Escola Universitària d'Enginyeria Tècnica
de Telecomunicació La Salle

Trabajo Final de Máster

Máster Universitario en Ingeniería de Telecomunicaciones

Noisitapp:
la app de Android para grabar y etiquetar muestras sonoras

Alumno
Marc Hermosilla Sánchez

Profesores Ponentes
Rosa Mª Alsina
Joan Claudi Socoró

ACTA DEL EXAMEN
DEL TRABAJO FINAL DE MÁSTER

Reunido el Tribunal calificador en la fecha indicada, el alumno

D. Marc Hermosilla Sánchez

Expuso su Trabajo Final de Máster, titulado:

Noisitapp: la app de Android para grabar y etiquetar muestras sonoras

Acabada la exposición y contestadas por parte del alumno las objeciones formuladas por los Sres. Miembros del tribunal, éste valoró dicho Trabajo con la calificación de

Barcelona,

VOCAL DEL TRIBUNAL VOCAL DEL TRIBUNAL

PRESIDENTE DEL TRIBUNAL

Resumen

La utilización de sistemas autónomos para el reconocimiento de patrones está en auge; dotar a un sistema de un aprendizaje autónomo, con el objetivo de automatizar tareas manuales parametrizando datos. Para poder construir estos sistemas, se requieren bases de datos representativas que sirvan para entrenar y evaluar estos algoritmos de reconocimiento de patrones. Con el propósito de recoger dichas bases de datos, Noisitapp permite grabar y etiquetar muestras sonoras de manera sencilla, poniendo a disposición de cualquier usuario con un dispositivo *Android*, la capacidad de registrar sonidos y subirlos a la nube. El desarrollo tiene en cuenta la aplicación en *Android*, la gestión de usuarios y registro de archivos en *Firebase*, así como la interacción con las diferentes APIs externas necesarias para su funcionamiento.

Todos los estudios han sido realizados en el Grupo de Recerca en Tecnologías Media (GTM) de La Salle Campus Barcelona, Universitat Ramon Llull.

Palabras clave: android, aplicación móvil, machine learning, base de datos, firebase

Abstract

The usage of autonomous sample classification systems for pattern recognition is increasing exponentially; provide a system with autonomous learning, in order to automate manual tasks parameterizing data. In order to build these systems, representative databases are required to train and evaluate these pattern recognition algorithms. In order to collect these databases, Noisitapp allows recording and labeling sound samples in a simple way, making available to any user with an Android device, the ability to record sounds and upload them to the cloud. The development takes into account the application in Android, the user management and file registration in Firebase, as well as the interaction with the different external APIs necessary for its operation.

All the studies have been realised with the Research Group on Media Technologies (GTM) of La Salle Campus Barcelona, Universitat Ramon Llull.

Keywords: android, mobile app, machine learning, database, firebase

Resum

La utilització de sistemes autònoms de classificació de mostres per al reconeixement de patrons està augmentant exponencialment; dotar un sistema d'un aprenentatge autònom, amb l'objectiu d'automatitzar tasques manuals parametritzant dades. Per a poder construir aquests sistemes, es requereixen bases de dades representatives que serveixin per entrenar i avaluar aquests algoritmes de reconeixement de patrons. Amb el propòsit de recollir les bases de dades, Noisitapp permet gravar i etiquetar mostres sonores de manera senzilla, posant a disposició de qualsevol usuari amb un dispositiu *Android*, la capacitat d'enregistrar sons i pujar-los al núvol. El desenvolupament té en compte l'aplicació en *Android*, la gestió d'usuaris i registre d'arxius en Firebase, així com la interacció amb les diferents APIs externes necessàries per al seu funcionament.

Tots els estudis han estat realitzats amb el Grup de Recerca en Tecnologies Mèdia (GTM) de La Salle Campus Barcelona, Universitat Ramon Llull.

Paraules clau: android, aplicació mòbil, machine learning, base de dades, firebase

Índice del contenido

1. Introducción 1
   1.1. Marco del trabajo 1
   1.2. Objetivos 2
   1.3. Fases del trabajo 3
   1.4. Estructura de la memoria 4
2. Estado del arte 6
   2.1. Aplicaciones móviles de grabación de muestras sonoras 6
      2.1.1. ASR Voice Recorder 6
   2.2. Aplicaciones móviles de grabación y reconocimiento del sonido 8
      2.2.1. Hush City: identificación de áreas tranquilas en la ciudad 8
      2.2.2. Shazam: reconocimiento de canciones 9
      2.2.3. BirdNET: identificación de cantos de aves 10
3. Fundamentación teórica 11
   3.1. Machine learning: aprendizaje automático y reconocimiento de patrones 11
      3.1.1. Aprendizaje supervisado 12
      3.1.2. Aprendizaje semi-supervisado 13
   3.2. Android 14
      3.2.1. Arquitectura Modelo - Vista - Controlador 15
      3.2.2. Interfaz de programación de aplicaciones 16
   3.3. Firebase 18
      3.3.1. Firebase Authentication 18
      3.3.2. Firebase Real Time Database 19
      3.3.3. Firebase Storage 19
      3.3.4. Firebase Crashlytics 19
4. Análisis de requisitos 21
   4.1. Requisitos funcionales 21
   4.2. Requisitos técnicos 24
      4.2.1. Gestión de usuarios 24
      4.2.2. Almacenamiento de las grabaciones 25
   4.3. Requisitos no funcionales 25
   4.4. Casos de uso 26
5. Diseño 27
   5.1. Perspectiva general: arquitectura del sistema 27

5.2. Aplicación Android 28
5.2.1. Modelo de clases 28
5.2.2. Vista de las pantallas 32
5.2.3. Controlador de tareas 43
5.3. Firebase: Real-time Database y Storage 48
5.3.1. Gestión de los usuarios y las grabaciones 48
5.3.2. Almacenamiento de los archivos de audio 51
6. Verificación del sistema, pruebas e incidencias 52
6.1. Verificaciones del sistema 52
6.2. Tareas de validación del sistema 52
6.3. Gestión de incidencias 53
7. Estudio del coste económico y temporal del trabajo 55
8. Conclusiones y líneas de futuro 56
8.1. Conclusiones 56
8.2. Líneas de futuro 57
9. Bibliografía 59
10. Anexo 61
10.1. Anexo A 61

Índice de figuras

Figura 1. Diagrama del flujo de fases del trabajo 3
Figura 2. Captura de pantalla de la aplicación móvil ASR Voice Recorder mientras se realiza una grabación. 7
Figura 3. Captura de pantalla del panel de control facilitado por ASR Voice Recorder. ... 7
Figura 4. Captura de pantalla de la aplicación móvil Hush City: tras introducir nuevos datos de un área tranquila, solicita una validación final del contenido aportado por el usuario para acabar el proceso. 8
Figura 5. Hush City Map: Mapa de las áreas tranquilas de Cataluña registradas en los servidores de Hush City. 9
Figura 6. Captura de pantalla de la aplicación móvil BirdNET durante la selección temporal de una grabación. 10
Figura 7. Diagrama de bloques de un sistema de aprendizaje supervisado [7] 12
Figura 8. Diagrama de bloques de un sistema de aprendizaje semi-supervisado [7] 13
Figura 9. Estadísticas de uso de los diferentes sistemas operativas desde el mes de julio de 2019 hasta el mes de abril de 2020. Extraído de StatCounter.com. 15
Figura 10. Diagrama de módulos de una arquitectura Modelo - Vista - Controlador 16
Figura 11. Máquina de estados de la clase MediaRecorder [8] 18
Figura 12. Diagrama de casos de uso para los roles de usuario y administrador. 26
Figura 13. Diagrama de bloques de la arquitectura del sistema Noisitapp 27
Figura 14. Estructura y variables de la clase User. 29
Figura 15. Estructura y variables de la clase Recording 30
Figura 16. Estructura y variables de la clase RecordFormat. 31
Figura 17. Estructura y variables de la clase MoreInformation 31
Figura 18. Esquema del flujo entre las vistas de Noisitapp 33
Figura 19. Pantalla de inicio de sesión de Noisitapp 34
Figura 20. Pantalla para el registro de usuario de Noisitapp. 35
Figura 21. Pantalla de restablecimiento de la contraseña de usuario de Noisitapp. 36
Figura 22. Pantalla del panel de control de Noisitapp, con las grabaciones del usuario. 37
Figura 23. Pantalla de consulta y edición de la grabación de Noisitapp. 39
Figura 24. Pantalla de registro para nuevas grabaciones de Noisitapp durante la grabación de un nuevo archivo de audio. 40
Figura 25. Pantalla emergente para la selección de etiquetas de una nueva grabación de audio. 41
Figura 26. Pantalla de consulta y edición de la información del usuario de Noisitapp. ... 42
Figura 27. Pantalla con información acerca de Noisitapp. 43
Figura 28. Configuración del objeto MediaRecorder para el formato de grabación. 45
Figura 29. Función checkIfUserExistAndFillData de la clase DashboardActivity. 47
Figura 30. Captura de pantalla de la consola de Firebase para la base de datos Real-time Database. 49
Figura 31. Captura de pantalla de la consola de Firebase para un usuario y grabación determinados. 50
Figura 32. Captura de pantalla del entorno web de Firebase Crashlytics. 53

Glosario

- ***Smartphone***: Teléfono móvil construido sobre una plataforma informática móvil, con capacidad para almacenar datos y realizar actividades asemejándose a una minicomputadora.
- ***Android***: Sistema operativo basado en *Linux* para dispositivos móviles.
- **Actividad**: Cada una de las pantallas de la aplicación móvil.
- **API** (*Application Programming Interace*): Interfaz de Programación de Aplicaciones. Son un conjunto de funciones, métodos o procedimientos que ofrece una biblioteca para ser utilizado por otro *software* como una capa de abstracción.
- ***Back-End***: También conocido como servidor, es un nodo que forma parte de una red y provee de servicios a nodos denominados clientes, o *Front-End*, que consumen del servidor como servicio remoto de otro ordenador.
- ***Script***: Archivo de órdenes, archivo de procesamiento por lotes o guion es un programa usualmente simple, que por lo regular se almacena en archivos de texto plano.
- ***Smart City*:** Área urbana que utiliza diferentes tipos de sensores electrónicos de IoT, *Internet of Things*, para recopilar datos y utilizar los conocimientos adquiridos a partir de esos datos para administrar activos, recursos y servicios, de manera eficiente y sostenible.
- **SQL**: *Structured Query Language*
- **JSON**: *JavaScript Object Notation*

1. Introducción

En este Trabajo Final de Máster se desarrolla la arquitectura necesaria para permitir, al usuario de una aplicación móvil, la capacidad de realizar grabaciones de audio de sonidos cotidianos de su entorno, etiquetarlas y subirlas a la nube mediante un dispositivo móvil.

En esta primera parte del documento se redacta la estructura que se encontrará el lector a lo largo de este. El presente ofrece una guía documental acerca del desarrollo llevado a cabo para Noisitapp, la *app* que permite grabar y etiquetar muestras sonoras.

1.1. Marco del trabajo

La contaminación acústica y su impacto se han convertido en una de las principales preocupaciones ambientales en las ciudades de hoy en día. El ruido provocado en diversas zonas de la ciudad afecta a la salud y bienestar de los ciudadanos, provocando, entre otros efectos, pérdida auditiva, insomnio, estrés e incluso baja de la productividad en casos de exposición excesiva [1]. En el mundo de las *Smart Cities*[1] éste es uno de los principales focos de atención y preocupación, situando el análisis acústico ambiental en el punto de mira de múltiples estudios de impacto socioeconómico.

Con la intención de analizar y estudiar diferentes entornos acústicos, la tecnología basa sus estudios y se apoya en la elaboración de sistemas autónomos de aprendizaje automático. La implementación de estos supone un avance tecnológico de gran importancia, debido a su amplio rango de posibles aplicaciones en entornos reales. En múltiples ocasiones, estos sistemas llegan a sustituir en buena parte las tareas manuales realizadas por el ser humano, pues consiguen reconocer patrones, pudiendo, así, predecir un resultado en base a una información de entrada al sistema. Para poder construir este tipo de sistemas, se necesitan bases de datos que representen lo mejor posible la realidad que se pretende parametrizar, con el objetivo final de obtener resultados altamente significativos que clasifiquen patrones de manera más eficaz. En el campo del reconocimiento de audio, la construcción de estos sistemas y su calidad quedan especialmente sujetas a la calidad de la base de datos que se use para entrenar el sistema. En definitiva, uno de los retos en cuanto a la elaboración de estos sistemas es recoger muestras sonoras representativas para entrenar, en segundo lugar, algoritmos de reconocimiento de patrones.

En los últimos años, el incremento del uso de la tecnología móvil favorece, de forma exponencial, la inclusión de ésta en la sociedad. Aunque hoy en día existen diversos sistemas operativos, *Android* sigue encabezando la lista de los más utilizados en el mercado, teniendo un incremento en usuarios y en mejoras de rendimiento en comparación con el resto de las tecnologías [2].

[1] Área urbana que utiliza diferentes tipos de sensores electrónicos de IoT, *Internet of Things*, para recopilar datos y utilizar los conocimientos adquiridos a partir de esos datos para administrar activos, recursos y servicios, de manera eficiente y sostenible.

La tecnología móvil *Android* supone, por tanto, uno de los recursos idóneos a la hora de dar una solución de carácter técnico a la problemática anteriormente expuesta.

1.2. Objetivos

Como se ha introducido anteriormente en este documento, el desarrollo argumentado en el trabajo pretende ofrecer una solución cómoda, sencilla y al alcance de la mayoría de la población para la recogida de datos sonoros en entornos cotidianos reales.

Las tareas que se han desarrollado persiguiendo los objetivos son las siguientes:

1. Estudiar y analizar las necesidades acerca de la construcción de sistemas autónomos de reconocimiento de patrones sonoros, más concretamente en la generación de bases de datos de paisajes sonoros especialmente entornos urbanos, pero también en entornos suburbanos y naturales, con el fin de conocer qué detalles son importantes registrar. En definitiva, existen diversas soluciones tecnológicas aplicables que podrían llegar a facilitar la recogida de muestras sonoras y, un correcto estudio de la situación actual permitiría escoger la tecnología más indicada a implementar, para llegar el objetivo general del trabajo.
2. Estudiar técnicas de implementación para *Android* y tecnología para dispositivos móviles. Escogiendo este sistema operativo como la tecnología indicada para la implementación, destacando la facilidad y extensa documentación que existe en la red, sirviendo a su vez de apoyo y soporte durante todo el trabajo. El hecho de que la aplicación esté disponible para *Android*, permite la recogida de muestras en cualquier momento, en cualquier lugar, mientras este disponga de conexión a Internet, por lo que permitirá añadir la participación ciudadana en la generación de la base de datos de los paisajes sonoros.
3. Analizar soluciones en la nube para la gestión de usuarios y el almacenamiento de archivos de audio, así como de etiquetas generados por éstos, y a su vez estudiar alternativas en cuanto a las posibles estructuras de bases de datos para permitir una correcta sintaxis y extracción de datos en un futuro.
4. Implementación técnica de una aplicación *Android* cumpliendo los siguientes requisitos: fácil de usar, disponible para el mayor número de dispositivos móviles y recogiendo el mayor porcentaje de datos posibles en la grabación de datos.
5. Obtención de una base de datos estructurada, sólida y debidamente etiquetada, que permita la posterior extracción y tratamiento de dichos archivos de audio. Uno de los grandes beneficios del trabajo presentado, es la construcción escalada de bases de datos de archivos de audio que, posteriormente, serán extraídos y tratados para computar en el entrenamiento de sistemas autónomos de reconocimiento de patrones.

Los objetivos del trabajo que se presentan, en conclusión, derivan de la necesidad vigente de facilitar, al mayor número de personas, la recogida de muestras para su posterior parametrización. Cabe destacar que, el hecho de que la solución sea implementada en formato aplicación independientemente de la calidad sonora que esta pueda ofrecer, permitirá recoger muestras en cualquier momento, lo que se traduce en que podrán registrarse grabaciones en momentos y entornos reales donde, quizás anteriormente, no podían realizarse. Gracias a las facilidades de implementación de la tecnología *Android* y

las integraciones disponibles que esta permite, se logra desarrollar un sistema sólido de grabación y etiquetado de muestras sonoras.

1.3. Fases del trabajo

A continuación, se detallan los puntos que se han seguido para desarrollar la aplicación, puntos básicos para un proyecto de implementación de estas características, representado en el diagrama del flujo de fases del trabajo de la Figura 1.

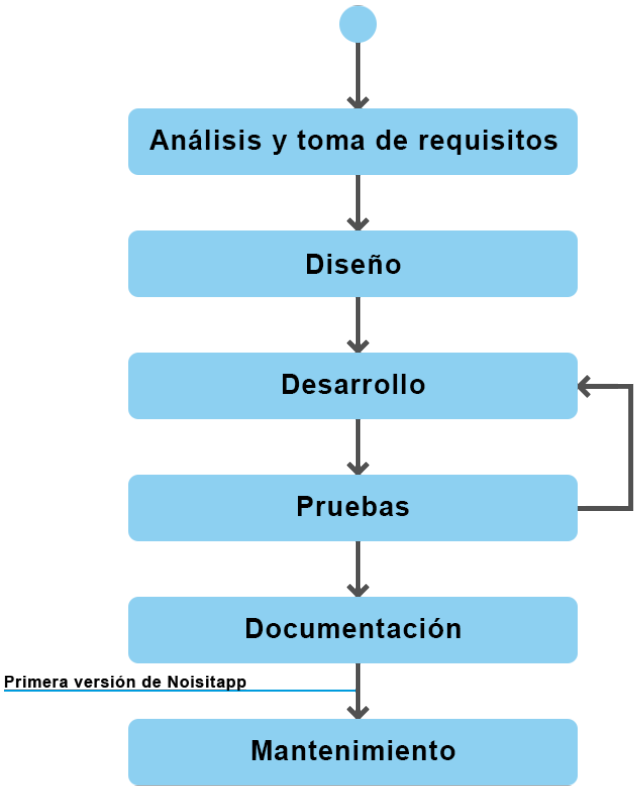

Figura 1. Diagrama del flujo de fases del trabajo

La primera etapa del proyecto es la de análisis y toma de requisitos técnicos. Conjuntamente con los tutores del proyecto, realizar un intercambio de ideas sobre las diferentes funcionalidades que debe tener la aplicación, documentando la lista de requerimientos posibles. En esta misma etapa, deben estudiarse las capacidades técnicas y la jerarquía de tareas que se llevarán a cabo, con el fin de obtener un mapa de ruta aproximado realista y una estimación temporal necesaria.

