



VÍCTOR BERENGUER MADRID

PROYECTO FINAL DEL CICLO FORMATIVO DE GRADO SUPERIOR EN
DESARROLLO DE APLICACIONES MULTIPLATAFORMA

IES SEVERO OCHOA

2024

Contenido

1. Introducción	5
2. Objetivos	7
2.1. Objetivos específicos	7
3. Requisitos de la aplicación	7
3.1. Requisitos funcionales	7
3.1.1. Inicio de sesión.....	7
3.1.2. Pantalla principal	8
3.1.3. Barra lateral.....	8
3.1.4. Ítems.....	8
3.2. Requisitos no funcionales.....	9
3.2.1. Accesibilidad	9
3.2.2. Seguridad.....	9
3.2.3. Compatibilidad.....	9
4. Planificación	10
5. Herramientas utilizadas.....	11
5.1. Software.....	11
5.1.1. Android Studio	11
5.1.2. Firebase.....	11
5.1.3. Cloud Firestore.....	11
5.1.4. Draw.io	12
5.1.5. Glide.....	12
5.1.6. CardView.....	12
5.1.7. Navigation Drawer View	12
5.2. Tecnologías	12
5.2.1. Kotlin.....	12

5.2.2. Git/GitHub.....	12
6. Casos de uso.....	13
7. Diseño	15
7.1. Base de datos.....	15
7.1.1. Diagrama Entidad – Relación	15
7.1.2. Diagrama UML.....	16
7.2. Arquitectura.....	16
7.3. Diseño de prototipo	18
8. Desarrollo.....	19
8.1. Firebase	19
8.2. Estructura del proyecto.....	22
8.3. Vistas.....	23
9. Pruebas	25
10. Conclusiones.....	27
11. Bibliografía y recursos utilizados.....	28
Anexo A. Manual del usuario	29
A.1. Inicio de sesión	29
A.2. Acceder a álbum	29
A.3. Acceder a artista.....	30
A.4. Añadir a favoritos	30
A.5. Acceder a favoritos.....	31
A.6. Eliminar de favoritos.....	31
A.7. Búsqueda.....	32
A.8. Cerrar sesión.....	32

1. Introducción

La música, como es obvio, le gusta a muchas personas, incluso les resulta imprescindible para el día a día. Además, muchas de ellas, se interesan por descubrir nueva música, estar al tanto sobre las novedades de sus artistas favoritos u opinar sobre ella con más gente.

Todo esto resulta algo complicado de realizar ya que no existen plataformas concretas que den soporte a estas opciones y siempre suele ser por medio de extras en aplicaciones o webs de forma indirecta. Existen plataformas para valorar música, se pueden utilizar las redes sociales para opinar sobre ella, descubrir las últimas novedades, pero no de forma personalizada.

Con *Music Heads*, se busca que los usuarios puedan buscar personalizar su experiencia musical. Establecer un diario de los álbumes escuchados, añadir nuevos artistas o álbumes, escribir reseñas, lista de proyectos que se quieren escuchar, ranking de discos con mejor valoración, además de lo anteriormente mencionado, entre otras opciones son las características que hacen de *Music Heads* una aplicación musical completa.

Music Heads está desarrollada en *Kotlin* ya que está pensada para dispositivos móviles, además, utiliza la plataforma *FireBase* para almacenar los datos.

Introduction

Music, as is obvious, is enjoyed by many people and is even essential for their daily lives. Moreover, many people are interested in discovering new music, staying updated on their favorite artists' latest releases, or discussing music with others.

All of this can be somewhat challenging to achieve because there are no specific platforms that support these options directly; it's usually through additional features in apps or websites indirectly. There are platforms to rate music, social media can be used to discuss it, and discover the latest releases, but not in a personalized manner.

With *Music Heads*, the goal is for users to personalize their musical experience. Establish a journal of listened albums, add new artists or albums, write reviews, create a list of projects they want to listen to, rank the highest-rated albums, and more, making Music Heads a complete music application.

Music Heads is developed in *Kotlin* as it is designed for mobile devices and utilizes the *FireBase* platform to store data.

2. Objetivos

Principalmente, la aplicación quiere facilitar a los usuarios el camino para desarrollar sus gustos por la música, ya sea encontrando nuevos o desarrollando los ya obtenidos.

2.1. Objetivos específicos

- Disponer de un sistema de *Login/Logout* de usuarios en la aplicación.
- Visualizar diversos tipos de listas en función de cada sesión de usuario.
- Disponer de un sistema de gestión de artistas y álbumes.
- Presentar distintas alternativas de búsqueda ya sea de artista por nombre o de álbum por título.

Mejoras

- Aumentar el número de listas visualizables.
- Mejorar ítems tanto de artistas como álbumes.
- Disponer de un sistema de calificación de álbumes.

3. Requisitos de la aplicación

3.1. Requisitos funcionales

Se han establecido una serie de funciones que la aplicación dispondrá. Entre estas funciones, se incluyen tanto aquellas que se han podido alcanzar como las que se tiene la intención de alcanzar en un futuro. Se listarán a continuación:

3.1.1. Inicio de sesión

Al ejecutar la aplicación, lo primero que aparecerá será la pantalla de inicio de sesión con la cual se accederá a los datos del usuario y, al mismo tiempo, un cierre de sesión.

3.1.2. Pantalla principal

Cuando el usuario accede a su sesión, entrará en la pantalla principal de la aplicación. Deberá aparecer un listado de los últimos álbumes y reseñas. Además de un botón de búsqueda y añadir un nuevo *ítem*.

3.1.3. Barra lateral

Si el usuario desliza de izquierda a derecha en su pantalla, se mostrarán el resto de secciones de la aplicación. Deberá disponer de:

- Un apartado con el que dirigirse a la pantalla principal.
- El perfil del usuario en el que ver la lista de álbumes favoritos.
- Distintas listas globales en las que visualizar los álbumes mejor valorados o filtrados por género.
- Lista con los álbumes que el usuario ha marcado para escuchar.
- Diario en el que se marcan los proyectos escuchados en formato diario y la valoración adjudicada por el usuario.

3.1.4. Ítems

Cuando se accede a cualquier artista o álbum deberá aparecer la siguiente información sobre dicho ítem:

- Imagen del ítem.
- Título o nombre, fecha de nacimiento o creación, duración y número de canciones en caso de álbum.
- En caso de artista o grupo:
 - Breve biografía.
 - Proyectos musicales en los que ha trabajado.
- En caso de álbum:
 - Listado de canciones dentro del álbum.
 - Nota media dada por los usuarios.
 - Reseñas de los usuarios.
 - Participantes del proyecto (cantantes, productores, escritores, etc...)
 - Género musical del proyecto.

- Botón con diferentes funciones como marcar como escuchado, adjudicar una nota, añadir al diario, marcar para escuchar más tarde o favorito.

3.2. Requisitos no funcionales

Estos requisitos son aquellos que no realizan una función directa en la aplicación si no que se encuentran implícitos en la aplicación. Su principal objetivo es mejorar la experiencia del usuario.

3.2.1. Accesibilidad

La aplicación deberá ser intuitiva para los usuarios de forma que resulte sencillo navegar por la misma, al igual que las distintas funciones deberán ser sencillas de usar para el usuario medio.

