



Instituto Superior
Tecnológico del Azuay

Guía Práctica 4: Consumir API REST con retrofit

NOMBRE:

Juan Carlos Inga

CURSO:

M4B

DOCENTE:

Ing. Patricio Pacheco

MATERIA:

Desarrollo de Aplicaciones Móviles

FECHA DE ENTREGA:

jueves, 10 de marzo de 2022

Introducción

Esta guía práctica está centrada en cómo consumir una API o servicios web desde una aplicación Android. Si por ejemplo tienes una base de datos, pero no tienes una API creada. Entonces primero deberías definir una API. Una API es un intermediario entre una base de datos y una aplicación móvil.

Objetivo

Consumir una API (servicios web) utilizando el IDE de Desarrollo de Android Studio con la librería Retrofit y procesar la respuesta JSON obtenida.

Procedimiento

1. AÑADIR LA LIBRERÍA DE RETROFIT A NUESTRO PROYECTO.

Eso significa que debemos ir a nuestro archivo build.gradle y añadir las siguientes líneas:

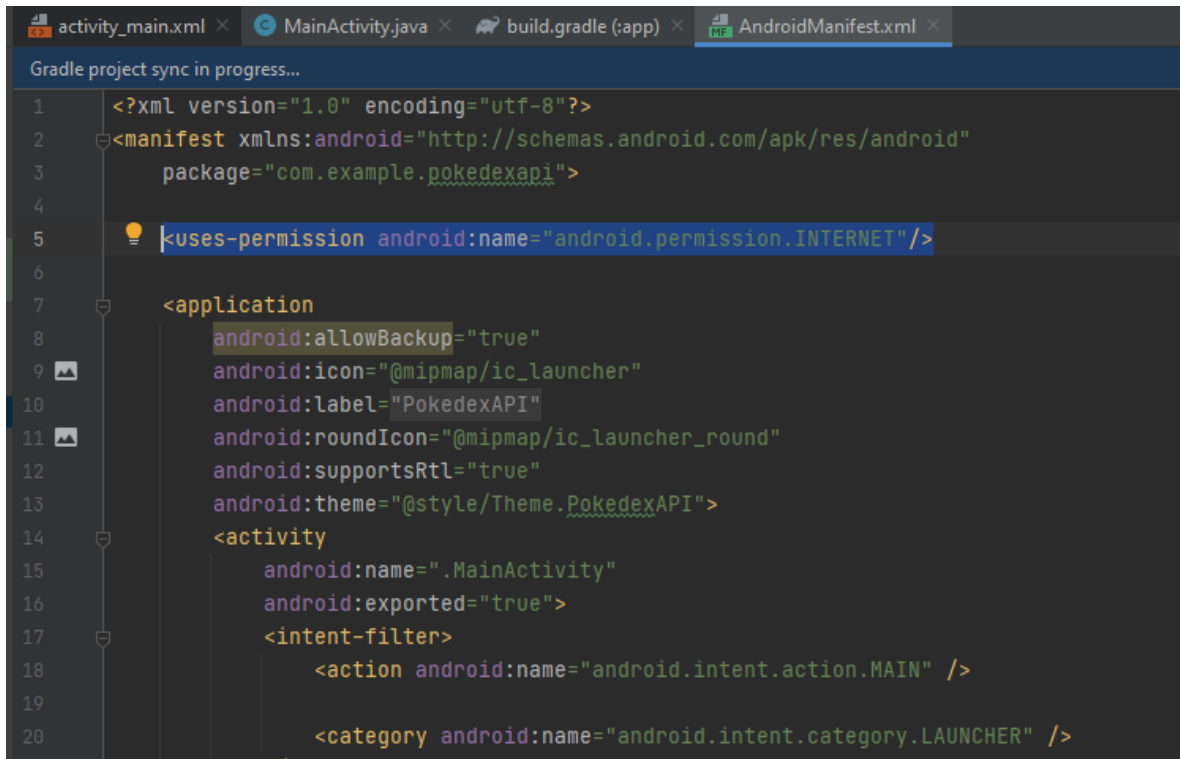
```
dependencies {  
  
    implementation 'androidx.appcompat:appcompat:1.4.1'  
    implementation 'com.google.android.material:material:1.5.0'  
    implementation 'androidx.constraintlayout:constraintlayout:2.1.3'  
  
    implementation 'com.squareup.retrofit2:retrofit:2.3.0'  
    implementation 'com.squareup.retrofit2:converter-gson:2.3.0'  
    implementation 'com.squareup.okhttp3:logging-interceptor:3.9.1'  
  
    testImplementation 'junit:junit:4.+'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'  
}
```

