Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Grado en Ingeniería informática

Trabajo Fin de Grado

Diseño de una Aplicación para un Reproductor de música local para el sistema operativo Android

Autor: Wei Zheng

Tutor(a): Vicente Martinez Orga

Madrid, junio 2022

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en ingeniería informática
Título: Diseño de una Aplicación para un Reproductor de música local para el sistema operativo Android
junio 2022

Autor: Wei Zheng

Tutor:
Vicente Martinez Orga
Departamento de Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

Hoy en día, las aplicaciones gratis tienen las siguientes características: publicidad, obtención de los datos privados, la necesidad de internet y la ocupación de la aplicación. Mi idea es diseñar una aplicación Android sencilla, solamente necesita una pequeña parte de la memoria, sin necesidad de internet, aunque estás conectado a internet, no te recogerá los datos privados tampoco te aparecerá las publicidades.

El objetivo principal de este proyecto es diseñar una aplicación de reproductor de música local para el sistema operativo Android, la aplicación es compatible en cualquier dispositivo de Android (Tablet, móvil, tv…) adaptándose de su dimensión. Utilizando la herramienta "Android Studio" para debugear y realización de prueba en el simulador. Por otra parte, el lenguaje "JAVA" para implementar las funcionalidades y XML para la interfaz gráfica.

Las funcionalidades de esta aplicación son: acciones de pausa, siguiente y anterior, obtener una lista de música local acompañado con el nombre, el cantante, el álbum y la duración de la música, los usuarios podrán deslizar la pantalla para navegar la lista la última funcionalidad es modo de reproducir las músicas, aleatorio o por el orden de la lista.

Abstract

Nowadays, free applications feature characteristics such as advertising, obtaining user’s private data, the need for internet connectivity, and storage space. My idea is to design a simple Android application that requires little storage and does not require internet access. It will not request any personal information and it will not generate any advertisement. This project's main purpose is to provide a local music player for Android systems. The software will be compatible with any Android devices including tablets, phones and televisions, adapting to any screen size. To debug and test on the simulator, Android Studio will be utilized. The functionality will be constructed in Java, while the graphical interface will be done in XML. This application allows users to play and pause songs, skip and return to them, and create local playlists with the name, performer, album, and duration of the playlist. Users will be able to navigate through by scrolling across the screen, and there will be an option to shuffle the playlist.

Tabla de contenidos

Tabla de ilustraciones

1 Introducción y objetivos

1.1 Introducción

Hoy en día, vivimos en un mundo donde la música es parte imprescindible de nuestra vida cotidiana: en el entorno del trabajo, de camino a casa, mientras hace ejercicio, etc. Todo lo que necesita es un dispositivo de audio, ya sean un altavoz o auriculares, y un dispositivo inteligente, puede escuchar toda la música que quieras, en cualquier lugar y cualquier momento. Pero, la publicidad en aplicaciones ha llegado para quedarse. Cada vez hay más empresas que comprenden que al publicar anuncios en las diferentes aplicaciones, pueden atraer su cliente objetivo.
En un estudio de AppsFlyer [1] estima que la inversión en publicidad en aplicaciones creció 65% en 2020, alrededor de 64,100 millones de dólares.
Por otra parte, las aplicaciones en nuestros dispositivos pueden recoger gran cantidad de datos de carácter personal.
Este proyecto es diseñar una aplicación offline, sin anuncios molestos mientras escuchas tus canciones favoritas, tampoco recogerá los datos privados de los usuarios. Esta aplicación no requiere comunicación con el exterior, sino que se realiza de forma local, y por tanto la aplicación ocupa menos espacios en la memoria, ejecutable en cualquier dispositivo de gama alta o baja.
Por último, la aplicación utilizará el sistema operativo Android [2], ya que en el 2021 hay alrededor de 3,000 millones de dispositivos con el sistema operativo Android, es decir, lo equivalente a la población de China e India. Satisface la mayoría de los dispositivos, además la aplicación es compatible para diferentes dimensiones de las pantallas.

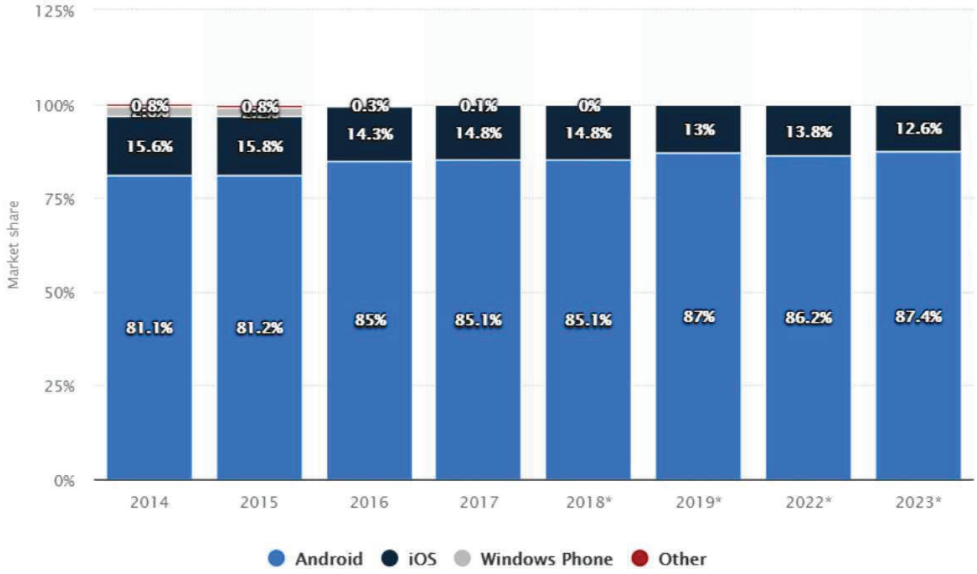

Ilustración 1 merado de los sistemas operativos de móviles

1.2 Objetivos

El objetivo principal de este proyecto es diseñar una aplicación de reproductor de música local para el sistema operativo Android. Utilizando la herramienta "Android Studio" [1] para debugear y realización de prueba en el simulador. Por otra parte, el lenguaje "JAVA" [2] para implementar las funcionalidades y XML [3] para la interfaz gráfica.

- Familiarizar con la herramienta "Android Studio".
- Estudio de listas dinámicas "RecyclerView".
- Estudio de diseño relativo "RelativeLayout y CardView".
- Estudio de "TextView, ImageView y Button".
- Estudio de "ContentResolver y RecyclerViewAdapter".
- Estudio de control de reproducción de video o música "MediaPlayer".
- Diseño de la interfaz de usuario, donde muestra una lista con las canciones, cantantes, álbum y duraciones.
- Implementación de las funcionalidades "pausa, anterior y siguiente".
- Implementación de modo de reproducción "aleatorio o por el orden de la lista".
- Implementación de avisos simple "Toast".

2 Estado de arte

2.1 Android

2.1.1 Introducción a Android

Android [2] es un sistema operativo móvil diseñado para dispositivos móviles con pantalla táctil como teléfonos inteligentes o tabletas, pero también se puede encontrar en otros dispositivos como relojes inteligentes, televisores e incluso sistemas multimedia en algunos modelos de coches. El sistema operativo desarrollado por Google basado en Linux Kernel[6] y otro software de código abierto se ha convertido en la fuerza principal para la popularización de muchos dispositivos inteligentes debido a la conveniencia de usar una gran cantidad de aplicaciones. una manera fácil.

En primer momento fue desarrollado por Android Inc, luego adquirido por Google en 2005 y presentado dos años después, en 2007, para mejorar los estándares abiertos en los dispositivos móviles. El código fuente principal de es conocido comúnmente como Android Open Source Project (AOSP) y se destaca como el sistema operativo móvil más utilizado del mundo con un porcentaje de mercado de más del 90% en 2018, un número que le sitúa muy por encima de IOS, competencia directa.[1]

Ilustración 2 Logo de Android

2.1.2 Arquitectura de la plataforma "Android"

Android es un software de código abierto basado en Linux diseñado para una amplia gama de dispositivos y modelos. El siguiente diagrama muestra los componentes principales de la plataforma Android.[2]

---

[1] Información extraída de (https://www.adslzone.net/reportajes/software/que-es-android/).

[2] Información extraída de (https://developer.android.google.cn/guide/platform?hl=es-419).

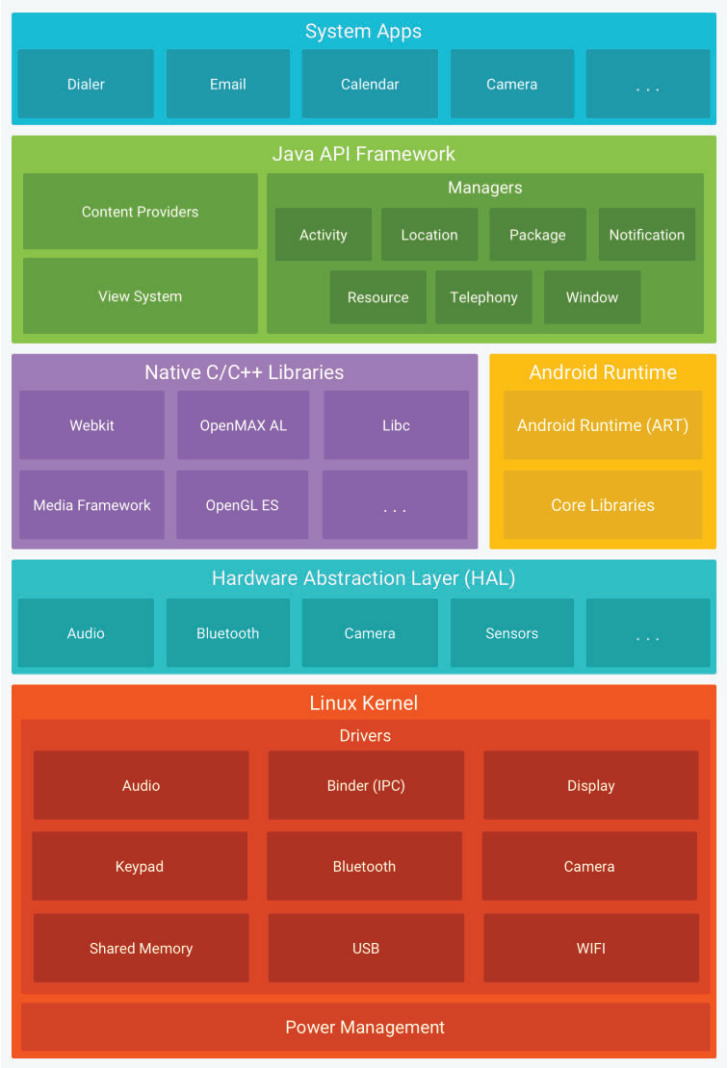

Ilustración 3 Arquitectura de Android

1. Kernel de Linux

La base de la plataforma Android es kernel Linux. Por ejemplo, el tiempo de ejecución de Android (ART) depende de Kerneel Linux para funcionalidades subyacentes, como la creación de subprocesos y la administración de memoria de bajo nivel.
El uso del kernel de Linux permite que Android aproveche funciones de seguridad claves y, al mismo tiempo, permite a los fabricantes de dispositivos desarrollar controladores de hardware para un kernel conocido.[3]

[3] Información extraída de (https://developer.android.google.cn/guide/platform?hl=es-419).