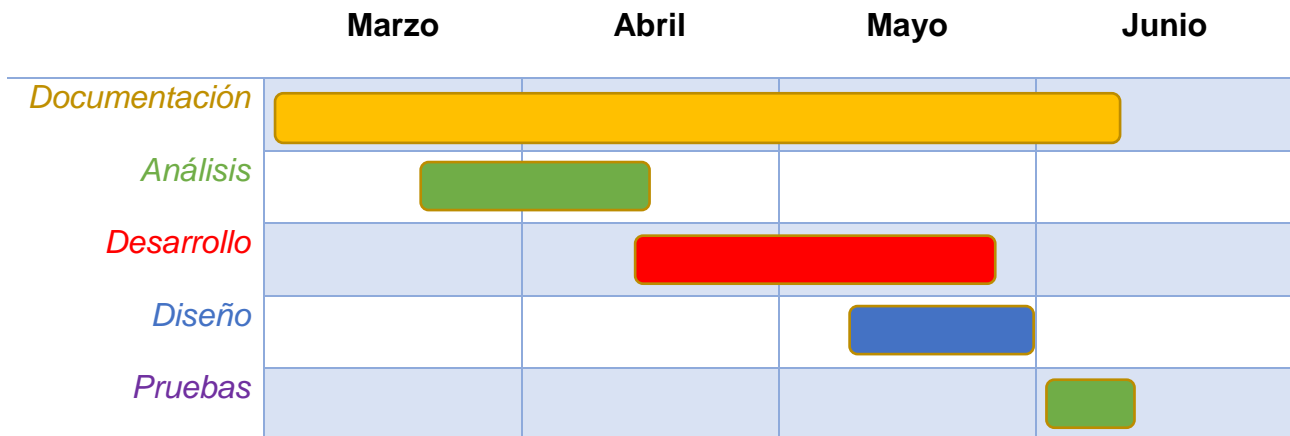
3.2.2. Seguridad

Aunque la aplicación no requiera de datos de suma importancia, deberá mantener la seguridad del usuario mediante un *Login*.

3.2.3. Compatibilidad

La aplicación no tendrá una complejidad importante, por lo que se buscará la compatibilidad también con los dispositivos menos actuales.

4. Planificación



Las fases de realización del proyecto se dividen en las siguientes:

- ❖ Documentación: durante toda la realización se documentará el proyecto al mismo tiempo que se desarrollan las distintas partes.
- ❖ Análisis: se realizará un análisis de todos los objetivos alcanzables, no alcanzables, requisitos funcionales y no funcionales, las herramientas que se utilizarán, etc.
- ❖ Desarrollo: en el momento en que se establecen los anteriores apartados, se comenzará con el desarrollo del prototipo.
- ❖ Diseño: alrededor de la mitad del desarrollo del prototipo, se comenzará a tener en cuenta el diseño de la aplicación, para terminar de definirlo una vez acabado el desarrollo.
- ❖ Pruebas: para acabar, se realizarán distintos test para comprobar el correcto funcionamiento de la aplicación, al mismo tiempo que se finaliza la documentación.

5. Herramientas utilizadas

A continuación, se tratarán las herramientas utilizadas, es decir, los recursos software utilizados. Entre estos se incluyen *IDEs* (Entornos de Desarrollo), sistemas gestores de bases de datos o lenguajes de programación utilizados.

5.1. Software

En este apartado se incluyen las herramientas que se utilizarán para la creación del prototipo. Es decir, los entornos empleados para manipular el código del programa.

5.1.1. Android Studio

Utilizado como Entorno de Desarrollo para el desarrollo de la aplicación. Es una especialización de *IntelliJ IDEA* dedicada a la creación de aplicaciones *Android*. Contiene soporte para el sistema de automatización de construcción de código *Gradle*, plantillas para la creación de diseños de interfaces *Android* y un emulador de un dispositivo *Android* en el que ejecutar las aplicaciones creadas, entre otras muchas características.

5.1.2. Firebase

Se ha utilizado *Firebase* para el almacenamiento de datos y autenticación de la aplicación. *Firebase* proporciona una serie de servicios como almacenamiento de bases de datos, autenticación, además de la integración de diversas aplicaciones como *Android*, *iOS*, *JavaScript*, entre otros. La plataforma está ubicada en la nube incorporada por *Google Cloud Platform*.

5.1.3. Cloud Firestore

La herramienta que utiliza *Firebase* como base de datos No SQL. Aquí se han almacenado los datos de los usuarios, álbumes y artistas.

5.1.4. Draw.io

Software online utilizado para realizar diagramas de casos de uso, modelo Entidad-Relación y diagrama UML.

5.1.5. Glide

Implementación en *Kotlin* para cargar imágenes desde *Cloud Firestore*.

5.1.6. CardView

Implementación en *Kotlin* para mostrar las listas y cargar contenido en forma de tarjetas.

5.1.7. Navigation Drawer View

Plantilla base de *Kotlin* que permite usar una barra lateral en la aplicación.

5.2. Tecnologías

En cuanto a las tecnologías, se incluyen principalmente aquellos lenguajes de programación utilizados, así como las librerías que se emplean para la construcción de la aplicación.

5.2.1. Kotlin

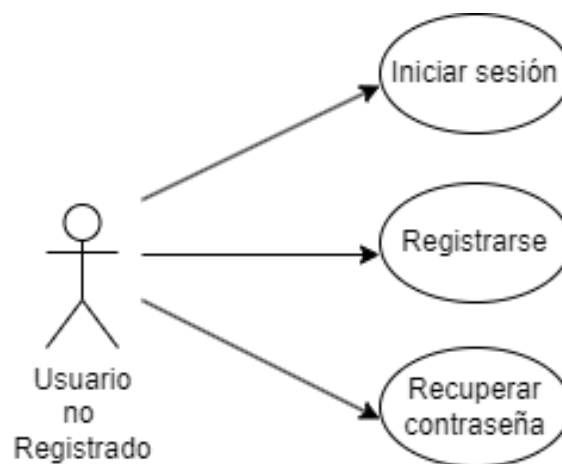
Es el lenguaje de programación elegido para el diseño y la creación del código que dará vida a la aplicación. Según *Google*, es el lenguaje preferido para los desarrolladores de aplicaciones *Android* y está diseñado para integrar de forma total *Java*. Es multiplataforma, estáticamente tipado y con interferencia de tipos.

5.2.2. Git/GitHub

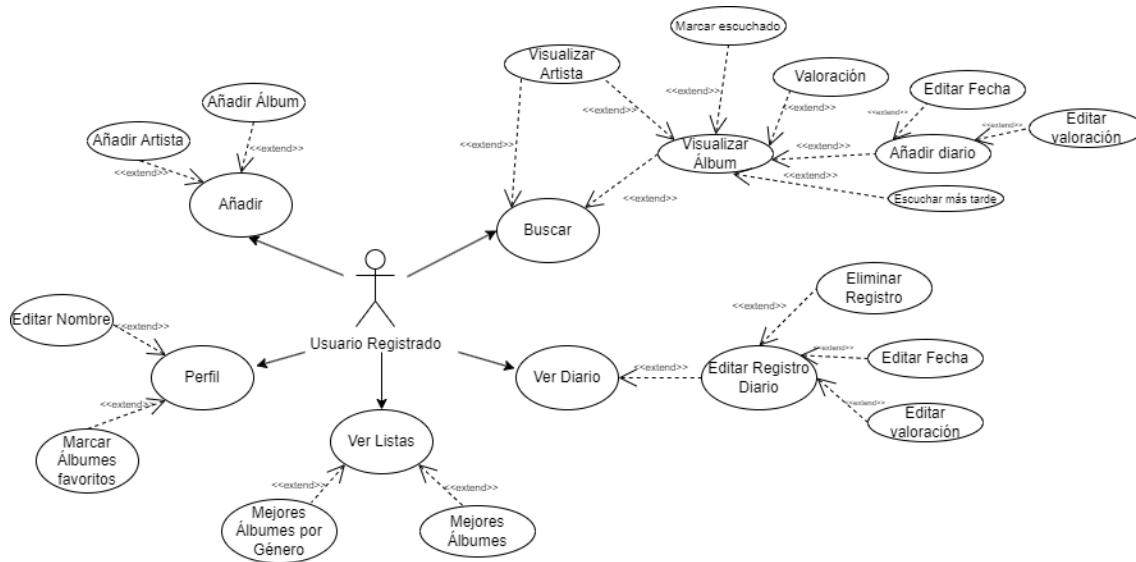
Software utilizado para el mantenimiento de versiones, con el cual se irá actualizando el desarrollo del proyecto. Android Studio, posee compatibilidad con Git por lo que resultará más cómodo el control de versiones. Se hará uso de la herramienta GitHub para alojar el proyecto.

