UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

# Desarrollo de una web para la evaluación del nivel de ruido generado por los aviones en aeropuertos.

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Dongo Flores, Francisco

Tutor/a: Hernández Orallo, Enrique

CURSO ACADÉMICO: 2022/2023

# Resumen

El objetivo de este proyecto es desarrollar una aplicación web que permita calcular en tiempo real el ruido generado por los aviones durante su vuelo a partir de datos tales como su posición, velocidad o modelo.

Para este estudio se analizarán los niveles de ruido generados por aviones que partan o se dirijan hacia el aeropuerto de Manises (Valencia). Los datos se obtendrán a partir de una API que proporcione información en tiempo real y se procesarán para después ser utilizados en una librería que calcule los perfiles de ruido. Los resultados obtenidos dependerán de la ubicación del usuario y serán proporcionados a modo de listado en la página web que este usuario consultará. La herramienta se diseñará para que cualquier persona pueda hacer uso de ella de forma sencilla, sin que sea necesario tener conocimientos especializados.

Se pretende construir una herramienta útil que amplíe las posibilidades de recogida de información relacionada con la contaminación acústica generada por aeronaves. Esto abrirá la posibilidad a la realización de estudios que relacionen el ruido generado con el tipo de modelo o motor de las aeronaves, lo que permitirá una selección o desarrollo de aviones menos ruidosos en el futuro, algo que contribuirá a la mejora de la calidad de vida de las personas que residan cerca de los aeropuertos.

**Palabras clave:** Aplicaciones Web, Servicios Web, Trayectoria de aeronaves, Ruido generado por aeronaves

# Abstract

The objective of this project is to develop a web application that allows the calculation in real time of noise generated by airplanes during flight based on data such as their position, speed, or model.

For this study, noise levels generated by airplanes departing or arriving at Manises Airport (Valencia) will be analyzed. Data will be obtained from an API that provides real-time information and will be processed to be used in a library that calculates noise profiles. The results obtained will depend on the user's location and will be provided as a list on the website that the user will consult. The tool will be designed so that anyone can access it easily, even without specialized knowledge.

The aim is to build a useful tool that expands the possibilities of collecting information related to noise pollution generated by airplanes. This will open up the possibility of conducting studies that relate noise generated with the type of airplane model or engine, which opens the door to selecting or developing less noisy airplanes in the future, something that will contribute to improving the quality of life of people living near airports.

**Key words:** Web applications, Web services, Aircraft trajectory, Aircraft noise

# Índice general

# Índice de figuras



# Índice de tablas

# CAPÍTULO 1
# Introducción

En este capítulo se expone la motivación que ha llevado a realizar este proyecto, se plantean los objetivos que se pretenden cumplir y finalmente se proporciona una enumeración de las distintas secciones que componen este documento.

## 1.1 Motivación

La contaminación acústica es una de las plagas modernas que afecta a nuestra salud sin que seamos plenamente conscientes de ello. En el caso concreto del ruido generado por los aviones, existen estudios que relacionan una exposición frecuente a este tipo de ruidos con problemas cardiovasculares o de insomnio, entre otros [1]. Con este proyecto se pretende proporcionar una herramienta que permita realizar estimaciones de ruido en uno de los puntos más conflictivos de su trayectoria, el aeropuerto y sus cercanías.

## 1.2 Objetivos

El objetivo principal de este trabajo es desarrollar de forma exitosa una aplicación web que permita estimar el ruido generado por los aviones cercanos a la posición del usuario, tomando como referencia el aeropuerto de Manises. Este objetivo se desglosará en otros objetivos secundarios:

- Obtener la información necesaria para realizar los cálculos a partir de diversas APIs y bases de datos.
- Desarrollar un programa que permita procesar esa información para calcular el perfil de ruido.
- Diseñar una página web que integre el programa mencionado en el anterior punto y muestre al usuario los resultados.

Es decir, el resultado de cumplir con estos objetivos debe ser una página web que a partir de una ubicación predeterminada o calculando automáticamente la del usuario informe sobre el ruido generado por los aviones cercanos a dicha ubicación.

## 1.3 Impacto esperado

Con el desarrollo de esta aplicación web se espera poner a disposición del usuario una herramienta que le permita monitorizar en tiempo real el ruido generado cerca de los aeropuertos. Esto le facilitará a la ciudadanía que resida en estas zonas el realizar un seguimiento y control de cuáles son los modelos y rutas de aviones que mayor polución sonora generan.

Abordando la cuestión desde un punto de vista industrial, también se pretende facilitar la recogida de información sobre cómo los distintos motores, modelos de avión y trayectorias influyen en el ruido generado por los aviones. Esto abre la puerta a potenciar la utilización y fabricación de alternativas menos ruidosas.

## 1.4 Estructura de la memoria

La memoria de este trabajo se encuentra estructurada de la siguiente manera:

- **Introducción.** Exposición tanto de la motivación para realizar este proyecto como de los objetivos planteados para su desarrollo exitoso.
- **Estado del arte.** Estudio y comparación de aplicaciones que proporcionen funcionalidades similares a la que se pretende desarrollar.
- **Análisis del problema.** Se analiza lo que se espera de la aplicación y en base a ello se propone una solución válida, identificando las etapas por las que pasará su desarrollo, los posibles riesgos y las oportunidades de innovación y negocio.
- **Diseño de la solución.** Se identifican los distintos bloques que componen la aplicación para después realizar una descripción extensa del funcionamiento y comunicación entre ellos. También se enumeran las tecnologías que serán usadas para desarrollar la solución.
- **Desarrollo de la solución.** Se profundiza en la implementación de los distintos bloques identificados en el apartado anterior.
- **Pruebas y despliegue.** Se enumera un listado de las pruebas más relevantes llevadas a cabo para comprobar el funcionamiento de la aplicación y se explica cómo llevar a cabo su despliegue, mostrando también su funcionamiento.
- **Conclusiones.** Resumen del trabajo y relación entre los resultados obtenidos y los objetivos propuestos al inicio del mismo.

A estos capítulos se les suman tres anexos que proporcionan información adicional:

- **Elección de la API.** Análisis y comparación de las diferentes APIs disponibles que proporcionen datos en tiempo real sobre la trayectoria de los aviones, justificando la elección final.
- **Objetivos de Desarrollo Sostenible.** Justificación de la relación entre el trabajo desarrollado y los Objetivos de Desarrollo Sostenible.

- **Glosario.** Definición de términos relacionados con la materia.

# CAPÍTULO 2
# Estado del arte

Antes de comenzar el desarrollo de la aplicación web se ha realizado un estudio para obtener información sobre las características de otras soluciones que ya hayan sido desarrolladas y cumplan funcionalidades similares a la que el proyecto pretende abarcar. Este capítulo describe este análisis y realiza una reflexión final en la que se señalan las diferencias entre las soluciones estudiadas y la que se quiere desarrollar.

## 2.1 Soluciones similares

Resulta imposible realizar este análisis sin mencionar Envirosuite [3]. Este servicio combina estrategias de predicción, algoritmos de cálculo e información en tiempo real para proporcionar una fuente unificada de datos fiables relacionados con diferentes cuestiones medioambientales. Esta información es después suministrada a sus clientes para que puedan mejorar sus servicios, llegándose a cubrir varios sectores como la minería, el industrial o los relacionados con el tratamiento del agua. En el caso de la industria aerocomercial destacan los siguientes servicios: ANOMS y WebTrack.

#### ANOMS (Airport Noise and Operations Monitoring System)

Es un software que permite la monitorización y tratamiento preciso de la información relacionada con el ruido generado por los aviones y el seguimiento de sus trayectorias. Se basa en un sistema modular que se sirve de diversas fuentes de información: sensores de ruido, radares, información sobre el clima, etc. las cuales combina para proporcionar información lo más precisa posible que después pueda ser utilizada como apoyo para el diseño de un plan de reducción de ruido. ANOMS se define a si mismo como *el software líder a nivel mundial para sistemas de monitorización de ruido de aeropuertos, informes de ruido de aeropuertos y tecnología de seguimiento de vuelos* [4] y presenta diferentes servicios a analizar.

- **ANOMS.** Es el servicio base. Monitoriza y proporciona información en tiempo real sobre las emisiones de ruido generadas por los vuelos de aviones (figura 2.1). También posee una gran capacidad analítica y de reporte de información sobre planes de reducción del mismo. Entre la información devuelta se encuentra la exposición al ruido, el seguimiento de trayectorias e informes sobre el tráfico aéreo.

- **ANOMS Airline Compliance** [5]. Es una herramienta centrada en la gestión en aeropuertos de los procedimientos para la reducción de ruido (figura 2.2). Sus funciones permiten notificar sobre casos de cumplimiento de dichos procedimientos y casos en los que se debe mejorar la gestión. También ofrece otros servicios como la generación de informes relacionados con el ruido de los aviones.
- **ANOMS Carbon Emissions** [6]. Es una plataforma que evalúa de forma automática los niveles de emisión de gases de efecto invernadero y proporciona un cálculo de las emisiones de CO2 para cada una de las fases de que atraviesa un avión durante su vuelo.

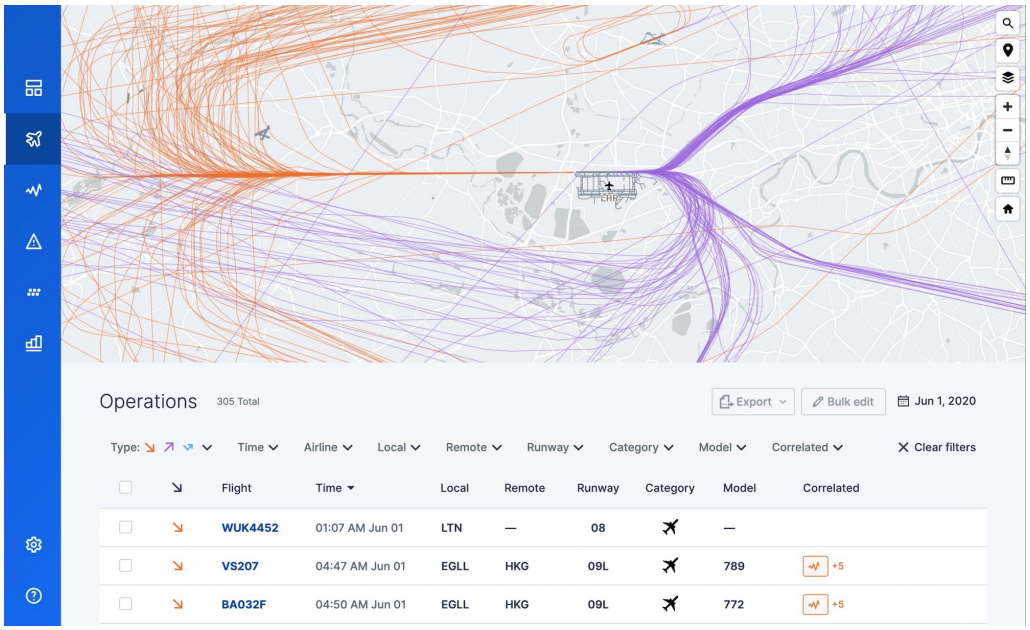


**Figura 2.1:** Panel de funcionamiento de ANOMS

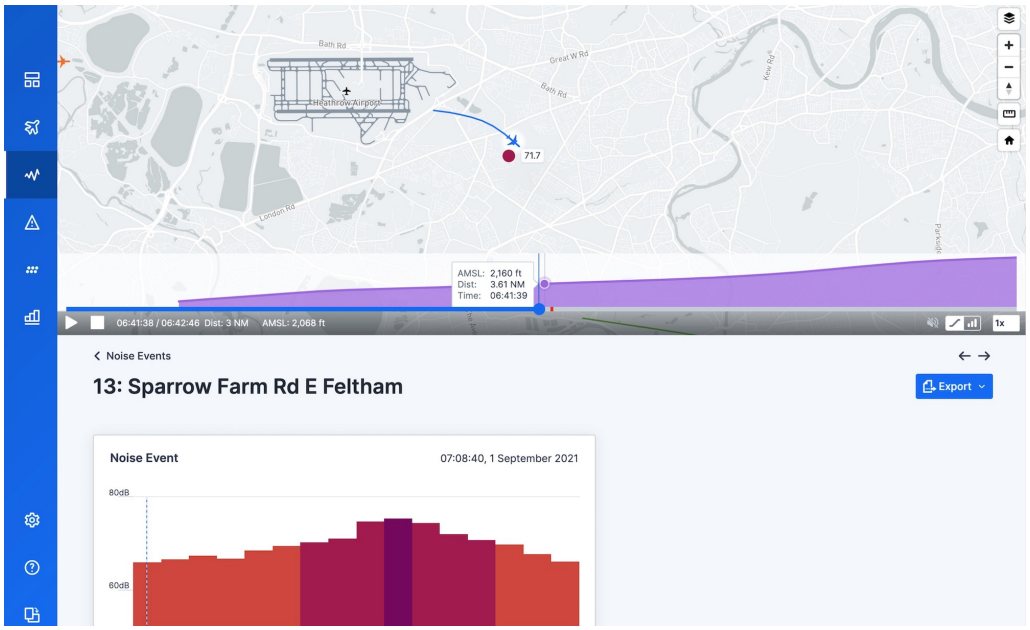


**Figura 2.2:** Detección de picos de ruido cerca de un aeropuerto utilizando ANOMS Airline Compliance

- **ANOMS Advanced** [7]. Presenta una funcionalidad similar a la del programa base ANOMS, existiendo la diferencia de que en este caso se trata de un software basado en la nube.

ANOMS resulta por tanto una herramienta muy potente para la recogida de datos relacionados con el ruido, disponiendo además de varias funcionalidades que se centran en las zonas cercanas a aeropuertos. El principal inconveniente que deriva de querer utilizar esta solución es que se trata de un sistema que no está enfocado para un uso personal de recogida y estudio de datos, sino para ser usado en aeropuertos. Para poder utilizarlo es necesario contactar con la empresa a través de un formulario en su página web, siendo el objetivo obtener atención personalizada sobre cómo mejorar las condiciones de un aeropuerto en concreto.

#### WebTrack

Se trata de una aplicación web que permite el análisis tanto histórico como en tiempo real de datos relacionados con vuelos de aviones [8]. Esta página incluye una función que permite medir el ruido generado por dichos aviones mediante la recogida de datos procedentes de sensores ubicados en distintos puntos geográficos (figura 2.3) que miden y almacenan datos de ruido de los últimos cinco minutos. La diferencia con el proyecto a desarrollar es que este no hace uso de sensores, sino de datos directamente relacionados con el avión como su posición o su modelo, lo que permite calcular el ruido generado por un avión en concreto en vez de calcular el ruido ambiente que habrá en un momento concreto. Además, al tener en cuenta la posición del usuario permite mayor flexibilidad, ya que pueden obtenerse datos de ruido para distintos puntos geográficos, mientras que si se usan sensores hay regiones cercanas al aeropuerto que pueden quedar sin cubrir.

