

DIVISIÓN DE INGENIERÍA EN SISTEMAS COMPUTACIONALES



**MANUAL DE PRÁCTICAS DE LA
ASIGNATURA TÓPICOS AVANZADOS
DE PROGRAMACIÓN.**

 INGENIERÍA EN SISTEMAS COMPUTACIONALES PRÁCTICA No. 1	
---	---

DATOS GENERALES	
ASIGNATURA: Inteligencia Artificial.	
TÍTULO DE LA PRÁCTICA: Práctica 1: El consumo del API REST Movies con Retrofit y MVVM URL	
DOCENTE: M. EN CC. MARTHA G. MORALES HUERTA	
ESTUDIANTE(S): BAÑUELOS GARCIA ROBERTO ANGEL	FECHA: 15/12/23

OBJETIVO DE LA PRÁCTICA: Realizar el consumo del API REST Movies con Retrofit y MVVM URL	
COMPETENCIA(S) ESPECÍFICA(S):	COMPETENCIA(S) GENÉRICA(S)(8)
<input type="checkbox"/> Resolver problemas en base a técnicas de búsqueda en espacio de estado.	<ul style="list-style-type: none"> ● Capacidad de análisis y síntesis. ● <input type="checkbox"/> Capacidad de organizar y planificar. ● <input type="checkbox"/> Habilidad para buscar y analizar información proveniente de fuentes diversas. ● <input type="checkbox"/> Solución de problemas. ● <input type="checkbox"/> Toma de decisiones. ● <input type="checkbox"/> Trabajo en equipo. ● <input type="checkbox"/> Capacidad de aplicar los conocimientos. ● <input type="checkbox"/> Habilidades de investigación. ● <input type="checkbox"/> Capacidad de generar nuevas ideas. ● <input type="checkbox"/> Liderazgo. ● <input type="checkbox"/> Habilidad para trabajar en forma. ● Autónoma. ● <input type="checkbox"/> Búsqueda del logro.

REQUERIMIENTOS	
RECURSOS MATERIALES	RECURSOS TÉCNICOS/TECNOLÓGICOS
<ul style="list-style-type: none"> Computadora/laptop Internet 	<ul style="list-style-type: none"> Software Android Studio <ul style="list-style-type: none"> Kotlin Cuenta en The Movie DataBase

MARCO TEÓRICO

¿Qué es Android Studio?

Android Studio es el entorno de desarrollo integrado (IDE) oficial que se usa en el desarrollo de apps para Android. Basado en el potente editor de código y las herramientas para desarrolladores de IntelliJ IDEA, Android Studio ofrece aún más funciones que mejoran tu productividad cuando compiles apps para Android, como las siguientes:

Un sistema de compilación flexible basado en Gradle

Un emulador rápido y cargado de funciones

Un entorno unificado donde puedes desarrollar para todos los dispositivos Android

Ediciones en vivo para actualizar elementos componibles en emuladores y dispositivos físicos, en tiempo real

Integración con GitHub y plantillas de código para ayudarte a compilar funciones de apps comunes y también importar código de muestra

Variedad de marcos de trabajo y herramientas de prueba

Herramientas de Lint para identificar problemas de rendimiento, usabilidad y compatibilidad de versiones, entre otros

Compatibilidad con C++ y NDK

Compatibilidad integrada con Google Cloud Platform, que facilita la integración con Google Cloud Messaging y App Engine

En esta página, encontrarás una introducción a las funciones básicas de Android Studio.

Para acceder a un resumen de los cambios más recientes, consulta las notas de la versión de Android Studio.

3

¿Qué es Material Dising 3?

