



## GUÍA PRÁCTICA

### 1. Datos Generales

<b>Carrera:</b>	<b>Tecnología Superior en Desarrollo de Software</b>
<b>Período académico:</b>	<b>Diciembre 2021 – Abril 2022</b>
<b>Asignatura:</b>	<b>Desarrollo de Aplicaciones Móviles</b>
<b>Unidad N°:</b>	<b>4 Consumiendo Web Services, Acceso a Datos, Sincronización de Información</b>
<b>Tema:</b>	<b>Consumiendo WebServices</b>
<b>Ciclo-Paralelo:</b>	<b>M4A</b>
<b>Fecha de inicio de la Unidad:</b>	<b>15/02/2022</b>
<b>Fecha de fin de la Unidad</b>	<b>16/03/2022</b>
<b>Práctica N°:</b>	<b>4</b>
<b>Horas:</b>	<b>12</b>
<b>Docente:</b>	<b>Ing. Patricio Pacheco</b>
<b>Estudiante:</b>	<b>Mariela Alexandra León Yunga.</b>

### 1. Contenido

#### 1.1 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

#### 1.2 Objetivo de la Guía

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

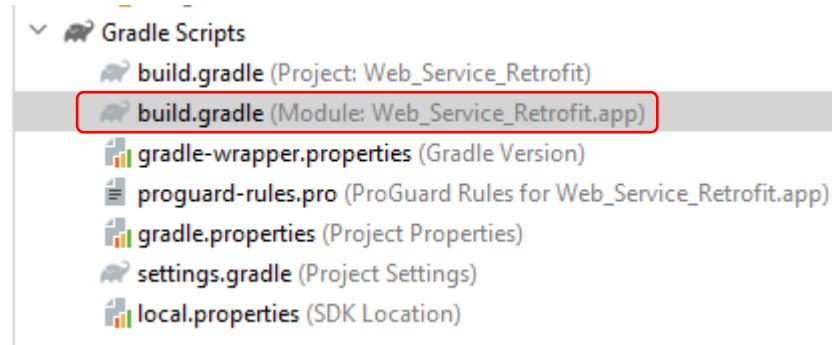
#### 1.3 Materiales, herramientas, equipos y software

- Equipos de computación,
- Android Studio
- Internet,
- Material Guía (Talleres, ejercicios prácticos).

## 2 Procedimiento

### 1. Añadir la librería de Retrofit a nuestro proyecto.

En este caso usaremos el método más común y recomendado: añadiremos Retrofit vía Gradle.



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

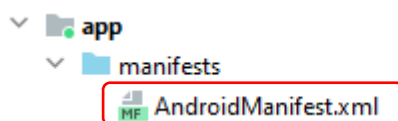
```
dependencies {  
  
    implementation 'androidx.appcompat:appcompat:1.4.2'  
    implementation 'com.google.android.material:material:1.6.1'  
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'  
    implementation 'com.squareup.retrofit2:retrofit:2.4.0'  
    implementation 'com.squareup.retrofit2:converter-gson:2.4.0'  
    testImplementation 'junit:junit:4.13.2'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'  
}
```

Dentro del archivo debes agregar 2 dependencias. Una para Retrofit y otra para GSON.

### 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



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.mariela.leon.web_service_retrofit">

