ESCUELA DE INGENIERÍA DE
FUENLABRADA

GRADO EN INGENIERÍA EN SISTEMAS
AUDIOVISUALES Y MULTIMEDIA

TRABAJO FIN DE GRADO

DESARROLLO DE UNA MESA DE MEZCLAS VIRTUAL PARA DISPOSITIVOS ANDROID

Autor: María de la Osa Martínez

Tutor: David Gualda Gómez

Curso académico 2022 / 2023

Agradecimientos

Siempre he creído saber a quién dedicaría este largo camino, o cómo lo haría, pero, a veces, las cosas no se dan como se creen y el camino cambia, así como la gente que te acompaña en él.

En primer lugar, este trabajo va dedicado a quien fue, es, y será, mi gran mentor en la vida, de quien quiero seguir los pasos, mi padre. Que, si no fuera por haber querido llevarle la contraria, no estaría hoy aquí, a pocos pasos, de por suerte, poder llamarme ingeniera. Así como agradecer, eternamente, a mi madre y a mi hermana, por ser mis apoyos en todo momento, que, incluso sin pedírselo, han sabido qué hacer o qué decir.

Por supuesto, dar las gracias a David, mi tutor, que me ha acompañado en este largo proceso, dándome ánimos en los peores momentos y contagiándome su positivismo... También, gracias por todo el conocimiento trasmitido tanto en este caso, como en las clases.

Mencionar a los profesores y profesoras del grado, que, aunque a veces no me hayan gustado sus decisiones, me han ayudado a crecer y a formarme como profesional.

Por último, pero no menos importante, agradecer a la gente que me ha acompañado en este camino, sigan a mi lado o no, porque me han hecho crecer como persona y como profesional. A mis amigas y amigos que han sufrido y escuchado mi lucha en esta etapa.

Es necesario cerrar capítulos para empezar otros nuevos y ver qué depara la vida. Ahora es el momento, estoy preparada.

Resumen

En este Trabajo de Fin de Grado (TFG) se expone de forma extensa el desarrollo de una aplicación Android de lenguaje Kotlin, que permite emular el funcionamiento de una mesa de mezclas. Se trata de un dispositivo que posibilita la mezcla de diferentes pistas de audio.

La aplicación puede ser enfocada al uso por parte de cualquier usuario, posea conocimientos del mundo del audio o no, gracias a la sencillez que presenta la interfaz, así como de las instrucciones que se encuentran dentro de ella, enseñando al usuario las diferentes opciones que tiene en la mezcla de pistas de audio, ya sean introducidos manualmente, o usando los que vienen por defecto al instalar la aplicación.

En último lugar, hay que mencionar que los resultados que se han obtenido a lo largo de este estudio son satisfactorios, puesto que se ha logrado la finalidad buscada de poder mezclar audios en una aplicación creada para Android.

Palabras clave

Aplicación, Android, Kotlin, mesa de mezclas, pistas, audio.

Abstract

In this Final Degree Project (TFG), it is extensively exposed the developement of a Kotlin language Android application that emulates the funcionality of a mixing console, which is a device that enables the mixing of different audio tracks.

The application can be focused on the use by any user, whether they have knowledge of the world of audio or not, thanks to the simplicity of the interface, as well as the instructions found within it. Showing the user the different options that has in the sound mix, whether the audio files are entered manually or using the ones that come by default when installing the application.

Lastly, it should be mentioned, that the results obtained by this study are sastisfactory, since the desired purpose of being able to mix audios in an application created for Android has been achieved.

Keywords

Application, Android, Kotlin, mixing console, tracks, audio.

Índice de contenidos

Índice de figuras

Índice de tablas

Acrónimos

AAC	Advanced Audio Coding
App	Application
API	Application Programming Interface
APK	Android Application Package
CD	Compact Disk
dB	Decibelios
DJ	Disc-jockey
DSP	Digital Signal Processor
FLAC	Free Losslesss Audio Codec
ID	Identification
IDE	Integrated Development Enviroment
iOS	iPhone Operating System
IU	Interfaz de Usuario
KT	Kotlin
Linux	Lovable Intellect Not Using XP
MP3	MPEG-1 Audio Layer 3
OGG	Ogg Vorbis
PFL	Pre-Fade Listen
RAE	Real Academia Española
RAM	Random Access Memory
RCA	Radio Corporation of America
TFG	Trabajo Fin de Grado
URI	Uniform Resource Identifier
UX	User Experience
WAV	Waveform Audio File Format
XML	Extensible Markup Language

1. Introducción

El uso de los *smartphones* o *tablets* en el día a día de las personas cada día es mayor. Tomando esto como referencia, y la motivación personal de conocer los equipos utilizados en el ámbito de la música, se ha querido implementar una aplicación que emule una mesa de mezclas virtual. Además, se ha querido realizar una aplicación diferente a las encontradas en la Play Store de Android, puesto que se ha potenciado la sencillez y la manejabilidad del usuario final, lo que la convierte en una opción ideal para principiantes que se quieran introducir en mezclas sencillas de audio.

1.1. Presentación

En la memoria presentada se proporciona detalladamente el proyecto llevado a cabo como Trabajo de Fin de Grado (TFG) de Ingeniería en Sistemas Audiovisuales y Multimedia. El proyecto ha sido el diseño y desarrollo de una aplicación Android que permite al usuario la mezcla de sonidos, ya sean grabados en el momento, desde la biblioteca del dispositivo o mediante audios por defecto.

Así como el desarrollo implementado en Android Studio, se ha investigado en diversos temas, desde el lenguaje de programación utilizado, hasta el uso de una mesa de mezclas. El estudio de éstas y otras competencias se enumeran a lo largo de los capítulos que se pueden encontrar en esta memoria.

1.2. Estructura de la memoria

Los capítulos que se encuentran a lo largo del TFG son los siguientes.

1. **Introducción.** Presentación inicial del documento que permite llevar a cabo la contextualización del lector sobre la memoria.
2. **Objetivos**. Se concretan los motivos principales tras el proyecto, así como la planificación temporal, realizada mediante un diagrama de Gantt que aproxima los tiempos tomados en las diferentes partes del trabajo.
3. **Definiciones y conceptos**. Se recogen las descripciones de las herramientas con las que se ha procedido a la creación del prototipo explicado a lo largo de la memoria, desde la definición de Android y lo relacionado con ello, hasta la ganancia de audio o las diferentes extensiones que pueden tener los archivos.

4. **La mesa de mezclas**. Dado que el proyecto realizado es un prototipo que recoge las funciones base de las mesas de mezclas, se ha decidido la escritura de un capítulo que recoge la historia, los tipos que se pueden encontrar y las partes que las componen.
5. **Desarrollo**. Expone de forma prolongada el diseño y la implementación realizados en Android Studio. Los diferentes subapartados muestran los archivos que forman el conjunto de la aplicación final entregada.
6. **Resultados**. Presentación del efecto del prototipo realizado.
7. **Costes**. Muestra una aproximación del valor total del proyecto.
8. **Conclusiones**. Recopilación final del análisis de los datos recopilados a lo largo del trabajo. Destacando las habilidades adquiridas y mostrando posibles líneas de mejora que se podrían ejecutar en el futuro.
9. **Bibliografía**. Recapitulación de las referencias consultadas.
10. **Anexos**. Información complementaria.

2. Objetivos

2.1. Objetivos del proyecto

En la realización de este prototipo se tienen los siguientes objetivos:

- Explicación detallada de los conceptos y el funcionamiento de las mesas de mezclas.
- Desarrollo de un prototipo de mesa de mezclas en Android y su posterior validación como usuario final.

2.2. Fases de desarrollo del proyecto

2.2.1. Investigación

Prácticamente durante toda la realización del prototipo y la memoria de éste, ha sido necesaria la etapa de investigación. Tanto de datos referentes a las mesas de mezclas, como de los procesos a seguir desde la creación de una aplicación de Android Studio, así como de la posterior creación de la aplicación final en Kotlin.

2.2.2. Diseño y desarrollo

La segunda fase en la que se ha trabajado ha sido la más extensa, puesto que es donde se han realizado las tareas de diseño y desarrollo de la aplicación.

En primer lugar, se ha llevado a cabo un diseño previo de cómo se plantearía la aplicación, decidiendo el número de pistas a crear, así como de las acciones que podría realizar el usuario.

Una vez se ha trabajado en el diseño en papel, se pasó a la plataforma de Android Studio, mediante la creación la interfaz de usuario.

Para concluir esta etapa, se ha desarrollado el código, uniendo lo que previamente se ha diseñado con las funciones permitidas en Android Studio, para que la aplicación sea funcional para el usuario.

2.2.3. Evaluación de resultados

Para esta fase, se ha probado la aplicación en diferentes dispositivos móviles, verificando las diversas casuísticas que se pueden tratar en la aplicación final de Android, de forma que se han probado todas y cada una de las pistas, con los audios por defecto, grabaciones y sonidos sacados de la memoria del dispositivo.

2.2.4. Documentación

Esta etapa se ha prolongado a lo largo del proyecto, puesto que se comenzó con los capítulos de los que no dependía la parte de código, es decir, los objetivos, y lo que se puede encontrar como punto número 4 de la memoria, que explica desde la historia de las mesas de mezclas, como las partes o los diferentes tipos que se pueden encontrar hoy en día. Para, más adelante, finalizar con la explicación de las demás actividades realizadas, como el desarrollo de la aplicación, los resultados y la estructura de la memoria.

2.3. Diagrama de Gantt

Meses	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Investigación																
Diseño																
Desarrollo																
Resultados																
Documentación																

Tabla 1. Diagrama de Gantt.

3. Definiciones y conceptos

A continuación, se explican diferentes conceptos que se han sido necesarios para el desarrollo del proyecto.

3.1. Android

Android fue creado en 2005 como sistema operativo de código abierto disponible para dispositivos móviles, y más tarde expandiéndose a otros dispositivos con el paso del tiempo, convirtiéndose así en multiplataforma, se basó en el programa libre de Linux [1] .

Las versiones concentradas entre la 1 y la 9 eran denominadas bajo nombres de postres, aspecto que cambió a partir de la 10 hasta la versión 13, lanzada en agosto de 2022, que pasaron a tener el nombre de la versión [2].