Tras recoger los requerimientos, se diseña toda la estructura del proyecto teniendo en cuenta los diferentes agentes implicados. En este caso, debe tenerse en cuenta el dispositivo móvil y su interacción con la nube gestionando los usuarios y almacenando los archivos de audio. En esta etapa, además, se realiza un diseño estético del comportamiento de la aplicación móvil, en cuanto a flujo de pantallas que tendrá.

A la vez que se van diseñando estructura y flujo de pantallas, se da comienzo al desarrollo técnico en tecnología *Android,* mediante el *software* de desarrollo *Android*. Dentro de este apartado, se realiza la integración de la aplicación móvil con el servidor en la nube de *Firebase.*

El siguiente paso está altamente relacionado con el desarrollo y se realiza de manera paralela, siguiendo el modelo de desarrollo evolutivo como método de desarrollo del *software* [3]. Al implementar una pequeña parte del código en *Android,* se definen unas pruebas con tal de validar el *software*, tratando así de garantizar el correcto funcionamiento de este cuando el usuario final utilice la aplicación.

Paralelamente al desarrollo de la aplicación, se redacta la documentación de este. Es de suma importancia que esta redacción quede correctamente documentada para cada *software* que se desarrolla, ya que de esta manera el mantenimiento del mismo programa es mucho más sencillo. Es muy probable que la primera persona que programa no acabe manteniendo el *software* en un futuro, de manera que es necesario tener una documentación actualizada argumentando el desarrollo con decisiones tomadas durante el proceso. A partir de este punto es cuando se constata que se dispone de la primera versión válida de Noisitapp.

Finalmente, la última etapa supone la de mantenimiento de la aplicación. Es imprescindible mantener el *software* una vez desarrollado y lanzada la primera versión de este, ya que tras el primer lanzamiento podrán surgir errores o, simplemente, querrán realizarse mejoras en el rendimiento del sistema, entre otras tareas.

1.4. Estructura de la memoria

En el Capítulo 1 se contextualiza el trabajo dentro del marco acústico ambiental en las ciudades contemporáneas, exponiendo así la problemática en su elaboración y solución propuesta a estos hechos, basada en tecnología móvil *Android*, enumerando también los diversos objetivos perseguidos.

En el Capítulo 2 se elabora una visión del estado del arte actual con la finalidad de introducir i contextualizar científicamente el trabajo, enumerando una serie de aplicaciones en los campos de reconocimiento de sonidos y registro de archivos de audio, las cuales también han servido como influencia e inspiración.

A lo largo del Capítulo 3 se presentan los diferentes sistemas que intervienen en el desarrollo de la aplicación, a modo de explicación teórica, que servirán de base para la implementación práctica que se presenta en el Capítulo 5.

En el Capítulo 4 se detallan los requisitos funcionales, técnicos y no funcionales a método de requerimientos básicos, a partir de los cuáles se resuelve en la implementación.

En el Capítulo 5 se presenta el desarrollo práctico realizado en el trabajo, así como los flujos de interacción entre los sistemas presentados en el Capítulo 3, las diferentes

actividades y los procesos realizables por la aplicación. A su vez, también se muestra la estructura de la base de datos.

En el Capítulo 6 se resumen las comprobaciones realizadas en la fase de pruebas del trabajo.

En el Capítulo 7 se realiza una aproximación del coste temporal y económico de lo que ha supuesto realizar el trabajo, detallando las tareas y el tiempo dedicado a cada una de ellas.

En el Capítulo 8 se extraen las principales conclusiones a las que se han llegado a lo largo del presente trabajo teórico-experimental para acabar planteando múltiples líneas de futuro para seguir con el desarrollo realizado.

En la bibliografía se encuentran todas las referencias a la información consultada y citada a lo largo de este documento.

Finalmente, en la sección de anexos, se realiza una extensa explicación de diversas cuestiones que se han tratado de manera breve durante el resto de los capítulos, así como parte de los *scripts* aportados al proyecto.

2. Estado del arte

En este apartado se realiza una breve explicación sobre el estado actual de estudios, aplicaciones e implementaciones que se han realizado al largo del tiempo en campos científicos y tecnológicos análogos, con el propósito de contextualizar este proyecto en el panorama de la ciencia y la tecnología contemporáneas.

A lo largo de la historia de la ingeniería y la ciencia, múltiples desarrolladores han implementado aplicaciones móviles con el objetivo de registrar muestras sonoras, pensadas para diversos casos de uso y diferentes funcionalidades. Tras analizar varias de ellas, pueden agruparse en dos ramas: aplicaciones móviles dedicadas a la grabación de muestras sonoras; y aplicaciones móviles dedicadas al reconocimiento del sonido en entornos reales.

2.1. Aplicaciones móviles de grabación de muestras sonoras

Una de las funciones principales de los dispositivos móviles de hoy en día es la función de grabación de sonido, captando la señal del micrófono del dispositivo. Esta grabadora de voz, en la mayoría de las ocasiones, viene ofrecida por el mismo dispositivo móvil de manera, permitiendo registrar y gestionar muestras sonoras. Debido a que estas están ideadas para realizar grabaciones sencillas, no ofrecen un rango suficiente de funcionalidades al usuario, de manera que son muchos los desarrolladores que ofrecen otras alternativas en este campo para darle solución, como es el caso de NLL Apps con su propuesta de aplicación móvil ASR *Voice Recorder*.

2.1.1. ASR Voice Recorder

Con más de un millón de descargas tras su lanzamiento, NLL Apps, desarrolladores ingleses de esta aplicación móvil para *Android*, permiten realizar grabaciones de alta calidad, con diversas funcionalidades adicionales a las que permiten las aplicaciones nativas de registro de notas de voz para móvil. Las grabaciones realizadas por este tipo de aplicaciones se almacenan en el propio dispositivo, por lo que imposibilita el hecho de acceder a ellas desde otro terminal móvil, pero en el caso de ASR *Voice Recorder* permite compartir las grabaciones mediante múltiples canales: desde por correo electrónico hasta por *Webhook*. Como se puede ver en la Figura 2, la aplicación móvil visualiza la onda captada por el micrófono.

Figura 2. Captura de pantalla de la aplicación móvil ASR Voice Recorder mientras se realiza una grabación.

En cuanto al formato de grabación, permite que sea el usuario quien escoja los detalles, pudiendo seleccionar entre AMR, FLAC, M4A, MP3 (marcado por defecto), OGG, WAV y WAV HQ. Curiosamente los desarrolladores de la aplicación no indican detalles acerca de la calidad de grabación en la descripción de esta.

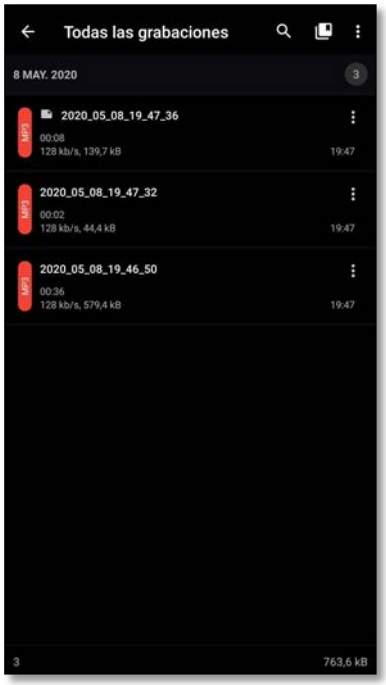

Figura 3. Captura de pantalla del panel de control facilitado por ASR Voice Recorder.

Una vez se realiza la grabación, ASR *Voice Recorder* permite gestionar todas las grabaciones desde un panel de control, como se puede apreciar en la Figura 3.

2.2. Aplicaciones móviles de grabación y reconocimiento del sonido

Como se ha comentado anteriormente, otra de las ramas en las que se pueden fragmentar las aplicaciones móviles dentro del campo de la grabación de audio, es el del reconocimiento y tratamiento del propio sonido captado por el dispositivo. Generalmente, este tipo de aplicaciones están pensadas con dos tipos de finalidades: extraer datos para ofrecerlos con objetivos comerciales, como es el caso de Shazam; o bien con fines científicos y de estudio de la tecnología en el campo de la acústica y reconocimiento de patrones, como es el caso de Hush City y BirdNET. Cabe destacar que, en el segundo caso, estas aplicaciones son desarrolladas, en gran parte, por proyectos universitarios.

2.2.1. Hush City: identificación de áreas tranquilas en la ciudad

Esta aplicación móvil gratuita de ciencia ciudadana, permite a las personas identificar y evaluar áreas tranquilas en las ciudades con el objetivo de crear un mapa de áreas abiertas [4]. El objetivo de la Dr. Antonella Radicchi y el equipo de investigadores de *Institute of City & Regional Planning del Technical University de Berlin*, es el de orientar planes y políticas para una vida más saludable, en respuesta a las cuestiones enmarcadas por las políticas medioambientales europeas.

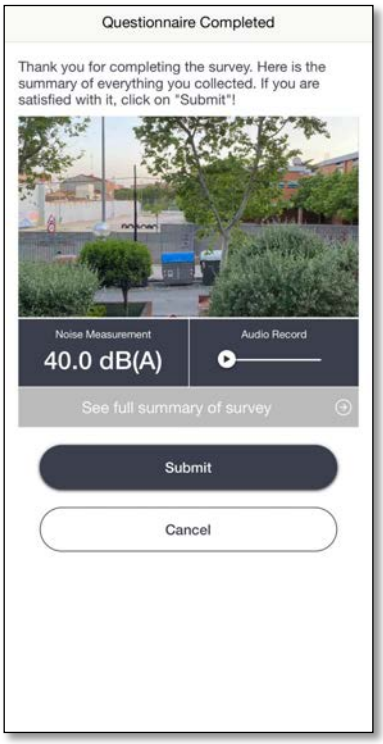

Figura 4. Captura de pantalla de la aplicación móvil Hush City: tras introducir nuevos datos de un área tranquila, solicita una validación final del contenido aportado por el usuario para acabar el proceso.

Para poder registrar un área tranquila, Hush City solicita al usuario que se facilite información de diferente tipo, además de una grabación con el dispositivo móvil de 30 segundos de duración mínima. Tras completar un formulario con preguntas cualitativas

acerca del contenido del audio, la aplicación pide la realización de una fotografía que servirá de información visual complementaria a los datos rellenados, como puede apreciarse en la Figura 4.

Además de aportar grabaciones de áreas tranquilas, el usuario también puede visualizar las entradas realizadas por el resto de los usuarios de esta aplicación. En el caso del territorio catalán, podemos encontrar una decena de registros tomados en diferentes provincias, como se puede ver en la Figura 5.

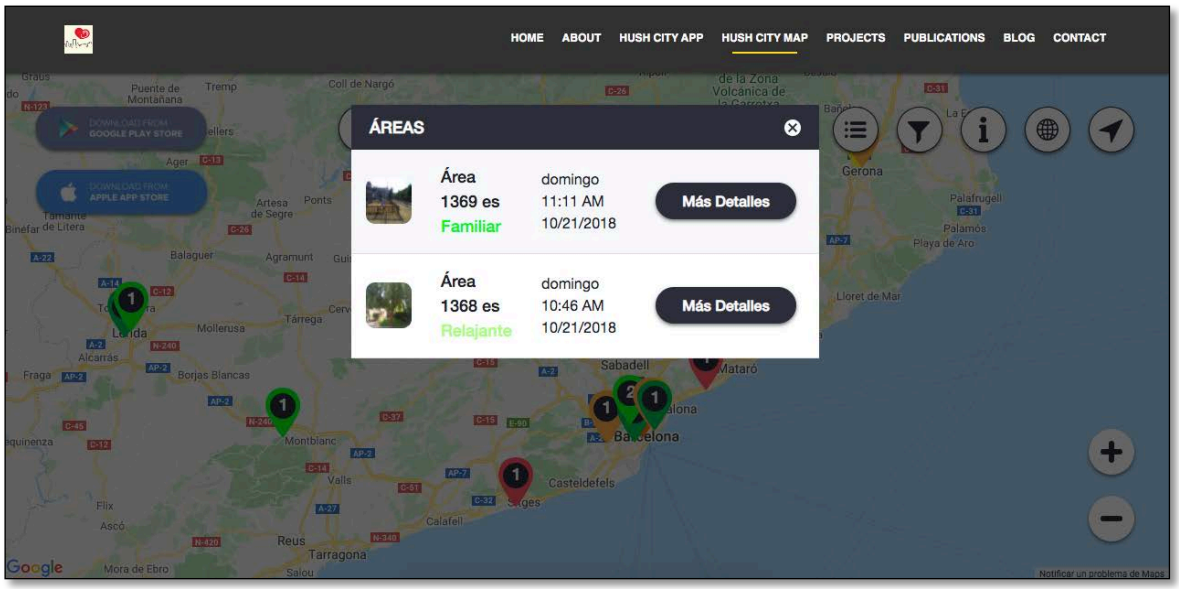

Figura 5. Hush City Map: Mapa de las áreas tranquilas de Cataluña registradas en los servidores de Hush City.

2.2.2. Shazam: reconocimiento de canciones

Uno de los campos de reconocimiento de sonidos es el caso de la detección de música, de canciones, como es el caso de la aplicación para dispositivos móviles *Shazam*. El programa compara un clip de audio que recibe a través del dispositivo móvil y lo compara con una base de datos, detectando la canción que está captando. La aplicación tiene un porcentaje de acierto cualitativamente alto.

Recientemente, la empresa que gestiona el *software, Shazam Entertainment Ltd*., asegura que, gracias a la extracción de información que obtienen de más de 100 millones de usuarios con más de 20 millones de búsquedas en un día, son capaces de predecir, con 33 días de antelación, la siguiente canción que ocupará el número 1 en la lista *Billboard,* que se configura con las canciones más vendidas de los Estados Unidos [5].

Shazam no es el único *software* en este campo; *Google Assistant, Soundhound* y *MusiXmatch*, entre otros, también son desarrolladores de la misma naturaleza, con tecnologías parecidas.

2.2.3. BirdNET: identificación de cantos de aves

Otro de los campos de desarrollo de este tipo de tecnologías es el de la biología, como es el caso de BirdNET, la aplicación móvil que permite reconocer cantos de aves. Esta aplicación, disponible para dispositivos *Android*, permite grabar la señal proveniente del micrófono del móvil y permite al usuario seleccionar una franja temporal visualizando el espectrograma de la grabación. BirdNET es un algoritmo desarrollado conjuntamente por investigadores de *The Cornell Lab of Ornithology y Chemnitz University of Technology*, con el propósito de comprender cómo, los ordenadores, pueden ayudar a reconocer sonidos que provienen de pájaros [6].

Gracias al potente análisis frecuencia que realiza la aplicación, esta procesa el rango seleccionado por el usuario informando, con destacable exactitud, de qué ave proviene el sonido que se ha grabado.

Figura 6. Captura de pantalla de la aplicación móvil BirdNET durante la selección temporal de una grabación.

Como se puede ver en la Figura 6, BirdNET permite seleccionar el inicio y final temporal del sonido que se requiere analizar, pudiendo verse el espectrograma frecuencial de la grabación realizada. Tras la grabación, se selecciona el fragmento y se analiza, ofreciendo la aplicación el resultado del reconocimiento.

3. Fundamentación teórica

En este apartado se detallan todos los conocimientos teóricos de carácter técnico involucrados en el desarrollo del algoritmo de la aplicación. Estos fundamentos teóricos que se presentarán a continuación sirven de base y han permitido el desarrollo satisfactorio del trabajo.

3.1. Machine learning: aprendizaje automático y reconocimiento de patrones

El aprendizaje automático es el subcampo de las ciencias de la computación y una rama de la inteligencia artificial en la cual el objetivo es desarrollar técnicas que permitan que los ordenadores aprendan. De forma más concreta, se trata de dotar a algoritmos con la capacidad de generalizar, de forma dinámica, diferentes comportamientos en función de unos datos de entrada correspondientes a uno o diversos eventos concretos, y que estos sean capaces de aprender y/o analizarlos de manera que en la salida que se obtenga información útil y necesaria para dar solución a problemas concretos.

Esta rama pretende estudiar el reconocimiento de patrones y el aprendizaje por parte de los computadores. El propósito final es encontrar una manera en la que los ordenadores sean capaces de aprender únicamente a partir de datos.

Existen diferentes tipos de algoritmos de aprendizaje automático [7]. Algunos sistemas intentan eliminar toda necesidad de intuición o conocimiento experto de los procesos de análisis de datos, mientras que otros intentan establecer un marco de coexistencia entre el experto y el ordenador. De todas maneras, la colaboración humana no puede ser reemplaza en su totalidad, ya que el diseñador del sistema debe especificar el trato de los datos de entrada y los métodos de manipulación y caracterización de estos. Los siguientes tipos de aprendizaje automático son basados en la creación de sistemas capaces de deducir una función de salida a partir de datos de entrada, la diferencia entre ellos recae en los datos de entrada.

- Aprendizaje supervisado: El sistema usa datos de entrada juntamente con la información de qué son esos datos, por ejemplo, en forma de etiquetas.
- Aprendizaje semi-supervisado: Entre el aprendizaje supervisado y el no supervisado, usa datos de entrada identificados y no identificados.
- Aprendizaje no supervisado: Al contrario que el supervisado, el sistema usa datos de entrada no identificados.