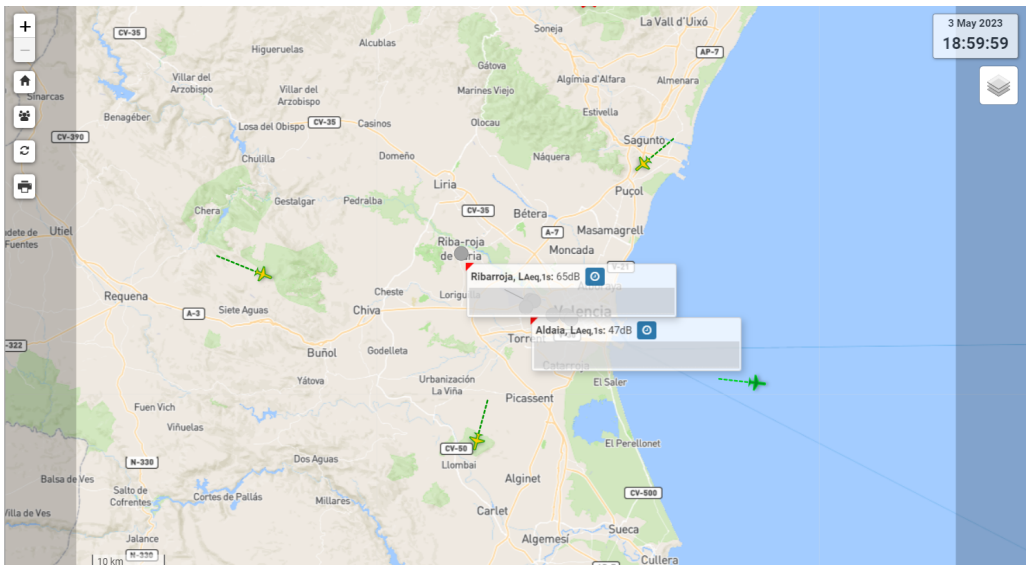


**Figura 2.3:** Medida de niveles de ruido con sensores utilizando WebTrack

A pesar de que ANOMS y WebTrack son los servicios principales a analizar, Envirosuite también proporciona otras soluciones relevantes relacionadas con la medición de ruido [3]:

- **NoiseOffice.** Se trata de un servicio por suscripción que proporciona acceso a grandes volúmenes de información relacionada con las emisiones de ruido (impacto de ruido, datos a largo plazo, informes rutinarios, etc.) además de herramientas para poder operar con dichos datos. El objetivo es ofrecer una herramienta funcional pero a un coste asequible.
- **Terminales de monitorización de ruido.** Envirosuite ofrece una amplia selección de terminales de monitorización de ruido diseñadas para exteriores y pensadas para una gran cantidad de condiciones climáticas. También existe la posibilidad de proporcionar al cliente herramientas de hardware que permitan la integración en otros sistemas de los datos recogidos en estas terminales.

Tal y como se ha mencionado al iniciar este capítulo, la mayoría de soluciones interesantes a analizar vienen proporcionadas por Envirosuite. Sin embargo, existe una aplicación que no está relacionada con esta plataforma y cuyo estudio también resulta relevante: NoiseTube.

#### NoiseTube

Es una aplicación móvil que permite utilizar el dispositivo móvil como un sensor de ruido [9], permitiendo además a los usuarios compartir los datos obtenidos, participando así en la construcción de un mapa de ruido [10]. La principal desventaja que plantea el uso de esta aplicación frente a la que se pretende desarrollar es, como ya se ha comentado en el caso de WebTrack, la imposibilidad de determinar la cantidad de ruido generada por un avión en concreto.

## 2.2 Propuesta

Tal y como se ha podido concluir a través del estudio de soluciones similares a la planteada, a día de hoy ya existen algunas alternativas interesantes a la hora de medir el ruido generado por los aviones en las cercanías a los aeropuertos. Algunos de los aspectos más interesantes son la posibilidad de gestionar estos datos a través de una plataforma basada en la nube o la estrecha relación que se pretende establecer con los aeropuertos a fin de mejorar las condiciones de contaminación acústica en los mismos.

Sin embargo, estas soluciones basan todo su funcionamiento en el uso de sensores para determinar el ruido ambiente en un momento concreto. Lo que se pretende con este proyecto es poder obtener datos individualizados para cada avión para que resulte posible extraer conclusiones en función de características como por ejemplo, el modelo o el tipo de motor. Además, el utilizar la ubicación del usuario amplía el número de lugares desde los que es posible realizar mediciones de ruido, ya que los sensores se encuentran instalados de forma estática en determinados puntos geográficos.

También cabe destacar que la mayoría de estas soluciones (especialmente las propuestas por ANOMS) trabajan con información y software bastante técnico, ya que se tratan de programas y aplicaciones dirigidas a personal con conocimientos sobre

la materia. Este proyecto pretende construir una aplicación web con una interfaz sencilla y con datos fácilmente interpretables por cualquier tipo de usuario, facilitando así el que pueda ser utilizada por gente que resida cerca de los aeropuertos.

# CAPÍTULO 3
# Análisis del problema

En este capítulo se realizará el análisis del problema con el objetivo de proponer la solución que mejor integre las características y funcionalidades que se espera que cumpla la aplicación. Como se verá a continuación, las diferentes opciones de diseño quedarán notablemente reducidas debido a los requisitos tan concretos que plantea el proyecto. Tras analizar esta cuestión se expondrá el plan de trabajo que se seguirá para llevar a cabo con éxito el desarrollo y se analizarán tanto los riesgos como las oportunidades de innovación y negocio que identificadas.

## 3.1 Análisis

Tal y como se ha visto en el capítulo anterior, existen multitud de soluciones que proporcionan servicios relacionados con el cálculo de ruido. En cierto modo, el objetivo de la aplicación que se pretende desarrollar no difiere mucho de el de estas soluciones: se quiere proporcionar un servicio que permita medir el ruido generado por los aviones en tiempo real. Tomando esto como referencia el abanico de opciones puede parecer en principio bastante grande: se puede escoger entre realizar un proyecto web o una aplicación de móvil, barajar la utilización de distintas técnicas para medir el ruido, proporcionar información a cada usuario o que por el contrario integrarla en una plataforma IoT con múltiples fuentes de datos, etc. Sin embargo, el proyecto que se quiere desarrollar presenta una serie de requisitos que van a delimitar rápidamente esta lista de opciones:

- La solución debe proporcionar datos de ruido para cada avión que se encuentre en la región de estudio.
- La solución debe proporcionar datos en tiempo real.
- La solución debe ser lo más accesible posible a todo tipo de usuarios.

**Cálculo del ruido**

La aplicación web que se quiere desarrollar debe ser capaz de asociar el perfil de ruido correspondiente a cada avión, permitiendo así conocer qué modelos de avión y de motor son menos ruidosos, o en qué puntos de la trayectoria se percibirá un mayor nivel de ruido. Por tanto, un sensor, a pesar de ser un instrumento cómodo

de usar y preciso en sus medidas, no resultará útil en esta situación. Es por ello que queda descartado de antemano cualquier tipo de solución en la que esté implícito el uso de sensores de ruido. Este descarte sitúa el foco en la necesidad de calcular estos perfiles de ruido por cuenta propia. Aunque este aspecto será comentado más en detalle en el apartado de *Solución propuesta*, conviene aclarar que el tipo de cálculo que se realizará requerirá de datos concretos (conocer la posición del avión, saber si se está acercando o alejando del aeropuerto...), lo que acota aún más el abanico de posibilidades.

#### Obtención de datos en tiempo real

Para obtener los datos necesarios para el cálculo de ruido será necesario recurrir a APIs que proporcionen información sobre los aviones y sus trayectorias en tiempo real. Estas APIs generalmente requieren que el usuario se registre en la página web en cuestión y adquiera alguno de los planes que se ponen a su disposición. Una gran parte del trabajo de investigación ha estado centrado en encontrar información sobre las diferentes APIs disponibles en el mercado, analizando las ventajas, desventajas e incluso fallos de estas, a fin de escoger la que presente un mejor funcionamiento. Todo este trabajo ha sido documentado en el apéndice *Elección de la API*.

#### ¿Aplicación web o móvil?

Una vez abordados los aspectos relacionados con la lógica interna de la aplicación queda reflexionar sobre cómo se le proporcionará al usuario la información que solicite, si a través de una aplicación web o mediante la instalación de una aplicación móvil. En este caso se ha optado por desarrollar una aplicación web, siendo estas algunas de las ventajas que han influido en la toma de esta decisión [17]:

- **Accesibilidad.** Uno de los objetivos que se pretenden alcanzar es crear una aplicación que sea accesible para cualquier tipo de usuario. Se ha considerado que una aplicación web tendrá un mayor alcance debido a que no es necesario realizar ningún tipo de instalación para poder usarla. Además, resultará sencillo compartirla, por ejemplo, a través de un enlace o de un correo electrónico.
- **Facilidad de actualización.** Una aplicación web se puede actualizar fácilmente y sin interrupción para los usuarios, lo que significa que los cambios y mejoras pueden implementarse de manera más rápida y sencilla. En cambio, una aplicación móvil requiere que los usuarios la actualicen manualmente, además de que las nuevas versiones deberán ser aprobadas por la tienda digital en la que se distribuya, lo que puede causar interrupciones en el servicio.

En resumen, la funcionalidad del proyecto a realizar estará condicionada por la necesidad calcular perfiles de ruido a partir de datos en tiempo real. A continuación se realizará una descripción general del funcionamiento de la aplicación web y se enumerarán las fases atravesadas a lo largo de su desarrollo, comentando las acciones que se han llevado a cabo en cada una de ellas.

## 3.2 Solución propuesta

### 3.2.1. Funcionamiento general

El proyecto a desarrollar será una aplicación web que contará con una página principal en la que el usuario pueda solicitar el conocer los perfiles de ruido de los aviones cercanos al aeropuerto de Manises pulsando sobre un botón 'Calcular ruido'. En esta página el usuario podrá indicar su ubicación actual o algún punto geográfico del mapa de Valencia como parámetro a incluir dentro de dichos cálculos. Una vez pulsado este botón, la página mostrará el listado de aviones que tengan como origen o destino el aeropuerto de Manises y el ruido que percibe el usuario desde la posición indicada para cada uno de dichos aviones. También se mostrarán algunos otros datos relevantes sobre el vuelo como el identificador del vuelo, el modelo de avión, o el IATA de los aeropuertos de origen y destino. Se omitirán los aviones cuya emisión de ruido no sea perceptible por el usuario (es decir, aviones que tengan como origen o destino el aeropuerto de Manises pero que se encuentren demasiado lejos del mismo).

### 3.2.2. Modelo de ruido ECAC

Previamente se ha hecho hincapié en la necesidad de calcular el ruido para cada avión en base a datos como su modelo, tipo de motor, etc. En esta sección se proporcionan algunos detalles sobre el modelo utilizado para realizar este cálculo.

#### Modelo de ruido

La construcción de perfiles de ruido se basará en el método de computación de contornos de ruido en las cercanías a los aeropuertos incluido en el ECAC.CEAC [11]. Se utilizará un modelo de ruido simplificado en el que se considera un avión como una fuente uniforme de ruido. Este modelo se basa en la trayectoria adoptada por el avión al aproximarse o alejarse del aeropuerto, describiéndola como un conjunto de segmentos y calcula el ruido para cada uno de ellos, tomando como referencia un punto de observación (la posición del usuario que ejecute la aplicación) y la distancia que existe con dicho punto.

#### Métricas de ruido

Un avión es una fuente sonora en movimiento, lo que significa que sus niveles de ruido pueden variar a lo largo del tiempo. Por ejemplo, el ruido generado aumentará cada vez más en el proceso de descender aproximándose hacia el aeropuerto. En estos casos es útil medir el máximo nivel de ruido (*Lmax*) de un evento en particular. Sin embargo, *Lmax* deja fuera de la ecuación la duración del evento, siendo esta una variable de gran importancia (el nivel máximo de ruido emitido por un disparo de arma de fuego y un tren de mercancías es similar, pero la duración de ambos eventos es muy distinta). Para tener en cuenta también el aspecto temporal se pueden utilizar distintas medidas, siendo SEL (*Sound Exposure Level*) la más interesante en este caso, ya que es la utilizada para el modelado de contornos de ruido de aeronaves. SEL

es una medida acumulativa descrita en la norma ISO 1996-1 [14] que representa la energía acústica total de un evento considerando que su duración ha sido de un segundo. [13]

#### Uso de ANP Database

La metodología ECAC obtiene el nivel de ruido de un evento haciendo uso de la base de datos ANP (Aircraft Noise and Performance Database) [15]. Esta base de datos recolecta, verifica y pone a disposición de sus usuarios información sobre los distintos modelos de avión que después puede ser utilizada en el modelado de perfiles de ruido. Mediante su uso se realiza el cálculo de perfiles de ruido en las zonas que rodean los aeropuertos [16], tomando como base la construcción de una tabla de datos Potencia-Distancia-Ruido (Noise-Power-Distance o NPD) para proporcionar métricas de ruido (SEL y *Lmax*) a partir de la distancia del observador y la configuración de potencia de la aeronave.

Todos los cálculos basados en el modelo ECAC se realizarán, como se verá en los capítulos *Diseño de la solución* y *Desarrollo de la solución*, utilizando una librería que a partir de un conjunto de datos en un formato determinado sea capaz de proporcionar el perfil de ruido correspondiente.

### 3.2.3. Plan de trabajo

A continuación se enumerarán las fases atravesadas a lo largo del desarrollo, comentando las acciones que se han llevado a cabo en cada una de ellas.

El desarrollo de este proyecto ha seguido las fases típicas del ciclo de vida del software, ajustándose a un modelo en V en el que en cada etapa se han ido realizando las comprobaciones necesarias para verificar si resultaba adecuado pasar a la siguiente fase o si era necesario corregir errores de etapas anteriores [2]. A continuación se mencionan las acciones realizadas en cada una, sin entrar en demasiado detalle ya que esto es tarea de los siguientes capítulos, *Diseño de la solución* y *Desarrollo de la solución*.

| Semana | Etapa |
|---|---|
| Semana 0 | Planificación y análisis. |
| Semana 1<br>Semana 2<br>Semana 3 | Diseño. |
| Semana 4<br>Semana 5<br>Semana 6<br>Semana 7<br>Semana 8<br>Semana 9<br>Semana 10<br>Semana 11<br>Semana 12 | Implementación, pruebas y despliegue. |
| Semana 13<br>Semana 14 | Documentación. |

**Tabla 3.1:** Distribución semanal de las tareas.

- **Planificación.** Esta etapa ha sido principalmente centrada en la planificación temporal del proyecto. Se ha realizado una distribución de tareas delimitadas por períodos de una o dos semanas, teniéndose en cuenta cierta flexibilidad para modificar las asignaciones en caso de que aparecieran imprevistos. La distribución final queda tal y como se puede apreciar en la tabla 3.1.

  Otro asunto importante a tratar era el aspecto financiero, ya que muchas de las APIs consideradas como posibles alternativas para la obtención de datos funcionaban mediante planes de pago. Sin embargo, este aspecto finalmente acabó fuera de la ecuación al optarse por una alternativa gratuita para la obtención de datos.

- **Análisis.** Esta fase ha estado centrada tanto en la búsqueda de una API adecuada para la obtención de datos como en el entendimiento de la librería de ruido. El primer proceso aparece documentado en el apéndice *Elección de la API*, mientras que el segundo se ha basado en el estudio de los parámetros de entrada que requiere dicha librería. Para ello, se ha utilizado su comando *ANCM_Lib_Win64 -H*, que proporciona información detallada sobre cada uno de ellos y el formato en el que se deben proporcionar.

- **Diseño.** Una vez comprendido cuál debe ser el funcionamiento del software se ha realizado el diseño de la estructura general del proyecto, identificándose así las diferentes partes del mismo y detallándose cómo deben comunicarse entre ellas. Al tratarse de un proyecto web que debe recibir datos de una fuente

externa y utilizarlos para ejecutar nuestro programa de cálculo de ruido, se ha creído conveniente diseñar una estructura cliente-servidor en la que sea el cliente el que se comunique tanto con el servidor como con la API. Este proceso finaliza con la elección de las herramientas que se utilizarán (API, lenguajes de programación, entorno de desarrollo, etc.) y pone nuevas cuestiones sobre la mesa relacionadas con la implementación y el desarrollo de la solución que nos llevan directamente a la siguiente fase.

- **Implementación.** Se profundiza en el desarrollo del código necesario para dotar de funcionalidad a la aplicación. Cabe destacar el importante papel que juega al inicio de esta fase la investigación y comparación de diferentes APIs realizada en fases anteriores a fin de seleccionar la que mejor se ajuste a lo que el proyecto necesita, ya que un imprevisto en este aspecto puede requerir el realizar cambios que modifiquen sensiblemente código que ya se haya implementado.