En la actualidad, se posiciona como el sistema operativo más utilizado del mundo, por encima de más de la mitad del mercado, seguido por iOS, sistema operativo creado por la empresa Apple, como se muestra en la Fig. 1. En los últimos años, ha sufrido una bajada de números en países como China debido a una limitación de dicho gobierno con el acceso a Google, lo que ha llevado a la creación de otros sistemas operativos.

Fig. 1. Comparativa usuarios de Android frente a iOS en 2022 [3].

Posee diferentes características entre las que se pueden mencionar la interfaz, con un diseño intuitivo para el usuario, las aplicaciones, o *apps*, que se encuentras recogidas en la Play Store, o tienda oficial, la adaptabilidad, puesto que es un sistema multiplataforma de código abierto que puede atraer a la creación de diferentes aplicaciones por programadores, y la seguridad.

3.2. Android Studio

Android Studio es el entorno de desarrollo integrado oficial para el sistema operativo Android, Integrated Development Environment (IDE), desbancando así a Eclipse como IDE oficial.

Existe la posibilidad de programar en diferentes lenguajes, tales como Java, JavaScript, C++ y Kotlin, siendo este último el usado para el desarrollo de este proyecto.

La aplicación presenta una interfaz que posee un editor de códigos, un sistema de compilación y funciones adicionales que permiten añadir recursos a la aplicación para facilitar el uso. Además, contiene un emulador que permite visualizar la aplicación a crear en diferentes dispositivos que posean el sistema operativo.

En la Fig. 2 se muestra una vista de la aplicación de Android Studio para Windows, a la izquierda se encuentran los archivos del proyecto, en la parte central el código, y a la derecha un emulador que viene por defecto en la aplicación, aunque puede ser modificado para que se asemeje a las características de cualquier dispositivo Android.

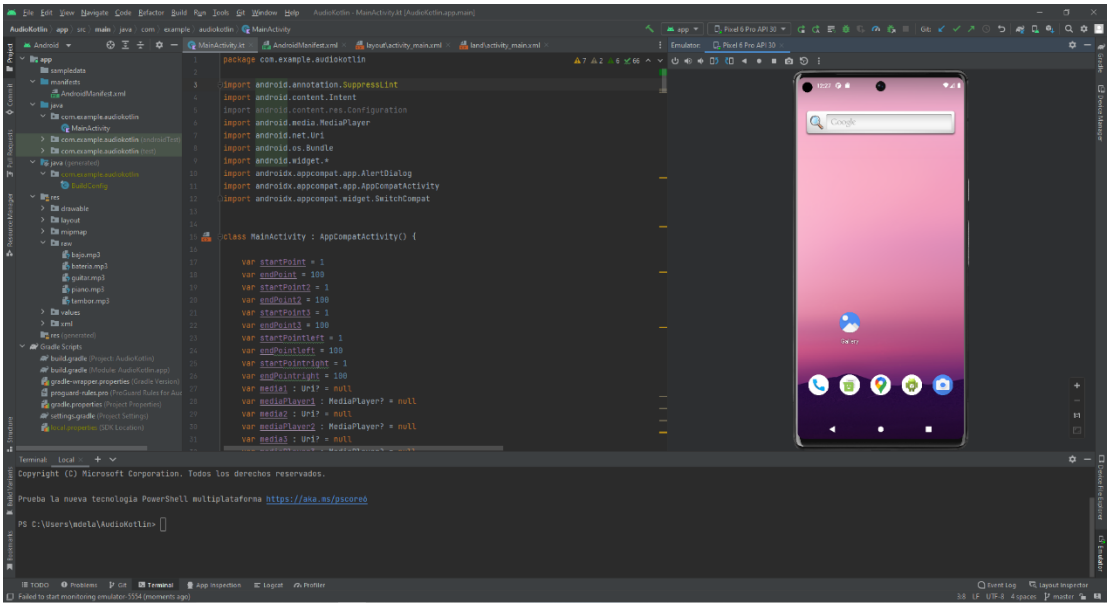

Fig. 2. Vista de Android Studio.

3.3. Kotlin

Kotlin es uno de los lenguajes de programación multiplataforma utilizado en el desarrollo de aplicaciones *software*.

Creado por JetBrains, en 2010 creó la primera versión, aunque no fue hasta 2012 el momento en el que pasó a ser código abierto. A partir de 2017 se popularizó debido al soporte otorgado por parte de Google, convirtiéndose, desde entonces, en uno de los lenguajes más importantes en la creación de aplicaciones móviles.

Se trata de un lenguaje de programación versátil y moderno con una sintaxis sencilla y diferentes herramientas de desarrollo integradas [4].

Las principales características de este lenguaje son:

- Una curva de aprendizaje sencilla, debido a su sintaxis intuitiva y fácil de usar. Además de, gracias a que se trata de un código abierto, se posee mucha información complementaria para la resolución de problemas.
- A diferencia de Java, se elimina la redundancia del código, lo que permite una optimización de la escritura final. Con respecto a esto, también permite interoperar con Java, el otro lenguaje usado para la creación de apps Android, permite la interacción entre ambas sintaxis.
- Simplificación de las llamadas de red y el acceso a las bases de datos, lo que permite dejar de lado los *callbacks*, que permite la optimización de la programación asíncrona.
- Posee la flexibilidad de ser usado con programación lineal o con programación orientada a objetos, lo que permite una mejor experiencia para el usuario que lo desarrolla.
- Seguridad frente a Java, ya que por defecto los objetos son not-null, lo que hace que el programa no compile en el caso de asignar una variable como null.

3.4. Aplicaciones nativas de Android

Las aplicaciones nativas son software diseñadas para el uso de un sistema operativo concreto, escribiéndose en un lenguaje compatible a dicho sistema. Se usan con los recursos dados por el fabricante del sistema operativo, lo que permite que se tenga un acceso prácticamente completo a los diferentes componentes del dispositivo, como por

ejemplo el micrófono, como se puede observar en la grabación de sonidos para su posterior uso como pistas de audio de este TFG, o como las diferentes APIs específicas del sistema operativo.

Por lo general, están desarrolladas en Sowft o Objective-C para las aplicaciones de iOS y macOS, C# para Windows y Java o Kotlin para Android. A pesar de que se admiten diferentes lenguajes para cada sistema operativo, eso no implica una compatibilidad entre sistemas, es decir, si una aplicación ha sido desarrollada en Kotlin, como es el caso de este proyecto, no será funcional en iOS.

Poseen la ventaja estar optimizadas para maximizar el aprovechamiento de los recursos del sistema operativo, por lo que suelen tener un mejor rendimiento, y pueden estar acorde a una mejora de la Experiencia de Usuario (UX), ya que siguen las directrices del sistema operativo.

Por otro lado, requieren un conocimiento del lenguaje en el que se programen, puesto que pueden crearse versiones de sistemas operativos que hagan que sea necesario el uso de modificaciones en código para el correcto uso de la aplicación, así como el coste que implica esto.

3.5. Extensiones de archivos de audio

Android, en la actualidad, permite la compatibilización de casi cualquier tipo de archivo de audio en sus aplicaciones. Algunos de los más comunes son los enumerados a continuación.

Uno de los formatos de audio más populares, no solo para la reproducción en Android, es el MP3 (MPEG-1 Audio Layer 3). Se trata de un formato de audio comprimido con pérdidas, ya que elimina el sonido al que no alcanza el oído humano, enfocándose así en los datos reales. Es usado en casi todos los dispositivos compatibles y reduce el tamaño del archivo original.

De una alta calidad, mayor que el formato MP3, está el AAC (Advanced Audio Coding), utilizado para el almacenamiento y la transmisión de audio. Es un formato de audio comprimido con pérdida utilizado para aplicaciones de iOS y Youtube entre otras. A pesar de esto, es menos popular.

También considerado un formato de audio comprimido con pérdida, se tiene el OGG, Ogg Vorbis, de alta compresión, incluso más que para el formato MP3, compatible

con la mayoría de los dispositivos Android y de calidad peor que los anteriormente mencionados.

Como archivo compatible con un formato de audio sin comprimir, se encuentra el WAV (Waveform Audio File Format), son los más grandes en tamaño debido a no tener pérdidas. En la actualidad se usa para el almacenamiento de grabaciones en Compact Disk (CD).

Con una alta calidad, similar a la del WAV, pero con un archivo más pequeño, se encuentra FLAC, Free Lossless Audio Codec, que se trata de un formato de audio sin pérdidas. Se posiciona como una alternativa popular del formato MP3 [5].

En al Anexo A.3 se encuentran los formatos y códecs de audio compatibles con Android Studio.

3.6. Ganancia frente a volumen

Según la Real Academia Española (RAE), la palabra ganancia está definida como la acción y efecto de ganar [6].

En audio, la ganancia es una relación entre dos estados, en este caso, las potencias de salida y entrada [7]. Es comúnmente confundida con la definición de volumen, que, entre sus definiciones por la RAE, es considerado como la intensidad del sonido [8]. Mientras que la ganancia se mide mediante los decibelios (dB), el volumen se trata de una percepción subjetiva de la potencia del sonido.

A medida del incremento en la ganancia, se ve incrementada también la distorsión, debido a que la señal de entrada para a ser menor que la señal de salida. Por el contrario, con la subida del volumen, la señal no se distorsionará.

Por esta razón, la ganancia se utiliza para la jugar con la distorsión deseada y después se modificará el volumen a gusto del usuario [9].

4. Mesas de mezclas

Una mesa de mezclas, también conocida como consola de mezclas o mezcladora, es un dispositivo electrónico, al que se le conectan elementos, que permite al usuario tener control sobre las señales sonoras que pueden ser procesadas para obtener una mezcla de audio.

4.1. Historia

No existe una patente exacta de la primera mesa de mezclas. Sin embargo, a raíz de solucionar diferentes problemas que han surgido a lo largo de la historia, se fue llegando a lo que se terminó conociendo como mesa de mezclas analógicas, en la actualidad, casi extintas debido a las digitales.

Para llegar a este desarrollo, es necesario echar la vista atrás, a principios del siglo XX. Durante 1922, en la estación WEAF, con sede en Nueva York, los ingenieros de AT&T utilizaron consolas para la nivelación y combinación de señales de los micrófonos de la estación radial.