Realizado el preámbulo de las ciencias de *machine-learning* con los diferentes tipos de aprendizaje automático se hace más sencilla la contextualización de la materia que nos ocupa: permitir la captación de muestras sonoras y su etiquetación con el objetivo de entrenar un sistema de aprendizaje supervisado o semi-supervisado.

Para que una máquina sea capaz de reconocer patrones a partir de muestras no identificadas, se requiere un proceso previo de aprendizaje en base a un corpus de datos conocidos. Este proceso de aprendizaje se le conoce con el nombre de entrenamiento del sistema.

3.1.1. Aprendizaje supervisado

El aprendizaje supervisado consiste en el entrenamiento de un sistema de clasificación basándose en eventos transcurridos previamente. De estos eventos se extraen una serie de características, se parametrizan, tratando de que sean lo más representativas posibles de las clases de eventos que se quieren diferenciar. Con estas características se crea una base de datos, con la cual se generará uno o más modelos matemáticos en el proceso de entrenamiento del sistema. Una vez estos han sido generados, servirán para que, a partir de una muestra de entrada sin ninguna etiqueta ni indicio de la clase a la que pertenece, encontrar, mediante los modelos matemáticos generados, qué grupo representa mejor la nueva observación.

A continuación, en la Figura 7 se puede ver el diagrama de bloques de un sistema de detección por clasificación basado en el aprendizaje supervisado para un caso específico, en un contexto de clasificación binaria, es decir, de dos clases: *Road Traffic Noise,* o RTN, y *Anomalous Noise Event,* o ANE.

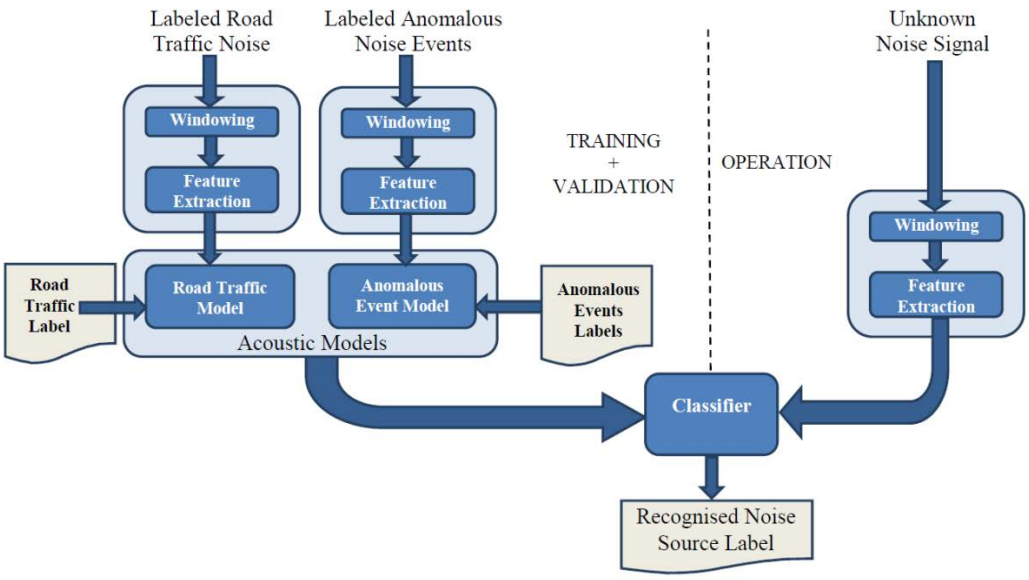

Figura 7. Diagrama de bloques de un sistema de aprendizaje supervisado [8]

Por tanto, el aprendizaje supervisado depende de entradas previamente etiquetadas e identificadas. La construcción de modelos representativos a los parámetros iniciales con la finalidad de crear una función capaz de predecir el valor correspondiente a cualquier objeto de entrada después de haber visto una serie de ejemplos, los datos de entrenamiento.

Este tipo de aprendizajes se dan de manera usual en sistemas de reconocimiento de voz, detección de correo no deseado y reconocimiento de escritura, entre otros.

3.1.2. Aprendizaje semi-supervisado

El aprendizaje semi-supervisado es una técnica de aprendizaje automático que se encuentra entre el aprendizaje no supervisado, el cual usa datos de entrenamiento sin etiquetas, y el aprendizaje supervisado, el cual consta de una grande cantidad de datos de entrenamiento debidamente etiquetados. En el caso particular, la base del aprendizaje, los datos etiquetados, es el esfuerzo. La máquina es capaz de aprender diversas situaciones en base a pruebas y errores. Aunque conoce los resultados desde el principio, no sabe cuáles son las mejores decisiones para llegar a obtenerlos. Lo que sucede es que el algoritmo asocia los patrones de éxito progresivamente, para repetirlos una y otra vez hasta perfeccionarlos y ser infalible. Ejemplos de este tipo de aprendizajes los encontramos en la navegación de un vehículo en piloto automático, en la toma de decisiones, etc.

A continuación, en la Figura 8 podemos ver el diagrama de bloques de un sistema de detección de eventos anómalos propuesto para ruidos basado en el aprendizaje semi-supervisado para un caso específico en un contexto de clasificación binario, es decir, de dos clases: *Road Traffic Noise,* o RTN, y *Anomalous Noise Event,* o ANE.

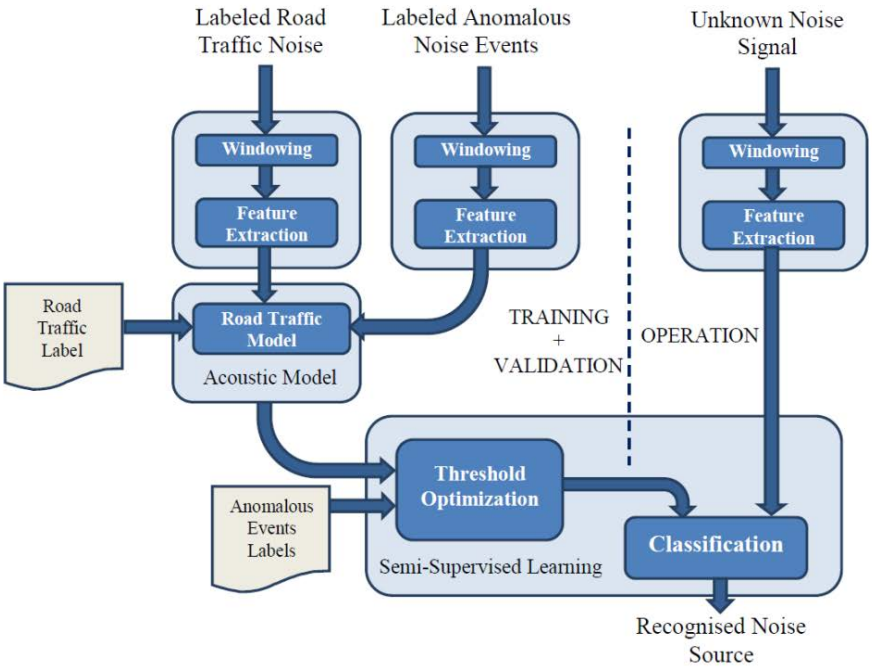

Figura 8. Diagrama de bloques de un sistema de aprendizaje semi-supervisado [8]

En el proceso de entrenamiento únicamente se construye el modelo de una de las clases a ser clasificadas, al contrato que en el método supervisado en el cual se construía un modelo por cada una de las clases.

Una de las grandes ventajas que tiene este método es el hecho de no tener que etiquetar una cantidad tan grande de datos, como en el caso del método supervisado, ya que este proceso puede llegar a ser muy costoso tanto a nivel económico como temporal.

Una vez expuestos los detalles teóricos acerca de los tipos de aprendizajes automáticos y de reconocimiento de patrones, cabe destacar la gran importancia que alberga la base

de datos de entrenamiento al sistema, tratando de que esta sea lo más representativa del contexto que se pretende analizar.

3.2. Android

Android es un sistema operativo basado en el *kernel* de *Linux*, diseñado principalmente para dispositivos móviles con pantalla táctil, tales como *smartphones* o tabletas. Inicialmente desarrollado por *Android Inc*, y respaldado económicamente por *Google*, que más tarde, en el año 2005 adquirió la empresa. Uno de los aspectos fundamentales del sistema operativo de *Android* fue su orientación a la multiplataforma, algo realmente novedoso, debido a que hace unos años, un sistema operativo se asociaba a un único dispositivo. Rápidamente esta característica hizo que *Android* alcanzara sus objetivos, convirtiéndose en el sistema operativo más utilizado.

Android no ha parado de evolucionar desde el lanzamiento de la primera versión. En febrero de 2009, *Google*, lanza su primera actualización 1.1 que se diferenciaba de la primera en poder adjuntar archivos en los mensajes. Echando la vista atrás y comparándolo con la versión actual, puede verse la gran evolución que ha desempeñado este sistema operativo, teniendo en cuenta el gran número de usuarios de los que dispone. Todas las versiones reciben un nombre de diferentes postres, en inglés, siguiendo un orden alfabético.

Lista de versiones de Android:

1. *Apple Pie* – versión 1.0
2. *Banana Bread* – versión 1.1
3. *Cupcake* – versión 1.5
4. *Donut* – versión 1.6
5. *Éclair* – versión 2.0/2.1
6. *Froyo* – versión 2.2
7. *Gingerbread* – versión 2.3
8. *Honeycomb* – versión 3.0/3.1/3.2
9. *Ice Cream Sandwich* – versión 4.0
10. *Jelly Bean* – versión 4.1/4.2/4.3
11. *Kit Kat* – versión 4.4
12. *Lollipop* – versión 5.0/5.1
13. *Marshmallow* – versión 6.0
14. *Nougat* – versión 7.0/7.1
15. *Oreo* – versión 8.0/8.1
16. *Pie* – versión 9.0

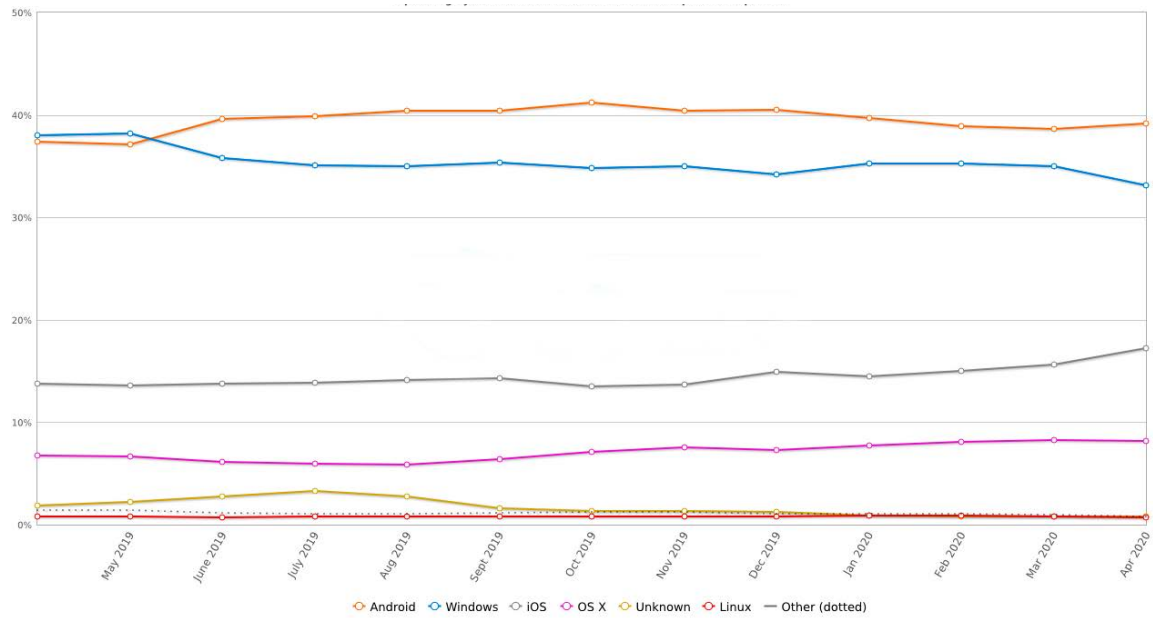

Figura 9. Estadísticas de uso de los diferentes sistemas operativas desde el mes de julio de 2019 hasta el mes de abril de 2020. Extraído de StatCounter.com

Como puede apreciarse en el gráfico de la Figura 9, *Android* se ha convertido en uno de los sistemas operativos de *smartphones* más populares del mundo y, de todos los usuarios de *smartphones* en todo el mundo, el 88% usa este sistema. Tras las 16 versiones, ya hace más de una década de su lanzamiento inicial, tiempo durante el cual, *Android* de *Google* ha recorrido un largo camino desde estar a punto de dejar de ofrecer mantenimiento al sistema hasta convertirse en uno de los mayores proyectos de *Google*.

Teniendo en cuenta los objetivos que pretende conseguir este trabajo, presentados en el punto 1.2, la tecnología *Android* supone el mejor escenario para implementar un sistema sencillo, pero eficaz, de captación y etiquetaje de muestras sonoras.

3.2.1. Arquitectura Modelo - Vista - Controlador

El patrón Modelo - Vista - Controlador, también conocido como modelo MVC, es un modelo de arquitectura que separa la aplicación en tres componentes lógicos principales: el modelo, la vista y el controlador. Cada uno de estos componentes son construidos para abarcar un aspecto específico del desarrollo de la aplicación. El modelo MVC, representado en la Figura 10, es uno los estándares más usados en la industria de desarrollo web y de aplicaciones, para crear proyectos extensibles y escalables.

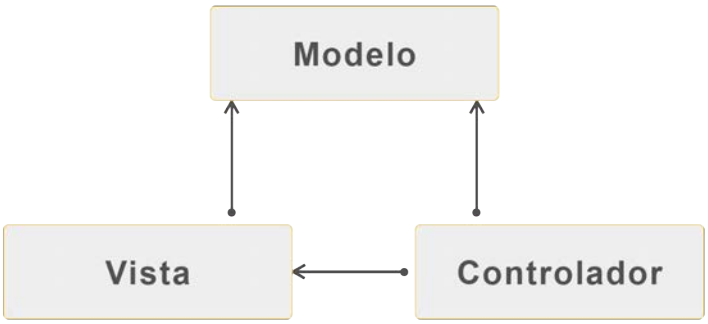

Figura 10. Diagrama de módulos de una arquitectura Modelo - Vista - Controlador

El componente Modelo corresponde a la lógica relacionada con los datos con los que trabaja el usuario. Esto puede representar los datos que se transfieren entre los componentes Vista y Controlador o cualquier otro dato relacionado con la lógica. Por ejemplo, un objetivo Usuario recuperará la información del cliente de la base de datos, la gestionará y actualizará los datos de la base de datos o la usará para presentarlos a la Vista.

El componente Vista se utiliza para la lógica de la interfaz de usuario de la aplicación. Por ejemplo, la vista Usuario incluirá todos los componentes de la interfaz de usuario, como cuadros de texto y menús desplegables, entre otros, con los que interactúa el usuario final.

Los controladores actúan como una interfaz entre los componentes Modelo y Vista para procesar toda la lógica y solicitudes entrantes, manipular los datos utilizando el componente Modelo e interactuar con las Vistas para representar el resultado final. Por ejemplo, el controlador cliente manejará todas las interacciones y entradas de la Vista del Usuario y actualizará la base de datos usando el Modelo del Usuario. Se usará el mismo controlador para ver los datos del Usuario.

3.2.2. Interfaz de programación de aplicaciones

Una interfaz de programación de aplicaciones, del inglés *Application Programming Interface* (API), es una interfaz informática que define las interacciones entre múltiples intermediarios del *software*. Define los tipos de llamadas o solicitudes que se pueden hacer, cómo hacerlas, los formatos de datos que se deben usar, las convenciones a seguir, etc. También puede proporcionar mecanismos de extensión para que los usuarios puedan ampliar la funcionalidad existente de varias maneras y en diferentes grados.

A continuación, se presentan las APIs utilizadas durante la implementación de Noisitapp, teoría a partir de la cual se ha desarrollado la comunicación entre los diferentes sistemas que componen la arquitectura.

3.2.2.1. **Google Maps API**

Google Maps es un servicio de mapeo web desarrollado por *Google*. Ofrece imágenes satelitales, fotografías aéreas, mapas de calles, vistas panorámicas interactivas de 360° de calles, condiciones de tráfico en tiempo real y planificación de rutas para viajar a pie, en automóvil, en bicicleta y en avión, o en transporte público. Además del producto más popular de este servicio, el mapa interactivo que ofrecen en su web pública, *Google* también permite la integración de sus servicios de mapeo para aplicaciones de terceros, mediante la implementación en contacto son su API. La API de *Google Maps*, ofrece la capacidad de integrar los servicios de mapeo virtuales en aplicaciones web o móvil, pudiendo ofrecer al usuario final, la posibilidad de visualizar los mapas, trazar rutas con diferentes puntos e incluso consultar información acerca de comercios y sitios registrados en la plataforma.

En el contexto del trabajo actual, la API de *Google Maps* permitirá registrar datos acerca de la ubicación del dispositivo móvil que registra la muestra sonora, de manera que se conseguirá tener información adicional al archivo de audio y sus etiquetas. Gracias a este servicio que ofrece *Google Maps*, se podrá contextualizar al usuario en un lugar físico del mapa.

3.2.2.2. **Clase MediaRecorder**

MediaRecorder es la clase de *Android* que permite registrar audio y vídeo. Mediante la creación de un objeto de este tipo en el algoritmo, se podrá configurar la calidad y formato del grabador, comenzar la grabación, pausarla y finalizarla, quedando registrado un archivo de audio con las características indicadas en su configuración. *MediaRecorder* está basada en la siguiente máquina de estados indicado en la Figura 11.