- **Pruebas.** Para verificar que la aplicación funciona como se espera es necesario realizar una serie de pruebas en cada uno de los puntos de su estructura. En el caso de la API nos interesará saber si proporciona datos de calidad (datos en tiempo real relacionados con el aeropuerto de Manises), en el del servidor necesitaremos comprobar si ejecuta correctamente la librería de cálculo de ruido y en el cliente habrá que verificar si procesa correctamente los datos recibidos. También resulta necesario verificar si las comunicaciones entre estos componentes se realizan de manera exitosa. Esta fase está, por tanto, altamente relacionada con la anterior, resultando ideal realizar iteraciones entre ambas, lo que justifica la elección del modelo en V mencionado al comienzo de esta sección.

- **Instalación o despliegue.** El completar esta fase ha resultado algo casi automático, ya que aunque en algunos tramos del desarrollo se han utilizado datos predefinidos para realizar comprobaciones, ha habido una gran parte de la implementación en la que se ha optado por realizar breves despliegues para comprobar el correcto funcionamiento de todos los componentes. Puede considerarse que esta fase se ha desarrollado de forma iterativa junto a las de implementación y pruebas.

- **Uso y mantenimiento.** Dado que la aplicación se ha desarrollado en un ambiente académico sin que existan por el momento planes reales de distribución del software esta fase ha sido omitida.

- **Documentación.** Se describe todo el proceso de análisis, investigación e implementación llevado a cabo. Aunque esta tarea está presente durante todo el desarrollo del proyecto su culminación se realiza en las últimas semanas, cuando la aplicación ya ha sido desarrollada por completo.

## 3.3 Análisis de riesgos

La aplicación web a desarrollar depende por completo de los datos proporcionados en tiempo real sobre los aviones, ya que sin esta información no resulta posible el cálculo de los perfiles de ruido. El hecho de que estos datos sean proporcionados por

una API plantea una serie de riesgos tecnológicos y de integración que se expondrán a continuación.

#### Datos desactualizados

Aunque algunos datos como el modelo del avión o el tipo de motor serán los mismos para todo su trayecto, existen otros como su posición que deben estar constantemente actualizándose. Una frecuencia demasiada baja de actualización repercutiría en la precisión de los cálculos realizados por nuestra aplicación web.

- **Impacto sobre el proyecto:** Los cálculos a partir de datos desactualizados pueden dar como resultado perfiles de ruido imprecisos o totalmente incorrectos, dependiendo de lo baja que sea la frecuencia de actualización.
- **Medidas:** La mayoría de APIs proporcionan no solo los datos sobre la trayectoria y modelo del avión sino también una marca de tiempo Unix [18] que representa el momento en el que fue recogida la información. Se puede establecer otra marca de tiempo al enviar la solicitiud a la API y descartar los datos proporcionados por esta en caso de que la diferencia entre ambas marcas sea muy grande.

#### Caída en la disponibilidad de los datos

La API, al igual que cualquier otro servicio en línea, puede fallar o dejar de estar disponible durante un período de tiempo, bien a causa de alguna tarea de mantenimiento o como consecuencia de sufir algún tipo de ataque informático.

- **Impacto sobre el proyecto:** Una caída en la disponibilidad de la API implica que no exista una respuesta a las peticiones que se envíen solicitando los datos necesarios para los cálculos. Esto se traduciría en la imposibilidad de proporcionar datos de ruido al usuario.
- **Medidas:** Muchas APIs permiten a sus clientes consultar el porcentaje de disponibilidad que han tenido a lo largo de los días y semanas. La solución a este problema pasa por analizar estos porcentajes y escoger APIs que presenten un porcentaje alto de disponibilidad, a ser posible cercano al 100 %. También puede establecerse una lista de APIs de emergencia que consultar en caso de que la principal falle, aunque esto supondría un trabajo extra de investigación e implementación, ya que no todas las APIs devuelven los mismos conjuntos de datos o datos en el mismo formato.

#### Problemas de escalabilidad y rendimiento

La escalabilidad y el rendimiento del proyecto web vendrán determinadas por la escalabilidad y rendimiento de la API seleccionada. Esto puede suponer un problema si se da un tráfico alto de usuarios que emitan grandes volúmenes de peticiones y la API no está preparada para ello.

- **Impacto sobre el proyecto:** Si la API no es capaz de manejar un alto número de peticiones simultáneas los usuarios que quieran utilizar la página web puede experimentar problemas de rendimiento y lentitud en la respuesta.
- **Medidas:** Uso de herramientas de balanceo de carga que permitan optimizar el rendimiento de la aplicación web enviando solicitudes de manera eficiente.

#### Cambios en la API

La lógica interna de la aplicación web se construye adaptándose al contenido de las respuestas que recibirá por parte de la API, pero este contenido puede cambiar con el tiempo. Una alteración en la estructura de los datos recibidos o su en formato puede provocar que el código desarrollado deje de ser válido o genere respuestas erróneas.

- **Impacto sobre el proyecto:** El procesamiento de datos dejaría de realizarse correctamente, lo que puede traducirse en un cálculo incorrecto de los perfiles de ruido e incluso puede generar errores que hagan que la aplicación deje de funcionar.
- **Medidas:** Consultar con regularidad las actualizaciones de la API, manteniendo relación si es necesario con el equipo de desarrollo para estar al día. También se puede optar por realizar una implementación flexible y modular que haga fácil actualizar la API es caso de cambios.

#### Costo y facturación excesivos

Las APIs platean planes de pago cuyo precio aumenta de manera proporcional al número de consultas que se realizan. Esto puede llegar a suponer un problema en caso de que exista un alto tráfico de peticiones desde la aplicación web.

- **Impacto sobre el proyecto:** Alcanzar el número límite de consultas ofrecidas en el plan contratado tendría un efecto similar a una caída en la disponibilidad de la API, ya que se dejarían de suministrar los datos necesarios para los cálculos. La contratación de planes que permitan grandes volúmenes de consultas también puede suponer un problema a nivel de presupuesto.
- **Medidas:** Limitar el número de consultas que los usuarios pueden realizar o almacenar en caché los datos que no cambien con frecuencia, además de disponer de un sistema de facturación y contablidad adecuado para monitorear los costos. También es posible contactar con el equipo de desarrollo de la API para llegar a un acuerdo sobre un plan especializado que se adapte a las necesidades del proyecto.

Además de los riesgos relacionados con el correcto funcionamiento del proyecto, también hay que tener en cuenta que se pretende proporcionar un servicio al usuario, por lo que también existen una serie de riesgos relacionados con su experiencia que deben ser considerados.

#### Dificultades interpretando la información

El usuario de a pie no estará necesariamente familiarizado con algunos de los conceptos con los que la aplicación necesita trabajar para poder proporcionar la información.

- **Impacto sobre el proyecto:** Si la información se presenta de forma poco clara o utilizando términos complejos el usuario puede no ser capaz de interpretarla, lo que hará que su uso de la aplicación caiga en picado.
- **Medidas:** Mostrar la información al usuario de la forma más sencilla posible, evitando tecnicismos innecesarios y resaltando los datos de mayor relevancia, como el nivel de ruido.

#### Interfaz de usuario excesivamente compleja

Muchas de las aplicaciones relacionadas con la obtención de datos sobre trayectorias de aviones y perfiles de ruido presentan interfaces complejas debido al gran número de funciones que ofrecen. Esto es útil para usuarios que poseen un conocimiento medio o incluso avanzado sobre la materia, pero en este caso un panel con muchas opciones puede llegar a generar cierta frustración.

- **Impacto sobre el proyecto:** Una interfaz demasiado compleja puede dificultar al usuario el saber cómo debe operar para calcular el ruido. Si el usuario queda saturado por la complejidad de la página puede decidir dejar de utilizarla.
- **Medidas:** Diseñar interfaces lo más sencillas e intuitivas posibles que faciliten al usuario el comprender y recordar su uso.

#### Falta de retroalimentación

Prácticamente la totalidad del funcionamiento de la aplicación se lleva a cabo sin que el usuario sea consciente de ello, ya que lo único que tiene que hacer es presionar un botón para que se efectúen los cálculos. No proporcionarle información en todo momento puede llevarle a confusión, ya que no será capaz de distinguir si la aplicación está funcionando correctamente o no.

- **Impacto sobre el proyecto:** Un usuario al que le genere desconcierto no conocer en qué estado se encuentra la aplicación o si está funcionando con normalidad puede tener dudas sobre la fiabilidad de los datos proporcionados.
- **Medida:** Hacer uso de diálogos en todo momento, tanto a través de mensajes que aparezcan como parte de la página como utilizando alertas para indicar al usuario si los datos han sido procesados correctamente o si ha ocurrido algún error.

## 3.4 Oportunidades de negocio e innovación

La solución que se pretende desarrollar se caracteriza por su versatilidad. El uso del modelo ECAC hace que entre en juego la posición del observador como una de las variables para calcular el ruido, lo que abre la posibilidad al desarrollo de aplicaciones centradas en usuarios poco especializados pero que deseen conocer datos relacionados con la emisión de ruidos en el área en el que se encuentren. El uso de ANP Database también permite relacionar la solución directamente con el cálculo de perfiles de ruido asociados a distintos tipos de avión o modelos de motor.

Además, este proyecto puede funcionar no solo como aplicación individual, sino que también existe la posibilidad de que, tras un proceso de integración, sea compatible con otras aplicaciones con gran potencial relacionadas con esta cuestión (como las analizadas en el capítulo *Estado del arte*), algo que resulta muy positivo si se pretende aumentar el abanico de funciones de dichas aplicaciones.

Finalmente, cabe destacar que, aunque esta solución está pensada para realizar cálculos utilizando datos reales proporcionados por alguna API, su implementación hace posible el diseño de una versión en la que sea el propio usuario el que introduzca estos valores, algo que puede ser interesante de cara a utilizar el proyecto como una especie de simulador de ruido.

# CAPÍTULO 4
# Diseño de la solución

En este capítulo se identificarán los distintos bloques que componen la aplicación, profundizando en su funcionamiento y en la forma en la que se comunican entre ellos. Además, se seleccionarán las tecnologías que se emplearán para su implementación.

## 4.1 Arquitectura del sistema

El sistema presenta una arquitectura cliente-servidor diseñada para garantizar un flujo continuo de información entre los diferentes componentes del sistema y ofrecer una solución rápida y precisa para calcular los perfiles de ruido de los aviones. Este sistema está compuesto por los siguientes bloques:

- La API que proporciona la información en tiempo real sobre los aviones.
- La página web desde la que el cliente solicita conocer los perfiles de ruido.
- El servidor que ejecuta la librería que calcula dichos perfiles de ruido.
- La API que proporciona un mapa a modo de apoyo visual para seleccionar ubicaciones geográficas.

Para que todo el mecanismo funcione es necesario establecer una comunicación entre todos los componentes que permita gestionar el intercambio de datos. El cliente solicitará a la API los datos en tiempo real, después los procesará para ajustarlos al formato requerido por la librería y se los enviará al servidor. El servidor recibirá estos datos, calculará el perfil de ruido a partir de ellos y se lo devolverá al cliente, el cual se lo mostrará al usuario a través de la página web. Este intercambio de datos aparece representado gráficamente en la figura 4.1.

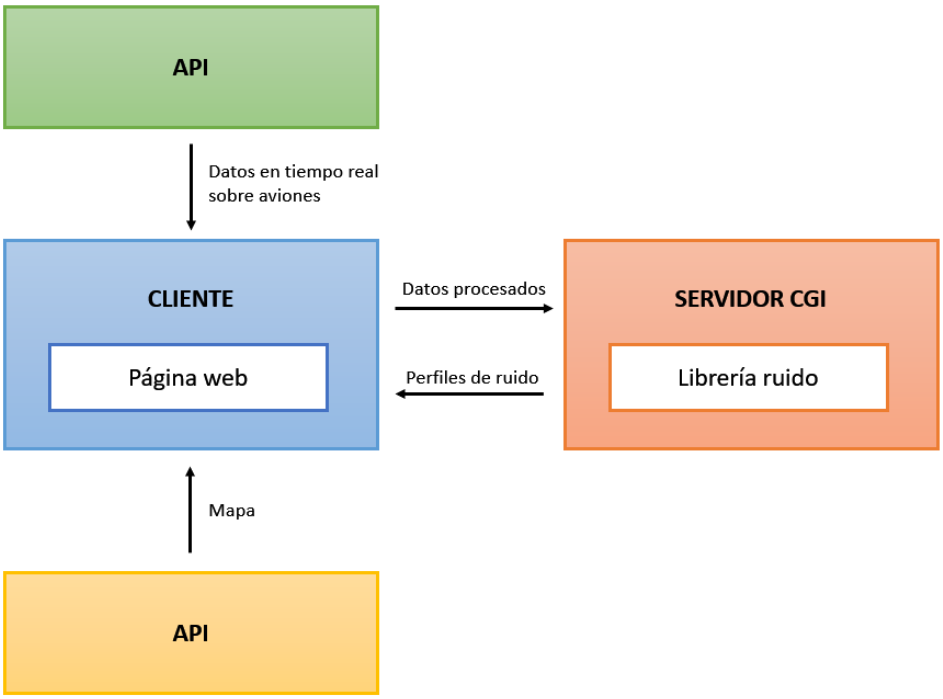


**Figura 4.1:** Esquema de funcionamiento de la aplicación web.

Una vez esclarecida la estructura general, vamos a profundizar en el funcionamiento de cada uno de estos componentes y en los detalles de la comunicación entre ellos.

## 4.2 Diseño detallado

### 4.2.1. API

La librería requiere de datos como la latitud, longitud, altitud, posición y modelo del avión del que se pretende calcular el perfil de ruido. Además, estos datos deben ser proporcionados en tiempo real, para poder ser utilizados en el cálculo de aviones que se encuentren cerca del aeropuerto en el momento de uso de la web. La tarea de recolectar todos estos datos y ponerlos a disposición del cliente le corresponde a la API.

La mayoría de APIs que proporcionan este tipo de información almacenan datos de todos los vuelos activos en el mundo. Un usuario puede solicitarlos a través de una petición a la que se le pueden añadir algunos filtros para acotar el resultado de la consulta. La API devolverá como respuesta a dicha petición un vector de aviones, indicando para cada uno las características antes mencionadas además de otra información que puede resultar muy útil para operar de forma cómoda. Entre los datos más relevantes devueltos por este tipo de APIs se encuentran los siguientes:

- **ICAO 24.** Se trata de un número único de 24 bits que identifica de forma unívoca cada avión.

- **Aircraft ICAO.** Es un código alfanumérico de 2 a 4 caracteres que identifica modelo del avión.
- **Latitud, longitud y altitud.** Posición geográfica del avión. La latitud y la longitud suelen proporcionarse en grados, mientras que la altitud se proporciona en metros.
- **Velocidad.** Velocidad horizontal del avión. Se suele proporcionar en km/h.
- **Updated.** Proporciona una marca de tiempo Unix [18] que representa el momento en el que fueron recogidos los datos proporcionados. Esta característica no es necesaria para realizar el cálculo de ruido pero sí es útil para verificar que los datos proporcionados han sido recogidos en tiempo real.

Dado que nos interesa recoger datos relacionados con aviones que tengan como origen o destino el aeropuerto de Manises, también nos resultará interesante conocer el ICAO o IATA de los aeropuertos de origen y de destino, aunque este es un dato que no todas las APIs proporcionan.

#### Validez de los datos de la API

Hay que tener en cuenta que la información proporcionada por las APIs no siempre es adecuada para el uso que se pretende en este proyecto. Es posible encontrarse con casos en los que la información no se actualiza o lo hace con una frecuencia menor de la debida (esto puede comprobarse analizando el parámetro *Updated*). También son frecuentes los casos en los que la API no devuelve suficiente información, bien por que no detecte ciertos vuelos o bien porque no tenga registrado algún aeropuerto. Esto puede deberse a limitaciones del plan utilizado (normalmente los planes gratis ofrecen menos información) o a limitaciones de la propia API.

