

PRC6_AlfonsoRincon

Alfonso Rincón Cuerva
2º DAM

Programación Multimedia y de Dispositivos Móviles

ÍNDICE

HERRAMIENTAS Y TECNOLOGÍAS.....	1
VERSIONES	1
EMULADORES.....	1
PLATAFORMA	2
LENGUAJES.....	2
PROBLEMA PLANTEADO	2
SOLUCIÓN AL PROBLEMA.....	2
FUNCIONES UTILIZADAS.....	2
PRUEBAS REALIZADAS	4
ESTRUCTURA DE LA APLICACIÓN.....	5
EJECUCIÓN DE LA APLICACIÓN	6
COSAS QUE HE APRENDIDO	6
BIBLIOGRAFÍA.....	7

HERRAMIENTAS Y TECNOLOGÍAS

VERSIONES

En el archivo **libs.versions.toml** encontramos versiones de diferentes bibliotecas y plugins del proyecto.

- **agp:** 8.5.2.
- **activity:** 1.10.30

```
1 [versions]
2 agp = "8.5.2"
3 junit = "4.13.2"
4 junitVersion = "1.2.1"
5 espressoCore = "3.6.1"
6 appcompat = "1.7.0"
7 material = "1.12.0"
8 activity = "1.10.0"
9 constraintlayout = "2.2.0"
```

La versión de Gradle es de 8.7.

```
1 #Sun Dec 22 11:50:22 CET 2024
2 distributionBase=GRADLE_USER_HOME
3 distributionPath=wrapper/dists
4 distributionUrl=https\://services.gradle.org/distributions/gradle-8.7-bin.zip
5 zipStoreBase=GRADLE_USER_HOME
6 zipStorePath=wrapper/dists
7
```

En el archivo **build.gradle.kts**, podremos ver lo siguiente:

- **minSdk:** en esta aplicación es de 23. Esto significa que la versión de API mínima que soporta la app es la 23.
- **targetSdk:** es la versión de SDK para la cual está optimizada la app. En esta app, es de 34.

```
defaultConfig {
    applicationId = "edu.pruebas.prcó_alfonsorincon"
    minSdk = 30
    targetSdk = 34
    versionCode = 1
    versionName = "1.0"
```

EMULADORES

- **Pixel 6 API 32:** 6,0" 1080x2340 440dpi. **System image:** R (API 30)
- **Pixel 6 API 34:** 6,4" 1080x2400 420dpi **System image:** R (API 30)

PLATAFORMA

El programa fue creado en la plataforma **Android Studio**, que es un entorno de desarrollo integrado oficial para crear aplicaciones Android. Incluye características como editor de código, diseñador de interfaces, emulador de Android y sistema de compilación **Gradle**.

LENGUAJES

Los lenguajes utilizados para la creación de esta aplicación son los siguientes:

- **Java**: para la lógica de la aplicación e interactuar con los usuarios.
- **XML**: para agregar características y estilos a la aplicación.
- **Gradle**: es el sistema de compilación que utiliza Android.

PROBLEMA PLANTEADO

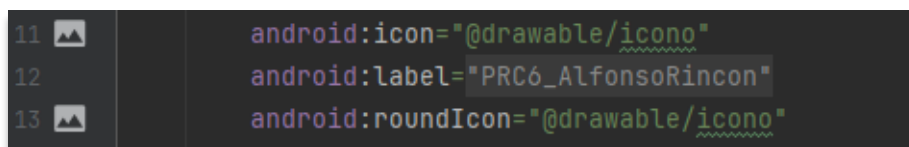
Se nos pide crear una aplicación que muestre audios y videos de diferentes formas. Podremos reproducir audios locales, los cuales nos permitirán pausar, avanzar o retrasar, al igual que podremos reproducir videos tanto locales como en streaming. El contenido de cada uno de ellos se nos provee en un JSON.

SOLUCIÓN AL PROBLEMA

Esta aplicación se ha creado mediante la utilización de un RecyclerView y el JSON. A través de este, se meten los elementos en el Recycler, cada uno con sus datos y acciones en cuestión. Los videos se reproducen en otra ventana, mientras que los audios se reproducen en la misma. Ambos tendrán un MediaController.