3.2.2.3. **Clase MediaPlayer**

La clase *MediaPlayer* proporciona los controles para reproducir medios, archivos de audio. *MediaPlayer* proporciona los controles de pausa, reproducción, detención y búsqueda, así como las propiedades de velocidad y reproducción automática que se aplican a todos los tipos de medios. También proporciona las propiedades de equilibrio, silencio y volumen que controlan las características de reproducción de audio.

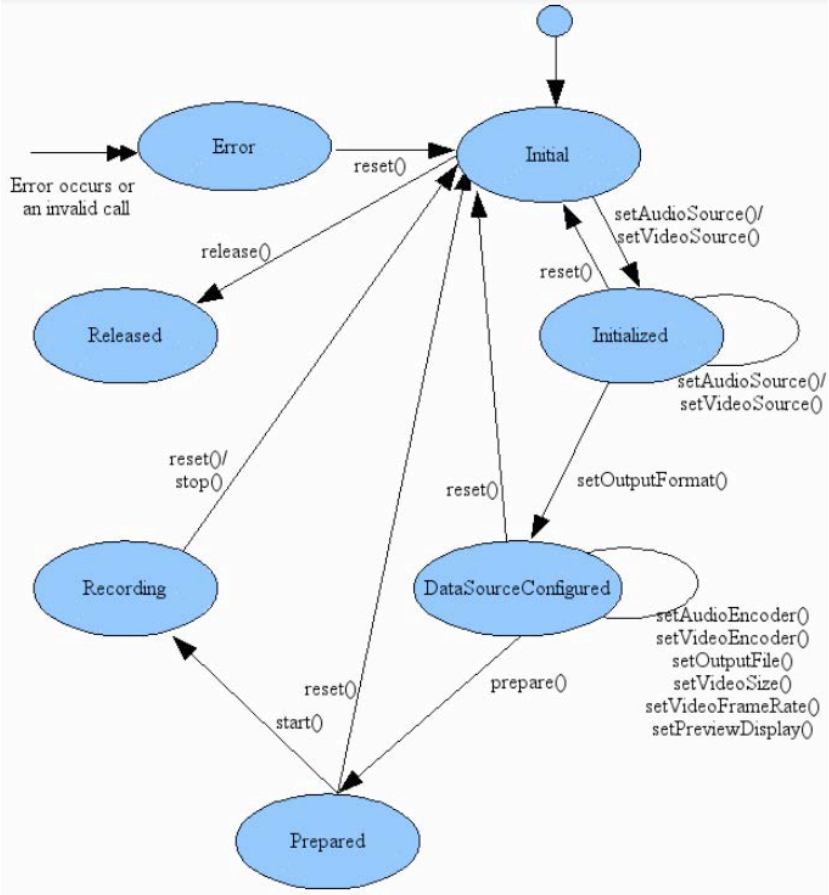

Figura 11. Máquina de estados de la clase MediaRecorder [9]

3.3. Firebase

Firebase es un conjunto de herramientas destinadas al uso de aplicaciones web y móviles, actuando como un servicio de *back-end*. Estas, cubren una gran parte de los servicios que los desarrolladores normalmente tendrían que construir ellos mismos, de manera que pueden centrarse los esfuerzos en la experiencia de la aplicación que se construye en sí.

Actualmente, la plataforma de *Firebase* tiene 19 productos, los cuales son usados en más de 1.5 millones de aplicaciones móviles, que cubren diferentes necesidades en aplicaciones móviles, tales como análisis, autenticación, bases de datos, configuración, almacenamiento de archivos, mensajería a la aplicación, entre otros. Todos sus servicios están alojados en la nube y permiten una escalabilidad instantánea con un esfuerzo ínfimo por parte del desarrollador.

3.3.1. Firebase Authentication

Firebase Authentication es un servicio que puede autenticar usuarios. Es compatible con los proveedores de inicio de sesión como *Facebook*, *GitHub*, *Twitter* y *Google*, así como con otros proveedores de servicios como *Google Play Games*, *Apple*, *Yahoo* y *Microsoft*. Además, incluye un sistema de administración de usuarios mediante el cual los desarrolladores pueden habilitar la autenticación de usuarios con correo electrónico y

contraseña de inicio de sesión almacenados con *Firebase*. Este, a su vez, permite la gestión con las sesiones de los usuarios de la aplicación, permitiendo gestionar las ediciones de los datos en inicio de sesión para los usuarios, como la dirección de correo electrónico o la contraseña.

3.3.2. Firebase Real Time Database

Firebase proporciona una base de datos en tiempo real y un *back-end* como servicio. Este proporciona, a los desarrolladores de aplicaciones, una API que permite que los datos de las aplicaciones se sincronicen entre los diferentes terminales clientes y se almacenen en la nube de *Firebase*. La compañía ofrece bibliotecas de clientes, o *SDKs*, que permiten la integración con aplicaciones *Android*, iOS, *JavaScript*, *Java*, *Objective-C*, *Swift* y *Node.js*. También se puede acceder a la base de datos a través de una API REST y enlaces para varios marcos de *JavaScript* como *AngularJS, React, Ember.js* y *Backbone.js*. Los desarrolladores que usan la base de datos en tiempo real pueden proteger sus datos mediante el uso de las reglas de seguridad aplicadas por el lado del servidor de la compañía.

Además, esta, permite la realización de estructuras de bases de datos *noSQL*, en formato *JSON*, con lo que pueden almacenarse información asociada a cada usuario como, por ejemplo, el nombre del usuario, el número de grabaciones que ha realizado, el rol que desempeña en la aplicación, o cualquier otro dato de texto.

3.3.3. Firebase Storage

Firebase Storage proporciona el almacenamiento para cargas y descargas de archivos, de manera segura, para aplicaciones de *Firebase*, independientemente de la calidad de la red, que puede utilizarse para depositar imágenes, audio, vídeo u otro contenido generado por el usuario.

En el contexto del presente trabajo, *Firebase Storage* permite almacenar los archivos de audio que, los diferentes usuarios de la aplicación deseen registrar en la nube, pudiendo disponer de ellos, posteriormente, para descargarlos y escucharlos.

3.3.4. Firebase Crashlytics

Firebase Crashlytics es uno de los servicios de *Firebase* dentro del apartado de estabilidad y gestión de errores. Este, proporciona informes de bloqueos, creando informes detallados de los problemas que ha tenido la aplicación en los dispositivos móviles de los usuarios de esta. Los errores se agrupan en listas de pila similares y se clasifican según la gravedad del impacto en los usuarios.

Además, este, ofrece información detallada acerca de la línea del algoritmo en la que entra en conflicto con el dispositivo móvil y almacena los detalles acerca del error, con el objetivo de facilitar la tarea de resolución de errores a desarrolladores.

En la fase de pruebas y corrección de errores de este proyecto, *Firebase Crashlytics* supone una herramienta indispensable, registrando de manera detallada el error obtenido dando indicaciones para su posterior resolución.

4. Análisis de requisitos

Tras exponer los fundamentos teóricos en los que se basa el trabajo, en este capítulo se presenta la fase de análisis. Como una de las partes iniciales de todo proyecto de desarrollo de *software,* la definición de los requisitos supone el punto de partida a la implementación. En este, se muestra una vista global de la arquitectura ideada para el sistema. En el siguiente capítulo, se tendrá en cuenta este catálogo de requisitos como base para el diseño de todos los aspectos de la aplicación y es por ello por lo que la fase de análisis es de suma importancia para el devenir del trabajo. Es en este capítulo donde deben asentarse las bases, a modo de cimientos, del proyecto y, a partir de las cuales se construirán el resto de los bloques. El proceso para seguir se basa, en primer lugar, en una correcta definición de requerimientos y en la elección de una metodología de desarrollo acorde al trabajo.

A continuación, se desglosan las funcionalidades y las características a modo de catálogo de requisitos, teniendo en cuenta tanto requerimientos funcionales como no funcionales.

El catálogo de requisitos es la especificación del comportamiento que se espera de cualquier proyecto de *software.* Estudiando aplicaciones similares, mencionadas en el apartado 2, se predefinen una serie de requisitos que se consideran indispensables para el trabajo. A continuación, se muestra una enumeración y breve descripción de los requisitos establecidos para el diseño y desarrollo de la aplicación.

4.1. Requisitos funcionales

En este apartado se describen las diferentes interacciones que tendrán los usuarios con la aplicación móvil.

I. Registro de usuarios

1) La aplicación debe permitir al usuario introducir sus datos en el formulario de registro.
2) El sistema validará los datos introducidos por el usuario: que la dirección de correo electrónico sea una dirección válida y que las contraseñas introducidas, además de coincidir, sea una contraseña segura.
3) En caso de error, el sistema mostrará un mensaje de error en el campo en cuestión, si el dato a introducir no cumple las condiciones especificadas del formulario.
4) En caso de que los datos introducidos sean correctos, el sistema se encargará de guardar la información en la base de datos, en la nube.
5) El sistema enviará un mensaje al correo electrónico del usuario para que este valide su dirección de correo electrónico, formalizando así el alta y registro del usuario.
6) La aplicación redirigirá al panel de control de grabaciones del usuario con la sesión iniciada.

II. Inicio de sesión de los usuarios

1) El usuario deberá indicar los datos de su dirección de correo electrónico y la contraseña asociada para poder ingresar en la aplicación.
2) El sistema validará los datos introducidos por el usuario, contrastando la información con la base de datos en la nube.
3) En caso de que el usuario no esté registrado, o la contraseña sea correcta, la aplicación mostrará un mensaje de error.
4) En caso de que inicie sesión satisfactoriamente, el sistema redirigirá al usuario al panel de control de grabaciones del usuario con la sesión iniciada.

III. Recuperación de la contraseña del usuario

1) En caso de que el usuario no recuerde la contraseña asociada a la dirección de correo, este podrá solicitar el restablecimiento de la clave de acceso.
2) El usuario deberá rellenar el formulario, introduciendo la dirección de correo electrónico.
3) Una vez introduzca los datos, el sistema enviará un mensaje al correo electrónico introducido con instrucciones para cambiar la contraseña.
4) En caso de que la dirección de correo electrónica no esté en la base de datos en la nube, el sistema mostrará un mensaje de error al usuario solicitando, de nuevo, que ingrese los datos para proceder con la recuperación.

IV. Cierre de sesión del usuario

1) El usuario podrá cerrar sesión, en cualquier momento, a través del menú, indicado en la parte superior derecha del panel de control mediante tres puntos.
2) Cuando el usuario haga clic, la aplicación cerrará la sesión activa y redirigirá al usuario a la pantalla de inicio de sesión.

V. Cuenta de usuario

1) El usuario podrá consultar la información sobre su cuenta a través de un enlace directo, en el menú, indicado en la parte superior derecha del panel de control mediante tres puntos.
2) El usuario podrá consultar y/o modificar los siguientes datos:
    i. Nombre del usuario: consulta y modificación.
    ii. Dirección de correo electrónico asociada: consulta y modificación.
    iii. Contraseña: modificación.
    iv. Última grabación realizada: consulta.
3) En caso de que el usuario modifique algún dato asociado a su cuenta, el sistema pedirá que introduzca la contraseña actual y, en caso de que esta sea correcta, actualizará la base de datos en la nube, en caso contrario mostrará un mensaje de error.

VI. Consultar información sobre Noisitapp

1) El usuario podrá consultar información sobre la aplicación en cuestión a través del menú, indicado en la parte superior derecha del panel de control mediante tres puntos.
2) Cuando el usuario haga clic, le redirigirá a una nueva pantalla con información acerca de Noisitapp.

VII. Grabaciones del usuario

1) El usuario tendrá disponible un panel de control con las grabaciones realizadas anteriormente. En esta lista, aparecerán las grabaciones activas, almacenadas en la base de datos, indicando la siguiente información para cada grabación:
    i. Nombre de la grabación
    ii. Fecha y hora de cuándo se realizó la grabación, en formato: día de la semana, mes, día del mes, HH:MM:SS en UTC, huso horario y año.
    iii. Duración de la grabación
    iv. Localización donde se realizó la grabación
2) En el caso de que el usuario haga clic en una de las grabaciones, el sistema lo redirigirá a la pantalla de edición de la grabación donde, además de consultar y modificar información acerca de la grabación, podrá escucharla.
3) En este mismo panel de control, el usuario tendrá disponible un acceso a realizar una nueva grabación, a través de un botón.

VIII. Editar una grabación

1) El usuario tendrá disponible una pantalla de edición para la grabación que desea consultar. Adicionalmente, el usuario podrá escuchar la grabación seleccionada, a través de un botón con el que podrá reproducir y parar el audio. En esta lista de información, el usuario podrá consultar y/o modificar los siguientes datos:
    i. Nombre de la grabación: consultar y modificación.
    ii. Fecha y hora de cuándo se realizó la grabación, en formato: día de la semana, mes, día del mes, HH:MM:SS en UTC, huso horario y año: consulta.
    iii. Duración de la grabación en formato HH:MM:SS: consulta.
    iv. Localización donde se realizó la grabación: consulta.
    v. Modelo del dispositivo móvil con el que se registró la grabación: consulta.
    vi. Etiquetas asociadas a la grabación: consulta.
2) En caso de que el usuario modifique el nombre de la grabación, el sistema guardará el dato en la base de datos de la nube.
3) En caso de que el usuario desee eliminar la grabación, podrá realizarlo mediante un botón.

IX. Realizar una nueva grabación

1) El usuario podrá realizar nuevas grabaciones, a través de un botón situado en la pantalla del panel de control.
2) El usuario deberá rellenar el formulario, compuesto por la siguiente información:
    i. Nombre de la grabación
    ii. Etiquetas de los sonidos que aparecen en la grabación: El usuario podrá seleccionar, de entre una lista, los sonidos que aparecen en su grabación.
    iii. Información adicional: El usuario podrá complementar los datos, además de las etiquetas, que aporta a la grabación, respondiendo de forma binaria a una serie de preguntas que el sistema plantea.
3) Pudiendo realizar la grabación antes, o después, de introducir la información anterior, hará clic en el botón de grabar y parará la captación de sonido cuando lo desee. Durante ese tiempo, el sistema deberá controlar el resto de las acciones disponibles sobre la grabación, tales como escuchar el audio o subirlo a la nube.
4) Una vez el usuario desee finalizar la grabación, podrá escucharla y/o subirla.
5) Si el usuario indica que desea escuchar la grabación, el sistema reproducirá el audio a través de los altavoces del dispositivo móvil.
6) Si el usuario indica que desea subir la grabación al almacenamiento de la nube, el sistema verificará que se hayan introducido, al menos, un nombre para la grabación y una etiqueta.
7) Si el usuario ha introducido todos los datos correctamente, el sistema actualizará la base de datos y almacenará la grabación en la nube.
8) Si el usuario ha de introducir erróneamente algún dato, o no lo ha introducido, el sistema indicará, mediante un mensaje de error, el dato erróneo o carente.
9) Tras actualizar la base de datos, la aplicación redirigirá al usuario al panel de control, donde podrá consultar todas las grabaciones y, de entre ellas, la que acaba de realizar.

4.2. Requisitos técnicos

En este apartado se describen el comportamiento y estructura esperados en cuanto a la gestión de los usuarios y el almacenamiento de las grabaciones.

4.2.1. Gestión de usuarios

La base de datos que guardará la información de todos los usuarios de la aplicación estará alojada en un servidor externo, en la nube. Deberá establecerse la definición de una estructura determinada, que permita la escalabilidad, y que almacene la siguiente información:

1) Nombre del usuario
2) Dirección de correo electrónico
3) Identificador único del usuario
4) Grabaciones:
    i. Nombre de la grabación

   ii. Fecha y hora de cuándo se realizó la grabación, en formato: día de la semana, mes, día del mes, HH:MM:SS en UTC, huso horario y año.
   iii. Duración de la grabación en formato HH:MM:SS.
   iv. Modelo del dispositivo móvil con el que se registró la grabación.
   v. Latitud, Longitud y dirección de la localización en la que se realizó la grabación
   vi. Lista de etiquetas de los sonidos que aparecen en la grabación.
   vii. Ruta al archivo de audio
   viii. Formato de la grabación: canales de audio, codificación, *bitrate, sampling rate*, etc.
   ix. Más información de interés con indicaciones subjetivas del usuario.
5) Aceptación de los términos y condiciones

4.2.2. Almacenamiento de las grabaciones

El almacenamiento de los archivos de audio, correspondientes a las grabaciones que realicen los usuarios, serán alojadas en un servidor externo, en la nube. Puesto que los archivos de audio de todos los usuarios están en la misma carpeta, la base de datos de usuarios tendrá que ser capaz discernir entre qué archivo pertenece a qué usuario. Se conseguirá esta diferenciación almacenando el nombre del archivo de audio junto al resto de información asociada a la grabación, en la base de datos de usuarios.

El nombre de los archivos de audio, almacenados en el servidor, deberán seguir una determinada taxonomía, que permita identificar, al menos, al usuario que realizó la grabación. De esta manera, la base de datos de usuarios permitirá identificar las diferentes grabaciones y viceversa.

4.3. Requisitos no funcionales

Además de los requisitos técnicos y funcionales, expuestos anteriormente, cabe destacar los siguientes requisitos, no funcionales, acerca de los requerimientos para la fase de análisis. Estos especifican criterios que juzgan operaciones del sistema en lugar de su comportamiento:

1) La implementación de los diferentes agentes necesarios para el desarrollo de la aplicación deberá asegurar su correcto funcionamiento, precisando tras cada funcionalidad implementada, de un control de errores exhaustivo.
2) El sistema deberá permitir cierta escalabilidad en cuanto a usuarios y grabaciones, además de ser susceptible a futuras modificaciones y ampliaciones.
3) El sistema deberá estar correctamente documentado en un documento, momento a partir del cual se cerrará la versión 1 de la aplicación móvil.
4) La usabilidad es un punto clave dentro de la interfaz gráfica, por tanto, se tendrá que diseñar para que el usuario se sienta cómodo y pueda usar la aplicación de manera sencilla e intuitiva.
5) La base de datos de usuarios y el almacenamiento de grabaciones deberá permitir, en el futuro, la extracción de dicha información.
6) La información sensible de los usuarios deberá estar cifrada.