Unos años más tarde, en 1932, los ingenieros de la BBC diseñaron sus propios mezcladores para los estudios situados en Londres, Reino Unido, primero en Savoy Halley y más adelante en los estudios Broadcasting House, en Portland Place. Lugar en el que se lanzó la primera emisión de radio, y aún en uso en la actualidad [10].

Fig. 3. Broadcasting House en Portland Place, Londres [11].

Los empleados de Bell Telephone, guiados por Harvey Fletcher, probaron los experimentos de Fletcher sobre la transmisión sonora en tres canales, instalando un mezclador encargado de distribuir las señales de los tres micrófonos situados frente a la Orquesta Filarmónica de Philadelphia, en 1933. La patente del control panorámico de sonido fue en 1939, por el ingeniero William Garrity, por el sistema de Fantasound para la película de Fantasía, de Disney.

Sin embargo, la comercialización de los primeros mezcladores, más conocidos como mesas de mezclas, no fueron hasta mediados de los años 40, que, dejando de lado los *mixers* de soporte técnico de radioemisoras hizo que la firma Presto, en 1947, desarrolló los primeros mezcladores de audio especializados en la grabación de discos, más conocidos como los modelos 90A y 90B. En Chicago, en el año 1950, el ingeniero Bill Putnam fundó la compañía Universal Audio, y fue el encargado, junto a Paul McManus en 1959, en crear la primera consola modular, que estaba basada en el amplificador operacional UA 610, utilizado en sesiones de grabación de diversos artistas de la época en Universal Recording.

Un año más tarde, en 1951, se grabó la canción que es denominada la primera grabación de Rock&Roll, con el dinero de dicho tema, el productor, Sam Phillips, remodeló su estudio de Tennesse, donde modificó una consola RCA 76D, de los años 30, añadiéndole reverberación. Gracias a este cambio, surgieron varios temas de artistas como Elvis Presley o Johnny Cash.

A finales de la década de los 50 y comienzo de los 60, las producciones comenzaron a tener ingenieros de sonido que construían consolas ajustándose a las especificaciones que pedían los estudios de grabación, como, por ejemplo, la consola que diseñó David Gold para Goldstar Studios, que más adelante fue muy usada por el productor Phil Spector, que fue el encargado de desarrollar la técnica definida como “Muro de sonido”, que revolucionó la industria de la música [12].

A partir del legado de Spector con su fórmula de producción, basada en la creación de una base rítmica sólida apilando cada uno de los instrumentos en capas, uno sobre otro, y acoplando las voces y posibles arreglos necesarios, la producción musical dio un paso al siguiente nivel, puesto que consiguió un sonido muy limpio que se escuchaba sin problemas a través de las jukeboxes y las radios.

En 1964, Rupert Neve, un ingeniero británico, desarrolló en su empresa, Neve Corporation, la primera mesa de mezclas transistorizada, con un ecualizador, pasando de

las de válvulas que eran utilizadas hasta la fecha. Continuó con la creación de diferentes consolas y preamplificadores que hoy en día siguen siendo utilizadas [13].

A finales de los años 70, Neve, junto con BBC, consiguieron que la aplicación de las mesas de mezclas fuera sencilla, gracias al diseño de los sistemas de ecualización y filtrado, además de la aplicación de los controles. Esto llevaría un paso más allá la futura digitalización. La misma empresa creó la primera mesa de mezclas completamente digital, con el modelo Neve DSP, en 1981, todo el procesamiento de la señar digital se hacía en el dominio digital y la señal final podía ser de forma digital, utilizada para grabación, o podía ser convertida a analógico para la radiodifusión. Se dice que fue la mesa de mezclas más cara y compleja que se ha fabricado hasta la fecha [14].

Fig. 4. Modelo Neve DSP, 1981 [15].

Hasta entonces no se comenzó con la digitalización, ya que las mesas de mezcla de sonido profesional eran analógicas, lo que quiere decir que la señal atravesaba los procesos a realizarse a través de los canales, precisando así muchos canales y procesadores físicos e integrados o externos conectados por *routing*, lo que conllevaba a tener muchos cables que dificultaba su uso. A pesar de ello, no fue todo fácil, ya que, con este proceso, se encontraron problemas nuevos como la pérdida de bits, la memoria necesaria, o el efecto *clipping*, una forma de distorsión de onda que conlleva una salida desagradable para el oyente.

Con la digitalización, el trabajo pasó a estar enfocado en introducir el software en las mesas de mezclas a través de diferentes módulos y capas que cubrían las entradas y

salidas, teniendo así procesadores que fueran manipulables en un solo canal. Estos cambios consiguieron la reducción de los costes, además del espacio ocupado por los equipos [16].

Fig. 5. Evolución de las mesas de mezclas.

4.2. Tipos de mesas

Un estudio de producción es el lugar en el que se desarrolla todo el trabajo, en ocasiones, también puede servir como estudio de realización. El tamaño de la sala puede variar en tamaño dependiendo de la finalidad y el equipo que se use, aunque, por lo general, incluye como mínimo un micrófono, un magnetofón, procesadores de señal, altavoces, auriculares, control remoto, panel de conexiones, reproductor de discos compactos, aunque esto, con las nuevas tecnologías puede llegar a ser obviado, y, en lo que se va a profundizar a lo largo de este proyecto, una consola, más conocida como mesa de mezclas, siendo el elemento central de las conexiones entre los diferentes equipos de sonido.

Las mesas de mezclas pueden ser diferenciadas por su tecnología o su funcionalidad, a pesar de ello, todas trabajan de la misma manera, por lo que, sabiendo el manejo base, la única dificultad será las pequeñas discrepancias que haya entre modelos y marcas [17].

En primer lugar, comparando la tecnología utilizada, se puede encontrar:

- **Analógicas**. Son la base del resto de mesas, aunque se han visto sustituidas casi en su totalidad, en ellas, tanto las entradas como las salidas de la señal son analógicas. Trata señales continuas con variaciones de voltaje sin sufrir conversión alguna, también de forma analógica, es decir, los controles son de forma directa sobre las señales. No es necesario ningún *software* y carecen de muchos efectos por lo general, como por ejemplo puertas de ruido, compresores limitadores o eco.
- **Digitales**. El relevo generacional de las analógicas y apariencia similar. En este caso, las entradas y las salidas pueden ser analógicas o digitales, aunque el procedimiento al que se ve sometido el sonido se realiza en el dominio temporal. Suelen tener diferentes módulos repartido para hacer las diferentes acciones necesarias, como por ejemplo el encargado del procesamiento, llamado “DSP” (Digital Signal Processor), similar a las analógicas, o el módulo de interfaz, que contienen los convertidores de analógico a digital. Necesitan un software para ser controladas y pueden conectarse a dispositivos tales como ordenadores, *tablets* o incluso móviles.
- **Virtuales**. Se trata de un software con funciones limitadas en comparativa con las anteriores, se trata de un caso más simple, como puede ser este

proyecto, en el que se necesita un dispositivo electrónico, como por ejemplo un ordenador, para poder desempeñar la función deseada.

Cuando la comparativa se basa en la funcionalidad, se pueden encontrar: [18][19]

- **Split, o básicas**. Utilizadas de manera común en grabaciones estéreo o en radiodifusión televisiva.
- **In line**. Empleada en estudios de grabación cuando es necesario tener una salida por cada uno de los canales para la grabación y reproducción de un magnetófono multipista, que es aparato que permite la grabación de sonido que registra varias fuentes sonoras para más tarde unirlas en un solo sonido.
- **Monitores**. Es la que se puede observar en actuaciones en vivo o en los platós, cerca de donde esté el escenario principal, es la encargada de proporcionar sonido a los altavoces y a las personas que estén en el escenario, teniendo la capacidad de guardar diferentes escenas que podrán ser usadas.
- **Mesas de DJ**. Encargada de la modulación y ecualización de una señal de sonido, independientemente venga desde un micrófono o un altavoz. Suelen ser más manejables y trabaja de manera indiferente con señales digitales o analógicas.
- **Autoamplificadas**. Poseen una salida que permite conectar los altavoces de manera directa, llevan integrados amplificadores y son conocidas también como equipos de voces. Son algo más complejas de utilizar al tener todo lo necesario para una sonorización en un solo aparato.
- **Portátiles**. Alimentadas mediante baterías o una tensión continua exterior, son de menor tamaño y usadas para grabaciones de menor calibre o que no necesitan una potencia mayor.

4.3. Partes

A pesar de existir diferentes tipos, las mesas de mezclas poseen partes comunes a todas ellas y otras que cambian en función del tamaño, precio o función a desempeñar. Las mesas de mezclas están formadas por diferentes columnas, conocidas como canales,

cada una de estas está formada por diferentes *switches*, perillas y *faders*, que son los encargados de las modificaciones que sufrirá el sonido a su paso por ellos.

