



Instituto Superior
Tecnológico del Azuay

NOMBRE: ABIGAIL ZHINGRI

CURSO : M4"B"

TEMA: Guia Practica 4

MATERIA: Desarrollo de Aplicaciones Móviles

Introducción:

Retrofit es una librería para Android y Java compatible con Kotlin para hacer llamadas de red, obtener el resultado y “parsearlo” de forma automática a su objeto, esto facilita mucho realizar peticiones a un API y procesar la respuesta.

Implementación de Retrofit.

La API pública que utilicé fue esta : <https://jsonplaceholder.typicode.com/comments>.

1. En el manifest.xml agregamos el permiso de internet, esto permite que acceda al enlace de la api con la que vamos a trabajar.

```
package="com.example.zhimgri_retrofit">  
  <uses-permission android:name="android.permission.INTERNET"/>  
  
  <application
```

2.- Se añadió la librería retrofit al proyecto , en la parte de build.gradle , en la sección del módulo.

```
build.gradle (Project: Zhimgri_Retrofit) 25  
build.gradle (Module: Zhimgri_RetrofitApp) 26  
gradle-wrapper.properties (Gradle Version) 27  
proguard-rules.pro (ProGuard Rules for Zhimgri_RetrofitApp) 28  
gradle.properties (Project Properties) 29  
settings.gradle (Project Settings) 30  
local.properties (SDK Location) 31  
32  
33  
34  
35  
36  
37  
38  
39  
sourceCompatibility JavaVersion.VERSION_1_8  
targetCompatibility JavaVersion.VERSION_1_8  
dependencies {  
  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'  
  implementation 'androidx.appcompat:appcompat:1.4.1'  
  implementation 'com.google.android.material:material:1.5.0'  
  implementation 'androidx.constraintlayout:constraintlayout:2.1.3'  
  testImplementation 'junit:junit:4.+'  
  androidTestImplementation 'androidx.test.ext:junit:1.1.3'
```

3.- Agregue un nuevo paquete llamado “modelo” en que va la clase comentarios con todos los atributos y los getters y setters.

```

1 public class Comentarios {
2     private int postId;
3     private int id;
4     private String name;
5     private String email;
6     private String body;
7
8
9
10    public int getPostId() {
11        return postId;
12    }
13
14    public void setPostId(int postId) { this.postId = postId; }
15
16    public int getId() { return id; }
17
18    public void setId(int id) { this.id = id; }
19
20    public String getName() { return name; }
21
22    public void setName(String name) { this.name = name; }
23
24    public String getEmail() { return email; }
25
26    public void setEmail(String email) { this.email = email; }
27
28    public String getBody() { return body; }
29
30
31
32
33
34
35
36
37
38
39
40
41
42

```

4.- Agregue un nuevo paquete llamado “interfaz” en el cual va una clase interfaz , donde se va a importar la clase comentarios en un List, además con el método get Comentarios .

```

1 package com.example.zhimgri_retrofit.interfaz;
2
3 import ..
4
5
6
7
8
9
10 public interface MyApiService {
11     @GET("comments")
12     Call<List <Comentarios>> getComentarios();
13 }