#### Comunicación con el cliente

Para poder obtener los datos en el cliente será necesario realizar una petición a la API deseada. Esta petición puede contener de forma opcional una serie de parámetros (ICAO 24, aeropuerto de origen o destino, etc.) que actuarán a modo de filtro y necesariamente deberá hacer uso de una *key* única que sirve para identificar a cada usuario y se le proporciona cuando este se crea una cuenta en la página de la API. La información suele ser devuelta en formato JSON, aunque también puede devolverse en otros formatos como XML o CSV.

### 4.2.2. Página web

La página web se ejecutará en el cliente. Mediante una interfaz sencilla muestra información resumida sobre sus funciones al usuario a través de algunos textos. Haciendo uso de un mapa interactivo se puede seleccionar el punto geográfico que se desea indicar como posición del observador, siendo posible utilizar la ubicación actual de dicho usuario. Una vez escogida la posición deseada se debe pulsar en el botón 'Calcular ruido' para iniciar el proceso de envío de la petición a la API a partir

de la que se obtienen los datos necesarios. Estos datos son procesados y enviados al servidor CGI a través de la lógica interna de la página, que también es la encargada de mostrar la respuesta por pantalla.

#### Selección de la posición del observador

La página presenta un mapa en el que aparece marcada la ubicación del aeropuerto de Manises. Cada vez que el usuario hace clic sobre algún punto de este mapa, la latitud y longitud de la posición del observador se actualizan. Estos cambios en la posición son visibles en la propia página, ya que al lado del mapa aparece un pequeño texto describiendo esta funcionalidad y señalando cuál es la última posición indicada. El usuario también puede escoger utilizar su ubicación actual como posición, la cual se resaltará sobre el mapa.

#### Procesado de datos de la API

La página web será la encargada de enviar la petición a la API cuando el usuario pulse el botón 'Calcular ruido' y de recibir la información. Típicamente la información recibida será un vector de aviones que cumplan con las características y filtros especificados en la consulta. Para cada uno de estos aviones se deberá realizar el mismo procesamiento antes de que sus datos puedan ser enviados al servidor CGI:

- **Convertir latitud, longitud y altitud.** La latitud y longitud proporcionadas por la API suelen estar en grados y la altitud en metros. Sin embargo, la librería necesita que los datos sobre posición que le son proporcionados se encuentren en pies, por lo que habrá que realizar las conversiones oportunas. Además, la librería considera el aeropuerto de Manises el centro de coordenadas (0,0), por lo que también será necesario expresar las coordenadas del avión en función de las del aeropuerto de Manises.
- **Calcular si el avión está rodando.** En el momento en el que avión se encuentre en la pista de aterrizaje comenzará a rodar, lo que influirá en el ruido. Es necesario determinar a partir de la altitud si el avión se encuentra en la pista de aterrizaje o no.
- **Convertir velocidad a pies/segundo.** La API proporciona la velocidad en km/h, sin embargo, tal y como se ha mencionado en puntos anteriores, la librería necesita que los datos estén expresados en pies. Por tanto, la velocidad debe ser proporcionada en pies/segundo.
- **Determinar si el avión está ascendiendo o descendiendo.** Conociendo el aeropuerto desde el que ha despegado el avión podemos determinar si se dirige al aeropuerto de Manises o ha comenzado su trayecto desde el mismo. En el primer caso se considerará que el avión está descendiendo, y en el segundo que está ascendiendo.
- **Determinar el tipo de motor del avión.** La librería requiere que se le indiquen dos parámetros para cada avión:

- Si se utiliza un motor acoplado en el ala (*wing mounted*) o en el fuselaje (*fuselage mounted*).
- Si se utiliza un motor de tipo *turbo-fan* o *turbo-prop* [28].

El tipo de motor dependerá del modelo del avión, el cual será uno de los parámetros proporcionados por la API.

- **Localizar el fichero .csv en base al modelo.** Lo último que requiere la librería para poder calcular el perfil de ruido es que se le proporcione un .csv con un conjunto de datos que relacionan el modelo del avión con el ruido que genera en función de diversos parámetros. Este tipo de ficheros .csv se obtienen de la página *ANP Database* [15]. Una vez obtenidos, puede relacionarse el modelo del avión con el nombre del fichero para que queden asociados. Además, este fichero .csv también contiene información sobre el tipo de motor utilizado por ese modelo en concreto. Para relacionar todas estas variables se ha consultado con frecuencia la lista de vuelos del aeorpuerto de Manises para saber los modelos de avión más frecuentes en este aeropuerto y posteriormente se ha confeccionado la tabla 4.1 que relaciona cada modelo con su ICAO, el tipo de motor y el nombre del .csv. Como se puede observar, hay algunos modelos para los que no se ha conseguido encontrar el fichero .csv correspondiente. Es por ello que se han realizado una serie de aproximaciones a otros modelos de los cuales sí que se ha podido obtener este archivo:

  - El modelo Airbus A320neo se ha considerado como un Airbus A320.
  - El modelo Airbus A321neo se ha considerado como un Airbus A321.
  - El modelo Bombardier CRJ se ha considera como un Bombardier CL-600-2515.
  - El modelo Boeing 757-200 se ha considerado como un Boeing 757-300.
  - Los modelos Cessna 525A Citation CJ2 y Cessna 525 se han considera como un Cessna Citation CJ4 525C.

  Para estas aproximaciones se ha confeccionado la tabla 4.2, omitiéndose tanto los modelos Airbus A320 y A321 (ya que ya aparecen en la tabla original) como los ICAOs de estos modelos, ya que los que se usarán serán los de los modelos originales. Cabe destacar que puede darse el caso de que para alguno de los aviones proporcionados por la API no se posea el .csv deseado. En ese caso, el avión se descartará y no formará parte de la lista de aviones que se muestren al usuario.

#### Área de análisis limitada

La API devolverá datos que, como se verá en el capítulo *Desarrollo de la solución*, pueden filtrarse en base a distintos criterios. Sin embargo, no resulta posible filtrarlos por proximidad, es decir, mostrando únicamente datos de aviones cercanos al aeropuerto. Este filtro resulta imprescindible, ya que no tiene sentido proporcionar información sobre el ruido que generen aviones que se encuentran a cientos de kilómetros del aeropuerto. El momento en el que resulta interesante realizar mediciones

| Modelo | ICAO | Wing/Fuselage | Turbofan/turboprop | .csv |
|---|---|---|---|---|
| Airbus A319 | A319 | Wing | Turbofan | NPD_data_A319-131 |
| Airbus A320 | A320 | Wing | Turbofan | NPD_data_A320-232 |
| Airbus A321 | A321 | Wing | Turbofan | NPD_data_A321-232 |
| Airbus A320neo | A20N | Wing | Turbofan | |
| Airbus A321neo | A21N | Wing | Turbofan | |
| Bombardier CRJ | CRJX | Wing | Turbofan | |
| Boeing 737-800 | B738 | Wing | Turbofan | NPD_data_737-800 |
| Boeing 757-200 | B752 | Wing | Turbofan | |
| Boeing 767-300ER | B763 | Wing | Turbofan | NPD_data_767300 |
| Embraer E190 | E190 | Wing | Turbofan | NPD_data_EMB190 |
| ATR 72 | AT76 | Wing | Turboprop | NPD_data_ATR72 |
| Cessna 525A Citation CJ2 | C25A | Fuselage | Turbofan | |
| Cessna 525 | C525 | Fuselage | Turbofan | |

**Tabla 4.1:** Modelos relacionados con su ICAO, tipo de motor y fichero CSV.

| Modelo | Wing/Fuselage | Turbofan/turboprop | .csv |
|---|---|---|---|
| Bombardier CL-600-2515 | Wing | Turbofan | NPD_data_CRJ9-ER |
| Boeing 757-300 | Wing | Turbofan | NPD_data_757300 |
| Cessna Citation CJ4 525C | Fuselage | Turbofan | NPD_data_CNA525C |

**Tabla 4.2:** Aproximaciones a modelos relacionados con su tipo de motor y fichero CSV.

es cuando el avión aún se encuentra ascendiendo o está lo suficientemente cerca del aeropuerto para comenzar el descenso, algo que sucede entre diez y veinte minutos después del despegue o antes del aterrizaje y aproximadamente en un área de veinte kilómetros alrededor del aeropuerto [19]. Debido a esto, parte del procesamiento de datos consistirá en descartar todos los aviones cuya latitud y longitud no se encuentre dentro de este área.

Tampoco es útil que la posición proporcionada por el usuario sea demasiado lejana al aeropuerto, por lo que este filtro también se le aplicará a este dato. Para facilitarle al usuario el seleccionar puntos que cumplan con esta condición el mapa muestra un círculo alrededor del aeropuerto que colorea y delimita el área de exposición, por lo que lo único de lo que deberá asegurarse es de que al hacer clic sobre el mapa se esté seleccionando un punto dentro de este círculo. En caso contrario, al pulsar sobre el botón 'Calcular ruido' la página emitirá una alerta y será necesario introducir una nueva posición.

#### Conexión con el servidor CGI y volcado de resultados en la página

Una vez se hayan procesado los datos tal y como se ha detallado en las secciones anteriores será necesario establecer una comunicación con el servidor CGI para enviárselos. La página web es también la encargada de procesar la respuesta a dicha petición y mostrarle los datos al usuario. La información se presentará en forma de tarjetas, mostrándose una por cada avión. En esta tarjeta aparecerá resaltado el rui-

do en decibelios que está generando el avión, así como otros datos (identificador del vuelo, modelo del avión e IATA del aeropuerto de origen y de destino) que permitan identificar cada vuelo y diferenciarlo del resto.

### 4.2.3. Servidor CGI

El servidor CGI es quizás el componente con la funcionalidad más sencilla. Sin embargo resulta clave en el proceso, ya que es el responsable de recibir los datos proporcionados por la página web y proporcionárselos a la librería para que esta calcule el perfil de ruido. Para ello, ha de confeccionar una orden de línea de comandos que incluya todos estos parámetros y que filtre la respuesta obtenida por parte de la librería para devolverle a la página web únicamente la cifra numérica del ruido generado por el avión.

#### Librería de ruido

La librería basada en el modelo ECAC para el cálculo de perfiles de ruido es el corazón del proyecto. Se trata de una tecnología desarrollada por un equipo de investigación del que es parte el tutor de este proyecto, Enrique Hernández Orallo. Esta librería se ha desarrollado en el contexto de un estudio centrado en la reducción de la contaminación y el ruido mediante maniobras de aproximación basadas en la reinyección de aeronaves [12]. Para ponerla en funcionamiento se le deben proporcionar una serie de datos de entrada, a partir de los cuales se calculará el perfil de ruido. Estos datos se indican a través de un conjunto de parámetros:

- **Datos del segmento (parámetro -S).** Se proporcionan las coordenadas de dos puntos de la trayectoria del avión y se especifica si el avión está rodando, su *back angle*, su potencia y la velocidad del segmento.
- **Posición del observador (parámetro -O).** Se proporcionan las coordenadas de la posición del observador.
- **Ruta del archivo .csv (parámetro -F)**. Se indica la ruta desde el directorio actual hacia el archivo .csv que contiene la información proporcionada por ANP Database.
- **Noise metric (parámetro -N).** Se escoge entre tres tipos de estimación de ruido: ENP, LAm y SEL [26].
- **Avión ascendiendo o descendiendo (parámetro -M).** Se especifica si el avión está descendiendo o ascendiendo.
- **Tipo de motor (parámetro -A).** Se indica si el avión tiene un motor de tipo *wing mounted* o *fuselage mounted* así como si es de tipo *turbo-fan* o *turbo-prop* [28].

La librería requiere de algunos otros parámetros para funcionar como las condiciones meteorológicas o la velocidad de avance de referencia, pero se proporcionarán los valores especificados por defecto. Toda la información sobre estos parámetros y los formatos que admiten se ha conocido a partir de la ayuda obtenida ejecutando el comando *ANCM_Lib_Win64 -H*.

## 4.3 Selección de las tecnologías

A continuación se enumerarán las tecnologías usadas. Esta tarea puede dividirse en dos bloques: por un lado, el conjunto de tecnologías y páginas web utilizadas para obtener y procesar la información relacionada con los aviones y por otro, el conjunto de tecnologías utilizadas para desarrollar el proyecto web.

### 4.3.1. Tecnologías para obtener información sobre los aviones

**ANCM_Lib.** Librería que recibe el conjunto de datos relacionados con un avión y su trayectoria y calcula el perfil de ruido asociado.

**Airlabs Data API. [21]** Se trata de la API utilizada para obtener en el proyecto web la información relacionada con los aviones en tiempo real. Esta página devuelve datos como la altitud, latitud, velocidad, ICAO, etc. de un gran conjunto de aviones, permitiendo establecer filtros relacionados con estos parámetros en sus consultas. En su plan gratuito permite 1000 consultas mensuales. En el apéndice *Elección de la API* se realiza un estudio de las diferentes APIs con funcionalidades similares a esta y se justifica esta elección de entre todas las disponibles.

**Flightradar24. [22]** Es una página web que muestra de forma gráfica el punto geográfico en el que se encuentran los aviones en tiempo real. Sus filtros por ICAO o aeropuerto de destino y origen la convierten en un buen complemento a la hora de realizar pruebas con Airlabs.

**Aircraft Noise and Performance (ANP) Database. [15]** Es un recurso online desarrollado por el Eurocontrol Experimental Centre (EEC) que proporciona documentos relacionados con el contorno de ruido de los aviones. Necesaria para el buen funcionamiento de la librería.

### 4.3.2. Tecnologías para desarrollar el proyecto web

**HTML, CSS y Bootstrap.** HTML y CSS han sido utilizados para la construcción de la estructura y elaboración del diseño gráfico de la página web. Este proceso ha sido complementado mediante el uso de Bootstrap.

**Perl.** Es un lenguaje utilizado para un amplio abanico de propósitos que en este caso ha sido usado para desarrollar el funcionamiento del servidor CGI que ejecute la librería.

**Node.js y AJAX.** Node.js es un entorno de programación basado en Javascript. AJAX es una técnica de desarrollo consistente en el uso de peticiones para comunicarse con servidores. Se han utilizado para desarrollar la funcionalidad del frontend que permite realizar peticiones a la API, procesar la información recibida, enviarla al servidor CGI y procesar también la respuesta de este.

**Google Maps Platform. [20]** Es una API desarrollada por Google que permite incorporar Google Maps en aplicaciones web y móviles. Se ha utilizado para proporcionar al usuario una herramienta gráfica con la que le resulte sencillo seleccionar el punto geográfico que quiera proporcionar como posición del observador. Además

esta API también permite la posibilidad de que el usuario indique como posición su ubicación actual.

**Apache.** Se trata de un software servidor que procesa peticiones HTTP y envía como respuesta información en forma de página web. Se utilizará para crear y poner en funcionamiento el servidor CGI y para desplegar la página web.

**XAMPP.** Se trata de un panel que permite construir y poner en marcha un servidor Apache de forma sencilla.

A modo de resumen de las tecnologías de la aplicación web y el componente en el que se ubica cada una se proporciona el esquema de la figura 4.2.