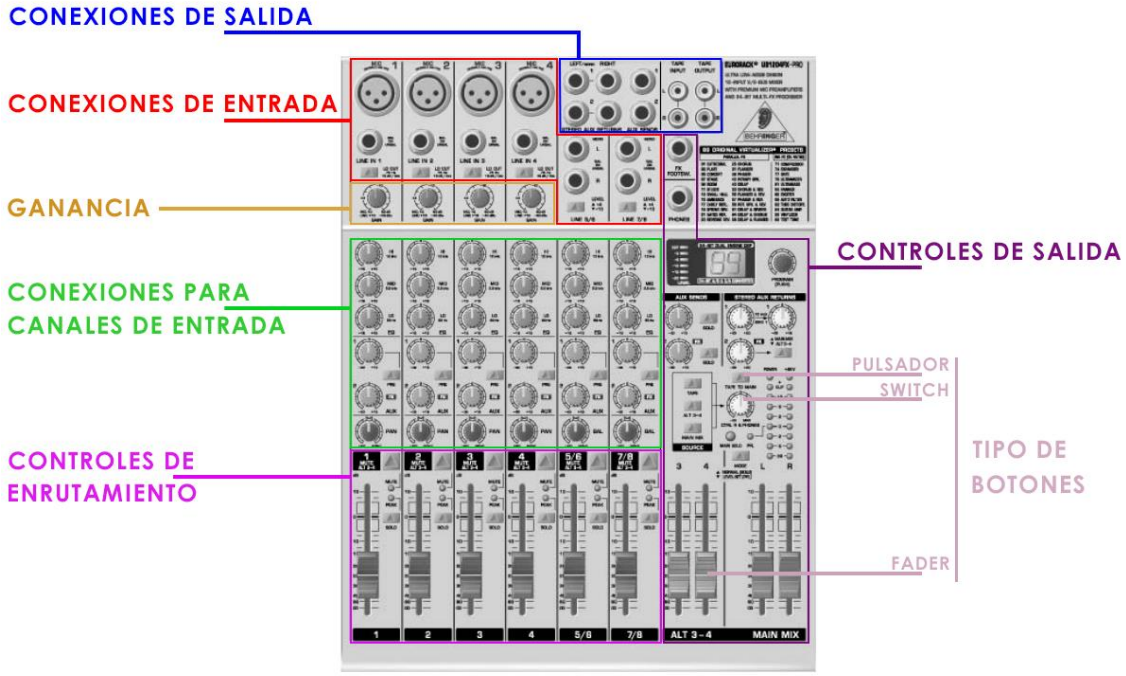

Fig. 6. Explicación de las partes de una mesa de mezclas [20].

Comenzando de arriba hacia abajo y de izquierda a derecha, lo primero que se encuentran son las conexiones de entrada, estas, en ocasiones, también se pueden encontrar en la parte posterior del panel. Pueden ser conectores XLR-Canon, indicados para micrófonos, o Jack-Plug (TRS), que son entradas de línea, siempre serán conexiones hembra. La entrada XLR, frente a la de línea, se diferencia en que posee un preamplificador, lo que permite que no llegue un sonido tan reducido como sería al conectarlo en otra entrada. También existen otras entradas digitales para la comunicación con tarjetas de audio.

Justo debajo, se encuentran los filtros de entrada o corte, son los encargados de cortar las bajas frecuencias que pueden ensuciar el ruido, puesto que las frecuencias entre los 20 Hz y los 80 Hz no son apreciables al oído humano, por lo que estos switches, dependiendo del modelo, pueden cortarlas entre los 60 u 80 Hz.

Continuando por la derecha, se encuentran las conexiones de salida, entre las que hay que destacar la salida Máster, también conocida como *Main*, que, como su propio nombre indica, es la principal de una consola, se tratan de salidas estéreos de dos canales, uno izquierdo y uno derecho. Dependiendo de la utilidad de la mesa de mezclas, esta salida servirá para la salida de un programa, grabar una señal o como salida hacia los

amplificadores. Las salidas de subgrupos, o alterna, se trata de una segunda señal, puede ser muy útil para grabar o monitorizar cuando no se quiere utilizar la salida Máster al no necesitar la señal sumada. En el caso de carecer de la salida de subgrupo, la consola podrá tener una salida definida como *Tape Out*, con conectores RCA (Radio Corporation of America), para el envío de la mezcla que sale de la consola. Por último, en este apartado de salidas, está la encargada del monitoreo, llamada Control Room, que es la que modifica el sonido para poder dejar la salida Máster en 0 y no cometer errores o que pueda introducirse ruido.

La primera perilla que se encuentra, en ocasiones de menor tamaño, corresponder a la ganancia. Es la encargada de potenciar el sonido cuando este es insuficiente y el *fader*, explicado más adelante, ya se encuentra en 0 dB. Si el potenciómetro modifica el valor dejando el *fader* por debajo de los 0 dB, no solo se aumentará la ganancia, sino que también lo hará el ruido.

En la siguiente posición se encuentran los ecualizadores, que sirven para modificar las frecuencias altas, medias y bajas o graves, designadas como HI, MID y LO respectivamente. Los valores que pueden tomar, por lo general, son entre -15 y +15 dB. Para algunos modelos, las frecuencias medias poseen dos perillas, una encargada para la ganancia, como se ha mencionado, y otra para la frecuencia, que suele ir entre 100 Hz y 8 KHz.

Los siguientes potenciómetros que se encuentran en el canal son los auxiliares, por los cuales se selecciona el volumen por el que saldrá la salida auxiliar, como su propio nombre indica. Esto permite que, si se quiere potenciar un canal por encima de otros, se pueda modificar aquí, dando, por ejemplo, importancia a la voz que puede salir de un micrófono conectado en el canal 1, frente a diferentes instrumentos conectados al resto de canales. En función de cómo sea la consola, podrá existir un auxiliar para PRE y otro para POST, o que, para PRE, exista un switch. El sonido mandado por PRE no sufre ninguna modificación de ecualización o del volumen del *fader*.

El panorámico, designado como PAN, es una perilla que se encargada de indicar a qué canal se enviará la señal, teniendo hasta 3 opciones, al izquierdo, al derecho o ambos.

El siguiente switch tiene dos funcionalidades, *MUTE*, que permite silenciar el canal, y ALT 3-4, que si se mantiene pulsado junto con otros canales serán enviados a la salida con el mismo nombre para grabar solo esas pistas.

El botón PFL o SOLO, que realiza una escucha previa de la señal, sirve para la prueba de sonido de cada uno de los canales, la escucha de ellos será por auriculares o

por la señal de Control Room. En algunos modelos, si esta opción no está activada pero el indicador luminoso que acompaña al switch está encendido, servirá para medir la sobrecarga, que indica que la señal es demasiado fuerte y será necesario bajar la ganancia o el *fader*.

Al final del mezclador se encuentra el *fader*, es la parte que abarca más espacio por su forma alargada, aunque en ocasiones puede ser una perilla más. La traducción de su nombre indica la atenuación del canal. Está acompañado por una escala de dB en valores negativos y solo unos cuantos positivos, esto se debe a que lo óptimo es que sea colocado en 0 dB, lo que indicará que la totalidad de sonido que sale por lo que esté conectado a la entrada de dicho canal está llegando. Si las salidas de los equipos conectados son muy altas, se podrá disminuir este valor.

Por último, la parte derecha de la mesa de mezclas son los controles de salida. En primer lugar, se encuentra la entrada para los auriculares, seguido de los controles de salida, que son prácticamente iguales a los mencionados anteriormente, aunque en este caso hay dos *faders* para *Main*, ya que, como se ha mencionado anteriormente, suele ser estéreo, es decir, tiene un *fader* izquierdo y uno derecho, y también los *faders* para los subgrupos. También están los indicadores del nivel de saturación, conocidos como vumeter, que lo indican mediante colores, ya sea en una escala de color de verde a rojo, pasando por amarillo, de menos nivel a más respectivamente, o, dependiendo del modelo, simplemente permaneciendo apagado y encendiéndose si satura [21][22].

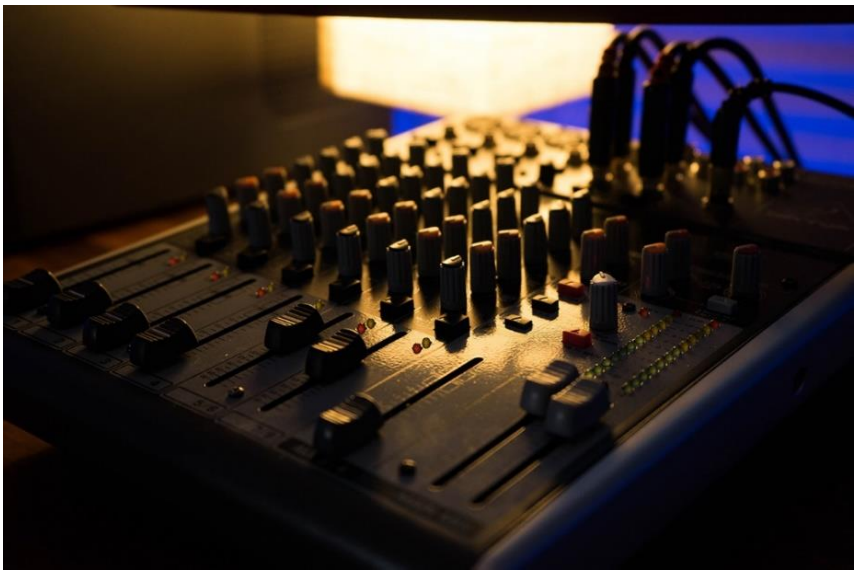

Fig. 7. Mesa de mezclas [23].

5. Desarrollo

Como se ha mencionado con anterioridad, Android Studio permite la programación en diferentes lenguajes, en este caso la elección tomada fue realizar el prototipo con Kotlin, de ahí también el nombre escogido, Audio Kotlin.

5.1. AndroidManifest.xml

Dentro de la raíz del proyecto, el archivo AndroidManifest.xml es el archivo que permite disponer de una lectura de la información imprescindible que realiza la aplicación. Desde los permisos que necesita para el correcto funcionamiento, en este caso, el poder acceder a los archivos del dispositivo a la hora de meter un audio y no usar los que se encuentran por defecto, como aspectos más básicos, por ejemplo, la orientación en la que se muestra la aplicación, que está forzada a la verticalidad.

Se trata de un archivo que se crea por defecto una vez se comienza el proyecto, y es la persona encargada del desarrollo quien puede añadir, quitar o cambiar diferentes atributos, siendo algunos de ellos necesarios siempre, como por ejemplo la información de la *app* (nombre, icono…) o de donde se toma la información del código, en este caso, .*MainActivity.kt*, archivo del que se hablará más adelante.

5.2. Activity_main.xml

Para la realización de una aplicación, existen diferentes archivos con los que trabajar, en este caso el archivo llamado *activity_main.xml* trata la interfaz de usuario, que puede ser encontrado en la carpeta *layout* (diseño), que a su vez está en la carpeta res, que viene de *resources* (recursos). Puesto que en la *app* propuesta existe exclusivamente la vista vertical, solo se posee un archivo con dicho nombre. En el caso de que haya más de una vista, ya sea por tener la posibilidad de apreciar la aplicación en horizontal, o porque exista más de un dispositivo compatible, se podrá tener más de un archivo con dicho nombre.