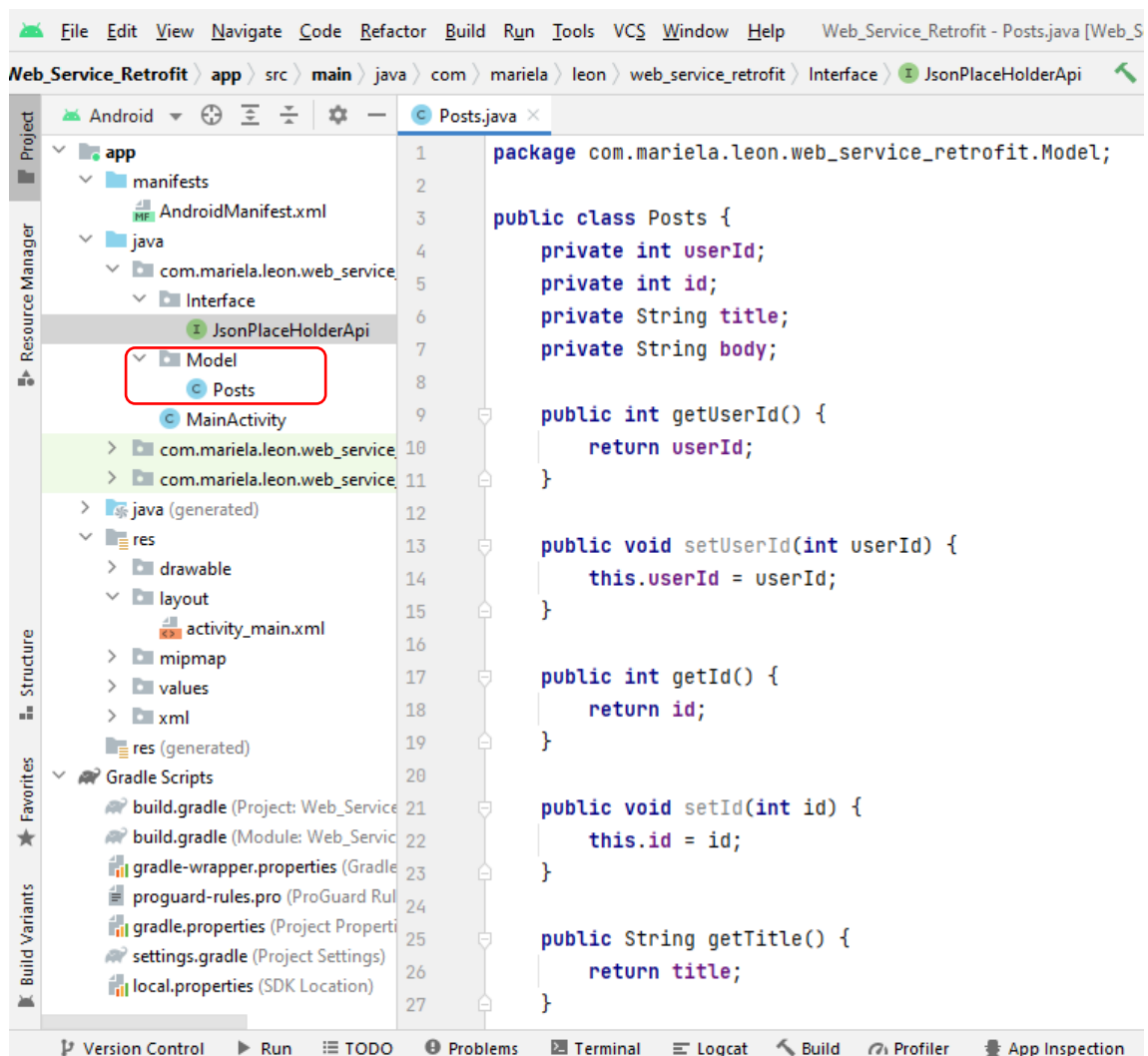
    <uses-permission android:name="android.permission.INTERNET"/>

```

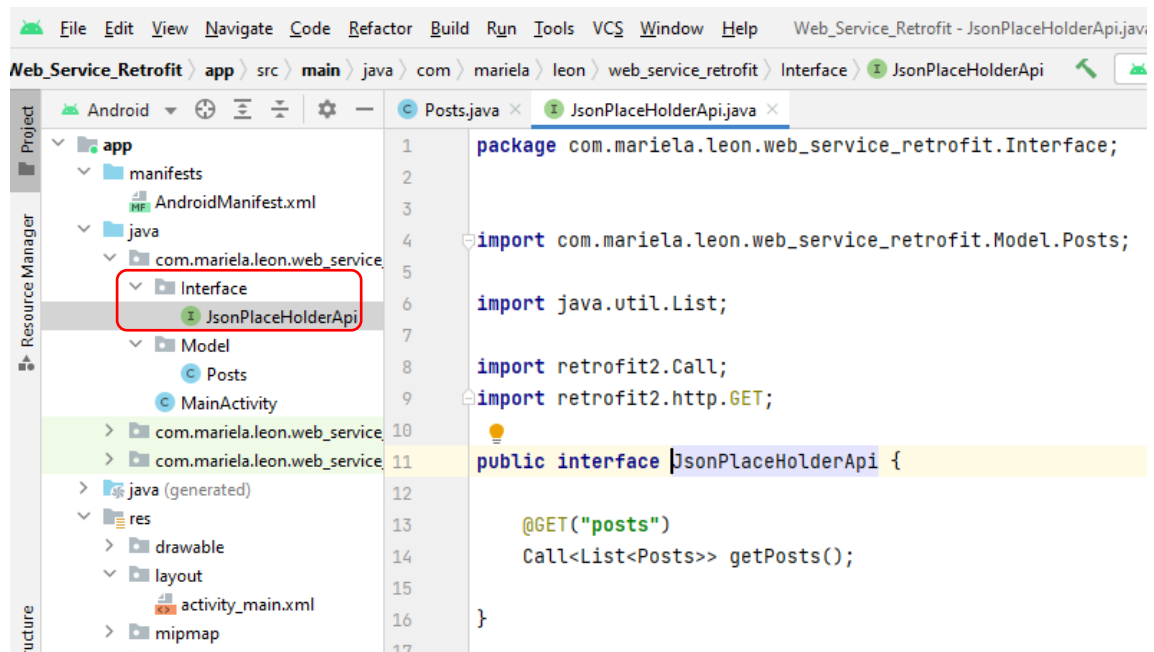
### 3. Crear Una clase y una interfaz

En la clase de Posts que hemos creado incorporamos todos los