JSON es un formato de respuesta que usan las API. Eso es lo que vamos a obtener y procesar. Pero GSON es una dependencia adicional, que funciona en conjunto con Retrofit para "convertir las respuestas JSON obtenidas en objetos Java". Retrofit los llama "converters", y existen varios de ellos. Incluso para "mapear" respuestas obtenidas en formato XML.

2. SOLICITAR PERMISOS

Antes de empezar a configurar Retrofit en nuestro proyecto, es importante que nuestra aplicación se pueda conectar a internet.

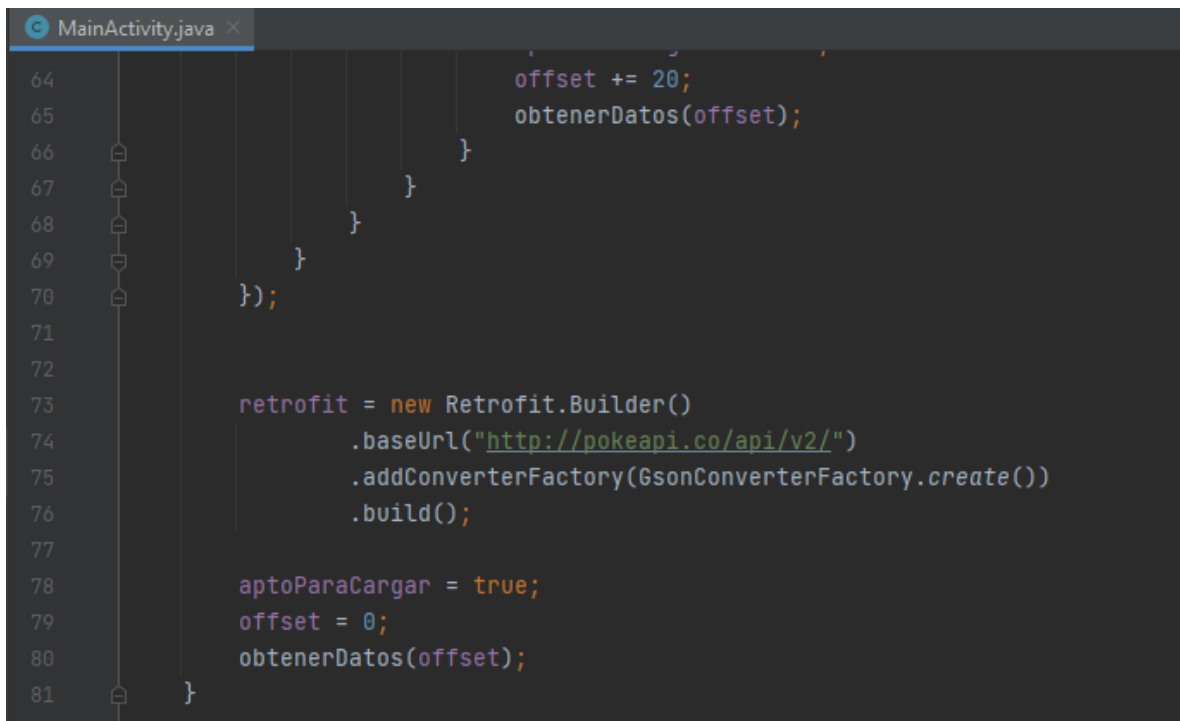
Para solicitar este permiso debemos añadir la siguiente línea a nuestro archivo manifest