Como se ha mencionado con anterioridad, Android Studio permite al usuario trabajar en la parte visual, ya sea mediante código o arrastrando hacia la pantalla del dispositivo virtual desde el apartado de *Palette* (paleta). También es posible habilitar

ambas vistas a la vez gracias a la opción *Split*, que coloca el código a la izquierda y el diseño a la derecha por defecto.

Esta fue la parte con la que se comenzó el proyecto en Kotlin, en la Fig. 8 se muestra la parte visual. Trabajando primero en la parte izquierda, las pistas se fueron creando de manera vertical, para más adelante crear la parte derecha, que es lo que se conocería como *Main* en una mesa de mezclas real, la parte donde se juntan las pistas y se puede alterar la ganancia final.

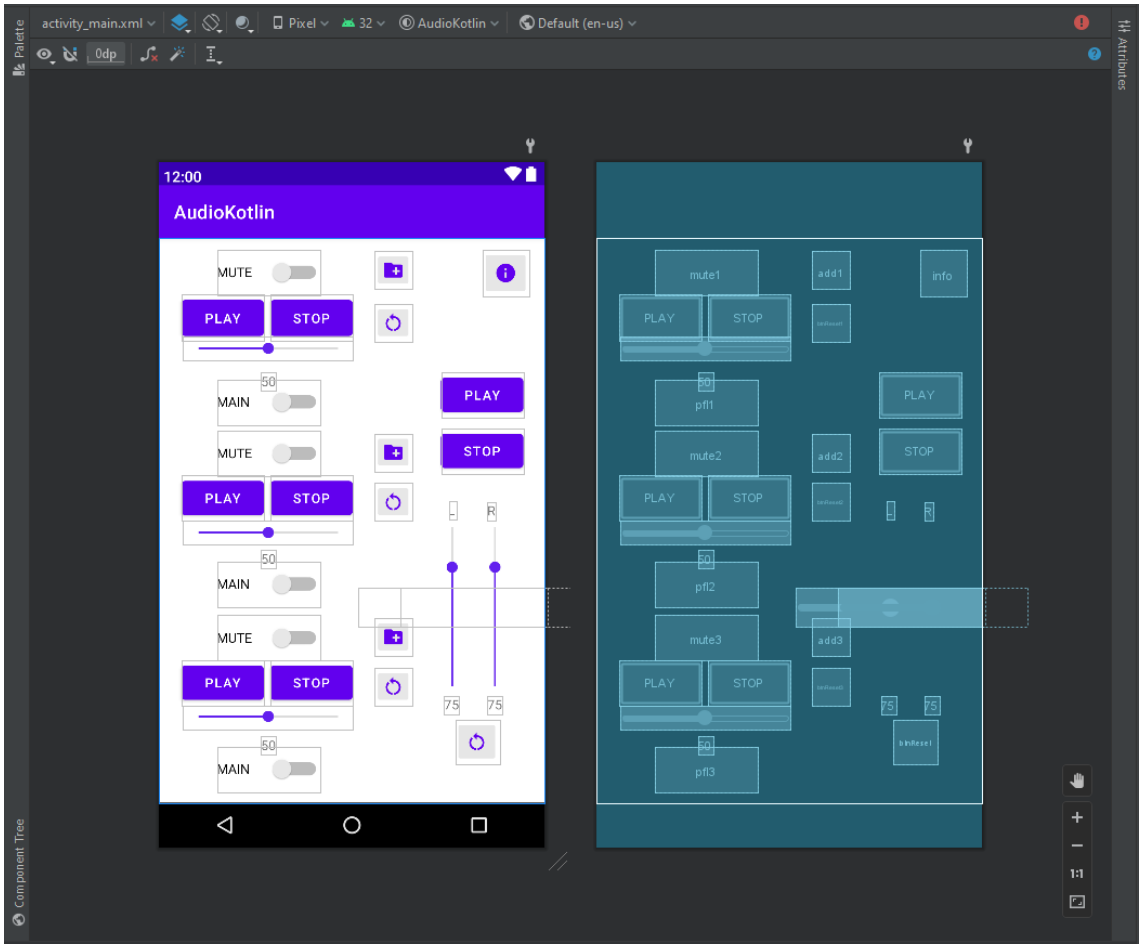

Fig. 8. Diseño en Android Studio del prototipo.

Las partes disponibles en el *Constraint Layout*, la clase de tipo *View* sobre la que se construyen las variables para el uso del usuario, se encuentran:

- ***Switch***: se trata de un objeto con dos posibles resultados, activado o desactivado. En ningún momento se pueden dar ambos casos a la vez. Para este caso aparece en dos ocasiones en cada pista, para *Mute* y para *Main*, ambos explicados más adelante.

- ***Button***: de los cuales existen a su vez dos tipos:
- **Botón con texto**, como su propio nombre indica, posee la actividad que realiza mediante texto plano y se utiliza tanto para *Play* como para *Stop*.
- **Botón de imagen**, en este caso con iconos que indican la acción, se pueden observar con la información, la carga de pistas de audios provenientes del dispositivo y para reiniciar la pista.
- ***Seek bar***: también conocidas como barras de progreso, funcionan con valores enteros entre el 0 y el 100 en este proyecto. Son equivalentes a la ganancia que se encontraría en una mesa de mezclas. Para las pistas individuales presenta un valor por defecto de 50, y para la salida Máster de 75 para cada caso, izquierda y derecha. Aparecen con la vista horizontal y vertical.
- ***Text view***: texto plano introducido para la mejor comprensión del usuario. En este caso, el texto fuera de las partes mencionadas está directamente relacionado con las ganancias, ya sea para indicar el valor en el que se encuentra la *seek bar*, o para diferenciar entre el lado izquierdo y el lado derecho.

En la Fig. 9 se muestra el flujograma correspondiente a las constantes mencionadas a lo largo del apartado.

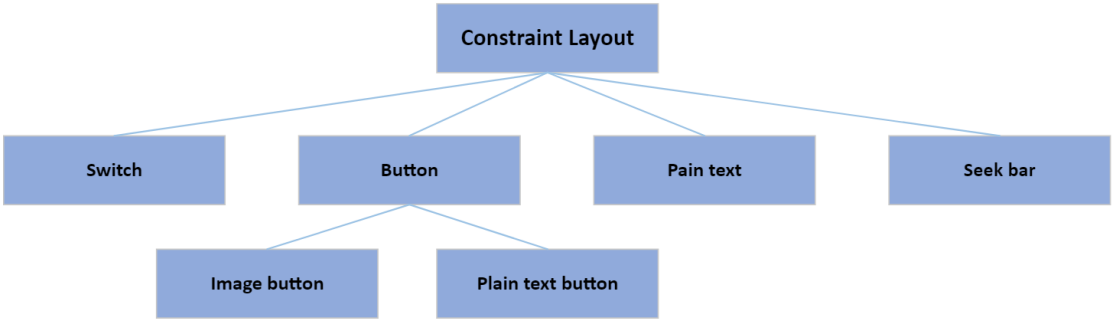

Fig. 9. Diagrama de constantes en archivo .xml

5.3. MainActivity.kt

En lo que se refiere a código, en la carpeta java se encuentra el archivo MainActivity.kt, que es el archivo de Kotlin que posee el desarrollo de las diferentes funciones que permiten el uso de la *app*.

Se comienza importando los diferentes paquetes y clases que son necesarias para el correcto funcionamiento del proyecto y se van a utilizar a lo largo del código. Le sigue la clase que comparte nombre con el archivo, MainActivity, donde se expone el resto de código. En la Fig. 10 se muestra un diagrama correspondiente a las diferentes funciones que se pueden encontrar, además de las variables declaradas.

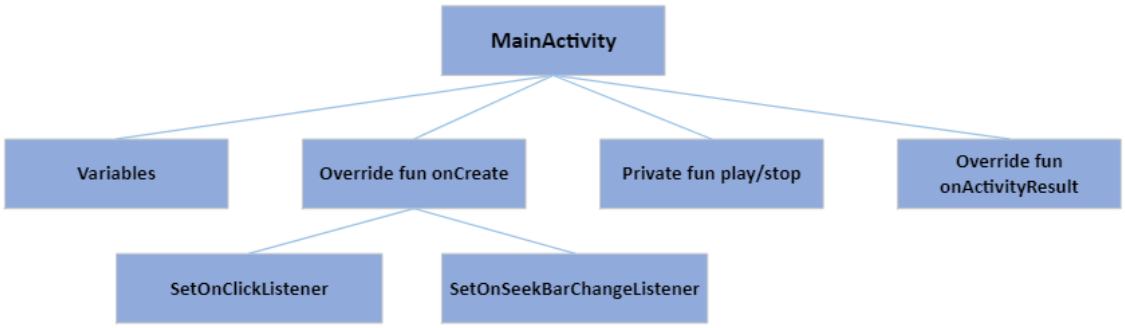

Fig. 10. Diagrama de variables y funciones de MainActivity.kt

5.3.1. Variables

Lo primero que se encuentra tras la definición de la clase, son las variables. Existen diferentes tipos de variables, las que vienen heredadas desde el archivo Activity_main.xml, mencionadas en el punto anterior y mostradas en la Fig. 7, tomadas como variables privadas, y las que son creadas de forma específica para esta parte del código. Las variables pueden ser inicializadas o no, en la creación, en función del tipo que sean, como por ejemplo si son de tipo numérico, como por ejemplo los valores de las barras de ganancia mostradas en las diferentes pistas, o de si se trata de un caso de verdadero o falso con un *boolean*. Hay algunas variables, como por ejemplo los botones, que no son necesarios inicializarlos, puesto que se trabaja con la pulsación del usuario en él y mirando en ese punto el estado.

Un caso de variable inicializada se puede encontrar con el caso de ver si una pista está siendo reproducida o no, se puede encontrar en el código como *private var booleanPlay1 : Boolean = false*, que lo que indica es que es de tipo booleano y que de primera instancia se tomará la variable *booleanPlay1*, como falso, lo que significa que no se está reproduciendo, cosa que viene explicada más adelante debido a diferentes casuísticas.

Para una variable sin inicializar, se puede observar el de *private lateinit var btnPlay1 : Button*, que muestra una variable de tipo botón que no da más información en este punto del código.