atributos que contine la página que hemos escogido.

También debemos generar los get and set.

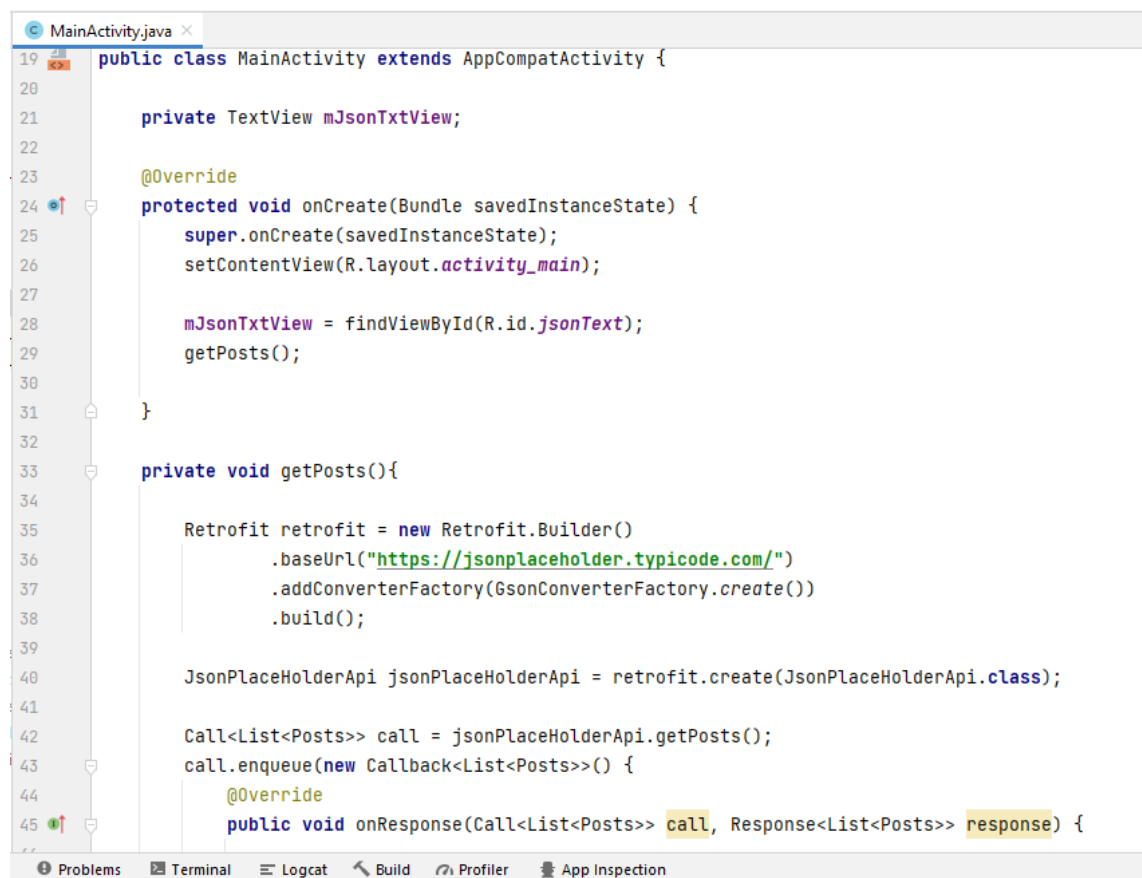


En la Interfaz que hemos creado, vamos a crear un método que va a ser el encargado de traer toda la información.