2. **Capa de abstracción de hardware (HAL)**

   La capa de abstracción de hardware (HAL) proporciona interfaces estándar que representan las capacidades de hardware de un dispositivo con el marco de trabajo de API de Java de nivel superior. La HAL consta de varios módulos de biblioteca, cada uno de los cuales utiliza una interfaz para un tipo específico de componente de hardware, como el módulo de la cámara o de Bluetooth. Cuando el marco de trabajo de una API realiza una llamada para acceder al hardware de un dispositivo, el sistema Android carga el módulo de biblioteca para ese componente de hardware en cuestión.

3. **Tiempo de ejecución de Android**

   Para dispositivos con Android 5.0 [4] (API nivel 21) y versiones posteriores, cada aplicación ejecuta sus propios procesos con sus propias instancias de Android Run Time (ART). El ART se escribió para ejecutar varias máquinas virtuales en dispositivos con poca memoria mediante la ejecución de archivos DEX, un formato de código de bytes diseñado específicamente para Android y optimizado para ocupar un espacio de memoria mínimo.

4. **Bibliotecas C/C++ nativas**

   Muchos componentes y servicios principales del sistema Android, como ART y HAL, se basan en código nativo que requiere bibliotecas nativas escritas en lenguaje C y C++. La plataforma Android proporciona API del marco de trabajo de Java para exponer la funcionalidad de algunas de estas bibliotecas nativas a las aplicaciones. Por ejemplo, puede acceder a OpenGL [5] a través de la API de OpenGL de Java del marco de trabajo de Android para agregar soporte para dibujar y manipular gráficos 2D y 3D.
   Si está desarrollando una aplicación que requiere lenguaje C o C++, puede usar el NDK de Android para acceder a algunas bibliotecas de plataformas nativas directamente desde su código nativo.

5. **Marco de trabajo de la API de Java**

   Todo el conjunto de funciones del SO Android está disponible a través de API escritas en el lenguaje Java. Estas API son los componentes básicos para crear aplicaciones de Android al simplificar la reutilización de los componentes del sistema central y los servicios del módulo.[4]

[4] Información extraída de (https://developer.android.google.cn/guide/platform?hl=es-419).

6. **Apps del sistema**

   En Android incluye un conjunto básico de aplicaciones para correo electrónico, SMS, calendario, navegación web y contactos, entre otros elementos. Las aplicaciones incluidas en la plataforma no tienen un estado especial entre las aplicaciones que el usuario elije instalar; Por lo tanto, la aplicación externa puede convertirse en el navegador web predeterminado del usuario, el sistema de SMS o incluso el teclado.[5]

2.1.3 Conocimientos utilizados el diseño

- **Objeto RelativeLayout:** RelativeLayout es un conjunto de vistas que muestran subvistas en posiciones relativas. Cada vista se puede colocar en relación con sus elementos del mismo nivel (como a la izquierda o debajo de otra vista) o en posiciones relativas al área RelativeLayout superior (como alineada a la parte inferior, izquierda o central).[6]

- **RecyclerView:** La clase RecyclerView nos permite mostrar una lista (o bien una grilla) de elementos.
  Lleva este nombre porque a medida que se renderizan los elementos de la lista, los elementos que dejan de observarse se reciclan para mostrar los elementos siguientes.
  RecyclerView es una versión mejorada de la clase ListView, principalmente cuando el número de elementos es variable, y/o los datos cambian continuamente.
  Es posible personalizar ListView para lograr lo mismo, pero implicará observar diferentes detalles para lograr el mismo rendimiento.[7]

- **CardView:**
  Las aplicaciones a menudo necesitan mostrar datos en contenedores de estilo similar. Estos contenedores se utilizan a menudo en listas para contener información de cada elemento. El sistema proporciona la API CardView como una manera fácil de ver la información de la tarjeta que tienen un aspecto coherente en la plataforma Estas tarjetas tienen una elevación predeterminada por encima del grupo de vistas que las contiene, por lo que el sistema dibujará una sombra debajo de ellas. Las tarjetas proporcionan una manera simple de incluir un conjunto de vistas al mismo tiempo que proporcionan un estilo coherente para el contenedor.[8]

---

[5] Información extraída de (https://developer.android.google.cn/guide/platform?hl=es-419).

[6] RelativeLayout información extraída de (https://developer.android.google.cn/guide/topics/ui/layout/relative?hl=es-419).

[7] RecyclerView información extradida de (https://programacionymas.com/blog/listas-dinamicas-android-usando-recycler-view-card-view).

[8] CardView información extraída de (https://developer.android.google.cn/guide/topics/ui/layout/cardview?hl=es-419).

- **TextView:** Es un widget que muestra un texto al usuario como su nombre indica. Claramente, esto la convierte en una de las vistas de interfaz de usuario más utilizadas para colocar títulos, encabezados, texto de información, etiquetas y más.[9]

- **ImageView:** Esta es una vista para mostar y manipular recursos de imágenes, como Drawables y Bitmaps. Varios efectos, discutidos en este tema, se pueden aplicar a la imagen. La fuente de la imagen puede configurarse en un archivo XML (carpeta de layout) o programarse en código Java.[10]

- **Button:** Un botón con texto o un icono (o ambos) comunica qué acción se llevará a cabo cuando el usuario haga clic en él.

- **ContentResolver:** El objeto ContentResolver analiza la autoridad del URI y la usa para "resolver" el proveedor comparando la autoridad con una tabla del sistema de proveedores conocidos. Después, el ContentResolver puede enviar los argumentos de la consulta al proveedor correcto.

- **MediaPlayer:** MediaPlayer es una API de Android que nos permite reproducir contenido multimedia (video o audio). Podemos reproducir, tanto archivos almacenados en el dispositivo como desde un flujo de datos que llega a través de Internet.

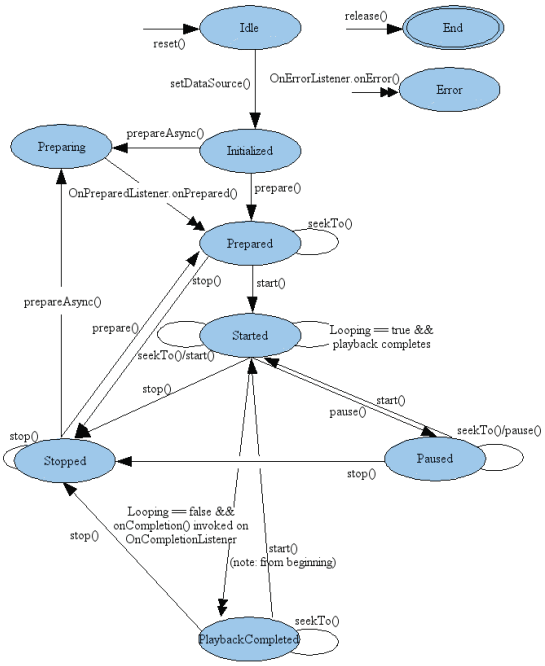

Ilustración 4 Diagrama de funcionamiento de MediaPlayer

[9] TextView información extraída de (https://www.develou.com/textview-en-android/).
[10] ImageView información extraída de (https://learntutorials.net/es/android/topic/4709/imageview).

- **Toast:** Es un aviso proporciona información simple sobre una acción en una pequeña ventana emergente. Solo ocupa la cantidad de espacio necesario para el mensaje, y la actividad en curso permanece visible y admite la interacción. Los avisos desaparecerán automáticamente después de que haya pasado el tiempo de espera.[11]

2.2 Herramienta "Android Studio"

Android Studio [1] es el entorno de desarrollo integrado oficial para la plataforma Android. Se anunció el 16 de mayo de 2013 en la conferencia Google I/O y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones de Android. La primera versión estable se lanzó en diciembre de 2014.

Está basado en el software IntelliJ IDEA de JetBrains y ha sido publicado de forma gratuita a través de la Licencia Apache 2. Está disponible para las plataformas GNU/Linux, macOS, Microsoft Windows y Chrome OS. Ha sido diseñado específicamente para el desarrollo de Android.

Desde el 7 de mayo de 2019, Kotlin [4] es el lenguaje preferido de Google para el desarrollo de aplicaciones de Android. Aun así, Android Studio admite otros lenguajes de programación, como Java y C ++.[12]

En la última versión proporciona las siguientes características:

- Soporte para construcción basada en Gradle.
- Refactorización específica de Android y arreglos rápidos.
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones y otros problemas.
- Plantillas para crear diseños comunes de Android y otros componentes.
- Un editor de diseño enriquecido que permite a los usuarios arrastrar y soltar componentes de la interfaz de usuario.

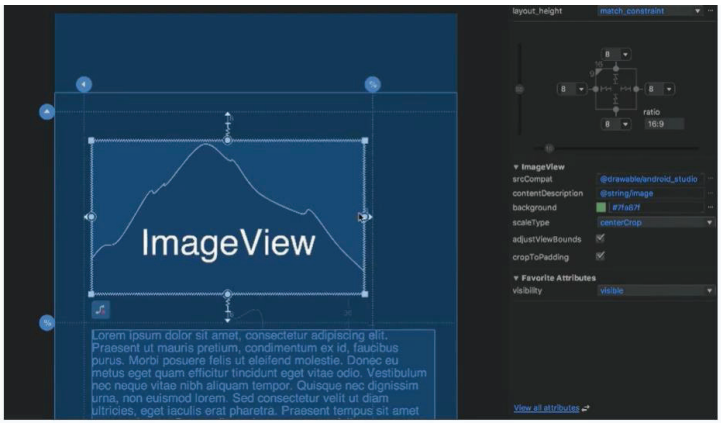

Ilustración 5 Imagen del editor de diseño

[11] Toast información extraída de (https://www.programadornovato.com/%F0%9F%93%B1-toast-avisos-en-android-studio-06/).

[12] Android Studio información extraída de (https://es.wikipedia.org/wiki/Android_Studio).

- Soporte para programar aplicaciones para Android Wear [5].
- Un dispositivo virtual de Android que se utiliza para ejecutar y probar aplicaciones.

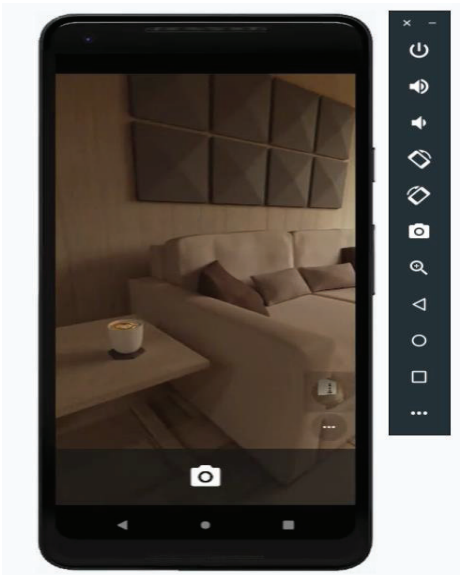

Ilustración 6 Imagen del emulador de Android Studio

- Renderizado en tiempo real.
- Consola de desarrollador: consejos de optimización, ayuda para la traducción, estadísticas de uso.

2.3 Java

Java es un lenguaje de programación y una plataforma informática que fue lanzado por primera vez en 1995 por Sun Microsystems. Java es rápido, seguro y fiable. Desde computadoras portátiles hasta centros de datos y consolas de juegos, computadoras avanzadas y teléfonos móviles e Internet. Java está en todas partes. Si es ejecutado en una plataforma, no es necesario volver a compilarlo para correr en otra. A partir de 2012, Java es uno de los lenguajes de programación más populares utilizados, especialmente para aplicaciones web de servidor de cliente, con unos diez millones de usuarios.