4.4. Casos de uso

A continuación, se detallan todas las funcionalidades del sistema a través de diagramas de casos de uso, en relación con los requisitos funcionales expuestos en el apartado 4.1 de este mismo capítulo. Se muestra, en la Figura 12, un diagrama genérico de las diferentes agrupaciones dentro de las funcionalidades disponibles. El diagrama genérico muestra cada uno de los diagramas funcionales descritos anteriormente.

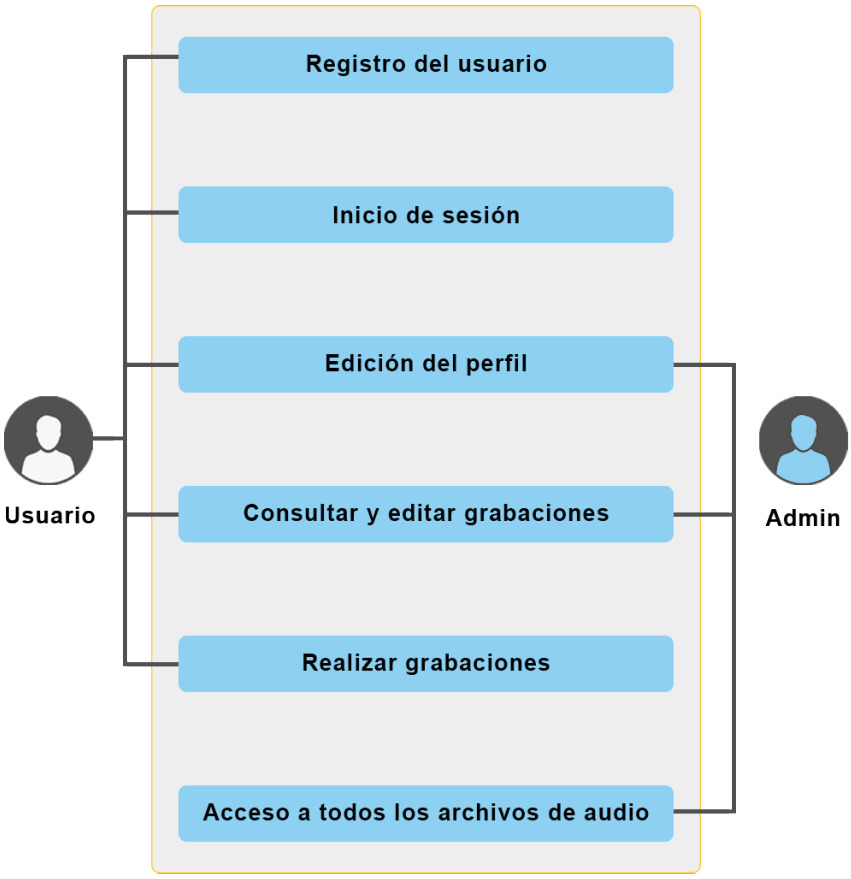

Figura 12. Diagrama de casos de uso para los roles de usuario y administrador

Como puede apreciarse en la figura anterior, se identifican dos roles que interactúan con el sistema: rol de usuario y rol de administrador. En el caso del rol de usuario, tendrá acceso a las tareas de registro, inicio de sesión, edición de su información, consulta, edición y grabación de archivos de audio. A su vez, el administrador, gestor de la base de datos y del mantenimiento del sistema, tendrá acceso a las dos bases de datos identificadas: base de datos de usuarios y almacenamiento de archivos de audio, con lo que tendrá permisos de control para la edición del perfil de los usuarios, consulta y edición de las grabaciones y acceso al repositorio completo de archivos de audio.

5. Diseño

En este capítulo se explicarán con detalle todas las fases del trabajo práctico realizado: desde la interacción entre los diversos sistemas implicados, las diferentes actividades realizables por la aplicación y los casos de uso que el usuario puede llegar a desempeñar en el manejo de la aplicación. En la parte final del capítulo, una vez se han presentado los diversos procesos que sigue el algoritmo, se presenta la estructura de la base de datos almacenada en la nube de *Firebase* para los usuarios y para el almacenamiento de archivos de audio.

5.1. Perspectiva general: arquitectura del sistema

La arquitectura de la aplicación móvil, como se presenta a continuación, se basa en un formato de cliente y servidor: constituyendo el servidor la base de datos y las API integradas con *Firebase*, y la aplicación móvil *Android* siendo el cliente. En la Figura 13 puede apreciarse el diagrama de bloques de la arquitectura implementada para el sistema, donde la gestión de usuarios y el almacenamiento de archivos se realiza en la nube, gracias a las posibilidades de comunicación con la API de *Firebase,* mientras que la aplicación de *Android,* Noisitapp, interactúa en comunicación constante con dichas bases de datos.

Figura 13. Diagrama de bloques de la arquitectura del sistema Noisitapp

La base de datos de Usuarios sirve para poder gestionar los datos de los diferentes usuarios dados de alta en el sistema, pudiendo realizar con ella labores de consulta y modificación de contenido. A su vez, la base de datos de archivos de audio permitirá almacenar las grabaciones que realicen los Usuarios. Cada usuario tendrá una correspondencia de cuáles son sus grabaciones dentro de la base de datos de archivos de audio.

La aplicación de *Android* deberá tener conexión a internet para poder intercambiar datos con la nube de *Firebase*, además, pedirá permiso al usuario para poder registrar la localización del dispositivo, acceder al micrófono y acceder a la memoria externa del teléfono móvil. La localización, junto con las posibilidades que ofrece la API de *Google Maps*, permitirá geolocalizar la latitud y la longitud desde donde se registra una grabación. El acceso al micrófono permitirá captar la señal entrante del propio micrófono del dispositivo móvil y, el acceso a la memoria externa permitirá almacenar la grabación en la memoria externa del teléfono móvil para subirla, posteriormente, a la base de datos de archivos.

Una vez expuesta la arquitectura del sistema al completo, se realizará una explicación de las diferentes partes desarrolladas; comenzando por la aplicación *Android*, documentando las diversas vistas y modelo de clases, para acabar detallando la estructura implementada en los servicios de la nube de *Firebase.*

5.2. Aplicación Android

Como se ha introducido anteriormente en este capítulo, la arquitectura implementa una estructura cliente-servidor, donde la aplicación *Android* desempeña la función de cliente. La estructura seguida para el desarrollo de esta se basa en el modelo MVC. Como documentado en el punto 3.2.1, la arquitectura Modelo - Vista - Controlador permite estructurar los diversos elementos en tres categorías: el modelo, la vista y el controlador, donde cada uno de ellos desempeña una función en el funcionamiento del *software.*

A continuación, se divide el diseño de la aplicación *Android* en las tres categorías, comenzando por el modelo, siguiendo por la vista y finalizando la explicación con el controlador.

5.2.1. Modelo de clases

El componente Modelo corresponde a la lógica relacionada con los datos con los que trabaja el usuario. Este puede representar los datos que se transfieren entre los componentes Vista y Controlador o cualquier otro dato relacionado con la lógica. En este trabajo, las clases que forman el Modelo son: la clase *User*, que implementa el objeto básico para controlar el usuario y sus variables; la clase *Recording*, que permitirá controlar las grabaciones realizadas por el usuario y las diferentes variables necesarias para almacenar un nuevo archivo de audio en la nube; la clase *RecordFormat*, siendo esta parte de la clase *Recording*, registra los formatos de grabación que se han utilizado para

registrar la nueva grabación; y la clase *MoreInformation*, que implementa la información adicional que el usuario desea registrar para nuevas grabaciones.

5.2.1.1. **User**

La clase User implementa el objeto básico para controlar el usuario que utiliza la aplicación. Esta clase es de las más importantes durante el desarrollo de la aplicación, puesto que esta permite gestionar la estructura de los usuarios en la base de datos también. Desde la primera pantalla de la aplicación, se solicita al usuario de la aplicación que ingrese con su cuenta, o bien se dé de alta, por lo que desde un inicio se precisa de la clase User para poder seguir navegando por el resto de las pantallas y funcionalidades de Noisitapp.

Como puede verse en la siguiente imagen, la estructura de la clase User es la siguiente:

```
public class User {
    private String uid, name, role, email;
    private Boolean terms, verified;
    private List<Recording> records;
```

Figura 14. Estructura y variables de la clase User.

Un usuario, controlado por la clase *User*, posee las siguientes variables:

- *Uid*: Esta variable de texto permite almacenar el identificador único generado por *Firebase* para registrar usuarios.
- *Name*: Esta variable de texto almacena el nombre del perfil del usuario.
- *Role*: Esta variable de texto almacena el rol del usuario, pudiendo este ser: amateur o experto. Pese a que, como podrá apreciarse durante el transcurso de este documento, esta variable no desempeña una función protagonista, en el futuro permitirá controlar la experiencia del usuario a través de la aplicación dependiendo del rol escogido.
- *Email*: Esta variable de texto guarda la dirección de correo electrónico del usuario, siendo necesario para identificarse con la base de datos de *Firebase.*
- *Terms*: Esta variable tipo booleano indicará si el usuario aceptó los términos y condiciones una vez se registró en la base de datos.
- *Verified*: Esta variable tipo booleano indicará si el usuario ha verificado su dirección de correo electrónico una vez se registró.
- *Records*: Esta variable es de las más importantes dentro de la clase *User*, puesto que almacena una lista dinámica de *Recording*, de grabaciones realizadas por el usuario.

5.2.1.2. **Recording**

La clase *Recording* implementa el objeto básico para controlar la grabación durante el uso de la aplicación y, junto a la clase *User*, son las clases más importantes ya que estas controlan el usuario y sus grabaciones. Esta clase controla los parámetros, tanto técnicos

como funcionales, de una grabación: desde el nombre de esta hasta las etiquetas indicadas por el usuario.

Como puede verse en la siguiente imagen, la estructura de la clase *Recording* es la siguiente:

Figura 15. Estructura y variables de la clase Recording

Una grabación, controlada por la clase *Recording*, posee las siguientes variables:

- *Str_name*: Esta variable de texto almacena el nombre de la grabación, indicado por el usuario.
- *Date*: Esta variable de texto almacena la fecha en la cual se realizó la grabación siguiendo el formato: día de la semana, mes, día del mes, HH:MM:SS en UTC, uso horario y año.
- *Duration*: Esta variable de texto almacena la duración de la grabación siguiendo el formato HH:MM:SS.
- *Path*: Esta variable de texto registra la ruta del archivo de audio en la base de archivos de audio.
- *mobileDevice*: Esta variable de texto registra el modelo del dispositivo móvil con el que se ha captado la grabación.
- *Address*: Esta variable de texto registra la dirección física desde donde se realizó la grabación.
- *Latitude* y *Longitude*: Estas variables de tipo *double*, almacenan la latitud y la longitud de las coordenadas del planeta desde donde se realizó la grabación. Servirán, una vez recogidas, para convertirlas en una dirección física que mostrar al usuario.
- *Labels*: Lista de variables de texto que permitirá almacenar las etiquetas seleccionadas por el usuario durante la grabación del archivo de audio. De manera dinámica, a medida que el usuario selecciona unas etiquetas u otras, esta elimina, añade o suprime los parámetros de la lista.
- *RecordFormat*: Esta variable de tipo *RecordFormat* almacena el formato de audio de la grabación.

5.2.1.3. **RecordFormat**

La clase *RecordFormat* permite gestionar el formato de grabación para las grabaciones que controla la aplicación. Esta supone una extensión de la clase *Recording*, que permite registrar los detalles técnicos en cuanto al formato del archivo de audio.

Como puede verse en la siguiente imagen, la estructura de la clase *RecordFormat* es la siguiente:

```
public class RecordFormat {
    private int audio_source;
    private int audio_format;
    private int audio_encoder;
    private int audio_channels;
    private int audio_encoding_bitrate;
    private int audio_encoding_samplingrate;
```

Figura 16. Estructura y variables de la clase RecordFormat.

El formato del audio, controlado por la clase *RecordFormat*, posee las siguientes variables:

- *Audio_source*: Esta variable de tipo entero, indica el dispositivo de captación usado para captar el audio, siendo por el micrófono del dispositivo móvil con el valor 1.
- *Audio_format*: Esta variable de tipo entero, indica el formato usado para captar el audio, teniendo por defecto el valor 2, formato de codificación de audio PCM 16 bits.
- *Audio_encoder*: Esta variable de tipo entero indica la codificación usada para registrar la grabación, siendo por defecto en formato AAC.
- *Audio_channels*: Esta variable de tipo entero indica los canales de los que dispone el archivo de audio, siendo por defecto un canal.
- *Audio_encoding_bitrate*: Esta variable de tipo entero indica la tasa de bits usados en la codificación del archivo de audio, siendo por defecto 128 kbps.
- *Audio_encoding_samplingrate*: Esta variable de tipo entero indica la frecuencia de muestreo del archivo de audio, siendo por defecto 48 kbps.

5.2.1.4. **MoreInformation**

La clase *MoreInformation* representa la información adicional con la que se complementa la información introducida con nuevas grabaciones. De la misma manera que *RecordFormat* recoge información relevante objetiva acerca del formato de audio usado en la grabación, la clase *MoreInformation* permite recoger información seleccionada por el usuario para una nueva grabación. Esta supone una versión inicial de lo que puede implementarse en un futuro, como se argumenta más adelante en este documento en el capítulo 8.2.

Como puede verse en la siguiente imagen, para esta versión introductoria e inicial de la clase, la estructura de *MoreInformation* es la siguiente:

```
public class MoreInformation {
    private boolean enabled;
    private boolean loud;
    private boolean annoying;
    private boolean like_to_hear;
```

Figura 17. Estructura y variables de la clase MoreInformation

La clase *MoreInformation* posee las siguientes variables:

- *Enabled*: Esta variable de tipo booleano indica si el usuario deseó introducir información adicional a la grabación y, por tanto, el resto de las variables de esta clase pueden representar información relevante introducida de manera expresa.
- *Loud*: Esta variable de tipo booleano indica si el usuario cataloga la grabación como ruidosa, teniendo el valor true en caso afirmativo y el valor false en caso contrario.
- *Annoying*: Esta variable de tipo booleano indica si el usuario cataloga la grabación como molesta, teniendo el valor true en caso afirmativo y el valor false en caso contrario.
- *Like_to_hear*: Esta variable de tipo booleano indica si el usuario cataloga la grabación como agradable, teniendo el valor true en caso afirmativo y el valor false en caso contrario.

El objetivo de esta clase reside en la complementación a las etiquetas seleccionadas por el usuario, teniendo información subjetiva acerca de sensaciones o emociones sobre la grabación registrada, siendo actualmente, en esta fase inicial, una información conceptual que permitirá un desarrollo más extenso en versiones posteriores.

5.2.2. Vista de las pantallas

El módulo Vista de la aplicación, como se ha fundamentado teóricamente en el capítulo 3.2.1, se utiliza para la lógica de la interfaz de usuario de la aplicación que permiten al usuario navegar a través de la aplicación móvil y realizar las diferentes tareas para las que está pensada la aplicación.

En la Figura 18 se ilustra el esquema general del flujo de vistas de la aplicación que puede realizar el usuario. A la izquierda, el inicio desde el cual el usuario comenzará a interaccionar con las vistas de Noisitapp. A partir de esta pantalla, el usuario podrá iniciar sesión y acceder al panel de control, registrar un usuario en caso de que no esté dado de alta o restablecer su contraseña. Una vez en el panel de control, momento a partir del cual el usuario ya está identificado y la aplicación posee sus credenciales para operar, se podrá acceder al menú desplegable en las vistas. En este punto, el usuario puede editar una de las grabaciones o bien crear nuevos archivos de audio.

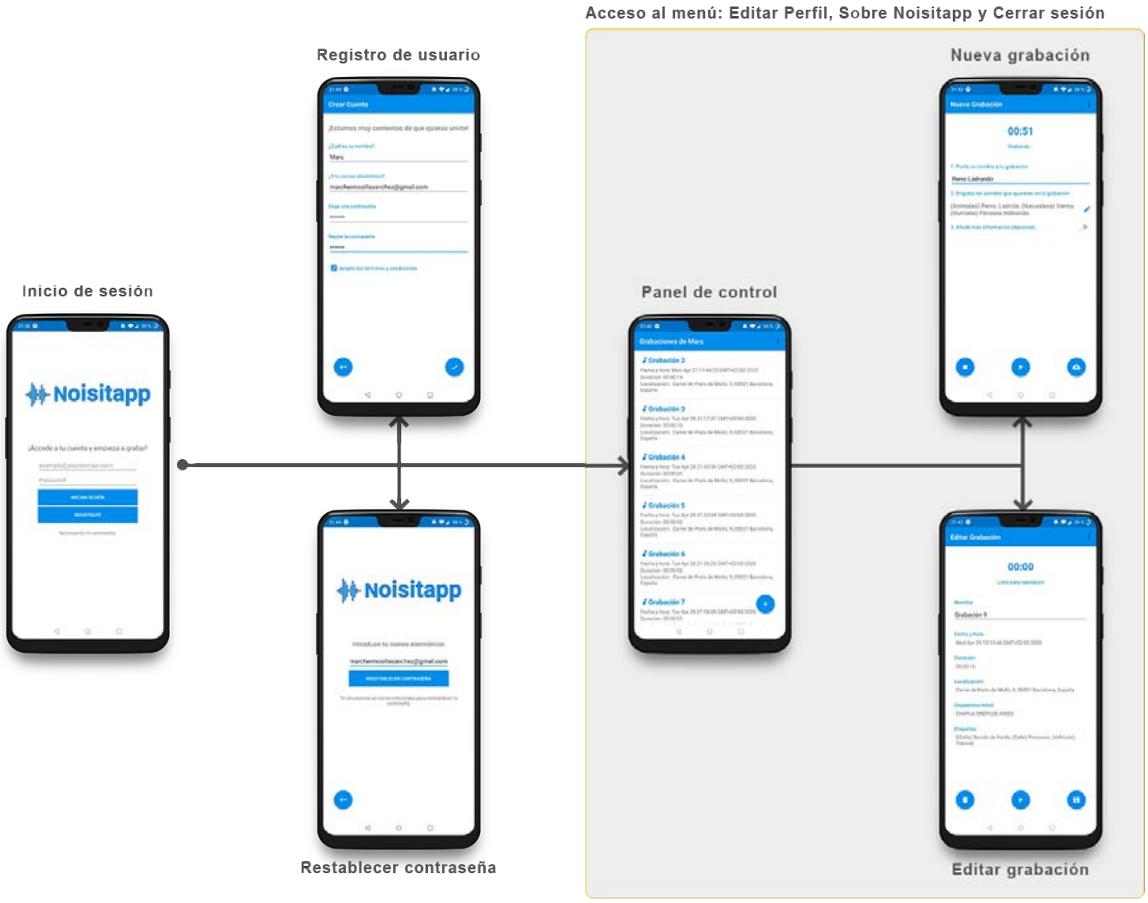

Figura 18. Esquema del flujo entre las vistas de Noisitapp

A continuación, se realiza una explicación sobre las diversas pantallas con las que se ha dotado a la aplicación realizando su documentación desde la primera pantalla con la que se encuentra el usuario al abrir la aplicación en orden temporal.