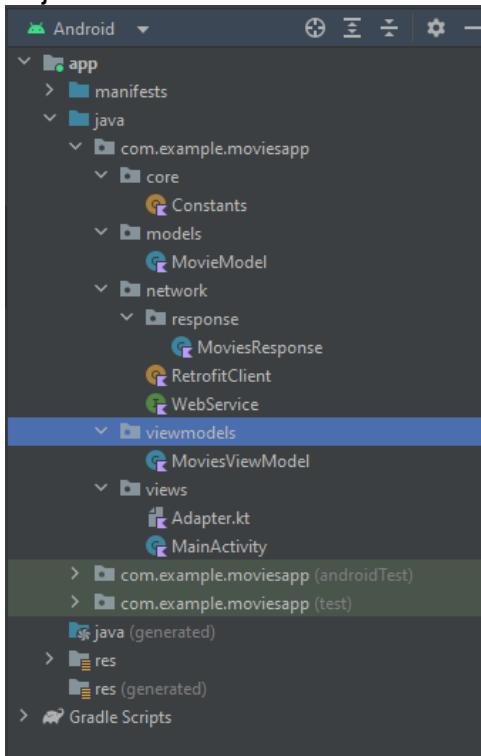
Material Design 3 es la siguiente evolución de Material Design. Incluye temas y componentes actualizados, y funciones de personalización de Material You, como el color dinámico. Es una actualización de Material Design 2 y es coherente con el nuevo estilo visual y la IU del sistema de Android 12 y versiones posteriores.

Retrofit

Retrofit simplifica el proceso de realizar solicitudes HTTP al proporcionar una interfaz fácil de usar y gestionar la conversión de datos JSON a objetos Java. Esto facilita la integración de servicios web en aplicaciones Android y mejora la legibilidad del código.