La primera característica, orientada a objetos, se refiere a un método de diseño y programación del lenguaje. Aunque hay muchas explicaciones para la orientación a objetos, la primera idea es diseñar programas de tal manera que los diferentes tipos de datos que utilizan estén relacionados con sus operaciones. Así, los datos y el código (función o método) se combinan en entidades llamadas como objetos.

La segunda característica, la independencia de la plataforma, significa que los programas escritos en Java pueden ejecutarse bien en cualquier tipo de

dispositivo. Este es el significado de ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo.[13]

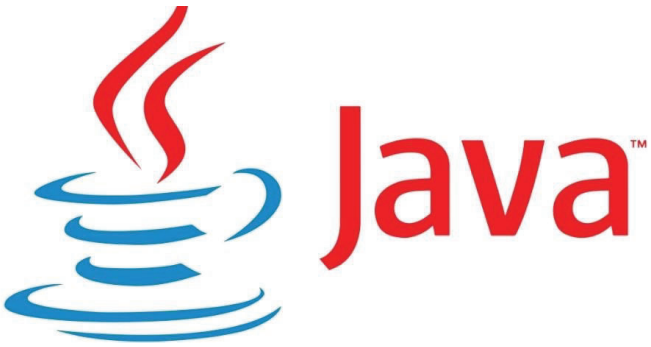

Ilustración 7 Logo de JAVA

2.4 XML (Extensible Markup Language)

XML es el acrónimo de Extensible Markup Language, es decir, es un lenguaje de marcado que define un conjunto de reglas para la codificación de documentos.

Un lenguaje de marcado es un conjunto de código que se puede aplicar en el análisis de datos o la lectura de textos creados por computadoras o personas. XML proporciona una base para definir elementos para crear un formato y general un lenguaje personalizado.

El archivo XML se divide en dos partes: prolog y body. La parte prolog incluye metadatos administrativos, como declaraciones XML, instrucciones de procesamiento opcionales, declaraciones de tipo de documento y comentarios. La parte body consta de dos partes: la estructura y el contenido (presentado en textos simples).[14]

Ilustración 8 Imagen de fichero de XML

[13] Java información extraída de (https://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29).

[14] XML información extraída de (https://ejemplos.net/que-significa-formato-xml/).

2.5 Git

Hoy en día, Git [6] es, con diferencia, el sistema de control de versiones moderno más utilizado del mundo. Git es un proyecto de código abierto maduro y mantenido activamente, desarrollado originalmente por Linus Torvalds, creador del popular sistema operativo kernel de Linux, en 2005. Una cantidad impresionante de proyectos de software empresarial que dependen de Git para el control de versiones, incluidos proyectos comerciales y de código abierto. Los desarrolladores que han trabajado con Git cuentan con una buena representación en la base de talentos disponibles para el desarrollo de software, y Git funciona perfectamente en múltiples sistemas operativos y entornos de desarrollo integrados (IDE).

Git, que tiene una arquitectura distribuida, es un ejemplo de un sistema de control de versiones distribuido (DVCS). En lugar de tener un espacio único para todo el historial de versiones del software, como es común en los sistemas de control de versiones populares como CVS o Subversión (también conocido como SVN), en Git, el código de cada desarrollador también es un archivo que puede mantener un historial completo de todos cambios.

Además de contar con una arquitectura distribuida, Git se ha diseñado teniendo en cuenta el rendimiento, la seguridad y la flexibilidad.[15]

Ilustración 9 Logo de Git

2.6 características de la aplicación.

2.6.1 Restricciones

- El sistema operativo del dispositivo tiene que ser “Android”, versión 5.0 Lollipop o superior.
- La aplicación se ejecutará en el local, por tanto, no es necesario tener una conexión a internet.
- Permitir los accesos de lecturas y escrituras de la aplicación en la memoria.

[15] Git información extraído de(https://www.atlassian.com/es/git/tutorials/what-is-git)

2.6.2 Entorno de desarrollo

- Sistema operativo: Windows10 Home, procesador basado en x64.
- Herramienta del desarrollo: Android Studio Artic Fox Patch 4.
    - Runtime versión: OpenJDK 64-Bit.
- Entorno de ejecución: Android 11.

3 Análisis de requisitos

3.1 Requisitos

Este apartado recopila los requisitos que deben tener la aplicación.

Los requisitos que debe desarrollar son los siguientes:

- La aplicación es capaz de leer los archivos de tipo Media.Audio en la memoria interna del dispositivo o en la memoria externa del dispositivo, por ejemplo “SD Card”.
- Mostrar una lista de canciones con las informaciones adicionales como nombre, cantante, álbum y duración de cada canción.
- El usuario podrá deslizar la pantalla para navegar la lista de música.
- El usuario podrá seleccionar una canción de la lista para reproducir desde principio automáticamente, sin pulsar el botón de “Play”.
- El usuario podrá parar la canción que está reproduciendo.
- El usuario podrá continuar con la canción que está en pausa, reproducir a partir del progreso que se ha reproducido.
- El usuario podrá seleccionar el modo de reproducción (aleatorio u ordenado).
- El usuario podrá reproducir siguiente canción, si el modo de reproducción es ordenado, reproducirá la siguiente canción a partir de la lista de reproducción, si está en modo aleatorio, reproducirá una canción aleatoria desde la lista de reproducción.
- El usuario podrá reproducir la canción previa, si el modo de reproducción es ordenado, reproducirá previa canción a partir de la lista de reproducción, si está en modo aleatorio, reproducirá una canción aleatoria desde la lista de reproducción.
- La aplicación mostrará un aviso proporciona información simple sobre una acción en una pequeña ventana emergente.
    - Si el usuario pulsa el botón de “play” sin seleccionar una canción, la aplicación proporcionará un aviso.

- Si el usuario ha seleccionado la primera canción y pulsa el botón de “last”, la aplicación proporcionará un aviso.
- Si el usuario ha seleccionado la última canción y pulsa el botón “next”, la aplicación proporcionará un aviso.

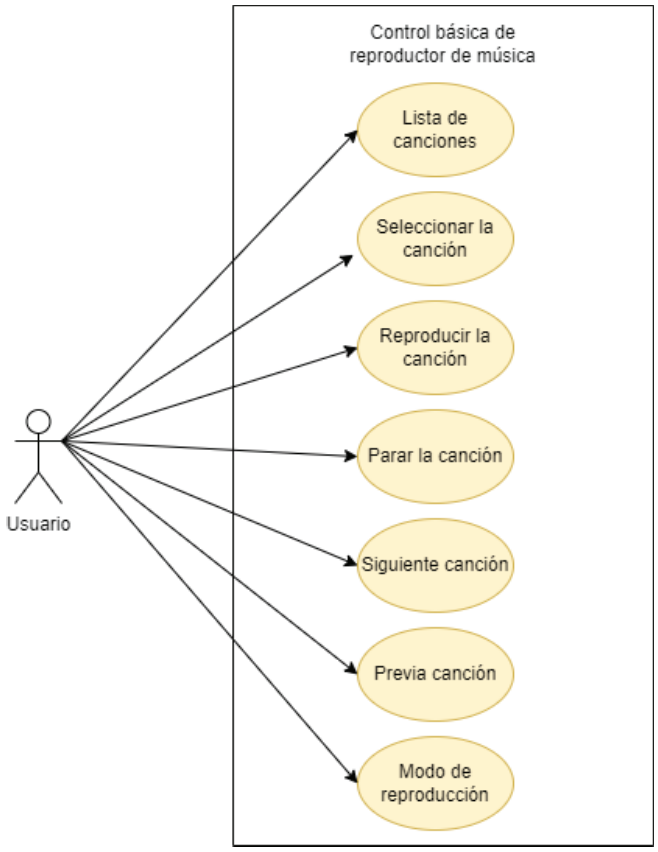

Ilustración 10 Diagrama de control

3.2 Tablas de funcionamientos

REQ01	Lectura de los archivos
Descripción	La aplicación es capaz de leer los archivos de tipo Media.Audio en la memoria
Dependencias	• Permisos de lectura y escritura en la memoria
comentario	La aplicación es capaz de leer cualquier tipo de audio de diferentes duraciones y diferentes ubicaciones en la memoria interna o externa (SD card).

REQ02	Mostrar las canciones
Descripción	Mostrar una lista de canciones con las informaciones relacionados con las canciones.

Dependencias	• REQ01
comentario	Mostrar una lista de canciones con las informaciones adicionales como nombre, cantante, álbum y duración de cada canción. El usuario podrá deslizar la pantalla para navegar las canciones. Si las informaciones son demasiadas largas, las descripciones se desplazarán automáticamente para mostrar las informaciones adicionales.

REQ03	Reproducción de la canción
Descripción	El usuario podrá reproducir la canción
Dependencias	• REQ01 • REQ02 • Haber seleccionado una canción en la lista de reproducción o la canción en pausa
comentario	El usuario podrá reproducir una canción automáticamente desde la lista de reproducción, o continuar con una canción que está en pausa, en este caso no hay seleccionar ninguna canción previamente.

REQ04	Parar la reproducción
Descripción	El usuario podrá para la canción
Dependencias	• REQ03 • Una canción en reproducción.
comentario	El usuario podrá parar la reproducción de música pulsando el botón de “pause”. La canción debe ser estado reproducción.

REQ05	Modo de reproducción
Descripción	El usuario podrá elegir el modo de reproducción.
Dependencias	• REQ01 • REQ02
comentario	Hay dos modos para elegir: seguir el orden de la lista de reproducción u aleatorio.

REQ06	Siguiente canción
Descripción	El usuario podrá reproducir la canción siguiente.

Dependencias	• REQ01 • REQ02 • REQ03 • REQ05
comentario	El usuario podrá reproducir la canción siguiente depende del modo de reproducción (orden de la lista de reproducción u aleatorio). Si está en la última canción, además en modo ordenado, si el usuario pulsa el botón "next", la aplicación deberá de proporcionar un aviso de "Estás en la última canción".

REQ07	Previa canción
Descripción	El usuario podrá reproducir la canción anterior.
Dependencias	• REQ01 • REQ02 • REQ03 • REQ05 • Una canción en reproducción.
comentario	El usuario podrá reproducir la canción anterior depende del modo de reproducción (orden de la lista de reproducción u aleatorio). Si está en la primera canción, además en modo ordenado, si el usuario pulsa el botón "last", la aplicación deberá de proporcionar un aviso de "Estás en la primera canción".

REQ08	Aviso
Descripción	La aplicación proporcionará una pequeña ventana de aviso
Dependencias	• REQ06 • REQ07
comentario	La aplicación proporcionará una pequeña ventana de aviso depende de las situaciones. • El usuario haber elegido la primera canción y pulsa "last". • El usuario haber elegido la última canción y pulsa "next". • El usuario pulsa "play", sin haber elegido una canción previamente.