FUNCIONES UTILIZADAS

ICONO: en el archivo AndroidManifest.xml, asignaré un nuevo icono para la aplicación.



```
11 <android:icon="@drawable/icono"
12 <android:label="PRC6_AlfonsoRincon"
13 <android:roundIcon="@drawable/icono"/>
```

CANCION: esta clase contiene los atributos serializables, constructores, getters y setters para cada elemento del JSON.

```

3 import com.google.gson.annotations.SerializedName;
4
5 public class Cancion { 11 usages  ▲ alfonso
6     @SerializedName("nombre") 3 usages
7     private String autor;
8
9     @SerializedName("descripcion") 3 usages
10    private String nombre;
11
12    private String tipo; 3 usages
13
14    @SerializedName("URI") 3 usages
15    private String URI;
16
17    private String imagen; 3 usages
18
19    // Constructores
20    public Cancion(String autor, String nombre, String tipo, String URI, String imagen) {
21        this.autor = autor;
22        this.nombre = nombre;
23        this.tipo = tipo;
24        this.URI = URI;
25        this.imagen = imagen;
26    }
27    public Cancion() { no usages  ▲ alfonso
28    }

```

CANCIONADAPTER: es el Adapter para poder meter las canciones en el RecyclerView.

VIDEOACTIVITY: es la actividad en la cual se reproduce el video, ya sea local, o de streaming. Este tendrá un MediaController para poder controlar el progreso del video, y un VideoView que es donde se pondrá el contenido.

```

} else if(tipo ==2 ) {
    if(nombreVideo != null && !nombreVideo.isEmpty()) {
        Uri videoUri=Uri.parse(nombreVideo);
        videoView.setVideoURI(videoUri);
        // Caso de error
        videoView.setOnErrorListener(new MediaPlayer.OnErrorListener() { ▲ alfonso
            @Override ▲ alfonso
            public boolean onError(MediaPlayer mp, int what, int extra) {
                Toast.makeText( context: VideoActivity.this, text: "Video no disponible", Toast.LENGTH
                return true;
            }
        });
        videoView.start();
    }
}

```

MAINACTIVITY: aquí se pondrá el Recycler con todos los elementos del JSON, y según el **tipo** que te este tendrá, se hará una cosa u otra. Los 0 reproducirán una canción con un **MediaPlayer**, los 1 abrirán la VideoActivity pasando el archivo como parámetro, y los 2 recibirán el enlace como parámetro y reproducirá el video en cuestión.

```
recyclerCanciones=findViewById(R.id.recyclerCanciones);
recyclerCanciones.setLayoutManager(new LinearLayoutManager( context: this));

checkboxesSeleccionados = new boolean[]{false, false, false};

String jsonString=leerJSON( archivo: "recursosList.json");
// Obtener los datos del JSON
if(jsonString!=null){
    Gson gson=new Gson();
    RecursosWrapper wrapper=gson.fromJson(jsonString, RecursosWrapper.class);
    listaCanciones=wrapper.getListaRecursos();

    listaCancionesSinFiltros=new ArrayList<>(listaCanciones);

    adapter=new CancionAdapter( context: this, listaCanciones);
    recyclerCanciones.setAdapter(adapter);
}else{
    listaCanciones=new ArrayList<>();
    listaCancionesSinFiltros=new ArrayList<>();
}
```

PRUEBAS REALIZADAS

Se realizan diversas pruebas de caja negra para verificar la funcionalidad del programa. Tras ello, se programan sus respectivas soluciones:

1. DISTINTAS RESOLUCIONES:

Se ha probado la aplicación en pantallas con diferentes resoluciones, además de en dispositivos móviles reales. En todos funciona correctamente.

2. CICLO DE VIDA

Se han hecho varias pruebas (cerrar app, abrirla, cambiar de actividad...) para comprobar el correcto funcionamiento del ciclo de vida.