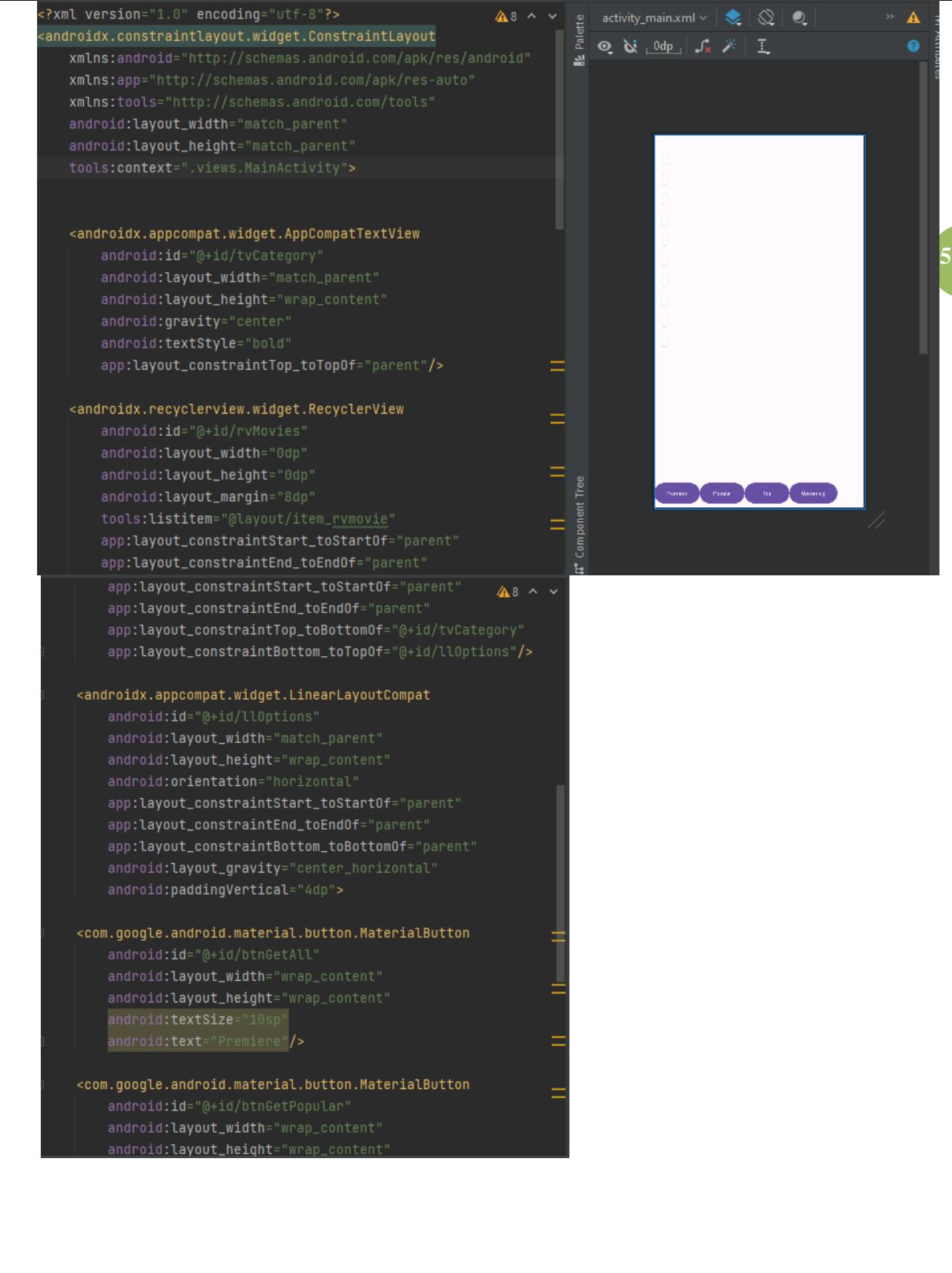
DESARROLLO

se han empleado diversas herramientas que han sido fundamentales para su implementación. A continuación, se presenta una descripción detallada de estas herramientas, acompañada de sus respectivos archivos, como se observa en la imagen adjunta.



4

Para poder dar inicio de esta práctica que se llevo acabo durante las clases , usaremos como herramienta principal el IDE de Android Studio y después comenzamos a crear nuestro proyecto que será en Kotlin, el cual contara con cuatro botones(Premiere, Popular,Top y upcoming) los cuales muestra los diferentes tipos de categorías que hay.



The screenshot shows the Android Studio interface with the XML code for the activity_main.xml file on the left and the layout preview on the right.

```

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".views.MainActivity">

    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/tvCategory"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textStyle="bold"
        app:layout_constraintTop_toTopOf="parent"/>

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rvMovies"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_margin="8dp"
        tools:listitem="@layout/item_rvmovie"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tvCategory"
        app:layout_constraintBottom_toTopOf="@+id/lloptions"/>

    <androidx.appcompat.widget.LinearLayoutCompat
        android:id="@+id/lloptions"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        android:layout_gravity="center_horizontal"
        android:paddingVertical="4dp">

        <com.google.android.material.button.MaterialButton
            android:id="@+id/btnGetAll"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textSize="10sp"
            android:text="Premiere"/>

        <com.google.android.material.button.MaterialButton
            android:id="@+id/btnGetPopular"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
    
```

The layout preview shows a white container with a blue border. At the bottom, there is a navigation bar with four purple buttons labeled "Peliculas", "Personajes", "Tops", and "Banners".

```
        android:textSize="10sp"
        android:text="Popular"/>

    <com.google.android.material.button.MaterialButton
        android:id="@+id/btnGetTopRated"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="10sp"
        android:text="Top"/>

    <com.google.android.material.button.MaterialButton
        android:id="@+id/btnGetUpcoming"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="10sp"
        android:text="Upcoming"/>
</androidx.appcompat.widget.LinearLayoutCompat>
</androidx.constraintlayout.widget.ConstraintLayout>
```