A screenshot of an IDE showing the AndroidManifest.xml file. The file is being edited, and a lightbulb icon indicates a suggestion for the <uses-permission> tag. The code defines the application package, permissions, and the main activity.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.example.pokedexapi">
4
5     <uses-permission android:name="android.permission.INTERNET"/>
6
7     <application
8         android:allowBackup="true"
9         android:icon="@mipmap/ic_launcher"
10        android:label="PokedexAPI"
11        android:roundIcon="@mipmap/ic_launcher_round"
12        android:supportsRtl="true"
13        android:theme="@style/Theme.PokedexAPI">
14        <activity
15            android:name=".MainActivity"
16            android:exported="true">
17            <intent-filter>
18                <action android:name="android.intent.action.MAIN" />
19
20                <category android:name="android.intent.category.LAUNCHER" />
21            </intent-filter>
22        </activity>
23    </application>
24</manifest>
```

3. CREAR UNA CLASE Y UNA INTERFAZ

ActivityMain

A screenshot of an IDE showing the MainActivity.java file. The code defines the main activity, including a Retrofit client for fetching data from the PokeAPI.

```
64         offset += 20;
65         obtenerDatos(offset);
66     }
67 }
68
69 }
70 });
71
72
73 retrofit = new Retrofit.Builder()
74     .baseUrl("http://pokeapi.co/api/v2/")
75     .addConverterFactory(GsonConverterFactory.create())
76     .build();
77
78 aptoParaCargar = true;
79 offset = 0;
80 obtenerDatos(offset);
81 }
```

Esta es la url base que no cambia, en este caso he usado un api a cerca de un videojuegos y serie llamada Pokemon.

Creación de interface

```
MainActivity.java x PokeapiService.java x
1 package com.example.pokedexapi.pokeapi;
2
3 import com.example.pokedexapi.models.PokemonRespuesta;
4 import retrofit2.Call;
5 import retrofit2.http.GET;
6 import retrofit2.http.Query;
7
8 public interface PokeapiService {
9     @GET("pokemon")
10    Call<PokemonRespuesta> obtenerListaPokemon(@Query("limit") int limit, @Query("offset") int offset);
11
12 }
13
```

Creamos la interface, el servicio de donde se accede al servicio de los datos, se obtiene a lista y retorna los datos, respuestas.

Creación Clase Respuesta

```
MainActivity.java x PokeapiService.java x PokemonRespuesta.java x
1 package com.example.pokedexapi.models;
2
3 import java.util.ArrayList;
4
5 public class PokemonRespuesta {
6     private ArrayList<Pokemon> results;
7
8     public ArrayList<Pokemon> getResults() {
9         return results;
10    }
11
12    public void setResults(ArrayList<Pokemon> results) {
13        this.results = results;
14    }
15 }
```

Este es el array de donde se están los datos de información, es mejor usar array para que json cumpla su trabajo.

Clase Pokemon

```
Pokemon.java x
1 package com.example.pokedexapi.models;
2
3 public class Pokemon {
4
5     private int number;
6     private String name;
7     private String url;
8
9     public String getName() {
10         return name;
11     }
12
13     public void setName(String name) {
14         this.name = name;
15     }
16
17     public String getUrl() {
18         return url;
19     }
20
21     public void setUrl(String url) {
22         this.url = url;
23     }
24 }
```

Se crea la clase con sus atributos, en este caso estás designados 3 variables para trabajar, generamos getters and setters y están listos para usar.

ActivityMain.xml

```
Pokemon.java x activity_main.xml x
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:tools="http://schemas.android.com/tools"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent"
6     tools:context=".MainActivity">
7
8     <androidx.recyclerview.widget.RecyclerView
9         android:id="@+id/recyclerView"
10        android:layout_width="match_parent"
11        android:layout_height="match_parent" />
12
13 </androidx.constraintlayout.widget.ConstraintLayout>
```

En esta parte modificamos el diseño de la actividad inicial, el cual usamos un recyclerview.

Creación de nueva actividad .xml