ESTRUCTURA DE LA APLICACIÓN

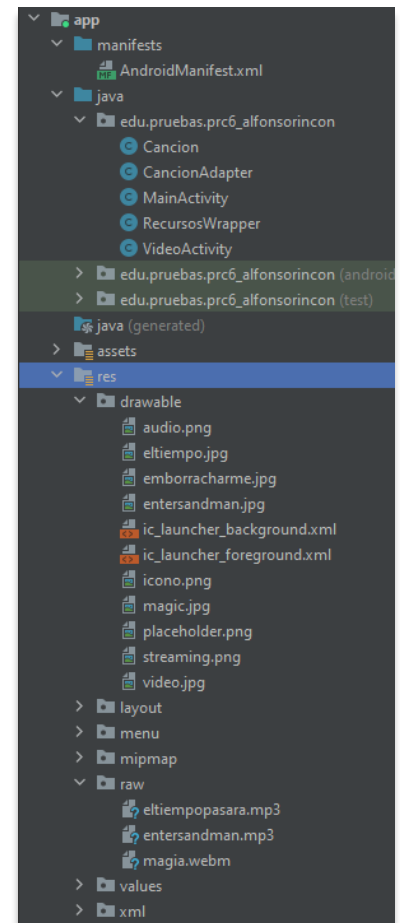
drawable: aquí están las imágenes de las portadas de los álbumes y de los tipos de multimedia.

menu: aquí está el ToolBar.

values: en sus archivos **colors.xml** y **string.xml** se encontrarán todas las variables de colores y strings que hemos creado.

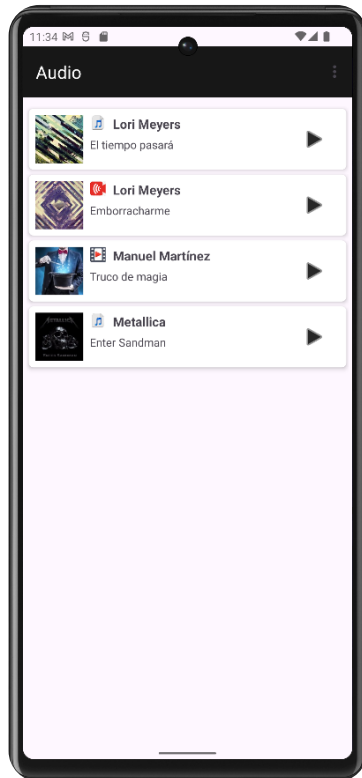
raw: contiene los archivos multimedia de audio y video.

assets: contiene el JSON con los datos de las canciones y videos.

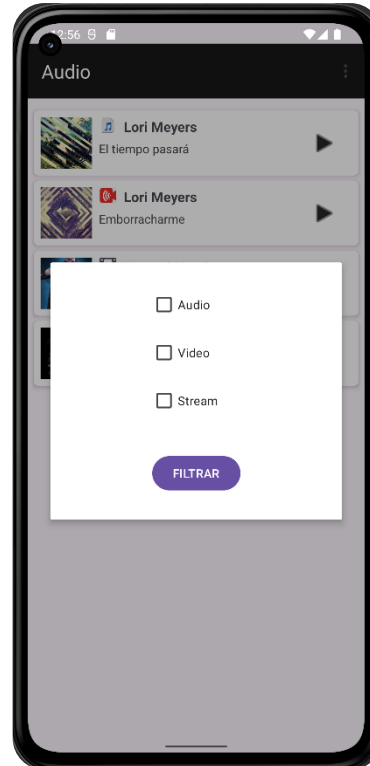


EJECUCIÓN DE LA APLICACIÓN

INICIO



FILTROS



COSAS QUE HE APRENDIDO

1. Usar un MediaController
2. Reproducir videos en un VideoView

BIBLIOGRAFÍA

With Sam. (2020, 20 mayo). How to create Video View with Media Controller on any Layout in Android Studio Latest Version [Vídeo]. YouTube. <https://www.youtube.com/watch?v=38vGzUg9pHc>