6. Casos de uso

Los diagramas de casos de uso, proporciona una visión general de las interacciones entre los usuarios o actores y el sistema, ayudando a los desarrolladores a comprender los requisitos funcionales del sistema y a identificar las diferentes acciones que el sistema debe ser capaz de realizar para satisfacer las necesidades de los usuarios. En el caso de este proyecto, se pueden dividir los casos de uso en dos situaciones diferentes:



La primera situación, es cuando un usuario, que no está registrado en la base de datos de la aplicación, accede a ella. Las acciones que podrá realizar están relacionadas con el *Login* a la aplicación, es decir, iniciar su sesión, en caso de estar registrado, registrarse, en caso de no estarlo y reestablecer su contraseña en caso de no recordarla.



Una vez el usuario está registrado y ha accedido a la aplicación, puede interactuar con ella de muchas formas:

- Buscar un artista o álbum, el cual visualizará posteriormente, además de –en caso de álbum– poder realizar diferentes acciones.
- Visualizar el Diario en forma de lista de registros para, a continuación, editarlos.
- Añadir un Artista o Álbum.
- Acceder a su perfil y realizar diferentes ajustes como Editar su nombre en la aplicación o marcar sus álbumes favoritos en forma de “vitrina”.
- Visualizar diferentes listas anteriormente nombradas.

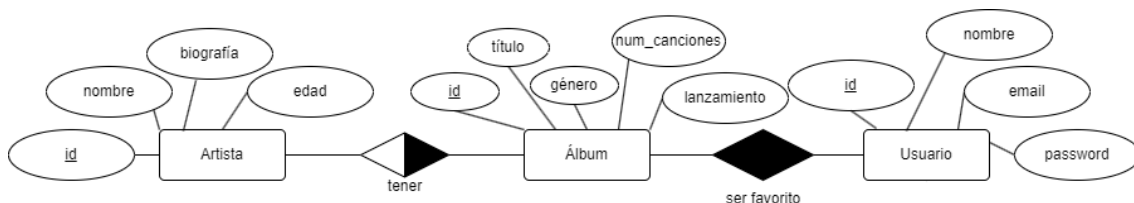
7. Diseño

7.1. Base de datos

Como se ha mencionado anteriormente, para almacenar los datos de la aplicación, se usará la herramienta de *Google, Firebase*, más concretamente *Cloud Firestore*. Esta herramienta hace uso de una base de datos NoSQL, la cual, está principalmente destacada por la flexibilidad y escalabilidad en la nube en la que se almacenarán y sincronizarán los datos necesarios para el posterior desarrollo de aplicaciones móviles y de servidor de manera escalable sencilla de usar.

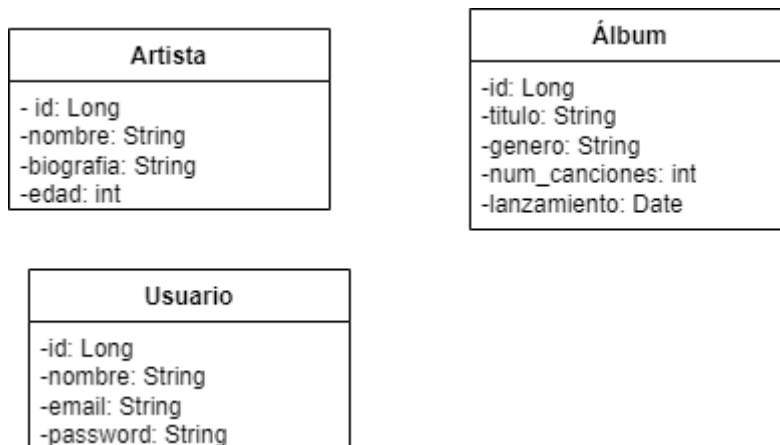
Dado que la base de datos está gestionada por *Google*, no requiere de un mantenimiento, un servidor externo o inactividad a causa de reinicio. Todo esto, sincronizando los datos en tiempo real entre clientes y servidores.

7.1.1. Diagrama Entidad – Relación



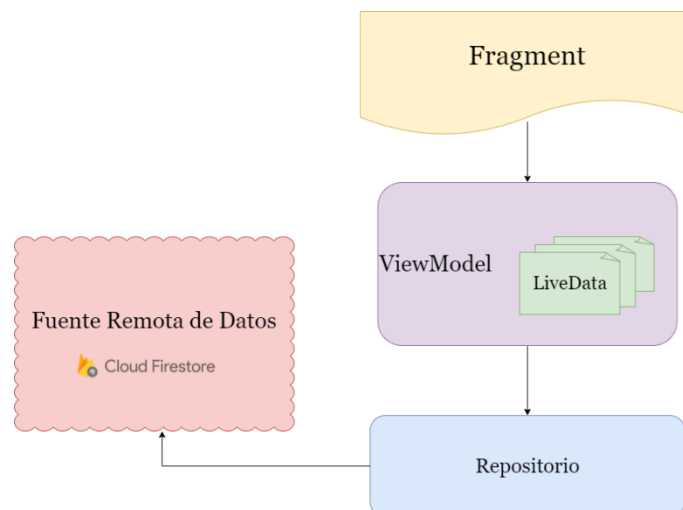
Aunque *Firebase* utiliza un modelo de datos NoSQL, lo que significa que no sigue una estructura tabular rígida con filas y columnas, sino que permite almacenar datos en documentos que se organizan en colecciones, representar la base de datos con el modelo Entidad-Relación, queda reflejada de forma más clara.

7.1.2. Diagrama UML



7.2. Arquitectura

La aplicación está más alineada a el patrón MVVM (*Model-View-ViewModel*). Esto es en gran parte por la integración de *Firebase* en la aplicación, además de la utilización de *Fragments* y *LiveData*.



Este patrón de diseño arquitectónico se utiliza en el desarrollo de aplicaciones para separar la lógica de la aplicación de la interfaz de usuario. Ayuda a gestionar la complejidad de las aplicaciones, facilitando la prueba y el mantenimiento del código.

El Model es responsable de manejar la lógica relacionada con los datos, como las llamadas a la base de datos y la comunicación con servicios remotos como en este caso *Firebase*. Contiene la estructura de los datos y la lógica para acceder y manipular esos datos.

El View o la vista es la interfaz de usuario que se presenta al usuario, en este caso el Fragment. Contiene los elementos visuales y maneja la interacción del usuario. Su función presentar datos al usuario y captar las acciones del usuario para enviarlas al ViewModel.

El ViewModel, es un intermediario entre el Model y el View. Mantiene y gestiona el estado de los datos para la interfaz de usuario y maneja la lógica de presentación.

El flujo de datos en una aplicación con arquitectura MVVM sería:

1. El ViewModel expone datos a la View a través del LiveData para observar los cambios de los datos.
2. El View observa estos datos y se actualiza automáticamente cuando los datos cambian.
3. Las acciones del usuario en el View (como clics o entradas de texto) se envían al ViewModel.
4. El ViewModel procesa estas acciones, actualizando los datos en el Model.
5. El Model actualiza los datos y el ViewModel observa estos cambios para reflejarlos en el View.