3.3 Diagrama del funcionamiento y estructura de la aplicación

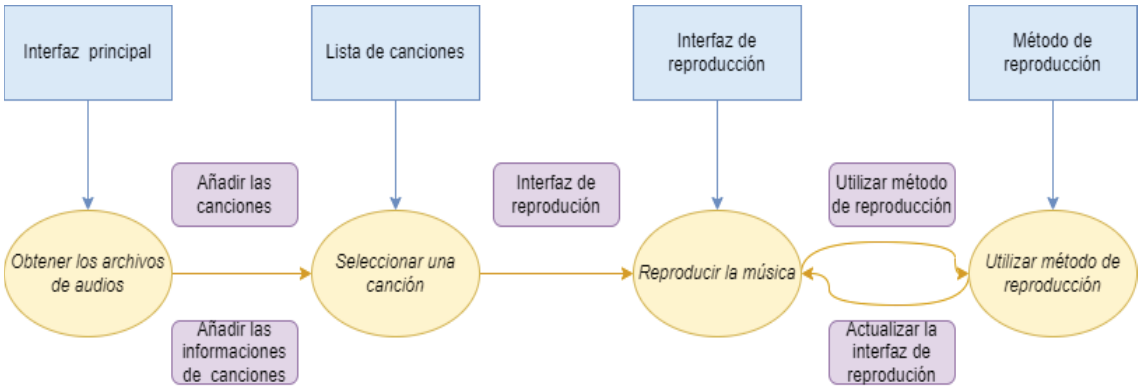

Ilustración 11: diagrama de funcionamiento

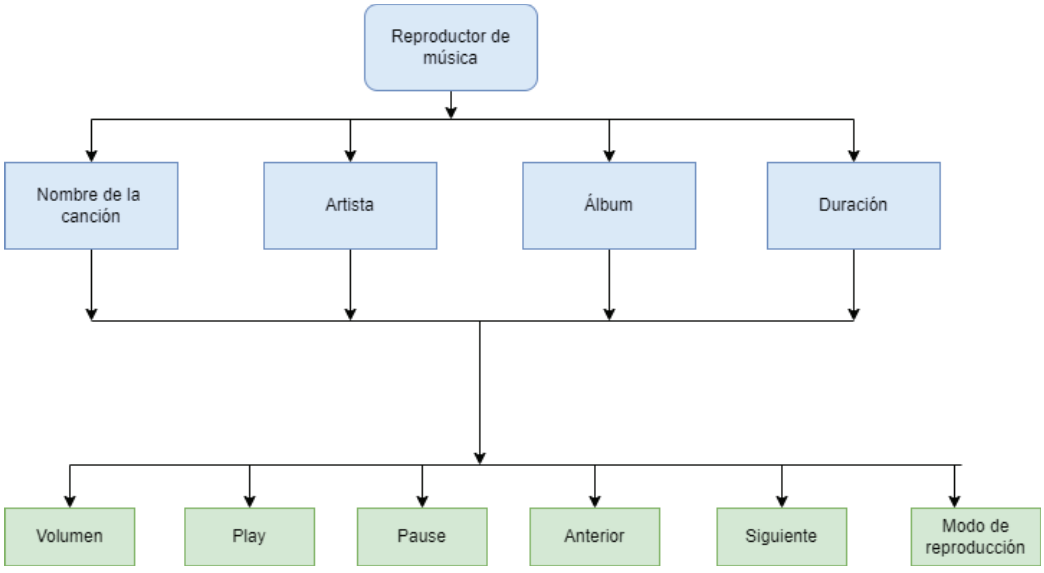

Ilustración 12 Diagrama de estructura

4 Desarrollo

4.1 Diseño

Esta aplicación no solo requiere funcionalidades perfectas, sino que también tiene una interfaz amigable y sencilla, por lo tanto, para un diseño exitoso, los módulos funcionales son los claves. En el apartado de análisis de requisitos, se define que es lo que proporciona la aplicación. Lo que tenemos que hacer ahora diseñar los modelos, la arquitectura general del sistema.

Lo más importante en el diseño de aplicación es la modularización de la aplicación. La modularización se refiere al proceso de dividir un sistema de en varios módulos capa por capa al resolver un problema complejo. Cada módulo cumple una función específica, y todos los módulos están organizados de cierta manera para completar las funciones requeridas por el sistema.

El propósito de dividir el sistema en múltiples módulos es reducir la complejidad del sistema, mejorar la legibilidad y el mantenimiento, pero la división de los módulos no puede ser arbitraria y debe mantenerse lo más independiente posible. Es decir, cada módulo solo realiza las subfunciones independientes requeridas por el sistema, y tiene la menor conexión con otros módulos y una interfaz simple, es decir, intenta lograr una alta cohesión y un bajo acoplamiento, mejorar la independencia de los módulos, y diseñar estructuras de software de alta calidad.

4.1.1 Esquema de la página principal

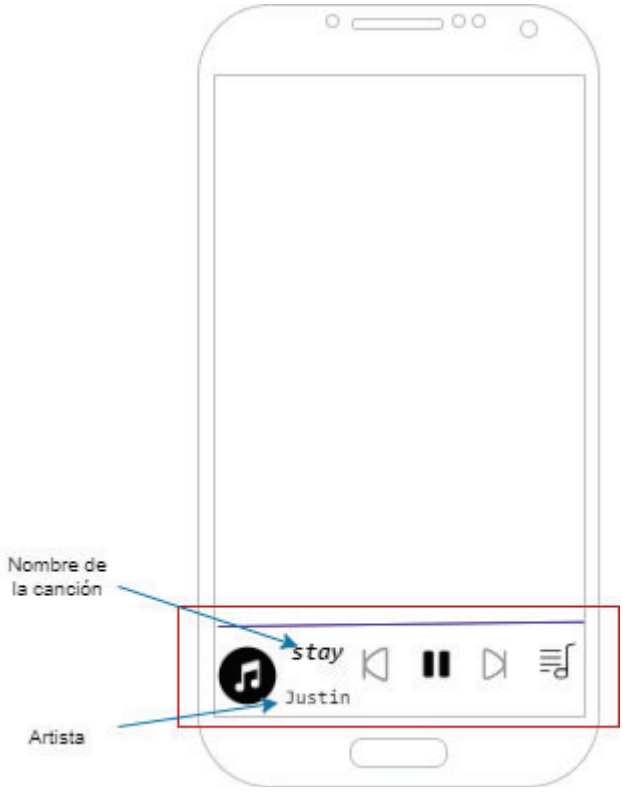

Ilustración 13 Esquema de página inferior

Esquema de página inferior

En la parte inferior de la pantalla, presentará un icono de la música, y cuatros botones.

- El botón de la canción previa.
- El botón de la canción siguiente.
- El botón de play/pause.
- El botón de modo de reproducción.

Entre el icono de la música y el botón de la canción previa, aparecerá el nombre de la canción que está reproduciendo y el nombre del artista.

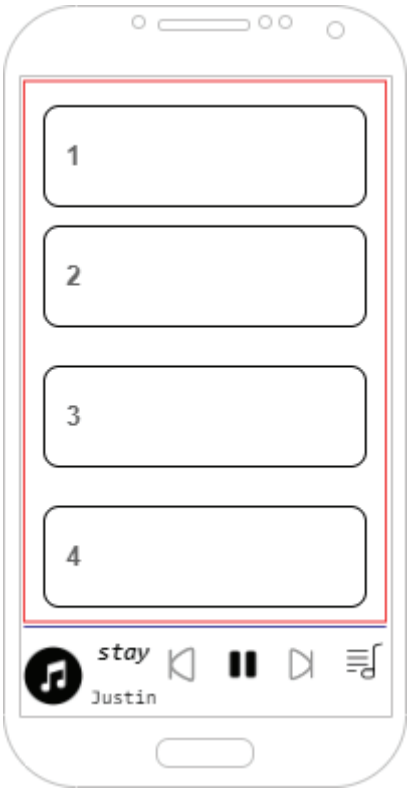

Ilustración 14 Esquema de página superior

Esquema de página superior

En la parte superior de la pantalla, presentará una lista con las canciones ordenadas mediante los números.

Para visualización de la lista, cada componente de la lista tiene forma de tarjeta. Los usuarios podrán deslizar la pantalla para navegar la lista de canciones.

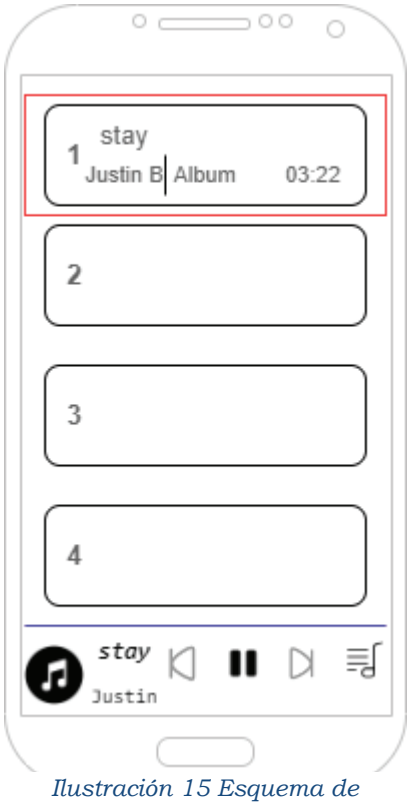

Ilustración 15 Esquema de componente

Esquema de componente

Cada componente de la lista tendrá el nombre de la canción, el artista de la canción, el álbum de la canción y la duración de la canción.

Ilustración 16 Esquema de aviso

Esquema de aviso

En la parte inferior de la pantalla, proporcionará una pequeña ventana de notificación/ aviso. Cuando el usuario realiza algunas acciones.