```
Pokemon.java x activity_main.xml x item_pokemon.xml x
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="wrap_content"
5      android:orientation="vertical">
6
7      <ImageView
8          android:id="@+id/fotoImageView"
9          android:layout_width="96dp"
10         android:layout_height="96dp"
11         android:layout_gravity="center_horizontal" />
12
13     <TextView
14         android:id="@+id/nombreTextView"
15         android:layout_width="match_parent"
16         android:layout_height="wrap_content"
17         android:gravity="center_horizontal"
18         android:text="Pokédex"
19         android:textAllCaps="true" />
20
21 </LinearLayout>
```

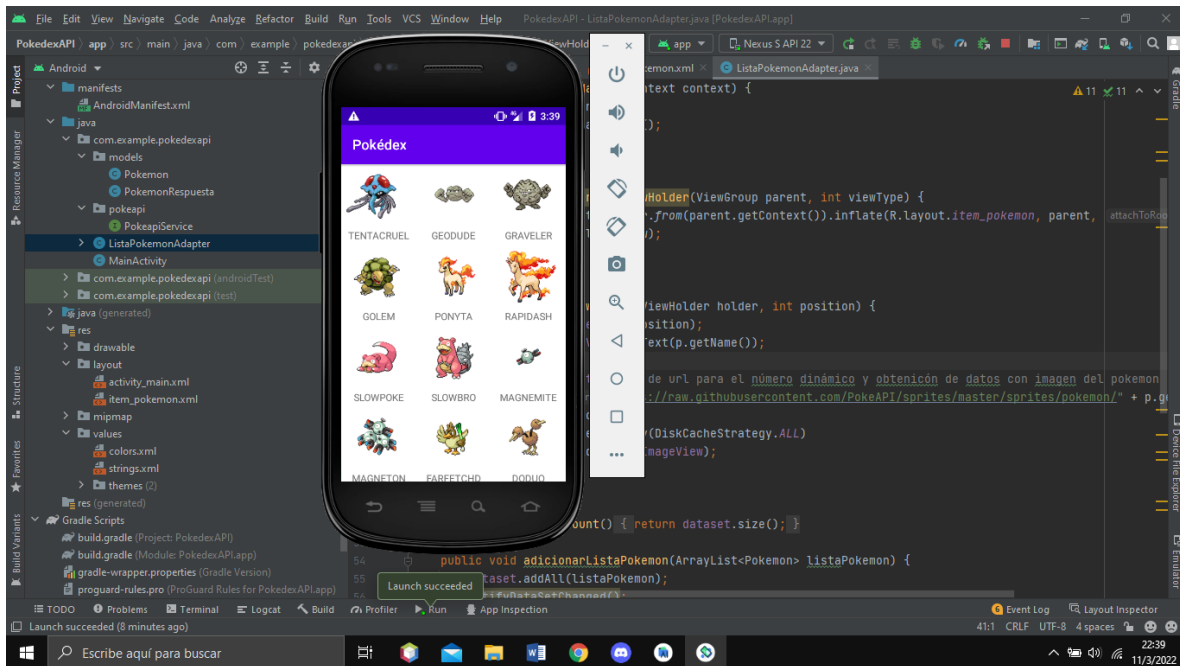
Este diseño es para cada ítem de los datos, en este caso mostraremos el nombre y la imagen del pokemon, más que nada para tener todo organizado y no en un solo lugar mezclado.

ListaPokemonAdapter

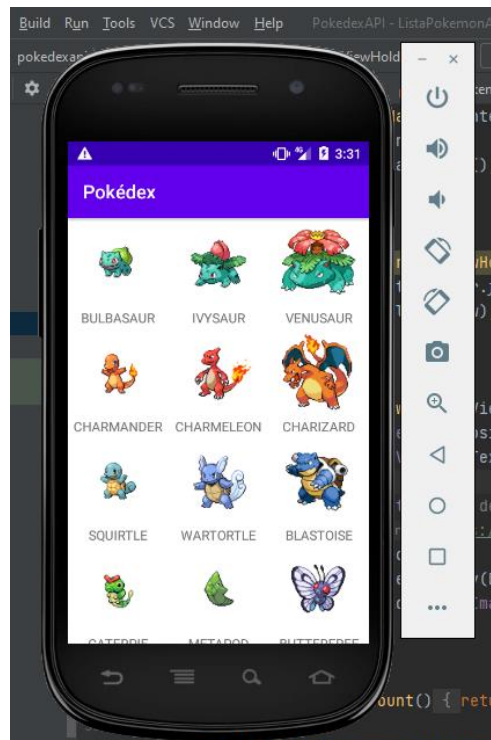
```
Pokemon.java x activity_main.xml x item_pokemon.xml x ListaPokemonAdapter.java x
1  package com.example.pokdexapi;
2
3  import ...
4
5
6  public class ListaPokemonAdapter extends RecyclerView.Adapter<ListaPokemonAdapter.ViewHolder> {
7
8      private ArrayList<Pokemon> dataset;
9      private Context context;
10
11      public ListaPokemonAdapter(Context context) {
12          this.context = context;
13          dataset = new ArrayList<>();
14      }
15
16      @Override
17      public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
18          View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_pokemon, parent, false);