7.3. Diseño de prototipo



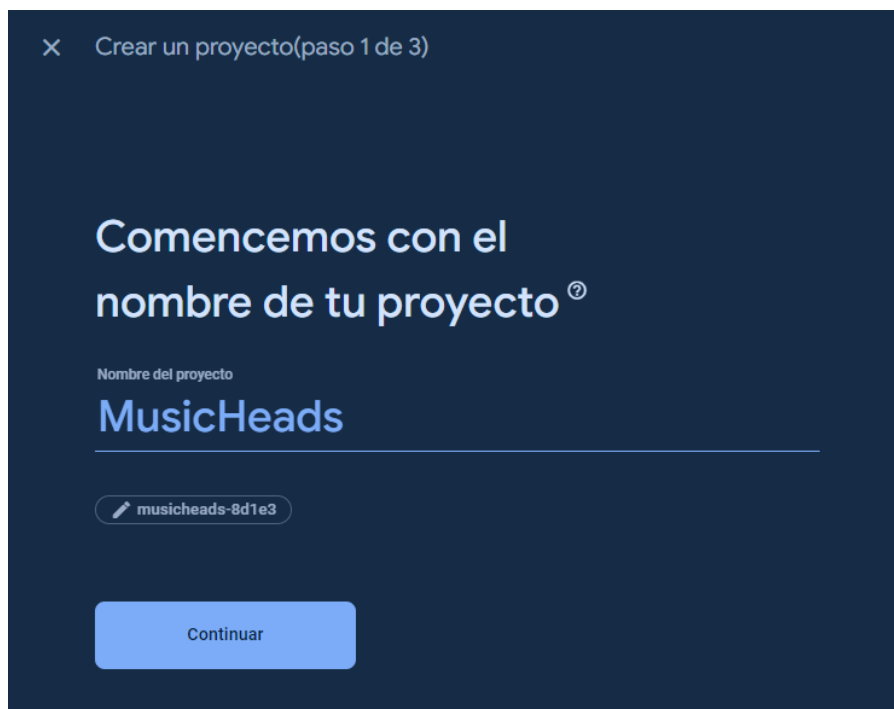
Este diseño de prototipo es una idea básica de cómo podría quedar el proyecto a grandes rasgos. La herramienta que he usado para crear este diseño es Fluid UI.

La estructura de los *Fragment* será en principio la misma y los colores pueden variar. El flujo de la aplicación queda como se ha comentado previamente en los casos de uso. Para este prototipo, no se desarrollarán todas las pantallas.

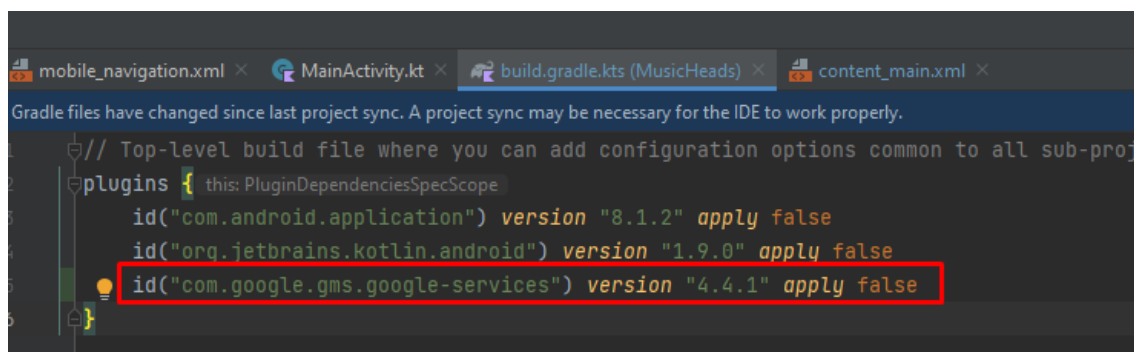
8. Desarrollo

Para comenzar con el desarrollo, el primer paso fue la instalación y configuración de Android Studio, en este caso, la versión 2023.3.1. A continuación, se configuró *Firebase* mediante la web:

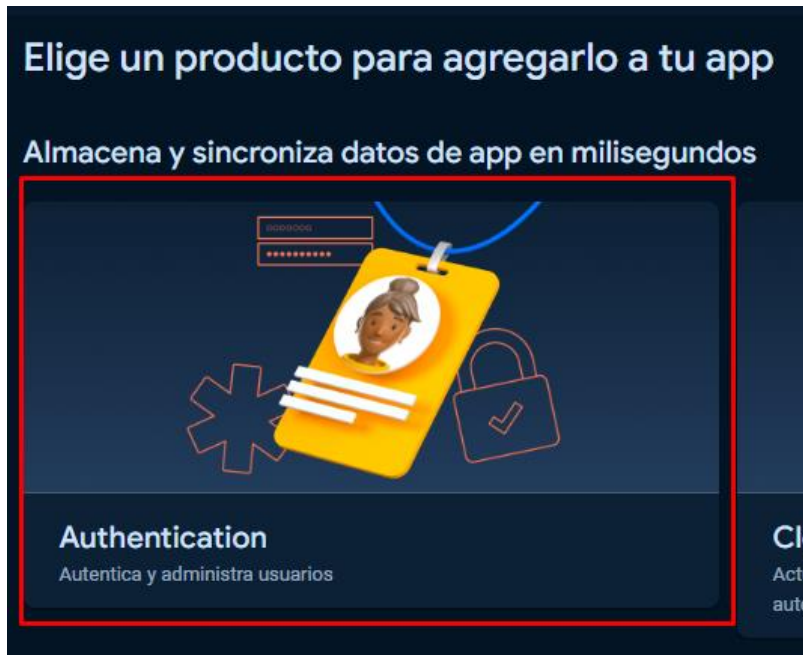
8.1. Firebase



Se creó un nuevo proyecto con el nombre del mismo y se asoció a Android Studio por medio de los archivos de configuración de Google, creado previamente registrando el paquete de la aplicación, además del SDK de *Firebase*.

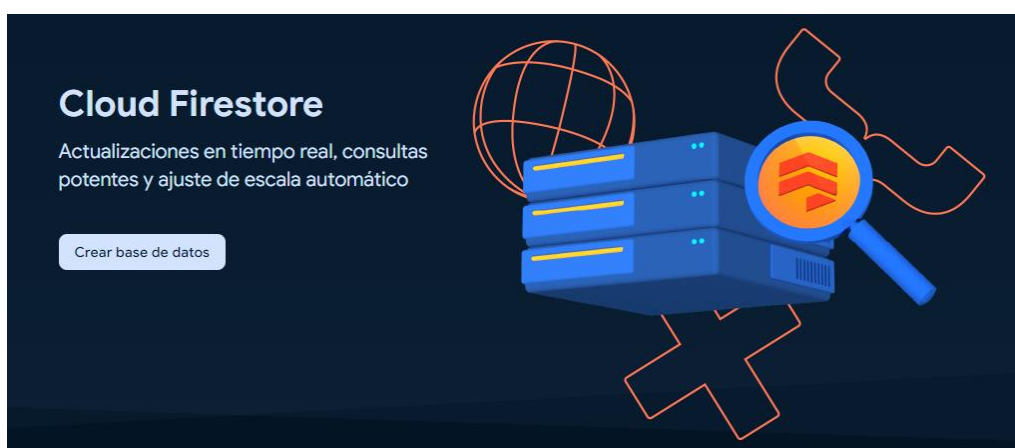


Una vez incluidos en el *Gradle* tanto a nivel de proyecto como de aplicación, quedaría asociado el proyecto en Android Studio con el proyecto en *Firebase*.

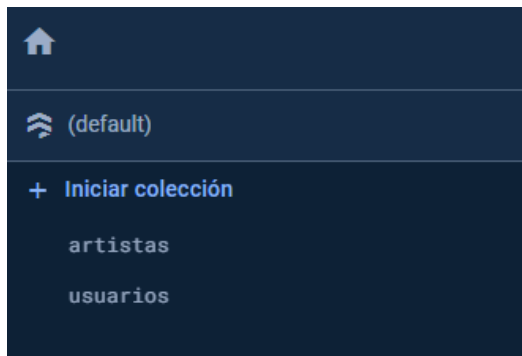


A continuación, de nuevo en *Firebase*, se agregaría la herramienta de *Authentication* para lograr implementar el registro, la autenticación y la administración de usuarios en la aplicación.

Aunque, se pueden añadir diversas opciones de autenticación con distintos proveedores como pueden ser número de teléfono, cuenta de *Google*, *Facebook*, etcétera, se ha optado inicialmente por correo electrónico y contraseña guardados en *Firebase*.