4.2 Implementación

4.2.1 Creación del proyecto

1. Pulsar ***File -> New -> New Project.*** *Mostrará una ventana de elección de activy.*

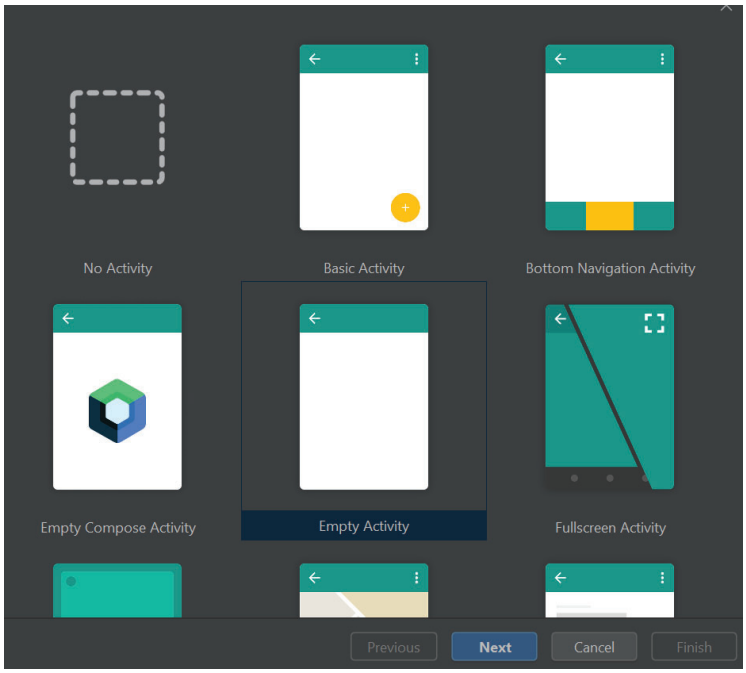

Ilustración 17 Imagen de la creación del proyecto

2. Cliquear en **Next**, y rellenar los campos necesarios. En la versión de Android elegimos 5.0 “Lollipop”.

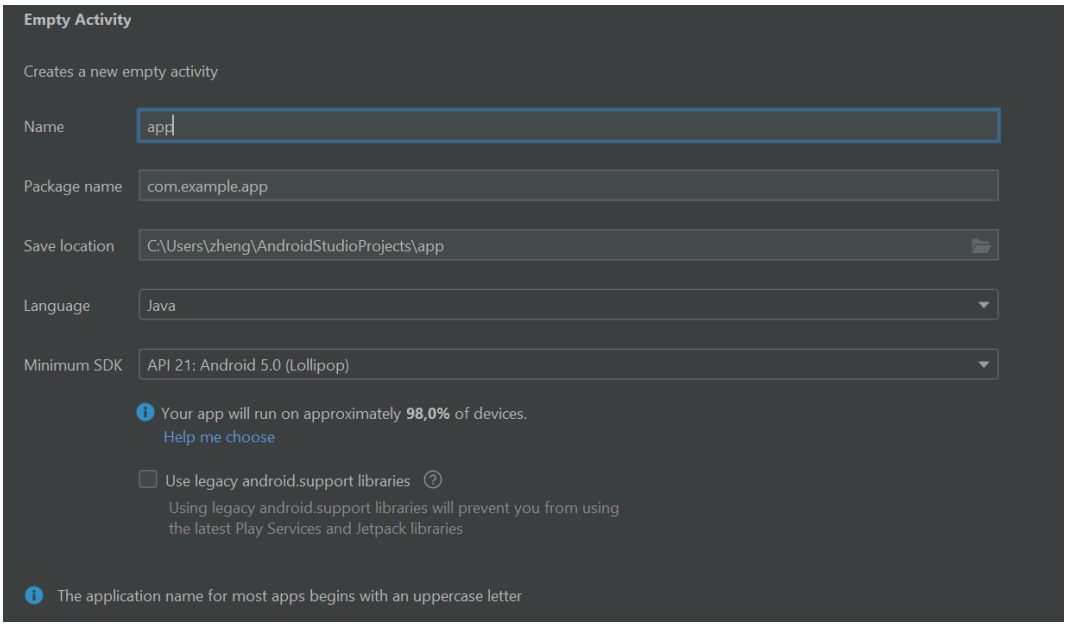

Ilustración 18 Campos a rellenar para la creación del proyecto

3. Cliquear en **Finish**, se creará el proyecto automáticamente.

4. Una vez finalizado, en la parte izquierda mostrará la estructura del proyecto.

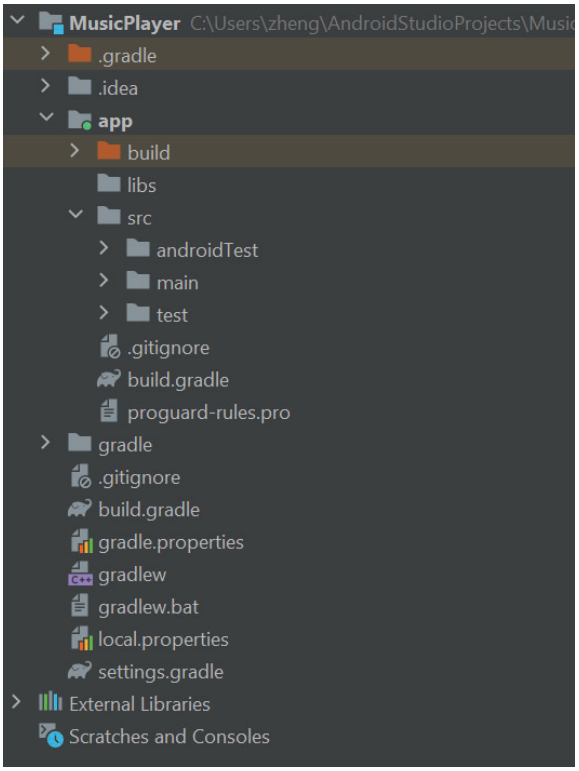

Ilustración 19 Imagen de la estructura del proyecto

Normalmente la mayor parte del tiempo y fuerzas las usaremos en la carpeta src(Source). Dentro de ella se encuentra la carpeta main, la cual contiene todos los archivos fuente Java para nuestra aplicación. La carpeta res (Resources) que contiene los recursos necesarios para el proyecto (iconos, sonido, diseños, etc.) y el archivo AndroidManifest.xml.

AndroidManifest es un archivo XML que contiene nodos descriptivos sobre las informaciones de una aplicación Android. Como los building blocks existentes, la versión de SDK usada, los permisos necesarios para ejecutar algunos servicios y muchas más.

Para este proyecto habría que añadir los permisos de lectura y escritura en la memoria de la tarjeta externa (SD card).

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.musicplayer">
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Music Player"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:requestLegacyExternalStorage="true"
        android:theme="@style/Theme.MusicPlayer">
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

Ilustración 20 AndroidManifiest.xml

En la carpeta layout encontrarás los archivos de diseño de todas tus actividades.En este proyecto existe el archivo activiy_main.xml e item_music.xml. Estos archivos representan el diseño de la interfaz de mi actividad principal.

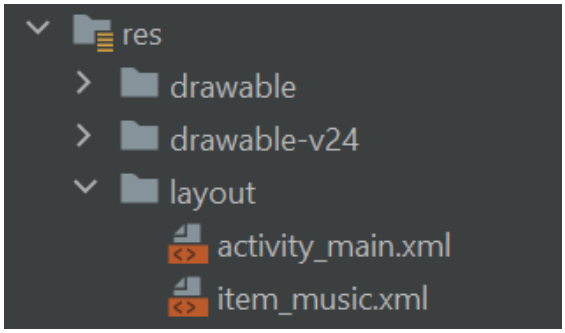

Ilustración 21 Imagen de la carpeta layout

4.2.2 Interfaz principal

Esta aplicación es sencilla, está dividida en dos partes. En la parte superior está formada por una lista de las canciones, parte inferior por los botones de interacción.

4.2.2.1 Preparación de las imágenes

En este proyecto necesita 4 imágenes para los botones, 1 imagen para el fondo de la aplicación, 1 icono de decoración, todos en formatos png, guardado en el directorio res/drawable.

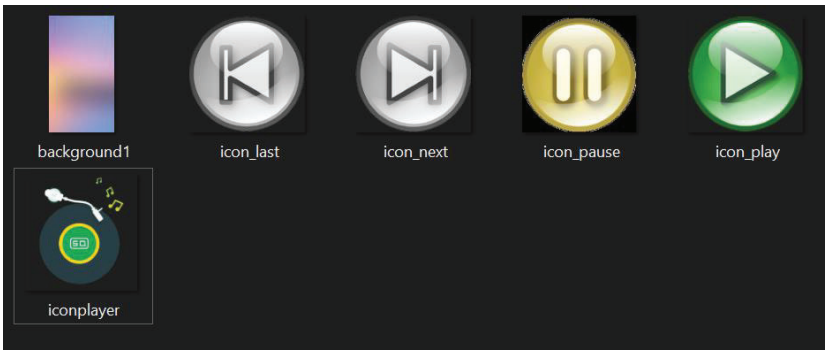

Ilustración 22 Recursos de las imágenes

4.2.2.2 Preparación de los recursos de String

Todos los recursos tercerizados para nuestro proyecto se encuentran dentro de la carpeta "res". La exclusión de los atributos de la aplicación a través de estos archivos externos reduce la complejidad del diseño en las interfaces.

El archivo string.xml es uno de los recursos más importantes, se encuentra dentro de la carpeta "values". Este fichero contiene todas las cadenas de caracteres que se muestran en los widgets (controles, formas, botones, vistas, etc.) de nuestras actividades.

En la parte izquierda, si realizamos doble clic en el archivo string.xml, podemos añadir los recursos que queremos. Para declarar las cadenas usaremos la etiqueta <string> y estableceremos el atributo name como identificador. Dentro de esta etiqueta pondremos la cadena de caracteres que se visualizará en los componentes de interfaz.

```
<resources>
    <string name="app_name">Music Player</string>
    <string name="stop">parar</string>
    <string name="next">siguiente</string>
    <string name="last">anterior</string>
    <string name="play">continuar</string>
</resources>
```

Ilustración 23 Recursos de string

4.2.2.3 Implementación del diseño

Cada interfaz de Android tiene su archivo de configuración de diseño, que contiene varios Layout de diseño y referencias a varios archivos de recursos, como imágenes, texto y colores. Cuando la aplicación se está ejecutando, puede usar el código para configurar el diseño de la interfaz. De esta manera, se pueden formar diferentes interfaces visuales y efectos. La interfaz principal del reproductor es una clase Activity. El proyecto de Android primero ejecutará el método Oncreate() cuando ejecute cada Activity, de la siguiente manera:

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
```

Ilustración 24 Imagende OnCreate()

Este método realiza principalmente la inicialización de la interfaz. Activity tiene un método para configurar el diseño: SetContentView(layoutResID), el parámetro es el ID del recurso, el ID está en el directorio del proyecto res/layout y el archivo de diseño de la interfaz principal el nombre es activity_main.xml.

Para la interfaz principal se ha dividido en dos partes, las dos partes están utilizado RelativeLayout. En la parte superior contiene una lista de las canciones con el nombre de la artista, nombre del álbum, nombre de la canción y la duración de la canción. En la parte inferior contiene el nombre de la canción que está reproduciendo y la artista, los botones (next, last, pause, play, mode) y un icono del disco.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/background1">
<!--parte inferior-->
    <RelativeLayout
        android:id="@+id/bottomLayout"
        android:layout_width="match_parent"
        android:layout_height="70dp"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="1dp"
        android:background="#33EEEEEE">
<!--La linea de division entre parte superior y parte interior-->
        <ImageView
            android:layout_width="match_parent"
            android:layout_height="0.5dp"
            android:background="#9933FA" />
```

Ilustración 25 Implementación del parte inferior

Ilustración 26 Resultado del parte inferior en el emulador

- **Parte inferior**

  En la parte inferior, utilizando ImageView para mostrar un icono del disco centralizado verticalmente.

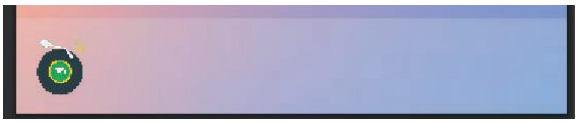

Ilustración 27 Resultado de la imagen del disco

```
<ImageView
    android:id="@+id/bottom_iv_main"
    android:layout_width="50dp"
    android:layout_height="50dp"
    android:layout_centerVertical="true"
    android:layout_marginLeft="8dp"
    android:src="@mipmap/iconplayer" />
```

Ilustración 28 Implementación de la imagen del disco

En la parte derecha de la imagen del icono, utilizando TextView para mostrar el nombre de la canción que está reproduciendo y la artista.

