Computación Ubicua, Móvil y en la Nube

(MyAnimeList)

2021-2022

Santiago Bauzá Hirschler

Matricula: bo0476

índice

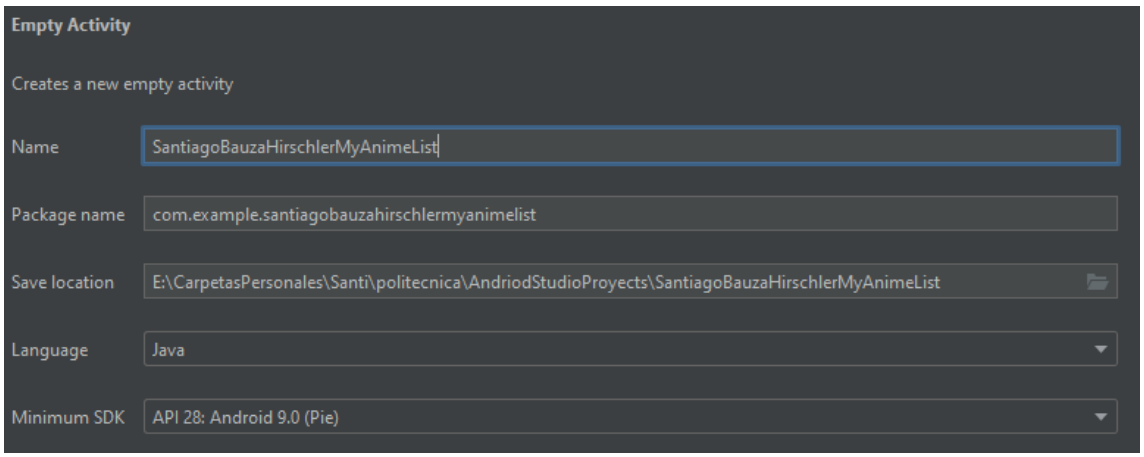
1. Introducción

El proyecto se enfoca en mostrar una lista de animes y tiene como objetivos el acceso a una API que devuelva datos en formato .json y visualizarlos en RecyclerView. También requiere unos mínimos indispensables como el uso de Retrofit2 + RxJava2 + Gson, RecyclerView con View-Holder y acceso a dos end points. El entorno del proyecto se va a desarrollar con Android Studio.

Por consiguiente, este documento aporta los pasos realizados para la resolución de estos objetivos con respectivas capturas de los datos recabados.

2. Entorno de trabajo

El proyecto se desarrollará en Android Studio y el dispositivo que se utilizará será Pixel 4 con Android 11.



Una vez creado el proyecto añadimos los permisos necesarios en Android manifest por ejemplo el permiso de internet y añadimos todas las librerías necesarias que ya sabemos de antemano que vamos a utilizar. En base a los objetivos descritos en la descripción.

AndroidManifest.xml

```
<uses-permission android:name="android.permission.INTERNET"/>
```

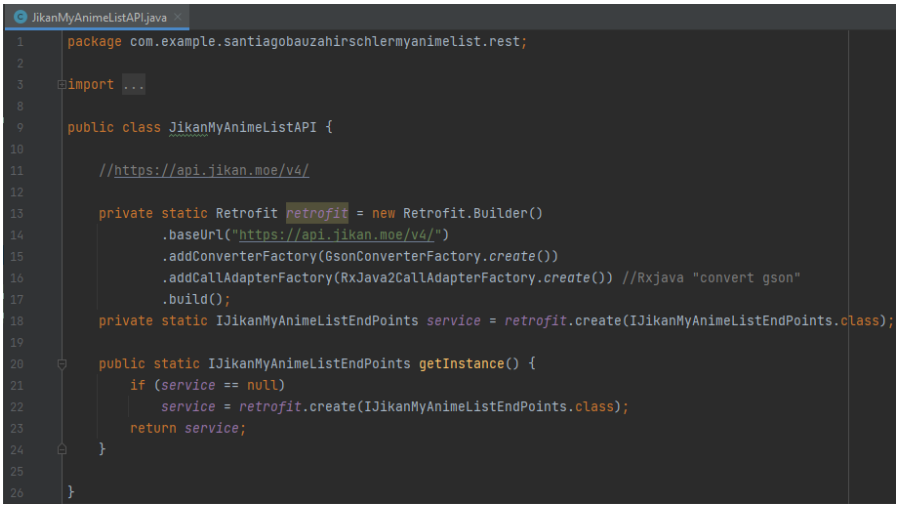
build.gradle (Module: SantiagoBauzaHirschlerFormula1.app)

```
implementation 'com.google.code.gson:gson:2.9.0'
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'

implementation 'com.squareup.retrofit2:adapter-rxjava2:2.3.0'
implementation 'io.reactivex.rxjava2:rxjava:2.2.2'
implementation 'io.reactivex.rxjava2:rxandroid:2.1.0'

implementation 'com.squareup.picasso:picasso:2.8'
```