```
1 package com.mariela.leon.web_service_retrofit.Interface;
2
3
4 import com.mariela.leon.web_service_retrofit.Model.Posts;
5
6 import java.util.List;
7
8 import retrofit2.Call;
9 import retrofit2.http.GET;
10
11 public interface JsonPlaceholderApi {
12
13     @GET("posts")
14     Call<List<Posts>> getPosts();
15
16 }
17
```

## Clase del MainActivity



```
19 public class MainActivity extends AppCompatActivity {
20
21     private TextView mJsonTextView;
22
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26         setContentView(R.layout.activity_main);
27
28         mJsonTextView = findViewById(R.id.jsonText);
29         getPosts();
30     }
31
32     private void getPosts(){
33
34         Retrofit retrofit = new Retrofit.Builder()
35             .baseUrl("https://jsonplaceholder.typicode.com/")
36             .addConverterFactory(GsonConverterFactory.create())
37             .build();
38
39         JsonPlaceholderApi jsonPlaceholderApi = retrofit.create(JsonPlaceholderApi.class);
40
41         Call<List<Posts>> call = jsonPlaceholderApi.getPosts();
42         call.enqueue(new Callback<List<Posts>>() {
43             @Override
44             public void onResponse(Call<List<Posts>> call, Response<List<Posts>> response) {
45
46             }
47         })
48     }
49 }
```

```

43      call.enqueue(new Callback<List<Posts>>() {
44          @Override
45          public void onResponse(Call<List<Posts>> call, Response<List<Posts>> response) {
46
47              if(!response.isSuccessful()){
48                  mJsonTextView.setText("Codigo: "+response.code());
49                  return;
50              }
51
52              List<Posts> postsList = response.body();
53
54              for(Posts post: postsList){
55                  String content = "";
56                  content += " userId: " + post.getUserId()+"\n";
57                  content += " id: " + post.getId()+"\n";
58                  content += " title: " + post.getTitle()+"\n";
59                  content += " body: " + post.getBody()+"\n\n";
60                  mJsonTextView.append(content);
61              }
62          }
63
64          @Override
65          public void onFailure(Call<List<Posts>> call, Throwable t) {
66
67              mJsonTextView.setText(t.getMessage());
68          }
69      }

```

En el MainActivity.xml contamos con los siguiente para dar formato de la información que se mostrara en el emulador.

```

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

    <androidx.core.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:id="@+id/jsonText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"

            android:layout_marginStart="25dp"
            android:layout_marginEnd="25dp"
            android:textColor="@color/black"
            android:textSize="16sp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"

```

Aquí podemos verificar que hemos traído la misma información que se encuentra en el servidor.

The screenshot displays the Android Studio IDE with a mobile emulator and a web browser. The emulator, titled 'Web\_Service\_Retrofit', shows a list of items with the following fields: 'userId', 'id', 'title', and 'body'. The 'id' field is circled in red in the emulator. The web browser, titled 'jsonplaceholder.typicode.com/posts', shows the JSON data for the first three posts. The 'id' field is circled in red in the JSON data, matching the values in the emulator (1, 2, and 3).

```
Web_Service_Retrofit
```

```
userId: 1  
id: 1  
title: sunt aut facere repellat provident occaecati excepturi optio reprehenderit  
body: quia et suscipit  
suscipit recusandae consequuntur expedita et cum  
reprehenderit molestiae ut ut quas totam nostrum rerum est autem sunt rem eveniet architecto
```

```
userId: 1  
id: 2  
title: qui est esse  
body: est rerum tempore vitae sequi sint nihil reprehenderit dolor beatae ea dolores neque fugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis qui aperiam non debitis possimus qui neque nisi nulla
```

```
userId: 1  
id: 3  
title: ea molestias quasi exercitationem repellat qui ipsa sit aut  
body: et iusto sed quo iure
```

```
{  
  "userId": 1,  
  "id": 1,  
  "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",  
  "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"  
},  
{  
  "userId": 1,  
  "id": 2,  
  "title": "qui est esse",  
  "body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\nqui aperiam non debitis possimus qui neque nisi nulla"  
},  
{  
  "userId": 1,  
  "id": 3,  
  "title": "ea molestias quasi exercitationem repellat qui ipsa sit aut",  
  "body": "et iusto sed quo iure\nvoluptatem occaecati omnis eligendi aut ad\nvoluptatem doloribus vel accusantium quis pariatur\nmolestiae porro eius odio et labore et velit aut"  
},  
{  
  "userId": 1,  
  "id": 4,  
  "title": "et iusto sed quo iure",  
  "body": "voluptatem occaecati omnis eligendi aut ad\nvoluptatem doloribus vel accusantium quis pariatur\nmolestiae porro eius odio et labore et velit aut"  
}
```

Link de la información que se muestra en el emulador: <https://jsonplaceholder.typicode.com/posts>

### 3 Resultados esperados

- Al final de la guía, el estudiante está en capacidad de consumir cualquier web service de tipo REST API.

### Conclusión

En este proyecto hemos utilizado las implementaciones de retrofit con el que hemos podido desarrollar el proyecto con un api que nos ofrece datos de prueba para poder testear nuestra app.

**Link del Repositorio:** <https://github.com/marielalexandra05/Guia-4-Retrofit>