5.2.2.1. **Inicio de sesión**

La pantalla de inicio de sesión será la primera con la que se topará el usuario al iniciar la aplicación. Pese a que, antes de comenzar a usar Noisitapp, el usuario deba registrarse, esta supone el punto de partida al abrirla y, desde este punto, navegar al registro de usuario, al inicio de sesión o a la recuperación de la contraseña en caso de que el usuario no la recuerde.

La pantalla de inicio de sesión se ha implementado de la siguiente manera:

Figura 19. Pantalla de inicio de sesión de Noisitapp

Mediante el formulario de la pantalla de inicio de sesión, mostrada en la imagen anterior, el usuario podrá introducir las credenciales de acceso al sistema para iniciar sesión, navegar hacia el alta de usuario haciendo clic en el botón de registro, o bien recuperar la contraseña en caso de que el usuario no la recuerde.

Si el usuario desea iniciar sesión introduciendo su dirección y contraseña, el sistema comprobará que este existe en la base de datos y redirigirá al usuario al panel de control de grabaciones. En caso de que el sistema no encuentre al usuario, mostrará un mensaje de error.

5.2.2.2. **Registro**

La primera tarea que realizará un usuario de la aplicación será la de registro de sus credenciales para poder, posteriormente, hacer uso de las funcionalidades de Noisitapp. Esta pantalla está dotada de un formulario de datos para dar de alta un usuario en la base de datos, siendo estos necesarios para poder realizar la tarea satisfactoriamente.

La pantalla de registro se ha implementado de la siguiente manera:

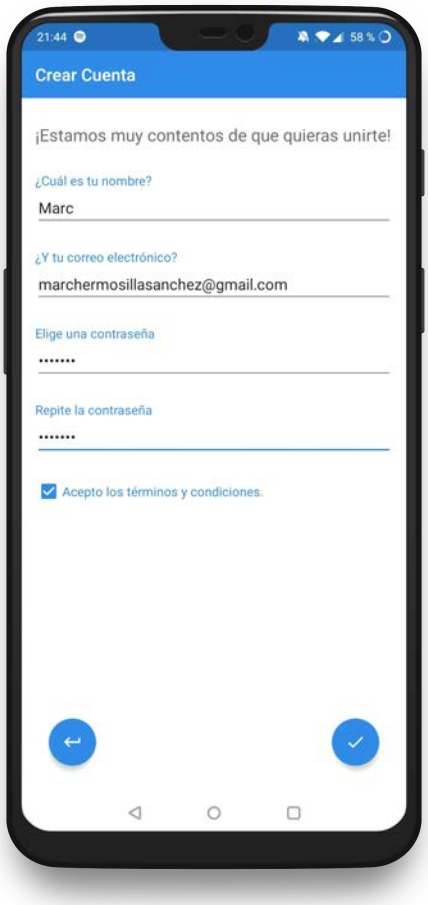

Figura 20. Pantalla para el registro de usuario de Noisitapp.

Mediante el formulario de registro de cuenta, mostrado en la imagen anterior, el usuario indicará la información solicitada por el sistema para tramitar el alta de este en la base de datos de *Firebase.*

Ambos campos de texto, que sirven para indicar el correo electrónico y la contraseña dados de alta en la base de datos, deben cumplir unos determinados requisitos: ser una dirección de correo electrónico válida para el email, y tener una seguridad determinada para la contraseña, debiendo ser como mínimo de seis caracteres.

5.2.2.3. **Recuperar la contraseña**

En el caso de que el usuario no recuerde la contraseña de su cuenta, se pone a disposición un formulario de recuperación de contraseña, donde la aplicación solicita el correo registrado para poder poner en marcha el restablecimiento de las credenciales.

La pantalla de recuperación de contraseña del usuario se ha implementado de la siguiente manera:

Figura 21. Pantalla de restablecimiento de la contraseña de usuario de Noisitapp.

Mediante el formulario de imagen anterior, se solicita al usuario que introduzca la dirección correo electrónico y, una vez hace clic en el botón de recuperación, se enviará un mensaje por correo electrónico con las instrucciones para poder recuperar la contraseña de la cuenta. En caso de que la información proporcionada sea errónea, el sistema mostrará un mensaje de error y, en caso de que esta sea correcta, enviará el mensaje y redirigirá al usuario a la pantalla de inicio de sesión.

5.2.2.4. **Panel de control**

Una vez el usuario ingresa en la aplicación con sus credenciales, ya sea tras el registro del usuario o este haya iniciado sesión, accede al panel de control con las grabaciones realizadas. Esta pantalla muestra los datos almacenados en la base de datos del usuario, conteniendo esta una lista dinámica de grabaciones que, dependiendo del número de archivos de audio que el usuario haya realizado, visualizará. En caso de que el usuario no haya registrado ninguna grabación todavía, esta pantalla no mostrará ninguna grabación si no que mostrará un mensaje animando al usuario a realizar su primera grabación.

Si el usuario desea realizar una nueva grabación, podrá acceder a la siguiente pantalla para captar un nuevo audio mediante el botón situado en la parte inferior derecha de la vista. Además, se ha implementado un menú, accesible a través del desplegable situado en el extremo derecha de la barra de herramientas, que permite acceder a las opciones de consulta y edición del perfil, consulta de la información acerca de Noisitapp y la capacidad de cierre de sesión del usuario.

El panel de control con las grabaciones realizadas por el usuario se ha implementado de la siguiente manera:

Figura 22. Pantalla del panel de control de Noisitapp, con las grabaciones del usuario.

Como se ha comentado anteriormente, se mostrarán tantas grabaciones como *Recordings* tenga el usuario registradas anteriormente y, por tanto, la lista de audios será dinámica y se modificará a medida que el usuario los añada o los elimine. Esta información proviene directamente de la base de datos del usuario, registrada en *Firebase.* En caso de que el usuario desee consultar y/o modificar la información sobre una grabación, podrá acceder a ello haciendo clic en una de las grabaciones. En cada una de las grabaciones, puede previsualizarse el nombre de la grabación, la fecha y hora de cuando se realizó el audio, la duración de este y la localización desde donde se captó.

Cabe destacar que, tras instalar la aplicación, la primera vez que el usuario acceda a esta pantalla, el sistema le solicitará permisos a la aplicación para poder acceder a la ubicación del dispositivo, leer y consultar datos de la memoria externa del teléfono móvil y acceder al micrófono para poder registrar muestras sonoras en pantallas siguientes. En caso de que el usuario deniegue el acceso a alguno de estos tres permisos, la aplicación los solicitará repetidamente en esta pantalla.

5.2.2.5. **Editar una grabación**

En caso de que el usuario decida consultar y/o modificar una de las grabaciones, podrá acceder a ello haciendo clic en esta en el panel de control. La acción anterior redirigirá al usuario a la pantalla de edición de la grabación. Una vez se accede a esta pantalla, el sistema descarga automáticamente el archivo de audio en la memoria externa del teléfono móvil, para que posteriormente pueda reproducirse si el usuario lo desea.

En la Figura 23, se presenta la implementación realizada para la pantalla de consulta y modificación de la información sobre una grabación concreta del usuario.

En esta pantalla, puede consultarse la siguiente información acerca de la grabación: nombre de la grabación, fecha y hora en la que se realizó la grabación, duración temporal del archivo de audio, localización desde la cual se registró la grabación, modelo del dispositivo móvil con el que se captó el sonido y etiquetas seleccionadas por el usuario al realizar la nueva grabación. Además, puede modificarse el nombre de la grabación borrando el nombre y haciendo clic en el botón de guardar, situado en la parte inferior derecha de la pantalla, sobrescribiendo dicha información en la base de datos de *Firebase*.

Figura 23. Pantalla de consulta y edición de la grabación de Noisitapp.

Mediante el botón de reproducción, el sistema reproducirá el archivo de audio al que hace referencia esta grabación. Si el usuario lo desea, haciendo clic en el botón eliminar, podrá borrar el registro de esta grabación de la base de datos. Tanto esta opción como la de guardado llevarán el usuario a la pantalla del panel de control de vuelta tras escribir la base de datos.

5.2.2.6. **Nueva grabación**

La pantalla de nueva grabación, accesible a partir del panel de control, permite al usuario registrar un nuevo archivo de audio y asociar información relevante acerca del mismo, en la base de datos de usuarios y en la base de datos de archivos de audio.

En la Figura 24, se presenta la implementación realizada para la pantalla de grabación de nuevos archivos.

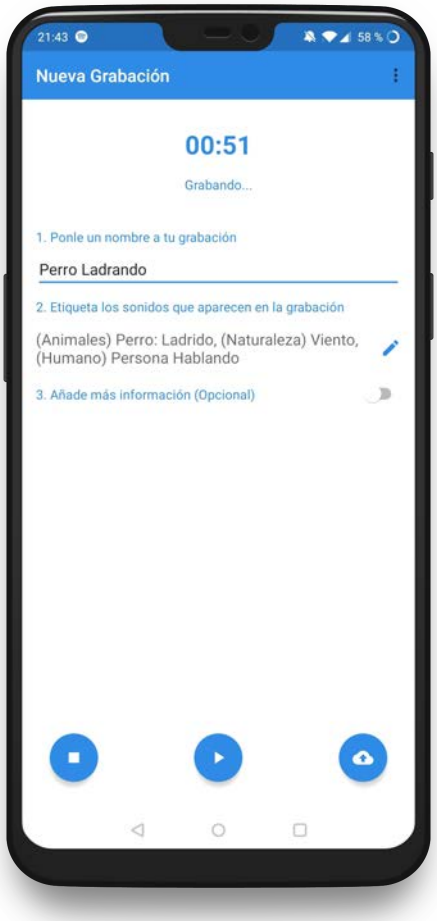

Figura 24. Pantalla de registro para nuevas grabaciones de Noisitapp durante la grabación de un nuevo archivo de audio.

Para registrar satisfactoriamente una nueva grabación, el usuario deberá haber dado permisos a la aplicación para acceder a la ubicación, a la lectura y escritura de la memoria externa y acceso al micrófono del dispositivo móvil. En caso de que no haya dado dichos permisos anteriormente, los solicitará repetidamente en esta pantalla. Una vez se han otorgado los permisos pertinentes, el usuario podrá proceder con el registro de una grabación. Esta vista la componen un campo de texto, para indicar el nombre de la grabación, un selector de etiquetas que abrirá una pantalla emergente y un selector para introducir información adicional a la grabación. Además, en la parte inferior, se muestran los botones de control de grabación, control de reproducción y subida de la grabación a la nube, de izquierda a derecha. El sistema permite que el usuario registre la grabación en el orden que le suponga más cómodo, pudiendo introducir primero nombre y etiquetas y realizando la grabación después, o viceversa.

La selección de etiquetas, que permitirá al usuario seleccionar cuáles son los sonidos que aparecen en el archivo de audio grabado, se ha implementado mediante una pantalla emergente adicional de la siguiente manera:

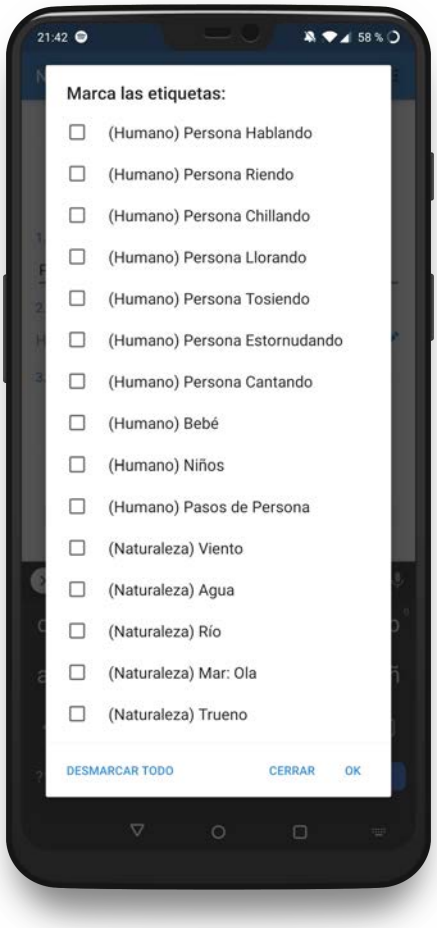

Figura 25. Pantalla emergente para la selección de etiquetas de una nueva grabación de audio.

Mediante la ventana emergente anterior, el usuario seleccionará los sonidos que aparecen en la grabación, pudiendo marcar y desmarcar las etiquetas sin necesidad de seguir un orden determinado. Una vez seleccionadas, estas aparecerán en la vista de nueva grabación, como puede apreciarse en la Figura 25. La taxonomía propuesta utilizada para las etiquetas de las muestras de audio se documenta debidamente en el apartado 5.3.1.1 de este mismo documento.

5.2.2.7. **Editar el perfil**

A partir del menú desplegable, el usuario podrá acceder a la pantalla de edición del perfil de usuario, mediante el cual podrá consultar y modificar la información asociada a sus credenciales de acceso y datos acerca de su cuenta en Noisitapp. Los datos que se muestran al usuario en esta pantalla son consultados a la base de datos de usuarios y, en caso de que se modifiquen los detalles, sobrescribiría la información en ella con las nuevas modificaciones.

La pantalla de consulta y modificación de la información del usuario es la siguiente:

Figura 26. Pantalla de consulta y edición de la información del usuario de Noisitapp.

Mediante el formulario en la pantalla de edición del perfil, el usuario puede consultar y modificar el nombre de la cuenta y la dirección de correo electrónico. Para poder realizar alguno de estos dos campos, deberá introducir la contraseña para verificar que es el usuario quien desea realizar dichas modificaciones. En el caso de que el usuario también desee modificar su contraseña, podrá realizarlo indicando la contraseña actual y la nueva. Además, el sistema muestra la última grabación registrada en la base de datos a modo informativo para el usuario.

5.2.2.8. **Sobre Noisitapp**

Esta pantalla, accesible también a partir del menú desplegable, muestra al usuario información acerca de la aplicación como tal, indicando datos relativos al motivo de desarrollo y mantenimiento de esta.

La pantalla con la información acerca Noisitapp es la siguiente:

Figura 27. Pantalla con información acerca de Noisitapp.

5.2.3. Controlador de tareas

Los controladores actúan como una interfaz entre los componentes Modelo y Vista para procesar toda la lógica y solicitudes entrantes, manipular los datos utilizando el componente Modelo e interactuar con las Vistas para representar el resultado final. En este apartado se documentarán las interacciones principales que realizan las *activities* en comunicación entre el Modelo y la Vista de la aplicación para poder realizar todas las tareas de gestión de usuarios y archivos de audio.

5.2.3.1. **Registro**

Para las tareas de registro, una vez el usuario rellena el formulario mostrado en la Figura 20, el sistema cogerá los datos de la vista para crear la clase *User*, que contendrá los parámetros básicos para solicitar a *Firebase* la creación de este.

En el caso de que *Firebase* tenga registrado al usuario en la base de datos, devolverá una excepción, que será mostrado en la pantalla como un mensaje de error, para informar al

usuario de que ya tiene una cuenta registrada. En caso contrario, si el usuario no existe en la base de datos, el controlador gestionará el alta en ella y devolverá un mensaje satisfactorio por pantalla. Tras completar el alta del usuario, el controlador llevará a la siguiente vista, al panel de control.

5.2.3.2. **Inicio de sesión**

Al iniciar sesión, de manera similar al registro, solicitará al usuario que rellene el formulario mostrado en la Figura 19, formado por dos campos de texto: dirección de correo electrónico y contraseña. Una vez el usuario haya completado los campos, el controlador cogerá la información y solicitará a *Firebase* la autenticación de las credenciales, devolviéndole este, si el usuario existe, el identificador de este. En caso de que el usuario no tenga una cuenta registrada en la base de datos, devolverá un mensaje de error al controlador para que este lo muestre por pantalla, informando así al usuario de la aplicación. Tras completar el inicio de sesión del usuario, el controlador llevará a la siguiente vista, al panel de control.

5.2.3.3. **Panel de control**

A diferencia del gestor de tareas del registro y del inicio de sesión, el controlador de tareas del panel de control deberá solicitar, en primer lugar, la información a la base de datos sobre el usuario que anteriormente inició sesión. Realizará tareas de lectura de la base de datos y esta le devolverá los detalles del usuario en cuestión, que se almacenarán en el objeto *User*.