```

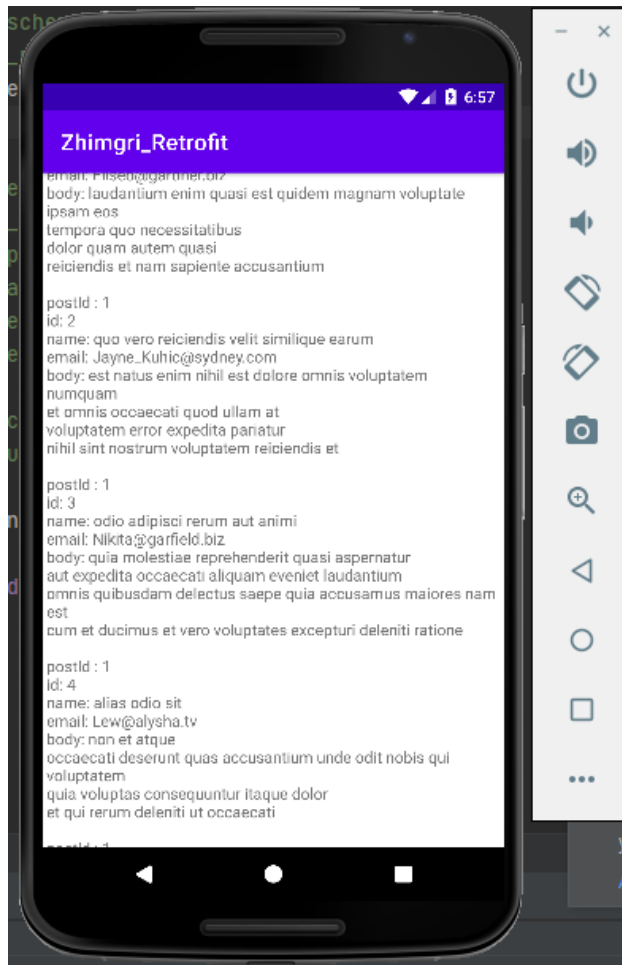
5.- En la clase Main Activity, realizamos la codificación con retrofit de la siguiente forma:

- Creamos un método llamado “get Comentarios” de tipo void .
- Aquí se realiza toda la codificación para implementar retrofit, además se importará la clase Comentarios donde se devolverán los datos de la clase.

```
MainActivity.java x activity_main.xml x MyApiService.java x build.gradle (:app) x Comentarios.java x
20
21 public class MainActivity extends AppCompatActivity {
22
23
24     TextView listView;
25
26     @Override
27     protected void onCreate(Bundle savedInstanceState) {
28         super.onCreate(savedInstanceState);
29         setContentView(R.layout.activity_main);
30
31         listView = findViewById(R.id.textviewdatos);
32         getComentarios();
33     }
34     private void getComentarios(){
35         Retrofit retrofit = new Retrofit.Builder().baseUrl("https://jsonplaceholder.typicode.com/").addConverterFactory(Gson
36         MyApiService myApiService = retrofit.create(MyApiService.class);
37
38         Call<List<Comentarios>> call = myApiService.getComentarios();
39         call.enqueue(new Callback<List<Comentarios>>() {
40             @Override
41             public void onResponse(Call<List<Comentarios>>call, Response<List<Comentarios>>response) {
42                 if (!response.isSuccessful()){
43                     listView.setText("Codigo: "+response.code());
44                     return;
45                 }
46                 List<Comentarios> datos=response.body();
47                 for (Comentarios d1:datos){
```

```
MainActivity.java x activity_main.xml x MyApiService.java x build.gradle (:app) x Comentarios.java x
37
38         Call<List<Comentarios>> call = myApiService.getComentarios();
39         call.enqueue(new Callback<List<Comentarios>>() {
40             @Override
41             public void onResponse(Call<List<Comentarios>>call, Response<List<Comentarios>>response) {
42                 if (!response.isSuccessful()){
43                     listView.setText("Codigo: "+response.code());
44                     return;
45                 }
46                 List<Comentarios> datos=response.body();
47                 for (Comentarios d1:datos){
48                     String mostrar = "";
49                     mostrar += "postId: " + d1.getPostId()+"\n";
50                     mostrar += "id: " + d1.getId()+"\n";
51                     mostrar += "name: " + d1.getName()+"\n";
52                     mostrar += "email: " + d1.getEmail()+"\n";
53                     mostrar += "body: " + d1.getBody()+"\n\n";
54                     listView.append(mostrar);
55                 }
56             }
57         }
58
59         @Override
60         public void onFailure(Call<List<Comentarios>>call, Throwable t) {
61             listView.setText(t.getMessage());
62         }
63     });
64 }
65 }
```

6.- Al ejecutar , en el emulador se observan los datos que hemos implementado.



Conclusiones :

Esta librería retrofit permite realizar peticiones al servidor ya sea de tipo get o set o incluso de otro tipo, además se puede gestionar diferentes tipos de parámetros. Para consumir aquella librería se debe importar la librería, en el Gradle en la parte de module.