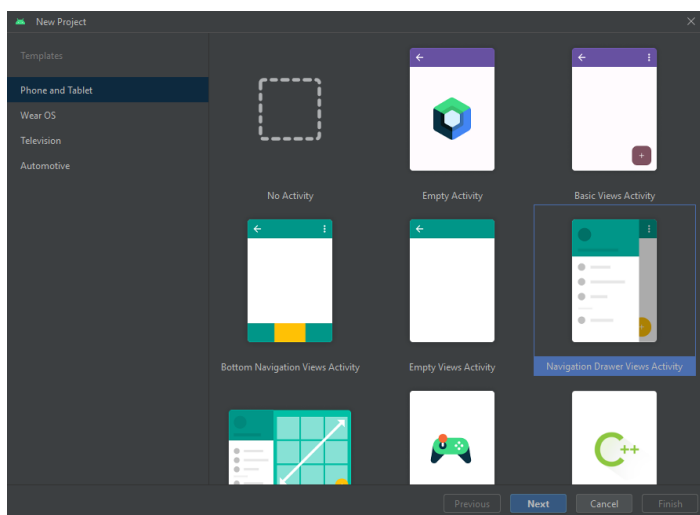
La herramienta utilizada de *Firebase* es *Cloud Firestore*, que cómo se ha mencionado antes, su función es el almacenamiento de los datos de la aplicación en una base de datos No SQL.



La base de datos, incluye dos colecciones iniciales, “artistas” y “usuarios”.

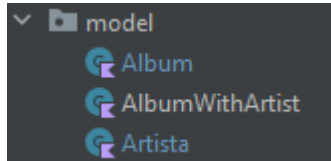
Cada documento “artista” contiene, además de sus propiedades, la sub-colección de “álbumes” y uno de los atributos de cada documento “álbum” siendo la ruta de referencia del artista al que pertenece.

Mientras que en el interior de la colección “usuarios”, cada documento “usuario” se identifica con el UID generado al crear el usuario mediante la herramienta de *Authentication* y contiene la sub-colección “favoritos” que tiene como propiedad la ruta de referencia de un documento “álbum”.



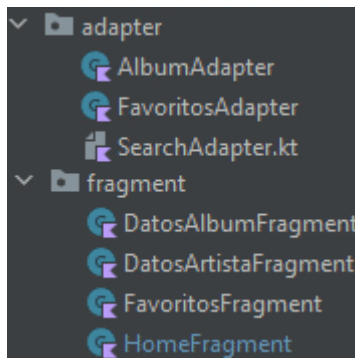
Al crear el proyecto en Android Studio, se eligió la *Navigation Drawer Views Activity* como plantilla para desarrollar la aplicación.

8.2. Estructura del proyecto



En el directorio *Model* se encuentran las clases que representan las entidades de datos y las interacciones con la base de daos.

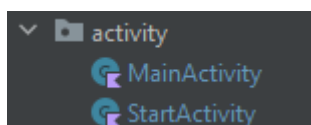
AlbumWithArtist es una clase auxiliar que asocia un álbum con su artista. Esta clase es útil para combinar los datos del álbum y del artista en una sola entidad, lo que facilita la manipulación de estos datos en la Interfaz como puede ser en un *Recycler View*.



Todas las clases que manejan la lógica de las pantallas están contenidas en los directorios *Adapter* y *Fragment*.

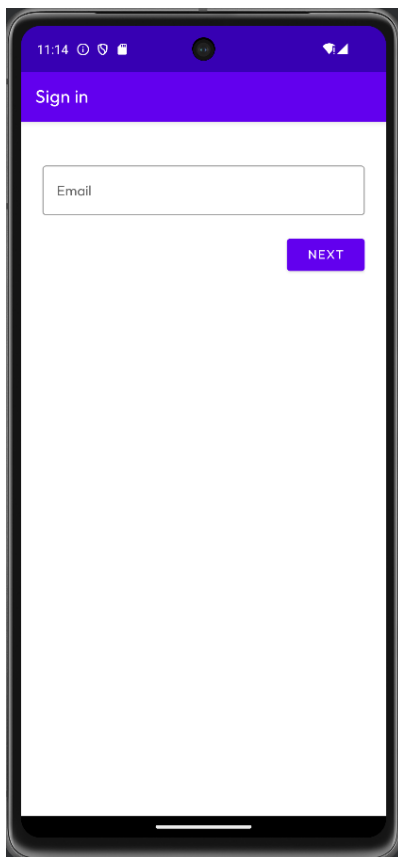
En *HomeFragment* se encuentra la pantalla principal en la que aparece un *RecyclerView* de álbumes. En *DatosAlbumFragment* y *DatosArtistaFragment*, se administra toda la lógica de lo que aparece en cada álbum y cada artista. *FavoritosFragment* contiene un *RecyclerView* de los álbumes marcados por el usuario como favoritos. En *SearchFragment* gestiona la búsqueda de ítems.

Mientras tanto, en los *Adapter*, contiene la lógica de los *RecyclerView*. En todas las clases, se maneja la conexión con *Firebase* para poder obtener los datos.

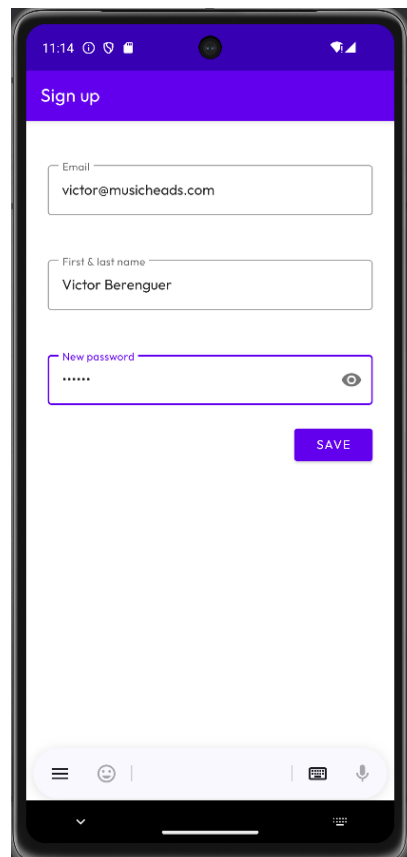


En *Activity* se encuentran las actividades que se ejecutan al iniciar la aplicación. En *StartActivity*, está contenida la autenticación de *Firebase* y en *MainActivity*, se encuentra la lógica para el funcionamiento de la barra lateral, es decir, el *Navigation Drawer*.