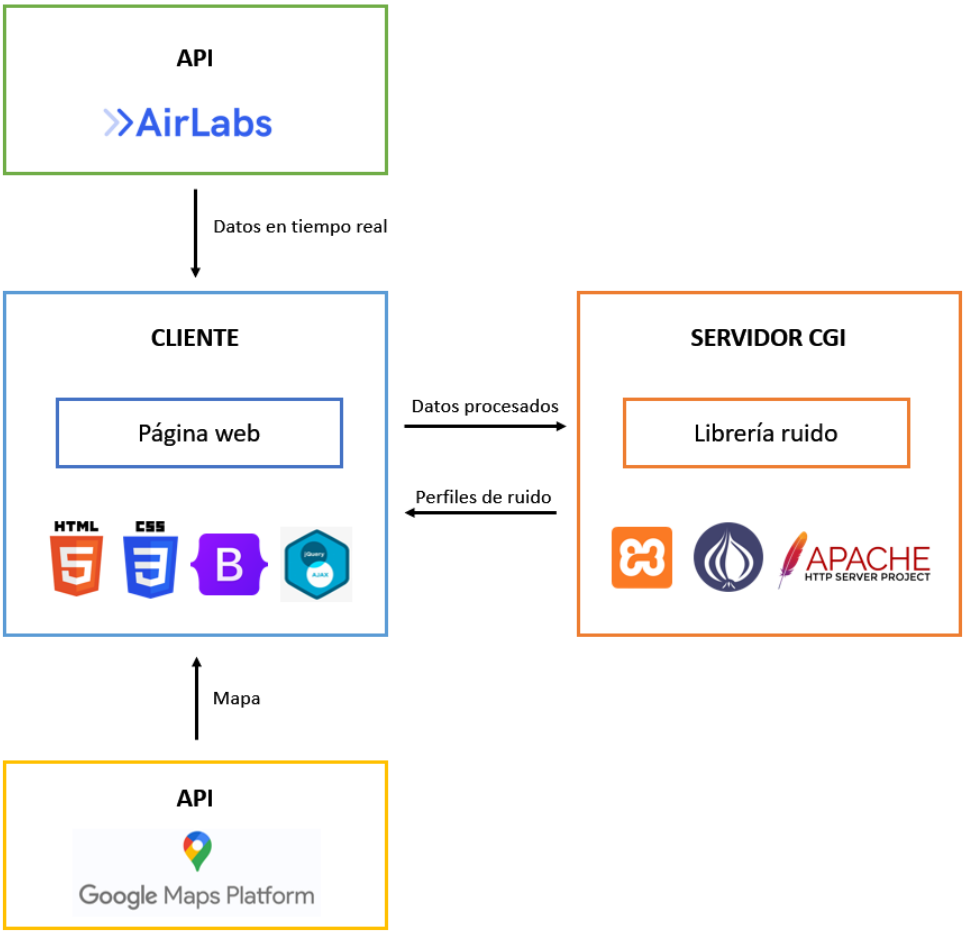


**Figura 4.2:** Esquema de tecnologías de la aplicación web.

# CAPÍTULO 5
# Desarrollo de la solución

En este capítulo se detallarán los aspectos técnicos relacionados con la implementación de los distintos bloques identificados en el apartado anterior. Este análisis se encuentra dividido en dos grandes secciones: *frontend* y *backend*.

## 5.1 Implementación del *frontend*

### 5.1.1. HTML, CSS y Bootstrap

#### Estructura de la página

La página presenta una paleta de colores azul, combinando tonos oscuros y claros que permiten diferenciar sus diferentes partes. La estructura se compone de tres elementos principales: el encabezado (*header*), el contenido principal (*main*) y el pie de página (*footer*). El encabezado incluye el título de la página y una breve descripción que informa sobre su funcionamiento, mientras que el pie de página tiene una función principalmente estética y muestra el logo de la universidad.

En el contenido principal se encuentran concentrados el resto de los elementos. Por un lado, se encuentra el script necesario para insertar el mapa de Google Maps, el cual deberá incluir una URL en la que uno de los parámetros sea la *key* asociada a la cuenta de Google Maps Platform utilizada. Junto a este mapa se presenta un breve texto que describe la delimitación de un área de veinte kilómetros para las posiciones, y se informa al usuario que al hacer clic en el mapa se actualizará la posición del observador. Parte de este texto es un enlace sobre el cual se puede pulsar para utilizar la posición actual. Otra parte de este texto es dinámica y muestra la última latitud y longitud seleccionadas, actualizándose automáticamente cada vez que estos valores cambien. Finalmente, también se encuentra el botón 'Calcular ruido', sobre el cual el usuario puede pulsar para iniciar el proceso que ocurrirá internamente. Esta sección principal (*main*) también contiene un par de divisiones (*divs*) vacías diseñadas para alojar el contenido devuelto por el servidor CGI una vez que los datos hayan sido procesados. En la figura 5.1 puede verse el aspecto final de esta página.

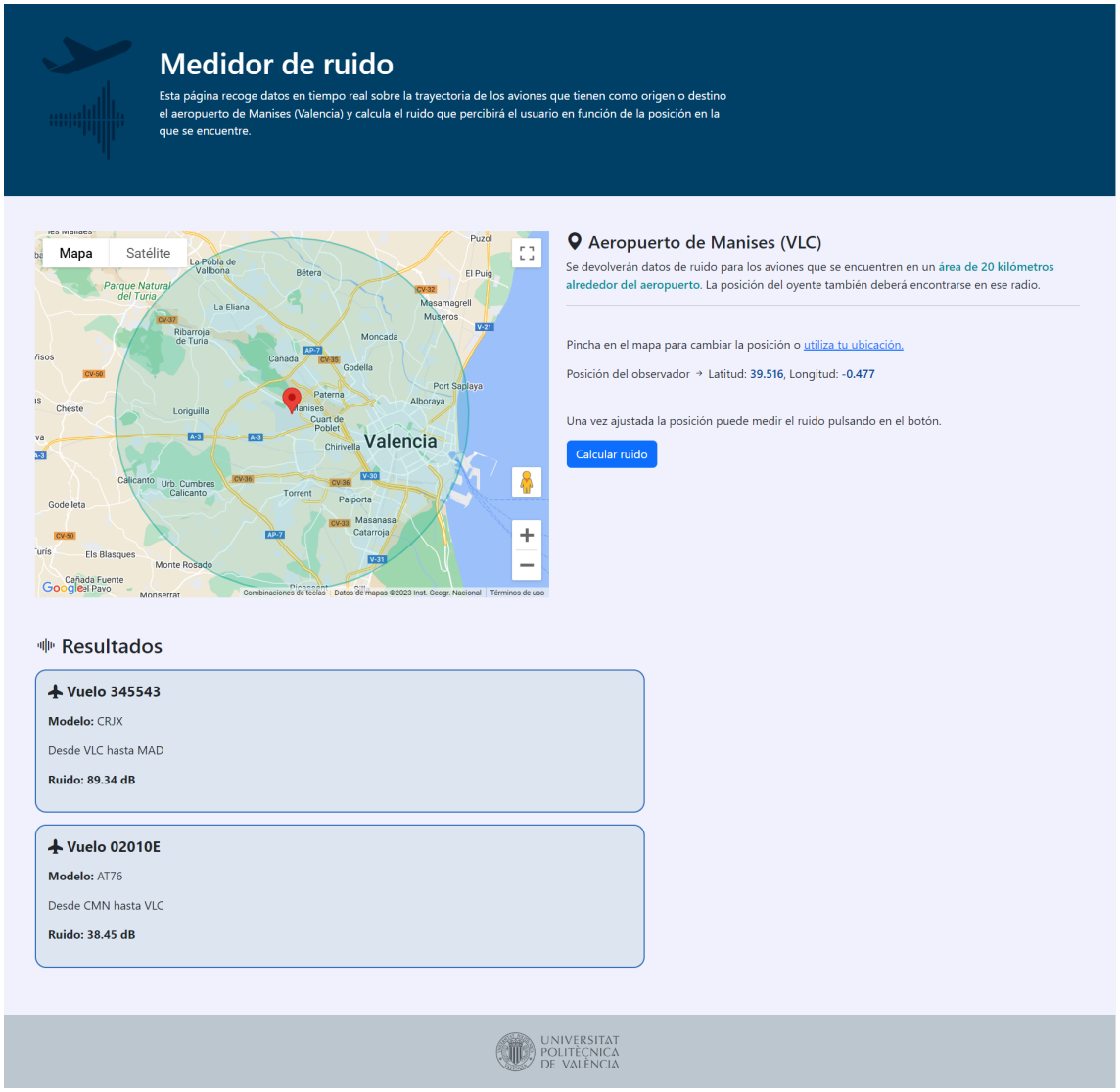


**Figura 5.1:** Aspecto final de la página web.

### Diseño responsive

Uno de los aspectos más importantes al diseñar una página web es proporcionar a los usuarios una experiencia de uso agradable, presentándoles una página que pueda adaptarse de forma dinámica a todo tipo de dispositivos y tamaños de pantalla (figura 5.2). Es por ello que se ha optado por un diseño responsive, un requisito prácticamente imprescindible [23]. Para cumplir con este objetivo se han utilizado propiedades para contenedores proporcionadas por Bootstrap, así como consultas de medios (media queries) en CSS, que permiten adaptar el tamaño y la alineación de los elementos según el tamaño del navegador o la pantalla en la que se estén visualizando.

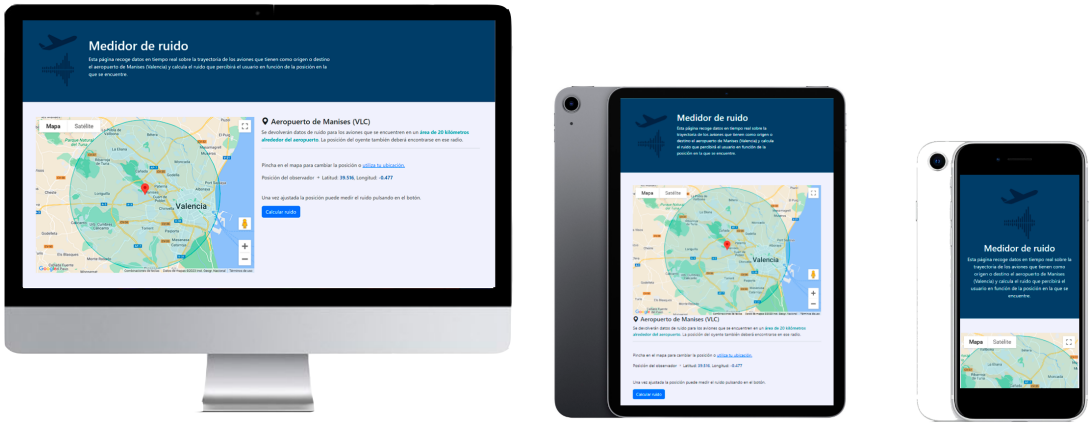


**Figura 5.2:** Adpatación dinámica a distintos tipos de pantalla

### 5.1.2. Javascript y AJAX

El código Javascript está dividido en dos grandes secciones. Por un lado se encuentran los métodos desarrollados para implementar el mapa de Google Maps insertado en la página web y definir el área alrededor del aeropuerto. Por otro, el código utilizado para gestionar el proceso de recogida y procesamiento de datos. Comenzaremos describiendo el primer bloque, explicando el funcionamiento del código desarrollado y de los métodos implementados.

#### Creación del mapa

El método *initMap()* contiene el código esencial para hacer que el mapa aparezca en la página web, ya que al llamar a este método se establece la conexión con la API Google Maps Platform y se especifican los aspectos visuales a tener en cuenta. Lo primero que se debe hacer es definir una variable que sirva para referenciar la división (*div*) de la página en la que va a alojarse el mapa, indicando el nivel de zoom y el centro de coordenadas. En este caso se ha escogido un nivel 11 de zoom para ofrecer una vista lo suficientemente alejada como para abarcar la ciudad de Valencia y los pueblos a su alrededor. Como centro se ha especificado el punto de latitud 39.489 y longitud -0.478, siendo estas las coordenadas del aeropuerto de Manises [24].

Posteriormente se ha añadido el código necesario para dibujar el círculo que delimite el área en el que se permitirán mediciones de ruido. Utilizando también Google Maps Platform se ha definido un círculo cuyo centro es el aeropuerto de Manises y su radio es de unos veinte kilómetros aproximadamente.

Por último, se ha asociado un listener tanto con el mapa como con el círculo que llame al método *dentroCirculo()* cada vez que el usuario haga clic sobre el mapa para seleccionar una posición. Este método es el encargado de actualizar tanto el texto de la página web que informa sobre la última latitud y longitud indicadas como las variables globales *lat_usu* y *lng_usu*, que contienen estos valores.

**Detectar posiciones dentro del área de 20 kilómetros**

Para calcular si una posición (latitud, longitud) está dentro o fuera de un círculo de radio determinado se puede utilizar la fórmula de la distancia entre dos puntos en la superficie de una esfera, conocida como fórmula de Haversine o fórmula del semiverseno [25]. Esta fórmula utiliza las coordenadas geográficas de ambos puntos (en este caso uno de los puntos será el aeropuerto de Manises y el otro la posición del observador) para calcular la distancia entre ellos, teniendo en cuenta que la Tierra es una esfera.

$$a = \sin^2\left(\frac{\Delta lat}{2}\right) + \cos(lat1) \cdot cos(lat2) \cdot \sin^2\left(\frac{\Delta long}{2}\right) \tag{5.1}$$

$$c = 2 \cdot atan2(\sqrt{a}, \sqrt{1-a}) \tag{5.2}$$

$$d = R \cdot c \tag{5.3}$$

Donde:

- lat1, long1 son las coordenadas del centro del círculo.
- lat2, long2 son las coordenadas de la posición que se quiere verificar.
- $\Delta$ lat = lat2 - lat1
- $\Delta$ long = long2 - long1
- R es el radio de la Tierra (en metros).

Si la distancia *d* calculada con la fórmula es menor que el radio del círculo, entonces la posición estará dentro del círculo. Si es mayor, se encuentra fuera. Hay que resaltar que esta fórmula asume que la Tierra es una esfera perfecta y no tiene en cuenta la forma real del globo terrestre, lo que implica que puede existir una ligera inexactitud en las mediciones, especialmente en distancias largas o cercanas a los polos. Sin embargo, resulta apropiada para el caso de estudio que se tiene entre manos.

Esta fórmula se implementa en el método *distanciaSemiverseno()*, el cual recibe la latitud y longitud de ambos puntos y devuelve la distancia que existe entre ellos. Este método es llamado a su vez desde el método *dentroCirculo2()*, al que se le proporciona la posición del observador expresada en latitud y longitud y devuelve un valor booleano en función de si el punto se encuentra dentro del círculo o no.

**Gestión de la posición del observador**

Como ya se ha mencionado anteriormente, resulta posible actualizar la posición del observador y por ende las variables *lat_usu* y *lng_usu* pulsando sobre el mapa, pero también se puede hacer utilizando la ubicación actual del usuario.

Para ello se debe pulsar sobre el texto con apariencia de enlace en el que así se indica. Esta acción será detectada por un código AJAX que, haciendo uso de la API

Google Maps Platform, calculará la latitud y longitud actuales, las cuales serán utilizadas para actualizar el contenido de *lat_usu* y *long_usu*. Además, al hacerlo aparecerá un pequeño marcador sobre el mapa con el texto 'Estás aquí', indicando al usuario la posición en la que se encuentra.

Con esto finaliza la implementación del código relacionada con la gestión del mapa. A continuación se detallará el contenido de los métodos y la petición AJAX que se han desarrollado para gestionar la comunicación con la API Airlabs y con el servidor CGI. El número de métodos utilizados en esta parte del código es mayor y su funcionamiento algo más complejo, por lo que únicamente se mencionarán en los siguientes párrafos, dejando su explicación para apartados posteriores.

Esta parte del código contiene dos variables *api_key* y *api_base* que son la base de la comunicación con la API Airlabs. Para llevar a cabo esta comunicación se ha implementado un manejador de eventos AJAX que detecta la pulsación del botón 'Calcular ruido' por parte del usuario. Cuando suceda, primero se verificará que la posición del oyente escogida es válida haciendo uso del método *dentroCirculo2()*. En caso negativo, aparecerá una alerta indicando al usuario que la posición debe encontrarse dentro del área y será necesario introducir una nueva posición. En caso afirmativo, se borrará el contenido del *div* que almacena las respuestas de una anterior consulta (si la ha habido) y se iniciará la petición AJAX a la url *api_base* utilizando como dato la *api_key* que identifica al usuario.