3. Red

Para poder hacer peticiones en la red debemos implementar el servicio de Retrofit, que hace sencillo conectar a un servicio web REST traduciendo la API. Como dependencia para trabajar con los datos .json necesitaremos un conversor GSON, nos ayudará a traducir los datos de su correspondiente cuerpo. Por otra parte, utilizaremos RxJava que servirá para procesar flujos de datos síncronos y asíncronos, que no restringe tipos de datos tradicionales. RxJava adapta de por sí la conversión de los datos GSON.

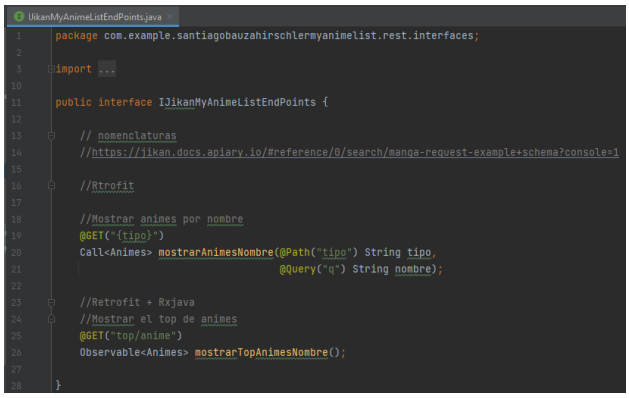
Creamos una clase de servicio red con los requisitos mencionados.



Como podemos observar en la imagen anterior tal y como se muestra en la base del URL debemos colocar la dirección web. A donde haremos nuestras respectivas peticiones.

4. Peticiones

Las peticiones que vamos a realizar son dos peticiones. En general lo que vamos a mostrar es una lista de animes, la primera petición mostrará los animes a partir de un nombre por parámetro y la segunda petición mostrará el top de animes. Quiero recalcar que el número de animes por página es un máximo de 25 y como mero prototipo trabajaremos sobre una sola página. A continuación, dejaré una imagen de la interfaz de los endpoints:

