UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de una aplicación móvil para la medida del ruido generado por aeronaves

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Fresquet Micó, Antonio

Tutor/a: Hernández Orallo, Enrique

CURSO ACADÉMICO: 2023/2024

Resum

Aquest treball de fi de grau tracta sobre el disseny d'una aplicació mòbil per a la detecció de soroll en temp real provenient d´aeronaus. L'objectiu es crear una ferramenta simple i façil d'usar que ajude a totes aquelles persones que estan sotmeses al soroll provenient d'avions, ja siga per viure o treballar prop d'aeroports.

El desenvolupament s'ha dut a terme amb una metodologia agile com es SCRUM i l'us d'algunes tecnologies com Android Studio o Jetpack Compose. El resultat es una aplicació totalment funcional amb una interfaz intuitiva i simple que permiteix la grabació de soroll provenient d'aeronaus de forma efectiva y ràpida.

Finalment arrivem a la conclusió, que destaca la falta de ferramentes de programari gratuites y simples per ajudar a l'individu a protegir tant la seua salut fisica i mental com la seua qualitat de vida.

Paraules clau: Aplicació Mòbil, Android, Servici Web, Trajectoria d'aeronaus, Salut i Benestar

Resumen

Este trabajo de final de grado trata sobre el desarrollo de una aplicación móvil para la detección de ruido en tiempo real proveniente de aeronaves. El objetivo es crear una herramienta simple y fácil de usar que ayude a todas aquellas personas que están sometidas al ruido proveniente de aviones, ya sea por vivir o trabajar cerca de aeropuertos.

El desarrollo se ha llevado a cabo con una metodología agile como es SCRUM y el uso de algunas tecnologías como Android Studio y Jetpack Compose. El resultado es una aplicación totalmente funcional con una interfaz intuitiva y simple que permite la grabación de ruido proveniente de aeronaves de manera efectiva y rápida.

Finalmente llegamos a unas conclusiones que destacan la falta de herramientas de sotfware gratuitas y simples para ayudar al individuo a proteger tanto su salud física y mental como su calidad de vida.

Palabras clave: Aplicación Móvil, Android, Servicios Web, Trayectoria de Aeronaves, Salud y Bienestar

Abstract

This bachelor thesis covers the desing and development of a real time aircraft noise detection mobile application. The objetive is to create an effective and simple tool for helping all those people who suffer from the noise produced by airplanes, wether they live or work near airports.

The development was possible thanks to agile programming methodologies like SCRUM and the use of technologies like Android Studio and Jetpack Compose. The result is a totally functional and easy to use application with an intuitive interface that allows for a recording of the aircrafts fast and efective.

Finally we got to the conclussion that remark the lack of free and simple sotware tools for helping the individual to protect both his physical and mental health as well as his wellbeing.

Key words: Mobile App, Android, Web Service, Airship Routes, Health and Wellbeing

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

En nuestra sociedad actual los vuelos comienzan a verse como cualquier otro medio de transporte y el número de estos no deja de aumentar cada año, incluso se espera que supere los niveles pre-COVID este 2024[1]. Y dado que esto solo se ha visto frenado por alguna crisis de carácter mundial como el ya mencionado coronavirus, es natural plantearse si un incremento indefinido no trae consigo algún tipo de riesgo para la población.

Uno de estos primeros posibles riesgos que viene a la mente es el ruido generado por las aeronaves en sí. Y es que una exposición prolongada a ruidos fuertes, como el generado por aeronaves, puede provocar insomnio o incluso enfermedades de carácter mental[2]. Actualmente existen regulaciones establecidas por la Comunidad Europea[3] sobre el ruido máximo que puede provocar un avión a poblaciones cercanas a un aeropuerto.

Sin embargo, no existe para los individuos de dichas poblaciones una forma clara y cuantitativa de poner a prueba que se cumplen estas normativas sin involucrar a las autoridades competentes.

1.1 Motivación

El ruido de las aeronaves es un problema cada vez más importante para los ciudadanos que viven o trabajan en las inmediaciones de los aeropuertos. La idea de este Trabajo Final de Grado (TFG) surge ante la necesidad de facilitar una herramienta accesible y fácil de usar a los individuos que residan en zonas afectadas por la contaminación acústica generada por aeronaves.

Por ello se cree que el desarrollo de una herramienta que mida el ruido en tiempo real e identifique la aeronave que lo provoca, podría proporcionar datos muy valiosos. Estos datos servirían para llevar un seguimiento y un control sobre el ruido generado por las aeronaves por parte de la ciudadanía de manera descentralizada y accesible.

1.2 Objetivos

El objetivo de este TFG es el desarrollo integral de una plataforma móvil dedicada a medir el ruido generado por aeronaves en tiempo real con el fin otorgar a los usuarios una herramienta para controlar que se cumplen las normativas de ruido establecidas por la Comunidad Europea.

Los subobjetivos subyacentes son los siguientes:

- Construir una interfaz gráfica sencilla e intuitiva
- Garantizar unas mediciones del ruido correctas
- Llevar a cabo el proceso de medición del ruido en 15 segundos o menos

1.3 Impacto Esperado

Desde el punto de vista de un ciudadano que viva en la inmediaciones de un aeropuerto, esto supondrá una herramienta poderosa para obtener datos y controlar la contaminación acústica de su entorno. Con esto, cabría esperar una mejora en la salud de la población ya que podría reducir los problemas médicos relacionados con el ruido.

Por otra parte, se espera que muchos aeropuertos deban tomar medidas en el caso de que los ciudadanos comiencen a reportar incidencias de ruido más elevado de lo que indica la Comunidad Europea. Esto incluso podría ser positivo para la economía de dichos ciudadanos ya que actualmente, las viviendas cercanas a los aeropuertos suelen tener un valor mas reducido a causa del elevado ruido o los riesgos de salud entre otros[4].

1.4 Metodología

La metodología de trabajo es el conjunto de rutinas y decisiones que se adoptan en un proyecto con el objetivo de llevar a cabo una correcta planificación de las tareas, objetivos y tiempos a cumplir por parte de todos los participantes. El objetivo de las metodologías es buscar la efectividad en el desarrollo al mismo tiempo que control sobre el proyecto e información sobre su estado actual.

En este caso la metodología seleccionada para llevar a cabo este proyecto es SCRUM[5]. SCRUM es una metodología ágil, las cuales se caracterizan por su carácter iterativo, es decir, desarrollar versiones del producto incompletas pero funcionales. De este modo el proceso de desarrollo es mas flexible y permite cambios en los requisitos del producto final. Estas iteraciones se llevan a cabo en lo que se denomina sprints, intervalos de tiempo entre una y tres semanas en los cuales se espera finalizar una versión funcional del producto software.
En la metodología SCRUM encontramos los siguientes roles:

- Product Owner: es la persona que posee toda la información necesaria sobre el producto final. Su trabajo consiste en comunicar su visión del producto para que los desarrolladores puedan llevarla a cabo.
- Scrum Master: es la persona encargada de controlar el proceso de los sprints y de la metodología en general, actuando así como un facilitador para los desarrolladores.
- Desarrolladores: son todas aquellas personas que trabajan desarrollando los entregables del proyecto. También deben crear el backlog, este se crea al principio del proyecto y es una lista con todas las tareas que deben completarse para el final del proyecto. El backlog puede cambiar durante el desarrollo para adaptarse a nuevos requisitos o restricciones.

Sin embargo, esta metodología al igual que el resto, están pensadas para ser realizadas por equipos, así que requerirá una cierta adaptación para poder aplicarla a este proyecto en el que solo trabajará una persona. Esto supone un reto considerable ya que será necesario que una misma persona realice los tres roles explicados más arriba.

1.5 Estructura de la memoria

La estructura de la memoria consistirá en lo siguiente:

- Capítulo 1. Introducción, se hablará de temas como la motivación detrás del proyecto, el impacto que se espera provocar o los objetivos que se esperan cumplir.
- Capítulo 2. Estado del arte, se comparará esta aplicación con otras que existan actualmente con una funcionalidad similar con el objetivo de ver las diferencias o las características de las que carecen.
- Capítulo 3. Análisis del problema, destacando las distintas necesidades y oportunidades, describiremos las soluciones posibles para la resolución más efectiva.
- Capítulo 4. Diseño de la solución, aquí hablaremos sobre las herramientas y las decisiones de diseño que llevaremos a cabo para el desarrollo más efectivo de la aplicación.
- Capítulo 5. Desarrollo de la solución, hablaremos sobre problemas o cambios que hayan surgido durante el desarrollo.
- Capítulo 6. Pruebas, trataremos el tipo de pruebas y controles a los que hemos sometido la aplicación.
- Capítulo 7. Conclusión, echaremos un vistazo al trabajo realizado para comprobar si los objetivos han sido logrados, lo aprendido realizando este TFG, así como errores que podríamos haber evitado.
- Capítulo 8. Trabajos futuros, aquí hablaremos de funcionalidades que por complejidad o tiempo no se pudieron incluir en la aplicación o características que podrían mejorarse de la misma.
- Capítulo 10. Referencias, en este capítulo se documentará de donde se obtuvo la información presentada a lo largo de este documento.

CAPÍTULO 2
Estado del Arte

En este capítulo se analizarán las soluciones que existen para la resolver de forma completa o parcial el problema. Al mismo tiempo veremos de lo que carecen y las características y funcionalidades que nuestra aplicación podría proporcionar para diferenciarse de las alternativas.

2.1 Aplicaciones para la detección y escucha de ruido

2.1.1. Detección de ruido general

Actualmente podemos encontrar una gran variedad de aplicaciones móviles y web que utilizan el micrófono del dispositivo para medir el ruido. Sin embargo, resultan demasiado ineficaces y generalistas para el objetivo descrito más arriba, ya que solamente miden el ruido ambiente sin aportar más información sobre el vuelo que podríamos estar grabando.

2.1.2. WEBTRAK

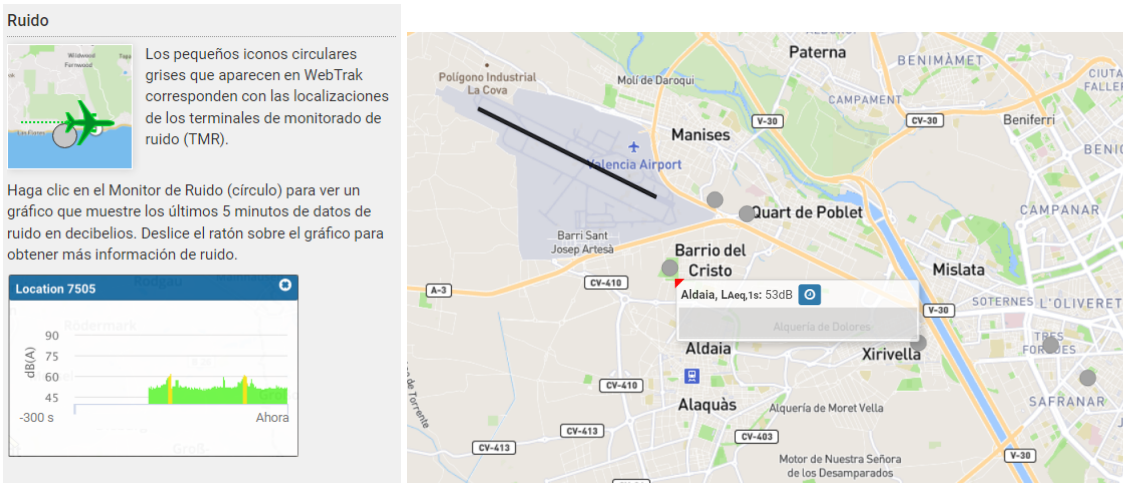

Figura 2.1: Capturas de pantalla de la web WEBTRAK

WEBTRAK[6] es un sitio web compuesto por varias páginas web operadas por Envirosuite Operations Pty Ltd. Envirosuite es una empresa dedicada a la medición de ruido en tiempo real de forma automática mediante software. Su página, posee diversas funcionalidades, entre ellas la visualización de rutas aéreas en tiempo real o la situación de

ruido en tiempo real gracias a los distintos micrófonos situados en los alrededores de los aeropuertos. Por ejemplo la empresa pública AENA[7], encargada de gestionar los aeropuertos en España, usa este sistema para proporcional a los habitantes de poblaciones vecinas a los aeropuertos con este tipo de información.