Es entonces cuando el controlador deberá comunicar dicha información a la vista para que actualice la pantalla con las grabaciones, en la lista dinámica. En caso de que el controlador detecte que el usuario no ha realizado ninguna grabación anteriormente, mostrará un mensaje concreto. A medida que el usuario navegue por la pantalla, por la lista dinámica, el controlador irá asignando información a la lista para que esta pueda mostrar siempre un número determinado de grabaciones. Además, se controlará si el usuario hace clic en alguna de las grabaciones, en el botón de nueva grabación o en alguna opción del menú, para cargar la vista seleccionada. Pueden verse las alternativas de navegación contempladas en el diagrama de flujos de la Figura 18.

5.2.3.4. **Editar una grabación**

Una vez el usuario selecciona una de las grabaciones del panel de control, el controlador del panel indicará qué grabación ha sido seleccionada, y pasará esta información al controlador de la siguiente pantalla. Una vez conocida la grabación seleccionada, el controlador solicitará su información a la nube para poder mostrarla de pantalla, como puede verse, por ejemplo, en la Figura 23.

El usuario, por tanto, podrá visualizar la información sobre esa grabación y, en el caso de que quiera modificar el nombre o eliminar la grabación, el controlador sobrescribirá el campo de texto introducido con el nombre de la grabación, alojado en la base de datos, o

eliminará la referencia de la lista de grabaciones del usuario, respectivamente. Una vez finalizada alguna de las tareas anteriores, el controlador ordenará a la vista que finalice para redirigir al usuario de nuevo al panel de control.

5.2.3.5. **Nueva grabación**

En la pantalla de nueva grabación, el controlador de tareas deberá controlar, en primer lugar, que el usuario haya permitido el acceso a la ubicación, el acceso a leer y grabar de la memoria externa del dispositivo móvil y al micrófono. En caso de que el usuario no haya aceptado los permisos, los solicitará de nuevo.

Esta vista, como puede verse en la Figura 24, controla un editor de texto para que el usuario pueda indicar el nombre de la grabación deseado, un selector de etiquetas que al seleccionarse abrirá una pantalla emergente como la de la Figura 25, un interruptor para mostrar unos campos determinados y tres botones: control de grabación, control de reproducción y control de subida del archivo de audio a la nube de *Firebase*. Sin necesidad de seguir un orden concreto, el usuario podrá rellenar el formulario -nombre de la grabación, etiquetas asociadas y accionar el interruptor para introducir información adicional- antes, durante o después de realizar la grabación. Cuando el usuario hace clic en el botón de grabación, el controlador de tareas configurará el objeto *MediaRecorder,* presentado en el apartado 3.2.2.2, el cual captará el sonido del micrófono del dispositivo móvil, hasta volver a hacer clic al mismo botón para finalizar la grabación. En este punto, el usuario podrá escuchar el audio mediante el botón de control de reproducción, el cuál configurará la escucha mediante el objeto *MediaPlayer*. Una vez el usuario desea subir la grabación, el controlador de tareas comprobará con la vista que se ha rellenado toda la información necesaria -nombre y etiquetas- para gestionar la subida y redirigir al usuario al panel de control. En caso de que la información no esté correctamente rellenada, gestionará el mensaje de error mediante la vista y, en caso de que no se suba correctamente la grabación, mostrará también un mensaje de error por pantalla para informar al usuario de que el procedimiento no ha sido exitoso.

En la Figura 28, se presenta la configuración utilizada para designar el objeto *MediaRecorder.* Por defecto, en esta primera versión de Noisitapp, se propone dicha configuración expuesta en la Figura 28, comentada detalladamente en el apartado 5.2.1.3.

```
private void setupMediaRecorder() {
    mediaRecorder = new MediaRecorder();
    mediaRecorder.setAudioSource(MediaRecorder.AudioSource.MIC);
    mediaRecorder.setOutputFormat(AudioFormat.ENCODING_PCM_16BIT);
    mediaRecorder.setAudioEncoder(MediaRecorder.AudioEncoder.AAC);
    mediaRecorder.setAudioChannels(1);
    mediaRecorder.setAudioEncodingBitRate(128000);
    mediaRecorder.setAudioSamplingRate(48000);
    mediaRecorder.setOutputFile(pathSave);
```

Figura 28. Configuración del objeto MediaRecorder para el formato de grabación.

5.2.3.6. **Escritura en Firebase**

Algunos de los controladores de tareas de la aplicación deben escribir de la base de datos de *Firebase*, como en el caso de la actualización de la base de datos de usuarios o la subida de archivos para almacenarlos. En ambos casos, pese a que sean bases de datos diferentes, precisan de una estructura similar, debiendo crear primeramente una referencia a la ruta de acceso completa del archivo, incluyendo el nombre. Las tareas de lectura y escritura se han hecho siguiendo la documentación de *Firebase* disponible online en [10], [11] y [12].

5.2.3.6.1. Actualización de la base de datos

Puesto que *Real-Time Database* es una base de datos *NoSQL*[2] basado en un archivo *JSON*[3], no se deberá eliminar la información y subir una nueva, sino que simplemente será necesario sobrescribir la información existente. Para ello, se necesitará el objeto *User*, con la información modificada en su interior, el objeto *FirebaseAuth*, permitiendo autenticar el usuario, y el objeto *DatabaseReference* con la ruta de la base de datos en la nube. Invocando la función *child* del objeto *DatabaseReference*, pasando por parámetro el objeto *FirebaseAuth* junto al identificador del usuario y el objeto *User*, se sobrescribirá instantáneamente la base de datos.

Unos apartados más adelante se muestra la estructura resultante de un objeto *User* en *Real-time Database*.

5.2.3.6.2. Subida de archivos

La subida de archivos se realiza, concretamente, cuando el usuario realiza una nueva grabación y desea almacenarla en la nube. Para poder realizar esta tarea, el controlador deberá gestionar dos objetos: *Uri* y *StorageReference,* además del propio archivo que desea subirse, almacenado en el dispositivo. El objeto *Uri* permitirá indicar la ruta actual del archivo que se subirá, mientras que *StorageReference* permitirá actualizar la base de datos en una ruta determinada de la base de datos. Para poder realizar dicha tarea, se invocará a la función *putFile* del objeto *StorageReference*, pasando por parámetro el propio objeto *Uri* del archivo en la memoria externa del dispositivo móvil.

Unos apartados más adelante se muestra la estructura resultante del almacenamiento de archivos de audio en *Firebase Storage*, la base de datos de grabaciones de los usuarios.

---

[2] Sistema de gestión de bases de datos donde la información no requiere de una estructura fija en tablas, a diferencia del tipo de bases de datos *SQL*.
[3] Formato de texto sencillo para el intercambio de datos. Se trata de un subconjunto de la notación literal de objetos de *JavaScript*.

5.2.3.7. **Lectura de la base de datos**

Una vez se han realizado las labores de escritura en la base de datos, se querrá, para seguir con el cometido de la aplicación, consultar información añadida anteriormente y leer sus valores. Para realizar esto, siempre deberemos gestionar una referencia a la base de datos en cuestión, con el objetivo de obtener un usuario en concreto, o un archivo de audio, accediendo a *Real-time Database* o a *Firebase Storage*, respectivamente. Como se ha indicado en el apartado 5.2.3.6, las tareas de lectura y escritura se han hecho siguiendo la documentación de *Firebase* disponible online en [10], [11] y [12].

5.2.3.7.1. Consulta de usuarios

Para la lectura de los usuarios o de información asociada a ellos, siempre se realizará de la misma manera, a partir de la función *checkIfUserExistAndFillData* a cada cambio de pantalla. Esta función permite conectar la base de datos y comprobar que hay una entrada para el usuario en cuestión, o si no la hay. Si el usuario tiene esa información se almacena un objeto tipo *User* y, si no la tiene, genera el mismo objeto con información por defecto.

Para ello, necesitará un objeto *DatabaseReference* junto a la referencia del usuario en un objeto *FirebaseAuth*.

```
/**
 * Function that connects to DB and checks whether there is an entry for that specific user
 * or not. If we have user data, we will save it to a variable and fill the UI
 */
private void checkIfUserExistAndFillData() {
    // Getting reference to database from specific logged in user
    DatabaseReference ref = this.database.getReference(mAuth.getCurrentUser().getUid());
    ref.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            if(dataSnapshot.exists()){
                // If the user has an entry to DB, we will fill UI with the data
                user = dataSnapshot.getValue(User.class);
                assert user != null;
                user.setUid(mAuth.getCurrentUser().getUid());
                mAdapter = new AdapterRecordings(user.getRecords(), DashboardActivity.this);
                mRecyclerView.setAdapter(mAdapter);
                Log.d("USER", "user: "+ user.getName());
                writeDataToScreen();

                // Hide Spinner
                spinner.setVisibility(View.GONE);
            } else {
                //Adding default data for user
                user.setUid(mAuth.getCurrentUser().getUid());
                user.setEmail(mAuth.getCurrentUser().getEmail());
                updateUserDB();
                writeDataToScreen();
            }
        }
```

Figura 29. Función checkIfUserExistAndFillData de la clase DashboardActivity.

Puede verse un ejemplo de la función *checkIfUserExistAndFillData* para la pantalla en la Figura 29.

5.2.3.7.2. Descarga de archivos

Para la descarga de archivos se realiza de manera similar a la consulta y lectura de la estructura de usuarios. Primeramente, deben generarse dos objetos: uno del tipo *StorageReference* y otro del tipo *FirebaseReference* para poder generar la estructura básica y buscar el archivo en la base de datos. Cabe destacar que, antes de comenzar la descarga, se consulta la ruta del archivo de audio, información disponible como atributo en la lista de grabaciones de un usuario. Si el archivo se encuentra, realiza una llamada a *downloadFile* que, mediante un objeto *DownloadManager* y otro objeto *Uri*, realiza la petición de descarga a una carpeta de la memoria externa del dispositivo móvil, ruta específicamente indicada en el objeto *Uri*.

5.3. Firebase: Real-time Database y Storage

Junto a la aplicación móvil, usada a través de un dispositivo *Android*, el servicio *back-end* de *Firebase* supone una mitad indispensable para la solventar los requisitos expuestos en el capítulo 4, puesto que permitirá el acceso a la información con cualquier dispositivo *Android* con conexión a Internet. Como se ha introducido anteriormente, las bases de datos utilizadas en *Firebase* son *Real-time Database* y *Storage,* con el fin de implementar una base de datos de usuarios y el almacenamiento a sus grabaciones, respectivamente.

5.3.1. Gestión de los usuarios y las grabaciones

Con tal de implementar una estructura escalable para la gestión de usuarios, se definen una serie de parámetros indispensables para cada uno de ellos, comenzando por un identificador aleatorio asignado por *Firebase*, o *uid*, a partir de la cual se implementarán el resto de las variables.

La estructura la base de datos para un usuario propuesta es la siguiente:

- Identificador de *User* 1:
    - Dirección de correo electrónico
    - Nombre
    - Grabaciones:
        - Identificador de *Recording* 1:
            - Nombre
            - Fecha y hora
            - Localización
            - Latitud y longitud
            - Duración
            - Dispositivo móvil usado
            - Etiquetas:
                - …
            - Ruta al archivo de audio asociado
            - Formato de grabación:
                - Número de canales
                - Codificación

        - *Bitrate*
        - *Sampling Rate*
        - Formato
        - Fuente origen
    - Información adicional:
        - …
    - …
  - Aceptación términos y condiciones
  - Verificación del correo electrónico
- Identificador de *User* 2:
  - …

Como puede apreciarse en la estructura anterior y se ha documentado en el apartado 5.2.1.1, el objeto tipo *User* tendrá una lista dinámica de objetos *Recording* y esta, a su vez, tendrá asociado un objeto tipo *RecordFormat* y otro *MoreInformation*. De esta manera la estructura propuesta podrá crecer de forma dinámica a medida que se añadan usuarios, tratando así su información asociada sin necesidad de leer toda la base de datos o escribirla y, haciéndolo únicamente cuando sea necesario.

Figura 30. Captura de pantalla de la consola de Firebase para la base de datos Real-time Database.

Como se puede apreciar en la Figura 30, la información de los usuarios se anida en paralelo siendo, su identificador, el elemento de búsqueda característico para cada usuario. De la misma manera, cada grabación que realice el usuario generará un objeto nuevo en la lista de grabaciones de ese usuario en concreto, con la información que puede verse en la Figura 31. Como se verá en el siguiente apartado, la diferencia entre la base de archivos de audio y la de usuarios es que, en la primera, todos los archivos están situados en la misma ruta, con nombres diferentes.

Figura 31. Captura de pantalla de la consola de Firebase para un usuario y grabación determinados.

Los nombres de estos archivos de audio son identificativos y, como se verá en el apartado 5.3.2.1, el nombre de archivo sigue una estructura determinada que permitiría conocer el usuario del que es esa grabación de audio. Además, las grabaciones, tienen incluida la lista dinámica de etiquetas, siguiendo estas una taxonomía determinada, comentada a continuación.

5.3.1.1. Taxonomía de las etiquetas de la base de datos

Con el fin de clasificar, evaluar e interpretar las grabaciones realizadas por el usuario, se define una taxonomía para las etiquetas propuestas a la hora de añadir información al archivo de audio. Con la finalidad de etiquetar los sonidos de manera común, se han definido una serie de etiquetas, a partir de las cuales, cuando se escucha un sonido que pertenece a una de ellas, el usuario etiqueta la totalidad de la grabación con ese código.

Puesto que no existe un estándar universal en cuanto a la taxonomía a seguir para etiquetar muestras sonoras, se decide seguir el sistema de etiquetado propuesto en [13], con el objetivo de sustentar la nomenclatura en un estudio especializado en este campo. Además, con la intención de complementar los estudios realizados en el artículo anterior, se complementa la base de datos con las propuestas en [14], donde se estandarizan las etiquetas de sonidos que provienen en entornos urbanos y suburbanos.

Es sumamente importante que las etiquetas que tengan disponibles los usuarios sean representativas y lo más exactas posibles de lo que pretende etiquetarse ya que, posteriormente, servirán para construir un algoritmo de *machine-learning* con ellas. La taxonomía propuesta utilizada puede consultarse en Anexo A.

En conclusión, se ha implementado la base de datos de los usuarios siguiendo la premisa de que esta sea escalable y, además de tener un acceso sencillo para su lectura y su escritura, permita relacionarse fácilmente con los archivos de audio, alojados en otra base de datos diferente.

5.3.2. Almacenamiento de los archivos de audio

El almacenamiento de archivos, como se ha comentado anteriormente, ha sido realizado con la tecnología de *Firebase Storage*. Esta base de datos, a diferencia de la de los usuarios, es ligeramente más sencilla ya que la totalidad de los archivos de audio están en la misma ruta, en la misma carpeta. La correlación entre los usuarios y sus grabaciones, por tanto, es a partir del nombre de archivo, como se indica a continuación.

5.3.2.1. Descripción de la base de datos

El nombre de los archivos generados contiene la información del identificador del usuario y un número aleatoriamente generado por números y letras, seguido de la extensión, como, por ejemplo:

4os553cOV5ZI5ZDnJfVfsR94hJl1_audio_record_ce6ccff6-82ce-4139-b1a3-e7c0ceb15f61.mp3

Observando el nombre del archivo y conociendo el formato, podemos saber que este audio pertenece al usuario con identificador “4os553cOV5ZI5ZDnJfVfsR94hJl1” en la base de datos.

6. Verificación del sistema, pruebas e incidencias

Una vez implementada la aplicación, a continuación, se explican las pruebas realizadas con tal de verificar el funcionamiento de todo el sistema.

Como se explica en el siguiente punto, se han utilizado dos tipos de pruebas. Las primeras están formadas por comprobaciones de caja negra, para validar funcionalidad por funcionalidad. En segundo lugar, se ha utilizado la publicación en fase *beta* para corregir los posibles errores de programación.

6.1. Verificaciones del sistema

Las pruebas realizadas con tal de validar el sistema se pueden clasificar en dos apartados: las comprobaciones realizadas en el momento del desarrollo y las pruebas generales de todo el sistema realizadas al finalizar la implementación. En primer lugar, tras el desarrollo de una funcionalidad se comprueba que la aplicación cumpla los requisitos inicialmente expuestos. Seguidamente, tras haber comprobado la unidad de cada funcionalidad, se realizan comprobaciones generales de todo el sistema para probar su integridad y asegurar que esta sería válida para lanzar una versión inicial *beta* a nuevos usuarios.

En la fase inicial, pese a que su publicación en el mercado de aplicaciones *Android* no se ha realizado, se han aplicado comprobaciones en entornos reales con 9 usuarios, pudiendo verificar el funcionamiento en los siguientes dispositivos móviles:

- *One Plus 6T*
- *One Plus 6*
- *One Plus 5T*
- *Huawei Nova 5T*
- *Huawei P10 Plus*
- *Huawei P20 Lite*
- *Xiaomi Redmi Note 8T*
- *BQ Aquaris X Pro*

6.2. Tareas de validación del sistema

Las tareas de validación que han realizado los usuarios de prueba de Noisitapp son las siguientes:

1. Registrar un nuevo usuario en el sistema.
2. Iniciar sesión en el sistema con credenciales de usuario.
3. Reestablecer la contraseña de usuario.
4. Grabación de una muestra de audio rellenando el formulario habilitado para complementar la información de la grabación.
5. Reproducción de la nueva grabación.
6. Consulta del panel de control con y sin grabaciones.
7. Edición del nombre de una grabación.
8. Edición del nombre de usuario.
9. Consulta de la información acerca de Noisitapp.

10. Reproducción de una grabación previamente realizada con el dispositivo.
11. Reproducción de una grabación no realizada con el dispositivo con la que se captó la muestra de audio.
12. Cerrar sesión de usuario.

Mediante la ejecución satisfactoria de las tareas anteriores, se ha validado el sistema con tal de obtener la primera versión de Noisitapp, procediendo entonces a su documentación.