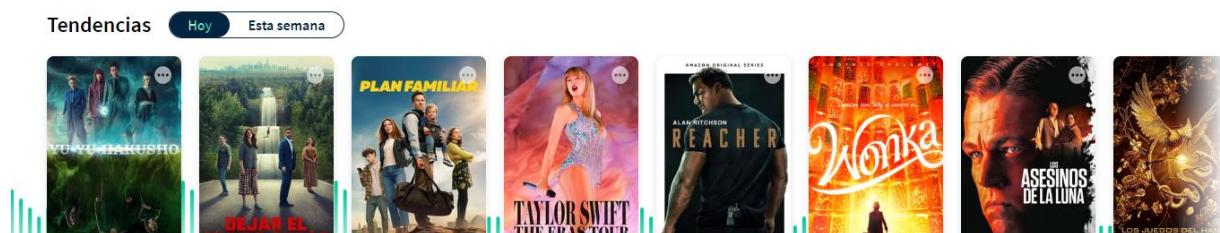
6

Cuando se hace clic en uno de los botones de categoría, filtra la lista de películas según la categoría seleccionada y actualiza el RecyclerView.

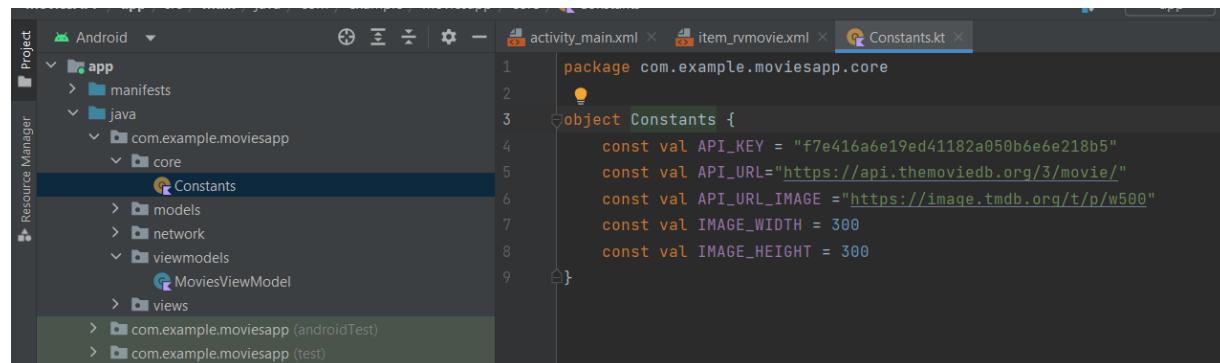
- ConstraintLayout: Este es el contenedor principal que se encarga de organizar todos los elementos secundarios según las restricciones que se le asignen.
- CardView: Es un contenedor que proporciona un efecto de "tarjeta" para los elementos, es decir, una especie de contorno que separa visualmente cada elemento de la lista.
- LinearLayoutCompat: Es un contenedor que permite organizar sus elementos secundarios en una sola línea, ya sea de manera horizontal o vertical. En este caso, su orientación es vertical, lo que significa que sus elementos hijos se organizarán uno debajo del otro.
- AppCompatImageView: Es un elemento de la interfaz de usuario que se utiliza para mostrar imágenes. En este caso, se utiliza para mostrar la imagen de la película.
- TextView: Es un elemento de la interfaz de usuario que se utiliza para mostrar texto. En este caso, se utiliza para mostrar información adicional sobre la película, como el año de lanzamiento, los actores, etc.