Se acerca de manera mucho mas satisfactoria a los objetivos que buscamos, sin embargo, se ve limitada por la situación estática de los micrófonos y el hecho de que los usuarios no puedan realizar sus propias mediciones. Es difícil comprobar que avión ha sido el causante del ruido debido al desfase de 30 minutos que tiene la información sobre los aviones.

2.1.3. ExPlane

ExPlane[8] es una aplicación móvil desarrollada para Android y iOS que consigue cumplir muchos de los objetivos establecidos. La aplicación detecta aviones en tiempo real cercanos a la posición del dispositivo y procede a realizar una grabación del ruido generado. Los datos que se obtiene de grabación son tales como los decibelios, el avión al que se ha grabado así como su identificador y altitud.

Sin embargo, su desarrollador, Roelof Meijer, ya no da soporte a la aplicación desde 2018 por lo que dejó de ser posible descargarla en los dispositivos con versiones de sistemas operativos más recientes.

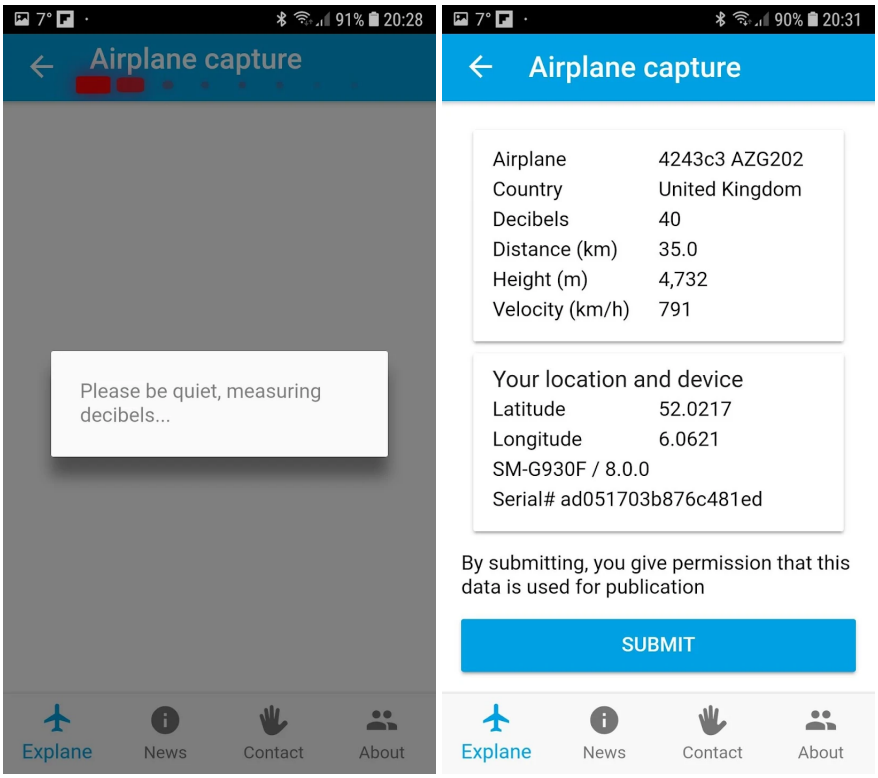

Figura 2.2: Capturas de pantalla aplicación ExPlane

2.2 Crítica a la situación tecnológica actual

Las aplicaciones que hemos descrito se encargan de lidiar con el problema en cierta medida. Desgraciadamente, o bien no están lo suficientemente especializadas, o carecen del soporte necesario para cumplir nuestros objetivos. La detección de ruido generado

por aeronaves es un problema muy concreto que afecta todos los días a ciudadanos de todo el mundo.

La evidente falta de recursos efectivos para los individuos que adolecen de dicha situación resulta muy crítica. Sin duda un factor importante es que este problema solamente afecta a un sector de la población más bien reducido cuando lo comparamos con la población total de una ciudad. Por otro lado que estas personas posean unos ingresos más reducidos tampoco favorece un cambio urgente en la situación actual.
Por estos motivos, no sorprende que empresas y compañías no se hayan lanzado a desarrollar más herramientas para los individuos que sufren de esta situación.

2.3 Propuesta

Considerando lo anterior y teniendo los objetivos en mente, la aplicación que se propone en este TFG pretende recopilar y mejorar las características más útiles y prometedores de las aplicaciones que hemos estudiado más arriba. La plataforma se podrá usar en dispositivos móviles, para que sea accesible a una gran mayoría de la población. El objetivo es hacerla simple y fácil de usar, para no solo garantizar el acceso a ella, sino para que se pueda usar de manera efectiva. Al mismo tiempo buscamos que esta simplicidad no comprometa la calidad de la aplicación, y esta aporte datos y resultados correctos.

Con todo esto la idea es obtener una herramienta para medir en decibelios el ruido generado por aeronaves en tiempo real. Que esta escucha aporte otra información relevante como la altitud de la aeronave o su identificador. Y por supuesto que estos resultados se puedan guardar y consultar posteriormente para su estudio o presentación.

CAPÍTULO 3
Análisis del problema

En este capítulo analizaremos el problema que se pretende resolver. Para ello estudiaremos distintos aspectos como los requisitos de la aplicación, sus casos de uso, el modelo de dominio, la solución propuesta, el plan de trabajo y el presupuesto.

La aplicación permitirá al usuario realizar grabaciones para medir el ruido provocado por aeronaves. Primero se obtendrá la ubicación del dispositivo que está realizando la grabación utilizando el GPS del mismo. La ubicación del dispositivo se cotejará con los datos de una API de tráfico aéreo en tiempo real, de esta manera sabremos a que avión pertenece el ruido que se ha registrado. Tras realizar la grabación, esta quedará registrada en el dispositivo para poder consultarla posteriormente, eliminarla o compartirla.

3.1 Requisitos

A continuación definiremos por categorías los requisitos que servirán como base para el diseño de la aplicación.

3.1.1. Funcionales

- El usuario podrá iniciar la grabación cuando se den las condiciones apropiadas
- El usurario podrá revisar grabaciones hechas con anterioridad
- El usuario podrá eliminar grabaciones hechas con anterioridad
- El usuario podrá compartir grabaciones hechas con anterioridad

3.1.2. No funcionales

- La aplicación deberá poder realizar una grabación en 15 segundos o menos
- La interfaz deberá ser intuitiva y fácil de usar para usuarios no familiarizados con la tecnología
- Los algoritmos de búsqueda y el uso de recursos deberán estar optimizados para minimizar el impacto en la memoria del dispositivo.

3.2 Casos de uso

En esta sección presentaremos el diagrama de casos de uso, junto con el análisis y la descripción de cada caso de uso.

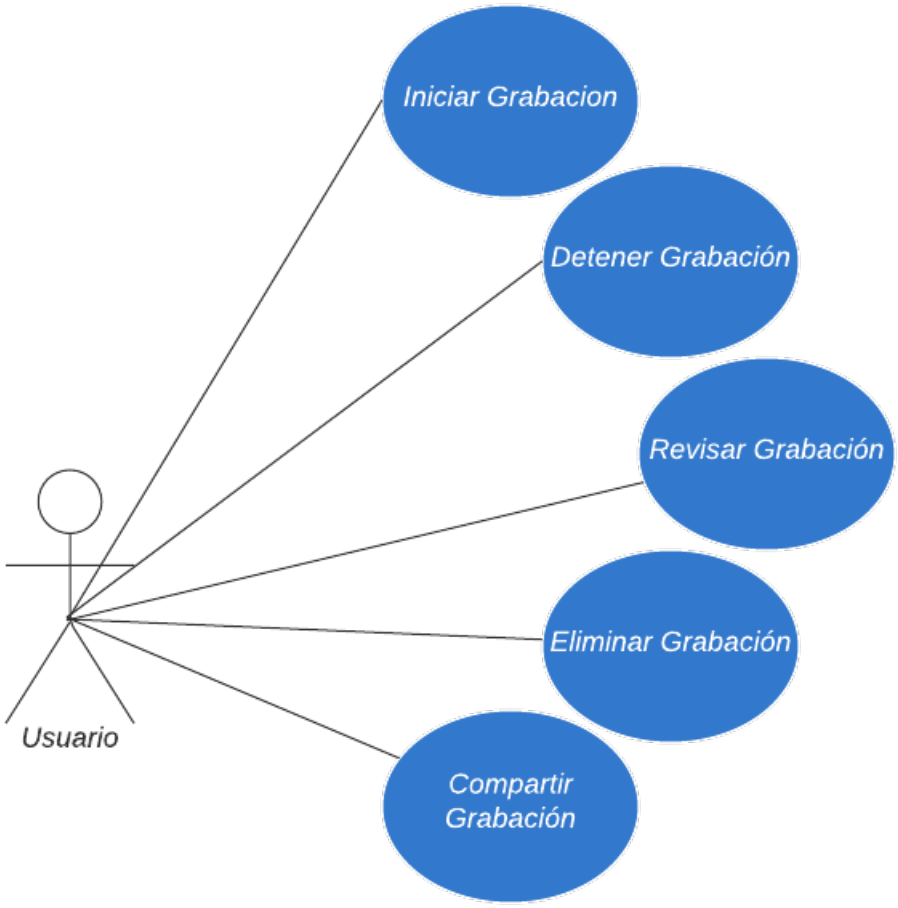

Figura 3.1: Diagrama de casos de uso del usuario

Caso de uso: Iniciar grabación
Descripción: El usuario tendrá la opción de iniciar la grabación
Actores: Usuario
Precondición: Tienen que darse una serie de condiciones tales como, que este sobrevolando una aeronave la posición del usuario, y que dicha aeronave esté por debajo de cierta altura
Relaciones:
Escenario principal:

1. El usuario selecciona la opción de comenzar a grabar.
2. El sistema comienza la grabación y finaliza cuando esta se complete.
3. El usuario finaliza la grabación cuando considere oportuno.
4. El usuario puede revisar los datos de la grabación.

Caso de uso: Revisar grabación
Descripción: El usuario tendrá la opción de revisar los datos de una grabación
Actores: Usuario
Precondición: Tienen que existir una grabación realizada previamente
Relaciones:
Escenario principal:

1. El usuario se dirige al apartado de grabaciones.
2. El sistema muestra todas las grabaciones y el usuario elige una.
3. El usuario puede revisar los datos de la grabación seleccionada.

Caso de uso: Eliminar grabación
Descripción: El usuario tendrá la opción de eliminar los datos de una grabación
Actores: Usuario
Precondición: Tienen que existir una grabación realizada previamente
Relaciones:
Escenario principal:

1. El usuario se dirige al apartado de grabaciones.
2. El sistema muestra todas las grabaciones y el usuario elige una.
3. El sistema elimina la grabación seleccionada.

Caso de uso: Compartir grabación
Descripción: El usuario tendrá la opción de compartir los datos de una grabación
Actores: Usuario
Precondición: Tienen que existir una grabación realizada previamente
Relaciones:
Escenario principal:

1. El usuario se dirige al apartado de grabaciones.
2. El sistema muestra todas las grabaciones y el usuario elige una.
3. El sistema muestra las opciones por las que se puede compartir la grabación.
4. El usuario escoge una opción y el sistema procede a compartir los datos de la grabación seleccionada por el método escogido.

3.3 Modelo de dominio

El modelo de dominio se usa para representar en forma de diagrama, los conceptos fundamentales de la aplicación junto con las relaciones que poseen entre ellos. Esto nos ayudara a entender de una manera más clara los conceptos claves y la lógica detrás de la aplicación. Es una parte esencial del análisis ya que proporciona una base sólida sobre la que fundamentar todo el proceso de desarrollo.