5.3.2. Override fun onCreate

El método en el que se encuentran casi todas las acciones de la mesa de mezclas se denomina *onCreate*, que es ejecutado por primera vez al crear la actividad tras el lanzamiento de la aplicación, solo se ejecuta en este punto. La etiqueta de *override*, indica que puede ser sobrescrito si es necesario a lo largo del código.

Se comienza con la función que se encuentra dentro de esta clase de *setContentView(R.layout.activity_main)* se indica al código que debe escoger ese archivo como parte gráfica. Esto es importante en los casos en los que hay más de una vista, puesto que permite diferenciar cual será la principal y cual la secundaria, por ejemplo, en el caso de que se cambie la posición del dispositivo, de vertical a horizontal.

Se continúa con dar valor a las variables comentadas anteriormente, esto se hace recuperando los puntos creados en el archivo Activity_main.xml, esto se hace mediante *findViewById*, tomando el ID que se ha dado en el archivo XML.

Tomando uno de los ejemplos mencionados en el apartado anterior, para reproducir en la pista 1, es necesario pulsar el botón denominado como *play*, esto es posible porque en el archivo XML se ha definido un botón con un id de 'btnPlay1', y de la misma forma en el archivo MainActivity.kt, por lo tanto, con *btnPlay1 = findViewById(R.id.btnPlay1)* recupera lo que en este caso es un botón para poder llamarlo a lo largo de las funciones y que el usuario pueda realizar la acción de reproducir el audio cuando el resto de casuísticas lo permitan.

Dentro de cada una de las funciones, se pueden encontrar otras para que el usuario pueda realizar la decisión deseada.

SetOnClickListener

Puesto que la mayoría de los objetos con los que el usuario pueden interactuar son botones, la función *setOnClickListener* es la más repetida a lo largo del código. Permite controlar qué hace el usuario y en función de la acción, lanza la consecuente al botón pulsado. Se utiliza para reproducir un contenido a través de los botones de *play*, se para con *stop*, se puede añadir un audio nuevo con *add*, o reiniciar con *reset*. A pesar de no

tener la misma forma visual, los botones de silenciado de pista, *mute*, también funcionan de la misma forma.

- *Play*: cuando se busca reproducir un audio, ya sea por defecto o no, se activará el booleano de la pista y se observará si el resto de las pistas están siendo reproducidas, si estas respuestas son negativas, entonces reproducirá el audio. Además, se observará el nivel de volumen de la barra para la reproducción. Tomará la función de *play* explicada posteriormente para saber qué audio reproducir. En el caso de la salida máster, se observará si la salida para *main* está activa en cada una de las pistas para mezclar las seleccionadas.
- *Stop*: para el audio de la pista o pistas que se estén reproduciendo en ese momento. En el caso de las pistas, se llama a la función privada con la que comparte nombre mencionada más adelante.
- *Mute*: esta opción solo está disponible para las pistas, se observa si el audio se está reproduciendo, y en el caso de que el botón esté activo, el volumen se pone a 0, para silenciar la pista, y en el caso de que no, tomar el volumen de la barra.
- *Reset*: para las pistas, se ajusta el volumen en 50, que es el valor por defecto, se elimina, en el caso de que haya sido añadido, el audio que se haya puesto desde el dispositivo para que vuelva a estar el que existe por defecto y, en el caso de estar activa, se elimina la opción de *mute*.
- *Add*: si el usuario no quiere usar ningún audio de los que están introducidos por defecto en la aplicación, gracias a esta opción puede obtener cualquier sonido que posea una extensión admitida y se encuentre en el dispositivo, o sea grabado en ese momento con la grabadora. Para este caso, se indica de que se trata de un audio y se llama a la petición de la función *onActivityResult*, explicada más adelante.
- Información: En la parte superior derecha, hay un botón que permite mostrar un breve manual de uso mediante un *pop-up* que para cerrarlo se puede realizar con la opción de “aceptar” que aparece abajo a la derecha.

SetOnSeekBarChangeListener

Para el caso de las *progress bar* que se utilizan para las ganancias, se usa la función *setOnSeekBarChangeListener*. A su vez, tiene diferentes opciones que permiten ver si ha habido algún cambio de valor.

Para el volumen, se obtiene el valor de la barra, se multiplica por 0,01 y se pasa a tipo Float, para que funcione como un porcentaje.

De manera adicional, aparece un *toast* en la parte inferior de la pantalla que indica el nuevo valor que toma la barra.

5.3.3. Private fun play y Private fun stop

Estas funciones se utilizan fuera de *onCreate,* y permiten introducir el audio en la pista. Se tratan de funciones privadas que las visibiliza solo en el archivo que las tiene declaradas. Para este caso existen dos opciones, para el caso de la reproducción y de parar cada una de las pistas.

- *Play*: crea la pista de audio, si se usa el valor por defecto, coge el archivo que está disponible en la carpeta de *raw* en la que se encuentran los diferentes recursos que se pueden utilizar. Si no, obtiene la URI del audio que será usado desde cualquier ubicación de almacenamiento del dispositivo. Además, activa el audio usado en bucle y lo activa para que cuando sea pulsado el botón *play* y se llame a esta función, el sonido comience.
- *Stop*: observa si se reproduce sonido, y si es así, lo para.

5.3.4. Override fun onActivityResult

Por último, para poder obtener los audios desde cualquier parte de memoria del dispositivo móvil, o incluso desde la aplicación de grabadora que viene instalada en ellos. Puesto que se tienen tres pistas, a cada una de ellas se le otorga un número como petición que será usado posteriormente con el botón llamado *add* disponible en cada una de las pistas.

Por otra parte, en el flujograma de la Fig. 11 se puede observar el flujograma correspondiente a la lógica seguida por la salida máster, o principal, de la aplicación.

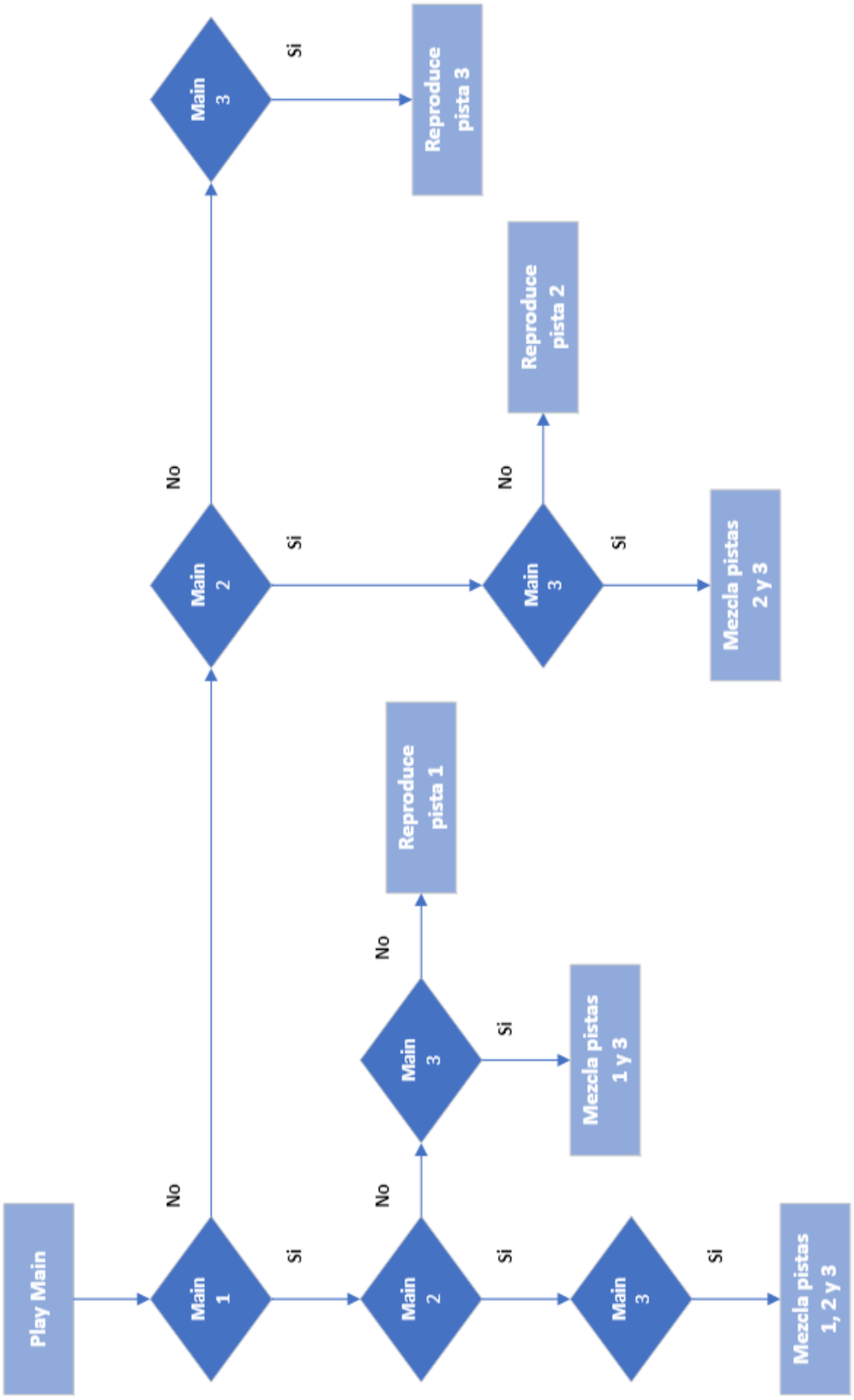

Fig. 11. Flujograma de la salida máster.

6. Resultados

Los diseños por los que ha pasado el prototipo explicado a lo largo del proyecto, fue pasando por diferentes modificaciones hasta llegar al que se puede observar en la aplicación final.