6.3. Gestión de incidencias

El software utilizado para poder realizar un seguimiento de las instalaciones de la aplicación en los dispositivos de los usuarios ha sido *Firebase Crashlytics.* Este programa captura las excepciones de la aplicación y las registra en el sistema web. Gracias a este registro de errores puede realizarse un seguimiento de los problemas de desarrollo en la aplicación. Durante la comprobación de la fase inicial para los dispositivos anteriores, se fueron arreglando los errores capturados por *Crashlytics* hasta conseguir una versión estable de la aplicación. A continuación, se muestra una captura de pantalla de la interfaz web de *Crashlytics:*

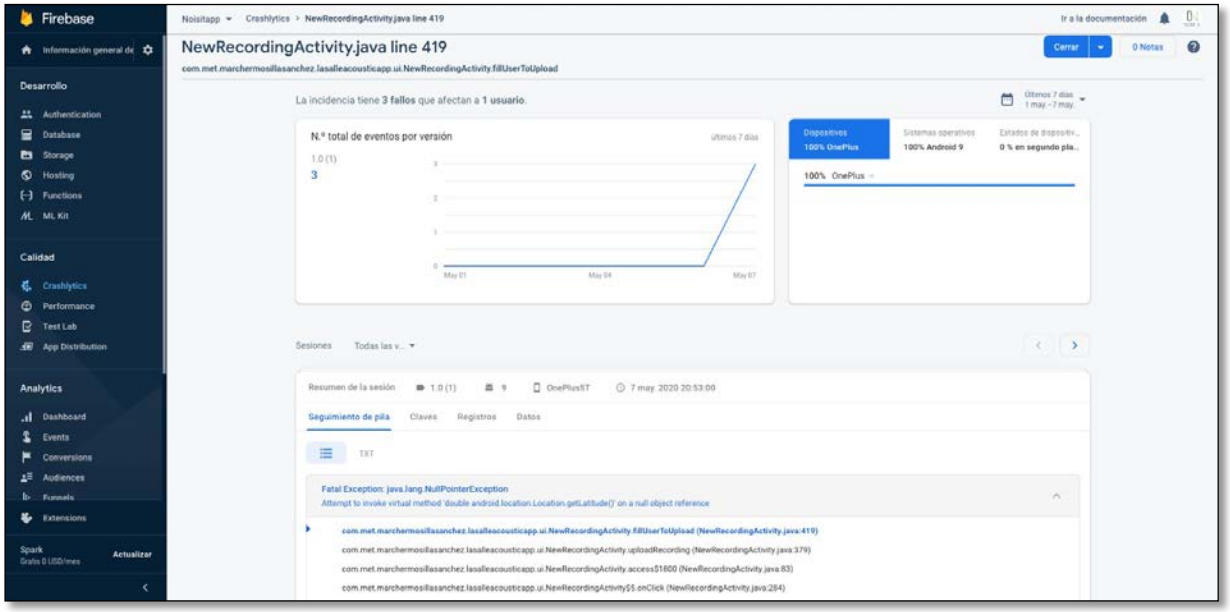

Figura 32. Captura de pantalla del entorno web de Firebase Crashlytics.

Como puede verse en la Figura 32, *Crashlytics* indica la referencia que reproduce el error con la excepción. En este caso, la línea 419 de la clase *NewRecordingActivity*, daba un error *NullPointerException*, en un dispositivo *One Plus 5T* con el sistema operativo *Android* 9.

Cabe destacar que, tras realizar las comprobaciones en entorno real con la lista de dispositivos móviles anteriores, *Crashlytics* detectó una incidencia que afectaba a gran parte de los usuarios. La incidencia estaba relacionada con la carpeta que el sistema crea

en el almacenamiento externo del dispositivo móvil. Gracias a la excepción capturada por *Firebase*, se pudo identificar que la aplicación no estaba creando correctamente la carpeta de almacenamiento de archivos en la ruta indicada, por lo que, al realizar una nueva grabación y configurar el objeto *MediaRecorder* para iniciar la grabación, el sistema no encontraba la ruta de dicha carpeta. Tras corregir la ruta donde la aplicación crea la carpeta de almacenamiento de grabaciones, volvieron a realizarse verificaciones en los dispositivos que habían dado problemas, para comprobar su correcto funcionamiento y dar por cerrada la incidencia.

7. Estudio del coste económico y temporal del trabajo

En este apartado se realizará una estimación temporal de las horas invertidas en este proyecto, distribuidas entre las diferentes tareas necesarias para su implementación.

Los primeros pasos realizados fueron los de análisis de la necesidad vigente entorno a la construcción de sistemas autónomos de aprendizaje automático, así como un estudio de las diferentes alternativas que resolvían dicha cuestión. Tras tener los primeros indicios de la problemática y su posible solución, se plasmaron los requisitos técnicos y se estudiaron las diversas herramientas tecnológicas que pretendían utilizarse para su implementación. Una vez claros los requisitos y las herramientas, comenzó el desarrollo de la aplicación: estructuración de *Real-time Database* y *Storage* de *Firebase* e implementación del algoritmo en *Android Studio.* Poniendo fin al desarrollo, la fase de pruebas y corrección de errores, donde se registraron varios *bugs* y se dió solución a decenas de incidencias, monitorizadas en la herramienta de *Firebase, Crashlytics*. Finalmente, la fase de documentación del trabajo final de máster y elaboración de la memoria.

El trabajo final de máster, como tal, ha ocupado un total de 400 horas, realizadas entre el 1 de junio del 2019 y el 25 de mayo del 2020. A continuación, se presentan las tareas realizadas junto a una aproximación del coste temporal dentro del proyecto:

- **Documentación y estudio de la tecnología:** 45 horas.
- **Reuniones con los tutores**: 15 horas.
- **Configuración de *Firebase*:** 25 horas.
- **Implementación en *Android Studio***: 200 horas.
- **Pruebas y corrección de *bugs***: 65 horas.
- **Redacción de la memoria**: 50 horas.

8. Conclusiones y líneas de futuro

8.1. Conclusiones

En este trabajo se propone la arquitectura de sistemas para una solución técnica de registro de muestras sonoras mediante dispositivos móviles. Mediante éste, se ha documentado toda la estructura propuesta de una aplicación móvil para dispositivos *Android* en comunicación con los servicios en la nube de *Firebase*. Realizado el trabajo de implementación de esta estructura, una de las principales conclusiones giraría entorno a la facilidad y versatilidad que ofrecen las tecnologías móviles elegidas para el desarrollo de la arquitectura propuesta, con el fin de dar, o complementar, la solución a un problema de recogida de datos para el futuro desarrollo de aplicaciones de reconocimiento automático de eventos sonoros o de paisajes sonoros mediante técnicas de *machine-learning*.

Como se ha comentado en diversos capítulos de este documento, en la elaboración de sistemas autónomos de reconocimiento de audio, la base de datos es de suma importancia para obtener buenos resultados en la identificación de patrones. La base de datos debe ser lo más representativa posible de la naturaleza que se pretende analizar y esto supone una barrera significativa cuando los datos necesarios para construirla son complejos de registrar. En este sentido, la tecnología móvil permite poner a disposición de millones de usuarios la capacidad de aportar muestras sonoras de su entorno mediante su teléfono móvil *Android*.

Cabe destacar, entonces, la versatilidad y comodidad en el desarrollo de aplicaciones móviles para dispositivos *Android*, junto a la avanzada integración con los servicios de *Firebase. Firebase,* como servidor, permite una sencilla comunicación con entornos web y aplicativos móviles, que establecen un contexto favorable para implementar soluciones como Noisitapp. Estas características, por tanto, suponen el escenario óptimo para proyectos en constante desarrollo y sus requisitos cambiantes, de manera que permiten adaptar partes de la estructura para satisfacer dichos cambios. En este proyecto, donde se plantea la continuidad en cuanto a estudios y funcionalidades, capacita la generalización de un sistema robusto a cambios futuros que puedan plantearse ante posibles situaciones que este pueda requerir.

Finalmente, debe destacarse el gran abanico de posibilidades que se abre cuando se implementan sistemas mediante tecnologías móviles. Como se expone a continuación de este apartado, en las líneas de futuro, el desarrollo expuesto en este trabajo supone la base a partir de la cuál seguir desarrollando múltiples funcionalidades que aporten valor, tanto al investigador que usará las muestras de audio y sus etiquetas, como para el usuario para realizar las grabaciones y gestionarlas más cómodamente. Teniendo en cuenta lo anterior, que el sistema se plantea para tecnologías móviles y los múltiples dispositivos móviles que pueden usarlo, este requiere de especial atención en la fase de pruebas en diversos sistemas operativos y modelos de telefonía del mercado. Esto permitirá garantizar el correcto funcionamiento para el mayor de usuarios disponibles. Por lo tanto, teniendo en cuenta todo lo anterior, la premisa más importante durante la programación de los módulos ha sido la competitividad del sistema y su escalabilidad, con la intención de que próximos desarrollos se realicen de la manera más sencilla posible, con el menor coste

temporal y asegurando el correcto funcionamiento en el mayor de dispositivos móviles posibles.

8.2. Líneas de futuro

El desarrollo de una solución de esta magnitud posibilita la realización de múltiples y diversas tareas relacionadas con la implementación técnica y la mejora de diferentes puntos. Durante el desarrollo del proyecto, como se ha comentado en el apartado anterior, la premisa más importante en la que se han basado todas las elecciones ha sido que este fuera escalable a futuro: en usuarios y en grabaciones. Actualmente, el sistema no dispone de un límite técnico en cuanto a la escalabilidad de este, ya que no lo determina el propio sistema como tal, sino capacidad de uso contratada en la *suite* de servicios de *Firebase.* Teniendo en cuenta que el proyecto estaba en estado embrionario, las decisiones tomadas han tenido y tendrán mucho peso para construir más funcionalidades y otros módulos ante la arquitectura propuesta en un futuro. Además, volviendo a hacer referencia al hecho de que se comenzaba un proyecto desde cero, supuso un gran esfuerzo el crear esa estructura a partir de la cuál empezar a construir y, por tanto, han aparecido otros puntos significativamente interesantes a trabajar como líneas de futuro. Tal como se ha estructurado la documentación del trabajo, separando la aplicación cliente en *Android* y el servidor en *Firebase,* se presentan las líneas de futuro de la misma manera. Estos se detallan a continuación:

1. Aplicación *Android:* Como cualquier desarrollo pensado para usuarios, a medida que se desarrolla la idea inicial se observan posibles puntos de mejora evidentes, que tendrían un impacto considerable en la experiencia de uso. Desde la experiencia de usuario, las posibles funcionalidades o la información que recoge el dispositivo para cada muestra de audio, entre otras. Las líneas de futuro que se plantean para la aplicación móvil son las siguientes:
    a. Que la información de inicio de sesión quede almacenada en preferencias compartidas del dispositivo móvil, de manera que la aplicación mantenga la información de acceso del usuario y, por lo tanto, no solicite dicha información al abrir la aplicación.
    b. Conocer mejor al usuario y sus preferencias de grabación: que este pueda asignar las preferencias en cuanto al formato de grabación del dispositivo, de manera que la configuración quede registrada en su perfil para nuevas grabaciones.
    c. Calidad de grabación: mejora de los formatos de grabación, tasa de bits e, incluso, que la aplicación sea capaz de calibrar el dispositivo móvil, con la intención de asemejar el micrófono del teléfono a un micrófono profesional.
    d. Control de grabación: mostrar más información al usuario durante la captación de nuevas muestras sonoras monitorizando, por ejemplo, la señal de entrada en temporal o en frecuencial. Además, que el sistema muestre información en cuanto a los niveles de presión acústica, tras los cálculos acústicos pertinentes. Otro punto interesante para el futuro sería dotar al usuario de una herramienta de edición de la grabación, con la que podría etiquetar mejor la grabación.
    e. Sistema de etiquetado: mejora en la metodología de selección de etiquetas e, incluso, añadir nuevas referencias a la lista. Además, habilitar al usuario la posibilidad de creación de etiquetas, con el objetivo de descubrir nuevos sonidos.

2. Servidor en *Firebase:* Transcurrido un tiempo tras lanzar la aplicación en productivo, a disposición de nuevos usuarios, extraer las muestras de audio recopiladas en la base de datos y usarlas para entrenar un sistema de reconocimiento de patrones. También, podría realizarse un análisis exhaustivo acerca de los datos recogidos como, por ejemplo, estudiar la relación entre localización y etiqueta con el fin de conocer qué sonidos se suelen dar según el punto físico del planeta.

En definitiva, el que sea un proyecto de esta magnitud, con dos sistemas en comunicación, permite que haya múltiples campos de mejora en varias direcciones, como se acaba de presentar. Al fin y al cabo, la importancia de este trabajo recae en la estructura del sistema que permitirá, en un futuro próximo, desarrollar, construir e implementar funciones en diferentes módulos de manera sencilla y cómoda.

9. Bibliografía

[1] G. H. &. M. Burgess, «Australian Academy of Science,» [En línea]. Available: https://www.science.org.au/curious/earth-environment/health-effects-environmental-noise-pollution. [Último acceso: 8 mayo 2020].

[2] R. Brandom, «There are now 2.5 billion active Android devices,» 7 Mayo 2019. [En línea]. Available: https://www.theverge.com/2019/5/7/18528297/google-io-2019-android-devices-play-store-total-number-statistic-keynote. [Último acceso: Mayo 2020].

[3] K. B. &. V. Rajlich, «Software maintenance and evolution: A roadmap,» *CiteSeerx,* 2000.

[4] A. Radicchi, «Hush City. A new mobile application to crowdsource and assess "everyday quiet areas" in cities.,» de *Invisible Places 2017 Sound, Urbanism and Sense of Place*, Sao Paulo Island, Azores, Portugal, 2017.

[5] M. González, «Xataka: Esta es la tecnología con la que Shazam puede predecir antes que nadie la canción del verano.,» 2015. [En línea]. Available: https://www.xataka.com/aplicaciones/esta-es-la-tecnologia-con-la-que-shazam-puede-predecir-antes-que-nadie-la-cancion-del-verano. [Último acceso: 2018].

[6] S. S. F. R. G. H. G. H. P. R. V. W. P. &. J. A. Kahl, «Overview of BirdCLEF 2019: Large-scale Bird Recognition in Soundscapes,» *CLEF 2019,* 2019.

[7] T. O. Ayodele, de *Types of Machine Learning Algorithms*, University of Portsmouth, United Kingdom, February 1st 2010.

[8] G. R. X. S. &. F. A. Joan Claudi Socoró, «Development of an anomalous noise event detection algorithm for dynamic road traffic noise mapping.,» 2015.

[9] «Android Developers - Docs: MediaRecorder,» [En línea]. Available: https://developer.android.com/reference/android/media/MediaRecorder.

[10] «Real-time Database: Read and Write Data in Android,» [En línea]. Available: https://firebase.google.com/docs/database/android/read-and-write.

[11] «Firebase Storage: Upload Files on Android,» [En línea]. Available: https://firebase.google.com/docs/storage/android/upload-files.

[12] «Firebase Storage: Download Files on Android,» [En línea]. Available: https://firebase.google.com/docs/storage/android/download-files.

[13] E. L. P. J. M. H. a. H. v. L. Salomons, «Inferring Human Activity Recognition with Ambient Sound on Wireless Sensor Nodes,» Vittorio, Basel, Switzerland, 2016.

[14] F. A. &. J. C. Socoró, «Description of Anomalous Noise Events for Reliable Dynamic Traffic Noise Mapping in Real-Life Urban and Suburban Soundscapes,» *MDPI: Applied Sciences,* p. 7. 146. 10.3390/app7020146, 4 February 2017.

10. Anexo

10.1. Anexo A

Los códigos de las etiquetas usadas son las siguientes:

- (Human) Person Speech = persona hablando
- (Human) Person Laughing = persona riendo
- (Human) Person Shouting = persona gritando
- (Human) Person Crying = persona llorando
- (Human) Person Coughing = persona tosiendo
- (Human) Person Sneezing = persona estornudando
- (Human) Person Singing = persona cantando
- (Human) Infant = bebé
- (Human) Children = niños
- (Human) Person Footsteps = pasos de persona

- (Nature) Wind = viento
- (Nature) Water = agua
- (Nature) Thunder = trueno
- (Nature) Rain = lluvia

- (Animals) Dog: Bark = perro ladrando
- (Animals) Dog: Howl = perro aullando
- (Animals) Cat = gato
- (Animals) Bird: Tweet = pájaro piolando

- (Vegetation) Leaves Rustling = hojas moviéndose

- (Construction) Jackhammer = martillo pneumático
- (Construction) Hammering = martillo
- (Construction) Drilling = taladradora
- (Construction) Sawing = sierra
- (Construction) Explosion = explosión
- (Construction) Engine running = motor encendido

- (Ventilation) Air Conditioner = aire acondicionado

- (Transport) Bicycle = bicicleta
- (Transport) Skateboard = monopatín
- (Transport) Boat = barco
- (Transport) Train = tren
- (Transport) Subway = metro
- (Transport) Car = coche

- (Transport) Police = coche de policía
- (Transport) Ambulance: Siren = sirena de ambulancia
- (Transport) Vehicle Horn = bocina de vehículo
- (Transport) Vehicle Brakes = frenos de un vehículo
- (Transport) Vehicle Wheels Passing = ruedas de un vehículo en contacto con el asfalto
- (Transport) Motorcycle = motocicleta
- (Transport) Bus = bus
- (Transport) Truck = camión
- (Transport) Garbage Truck = camión de la basura
- (Transport) Airplane = avión
- (Transport) Helicopter = helicóptero

- (Music) Live Music = música en directo
- (Music) Car Radio = radio del coche
- (Music) House Party = fiesta en una casa
- (Music) Club = club
- (Music) Ice Cream Truck = camión de los helados
- (Music) Background Music = música de fondo

- (Other) Alarm = alarma
- (Other) Bell = campana
- (Other) Unidentified Sound = sonido no identificado