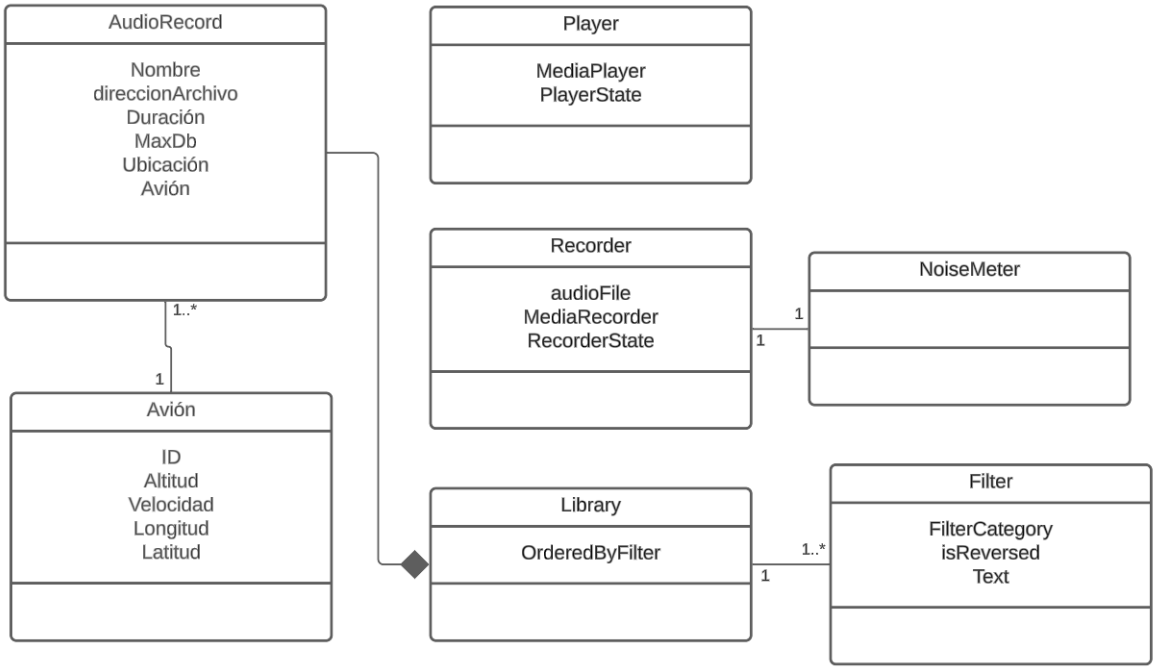

Figura 3.2: Modelo de dominio

En la figura 3.2 podemos observar el diagrama del modelo de dominio. No es demasiado amplio, ya que la dificultad en el desarrollo de la aplicación no radica en la magnitud de esta, sino en la complejidad de sus partes.

Los AudioRecords son la parte central de la aplicación. Estos hacen referencia a las grabaciones realizadas con éxito. Contienen información útil como la propia grabación o datos de carácter más pragmático como la fecha cuando se realizo la grabación o los Db máximos que alcanza dicha grabación. Todas estas grabaciones estarán asociadas con la aeronave cuyo ruido ha sido grabado. Sobre esta aeronave se obtienen datos que nos ayudan a contextualizar el ruido, como pueden ser su velocidad o su altitud, y un identificador, para que en caso que se desee, se puedan obtener más datos sobre este.

Estas grabaciones se almacenaran en el dispositivo y serán visibles al usuario gracias a la Library (la Librería). Aquí se podrán visualizar y reproducir las distintas grabaciones. Y haciendo uso de los Filter (Filtros) podremos cambiar el orden de las grabaciones de la librería para tener un mayor control sobre la visualización de los datos.

Por otra parte tenemos el Player (el reproductor) y el Recorder (la grabadora). El Player se encargará de reproducir las grabaciones realizadas previamente. En cuanto a dichas grabaciones las realizará el Recorder, el cual se ayudara del llamada NoiseMeter para medir y aplicar los logaritmos necesarios para la obtención del nivel de ruido.

3.4 Análisis de soluciones posibles

Con estas restricciones, necesidades y requisitos ahora pasamos a valorar que soluciones cumplirían estos de manera más satisfactoria y efectiva para la creación de nuestra herramienta. Partimos de la base de que la aplicación debe de ser móvil, por lo tanto las aplicaciones de escritorio quedarían excluidas.
Esto resulta una decisión lógica ya que una de las características fundamentales de la aplicación seria que esta se pudiera usar de forma fácil en cualquier parte donde se requiera, y las computadoras de escritorio así como los ordenadores portátiles limitarían mucho este aspecto. Por lo tanto nos quedaríamos con dos opciones posibles.

1. **Aplicación web**. Realizar la aplicación de forma web podría resultar muy apropiado. Especialmente por la compatibilidad con los diferentes tipos de dispositivos móviles. Actualmente encontramos varios tipos de sistemas operativos en el mercado y desarrollar una aplicación para cada uno de ellos es una dificultad y un coste añadido. Por ello la oportunidad que brinda las navegadores web para ejecutar la aplicación independientemente del tipo de dispositivo es un punto muy a tener en cuenta. Por no hablar de que no requeriría ningún tipo de espacio ni almacenamiento adicional en el dispositivo desde el que se usa.

   Por otra parte, nos encontramos que la dificultad del desarrollo se incrementaría en otros ámbitos. El acceso al micrófono y al GPS del dispositivo se complica dado que hay que proporcionar al navegador con una serie de permisos, así como una cantidad de datos que desde el punto de vista simple y rápido que buscamos en el uso de la aplicación no resultaría muy efectivo.

2. **Aplicación móvil**. Realizar esta herramienta en la forma de una aplicación móvil resulta la solución mas evidente a priori. Esto es así ya que aumenta mucho su simplicidad de uso, tan solo habría que instalar la aplicación una vez y sería muy rápido usarla cada vez que fuera necesario.
   También tendríamos un incremento de velocidad con respecto al uso de la ubicación y micrófonos, ya que tan solo nos pedirían permisos la primera vez que iniciamos la app.

   En cuanto a los puntos negativos, el mas evidente seria carencia de multiplataforma en la aplicación. Debido a restricciones de tiempo la aplicación solo se desarrollará para un único sistema. Esto reduce mucho la base de posibles usuarios, sin embargo no cierra la puerta a un desarrollo posterior en los sistemas operativos que se requieran.
   También hay que tener en cuenta que los sistemas operativos evolucionan, y con ello también deben hacerlo las aplicaciones móviles. Por lo tanto también habrá que asumir un cierto mantenimiento, no muy grande gracias a la reducida magnitud de la app, pero un trabajo extra a tener en cuenta.

Entre estas dos opciones nos inclinamos por la aplicación móvil. Creemos que si bien el desarrollo estará limitado a ciertos dispositivos, el resultado final será mucho mas satisfactorio, sobre todo debido a los tiempos de respuesta y a poder utilizar la aplicación sin conexión a internet, aunque con funciones limitadas. Por ello una vez decidido el tipo de aplicación que desarrollaremos faltará decidir para que sistema operativo lo haremos.

Y aquí nos hemos decidido por Android, el sistema operativo móvil de Google. Las razones detrás de esta elección se detallan en el punto 4.4.1, pero el factor determinante

fue el hecho de que android posee la mayor cuota de mercado de sistemas operativos para dispositivos móviles. Esto nos permite que la aplicación pueda ser más utilizada por más gente.

3.5 Plan de trabajo

A continuación explicaremos el plan de trabajo establecido, junto con las estimaciones del tiempo requerido para llevar a cabo ese plan.

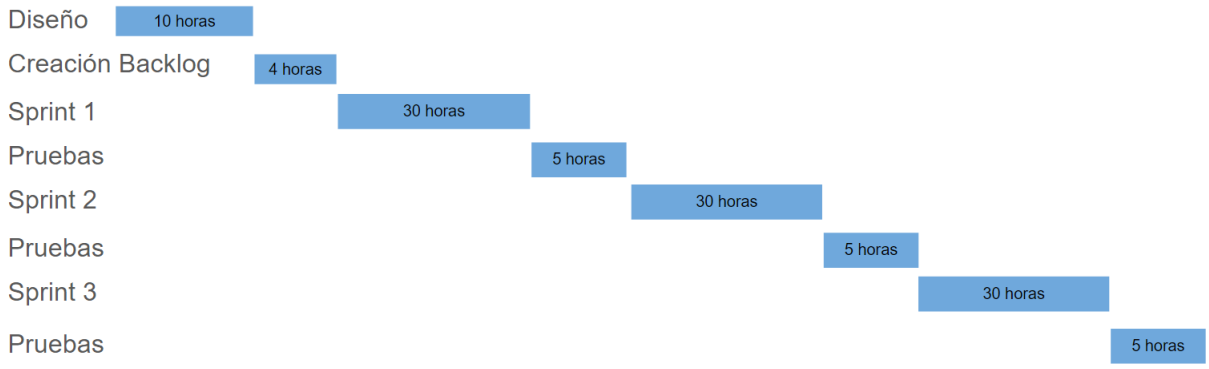

Figura 3.3: Diagrama de Gantt del plan de trabajo estimado

Como se observa en el diagrama, el total estimado del desarrollo serían aproximadamente 120 horas. Y las fases que componen dicho desarrollo son las siguientes:

- Diseño: en esta fase se estudiará como estructurar la aplicación así como identificar y escoger las herramientas necesarias para un desarrollo lo más efectivo posible.
- Creación del Backlog: como ya se explicó anteriormente, la metodologia escogida es SCRUM, por ello es indispensable la existencia del backlog. En su creación se deberán incluir todas las posibles funcionalidades de la aplicación o tareas de desarrollo que vayan a requerir tiempo de desarrollo.
- Sprints: se ha decidido llevar a cabo tres sprints de 10 días cada uno. Estos cuentan con su propia estructura interna y estimación que se muestra a continuación.

Figura 3.4: Diagrama de Gantt del plan de trabajo estimado para cada Sprint

Donde la planificación consiste en identificar los elementos del backlog que se realizarán durante el Sprint.
La revisión consistirá comprobar que la iteración de la aplicación funciona correctamente y detectar posibles errores y fallos.
Por ultimo, la retrospectiva resulta útil para comprobar como fue el trabajo desde un punto de vista laboral. Identificar problemas con horarios y herramientas buscando aumentar la eficiencia y la productividad de cara al siguiente Sprint.

- Pruebas: en este apartado se desarrollaran pruebas para comprobar que las funcionalidades recién introducidas funcionan correctamente. En caso contrario se documentaran los errores y se incluirán en el Sprint siguiente.

3.6 Presupuesto

En este apartado estudiaremos el presupuesto necesario para llevar a cabo este proyecto. A la hora de realizar un presupuesto, este se puede dividir en dos categorías, recursos humanos y recursos materiales.

3.6.1. Recursos humanos

El plan de trabajo explicado en el apartado anterior esta diseñado para ser llevado a cabo por una única persona. Por ello, solo contaríamos con un empleado.
Según las estimaciones de dicho plan de trabajo, el proyecto requeriría de unas 120 horas de trabajo, lo que convertido en jornadas laborales serian 3 semanas de trabajo completas. Por otro lado el salario medio de un desarrollador full stack, dado que realizará labores de desarrollo back end y front end, son 2.990 euros mensuales[9].
Por lo tanto, el coste humano de este proyecto sería de 2.243 euros.

3.6.2. Recursos materiales

En el aspecto material el proyecto requiere de muchos más elementos. A continuación se desglosan en la siguiente tabla:

Portátil	500€
Internet	20€/mes
Luz	60€/mes
API trafico aéreo	150€/mes

Tabla 3.1: Recursos materiales

Como podemos observar los gastos en el apartado hardware no son demasiado elevados comparadas con el coste humano. El mayor gasto sería el dispositivo de trabajo desde el cual se realizaría el desarrollo.

Sin embargo, si tuviéramos estos gastos durante varios meses, el mayor gasto se convertiría en la mensualidad de la API de tráfico aéreo. Esta API resulta indispensable para obtener información sobre las aeronaves como su posición o el aeropuerto de destino.
Existen varios precios para una misma API, estos precios dependen del número de consultas que se realizan al servidor al mes. Al rededor de este precio solemos obtener unas 10.000 consultas al mes, pero si no resultara ser suficiente siempre se puede optar por contratar una opción con un mayor número de consultas mensuales.

CAPÍTULO 4
Diseño de la Solución

A lo largo de este capítulo se tratará de explicar el diseño de la aplicación. Para ello estudiaremos las distintas capas en las que se ha dividido el sistema así como los componentes más destacables de este.

4.1 Arquitectura del Sistema

A continuación pasamos a describir la arquitectura que se ha utilizado para el desarrollo de la aplicación. La arquitectura en cuestión es MVVM (Model-View-ViewModel)[10]. Esta arquitectura fue inventada por los arquitectos de software Ken Cooper y Ted Peters con el objetivo de simplificar la programación de interfaces dirigida por eventos.

Esto es posible mediante la división de la aplicación en tres capas muy definidas, el modelo o domino (Model), la vista (View) y por último el modelo de la vista (ViewModel). Sin embargo, lo que diferencia a esta arquitectura de otras como por ejemplo MVP (Model-View-Presenter) o MVC (Model-View-Controller), es el uso de un Binder[11]. Este binder puede ser implementado de muchas formas pero su objetivo es la sincronización entre la vista y el modelo de la vista sin necesidad de codificar esta lógica de sincronización.