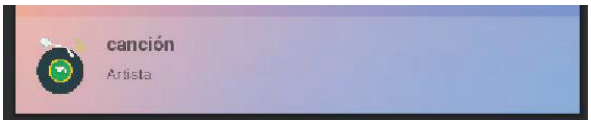

Ilustración 29 Resultado del nombre de la canción y la artista

```
<TextView                                          ⚠ 8  ✓ 3
    android:id="@+id/bottom_tv_song"
    android:layout_width="85dp"
    android:layout_height="wrap_content"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="10dp"
    android:layout_toRightOf="@id/bottom_iv_main"
    android:ellipsize="marquee"
    android:singleLine="true"
    android:text="canción"
    android:textSize="14sp"
    android:textStyle="bold" />

<TextView
    android:id="@+id/bottom_tv_singer"
    android:layout_width="75dp"
    android:layout_height="wrap_content"
    android:layout_below="@id/bottom_tv_song"
    android:layout_alignLeft="@id/bottom_tv_song"
    android:layout_marginTop="5dp"
    android:ellipsize="marquee"
    android:singleLine="true"
    android:text="Artista"
    android:textSize="11sp" />
```

Ilustración 30 Implementación del nombre de la canción y la artista

Los más importantes de esta parte son los botones (next, last, play, mode) estas imágenes están guardadas en res/drawable, los botones están alineados con el borde lateral derecho al mismo nivel vertical.

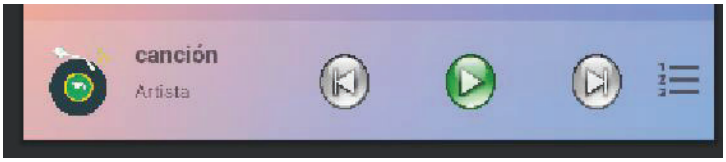

Ilustración 31 Resultado de los botones

```
<Button
    android:id="@+id/bottom_iv_play"
    android:layout_width="60dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="3dp"
    android:layout_marginRight="15dp"
    android:layout_toLeftOf="@id/bottom_iv_next"
    android:background="@drawable/icon_play"
    android:scaleX="0.5"
    android:scaleY="0.5" />
```

Ilustración 32 Implementación de la imagen de play

- **Parte superior**

En esta parte se ha utilizado RecycleView, ya que Recycleview se renderizan los elementos de la lista, los elementos que dejan de observarse se reciclan para mostrar los elementos siguientes.

Si bien un RecyclerView representa una lista de elementos, cada elemento debe tener una UI definida.Al usar Material Design se suele usar la clase CardView para definir la apariencia de cada elemento de un listado

Lo primero es definir un RecyclerView en nuestro Layout principal "activity_main.xml".

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/rv"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_above="@id/bottomLayout" />
```

Ilustración 33 Implementacion del layout superior

Una vez obtenemos la referencia del RecycleView, le asignamos un layout manager, y le asociamos un adapter (es un mecanismo de Android que hace de puente entre nuestros datos y las vistas contenidas en un ListView).

```
adapter = new MusicAdapter( context: this, musicData);
recyclerView.setAdapter(adapter);
LinearLayoutManager linearLayoutManager = new LinearLayoutManager( context: this, RecyclerView.VERTICAL,  reverseLayout: false);
recyclerView.setLayoutManager(linearLayoutManager);
```

Ilustración 34 Implementación del adaptador

Dentro del adapter:

- Contiene una clase interna ViewHolder, que permite obtener referencias de los componentes visuales (views) de cada elemento de la lista. Los componentes son: el nombre de la canción, el artista, el álbum y la duración, estos son los elementos que mostrarán en la lista de canciones.

```
class MusicViewHolder extends RecyclerView.ViewHolder{
    TextView idTv,songTv,singerTv,albumTv,timeTv;
    public MusicViewHolder(View itemView) {
        super(itemView);
        idTv = itemView.findViewById(R.id.item_music_number);
        songTv = itemView.findViewById(R.id.item_music_song);
        singerTv = itemView.findViewById(R.id.item_music_singer);
        albumTv = itemView.findViewById(R.id.item_music_album);
        timeTv = itemView.findViewById(R.id.item_music_duration);
    }
}
```

Ilustración 35 Implementación de ViewHolder

- Presenta un constructor para gestionar la lista de música.

```
public MusicAdapter(Context context, List<MusicBean> musicData) {
    this.context = context;
    this.musicData = musicData;
}
```

Ilustración 36 Constructor del adapter

- En el método onCreateViewHolder hacemos uso de la clase LayoutInflater para "inflar" un layout XML.
  El layout se define como “item_music.xml” la apariencia está representado por diferentes TextView a través de un CardView.

```
public MusicViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view = LayoutInflater.from(context).inflate(R.layout.item_music,parent, attachToRoot: false);
    MusicViewHolder holder = new MusicViewHolder(view);
    return holder;
}
```

Ilustración 37 Implementación de OnCreateViewHolder

```
<androidx.cardview.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginRight="10dp"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="10dp"
    app:contentPadding="10dp"
    app:cardCornerRadius="10dp"
    app:cardElevation="0.5dp"
    app:cardBackgroundColor="#33FFC0CB">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
        <TextView
            android:id="@+id/item_music_number"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text=""
            android:layout_centerVertical="true"
            android:textSize="24sp"
            android:textStyle="bold"/>
```

Ilustración 38 Implementación de los elementos utilizando CardView

- Contiene un método onBindViewHolder que enlaza nuestra data con cada ViewHolder. En este método configura todas las informaciones básicas de cada canción.

```
public void onBindViewHolder(@NonNull MusicViewHolder holder, @SuppressLint("RecyclerView")
final int position) {
    MusicBean musicBean = musicData.get(position);
    holder.idTv.setText(musicBean.getId());
    holder.songTv.setText(musicBean.getSong());
    holder.singerTv.setText(musicBean.getSinger());
    holder.albumTv.setText(musicBean.getAlbum());
    holder.timeTv.setText(musicBean.getDuration());
```

Ilustración 39 Implementación de onBindViewHolder

- Un objeto de escucha de eventos, cuando el usuario toca el elemento de la lista.

```
holder.itemView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) { onItemClickListener.OnItemClick(view,position); }
});
```

Ilustración 40 Implementación de onClick()

4.2.2.4 Obtención de música local

Para reproducir las canciones es imprescindible la obtención de música local.

Se ha creado una clase musicBean con los siguientes atributos:

- id
- song (nombre de la canción)
- singer (nombre de la artista)
- album
- duration (duración)
- path

```
public MusicBean(String id, String song, String singer, String album, String duration, String path) {
    this.id = id;
    this.song = song;
    this.singer = singer;
    this.album = album;
    this.duration = duration;
    this.path = path;

}
```

Ilustración 41 Constructor de MusicBean

Por otra parte, utilizando cursor del fichero para obtener los archivos de tipo "MediaStore.Audio.Media".

```
private void searchLocalMusic() {
    //cargar las musicas locales
    ContentResolver contentResolver = getContentResolver();
    Uri uri = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
    Cursor cursor = contentResolver.query(uri,  projection: null,  selection: null,  selectionArgs: null,  sortOrder: null);
    id = 0;
    while (cursor.moveToNext()) {
        //obtener titulo de la cancion
        @SuppressLint("Range") String song = cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.TITLE));
        // obtener artista de la cancion
        @SuppressLint("Range") String singer = cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.ARTIST));
        //obtener album
        @SuppressLint("Range") String album = cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.ALBUM));
        id++;
        String string_id = String.valueOf(id);
        //obtener path
        @SuppressLint("Range") String path = cursor.getString(cursor.getColumnIndex(MediaStore.Audio.Media.DATA));
        //obtener duracion
        @SuppressLint("Range") long duration = cursor.getLong(cursor.getColumnIndex(MediaStore.Audio.Media.DURATION));
        SimpleDateFormat sdf = new SimpleDateFormat( pattern: "mm:ss");
        String string_duration = sdf.format(new Date(duration));
        MusicBean bean = new MusicBean(string_id, song, singer, album, string_duration, path);
        //añadir a la coleccion de musica
        musicData.add(bean);
    }
```

Ilustración 42 Implementación de searchLocalMusic para buscar la media de audio.

4.2.2.5 Concepto de reproductor de audio en Android

El programa fuente de Android ha preparado una clase de interfaz, llamada MediaPlayer.

A continuación, se muestra un ejemplo de cómo reproducir un archivo de audio que está disponible como recurso local.

1. *MediaPlayer.reset();* operación de reinicio, preparación para que los medios se reproduzcan más tarde.
2. Agregar la fuente de datos de la canción, *MediaPlayer.setDataSource(String path)* el parámetro necesita pasar una ruta, convierta la ruta en una fuente de dato después de recibirla.
3. Preparación de reproducción de fuente de datos, utilizando el método *MediaPlayer.prepare*().
4. *MediaPlayer.start();* eproducir la fuente de datos, después de reproducir la fuente de datos, debe llamar al método *MediaPlayer.stop();* para detenerlo.

4.2.2.6 Implementación de la función Reproducir/Pausar

Puede ver el botón de reproducción en la interfaz de reproducción de música. Cuando haga clic en el botón de reproducción para reproducir la música o haga clic en el botón de pausa, se activa las operaciones de procesamiento, controlando así la reproducción de la música actual.

Cuando el reproductor no está reproduciendo ninguna canción o está en pausa, la interfaz principal deberá de presentar el botón de “play”.

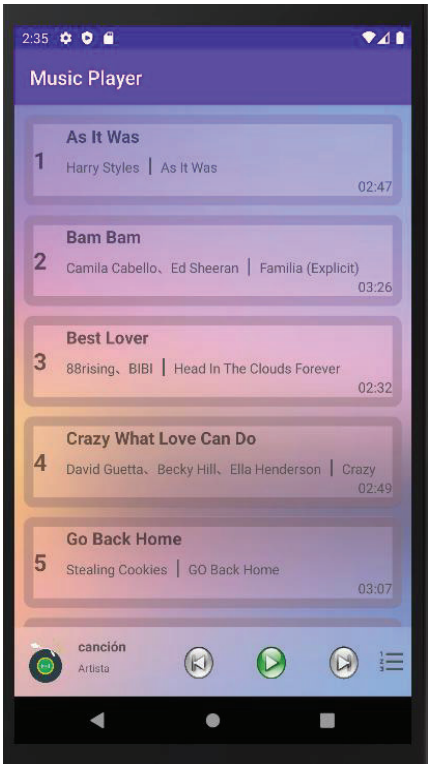

Ilustración 43 Estado de los botones cuando está en pausa

Cuando el usuario haga clic en el botón de “play” o toca cualquier canción de la lista, la interfaz principal deberá de cambiar el botón de “play” a “pause” utilizando la función “setBackgroundResource” con la imagen de “pause” guardada previamente en res/drawable, además, reproducir la música.

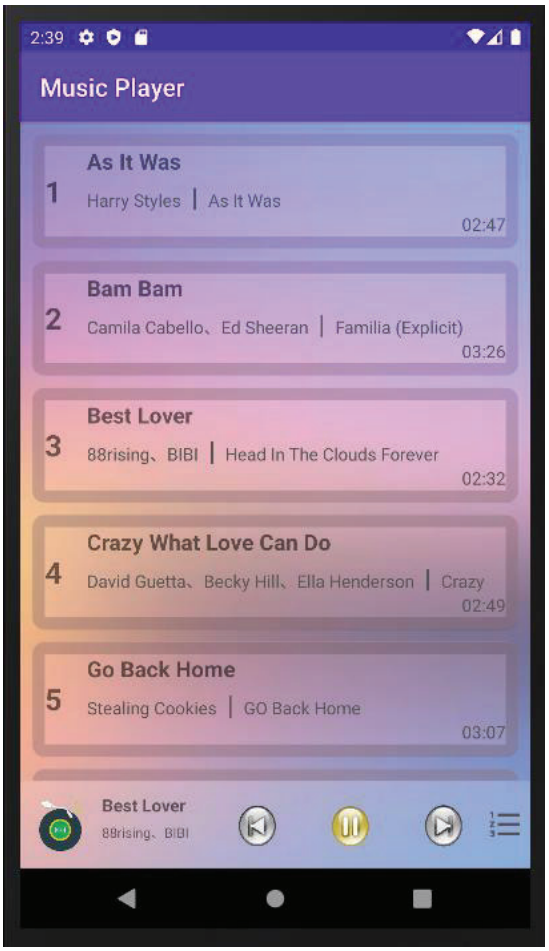

Ilustración 44 Estado de los botones cuando está reproduciendo una canción

Se ha definido una variable (currentSongPos) para guardar el progreso de reproducción, cuando el usuario haga clic en el botón de pausa, se guardará el progreso donde se ha parado. Cuando el usuario elige continuar con la reproducción, la aplicación continuará donde se ha parado.

A continuación, muestra los códigos de implementación.