Dado que no se le especifica ningún filtro adicional a la petición, Airlabs devolverá un vector formado por todos los aviones que se encuentren en pleno vuelo en ese momento. Es en el código donde se realizará un filtro que descarte todos los aviones excepto aquellos cuya propiedad *dep_iata* o *arr_iata* sea igual a *VLC* (es decir, se descartarán todos los aviones excepto aquellos que tengan como origen o como destino el aeropuerto de Manises). La lista de aviones restantes tras aplicar este filtro se almacenará en la variable *avionesFiltrados*, cuyos elementos serán recorridos uno a uno haciendo uso de un bucle for. Para cada iteración de este bucle se almacenará el avión en cuestión en la variable *avion*, y será a partir de este punto donde comenzará el procesamiento de datos detallado en el capítulo *Diseño de la solución*. Primero se explicará la utilidad de las variables *avionesDisponibles* y *avionEnArea* y después se detallará el contenido y función del resto de variables, que además coinciden en nombre con los parámetros que la librería requiere para funcionar correctamente.

#### Variables *avionesDisponibles* y *avionEnArea*

La variable *avionEnArea* hace uso una vez más del método *dentroCirculo2()* para almacenar un valor booleano que indique si el avión que se está analizando se encuentra dentro del área. Esta variable se utilizará más adelante como parámetro para diferenciar entre los aviones que se deben descartar y aquellos de los que se debe calcular el ruido que generan.

Por otro lado, la variable *avionesDisponibles* almacena el número total de aviones que se dirigen o proceden del aeropuerto de Manises (calculando la longitud del vector *avionesFiltrados*). No todos los aviones que cumplan esta condición serán válidos, ya que, como se acaba de indicar, pueden encontrarse fuera del área a analizar o puede no disponerse de su modelo ANP. Cada vez que un avión incumpla alguna

de estas dos condiciones, se le restará una unidad al valor de *avionesDisponibles*. Si una vez procesados todos los aviones el valor de esta variables es 0, se mostrará un mensaje en la página web indicando que no hay aviones cerca del aeropuerto en ese momento o que quizás haya alguno cuyo modelo ANP no se tenga registrado.

#### Variables *posicion1*, *posicion2* y *rodando*

La librería requiere de dos puntos de la trayectoria del avión que se corresponden con las variables *posicion1* y *posicion2*. Para obtener estas posiciones primero se consiguen las propiedades *lat*, *lng* y *alt* que representan respectivamente la latitud, la longitud y la altitud del avión y después se le pasan al método *convertirCoordenadas()*. Un proceso similar se sigue para la variable *rodando*, que almacena el resultado de pasarle al método *estaRodando()* la altitud.

#### Variables *backAngle* y *power*

El valor del ángulo de incidencia del avión y la energía de los motores vienen representados respectivamente por las variables *backAngle* y *power*. Se ha optado por darle a la variable *backAngle* el valor de 0 y a la variable *power* el valor de 7500, siendo ambos valores promedio. Esto se debe a la dificultad para obtener estos valores en tiempo real, aunque cabe destacar que el valor de *power* tiene un impacto importante sobre el ruido, sobre todo en el despegue.

#### Variables *velocidad*, *ascendiendo* y *noiseMetric*

La propiedad *speed* del avión representa su velocidad en kilómetros/hora, pero la librería necesita que esté expresada en pies/segundo, por lo que la primera acción a realizar será convertirla a metros por segundo y después pasársela al método *DeMetrosAPies()*. Por otro lado, la variable *ascendiendo* recogerá el resultado de ejecutar el método *estaAscendiendo()*, al cual será necesario proporcionarle el IATA del aeropuerto de origen.

Finalmente, la variable *noiseMetric* se definirá con el valor *LAm*, que indica que lo que se medirá será el máximo nivel de ruido alcanzado durante un periodo de medición, siendo el resultado expresado en decibelios. La librería soporta otros tipos de medición, como *SEL* o *PNL*, pero *LAm* es el más indicado para medir el ruido generado en un único evento, mientras que las otras medidas están más enfocadas a la comparación del ruido producido por varios aviones durante un mismo intervalo de tiempo [26].

#### Variable *observador*

La variable *observador* representa el punto en el que se encuentra el usuario que ejecuta la aplicación y ha de estar expresada en el mismo formato y unidades que *posicion1* y *posicion2*.

**Variable *caracteristiscas***

La variable *caracteristicas* es un array que recibe el resultado de pasarle a la función *tipoDeAvion()* el ICAO del avión. Dicho array está formado por tres componentes cuyo valor se asociará a las variables *mounted*, *propelled* y *archivo_csv*. Las dos primeras variables hacen referencia al tipo de motor y se les asociará un valor entre 0 y 1.

- En el caso de la variable *mounted*, indica si el motor del avión es de tipo *wing mounted* (valor 1) o *fuselage mounted* (valor 0).
- En el caso de la variable *propelled*, indica si el motor es de tipo *turbofan* (valor 1) o *turboprop* (valor 0).

En el caso de *archivo_csv*, el valor almacenado será el nombre del fichero .csv que se le debe pasar a la librería, dependiendo del modelo del avión.

La lista de variables y métodos y la forma en la que se comunican entre ellos aparece reflejada en el diagrama de la figura 5.3.

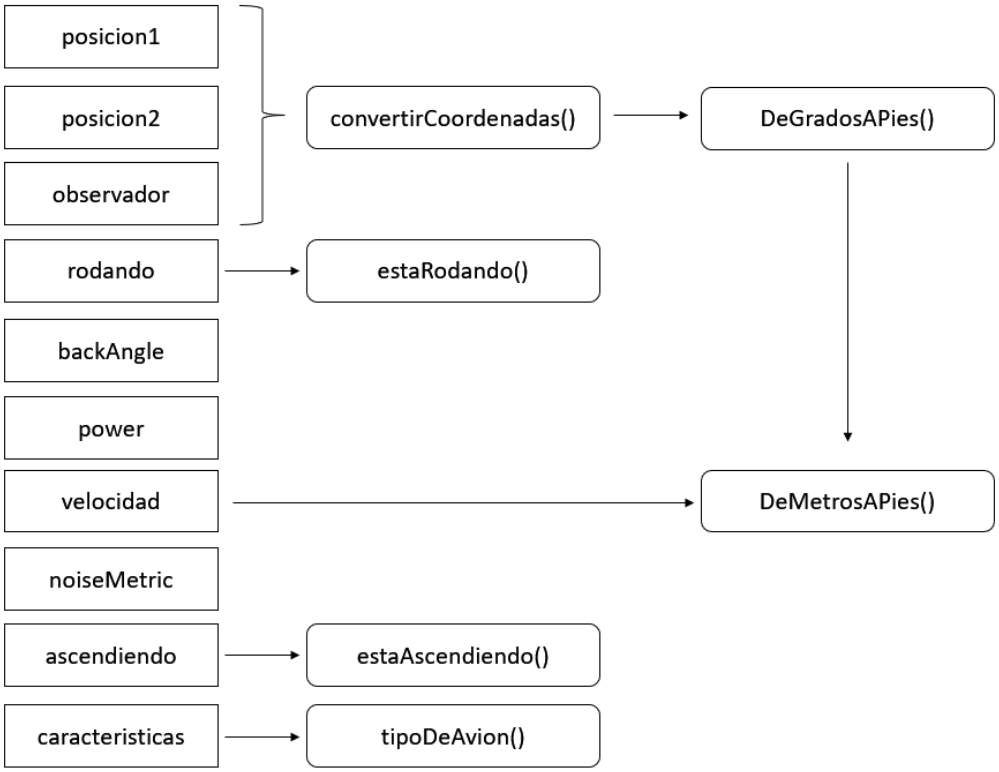


**Figura 5.3:** Diagrama de variables y métodos

Una vez finalizado el procesamiento de datos se construye una nueva petición AJAX que enviará todas las variables antes mencionadas al servidor CGI. Esta petición también gestiona en su apartado *success* la respuesta por parte del servidor: mediante el método *append()* construye una nueva tabla a partir del elemento *#respuesta* de la página HTML en la que incluirá el ruido generado y otros datos como el identificador del vuelo, el ICAO del modelo del avión y el IATA de los aeropuertos

de origen y de destino. Cabe destacar que esta petición solo se ejecuta si el valor de la variable *caracteristicas* no es nulo y la variable *avionEnArea* almacene el valor *true*, ya que de lo contrario es posible que no se disponga del modelo ANP del avión o que este se encuentre demasiado lejos del aeropuerto.

#### Listado de métodos

A continuación se ofrece un listado de los métodos mencionados junto con la explicación de su funcionamiento y su implementación.

- ***convertirCoordenadas().*** Esta función recibe la latitud y longitud de un avión en grados decimales y la altitud en metros y transforma todas a pies. Después calcula la posición relativa considerando el aeropuerto de Manises como el centro de coordenadas. Devuelve un array *[latitud, longitud, altitud]* con las variables ya procesadas.

  El código contiene una constante *coordenadasAeropuerto* que es un array que contiene la latitud, longitud y altitud del aeropuerto de Manises ya expresadas en pies. También se declaran tres variables, *latitud_ft*, *longitud_ft* y *altitud_ft*, almacenándose en cada una de ellas el valor de esos parámetros convertido en pies. Para realizar la conversión en el caso de la latitud y la longitud se hace uso de la función *DeGradosAPies()* y en el caso de la altitud de *DeMetrosAPies()*. Finalmente, se obtienen los valores relativos realizando para cada uno de ellos la operación *parámetro del avión - parámetro del aeropuerto* (por ejemplo, *latitud relativa = latitud del avión - latitud del aeropuerto*) y se almacenan en la variable *stringCoordenadas* expresados en el formato *latitud,longitud,altitud*, que es el aceptado por la librería.

- ***DeGradosAPies().*** Función que recibe como parámetro un valor en grados decimales y lo devuelve convertido a pies. Para ello, multiplica el valor en grados decimales por 111139 para transformarlo en metros [27] y se lo pasa a la función *DeMetrosAPies()*.

- ***DeMetrosAPies().*** Función que recibe como parámetro un valor en metros y lo devuelve convertido a pies. Para ello primero multiplica el valor en metros por 3,28084 para transformarlo en pies.

- ***estaRodando().*** Función que recibe la altitud en metros a la que se encuentra el avión y devuelve un valor numérico entre 0 y 1 que determina si el avión está rodando (1) o no (0). Para ello, realiza una comparación entre la altitud recibida y la altitud del aeropuerto y si la primera es menor concluye que el avión está rodando.

- ***estaAscendiendo().*** Función que recibe el IATA del aeropuerto de origen del avión y devuelve un carácter 'A' o 'D' en función de si el avión está ascendiendo (A) o descendiendo (D). Para ello se considera que si el IATA recibido es igual a *VLC* (el aeropuerto de origen es el de Manises) el avión estará ascendiendo. En caso contrario se encontrará descendiendo.

- ***tipoDeAvion().*** Función que recibe el ICAO del avión y devuelve un array *[mounted, propelled, archivo_csv]* indicando el tipo de motor y el archivo CSV a

utilizar dependiendo del modelo. Para ello, se hace uso de un switch que utilizando el ICAO recibido como condición asocia a las variables *mounted*, *propelled* y *archivo_csv* el valor correspondiente. En caso de que el ICAO proporcionado no coincida con ninguno de los contemplados en el switch se devuelve un valor nulo.

## 5.2 Implementación del *backend*

### 5.2.1. XAMPP y servidor Apache

Tras instalar XAMPP se dispondrá de un panel que permite gestionar distintos tipos de servicios (Apache, MySQL o Tomcat, etc.). Con tan solo pulsar el botón *Start* asociado al servicio deseado (en este caso el servidor Apache) este entrará en funcionamiento.

### 5.2.2. Servidor CGI y Perl

La instalación de XAMPP genera una estructura de carpetas en el equipo a la que es necesario recurrir para poder programar correctamente el servidor CGI. Concretamente nos interesará la carpeta *cgi-bin*, ya que todo código introducido dentro de ella será ejecutable accediendo a la url correcta. En este caso, queremos desarrollar un script en Perl llamado *libreria.cgi* y ubicado en la subcarpeta *noise*. La url para acceder a este script será *http://localhost/cgi-bin/noise/libreria.cgi*, y será la dirección que se le deberá proporcionar a la petición AJAX enviada desde el cliente.

En el código Perl se han declarado el mismo número de variables y con el mismo nombre que las que se han creado en el código Javascript. Están pensadas para almacenar el valor de estas últimas cuando sean recibidas a través de la petición AJAX. También se encuentra declarada la variable *cmd*, cuyo valor es un string que representa la orden que se debe lanzar para que la librería de ruido se ejecute con los valores recibidos. Esta orden se concatena a su vez con un *findstr NOISE* para que, de la salida proporcionada por la librería, únicamente nos quedemos con la línea en la que aparezca la cifra numérica de ruido, la cual será almacenada en la variable *output*. Finalmente, el valor de esta variable es devuelto al cliente en formato de texto plano.

# CAPÍTULO 6
# Pruebas y despliegue

A lo largo de este capítulo se profundizará en las pruebas llevadas a cabo para comprobar el correcto funcionamiento del código desarrollado y se describirá el proceso de despliegue del proyecto.

## 6.1 Pruebas

**Tiempo de actualización de la API**

La dinámica inicial para proporcionar a la librería las variables *posicion1* y *posicion2* era ejecutar una llamada distinta a la API para obtener el valor de cada una de estas variables. Sin embargo, esta solución implica que el tiempo de respuesta al usuario quede limitado por el tiempo mínimo de actualización en los datos de la API, ya que realizar dos peticiones demasiado seguidas tendría como resultado que el valor de ambas posiciones fuese el mismo. Para medir el tiempo que la API tarda en actualizar sus datos se han realizado varias llamadas dejando distintos intervalos de tiempo cada vez más grandes hasta que los datos devueltos entre dos ejecuciones distintas reflejaban cambios. Tras varias pruebas se ha determinado que la API actualiza sus datos cada trece segundos.

**Precisión de los datos de la API**

Con esta prueba se desea comprobar que los datos devueltos por la API son precisos y reflejan la situación en tiempo real de los aviones. La forma de llevarlo a cabo ha sido contrastando datos con otra página que además proporcione apoyo visual: *Flightradar24*.

Lo primero a señalar es que para esta prueba la información volcada en la página web también ha incluido la propiedad *hex* del avión, que es su ICAO 24, lo que permite identificarlo rápidamente de manera unívoca. Posteriormente ha entrado en juego el uso de *Flightradar24*. Esta página ofrece un mapa que muestra la ubicación e información de los aviones que se encuentran en vuelo durante el momento de consulta y además permite establecer varios filtros para mostrar solo un determinado conjunto de aviones. Lo que se ha hecho ha sido establecer un filtro por aeropuerto de origen o destino especificando el IATA *VLC*, de forma que en el mapa solo se muestren los aviones que se dirigen o proceden del aeropuerto de Manises. Pos-

teriormente, se ha seleccionado uno de estos aviones y se ha anotado su ICAO 24. Después se ha ejecutado la API *Airlabs* a través del código Javascript desarrollado en el cliente para comprobar que la información recogida por la misma coincide con la proporcionada por *Flightradar24*. El resultado es satisfactorio, ya que ambos servicios devuelven datos muy similares y actualizados durante todo el trayecto.

#### Comunicación entre el cliente y la API y procesado de datos correcto

Ambas comprobaciones se realizan en el código Javascript utilizando logs cuyo resultado puede verse accediendo a la consola del navegador, en el apartado de herramientas de desarrollador. En el caso de la API, se utiliza un único log que imprime el contenido del vector *avionesFiltrados*, mostrando así el resultado de la consulta original (ya filtrada utilizando el IATA del aeropuerto de Manises). Para el resto de variables se muestran logs (figura 6.1) después de las ejecuciones de los métodos que procesan las propiedades del avión. Con comprobar el valor mostrado por estos logs es suficiente para saber si los métodos están funcionando correctamente, ya que es sencillo inferir los valores que deben devolver a partir de los proporcionados por la API.