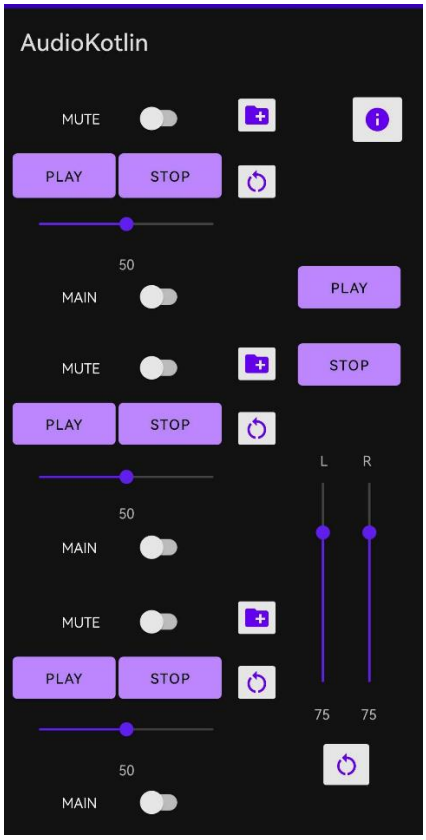

Fig. 12. Resultado de la aplicación creada

En primer lugar, la decisión de limitar el proyecto únicamente a dispositivos móviles ha hecho que el número de pistas desarrolladas sean 3, dado al espacio limitado que se posee y a la capacidad del usuario al uso de los botones o sliders.

Debido a la misma razón, se ha decidido prescindir de características de las mesas de mezclas originales, como puede ser los filtros de altas, medias y bajas frecuencias, ya que se ha optado por un diseño minimalista

En la Fig. 12 se muestra el resultado final al que se ha llegado tras el trabajo realizado. Una versión simplificada de una mesa de mezclas para dispositivos móviles.

Se trata de tres pistas con audios introducidos por defecto, los cuales pueden ser sustituidos por pistas de audio obtenidas de la memoria interna del dispositivo o incluso mediante una grabación que se realice en el momento.

Para la validación del resultado final del producto, primero se tuvieron que probar las diferentes partes por las que se compone la aplicación. En la carpeta de Github en la que se encuentra la aplicación, el archivo y esta memoria, también se puede encontrar un vídeo a modo demostración (https://github.com/mdelaosa/Memoria_TFG).

6.1. Creación de las pistas

En primer lugar, se ha trabajado con el uso de una pista, comprobando la funcionalidad de los botones introducidos para reproducir (*play*) y parar (*stop*) los audios que se encuentran introducidos por defecto en la aplicación.

Además de las implementaciones del botón de *mute* para silenciar la pista y del *fader* que funciona a modo de ganancia final de la pista, este caso incluye también el valor numérico que varía en función del que tiene la barra utilizada por el propio *fader*.

6.2. Salida Máster

Para la comprobación de la salida máster o principal, es recomendable el uso de auriculares, puesto que permite diferenciar la salida estéreo con más calidad que mediante los altavoces integrados en el dispositivo.

En este caso, la salida Máster se compone de los botones para reproducción y parar al igual que para las pistas, dos *faders*, en vez de solo uno, que se separa en L y R, que corresponden a izquierda y derecha.

Además, para poder llevar a cabo esta implementación, en las pistas se añadió un botón, que por defecto está desactivado, para que cuando se active, la salida salga y dependa de la máster, y no de cada una de las pistas, como se mostraba en el apartado anterior.

La validación que se ha llevado a cabo en este punto ha sido con la reproducción de cada una de las pistas por separado teniendo el botón de la salida máster activado, así como con la combinación de las diferentes casuísticas que se podrían dar.

6.3. Información

Para facilitar la utilización de la aplicación para el usuario final, se llegó a la conclusión que añadir un botón en la parte superior derecha sería lo más adecuado.

Se trata de un botón que al ser pulsado abre un diálogo con las instrucciones de uso, así como un botón en la parte inferior derecha, con el que se cierra.

6.4. Botones extra

Por último, para la mejora de la aplicación, se decidieron añadir dos botones en cada una de las tres pistas disponibles, para añadir un audio y para reiniciar los valores por defecto, y uno en la salida máster, solo para reiniciar los valores de las ganancias.

El botón de añadir audio permite al usuario no utilizar el sonido por defecto que tiene la aplicación, cogiendo así uno que esté en la librería de su teléfono, o mediante una grabación que se realice con el dispositivo. Este tipo de botón se encuentra solo en las tres pistas, puesto que la salida solo es una combinación de éstas.

El botón de reinicio de ganancia se encuentra tanto en las entradas como en la salida, aunque tienen ligeras deferencias. Para las pistas, el reinicio se lleva a cabo tanto en la ganancia, poniéndola en el valor por defecto, como en la desactivación del botón que silencia la pista si es que se ha activado previamente. Para la salida, solamente vuelve a poner el valor de las ganancias izquierda y derecha en el valor del que se parte al iniciar la aplicación.

7. Costes

Es necesario tener una idea aproximada de los costes que supone la creación de una aplicación Android. En este caso, para la aplicación de Audio Kotlin realizada, los cálculos serían los siguientes.

Para los requisitos mínimos de Android Studio, se necesita un ordenador de 64bits que posea 8 GB de RAM, así como una resolución de 1280x800, de ahí hacia arriba es especificaciones es todo válido [24]. En la actualidad, se pueden encontrar ordenadores con estas características desde los 350€ aproximadamente. Debido a que se trata de un software libre, no es necesario un coste extra con respecto a las licencias.

Además, es necesario tener un dispositivo Android, la gama media se encuentra a partir de los 200€ aproximadamente. En este caso, no es necesaria ninguna especificación concreta, puesto que todos los dispositivos pueden grabar, guardar y reproducir audio. Por lo que se tomará éste como valor de referencia.

Para obtener una experiencia de usuario mayor, dada la posibilidad de disponer de una salida en estéreo, se recomienda tener conectar unos auriculares, que no tienen que ser necesariamente de alta calidad, dado que existe un amplio abanico de precios, desde los 15€ hasta los 300€ en algunos casos, se ha tenido en cuenta un valor de 30€.

Por último, pero no menos importante, para la realización del proyecto se necesita un ingeniero/a como mano de obra. Teniendo como referencia las horas equivalentes a los créditos del TFG, que sería un total de 360h. Un salario medio de ingeniería en España actualmente es de 30.000€ al año aproximadamente, que supone 15,38€/h brutos [25]. Esto supondría un coste de 30€/h neto aproximadamente, lo que sería un total de 10.800€ de mano de obra.

Todo esto sería un total de lo mostrado en la siguiente tabla:

Tipo de coste	**Coste total en euros (€)**
Mano de obra	10.800€
Material	580€
Total	11.380€

Tabla 2. Tabla de costes.

8. Conclusiones

En este capítulo se describen las conclusiones finales obtenidas del TFG, las competencias empleadas para su desarrollo y las nuevas competencias adquiridas, además de posibles trabajos futuros que añadan más funcionalidad a la aplicación desarrollada.

8.1. Conclusiones finales

A lo largo del TFG, se ha realizado un estudio de las mesas de mezclas y diversos conceptos que comparten relación para la creación de una aplicación orientada a dispositivos Android que permite emular la parte principal de una mesa de mezclas, que es el mezclado de múltiples pistas de audio para su reproducción posterior

Con el objetivo de adaptarse a tamaños de pantalla reducidos, el diseño propuesto ha sido de carácter minimalista, compuesto por tres canales de entrada y la salida principal estéreo que realiza la función de mezclado, pudiéndose realizar diferentes combinaciones de mezcla en función de los canales de entrada.

El resultado final cumple con los objetivos del TFG y se ha realizado la validación de la aplicación en diferentes dispositivos y utilizando distintas salidas de audio (la del propio dispositivo y con diversos auriculares), siendo idónea para usuarios Android que quieren adentrarse en el mezclado de audio.

8.2. Competencias empleadas

Como se ha explicado a lo largo de la memoria, se trata de un prototipo que ha sido programado con Kotlin, para Android, para poder ejecutarse, se han empleado diferentes competencias aprendidas a lo largo de los estudios del grado.

Las diferentes asignaturas de programación han ayudado a la comprensión del lenguaje mencionado a pesar de no haber sido usado con anterioridad, puesto que comparte similitudes con otros ya aprendidos durante asignaturas como Informática I, Informática II, Protocolos para la Transmisión de Audio y Vídeo en Internet, Construcción de Servicios y Aplicaciones Audiovisuales en Internet, y Laboratorio de Tecnologías Audiovisuales en la Web.

Los conocimientos de audio han venido gracias a asignaturas como Ingeniería Acústica I, Ingeniería Acústica II, Electroacústica, Tratamiento Digital del Sonido,

Difusión de Audio y Vídeo, Filtrado Digital Avanzado, y Equipos y Sistemas de Audio y Video.

Mención también al conocimiento que se obtiene con Idioma Moderno, puesto que mucha de la información recolectada, tanto a nivel de memoria como a entendimiento de código, viene de textos y vídeos en inglés.

8.3. Competencias adquiridas

El principal conocimiento adquirido ha sido el del lenguaje de Kotlin, ya que es como se ha desarrollado todo. Se trata de uno de los dos lenguajes usado para las aplicaciones de Android y presenta la complejidad de no haber sido trabajado con anterioridad. Así como el uso de Android Studio, plataforma sin la que no se podría haber realizado todo.

Se ha profundizado en el conocimiento del audio, así como de las herramientas de trabajo en ese mundo, más allá del que se había recibido durante las asignaturas del grado.

Por último, pero no menos importante, se ha reforzado el entendimiento adquirido durante los últimos años de la búsqueda de información, así como la sintetización y desarrollo de un proyecto de tal magnitud como este TFG. Estos conocimientos se han visto reforzados, también, con la búsqueda de solución de problemas que se pueden encontrar durante la programación.

8.4. Trabajos futuros

Al ser un prototipo, el proyecto está abierto a posibles mejoras para el uso por usuarios finales, algunos de los ejemplos podían ser los mencionados a continuación:

- **Habilitar la vista para múltiples dispositivos.** Android Studio permite emular la aplicación en diferentes dispositivos, esto conlleva pruebas para múltiples casos, pero no para la totalidad de ellos, por lo que, dependiendo del tamaño de pantalla, se podrá ver diferente al diseño planteado. Esto podría ser válido para llevar la aplicación a *tablets*. Así como no tendría sentido hacerlo para Android TV debido a incompatibilidad del proyecto con el uso de los dispositivos.
- **Implementación de audios a partir de internet.** En la actualidad, se permite introducir un audio del dispositivo del usuario, sino usar unas pistas

introducidas por defecto en la aplicación, la posible mejora en la que trabajar sería usar audios cogidos de cualquier página de internet.