```
case R.id.bottom_iv_play:
    if (currentSongPos == -1) {
        Toast.makeText( context: this, text: "Hay que elegir una canción previamente", Toast.LENGTH_SHORT).show();
        return;
    }
    if (mediaPlayer.isPlaying()) {
        pauseMusic();
    } else {
        playMusic();
    }
    break;
```

Ilustración 45 Implementación del boton play

```
private void pauseMusic() {
    if (mediaPlayer != null && mediaPlayer.isPlaying()) {
        //guardar el progreso del reproducion.
        currentDuration = mediaPlayer.getCurrentPosition();
        mediaPlayer.pause();
        play.setBackgroundResource(R.drawable.icon_play);
    }
}
```

Ilustración 46 Implementación para pausar la musica

```
private void playMusic() {
    if (mediaPlayer != null && !mediaPlayer.isPlaying()) {
        // reproducir cancion desde principio
        if (currentDuration == 0) {
            try {
                mediaPlayer.prepare();
                mediaPlayer.start();
            } catch (IOException e) {
                e.printStackTrace();
            }
        } else {
            //peroducir la cancion donde se ha parado
            mediaPlayer.seekTo(currentDuration);
            mediaPlayer.start();
        }
        play.setBackgroundResource(R.drawable.icon_pause);
    }
}
```

Ilustración 47 Implementación para reproducir la musica

4.2.2.7 Implementación de modo de reproducción

La aplicación proporcionara un botón, donde los usuarios podrán elegir el modo de reproducción.

- Secuencial: utilizando la variable “currentSongPos” para retorna la canción anterior o siguiente canción.

```
currentSongPos--;
MusicBean bean = musicData.get(currentSongPos);
playSongPosition(bean);
```

Ilustración 48 Implementación del modo secuencial

```
currentSongPos++;
MusicBean bean = musicData.get(currentSongPos);
playSongPosition(bean);
```

Ilustración 49 Implementación de modo secuencial

- Aleatorio: utilizando el método *random() para obtener una posición aleatoria de la lista.*

```
currentSongPos = random();
MusicBean bean = musicData.get(currentSongPos);
playSongPosition(bean);
```

Ilustración 50 Implementación del modo aleatorio

4.2.2.8 Implementación de la canción anterior y siguiente canción

Cuando el usuario haga clic en el botón de “next” o “last”, la aplicación debe de cambiar la canción dependiendo del modo de reproducción.

```
case R.id.bottom_iv_last:
    if (mode.equals("aleatorio")) {
        currentSongPos = random();
        MusicBean bean = musicData.get(currentSongPos);
        playSongPosition(bean);
    } else {
        if (currentSongPos == 0) {
            Toast.makeText( context: this,  text: "Estás en la primera canción", Toast.LENGTH_SHORT).show();
            return;
        } else {
            currentSongPos--;
            MusicBean bean = musicData.get(currentSongPos);
            playSongPosition(bean);
        }
    }
```

Ilustración 51 Implementación de la canción anterior

```
case R.id.bottom_iv_next:
    if (mode.equals("aleatorio")) {
        currentSongPos = random();
        MusicBean bean = musicData.get(currentSongPos);
        playSongPosition(bean);
    } else {
        if (currentSongPos == musicData.size() - 1) {
            Toast.makeText( context: this,  text: "Estás en  la última canción", Toast.LENGTH_SHORT).show();
            return;
        } else {
            currentSongPos++;
            MusicBean bean = musicData.get(currentSongPos);
            playSongPosition(bean);
        }
    }
```

Ilustración 52 Implementación de la canción siguiente

4.2.2.9 Implementación de avisos

En esta aplicación deberá de proporcionar algunos avisos correspondientes a las operaciones ejecutados por el usuario.

- Cuando el usuario haga clic en el botón de play, pero sin haber seleccionado una canción previamente. La aplicación proporcionara un aviso en una pequeña ventana con una duración corta.
-

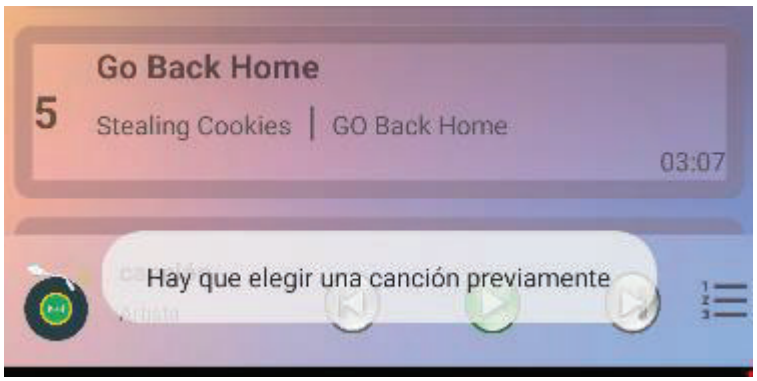

Ilustración 53 Resultado de la ventana de aviso cuando

no se ha seleccionado ninguna canción

```
if (currentSongPos == -1) {
    Toast.makeText( context: this,  text: "Hay que elegir una canción previamente", Toast.LENGTH_SHORT).show();
    return;
}
```

Ilustración 54 Implementación de la ventana para la reproducir una canción

- Cuando el usuario haga clic en el botón de next, pero es la última canción. La aplicación proporcionara un aviso en una pequeña ventana con una duración corta.

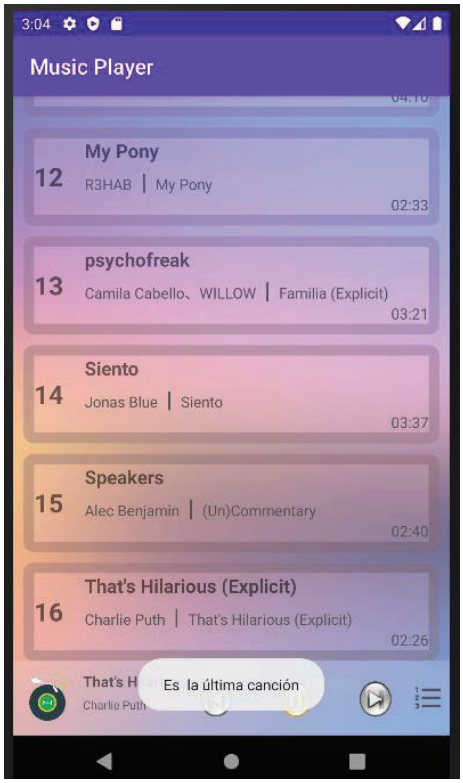

Ilustración 55 Ventana de aviso de la última canción

```
if (currentSongPos == musicData.size() - 1) {
    Toast.makeText( context: this,  text: "Es la última canción", Toast.LENGTH_SHORT).show();
    return;
```

Ilustración 56 Implementación de aviso para la última canción

- Cuando el usuario haga clic en el botón de last, pero es la primera canción. La aplicación proporcionara un aviso en una pequeña ventana con una duración corta.

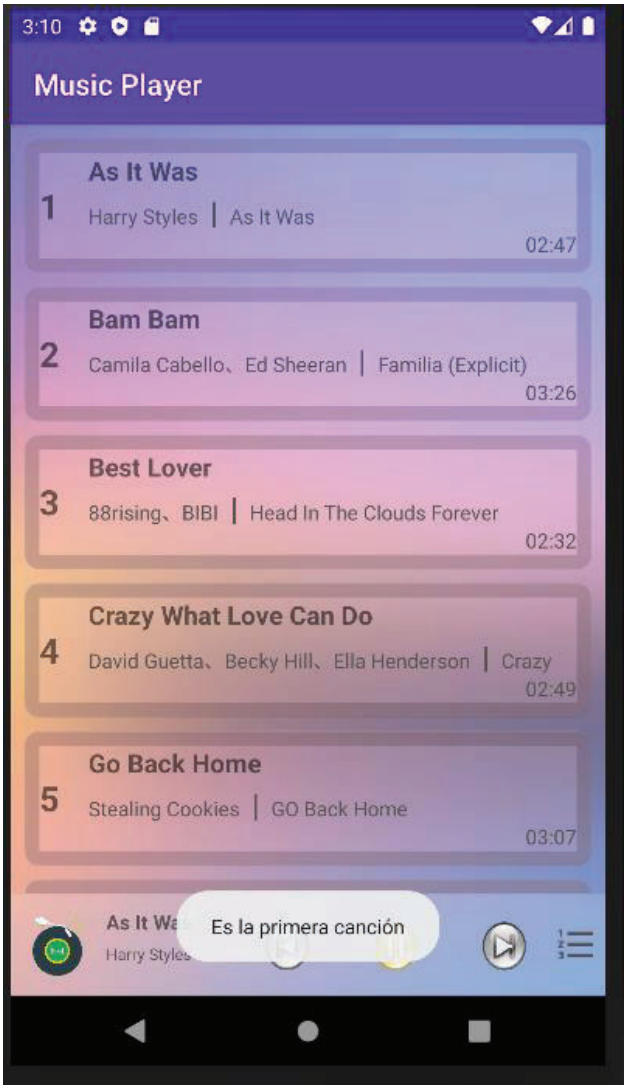

Ilustración 57 Ventana de aviso de la primera canción