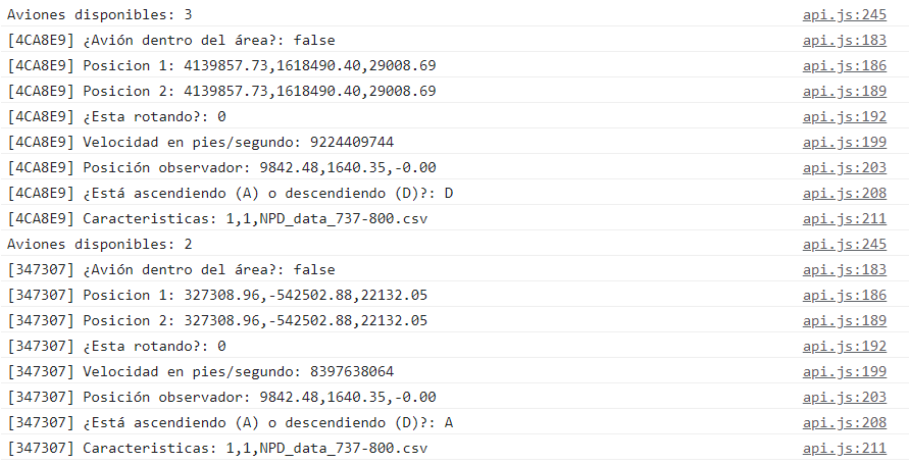


**Figura 6.1:** Los logs devuelven información sobre el procesamiento de datos

#### Mismo valor para *posicion1* y *posicion2*

El funcionamiento inicial del procesado de datos contemplaba el enviar dos peticiones AJAX distintas a la API, una para obtener la *posicion1* y otra para la *posicion2*. Sin embargo, la diferencia de trece segundos entre actualizaciones observada en pruebas anteriores y la limitación de mil consultas mensuales ponen en duda si esta se trata de la mejor alternativa.

Esta prueba pretende analizar la diferencia obtenida en el cálculo de ruido si en lugar de proporcionar dos posiciones distintas las variables *posicion1* y *posicion2* contienen el mismo valor. Para estudiarlo se han realizado dos llamadas a la librería, una para cada caso. El ruido obtenido en ambos casos ha sido exactamente el mismo, por lo que se ha optado por asociar el mismo valor a *posicion1* y *posicion2* y proporcionar de esta forma una respuesta más rápida al usuario, economizando también en el número de consultas. Hay que resaltar que esta alternativa solo es posible cuan-

do se utiliza *Lmax* como métrica de ruido, ya que el uso de SEL haría necesario el proporcionar dos puntos distintos que formen parte de la trayectoria del avión.

#### Comunicación entre el código cliente y la librería

Para comprobar que los datos enviados desde el cliente hacia el servidor (y viceversa) se están procesando de forma adecuada primero se ha ejecutado la librería por separado (sin hacer uso del proyecto), proporcionando como entrada un conjunto de datos concretos. Una vez ejecutada se ha anotado el ruido obtenido. Posteriormente, se ha creado una versión del proyecto que no obtenga los datos a partir de la API sino que ya tenga los valores necesarios asociados a las variables en cuestión, siendo estos valores los mismos que se han utilizado para ejecutar la librería por separado. Al ejecutar esta versión del proyecto se ha obtenido el mismo ruido que el anotado anteriormente, con lo que queda comprobado que la comunicación entre el cliente y el servidor funciona tal y como se espera.

## 6.2 Despliegue

Poner en funcionamiento la aplicación web es sencillo gracias al panel XAMPP. Tras instalarlo, lo único que se deberá hacer es comprobar que se respeta la estructura de directorios y ficheros pensada para el proyecto:

- El código que gestiona el servidor CGI debe encontrarse en la ruta */xampp/cgi-bin/noise/libreria.cgi*.
- El proyecto web debe encontrarse en la carpeta *proyecto-ruido* en la ruta */xampp/htdocs*.

Después basta con pulsar el botón *start* de Apache en el panel XAMPP e ingresar en el navegador la url *http://localhost/proyecto-ruido/home.html*. Para obtener los perfiles de ruido para los aviones habrá que pulsar en el botón *Calcular ruido*. En la figura 6.2 se puede apreciar el aspecto de la sección de resultados cuando hay aviones dentro del área de análisis. En este caso se han detectado dos vuelos:

- Un vuelo con el identificador 345543 que genera un ruido de 89.34 decibelios. Se trata de un Bombardier CRJ700 (modelo CRJX) que viaja desde Valencia (VLC) hacia Madrid (MAD).
- Un vuelo con el identificador 02010E que genera un ruido de 38.45 decibelios. Se trata de un ATR 72 (modelo AT76) que viaja desde el Aeropuerto Internacional Mohammed V ubicado en Marruecos (CMN) hacia Valencia (VLC).

En ambos casos se observa un ruido elevado, algo bastante coherente teniendo en cuenta que la posición del observador tiene una latitud 39.516 y una longitud -0.477 (figura 5.1), es decir, se encuentra realmente cerca del aeropuerto. En el caso del primer vuelo, los más probable es que se encuentre en pleno despegue.

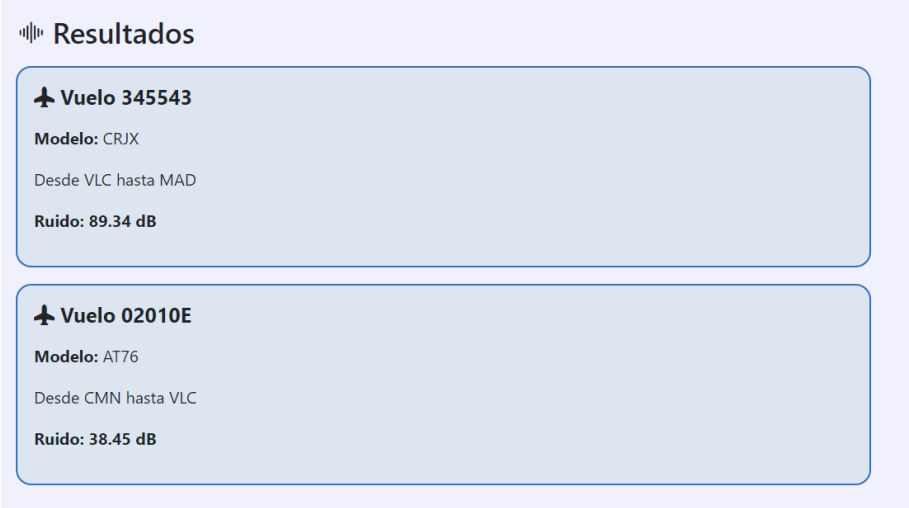


**Figura 6.2:** Respuesta cuando hay aviones dentro del área

En caso de que cuando el usuario solicite conocer el ruido generado por los aviones en su zona pero no haya ningún avión cercano, se le informará de ello, tal y como se puede ver en la figura 6.3.

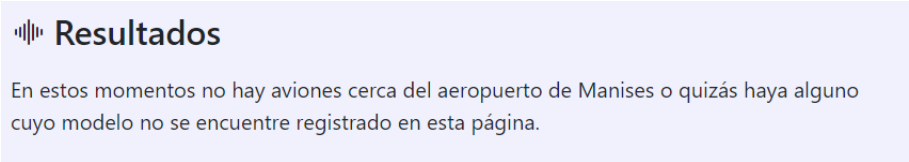


**Figura 6.3:** Respuesta cuando no hay aviones dentro del área

# CAPÍTULO 7
# Conclusiones

En este trabajo se ha desarrollado una aplicación web que permite la estimación del ruido generado por los aviones en las proximidades del aeropuerto de Manises, teniendo en consideración la ubicación del oyente.

El proceso ha iniciado con un estudio de soluciones similares existentes, donde se han analizado las diferencias y similitudes con el proyecto en desarrollo. Como resultado, se ha concluido que esta aplicación brinda una solución accesible para usuarios sin conocimientos técnicos, además de ser interesante desde el punto de vista de la innovación y la recolección de información sobre la generación de ruido por diferentes tipos de modelos y motores de avión.

Después de seleccionar las tecnologías adecuadas y diseñar la estructura del proyecto, se ha procedido con su desarrollo, cumpliendo así con los objetivos secundarios propuestos. En primer lugar, se han establecido los mecanismos necesarios para obtener información en tiempo real sobre los aviones. Esta etapa ha presentado varios desafíos, especialmente en la búsqueda de una API que ofreciera un equilibrio satisfactorio entre precio y calidad de los datos, lo que incluso ha requerido cambiar de API durante el desarrollo del proyecto (este proceso se detalla en el apéndice *Elección de la API*). La implementación del código para procesar los datos obtenidos ha resultado ser una tarea relativamente sencilla desde el punto de vista de la implementación, pero se ha extendido en el tiempo considerablemente debido a la gran cantidad de pruebas necesarias para garantizar su correcto funcionamiento, convirtiéndose así en una de las fases más largas del proyecto. Por otro lado, la implementación del servidor CGI ha sido una de las partes que ha demandado más investigación, ya que se han utilizado tecnologías y mecanismos con los que no estaba familiarizado, como Perl o XAMPP. Por último, la construcción del sitio web ha sido una de las partes más manejables en términos de diseño de la aplicación, pero también ha planteado otros desafíos, como la integración de la API de Google Maps o el uso de la fórmula de Haversine, que han requerido una investigación adicional debido a su carácter desconocido.

Además de un mayor dominio de tecnologías ya conocidas o el aprendizaje de otras nuevas, también se han adquirido nociones sobre la industria aeroespacial, ya que una gran parte del trabajo ha requerido el trabajar con terminología del sector y con plataformas completamente relacionadas con este mundo como Airlabs, Flightradar o ANP Database.

A modo de resumen puede concluirse que se ha logrado con éxito el objetivo principal del trabajo, así como los objetivos secundarios, a pesar de los problemas e imprevistos surgidos a lo largo del proceso, que han hecho que cada etapa de desarrollo presentara elementos desafiantes. Durante este proceso se han afianzado conocimientos y se han adquirido otros nuevos, relacionados tanto con la informática como con el campo aeroespacial.

## 7.1 Relación del trabajo desarrollado con los estudios cursados

El trabajo desarrollado presenta una relación estrecha con los conocimientos adquiridos en la carrera, y especialmente en la rama cursada, Tecnologías de la Información. Por una parte, el concepto de integración queda presente al ser necesario establecer comunicaciones e intercambiar información entre diferentes servicios como la API o el servidor CGI. Por otra, el desarrollo de una aplicación web pone sobre la mesa los conocimientos que se tienen sobre diseño y funcionalidad de páginas web. Es aquí donde se hace uso de tecnologías como HTML, CSS, Javascript o AJAX, presentadas a lo largo de la carrera y en las que se ha profundizado en los dos últimos años de educación, siendo también necesarios los conocimientos sobre programación para desarrollar con éxito la parte de procesamiento de datos. También cabe destacar la preocupación por otro de los conceptos estudiados: realizar un diseño centrado en el usuario. Se ha diseñado con la voluntad de brindar al usuario una experiencia en la que la facilidad de uso, la comprensión y la adaptabilidad de la página a distintos dispositivos (diseño responsive) sean conceptos presentes en todo momento. Durante el grado también se han afianzado una serie de competencias transversales de las cuales en este proyecto se han puesto en práctica las siguientes:

- CT01. Comprensión e integración.
- CT02. Aplicación y pensamiento práctico.
- CT03. Análisis y resolución de problemas.
- CT05. Diseño y proyecto.
- CT07. Responsabilidad ética, medioambiental y profesional.
- CT12. Planificación y gestión del tiempo.

## 7.2 Trabajos futuros

El proyecto desarrollado no solo cumple con los objetivos propuestos inicialmente sino que también abre la posibilidad a la construcción de versiones aún más completas que proporcionen un mayor número de funcionalidades al usuario. Algunas de las ideas más interesantes a desarrollar en este aspecto son las expuestas a continuación:

- **Selección de aeropuerto.** Resultaría interesante expandir la aplicación para posibilitar su uso en un mayor número de aeropuertos. Para ello, sería necesario

conocer las coordenadas de estos aeropuertos y obtener la tabla NPD de los modelos de avión más frecuentes.

- **Mostrar las trayectorias de los aviones en el mapa.** Durante el desarrollo de esta aplicación se ha utilizado Flightradar24 a modo de apoyo gráfico para conocer la posición en el mapa de los aviones y la trayectoria que recorren. Implementar una función similar en este proyecto web haría que este apoyo deje de ser necesario en tareas de análisis, proporcionando además información más visual al usuario.
- **Construcción de gráficas y tablas.** Actualmente la información sobre el ruido se proporciona al ser solicitada, pero no se almacena. El hacerlo podría abrir la puerta a la construcción de gráficas o tablas que permitan realizar un análisis de distintos aspectos interesantes, como por ejemplo los momentos de mayor nivel de ruido a lo largo del día.

# Bibliografía


[1] Wanj, Scarlett Sijia y Glied, Sherry y Williams, Sharifa y Will, Brian y Muenning, Peter Alexander. Impact of aeroplane noise on mental and physical health: a quasi-experimental analysis. *BMJ Open* 2022, 12 (5).

[2] *Ciclo de vida del software: Todo lo que necesitas saber.* Consultado en https://intelequia.com/blog/post/ciclo-de-vida-del-software-todo-lo-que-necesitas-saber.

[3] *Envirosuite. Airport Noise and Flight Track Monitoring Systems.* Consultado en https://www.airport-suppliers.com/supplier/emsbk/.

[4] *ANOMS.* Consultado en https://www.airport-suppliers.com/product/anoms/.

[5] *ANOMS Airline Compliance.* Consultado en https://www.airport-suppliers.com/product/anoms-airline-compliance/.

[6] *ANOMS Carbon Emissions.* Consultado en https://www.airport-suppliers.com/product/anoms-carbon-emissions/.

[7] *ANOMS Advanced.* Consultado en https://www.airport-suppliers.com/product/anoms-advanced/.

[8] *WebTrack.* Consultado en https://webtrak.emsbk.com/vlc.

[9] Maisonneuve, Nicolas y Stevens, Matthias y Niessen, Maria E y Steels, Luc. NoiseTube: Measuring and mapping noise pollution with mobile phones. *Information Technologies in Environmental Engineering: Proceedings of the 4th International ICSC Symposium Thessaloniki*. 2009, 215-228.

[10] Map some noise: how your smartphone can help tackle city sound pollution. Consultado en https://www.theguardian.com/cities/2014/sep/12/map-noise-how-smartphone-help-tackle-city-sound-pollution-noisetube.

[11] *CEAC, ECAC y otros.* Report on Standard Method of Computing Noise Contours around Civil Airports. *BMJ Open*. 2016.

[12] Casado, Rafael y Bermúdez, Aurelio y Hernández-Orallo, Enrique y Boronat, Pablo y Pérez-Francisco, Miguel y Calafate, Carlos T. Pollution and noise reduction through missed approach maneuvers based on aircraft reinjection. *Elsevier. Transportation Research Part D: Transport and Environment*. 2023, 114.

[13] *Fundamentals of Noise and Sound.* Consultado en https://www.faa.gov/regulations_policies/policy_guidance/noise/basics#:~:text=period%20of%20time.-,Noise%20Metrics,a%20one%2Dsecond%20time%20period.

[14] ISO. Acoustics — Description, measurement and assessment of environmental noise — Part 1: Basic quantities and assessment procedures. *ISO 1996-1:2016*. 2016.

[15] *Aircraft Noise and Performance (ANP) Database.* Consultado en https://www.aircraftnoisemodel.org/.

[16] *Policy support and research. Aircraft Noise and Performance (ANP) Database.* Consultado en https://www.easa.europa.eu/en/domains/environment/policy-support-and-research/aircraft-noise-and-performance-anp-data.

[17] *Mobile App vs. Web App: What's the Difference?* Consultado en https://www.springboard.com/blog/design/mobile-vs-website-app/#:~:text=One%20advantage%20that%20web%20apps,the%20benefit%20of%20your%20updates..

[18] Louis, Patrick. Time on Unix. 2020. 10.

[19] *When Do Planes Start Descending?* Consultado en https://www.skytough.com/post/when-do-planes-start-descending.