- **Rotación de la aplicación**. A pesar de tener el diseño en el archivo *land\activity_main.xml*, el proyecto no ha sido implementado para la vista en horizontal del dispositivo móvil.
- **Introducción de filtros y herramientas de ecualización**. Aunque se ha comentado que no se han introducido funcionalidades relacionadas con la ecualización debido al diseño minimalista y acotado de la mesa de mezclas, sería interesante introducir la posibilidad de un filtrado paso-alto en cada una de las entradas, así como valorar la posibilidad de introducir alguna funcionalidad de ecualización, pero manteniendo la simplicidad del diseño.

9. Bibliografía

[1] Pérez Porto, J., Merino, M. (2015). Android - Qué es, características, definición y concepto. https://definicion.de/android/

[2] Rodríguez, D. (2021). Definición de Android. https://conceptodefinicion.de/android/

[3] Véliz, D. (2023). Android o iOS: ¿qué sistema operativo consigue más conversiones? https://marketing4ecommerce.net/android-o-ios-que-sistema-operativo-consigue-mas-conversiones-flat101/

[4] Canorea, E. (2022). ¿Qué es Kotlin y para qué sirve? https://www.plainconcepts.com/es/kotlin-android/

[5] Cervera, A. (2023) Top 10 Formatos de archivo de audio más comunes. https://recoverit.wondershare.es/audio-recovery/top-10-common-audio-file-formats.html

[6] RAE (2022). Ganancia. https://dle.rae.es/ganancia

[7] Rivero, H. (2018). Diferencias entre ganancia y volumen. https://www.cpaonline.es/blog/sonido/diferencias-ganancia-volumen/

[8] RAE (2022). Volumen. https://dle.rae.es/volumen

[9] Sisniega Martín, Ó. (2018). ¿Cuál es la diferencia entre ganancia y volumen para un amplificador de guitarra? https://www.academialem.com/cual-es-la-diferencia-entre-ganancia-y-volumen-para-un-amplificador-de-guitarra/

[10] Beckwith, R. (2007). Broadcasting House 1932. https://web.archive.org/web/20070804023240/http://www.btinternet.com/~roger.beckwith/bh/bh32/bh32_t.htm

[11] Evans, M. (2007) Picture Library Broadcasting House in Portland Place, London. https://www.google.com/imgres?imgurl=https%3A%2F%2Fwww.prints-online.com%2Ft%2F164%2Fbbc-broadcasting-house-614637.jpg&imgrefurl=https%3A%2F%2Fwww.prints-online.com%2Fbbc-broadcasting-house-614637.html&tbnid=5kiWZgzizISxSM&vet=12ahUKEwivuL2o5ar1AhUV-4UKHSlXDZ0QMygXegUIARDoAQ..i&docid=x6a3xUxuHFQhWM&

w=250&h=330&itg=1&q=Broadcasting%20House%2C%20en%20Portland%20Place%2C%20London%201932&hl=es&ved=2ahUKEwivuL2o5ar1AhUV-4UKHSlXDZ0QMygXegUIARDoAQ

[12] Garrán, D. (2021) ¿Qué es el muro del sonido, la técnica de Phil Spector que revolucionó la música? https://los40.com/los40/2021/01/18/los40classic/1610964794_179211.html

[13] Diversos Autores. Mesa de mezclas de audio. https://es.wikipedia.org/wiki/Mesa_de_mezclas_de_audio

[14] Neve. (1983) DSP, Digital Signal Processing Desk Brochure. https://es.scribd.com/doc/56149271/Neve-DSP-Desk-Brochure-1983# by Zack Dagoba

[15] The Board of Trustees of the Science Museum. (1981) Neve DSP Digital Sound Desk Control Surface. England. https://collection.sciencemuseumgroup.org.uk/objects/co203262/neve-dsp-1-digital-sound-desk-digital-mixing-desk

[16] Guerrero Vaquerizo, I. Sistemas de Producción Audiovisual https://books.google.es/books?id=9FU7DwAAQBAJ&pg=PA80&lpg=PA80&dq=historia+digitalizaci%C3%B3n+mesa+de+mezclas&source=bl&ots=rKsreeSbKz&sig=ACfU3U0g0VWqxbh03d2gO1cJ52nUaqKRAA&hl=es&sa=X&ved=2ahUKEwiw_ZnJ4Oz3AhWr_rsIHSOcDwMQ6AF6BAgyEAM#v=onepage&q=historia%20digitalizaci%C3%B3n%20mesa%20de%20mezclas&f=false

[17] NEOmúsica. (2021) ¿Cómo elegir una mesa de mezclas para principiantes? https://neomusica.es/blog/como-elegir-una-mesa-de-mezclas-para-principiantes/

[18] Sounds Market Blog. (2019). Todo lo que debes saber antes de comprar una mesa de mezclas (o mesa DJ) https://soundsmarket.com/blog/mesa-de-mezclas#Mesa_de_mezclas_analogica

[19] El Rincón del Vago. Mesas de mezclas https://html.rincondelvago.com/mesas-de-mezcla.html

[20] Behringer (2002) User's Manual Eurorack UB1204-PRO/UB1204FX-PRO https://www.manualsdir.com/manuals/52288/behringer-ub1204fx-pro-ub1204-pro.html?original=1

[21] Radios Libres. (2015) Capítulo 5 - ¿Qué son todos esos botones? https://radioslibres.net/capitulo-5-que-son-todos-esos-botones/

[22] Mahjong. (2019) Entendiendo las mesas de mezclas. http://wintablet.info/2019/11/entendiendo-las-mesas-de-mezclas/

[23] Santigosa, C. (2018) Set-up https://www.gladoop.com

[24] Developers Android. Windows: Cómo verificar los requisitos del sistema https://developer.android.com/codelabs/basic-android-kotlin-compose-install-android-studio?hl=es-419#1

[25] Talent Website. (2023) Sueldo Ingeniero en España https://es.talent.com/salary?job=ingeniero

[26] Developers Android. Formatos de medios compatibles https://developer.android.com/guide/topics/media/media-formats?hl=es-419

10. Anexos

A.1 Instalación y Uso

La aplicación de Android puede ser fácilmente instalada tras la descarga del apk (Android Application Package) que se puede encontrar en Github:

https://github.com/mdelaosa/Memoria_TFG

Para que el dispositivo móvil permita la instalación, es imprescindible la opción de instalación de aplicaciones de orígenes desconocidos, que se puede encontrar en los ajustes de éste. Tras ello, la aplicación estará lista para la instalación y su posterior uso.

Es necesario recordar que es posible que la vista no sea la encontrada en la Figura 10, debido a las diferentes dimensiones en las pantallas.

A.2 Publicación

El código fuente de la aplicación de Android se sitúa en el link de Github: https://github.com/mdelaosa/AudioKotlin

Es posible clonar el repositorio para la visualización del código y/o emulación en Android Studio, así como consultar carpetas.

La presente memoria se encuentra en https://github.com/mdelaosa/Memoria_TFG

A.3 Formatos y códecs de audio compatibles con Android Studio

Formato/códec	Codificador	Decodificador	Detalles	Formatos de tipos de archivo / contenedores compatibles
AAC LC	•	•	Compatibilidad con contenido mono/estéreo/5.0/5.1 con tasas de muestreo estándar de 8 a 48 kHz.	• 3GPP (.3gp) • MPEG-4 (.mp4, .m4a) • ADTS raw AAC (.aac, decodificación en
HE-AACv1 (AAC+)	• (Android 4.1 y versiones posteriores)	•		

HE-AACv2 (AAC+ mejorado)		•	Compatibilidad con contenido estéreo/5.0/5.1 con tasas de muestreo estándar de entre 8 y 48 kHz.	Android 3.1+, codificación en Android 4.0+, ADIF no es compatible) • MPEG-TS (.ts, no admite búsquedas, Android 3.0+)
AAC ELD (AAC mejorado de bajo retraso)	• (Android 4.1 y versiones posteriores)	• (Android 4.1 y versiones posteriores)	Soporte para contenido mono/estéreo con tasas de muestreo estándar de 16 a 48 kHz	
AMR-NB	•	•	4.75 a 12.2 kbps con muestreo a 8 kHz	3GPP (.3gp)
AMR-WB	•	•	9 tasas de 6.60 kbit/s a 23.85 kbit/s con muestreo a 16 kHz	3GPP (.3gp)
FLAC	• (Android 4.1 y versiones posteriores)	• (Android 4.1 y versiones posteriores)	Mono/estéreo (sin multicanal). Tasas de muestreo de hasta 48 kHz (pero se recomienda hasta 44.1 kHz en dispositivos con salida de 44.1 kHz, ya que el reductor de muestreo de 48 a 44.1 kHz no incluye un filtro de paso bajo). Se recomienda 16 bits. no se aplicó ninguna interpolación para 24 bits.	FLAC (.flac) únicamente
GSM		•	Android admite la decodificación GSM en dispositivos de telefonía.	GSM (.gsm)
MIDI		•	MIDI tipo 0 y 1. Versión 1 y 2 de DLS. XMF y Mobile XMF.	• Tipo 0 y 1 (.mid,.xmf,.mxmf) • RTTTL/RTX

			Compatibilidad con los formatos de tono RTTTL/RTX, OTA y iMelody.	(.rtttl,.rtx) • OTA (.ota) • iMelody (.imy)
MP3		•	Tasa de bits constante (CBR) o variable (VBR) mono/estéreo de 8 a 320 Kbps.	MP3 (.mp3)
OPUS		• (Android 5.0 y versiones posteriores)		Matroska (.mkv)
PCM/WAVE	• (Android 4.1 y versiones posteriores)		MIC lineal de 8 y 16 bits (el límite de las tasas depende del hardware). Tasas de muestreo para grabaciones de MIC sin procesar a 8,000; 16,000; y 44,100 Hz.	WAVE (.wav)
Vorbis		•		• Ogg (.ogg) • Matroska (.mkv, Android 4.0 y versiones posteriores)

Tabla 3. Formatos y códecs de video [26].