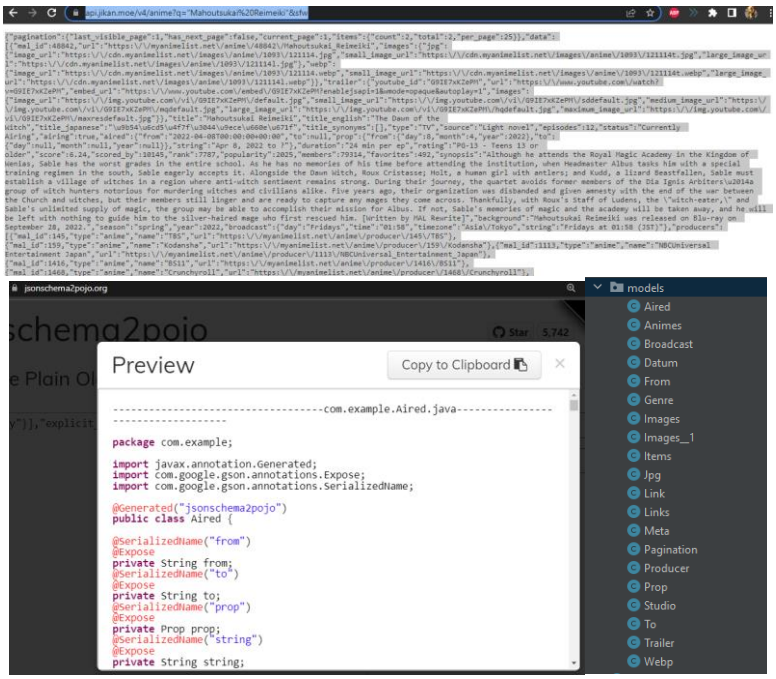


Como podemos observar en la imagen anterior se ha usado para representar dos peticiones una con Retrofit a secas y otra con Retrofit + RxJava que se diferencian por el método Call y Observable respectivamente.

4.1 Modelos

Para poder tratar los datos necesitamos analizarlos con el fin de poder crear sus correspondientes clases modelo que ayudarán en el recorrido de los datos que queramos obtener.

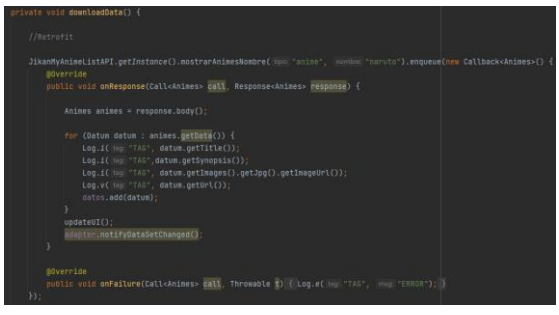
Se utilizó jsonschema2pojo para crear las clases modelos a partir del json que la API proporciona para poder tratar los datos y extraer la información necesaria.



4.2 Descargar datos

Para descargar los datos realizaremos una llamada al servicio al cual necesitaremos pasar la continuación del enlace al cual nos queremos referir para que nos devuelva los datos con los cuales vamos a trabajar respectivamente.

Dentro del método de descarga de datos se realizó de 2 maneras una con Retrofit y otra con Retrofit + RxJava. A continuación, dejaré unas imágenes.



Para comprobar su funcionamiento se realizó un registro selectivo para mostrar por consola algunos datos obtenidos.

5. RecyclerView con View-Holder

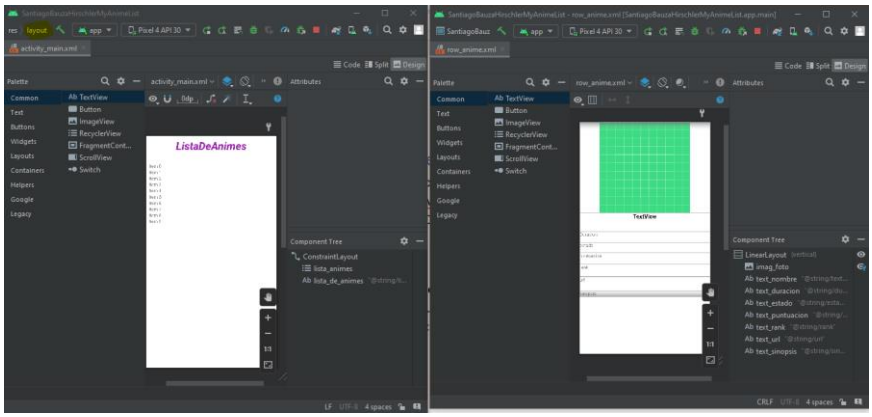
Para representar los datos en el dispositivo vamos a utilizar RecyclerView para mostrar los datos de una manera sencilla y el ViewHolder describirá una vista del elemento y los metadatos dentro de RecyclerView. Cómo vamos a utilizar una lista dinámica esto nos ayudará a adaptar cada objeto de la lista.

```
private AdapterAnimes adapter = new AdapterAnimes(datos);
```

```
RecyclerView lista = findViewById(R.id.lista_animes);
lista.setLayoutManager(new LinearLayoutManager( context: this));
lista.setAdapter(adapter);
```

5.1 Layout

La forma dentro del diseño y los elementos distribuidos están definidos dentro del layout, donde cada elemento añadido tiene un Id, al cual se va a hacer referencia a partir de unos métodos para representar los datos correspondientes.