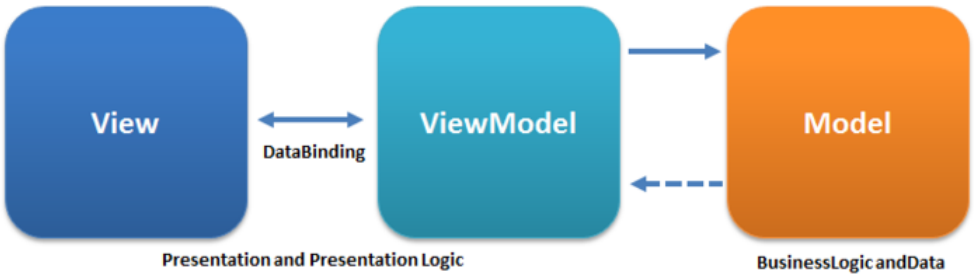

Figura 4.1: Diagrama de la arquitectura MVVM

Vamos a explicar las distintas capas de manera más detallada:

- **Vista:** las distintas vistas están compuestas por los elementos de la interfaz de usuario. Aquí es donde se encuentran todos los elementos visuales que permiten a los usuarios interactuar con la aplicación, así como presentarles los datos obtenidos por la capa de modelo.

- **Modelo:** el modelo o la capa de dominio es donde se encuentra la lógica de la aplicación que regula como se gestionan los distintos datos, al igual que como se obtienen o se tratan.
- **Modelo de la Vista:** esta capa se trata de una abstracción de la vista que, mediante sus propiedades, representa el estado de la interfaz y sirve como puente entre la capa de la Vista y la del Modelo.

4.2 Diseño Detallado

Comenzaremos este apartado detallando lo expuesto en la sección anterior sobre las distintas partes de la arquitectura MVVM.

- **Vista:** Las distintas vistas actúan como la cara de la aplicación, estas son la estructura y la apariencia que el usuario puede ver. Sin embargo, no contienen ningún tipo de lógica de negocio ni tratan los datos de la aplicación de forma significativa más allá de presentarlos al usuario.

  El flujo y los estados de la interfaz son gestionados por el Modelo de la Vista de forma automática. Esto es posible gracias al Binder que en este caso se trata de propiedades que al ser modificadas en el Modelo de la Vista fuerzan la recomposición de la vista para mostrar los nuevos cambios en la interfaz.
- **Modelo:** la capa de dominio o modelo esta compuesta por varias partes a su vez:
  - **Servicios:** partes de l código compartimentadas y bien definidas encargadas de realizar una tarea concreta. Un ejemplo de servicio en la aplicación sería el Servicio de Localización, encargado de obtener la ubicación del dispositivo.
  - **Objetos:** los objetos que reflejan aspectos del mundo real en la aplicación. Algún ejemplo de estos podría ser una grabación de audio o un Avión
  - **Repositorios:** los repositorios son los encargados de gestionar las llamadas y solicitudes a las APIs. En este caso solo tenemos un repositorio encargado de comunicarse con la API de tráfico aéreo.
- **Modelo de la Vista:** como ya hemos dicho el modelo de vista trata de abstraer una vista y reflejar su estado mediante una serie de propiedades. Estas propiedades actúan como Binder y se pueden ver afectadas por eventos producidos por el usuario o bien por la capa de modelo.

  En cualquier caso un cambio en alguna de estas propiedades, sea donde sea, dispara automáticamente la recomposición de la vista, es decir, la vista se refresca con los nuevos valores de las propiedades. Estas propiedades pueden ser desde una lista de grabaciones, a una variable que indica si la grabadora esta activa.

4.2.1. Diagrama de Clases

A continuación podemos observar en la figura 4.2 el diagrama de clases y Modelos de Vista que compone la aplicación.

En este diagrama observamos que no hay una gran cantidad de clases, esto es debido a que la dificultad del desarrollo de esta aplicación reside en la complejidad de sus partes en lugar de la extensión de esta. En las clases que observamos hemos de hacer una diferenciación con el propósito de las mismas en la arquitectura que hemos utilizado. Encontramos Modelos y Modelos de Vista.

Los Modelos de Vista son las clases llamadas *HomeViewModel* y *LibraryViewModel*. Como se puede observar, están directamente relacionadas con las principales clases de la capa de Modelos, como son *AudioRecord*, *Recorder*, *Library* y *Player*. Esto se corresponde con lo explicado en el punto anterior, ya que se puede ver como estas clases hacen de intermediarias entre las clases de la capa de Modelo y la capa de Vista.

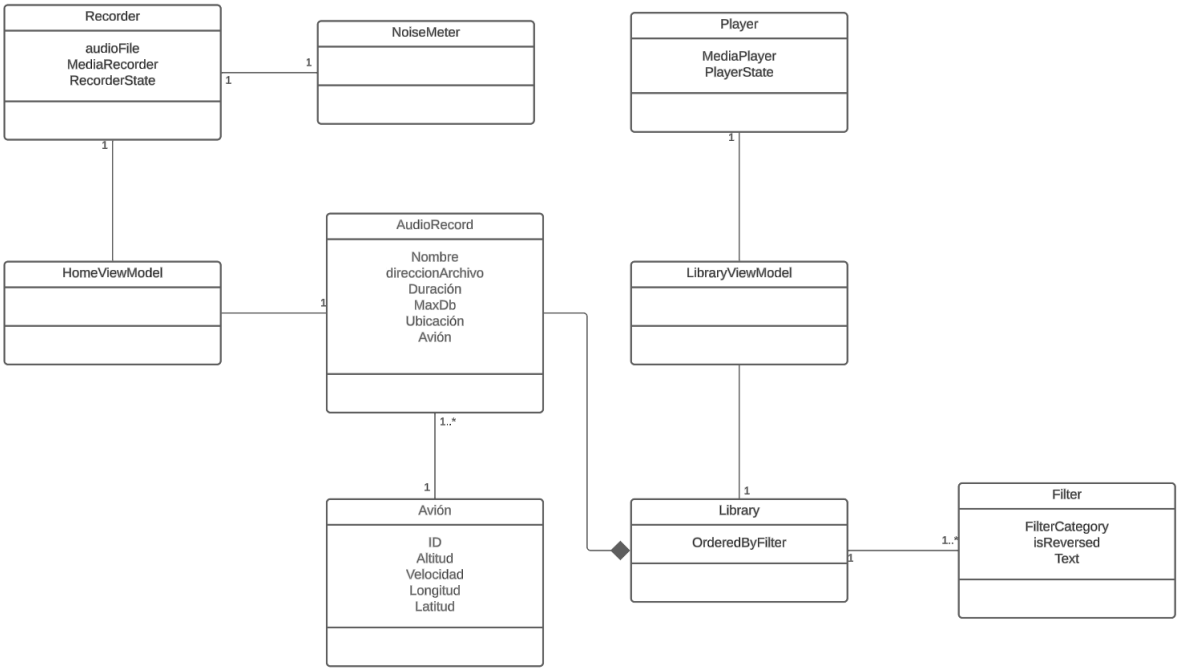

Figura 4.2: Diagrama de clases

4.2.2. Estructura de archivos

A la hora de estructurar los directorios y los archivos de la aplicación valoramos diferentes alternativas. Finalmente nos decidimos por estructurarlos por característica. Nos decidimos por esta forma de estructura debido a que se combina bien con la arquitectura MVVM y ayuda a mantener una estructura bien diferenciada y escalable. Como se puede observar en la figura 4.3 tenemos las capas de *data* y *model* justo debajo del directorio principal, *noiseplane*. Estas carpetas contienen la Lógica de Negocio que es integral a toda la aplicación.

Por otra parte en el directorio *screens* es donde comenzamos a ver la utilidad de esta estructura. Aquí tenemos dos directorios *homescreen* y *recordlibrary*, y en cada uno de estos una estructura identica que los separa en las tres capas explicadas al comienzo de este capítulo.

De esta forma, todas las clases o archivos que tan solo se utilicen para esta pantalla, se encontrarán en este directorio, ayudando a compartimentar y depurar la aplicación de forma eficiente.

4.3 Estructura de la aplicación

A continuación observamos el diagrama de flujo de las ventanas de la aplicación: Como podemos observar, no se trata de una aplicación con muchas ventanas ni diálogos, ya que uno de los objetivos es que la aplicación fuera lo mas sencilla posible para que

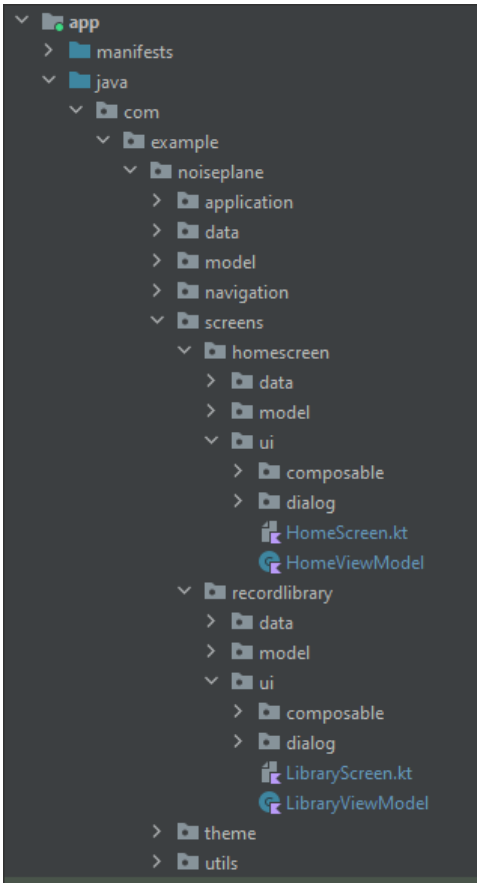

Figura 4.3: Estructura de archivos

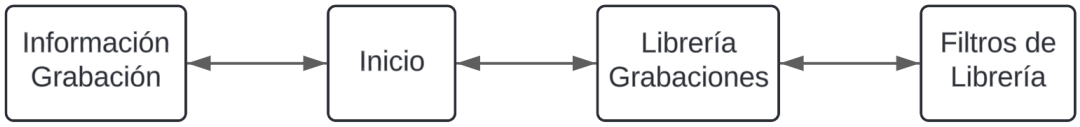

Figura 4.4: Diagrama de flujo de la aplicación

fuera intuitiva y fácil de usar.
Por ello, tenemos dos pantallas, Inicio y Librería Grabaciones, que actúan como pantallas principales e Información Grabación y Filtros de Librería serian diálogos contextuales de las pantallas principales.

4.3.1. Pantalla Inicio

En la figura 4.4 se puede ver los dos estados en el que podemos encontrar la pantalla de inicio. A la izquierda observamos el estado en el que se encuentra la aplicación al iniciarla. Los elementos que encontramos es el botón principal, que dispara la grabación y nos lleva al estado que se puede ver en la imagen de la derecha.
La idea que teníamos con este diseño es que el botón de grabar dominara totalmente la pantalla, ya que es la función entorno a la que gira toda la aplicación.

Por otra parte en la parte superior derecha encontramos un botón contextual, que si no se esta grabando nos llevara a la pantalla de la Librería de Grabaciones y si estamos realizando una grabación, esta se entenderá y nos mostrará el dialogo de dicha grabación.