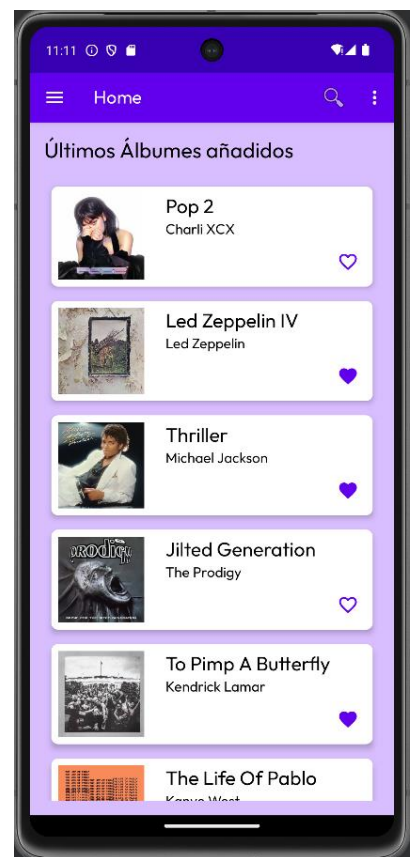
8.3. Vistas



Inicio de sesión



Registro



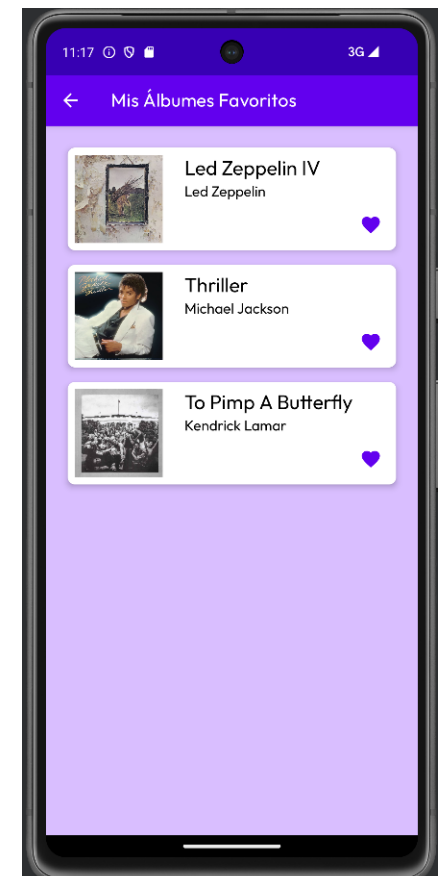
Pantalla Principal / Home



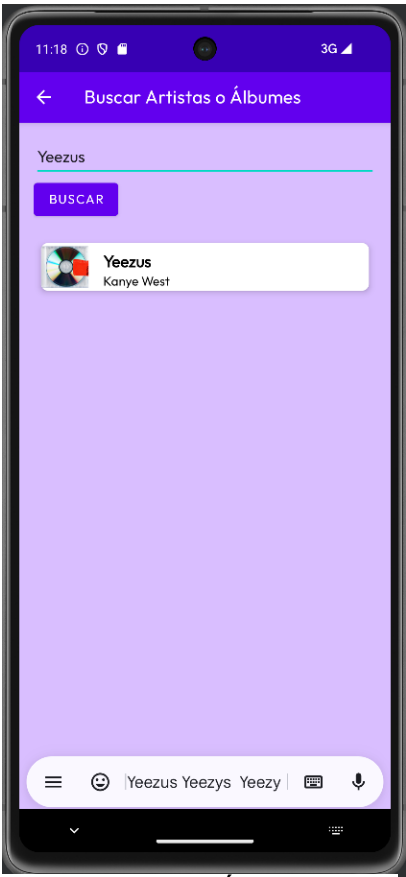
Datos Álbum



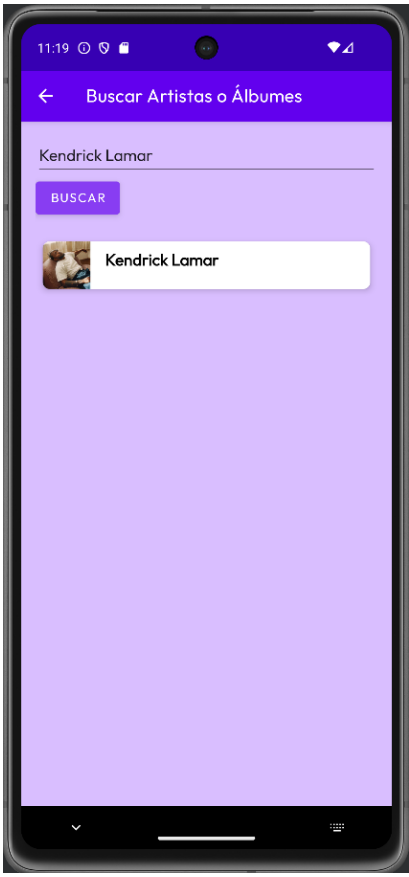
Datos Artista



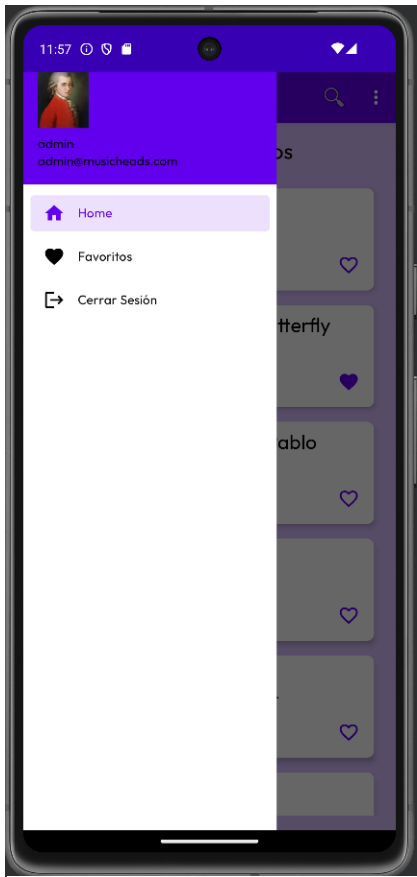
Álbumes Favoritos



Búsqueda Álbum











Búsqueda Artista



Barra Lateral






9. Pruebas

Device	API	Size on Disk	Actions
Pixel 7 API 34 Android API 34 Google Play x86_64	34	15 GB	   
Pixel 4 XL API 32 Android 12L Google APIs x86_64	32	9.6 GB	   

Para “testear” el dispositivo, se han utilizado diferentes dispositivos proporcionados por el propio Android Studio. La aplicación ha sido probada sobre todo en las versiones más recientes de Android, 12.0 y 14.0, las cuales equivalen a las APIs 32 y 34, respectivamente.

Pruebas realizadas

Iniciar sesión con un usuario almacenado	
Registrar nuevo usuario	
Listar correctamente los álbumes en Home	
Acceder al álbum seleccionado y mostrar correctamente los datos	
Acceder al artista del álbum anterior y mostrar correctamente los datos	
Guardar / Eliminar en favoritos desde Home	
Guardar / Eliminar en favoritos desde el álbum	
Listar correctamente los álbumes marcados por el usuario	
Eliminar de favoritos desde la lista de favoritos	

Listar correctamente el álbum / artista buscado	
Acceder a dicho álbum / artista	
Visualizar el correo electrónico y usuario en la barra lateral	
Acceder a las diferentes opciones de la barra lateral	
Cerrar sesión	

10. Conclusiones

Este proyecto ha sido algo complicado de realizar, sobre todo por dos cuestiones bastante claras.

La primera, es que durante el curso hemos aprendido a programar en *Kotlin*, sí, pero desarrollar una aplicación considero que conlleva mucho más aprendizaje del que hemos obtenido.

En cuanto a la segunda razón, por cuestiones académicas no he podido trabajar el proyecto tanto tiempo como me hubiera gustado, aun habiendo sido más de 40 horas, bastantes más.

También, relacionando las dos cuestiones entre sí, debido al insuficiente conocimiento previo que poseía respecto al resultado del proyecto, me ha acabado restando mucho más tiempo de trabajo, ya que he tenido que documentarme y aprender aspectos diferentes y nuevos de los anteriores.

Dicho esto, la idea del proyecto me resultó interesante cuando se me ocurrió y me lo sigue pareciendo ahora que he “acabado”, es decir, no me siento desgastado por la idea, por lo que me gustaría incluir muchísimas más opciones de las que hay.

Me gustaría agradecer a todos los profesores del ciclo y en especial a Patricia Martí Gran por ser nuestra tutora del curso y darnos los conocimientos de programación que no teníamos del primer curso, además de los nuevos y a Marina Navarro Pina por ser la tutora de mi proyecto y apoyar mis decisiones en el mismo, además de las laborales.