```
if (currentSongPos == 0) {
    Toast.makeText( context: this,  text: "Es la primera canción", Toast.LENGTH_SHORT).show();
    return;
```

Ilustración 58 Implementación de aviso para la primera canción

5 Pruebas

Para este proyecto, se ha utilizado dos dispositivos muy distintos. Para garantizar las funcionalidades básicas de la aplicación en los diferentes dispositivos, la compatibilidad de la pantalla y entre las versiones de Android.

Emulador de Android Studio.
Nexus 5X api 30, 1080 x 1920: 420dpi
Android 11(x86).

Ilustración 59 Imagen del emulador Nexus 5x

SAMSUNG GALAXY Tab A8 2019

1280 x 800 Andoid 9.

Ilustración 60 Imagen de Samsung Galaxy tab a8 2019

Las pruebas tienen una gran cantidad de trabajos y un proceso complejo. Además, necesitamos empezar desde diferentes puntos de vista, aclare el propósito de la prueba, refine el proceso de prueba, clasifique claramente el proceso de prueba, trate de ser lo más detallado y complejo posible.

El objetivo principal es la obtención de los audios, sin embargo, en el proceso de la prueba, ha aparecido un fallo de lectura, aunque los audios están guardados en la memoria previamente, como se muestra en la figura.

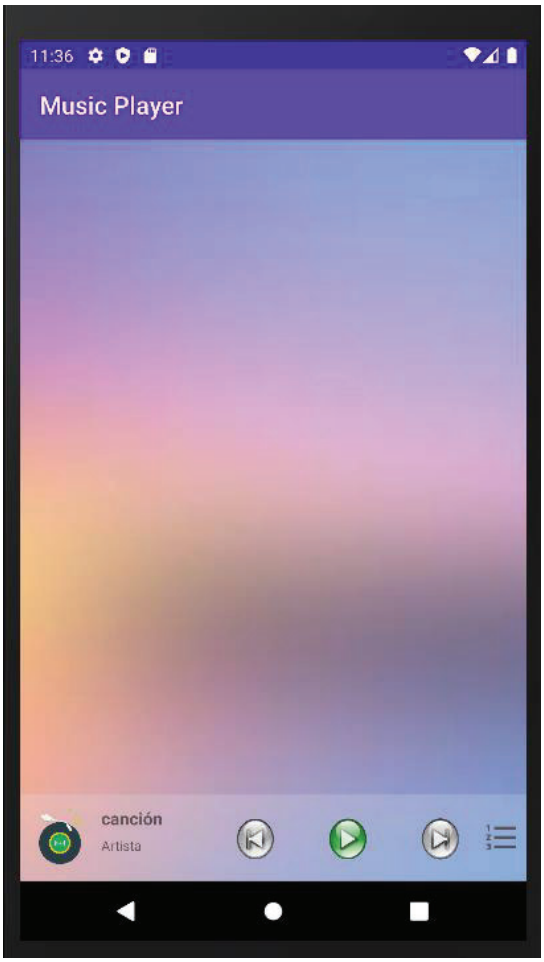

Ilustración 61 Imagen de la interfaz cuando no se ha accedido el permiso de memoria

Ya que los permisos de lectura y escritura no están permitidos en la configuración de la aplicación.

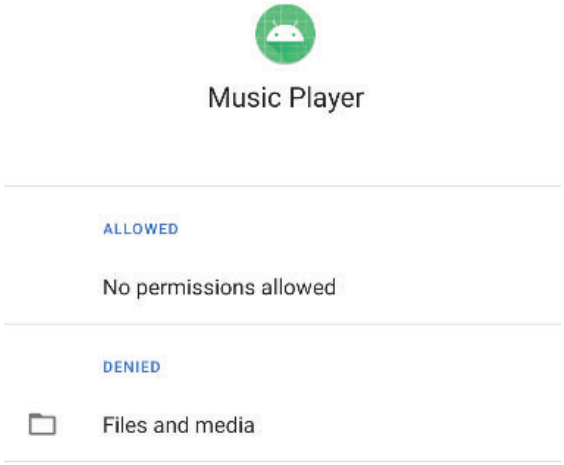

Ilustración 62 Imagen de la configuración de permisos

En la mayoría de los casos, se utiliza Logcat para detectar los fallos, a través de dicho herramientas se ubican los errores, se detectan para la corrección.

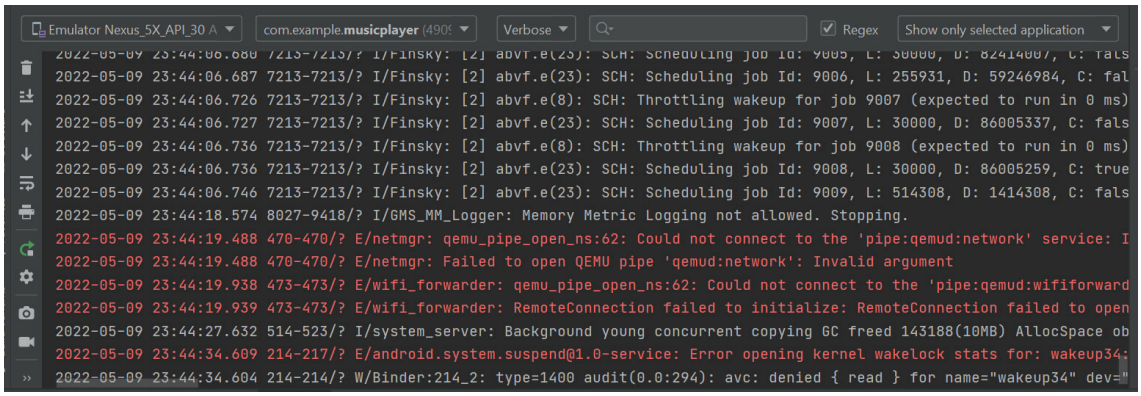

Ilustración 63 Ventana de Logcat

Durante los procesos de pruebas, se ha lanzado excepciones como "NullPointerException", "ArrayIndexOfBoundsException" y "IOException", algunos son fáciles de resolver, algunos son más complicados, pero con las herramientas y debugear los códigos se han resueltos estos problemas, también se ha consultado la documentación API para aclarar las implementaciones.

6 Resultados y conclusiones

A través del desarrollo de reproductor de música local en el sistema operativo Android, he aprendido la programación en Android desde cero hasta un nivel de comprensión relativamente complejo. El desarrollo del reproductor de música se ha completado. Sin embargo, debido a la falta de conocimientos en programación Android, todavía hay muchas deficiencias en muchos aspectos y algunas funciones no se han añadido. Por ejemplo, el reproductor no se ha agregado la sincronización de letras, visualización de la barra de progreso, animación de la imagen de álbum, etc. Las funcionalidades se pueden mejorar e interactuar más con los usuarios.

En cuanto al diseño de la pantalla principal, se ha tomado muchas decisiones para la interfaz del usuario, buscando todas las informaciones y documentaciones, se ha decidido utilizar “RelativeLayout” y “CardView” para la visualización de los elementos de la lista.

En los procesos de diseño y desarrollo, me di cuenta de que los conocimientos solo se pueden aprender con las prácticas, y los conocimientos teóricos no son suficientes para lograr el objetivo, solo combinando las teorías y las prácticas se pueden alcanzar un aprendizaje real.

El desarrollo de la aplicación no se puede hacer en un solo paso, necesita avanzar y mejorar continuamente. Al mismo tiempo, todavía hay muchos problemas en la aplicación, que deben descubrirse y resolverse en el uso futuro. La preparación antes de la programación es un proceso importante para reducir la carga de trabajo de la depuración e implementación del programa. Aunque hice mucha preparación antes de comenzar a programar, todavía se encuentra muchos problemas a la hora de implementar y diseñar. Algunos de los problemas fueron errores en el análisis, y otro solamente se encuentra cuando empecé a programar.

7 Análisis de Impacto

En septiembre de 2015 se establecieron una serie de objetivos globales, una oportunidad para que los países y sus sociedades emprendan un nuevo camino para mejorar la vida de todos. Con metas específicas a conseguir en 15 años, para erradicar la pobreza, proteger el planeta y asegurar la prosperidad para todos.

Entre los impactos de la aplicación, a continuación, muestra los que se destacan.

- Personal: Durante el desarrollo del proyecto, me ha ayudado a organizar y reflexionar sobre un proyecto desde principio. Además, los conocimientos adquiridos sobre la plataforma Android y las experiencias sobre nuevas herramientas.

- Empresarial: La aplicación implica muchos aspectos para una empresa, mejorar y mantener la aplicación. Desarrollar y testear los códigos para nuevas funcionalidades. Por otra parte, obtener derechos de autor de música.

- Social: La aplicación ha cambiado nuestra necesidad de los dispositivos. Evidentemente no hay que llevar un mp3 adicional para escuchar músicas, por lo tanto, la desaparición de mp3 es muy probable.

- Económico: La aplicación es totalmente gratuita, el desarrollo tampoco requiere muchos recursos humanos. Sin embargo, la aplicación puede tener un gran mercado en la plataforma Android y en los consumidores.

- Medioambiental: Es un reproductor local, no requiere Internet ni centros de datos, a nivel usuario, el consumo de la energía de aplicación es mínimo.

8 Bibliografía

[1] «Android Studio,» [En línea]. Available: https://developer.android.com/about. [Último acceso: 02 2022].

[2] «JAVA,» [En línea]. Available: https://www.java.com/es/download/help/whatis_java.html. [Último acceso: 03 2022].

[3] «XML,» [En línea]. Available: https://www.w3.org/standards/xml/core. [Último acceso: 22 03 2022].

[4] «Kotlin,» [En línea]. Available: https://kotlinlang.org/docs/home.html. [Último acceso: 05 2022].

[5] «Android wear,» [En línea]. Available: https://developer.android.com/wear. [Último acceso: 05 2022].

[6] «Git,» [En línea]. Available: https://git-scm.com/. [Último acceso: 02 2022].

[7] «AppsFlyer,» [En línea]. Available: https://www.appsflyer.com/es/. [Último acceso: 03 2022].

[8] «Android,» [En línea]. Available: https://www.android.com/intl/es_es/what-is-android/. [Último acceso: 03 2022].

[9] «Kernel Linux,» [En línea]. Available: https://es.wikipedia.org/wiki/N%C3%BAcleo_Linux. [Último acceso: 05 2022].

[10] «Android 5.0,» [En línea]. Available: https://www.android.com/intl/es_es/versions/lollipop-5-0/. [Último acceso: 03 2022].

[11] «OpenGl,» [En línea]. Available: https://www.opengl.org/. [Último acceso: 03 2022].

[12] . . Joshelu, «Fing: Tu escáner de red para Android,» , 2013. [En línea]. Available: http://xatakaon.com/seguridad-en-redes/fing-tu-escaner-de-red-para-android. [Último acceso: 18 5 2022].

[13] «android,» [En línea]. Available: https://www.adslzone.net/reportajes/software/que-es-android/.

9 Anexo

```java
public class MusicBean {
    private String id;
    private String song;
    private String singer;
    private String album;
    private String duration;
    private String path;
    public MusicBean() { }
    public MusicBean(String id, String song, String singer, String album,
                     String duration, String path) {
        this.id = id;
        this.song = song;
        this.singer = singer;
        this.album = album;
        this.duration = duration;
        this.path = path; }
    public String getId() { return id; }
    public void setId(String id) { this.id = id; }
    public String getSong() { return song; }
    public void setSong(String song) { this.song = song; }
    public String getSinger() { return singer; }
    public void setSinger(String singer) { this.singer = singer; }
    public String getAlbum() { return album; }
    public void setAlbum(String album) { this.album = album; }
    public String getDuration() { return duration; }
    public void setDuration(String duration) { this.duration = duration; }
```

Ilustración 64 Codigos de la clase MusicBean

```
<Button
    android:id="@+id/bottom_iv_next"
    android:layout_width="60dp"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_marginTop="3dp"
    android:layout_marginRight="40dp"
    android:background="@drawable/icon_next"
    android:scaleX="0.5"
    android:scaleY="0.5" />
<Button
    android:id="@+id/bottom_iv_play"
    android:layout_width="60dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="3dp"
    android:layout_marginRight="15dp"
    android:layout_toLeftOf="@id/bottom_iv_next"
    android:background="@drawable/icon_play"
    android:scaleX="0.5"
    android:scaleY="0.5" />
<Button
    android:id="@+id/bottom_iv_last"
    android:layout_width="60dp"
    android:layout_height="wrap_content"
    android:layout_marginTop="3dp"
    android:layout_marginRight="15dp"
```

Ilustración 65 Codigos para visualización de los botones play, last y next

Este documento esta firmado por

Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
Fecha/Hora	Sun May 29 20:48:50 CEST 2022
Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
Numero de Serie	561
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)