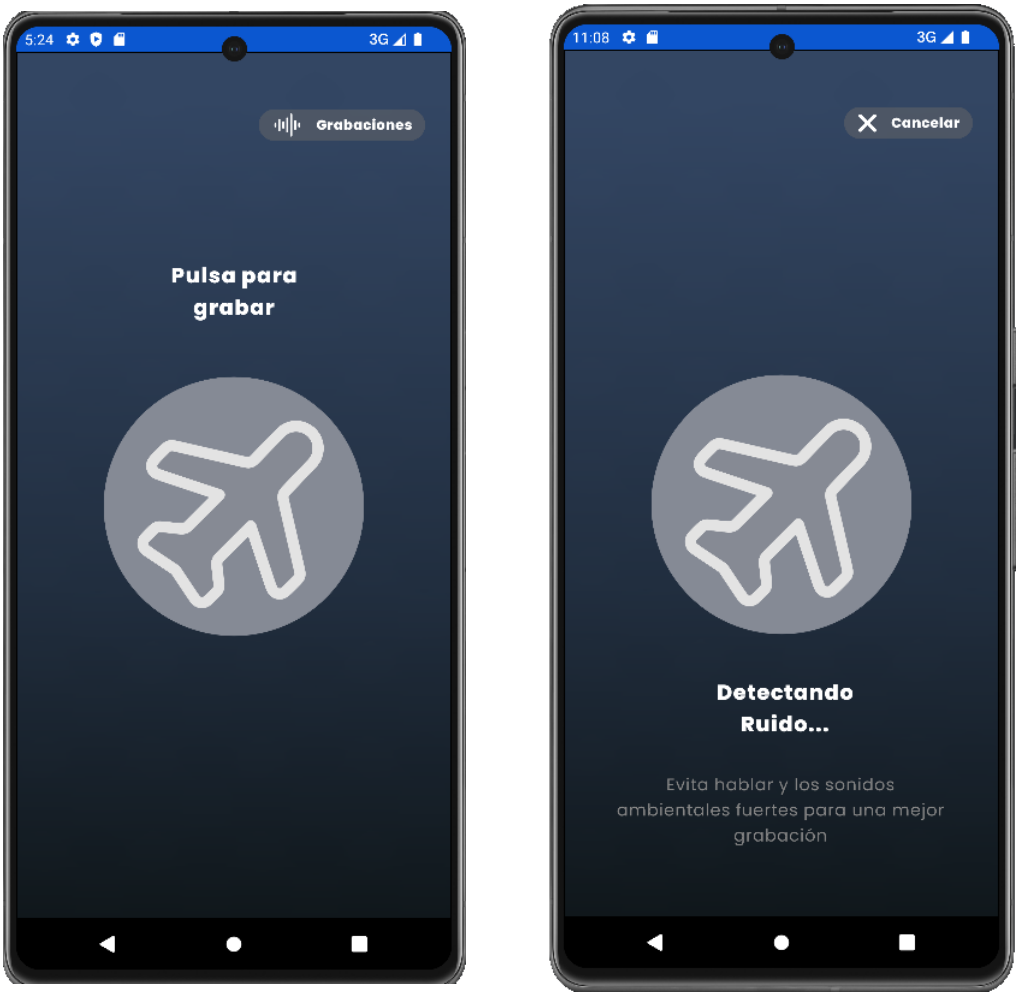

Figura 4.5: Dialogo Información Grabación

En la figura 4.6 vemos el dialogo que aparece cuando la grabación se ha completado con éxito. En este vemos datos tanto de la propia grabación, como la duración o los decibelios máximos alcanzados, como datos sobre el avión que se ha grabado, como puede ser su altitud o velocidad.

En la parte inferior de la ventana de dialogo encontramos dos botones. Estos como su nombre indica nos permiten desechar la grabación o guardarla en el almacenamiento del dispositivo. En cualquiera de estos dos casos volveríamos a la pantalla inicial.

En definitiva, este dialogo nos ofrece una previsualización de lo que sería el registro de la grabación sin tener que almacenarla en el dispositivo. De esta forma evitamos abarrotar la memoria de grabaciones que no han salido como esperaba el usuario.

4.3.2. Pantalla Librería

La segunda pantalla principal que encontramos en la aplicación es la librería de grabaciones. En esta pantalla aparecen todas las grabaciones que han sido realizadas y posteriormente guardadas.

Estas grabaciones se pueden visualizar gracias a una lista de tarjetas con 'scroll' vertical. Para tratar de mostrar el mayor número de grabaciones posibles en pantalla y que al mismo tiempo fueran legibles por el usuario, decidimos ocultar la información sobre el avión. Esta información se hace visible al hacer 'click' sobre una de las tarjetas de gra-

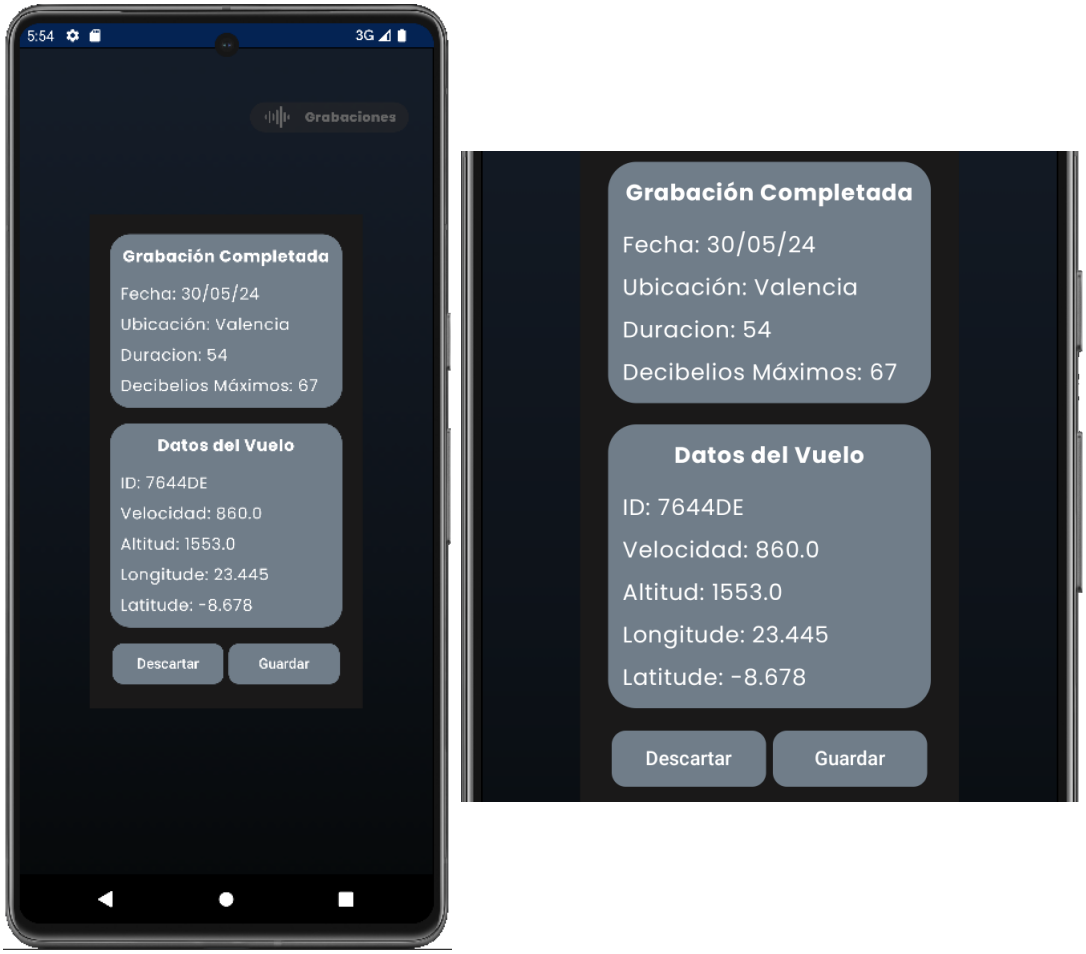

Figura 4.6: Dialogo Información Grabación

bación, esta se despliega, dejándonos el desplegable que se observa en la imagen de la derecha de la figura 4.7.

En esta tarjeta desplegable podemos encontrar toda la información que nos fue presentada en durante el dialogo de previsualización de la grabación. También, junto a los datos de la grabación, es posible volver a escuchar dicha grabación si lo deseamos.

En cuanto a los colores de las tarjetas, la idea detrás de estos esquemas es una rápida visualización de cuales son las grabación de mayor interés, que normalmente serán aquellas en las que se ha registrado un número más alto de decibelios. Por ello, las tarjetas con un decibelios normales o no perjudiciales poseen un color verde, y según el ruido registrado aumenta, el color se va tornando mas rojo e intenso para denotar así su gravedad.

Por otra parte, en la esquina superior izquierda encontramos una flecha para volver a la pantalla de Inicio. Esta navegación también se lleva a cabo usando los botones de navegación del propio dispositivo.

En cuanto a las funcionalidades de borrar y compartir grabaciones, decidimos implementarlas de manera que no abrumara al usuario con botones o iconos que le distrajeran de lo importante, las tarjetas con la información de las grabaciones.

De esta manera, decidimos incorporar las funcionalidades mencionadas arriba de una forma minimalista y sencilla, simplemente arrastrando la tarjeta sobre la que se quiera realizar la acción. Como observamos en la figura 4.8, si deslizamos hacia la derecha compartiremos la grabación y si deslizamos hacia la izquierda borraremos dicha grabación.

Figura 4.7: Pantalla Librería de Grabaciones

Por último, en la pantalla de la Librería de Grabaciones, encontramos un icono en la esquina superior derecha que al ser pulsado abrirá el diálogo que vemos en la figura 4.9. Este diálogo se trata de un menú para elegir el método de filtrado de la lista de grabaciones. Entre las opciones decidimos incluir el orden por fecha, duración y nivel de ruido. Cualquiera de las opciones se puede disponer de manera ascendente o descendente.
Al seleccionar cualquiera de estas opciones y volver a la pantalla de la Librería de Grabaciones, dicha pantalla se recargará para reflejar el nuevo orden elegido.

Figura 4.8: Funcionalidades Compartir y Borrar

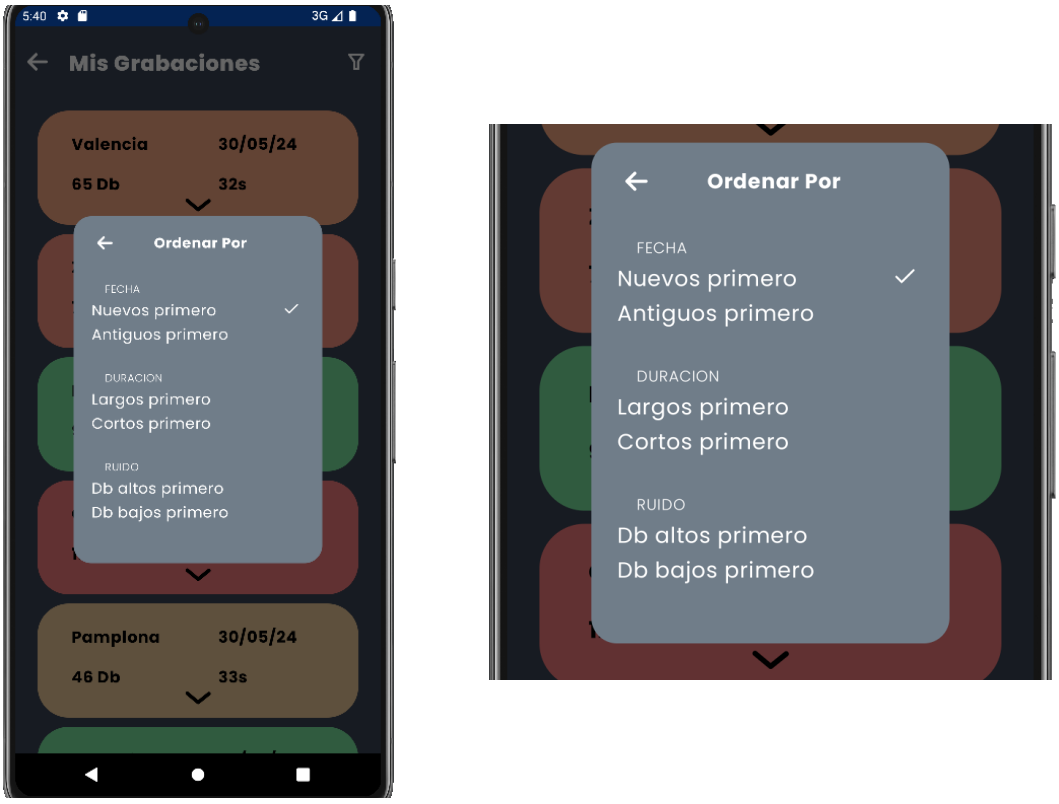

Figura 4.9: Diálogo Filtro Librería

4.4 Tecnología Utilizada

4.4.1. Sistema Operativo e IDE

El Sistema Operativo (SO) utilizado para el desarrollo de la aplicación es Android. Android está desarrollado por Google y actualmente es el sistema operativo para dispositivos móviles más usado con un 70 %[12] de dispositivos móviles llevando este sistema. Con esto nos aseguramos que la mayoría de individuos que necesiten esta aplicación tendrán acceso a un dispositivo Android.