11. Bibliografía y recursos utilizados

<https://firebase.google.com/products/realtime-database?hl=es-419>

<https://es.wikipedia.org/wiki/NoSQL>

https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93modelo_de_vista

<https://developer.android.com/topic/libraries/architecture/viewmodel?hl=es-419>

<https://m2.material.io/components/navigation-drawer#anatomy>

<https://firebase.google.com/docs?hl=es-419>

<https://firebase.google.com/docs/auth?hl=es-419>

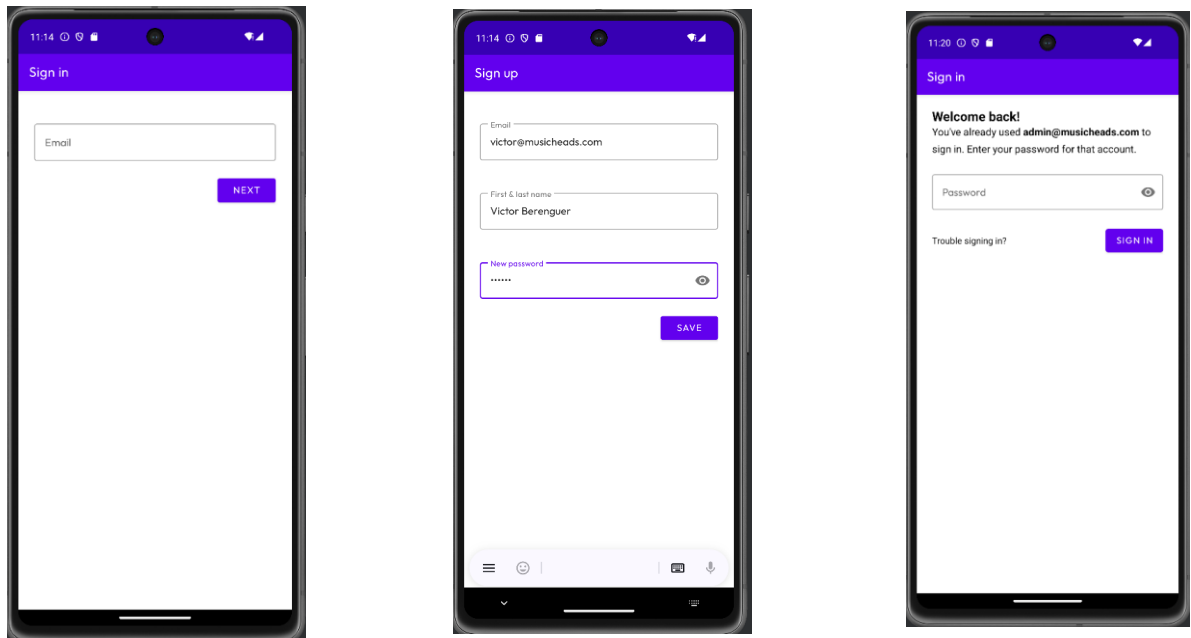
<https://firebase.google.com/docs/auth/android/firebaseui?hl=es-419>

<https://www.geeksforgeeks.org/image-loading-caching-library-android-set-2/>

<https://developer.android.com/develop/ui/views/layout/cardview?hl=es-419>

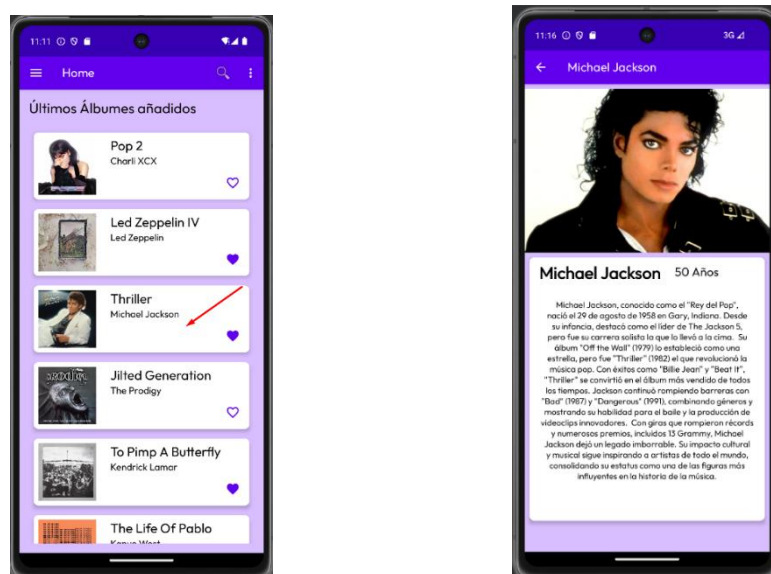
Anexo A. Manual del usuario

A.1. Inicio de sesión



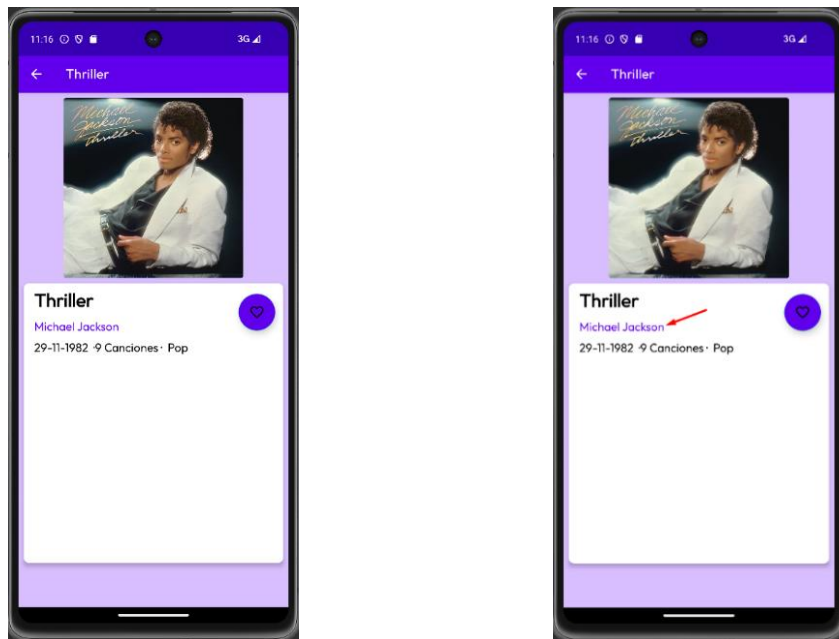
Para iniciar sesión, la aplicación primero pedirá un correo electrónico y en caso de que dicho correo coincida con uno registrado en la base de datos, pedirá la contraseña. Si no coincide, comenzará el registro con el email introducido, el nombre y la contraseña.

A.2. Acceder a álbum



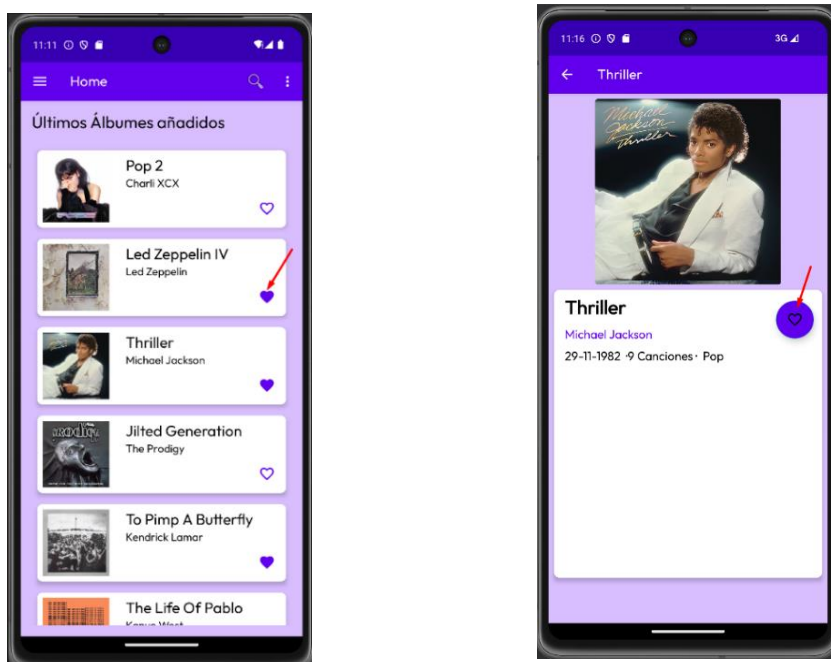
Para acceder a un álbum en concreto, desde Home, se pulsa sobre uno de los álbumes de la lista.