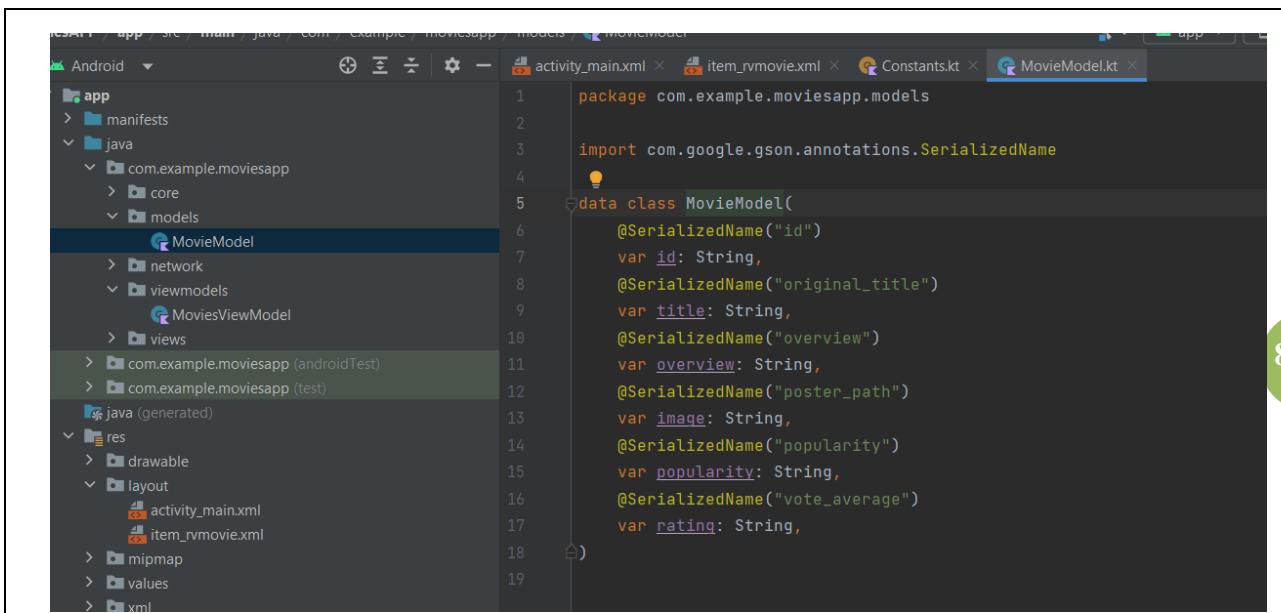
7



The Movie Database (TMDB) es una plataforma en línea que ofrece una amplia base de datos de películas, programas de televisión y contenido relacionado con el entretenimiento. Es una fuente de información detallada sobre películas, programas de televisión, actores, directores, equipos de producción y mucho más.



En core colocaremos la url de la api que usamos que en este caso fue TMDB

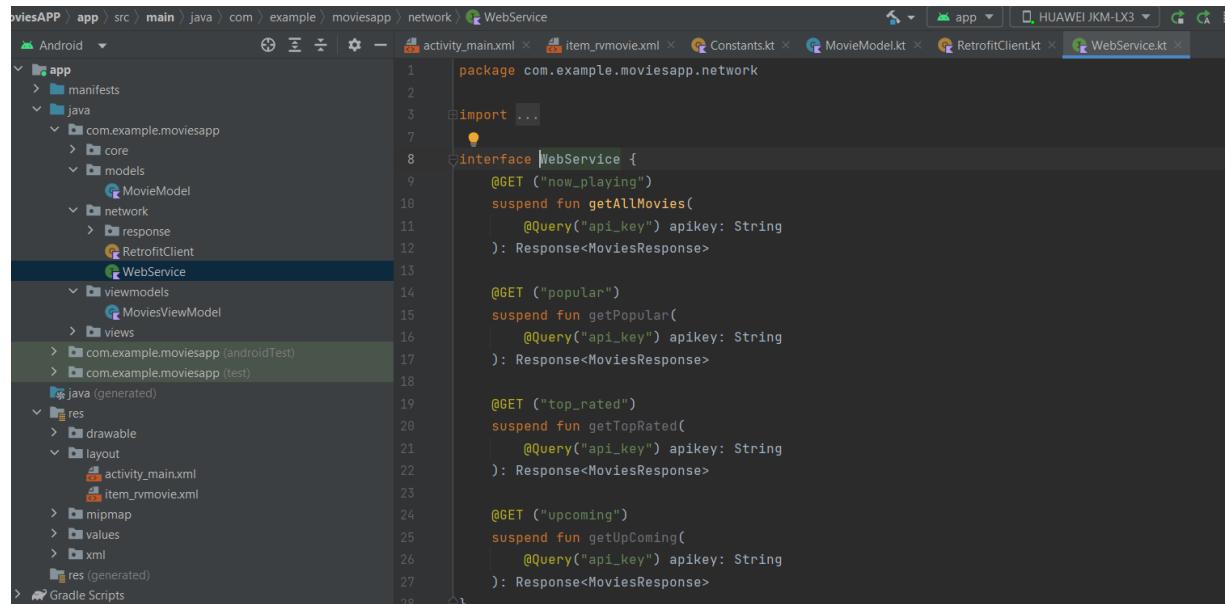


```

1 package com.example.moviesapp.models
2
3 import com.google.gson.annotations.SerializedName
4
5 data class MovieModel(
6     @SerializedName("id")
7     var id: String,
8     @SerializedName("original_title")
9     var title: String,
10    @SerializedName("overview")
11    var overview: String,
12    @SerializedName("poster_path")
13    var image: String,
14    @SerializedName("popularity")
15    var popularity: String,
16    @SerializedName("vote_average")
17    var rating: String,
18 )