Por consiguiente al haber elegido Android, desarrollamos el proyecto usando Android Studio, en concreto la versión 2022.3.1. Android Studio es el Entorno de Desarrollo Integrado (IDE) oficial utilizado para el desarrollo de aplicaciones para Android. Esta basado en el IDE IntelliJ IDEA, creado por JetBrains.

4.4.2. Lenguaje de Programación

En cuanto al lenguaje de programación, al tratarse de Android Studio las dos opciones que nos encontramos son Java o Kotlin. Una vez más, teniendo en cuenta la inclinación de Android por el desarrollo en Kotlin sobre Java, decidimos utilizar el primero.

Kotlin es un lenguaje orientado a objetos más moderno que Java e incluye algunas características que lo hacen un candidato más apropiado que Java para el desarrollo de aplicaciones. Algunas de estas características son la seguridad de valores Null o algunos aspectos integrados de programación funcional como funciones de orden superior o expresiones lambda.

4.4.3. Interfaz

Una de las decisiones más importantes a la hora de decidir las tecnologías con las que trabajaríamos fue el framework para la interfaz.
Actualmente Android Studio ofrece dos paradigmas:

- **Android View**: La opción más antigua de Android Studio para el desarrollo de interfaces. Estaba basado la programación orientada a objetos y XML.
- **Jetpack Compose**: su framework más reciente, esta basado en la programación declarativa y la composición.

Después de estudiar las dos opciones nos decantamos por Jetpack Compose, aun sabiendo que tiene una curva de aprendizaje más inclinada. Algunas de las razones por las que tomamos esta decisión se detallan a continuación:

- **Estado del Arte**: Android advierte que dará soporte a Jetpack Compose y pronto dejara de actualizar sus sistema basado en Vistas.
  Por otro lado podemos ver una tendencia en el sector de migrar hacia tecnologías declarativas como puede ser Flutter o Switf.
- **Limpieza del Proyecto**: con el sistema de Vistas de Android, la interfaz se divide en dos ficheros: uno XML y el otro de código.
  Con Jetpack Compose toda la interfaz se encuentra en un mismo archivo, lo que facilita tanto la depuración del código como su legibilidad.

- **Reutilización**: Jetpack Compose esta basado el paradigma de programación declarativo y la composición. Esto favorece a la reutilización de código a lo largo de todo el proyecto, pudiendo aplicar cambios de forma selectiva. Esto es posible debido a que se pueden crear elementos independientes de una vista y adaptarlos de forma personalizada según sea necesario para la interfaz.

4.4.4. Figma y Relay

Siguiendo al apartado de las interfaces cabe destacar estas dos herramientas que han resultado muy útiles para las etapas más tempranas del desarrollo.
Figma se trata de una aplicación web para la creación y prototipado de mock ups e interfaces gráficas para todo tipo de proyectos, desde aplicaciones móviles a paginas web.
Usamos esta aplicación para, a partir de los mock-ups básicos que creamos durante la etapa de diseño, prototipar rápidamente la apariencia de una interfaz con un aspecto más realista y pulido.

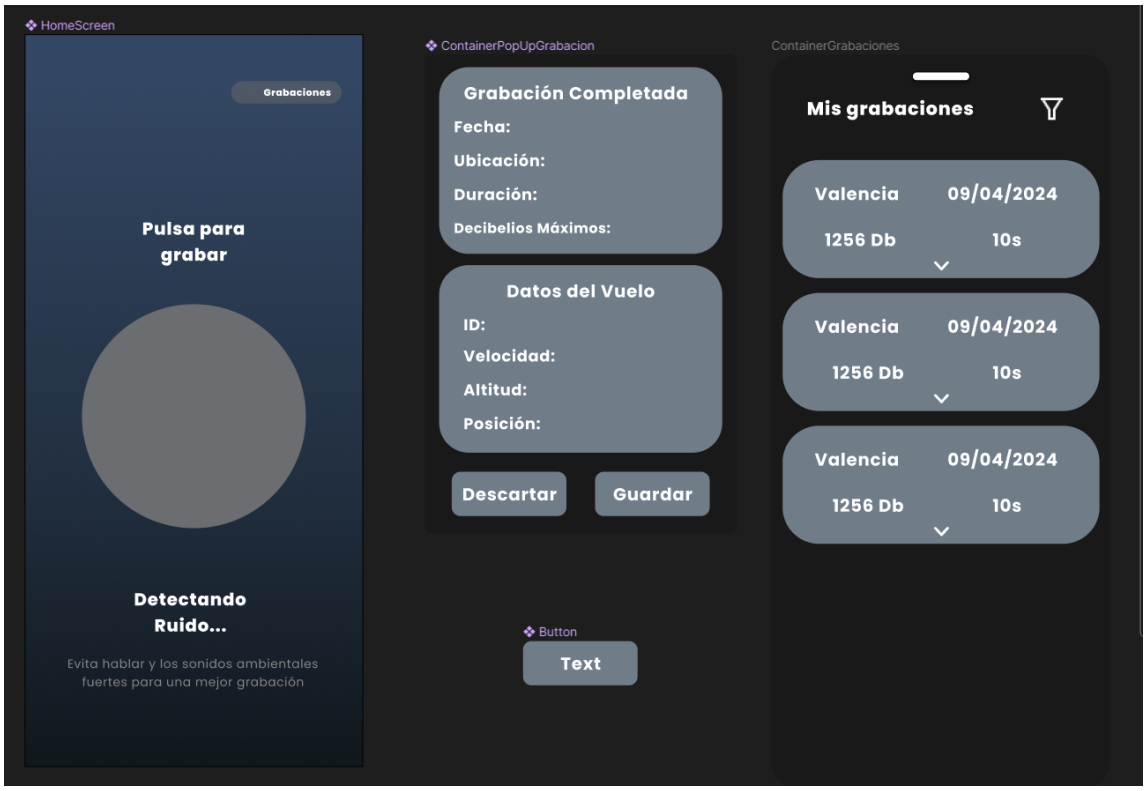

Figura 4.10: Prototipo Interfaz en Figma

Sin embargo, esto no queda ahí sino que gracias a Relay, un plug-in creado por Android para Figma y Android Studio, pudimos importar dichas interfaces al proyecto.
Relay es una herramienta de generación automática de código para Jetpack Compose a partir de un proyecto en Figma. Esto nos permitió editar e iterar entre las distintas versiones de interfaces hasta que obtuvimos el resultado que buscábamos. Por otra parte también nos permitió observar las interfaces en un dispositivo real y comprobar algunas funcionalidad.

No obstante, Relay se encuentra en una versión alpha actualmente, por lo que solamente lo utilizamos como herramienta de prototipado, e implementamos la interfaz de manera manual aplicando los cambios y mejoras correspondientes a la primera iteración de la interfaz producida por Relay.

4.4.5. Persistencia

En este proyecto la persistencia no es un elemento clave, ya que solamente tendríamos que almacenar en la memoria del dispositivo los registros de audio realizados a las aeronaves y recuperarlos todos para presentarlos en una librería al usuario. Dado esto, decidimos elegir la opción más simple y que nos permitiría centrarnos en otras partes más relevantes de la aplicación.

El sistema escogido se basa en la serialización de un objeto (en este caso el registro de la grabación) a un fichero JSON (JavaScript Object Notation) que se guarda en el almacenamiento interno del sistema.

Por otro lado, este objeto guardaría una referencia al archivo de audio correspondiente a la grabación en el memoria externa del dispositivo.

4.4.6. API tráfico aéreo

Una de las partes cruciales de esta aplicación es obtener datos fiables y de forma rápida de las aeronaves. Por ello, a la hora de elegir una API a partir de la cual acceder los datos tuvimos que analizar bien dichas opciones. La gran parte de estas APIs son de pago, ofreciendo alguna de sus versiones más limitadas de forma gratuita. Estas limitaciones muchas veces nos obligaban a dejar de lado algunas funcionalidades que habíamos planeado para la aplicación.

Sin embargo, encontramos la API de OpenSky Network [13]. OpenSky Network es una organización sin ánimo de lucro basada en Suiza que se dedica a proveer acceso a datos de seguimiento y control aéreo. Siendo un organización sin ánimo de lucro, podemos acceder a la totalidad de la API de forma gratuita, y para el alcance del desarrollo de la aplicación, de forma prácticamente ilimitada.

Las peticiones que se le hacen a la API son del tipo *GET /states/all*. Sin embargo, de esta forma obtendríamos todas las aeronaves que se encuentran actualmente en activo al rededor del mundo. Iterar a traves de todas ellas para identificar la que se encuentra más cerca requeriría una gran cantidad de tiempo de procesamiento. Afortunadamente la API nos ofrece la opción de configurar la petición para añadir los siguientes parámetros: *lamin, lomin, lamax* y *lomin*. Estos hacen referencia a la latitud y longitud máxima y minima y están expresados en el sistema de referencia WGS84. De esta forma podemos delimitar, usando estos parametros como vertices, un cuadrado entorno a la ubicación obtenida mediante el GPS del dispositivo de grabación.

De esta forma obtendremos una lista de aeronaves en activo mucho mas reducida, ya que podemos delimitar el cuadrado con un área que se ajuste solamente a la ubicación dada. Si aun así la lista resultante contiene más de un elemento, es decir, más de una aeronave, se obtendrá la posición de cada una de ellas y se identificará cual se encuentra mas próxima al usuario.

A continuación, cuando ya hemos identificado la aeronave a la que se está grabando se obtiene un objeto JSON que contiene un array de dos dimensiones. En la primera dimensión encontramos todas las aeronaves en el área establecida y en la segunda dimensión encontramos varios datos sobre la aeronave en cuestión. Está información no es tan completa como podría ser la de otras APIs de pago, que incluyen datos como el aeropuerto de origen y de destino, o la aerolínea. Sin embargo, para el propósito de esta aplicación, OpenSky ofrece los datos que buscábamos. Entre ellos encontramos:

- **Icao24:** Este es un identificador de 24 bits de la aeronave. Esta expresado en código hexadecimal y cuenta con 6 dígitos. Estos códigos no pertenecen a una aeronave de

por vida, si no que son asignados por el registro del país donde opera. Debido a esto los códigos icao24 suelen cambiar pero no con demasiada frecuencia. A partir de este dato se puede obtener datos como la aerolínea o cuando se registro la aeronave.

- **Longitud y Latitud:** obtenemos las coordenadas del avión en el momento de la consulta. Estas coordenadas también están expresadas en WGS84.
- **Velocidad:** creemos que es importante conocer la velocidad de la aeronave ya que esto influye en gran medida en el ruido producido.
- **Altitud:** este dato también es crucial para poner el ruido medido en un contexto más adecuado y hacernos una idea real sobre el sonido que estaba generando la aeronave.

CAPÍTULO 5
Desarrollo de la Solución Propuesta

Este capitulo se comentan varios aspectos del desarrollo de la aplicación, como particularidades o problemas que surgieron.

5.1 Primer Sprint

El plan de trabajo en el punto 3.5 se explicó que el desarrollo se realizaría en tres sprint de 30 horas cada uno. Las tareas de cada Sprint se agruparon según su importancia y el esfuerzo estimado de cada una de ellas.

Sin embargo, en el primer Sprint hubo un gran desfase de tiempo con respecto a la estimación inicial. Esto se debe principalmente a dos causas que se detallan a continuación

5.1.1. Medición de ruido

Una de las partes fundamentales de la aplicación consiste en la medición del ruido que generan las aeronaves cercanas. Por ello, una correcta medición del ruido resulta crucial para este propósito.

La primera pregunta que surge al llevar a cabo esta tarea es como se mide el ruido. Dado que el ruido es algo subjetivo y dependiente del sujeto que lo este percibiendo, típicamente no utilizamos unidades absolutas como el metro o los litros, si no unidades que expresen diferencias entre dos valores.

La más conocida de estas unidades es el decibelio, más concretamente el decibelio en la escala de presión sonora (Db SPL de sus siglas en ingles Sound Pressure Level). Esta escala de progresión logarítmica tiene como 0 Db el umbral de audición humana, que equivale aproximadamente a 20 micropascales. Por ello para medir lo que denominamos ruido en nuestro día a día utilizamos esta diferencia en la presión sonora que hemos adaptado a nuestra propia experiencia auditiva.