19          return new ViewHolder(view);
20      }
21
22      @Override
23      public void onBindViewHolder(ViewHolder holder, int position) {
24          Pokemon p = dataset.get(position);
25          holder.nombreTextView.setText(p.getName());
26      }
27  }
```

Aquí se gestionarán los datos y se asignará el nombre al pokemon, es decir se une el nombre con su respectiva imagen, capacidad de mostrar los datos en grillas o celdas.

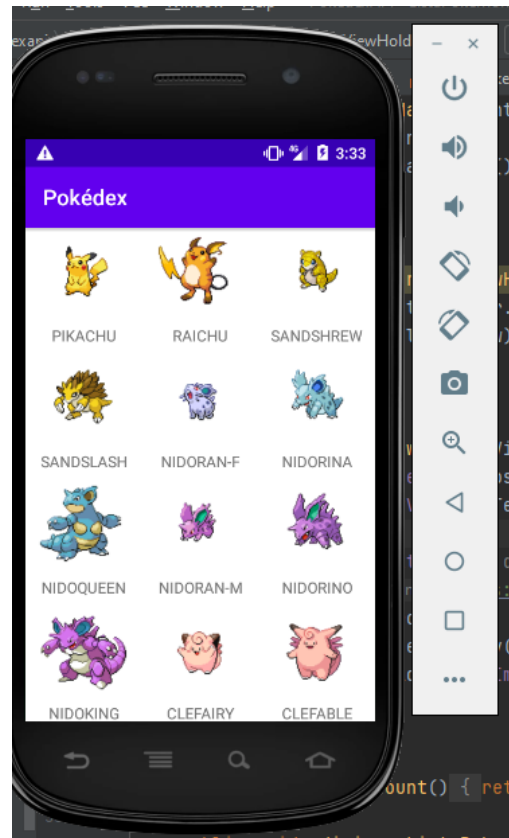
4. EJECUCIÓN DEL PROYECTO



Al ejecutar el proyecto, nuestro emulador muestra lo siguiente, cada pokemon, su imagen y su nombre debajo, como mencioné, todo está organizado para que esté relacionado cada uno; en sí; cada datos e imagen están en una celda o grilla como se muestra a continuación:



Al momento de bajar, se siguen descargando los demás datos que con métodos hemos definido en el proyecto:



Resultados Esperados

Esta sería la finalización del proyecto final, cumpliéndose las reglas y el uso e la librería retrofit para consumir cualquier web service de tipo REST API.

Bibliografía

Cómo instalar Android Studio | Desarrolladores de Android. (s. f.). Android Developers.

Recuperado 6 de enero de 2020, de <https://developer.android.com/studio/install?hl=es> 3. Firmas de Responsabilidad

Diseño de Interfaces » Pautas de diseño de dispositivos móviles. (s. f.). Recuperado 6 de enero de 2020, de <http://multimedia.uoc.edu/blogs/dii/es/disseny/pautes-dedisseny/pautes-de-dissen>

Flutter Inicio. (s. f.). Recuperado 18 de febrero de 2021, de <https://flutter.dev/docs/getstarted/install-de-dispositius-mobils/>