5.2 AdapterAnimes

La clase AdapterAnimes será la encargada de gestionar los ViewHolders. Para ello la clase recibirá los datos para posteriormente tratarlos y enviar cada ítem a su vista correspondiente para mostrarlo.

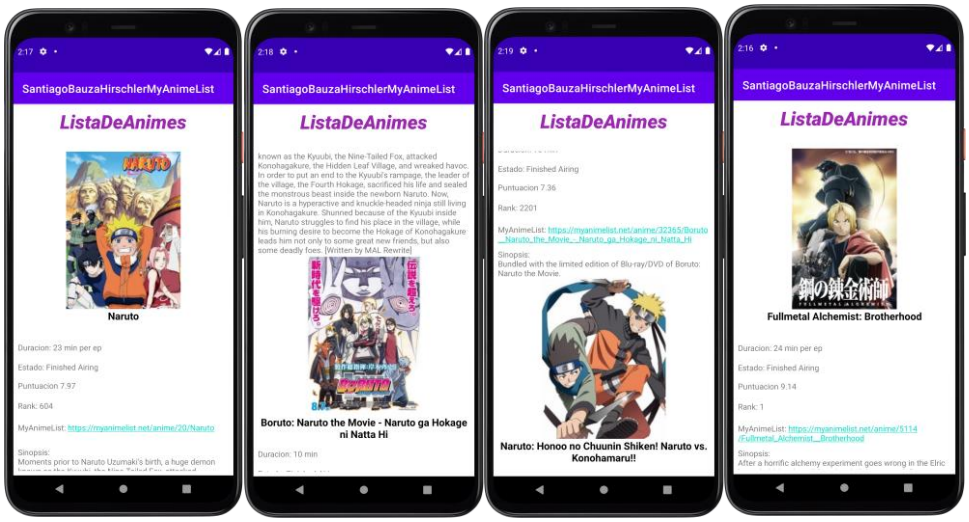
```
public class AdapterAnimes extends RecyclerView.Adapter<AdapterAnimes.ViewHolder> {

    private List<Datum> datos;

    public AdapterAnimes(List<Datum> datos) { this.datos = datos; }
```

```
@Override
public void onBindViewHolder(@NonNull AdapterAnimes.ViewHolder holder, int position) {
    holder.nombre.setText(datos.get(position).getTitle());
    holder.url.setText("MyAnimeList: "+datos.get(position).getUrl());
    holder.duracion.setText("Duracion: "+datos.get(position).getDuration());
```

```
public static class ViewHolder extends RecyclerView.ViewHolder {
    private TextView nombre,duracion,estado,puntuacion,rank,sinops
    private ImageView foto;

    public ViewHolder(@NonNull View itemView) {
        super(itemView);

        nombre = itemView.findViewById(R.id.text_nombre);
        foto = itemView.findViewById(R.id.imag_foto);
        duracion = itemView.findViewById(R.id.text_duracion);
```

6.Ejecucion visual

Podremos observar una lista de animes donde detallan la duración por capítulo, el estado de emisión, la puntuación, el ranking, el enlace informativo de MyAnimeList y la sinopsis.



7.Conclusion

El proyecto resultó ser bastante creativo pude aprender cómo tratar datos desde la red y poder representarlos de una manera fácil y sencilla en un dispositivo. También gracias a esto hemos adquirido unos conocimientos prácticos de cómo tratar y representar la información mediante una herramienta cómo es Android Studio para su utilización en trabajos futuros.

Como podemos ver en el punto 6 tras aplicar lo aprendido, hemos conseguido el resultado deseado.

8.Bibliografía

Anime

Material de clase

Implementaciones repositorio Maven

Jsonschema2pojo

Nomenclaturas

Stackoverflow

Endponints representacion

JikanAPI

JikanDocumentacion