Ahora que sabemos con que unidad se mide el ruido, debemos saber el cómo se mide. A grandes rasgos, la membrana de un micrófono responde a los cambios de presión en el aire generados por las ondas de sonido. El movimiento de dicha membrana causado por la presión es transformado en una señal eléctrica. (ver figura 5.1)

Esta conversión de presión a voltaje es clave, y viene determinada por la sensibilidad del micrófono. Dicha sensibilidad surge de diferencias en el amplificador u otros compo-

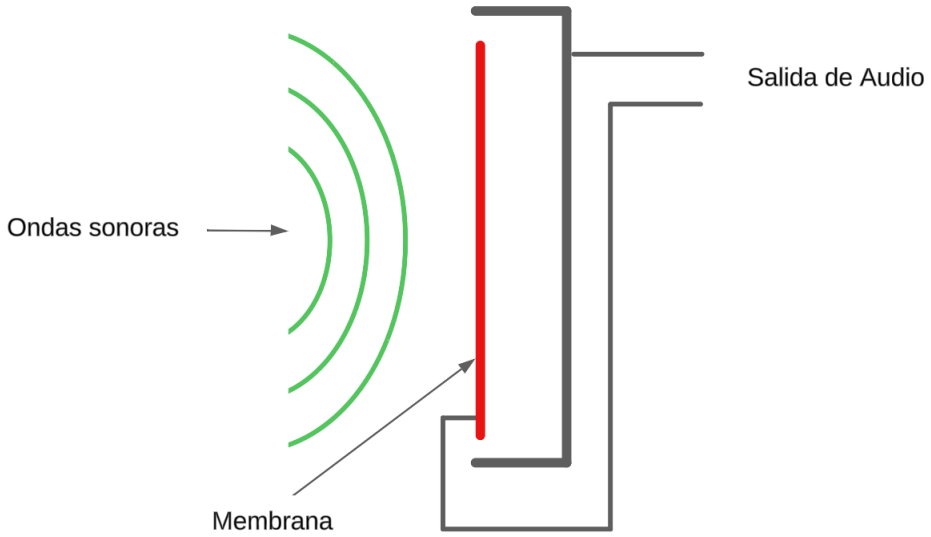

Figura 5.1: Esquema micrófono

nentes del micrófono y debe ser un valor conocido para poder rehacer la conversión de señal eléctrica a presión (Db SPL) y mostrarla al usuario.

Aquí es donde encontramos el primer problema. La sensibilidad del micrófono puede variar las lecturas de forma muy significativas y es un dato al cual no se puede acceder sin calibrar el dispositivo con herramientas externas. Es más, en dispositivos del mismo modelo, dispuestos a una misma distancia de una fuerte de ruido, obtendremos lecturas diferentes. Esto es debido a que la sensibilidad de un micrófono suele tener un rango de ± 3 Db[14]. Por lo tanto podemos determinar que hay una limitación de hardware a la hora de medir sonido comparado con equipos especializados en ello como los sonómetros[15].

Ante esto, valoramos distintas posibilidades y métodos para llevar a cabo la medición, lo cual nos llevo más tiempo de lo estimado. Finalmente decidimos que la solución que permitiría una mayor fidelidad en las lecturas a través de diferentes dispositivos seria una correlación entre mínimos y máximos, usando la clase nativa de Android MediaRecorder[16].

Para comenzar, las limitaciones del hardware de los teléfonos móviles nos obliga a imponer restricciones en el límite del ruido que la aplicación será capaz de registrar. En otras palabras hemos de acotar los Db SPL por el límite superior. Como ya hemos dicho más arriba el 0 en la escala Db SPL se ubica en 20 micropascales, este será nuestro límite inferior. En cuanto al límite superior decidimos fijarlo en 90 Db SPL, lo cual serían 0.6325 Pascales. Elegimos 90 Db basándonos en la información que podemos observar en la página WebTrak, comentada ya en el punto 2.1.2. En esta podemos ver que los micrófonos situados alrededor del aeropuerto de valencia registran valores entre 70 y 85 Db SPL. Por ello decidimos que un buen límite superior serían 90 Db.

Ahora pasando a la parte del código, realizamos la grabación de sonido usando MediaRecorder. Esta es una clase nativa de Android disponible desde la API de nivel 1 del sistema operativo. Esta clase posee el siguiente método: *getMaxAmplitude*, este método según su descripción en la documentación oficial, devuelve la amplitud máxima absoluta que fue muestreada desde la última llamada a este método. Esta amplitud de la que habla es la amplitud de la señal eléctrica que se obtiene tras recibir la presión por el micrófono del dispositivo. Esta señal se expresa en un numero de 16 bits (-32768, 32767), sin embargo al obtener su valor absoluto nos quedaría un rango de (0, 32767). En este caso 0 equivaldría a ningún cambio en la presión del micrófono y 32767 equivaldría a la presión límite y superiores que el micrófono es capaz de registrar.

```
private fun getAmbientSound(): Double{
    if (recorder != null){
        return recorder!!.getMaxAmplitude().toDouble()
    }
    return 0.0
}
```

Figura 5.2: Captura del método *getAmbientSound*

En el código de la figura 5.2, primero nos aseguramos que el objeto recorder, de tipo MediaRecorder, no sea nulo y tras ello obtenemos la amplitud de su señal.

```
private fun getDBSPL(): Double {
    val dbSPL: Double = 20 * log10( x: (getAmbientSound()/MAX_PRESSURE) / SOUND_REFERENCE)
    return if (dbSPL < 0) 0.0 else dbSPL
}
```

Figura 5.3: Captura del método *getDBSPL*

A continuación en el método de la figura 5.3, usamos la siguiente fórmula para obtener los Decibelios SPL a partir de la presión.

$$DbSPL = 20 * log10(p/p0)$$

Figura 5.4: Fórmula para la obtención de Db SPL a partir de la presión

En la fórmula, *p* equivaldría a la presión sonora y *p0* a la presión sonora de referencia, en este caso a 20 micropascales.

Por otra parte en el código encontramos dos constantes *MAX_PRESSURE* y *SOUND_REFERENCE*. *MAX_PRESSURE* equivale a 51805.5335968, este es el factor de conversión que obtenemos a partir de la amplitud máxima de la señal (32767) dividida por la presión atmosférica máxima que hemos establecido (0.6325 Pa). En cuanto a *SOUND_REFERENCE*, equivale a la presión de referencia en pascales que serían 0.00002 Pa. Este número es el límite inferior de la escala Db SPL.

```
private suspend fun measure(){
    var newNoise: Double
    while (state == RecorderState.RECORDING){
        delay( timeMillis: 1500)
        newNoise = getDBSPL()
        if (newNoise > maxNoise) maxNoise = newNoise
    }
    recorder = null
}
```

Figura 5.5: Captura del método *measure*

Por último, el proceso explicado anteriormente se llevaría a cabo durante toda la duración de la grabación a intervalos de 1.5 segundos. Esto se puede observar en el código de la figura 5.5. En dicho código, según se calculan los valores de ruido, estos se comparan con el ruido máximo registrado hasta el momento, si el nuevo ruido registrado es mayor, este pasa a ser el nuevo ruido máximo.

5.1.2. Nuevas Tecnologías

Durante la estimación del tiempo que requeriría este proyecto, fuimos conscientes de que usaríamos una gran cantidad de tecnologías y software con los cuales no poseíamos experiencia previa. Esto se reflejó en la estimación añadiendo un extra de tiempo a cada Sprint. Sin embargo, esta decisión no fue la correcta ya que solamente en el primer Sprint esta inexperiencia se tradujo en un desfase importante con respecto a la estimación inicial. De hecho, en el segundo Sprint la estimación de tiempo inicial fue ligeramente mayor que el tiempo requerido, y en el tercer Sprint el tiempo requerido fue considerablemente menor que la estimación inicial.
En retrospectiva resulta evidente que el tiempo de margen para aprender y habituarse al uso de estas nuevas tecnologías debería haberse aplicado de manera ponderada, siendo el primer Sprint al que se le otorgara una mayor cantidad de tiempo para familiarizarse con el nuevo entorno de desarrollo.

CAPÍTULO 6
Pruebas

En este capítulo se hablará de las pruebas realizadas a la aplicación y las conclusiones obtenidas a partir de las mismas.

6.1 Pruebas de usabilidad

Al final del desarrollo se llevaron a cabo las pruebas de usabilidad. Estas pruebas se llevaron a cabo en las inmediaciones del Aeropuerto de Manises, ya que es el punto más cercano en el cual se encuentran aviones a 300 metros de altitud o menos.

Estas pruebas se llevaron a cabo el día 4 de Junio en distintos puntos para asegurar su veracidad, dichos puntos se muestran a continuación.

Figura 6.1: Imagen del aeropuerto de valencia con las ubicaciones de las zonas de grabación

La aplicación se puso a prueba con tres dispositivos distintos. Dos de ellos son teléfonos móviles (OPPO A54s y Xiaomi 11 lite) y el ultimo una tableta (Samsung Galaxy Tab s7+), todos ellos de distintas marcas para que los resultados sean más completos y se pueda apreciar los márgenes de error en las grabaciones. Los resultados que se obtuvieron fueron los siguientes:

	TAP1094	PEI6FC	ANE001	Ruido Ambiental
Samsung Galaxy Tab S7+	68	58	67	29
OPPO A54s	68	63	70	33
Xiaomi 11 Lite	70	64	75	32

Tabla 6.1: Resultados en Db SPL en el punto de grabación 1

Las grabaciones en la localización número 1 se llevaron a cabo desde las 17:27 hasta las 17:43, hora en la que se realizó la última grabación. En este caso los aviones se aproximan desde el noroeste para realizar un aterrizaje. Dicha pista esta situada aproximadamente a unos 275 metros de distancia.

- **TAP1094:** la grabación a esta aeronave fue realizada a las 17:27 horas. El modelo del avión es ERJ-190 con origen en el aeropuerto de Lisboa.
- **PEI6FC:** la grabación a esta aeronave fue realizada a las 17:32 horas. El modelo del avión no se identificó pero aparentemente era una aeronave recreativa que realizó un vuelo de prueba ya que el origen y el destino fueron el aeropuerto de Valencia.
- **ANE001:** la grabación a esta aeronave fue realizada a las 17:43 horas. El modelo del avión es CRJ-1000 con origen del aeropuerto de Palma de Mallorca.

	PEI4FC	ANE8441	AEA4065	Ruido Ambiental
Samsung Galaxy Tab S7+	61	64	71	30
OPPO A54s	61	70	79	33
Xiaomi 11 Lite	63	73	76	36

Tabla 6.2: Resultados en Db SPL en el punto de grabación 2

Las grabaciones en la localización número 2 se llevaron a cabo desde las 15:53 hasta las 16:08, hora en la que se realizó la última grabación. En este caso los aviones se aproximan desde el noroeste para realizar un aterrizaje. Dicha pista esta situada aproximadamente a unos 200 metros de distancia.

- **PEI4FC:** la grabación a esta aeronave fue realizada a las 15:53 horas. El modelo del avión no se identificó pero aparentemente era una aeronave recreativa que realizó un vuelo de prueba ya que el origen y el destino fueron el aeropuerto de Valencia.
- **ANE8441:** la grabación a esta aeronave fue realizada a las 16:01 horas. El modelo del avión es CRJ-1000 con origen del aeropuerto de Menorca.
- **AEA4065:** la grabación a esta aeronave fue realizada a las 16:09 horas. El modelo del avión es Boeing 737-800 con origen del aeropuerto de Madrid.

	RYR53HX	ANE002	AEA4014	Ruido Ambiental
Samsung Galaxy Tab S7+	72	75	74	43
OPPO A54s	77	79	80	46
Xiaomi 11 Lite	80	82	79	47

Tabla 6.3: Resultados en Db SPL en el punto de grabación 3

Las grabaciones en la localización número 3 se llevaron a cabo desde las 16:25 hasta las 17:11, hora en la que se realizó la última grabación. En este caso los aviones despegan del aeropuerto desde el oeste hacia el este y sobrevuelan el punto de grabación entre 270 y 300 metros de altura. Esta localización esta situada sobre un puente que cruza la autovía A3, por lo que el ruido ambiental es considerable.

- **RYR53HX:** la grabación a esta aeronave fue realizada a las 16:25 horas. El modelo del avión es Boeing 737-800 con destino al aeropuerto de Treviso, Venecia.
- **ANE002:** la grabación a esta aeronave fue realizada a las 17:04 horas. El modelo del avión es CRJ-1000 con destino al aeropuerto de Palma de Mallorca.
- **AEA4014:** la grabación a esta aeronave fue realizada a las 17:11 horas. El modelo del avión es Boeing 737-800 con destino al aeropuerto de Palma de Mallorca.