[20] *¿Qué es Google Maps Platform?* Consultado en https://developers.google.com/maps/faq?hl=es-419#whatis.

[21] *Airlabs Data API.* Consultado en https://airlabs.co/.

[22] *Flightradar24.* Consultado en https://www.flightradar24.com.

[23] Almeida, Fernando y Monteiro, José. The Role of Responsive Design in Web Development. *Webology*. 2017, 14 (2).

[24] *Las coordenadas GPS para aeropuerto Valencia.* Consultado en https://www.123coordenadas.com/coordinates/971911-aeropuerto-valencia.

[25] *Distance on a sphere: The Haversine Formula.* Consultado en https://community.esri.com/t5/coordinate-reference-systems-blog/distance-on-a-sphere-the-haversine-formula/ba-p/902128#:~:text=For%20example%2C%20haversine(%CE%B8),longitude%20of%20the%20two%20points..

[26] *LAmax, SEL and Leq for aviation noise.* Consultado en https://anima-project.eu/noise-platform/lamax-sel-and-leq-for-aviation-noise.

[27] *How to Convert Distances From Degrees to Meters* Consultado en https://sciencing.com/convert-distances-degrees-meters-7858322.html.

[28] *Turofan & Turboprop engines* Consultado en https://oer.pressbooks.pub/introductiontoaerospaceflightvehicles/chapter/turbofan-turboprop-engines/#:~:text=The%20turbofan%20is%20a%20very,a%20very%20high%20bypass%20ratio..

# APÉNDICE A
# Elección de la API

En este capítulo se profundizará en uno de los aspectos que ha resultado clave en el desarrollo de la aplicación web: la elección de la API a la que solicitar datos en tiempo real sobre los aviones. Esta elección a priori parece simple por la cantidad de servicios disponibles, pero las particularidades de cada uno han hecho que sea necesario un estudio de varios de ellos a fin de seleccionar la mejor opción posible.

**Alternativas disponibles**

Existen una gran cantidad de APIs con un funcionamiento muy similar: cuando el usuario se registra obtiene una clave que se puede utilizar para enviar consultas que son respondidas con los datos en cuestión. Algunas de las alternativas más populares son las siguientes:

- Aviationstack.
- FlightAware.
- OpenSky.
- Cirium Flight Stats.
- Trawex.
- Flightslogic.
- OAG Flight Status Data.
- Aviation Edge.
- Skyscanner Travel APIs.
- AeroDataBox.
- Flightlabs.

**Limitaciones de los planes**

Uno de los principales factores que se ha tenido en cuenta a la hora de seleccionar API ha sido el monetario, ya que se pretendía escoger una API que tuviera un

plan con funcionalidades satisfactorias sin necesidad de gastar una gran cantidad de dinero. Además, en caso de que resultara interesante adquirir algún plan de pago, sería preferible hacerlo en una plataforma cuyo funcionamiento se conozca y se haya podido probar en profundidad antes de invertir dinero en ella.

Prácticamente todas las APIs que proporcionan el servicio que se necesitaba disponen de varios planes de pago y ofrecen funcionalidades cada vez más avanzadas en función de la cantidad de dinero que se esté dispuesto a pagar. Los planes más básicos suelen ser gratuitos e incluyen un número muy limitado de consultas, mientras que los planes más avanzados aumentan el límite de consultas y ofrecen más información así como atención al cliente personalizada. Muchos de estos servicios ofrecen también la posibilidad de contactar con el cliente para llegar a un acuerdo sobre un plan personalizado en función de sus necesidades.

En lo que al salto económico entre el plan gratuito y el plan de pago más barato se refiere, es bastante alto: aproximadamente cincuenta dólares. Hay algunos servicios que ofrecen un precio más bajo, de unos veinte o treinta dólares a modo de rebaja, pero esta finaliza tras el primer mes de uso, momento en el que el plan pasar a ser incluso más caro que los cincuenta dólares mencionados anteriormente. Además, las APIs que ofrecen este descuento no suelen ofrecer un plan gratis.

Dada esta situación, lo que se pretendía era encontrar un plan equilibrado: de un precio menor a cincuenta dólares (gratuito a poder ser) y con un número aceptable de consultas por mes. También se dejaron fuera de consideración las APIs que no ofrecían demasiada documentación o que requerían de contactar con el equipo para poder llegar a un acuerdo sobre el uso de sus datos. A continuación se ofrece una lista con las mismas APIs mencionadas en la anterior, pero especificando los motivos por los que fueron descartadas.

- **Aviationstack.** Dispone de un plan gratuito, pero solo permite 100 consultas mensuales. Además, en el caso de esta API ocurre algo destacable. En su plan gratuito solo permite consultas HTTP, y es en el resto de planes en los que se proporciona encriptación HTTPS. El problema surge al adquirir el plan gratuito e intentar realizar alguna petición a la API, momento en el que saltará el error *Your current Subscription Plan does not support HTTPS Encryption*. Para establecer una conexión con el servicio es necesaria la encriptación HTTPS, por lo que, aunque esta página proporcione un plan gratuito, en realidad es inservible.

- **FlightAware.** No dispone de plan gratuito. Tampoco sigue el sistema habitual de contratación de un plan concreto, sino que se cobra por número y tipo de consulta.

- **Cirium Flight Stats.** Tiene un plan gratuito de 20.000 consultas, pero solo puede usarse durante treinta días. Después es necesario cambiar a un plan de pago en el que se cobra por número y tipo de consulta. Para acceder al plan gratuito es necesario también registrarse en un formulario que solicita datos que otras APIs no: ciudad, teléfono, número postal, empresa para la que se desarrolla etc. Además, una vez registrado es necesario esperar a que la cuenta sea validada manualmente.

- **Trawex.** Para poder utilizar cualquier tipo de servicio es necesario solicitar acceso. Además, es una página más enfocada a datos comerciales que a detalles técnicos sobre vuelos, por lo que parece pensada para el desarrollo de páginas web de viajes.
- **Flightslogic.** Para poder utilizar cualquier tipo de servicio es necesario solicitar acceso.
- **OAG Flight Status Data.** Para poder utilizar cualquier tipo de servicio es necesario solicitar acceso.
- **Aviation Edge.** A pesar de que aparentemente tiene un plan gratis, cuando se intenta adquirir aparece en pantalla un mensaje del equipo de desarrollo informando de que ya no se encuentra disponible. Para mitigar el imprevisto se ofrece una rebaja importante en el primer mes de uso de cualquiera del resto de los planes, teniendo el más barato un precio de quince dólares. El problema reside en que tras ese mes de uso, el plan más barato pasa a costar doscientos dólares, lo que convierte a esta API en la opción más cara de todas.
- **Skyscanner Travel APIs.** Para poder utilizar cualquier tipo de servicio es necesario solicitar acceso.
- **AeroDataBox.** Esta API proporciona tanto un plan gratis de prueba con un límite de 100 consultas como varios planes de pago con un precio muy asequible (los dos siguientes planes cuestan respectivamente cinco y treinta dólares mensuales). También clasifica las consultas en cuatro *tiers* diferentes, estableciendo límites de consultas distintos para cada uno (cuanto más alto es el *tier* más limitado está el número de consultas). También ofrece a estudiantes la posibilidad de contactar con ellos para negociar un plan especial. En caso de estar dispuesto a gastar una cantidad no muy grande de dinero esta es probablemente la opción más adecuada, y la que se hubiera escogido si no se hubiera encontrado ninguna alternativa gratuita.
- **Flightlabs.** En su plan más barato presenta una de las peores relaciones precio-número de consultas: un límite de 500 consultas por un precio de treinta dólares mensuales. Esta API no tiene un plan gratuito, pero ofrece un período de prueba de siete días que presenta dos problemas importantes en su funcionamiento. El primero de ellos es dependiendo de si a la consulta se le añaden filtros o no, el número de vuelos devueltos puede cambiar: una consulta sin filtros devolvía tan solo un avión con destino/origen en el aeropuerto de Manises, mientras que una consulta especificando el IATA del aeropuerto como filtro devolvía cuatro aviones. En cualquier caso, esta información sigue siendo imprecisa, ya que el otro problema que plantea esta API es que los datos que ofrece (por lo menos en su plan gratuito) no están actualizados: los datos devueltos se corresponden con el estado del tráfico aéreo de aproximadamente unas dos horas antes a la consulta y no se actualizan en tiempo real.

La conclusión obtenida a partir del estudio realizado es que existen varias opciones bien documentadas y que pueden ofrecer un buen servicio, pero sigue siendo necesario realizar una buena comparación entre los planes que ofrecen, ya que por

un mismo precio no todas las alternativas son igual de rentables. También destaca el hecho de que a la hora de buscar planes gratuitos muchas presentan deficiencias o información demasiado limitada y otras ni siquiera plantean la posibilidad, lo que dificulta el conocer la calidad su funcionamiento sin gastar cierta cantidad de dinero.

Tras descartar el resto de opciones, la escogida para comenzar el desarrollo del proyecto fue Opensky. Esta API es completamente gratis y tras efectuar el registro permite 1.000 consultas diarias, con un límite de una consulta cada dos minutos.

#### OpenSky: la primera elección

El desarrollo de la aplicación web se comenzó utilizando OpenSky para recibir los datos en tiempo real. Sin embargo, en cuanto se quiso empezar con la implementación del código de filtrado de datos para analizar los vuelos con destino u origen en el aeropuerto de Manises esta API presentó un problema crítico: no tiene constancia de este aeropuerto. A este problema se sumaba otro adicional: la información devuelta para cada avión solo contemplaba el lugar de origen, no el destino, siendo además el dato que se indicaba el país de origen, no el aeropuerto.

La API proporciona otras funcionalidades que hubieran sido útiles para mitigar este segundo problema, ya que se puede realizar una consulta que devuelva las llegadas de aviones a un aeropuerto específico y otra para que devuelva las salidas, aunque no resulta muy cómodo, ya que se debe especificar el intervalo de tiempo para el que se quiere que se devuelvan los datos utilizando marcas de tiempo Unix. Aun así dado que esta API no tiene registrado el aeropuerto de Manises, esta opción también era inviable. Esto hizo que esta API también tuviera que descartarse finalmente.

#### Airlabs API: La mejor alternativa

En una segunda búsqueda de posibles APIs se encontró la alternativa que finalmente fue utilizada para desarrollar el proyecto: Airlabs API. Esta API presenta un plan gratuito con un límite de 1.000 consultas mensuales con datos precisos y actualizados en tiempo real, lo que lo convierte en uno de los mejores planes de todos los analizados. Además, entre la información devuelta para cada vuelo se proporciona el IATA del aeropuerto de origen y de destino, lo que facilita mucho el desarrollo del filtro que se pretendía implementar.

El único problema que plantea esta API es que la información que ofrece en su plan gratuito es limitada. Esta limitaciones no aparecen especificadas, pero sí que se menciona que existen en el lista de características del plan. Resultó que una de ellas era que los datos de aviones que entraban en un radio de unos veinte kilómetros alrededor del aeropuerto de Manises dejaban de estar disponibles, lo cual suponía un gran problema, ya que ese es justo el área de mayor interés para este proyecto. Sin embargo, tras contactar con el equipo explicando la situación ampliaron el plan para que esta información pudiera estar disponible sin necesidad de pagar ninguna cantidad extra de dinero.

# APÉNDICE B
# Objetivos de Desarrollo Sostenible

En este apéndice se muestra el grado de relación del proyecto desarrollado con los Objetivos de Desarrollo Sostenible (ODS) utilizando la tabla B.1 y posteriormente se justifica la relación establecida.

| **Objetivos de Desarrollo Sostenibles** | 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 |

**Tabla B.1:** Relación del proyecto con los Objetivos de Desarrollo Sostenible.

De los anteriores objetivos el proyecto se relaciona con:

- **Salud y bienestar.** El proyecto está directamente relacionado con la estimación del ruido generado por los aviones, siendo este un problema que tal y como se mencionó en el apartado de motivación (en el capítulo *Introducción*), puede afectar a la salud mental y física de las personas expuestas durante largos períodos de tiempo e influir en su día a día: aumentando sus niveles de estrés, provocando trastornos del sueño, etc. Conocer y documentar el impacto que

provocan los aviones es el primer paso para poder tomar acción en base a los resultados obtenidos y comenzar el diseño de planes que mejoren la situación de las personas que residen cerca de los aeropuertos.

- **Industria, innovación e infraestructuras.** La solución diseñada permite relacionar el ruido generado con el tipo de motor y de modelo de avión, lo que puede ser útil para realizar investigaciones sobre cuáles son los modelos y motores más silenciosos y, de este modo, contribuir a la reducción de la contaminación acústica. Se abre por tanto la posibilidad a fomentar el uso de los modelos más eficientes en términos de emisión de ruido.

- **Energía asequible y no contaminante.** En este caso se aborda la contaminación desde el punto de vista acústico, un problema que afecta principalmente a las ciudades y a la calidad de vida de las personas que viven en ellas. Una reducción del ruido en nuestras ciudades puede contribuir a la mejora del medio ambiente y de la calidad de vida de las personas que residen en las cercanías de los aeropuertos. Además, un ambiente más silencioso puede promover la actividad física al aire libre y la interacción social, lo que puede mejorar la salud y el bienestar de las personas. Este tipo de impacto hace que este punto también está directamente relacionado con los objetivos de **Acción por el clima** y **Ciudades y comunidades sostenibles**, aunque se ha considerado que la relación es baja ya que las mejoras en las emisiones de ruido causadas por aviones no mejorarán la calidad de vida de todas las ciudades y pueblos sino únicamente las de aquellas situadas cerca de algún aeropuerto.

En resumen, el proyecto de documentación del ruido generado por los aviones contribuye a varios de los Objetivos de Desarrollo Sostenible de la ONU, en concreto a los relacionados con la salud y bienestar, industria e innovación, ciudades y comunidades sostenibles y acción por el clima. La solución diseñada permite obtener información valiosa sobre los niveles de ruido y su relación con el tipo de motor y modelo de avión, lo que puede ser utilizado para promover la innovación y el desarrollo de tecnologías más eficientes y sostenibles. Además, la reducción del ruido generado por los aviones puede mejorar la calidad de vida de las personas y el medio ambiente.

# APÉNDICE C
# Glosario

**ICAO 24.** Estándar internacional utilizado en la aviación para la identificación única de aeronaves. También conocido como código de transpondedor, el ICAO 24 asigna a cada aeronave un código alfanumérico de 24 bits que se transmite a través del transpondedor de la aeronave. Este código se utiliza para identificar de manera única a cada aeronave en el espacio aéreo y en los sistemas de seguimiento y vigilancia.

**Transpondedor.** Dispositivo utilizado en telecomunicaciones que emite una señal en una frecuencia determinada cuando lo estimula otra señal externa apropiada.

**ICAO.** Se refiere a la Organización de Aviación Civil Internacional, una agencia especializada de las Naciones Unidas encargada de promover la seguridad y la eficiencia en la aviación civil a nivel internacional. En el contexto de este documento, el término 'ICAO' se utiliza para hacer referencia a los códigos emitidos por esta agencia, que identifican de manera única los tipos de aeronaves mediante un código alfanumérico de cuatro letras (por ejemplo, el ICAO del modelo de avión Boeing 747 es 'B747') y los aeropuertos y modelos de avión mediante un código de cuatro letras (por ejemplo, el ICAO del Aeropuerto de Manises, ubicado en Valencia, es 'LEVC').

**IATA.** Se refiere a la Asociación Internacional de Transporte Aéreo, una organización comercial que representa a las aerolíneas de todo el mundo. En el contexto de este documento, el término 'IATA' se utiliza para hacer referencia a los códigos emitidos por esta organización, que permiten identificar de manera única un aeropuerto en el ámbito de la industria de la aviación comercial (por ejemplo, el código IATA del Aeropuerto de Manises, ubicado en Valencia, es 'VLC').

**ANP.** Aircraft Noise and Performance.

**ECAC.** European Civil Aviation Conference.

**CEAC.** Conference Europeenne de l'aviation civile.