```

Cada campo de la clase está anotado con `@SerializedName("nombre_del_campo_en_JSON")`, lo que especifica cómo se mapearán los campos de la clase Kotlin a los nombres correspondientes en un objeto JSON. Esto es útil al convertir datos JSON en objetos Kotlin y viceversa usando la biblioteca GSON.



```

1 package com.example.moviesapp.network
2
3 import ...
4
5 interface WebService {
6     @GET ("now_playing")
7     suspend fun getAllMovies(
8         @Query("api_key") apikey: String
9     ): Response<MoviesResponse>
10
11     @GET ("popular")
12     suspend fun getPopular(
13         @Query("api_key") apikey: String
14     ): Response<MoviesResponse>
15
16     @GET ("top_rated")
17     suspend fun getTopRated(
18         @Query("api_key") apikey: String
19     ): Response<MoviesResponse>
20
21     @GET ("upcoming")
22     suspend fun getUpComing(
23         @Query("api_key") apikey: String
24     ): Response<MoviesResponse>
25
26 }

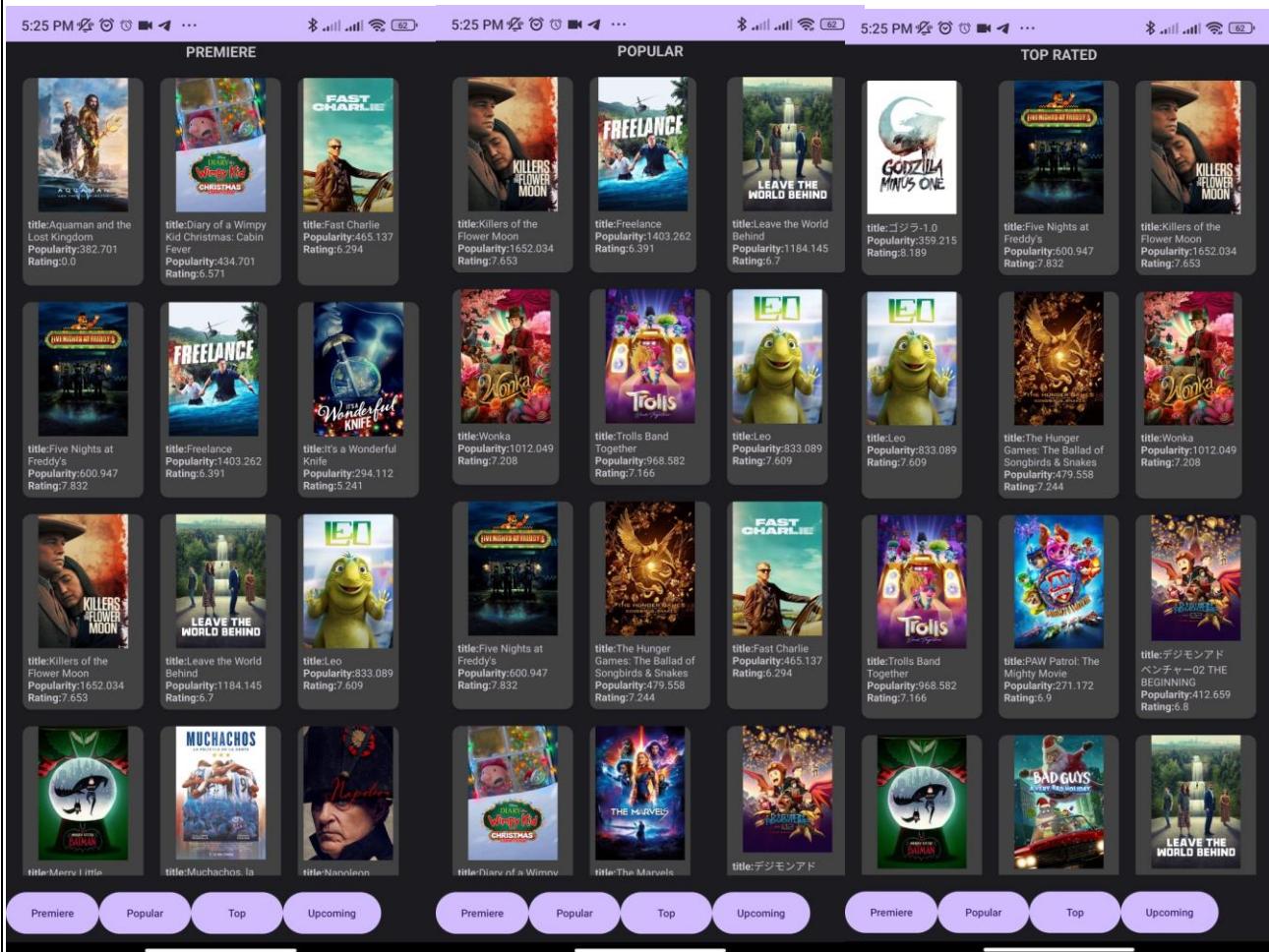
```

Esta interfaz actúa como una puerta de entrada a un servicio en línea que ofrece información sobre películas. Cada función dentro de esta interfaz representa un tipo diferente de solicitud que se puede hacer al servicio:

- `getAllMovies()`: Obtiene todas las películas que están actualmente en los cines.
- `getPopular()`: Recupera las películas más populares.
- `getTopRated()`: Trae las películas mejor valoradas.

- `getUpComing()`: Obtiene información sobre las películas próximas a estrenarse.

RESULTADOS:



CONCLUSIONES

Como conclusión la practica me costo un poco de trabajo y realmente pensé que no me iba a salir, pero se logro con éxito gracias a esta practica ya tengo un poco mas de conocimiento sobre el consumo de APIs.

FUENTE(S) DE INFORMACIÓN:

Bibliografía

- *Introducción a Android Studio*. (n.d.). Android Developers. <https://developer.android.com/studio/intro?hl=es-419>
- *The Movie Database (TMDB)*. (s. f.). The Movie Database. <https://www.themoviedb.org/>

- Ramos, J. (s. f.). *Cómo consumir una API y procesar la respuesta*  Con Retrofit.

Programación y más. <https://programacionymas.com/blog/consumir-una-api-usando-retrofit>

10

NOMBRE Y FIRMA DEL DOCENTE	EVALUACIÓN
MCC. Martha G. Morales Huerta	