6.1.1. Conclusiones

Como podemos observar en todas las tablas, hay una diferencia entre las mediciones realizadas sobre un avión entre los distintos dispositivos. Esto como se comentó en el capítulo 5, se debe a que los micrófonos cuentan con diferentes sensibilidades y aun con una misma sensibilidad, encontramos diferentes márgenes de error.

Sin embargo, estas diferencias son esperables y no muy significativas. Por otra parte comparando los resultados de las distintas tablas podemos ver algunas similitudes.
En los resultados de la tabla 6.1 y 6.2, que se realizaron en zonas donde se grababan a las aeronaves aterrizar vemos una gran similitud en las grabaciones, siendo en la tabla 6.2 mas elevadas debido a que nos encontrábamos más cerca de la pista de aterrizaje.

Por otro lado vemos que en la tabla 6.3, en la cual se grabaron a los aviones ganando altura, hay un ruido ambiente muy elevado pero también nos encontramos con que los resultados del ruido detectado en las grabaciones también son significativamente mayores. Esto podría indicar que los aviones durante los despegues son algo más ruidosos que durante los aterrizajes o bien que el hecho de que el ruido ambiente sea considerablemente más alto produce un mayor nivel en el ruido detectado durante la grabación.

CAPÍTULO 7
Conclusión

Este capítulo trata de tomar en retrospectiva el trabajo realizado, analizar lo que se ha aprendido, que cosas cambiaríamos del desarrollo y que conocimientos aprendidos a lo largo de la carrera hemos aplicado.

7.1 Objetivos

Los objetivos presentados al comienzo de este trabajo han sido cumplidos satisfactoriamente. Se ha completado el desarrollo de una plataforma, en este caso una aplicación móvil, que permite medir el ruido generado por aeronaves en tiempo real.

La interfaz es sencilla e intuitiva, esto se ha logrado reduciendo al máximo el número de pantallas y elementos en la misma. Como resultado, tenemos una interfaz fácil de usar y que no requiere apenas aprendizaje.

Las mediciones de ruido vienen dadas por el micrófono del dispositivo en el que se instala la app. Esto presenta algunas dudas sobre lo correctas que pueden ser estás mediciones, sin embargo en las pruebas realizadas los resultados fueron satisfactorios y garantiza en gran medida la veracidad de dichas grabaciones.

Otro de los retos a los que creíamos que nos enfrentaríamos al comienzo del desarrollo es la rapidez con la que la aplicación podría obtener la información de ubicación y de la aeronave que está siendo escuchada. Sin embargo, las dificultades eran totalmente infundadas. La estructura simple y la gestión de las llamadas y los datos de la API hacen que la aplicación sea realmente rápida y efectiva.

7.2 Análisis personal del trabajo realizado

Realizar un trabajo en solitario de tal envergadura ha sido un reto mucho mayor de lo esperado. La poca familiaridad con el entorno de desarrollo así como alguno de sus frameworks ha hecho del comienzo del desarrollo un proceso mucho más largo de lo estimado en un comienzo. Sin embargo, esta lenta curva de aprendizaje también es la causante de haber aprendido de manera sólida y profunda las bases de estas tecnologías. Esto me genera una seguridad y confianza, tanto a la hora de afrontar otros proyectos similares como para profundizar más y mejorar mi conocimiento sobre ellas.

Por otro lado, creo que la estimación optimista inicial ha sido uno de los mayores errores cometidos durante todo el trabajo. En concreto, la planificación de tiempo de los sprints de desarrollo
Dividí el desarrollo en tres sprints de una carga de trabajo y estimación de tiempo similar.

Cuando debí otorgarle a los primeros sprints un mayor tiempo para resolver problemas y dudas causados por el aprendizaje de las nuevas tecnologías.

7.3 Relación del trabajo desarrollado con los estudios cursados

Los conocimientos aprendidos durante toda la carrera, tanto durante las clases como en las practicas de empresa, han jugado un papel muy importante en el desarrollo de este trabajo.

En las etapas tempranas de análisis y diseño de la aplicación asignaturas como ISW y PSW fueron cruciales, ya que fueron de gran ayuda a la hora de llevar una metodología de trabajo bien definida y clara, lo que facilitó mucho el trabajo.
También cabe mencionar que tecnologías examinadas durante la carrera que creía que no serian puestas en practica durante este trabajo han resultado ser muy útiles. Un ejemplo de esto es el estudio de lenguajes de programación no orientados a objetos ni imperativos, como son Prolog o Haskell. La forma alternativa de pensar ha sido realmente útil para la elaboración de la interfaz gráfica con un framework que usa la programación declarativa.

También cabe mencionar las practicas de empresa que realicé en una empresa de software ERP. Participar en reuniones de SCRUM o comprobar el flujo de trabajo que llevan en una empresa dedicada al software ha sido realmente inspirador para la forma en la que he enfocado el proyecto.

En definitiva tener la posibilidad de ver y participar en una mediana empresa de software me aportó muchos conocimientos del "mundo real"que, unidos a todos los conocimientos teóricos y prácticos aprendidos en la carrera han dado como resultado este trabajo y aplicación con los que estoy satisfecho.

CAPÍTULO 8
Trabajos futuros

En este capítulo se habla de las posibles mejoras o funcionalidades añadidas que, debido a falta de tiempo o una complejidad excesiva, se dejaron fuera del desarrollo.
Sin embargo, eso no quita que, como con muchos productos de software reales, se pueda actualizar en un futuro se pueda actualizar para hacerlo más funcional y atractivo al usuario.

8.1 Reconocimiento inteligente de ruido

Uno de las funcionalidades que creemos que podría aportar más valor a la aplicación seria el reconocimiento inteligente de ruido. Actualmente, la aplicación detecta el ruido proveniente del ambiente sin asegurarse de que este ruido pertenece a una aeronave.

Por ello creemos que, entrenar un modelo mediante muestras de ruido generado por aviones, que ayude a identificar en que grado el ruido escuchado se trata de una aeronave, ayudaría a la robustez y la veracidad de la aplicación.

Sin embargo, esta funcionalidad requeriría una gran cantidad de tiempo tanto de aprendizaje como de diseño e implementación, por lo que se decidió excluirla de este proyecto.

8.2 Mapa con posición de las grabaciones

Otra de las funcionalidades que decidimos excluir por falta de tiempo es la de un mapa que muestre todas las posiciones donde se han realizado grabaciones con éxito. Este mapa seria una gran ayuda visual para los usuarios para observar donde se concentran las grabaciones que realizan así como el nivel de ruido que se alcanza en cada ubicación.

Pese a su evidente utilidad, el esfuerzo necesario para implementar de forma correcta esta característica supera con creces al de otras funcionalidades que sí se incluyeron en la aplicación. Esta fue una de las tareas planeadas que se quedaron en el backlog del proyecto.

8.3 Multiplataforma

Por último, también habría sido de gran ayuda para llegar a un mayor número de usuarios, el realizar la aplicación usando una herramienta para generación de código multiplataforma, como puede ser Flutter.

Sin embargo, esta idea se descartó ya que se prefirió la simplicidad del desarrollo nativo en Android. Otro de los factores que llevaron a tomar esta decisión fue la poca experiencia con estas herramientas, por lo que se decidió llevar un desarrollo con las tecnologías nativas de Android.

Bibliografía

[1] Annual Review 2023. International Air Transport Association. Consultado en https://www.iata.org/contentassets/c81222d96c9a4e0bb4ff6ced0126f0bb/annual-review-2023.pdf.

[2] WANG, S.S., GLIED, S., WILLIAMS, S., WILL, B. y MUENNIG, P.A., 2022 Impact of aeroplane noise on mental and Physical Health: A quasi-experimental analysis. *BMJ Open, vol. 12, no. 5. DOI 10.1136/bmjopen-2021-057209.*

[3] Regulation (EU) No 598/2014 of the European Parliament and of the Council on the establishment of rules and procedures with regard to the introduction of noise-related operating restrictions at Union airports within a Balanced Approach and repealing Directive 2002/30/EC 2014.

[4] BEŁEJ, M., CELLMER, R. y GŁUSZAK, M. 2020. The impact of airport proximity on single-family house prices—evidence from Poland. *Sustainability, vol. 12, no. 19, pp. 7928. DOI 10.3390/su12197928.*

[5] What is Scrum? Scrum.org. Consultado en https://www.scrum.org/resources/what-scrum-module..

[6] Explane, the app to Register Aviation Noise. Consultado en https://explane.org/.

[7] Red de Aeropuertos de España. Aena Consultado en https://www.aena.es/es/pasajeros/nuestros-aeropuertos.html.

[8] Webtrak Consultado en https://webtrak.emsbk.com/.

[9] Sueldo: Full stack developer. GlassDoor Consultado en https://www.glassdoor.es/Sueldos/full-stack-developer-sueldo-SRCH_KO0,20.htm.

[10] ANDERSON, C. The Model-View-ViewModel (MVVM) Design Pattern. *Pro Business Applications with Silverlight 5. S.l.: Apress, pp. 461-499.*.

[11] LOU, T. 2016. A Comparison of Android Native App Architecture MVC, MVP and MVVM. *thesis Aalto University*

[12] Mobile Operating System Market Share Worldwide. StatCounter Global Stats Consultado en https://gs.statcounter.com/os-market-share/mobile/worldwide.

[13] The OpenSky Network API. The OpenSky Network API 1.4.0 documentation. Consultado en https://openskynetwork.github.io/opensky-api/

[14] Especificaciones sensibilidad micrófonos MEMS. DB Products Ltd. Consultado en https://www.db.com.hk/mems-microphone.

[15] HONG, D. BCIT SCHOOL OF HEALTH SCIENCES, ENVIRONMENTAL HEALTH, HEACOCK, H. y SHAW, F. 2017. The practicality of using a smartphone as a sound level meter. *BCIT Environmental Public Health Journal, DOI 10.47339/ephj.2017.87.*

[16] MediaRecorder. Android Developers. Consultado en https://developer.android.com/reference/android/media/MediaRecorder

APÉNDICE A
Objetivos de Desarrollo Sostenible

En este anexo se comprueba en que medida este proyecto impacta en los Objetivos de Desarrollo Sostenible (ODS).

En 2015, la Organización de las Naciones Unidas aprobó la Agenda 2030 sobre el Desarrollo Sostenible. La idea de este es que los paises, empresas y sociedades por igual comiencen a tomar medidas para mejorar la vida de todos por igual.
La Agenda define 17 ODS de aplicación universal para impulsar el crecimiento económico, el compromiso con las necesidades sociales y la protección del medio ambiente.

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

Con este proyecto creemos que podríamos mejorar al menos 8 de los 17 ODS que se presentan en la tabla de arriba.
Algunos de los más importantes son:

- **Salud y Bienestar**: El objetivo de este trabajo es el desarrollo de una aplicación para detectar el ruido generado por aeronaves. Este ruido puede llegar a ser muy perjudicial física y mentalmente para las personas que están sometidas a él constantemente, como aquellos que viven o trabajan cerca de aeropuertos.
  Por esto mismo, creemos que esta aplicación sería de gran utilidad para que estas personas tengan una herramienta para proteger y garantizar su salud y bienestar.

- **Reducción de las Desigualdades**: A colación del ODS anterior, actualmente las viviendas que encontramos cerca de aeropuertos poseen un valor de mercado inferior al de otras zonas. Esto se debe principalmente al ruido al que están sometidas por parte de aeropuertos. Esto provoca que las personas con menos recursos se puedan ver empujadas a estas zonas.
- **Ciudades y Comunidades Sostenibles**: La mayoría de ciudades con una población considerable poseen uno o más aeropuertos. Esto significa que pueden someter a algunos ciudadanos a los problemas que conlleva vivir cerca de un aeropuerto. Por ello es crucial para la creación de comunidades sostenibles el cuidado y la precaución a la hora de edificar viviendas o sitios de trabajo cerca de aeropuertos.
- **Paz, Justicia e Instituciones Sólidas**: Actualmente existen políticas de ruido máximo soportable por parte de aeropuertos en comunidades próximas. Sin embargo dicho ruido puede ser difícil de comprobar en algunos ocasiones. Esta aplicación pretende otorgar una herramienta sencilla y gratuita para que las personas puedan asegurarse de que se están respetando la justicia y sus derechos.