.

[12] React. Biblioteca de JavaScript para construir interfaces de usuario. Obtenido en https://reactjs.org/.

[13] Vue.js. Framework progresivo de JavaScript para interfaces de usuario. Obtenido en https://vuejs.org/.

[14] Angular. Plataforma de desarrollo para aplicaciones web. Obtenido en https://angular.io/.

[15] Amazon EC2. Servicio de computación en la nube. Obtenido en https://aws.amazon.com/ec2/.

[16] Nginx. Servidor web y proxy inverso. Obtenido en https://nginx.org/.

[17] GitHub. Plataforma para control de versiones y colaboración. Obtenido en https://github.com/.

[18] Netlify. Plataforma de hosting para aplicaciones web modernas. Obtenido en https://www.netlify.com/.

[19] Raspberry Pi. Computadora de bajo costo. Obtenido en https://www.raspberrypi.org/.

[20] Visual Studio Code. Editor de código fuente. Obtenido en https://code.visualstudio.com/.

[21] Readsb. Decodificador de señales ADS-B. Obtenido en https://github.com/wiedehopf/readsb.

[22] Leaflet. Biblioteca de JavaScript para mapas interactivos. Obtenido en https://leafletjs.com/.

[23] AeroDataBox. API para obtener datos de aviación. Obtenido en https://rapidapi.com/aedbx-aedbx/api/aerodatabox/.

[24] ADSBexchange. Red de seguimiento de vuelos en tiempo real. Obtenido en https://www.adsbexchange.com/.

[25] FlightAware. Aplicación de seguimiento de vuelos. Obtenido en https://www.flightaware.com/adsb/flightfeeder/.

[26] AeroAPI. API para obtener datos de aviación. Obtenido en https://flightaware.com/commercial/aeroapi/.

[27] Aviationstack. API para datos de vuelos en tiempo real. Obtenido en https://aviationstack.com/.

# ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

| **Objetivos de Desarrollo Sostenible** | **Alto** | **Medio** | **Bajo** | **No procede** |
|---|---|---|---|---|
| ODS 1. **Fin de la pobreza.** | | | | x |
| ODS 2. **Hambre cero.** | | | | x |
| ODS 3. **Salud y bienestar.** | | | | x |
| ODS 4. **Educación de calidad.** | | | x | |
| ODS 5. **Igualdad de género.** | | | | x |
| ODS 6. **Agua limpia y saneamiento.** | | | | x |
| ODS 7. **Energía asequible y no contaminante.** | | | | x |
| ODS 8. **Trabajo decente y crecimiento económico.** | | x | | |
| ODS 9. **Industria, innovación e infraestructuras.** | x | | | |
| ODS 10. **Reducción de las desigualdades.** | | | | x |
| ODS 11. **Ciudades y comunidades sostenibles.** | x | | | |
| ODS 12. **Producción y consumo responsables.** | | | x | |
| ODS 13. **Acción por el clima.** | | x | | |
| ODS 14. **Vida submarina.** | | | | x |
| ODS 15. **Vida de ecosistemas terrestres.** | | | | x |
| ODS 16. **Paz, justicia e instituciones sólidas.** | | | | x |
| ODS 17. **Alianzas para lograr objetivos.** | | x | | |

# 1 ODS 9: Industria, Innovación e Infraestructura

Este proyecto contribuye a la modernización y eficiencia del transporte aéreo, una parte clave de la infraestructura global. Al proporcionar herramientas para el seguimiento de vuelos y la gestión de información de aviones, el proyecto fomenta la innovación en este sector, lo que podría mejorar la seguridad, la eficiencia y la sostenibilidad de la aviación.

# 2 ODS 11: Ciudades y Comunidades Sostenibles

El transporte aéreo es un componente clave en la movilidad global y el desarrollo urbano. Este proyecto podría contribuir a la planificación urbana sostenible y a la gestión de tráfico aéreo, asegurando que las ciudades y los aeropuertos operen de manera más eficiente y con menor impacto ambiental como la contaminación acústica.

# 3 ODS 13: Acción por el Clima

La aviación es una de las industrias más desafiantes en términos de reducción de emisiones de gases de efecto invernadero. Este proyecto puede ayudar a identificar oportunidades para reducir la huella de carbono del transporte aéreo, ya sea optimizando rutas de vuelo o promoviendo prácticas más sostenibles en la industria.

# 4 ODS 8: Trabajo Decente y Crecimiento Económico

Al mejorar la eficiencia del sector aéreo, este proyecto también puede tener un impacto positivo en el crecimiento económico y la creación de empleo en la industria de la aviación y sectores relacionados, como el turismo y la logística comercial.

# 5 ODS 17: Alianzas para Lograr los Objetivos

La colaboración y el intercambio de datos entre diferentes actores del sector aéreo, como aerolíneas, aeropuertos y autoridades de aviación, son cruciales para el éxito del proyecto. Este proyecto puede llegar a fomentar la colaboración entre entidades públicas y privadas para mejorar la seguridad y eficiencia en la aviación.