A.3. Acceder a artista



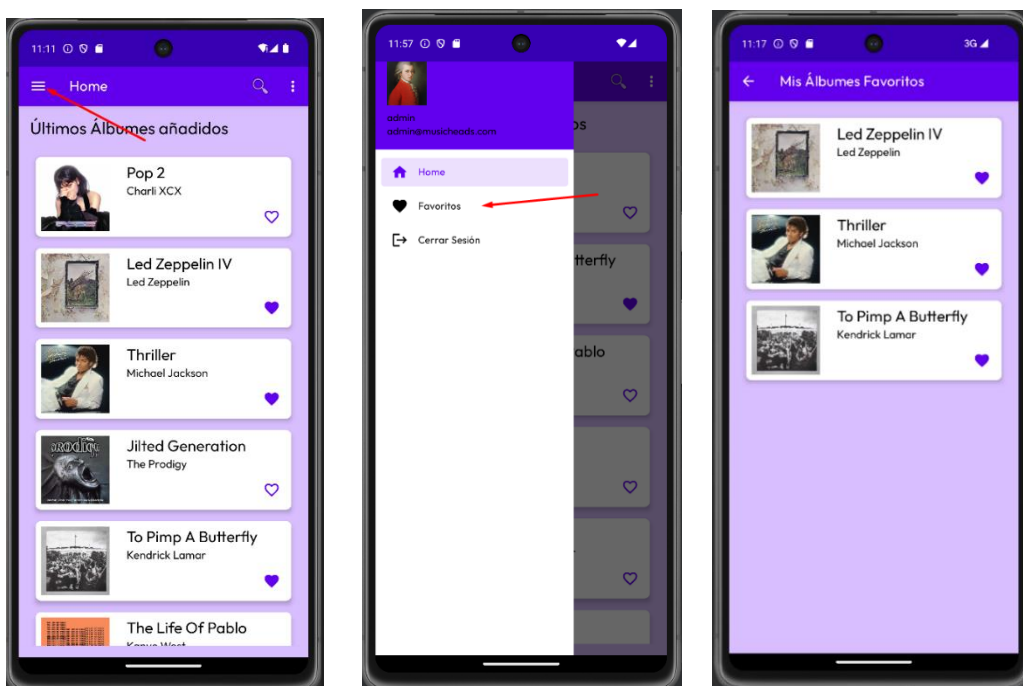
Para acceder a la pantalla del artista de un álbum, bastará con pulsar sobre su nombre.

A.4. Añadir a favoritos



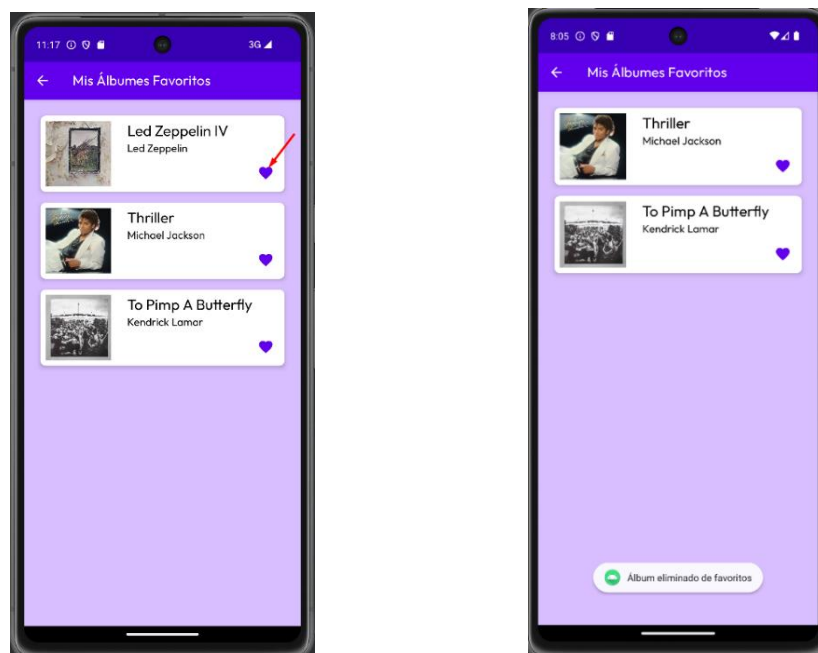
Es posible añadir un álbum en favoritos si se pulsa en el botón del corazón, tanto desde la lista de álbumes principal como desde el propio *Fragment* del álbum. Si se añade a favoritos, el corazón se rellenará, si se vuelve a pulsar, se eliminará de favoritos y se vaciará el corazón.

A.5. Acceder a favoritos



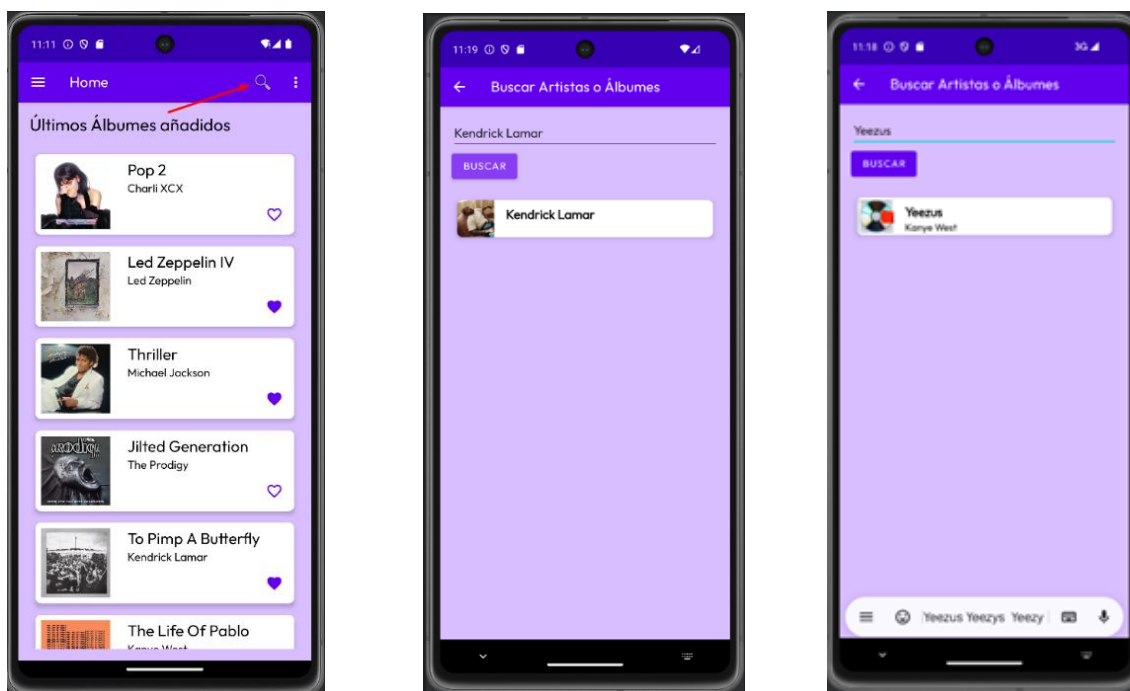
Para acceder a favoritos, desde la pantalla principal, se puede arrastrar desde izquierda a derecha para abrir la barra lateral o pulsar en las líneas de arriba a la izquierda y, posteriormente, pulsar en la opción de favoritos.

A.6. Eliminar de favoritos



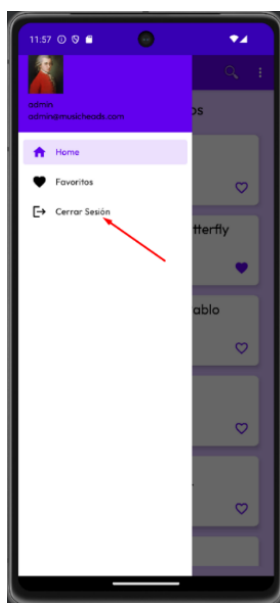
Desde favoritos, se podrá eliminar un álbum de esta lista si se pulsa en el corazón.

A.7. Búsqueda



Para hacer búsquedas de álbumes o artistas, se deberá pulsar sobre la lupa en la pantalla principal. Si se introduce el título completo del álbum o el nombre del artista y se pulsa en el botón de buscar, se incluirá en la lista de búsquedas. Si se pulsa sobre el resultado se accederá al *Fragment* de ese ítem.

A.8. Cerrar sesión



Para cerrar la sesión de usuario, bastará con volver a abrir la barra lateral y pulsar en la opción de Cerrar sesión.