Instituto Superior
Tecnológico del Azuay

GUIA PRACTICA 4

NOMBRE:

David Becerra

CURSO:

M4B

DOCENTE:

Ing. Patricio Pacheco

MATERIA:

Desarrollo de Aplicaciones Móviles

FECHA DE ENTREGA:

viernes, 18 de marzo de 2022

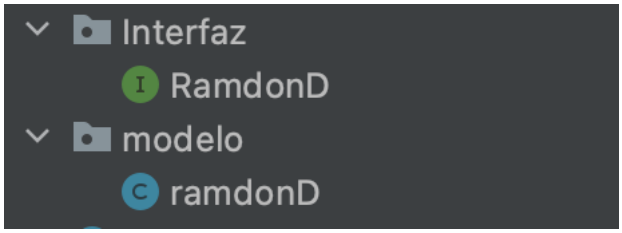
Link de Github: https://github.com/ForaneoBlack/App_Retrofit.git

Para comenzar se crea el Proyecto en Android Studio.

1. Se añade dependencias build.glade para comenzar a manejar las librerías para llamar a api publicas o locales, en este caso utilizaremos la librería de retrofit y gson para el procesado de datos.

```
implementation 'com.squareup.retrofit2:retrofit:2.5.0'
implementation 'com.squareup.retrofit2:converter-gson:2.5.0'
implementation 'com.squareup.okhttp3:logging-interceptor:3.9.1'
```

2. Despues crearemos dos paquetes uno sera el modelo y el otro la interfaz.



3. En el modelo se incluira todas las variables que se necesiten para llamar a los datos que deseemos de la api en este caso, incluiremos estos datos, al iugal que tambien generar los getter and setter de los datos.

```
public class ramdonD {

    private int userId;
    private int id;
    private String title;
    private String body;

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getBody() {
        return body;
    }
}
```

4. En la interfaz el llamado de los variables del modelo y tambien la de donde se extraera los datos, este caso solo sera de post

```java
public interface RamdonD {

    @GET("posts")
    Call<List<ramdonD>> getRamdons();

}
```

Call de la librería de retrofit para poder llamar al api y extraer datos

5. En el layout se realizo una pequeña modificacion para poder examinar los datos con el textView tambien utilizando NestedScrollView

```xml
<android.support.v4.widget.NestedScrollView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">

    <TextView
        android:id="@+id/ramdonText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</android.support.v4.widget.NestedScrollView>
```

6. En el main comenzamos con la parte de llamar al textView

```java
private TextView ramdonText;
```

7. En el create definimos lo siguiente:

```java
ramdonText = findViewById(R.id.ramdonText);
```

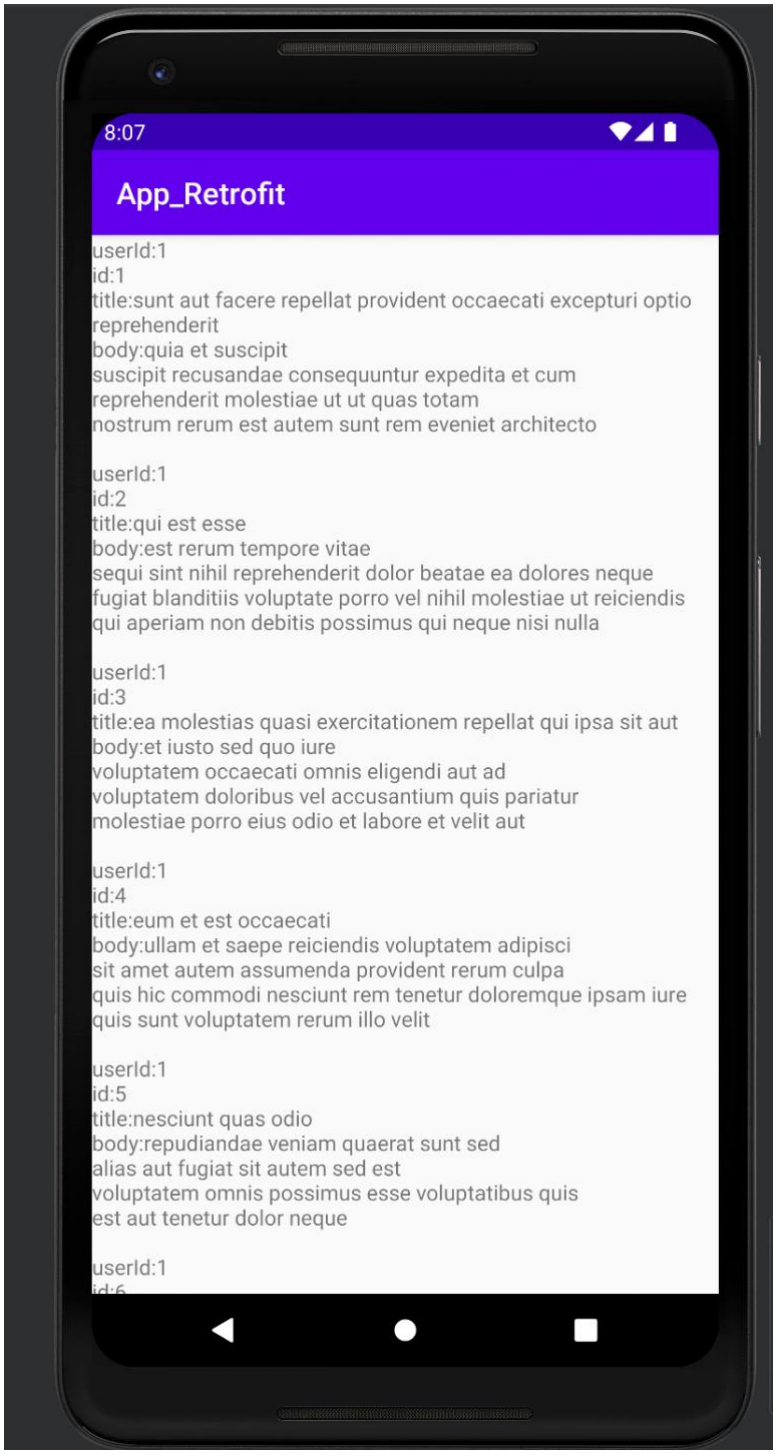
8. Luego creamos un metodo para llamar al api y los datos consigo:

```
private void getRamdon() {
    Retrofit retrofit = new Retrofit.Builder()
            .baseUrl("https://jsonplaceholder.typicode.com/")
            .addConverterFactory(GsonConverterFactory.create())
            .build();

    RamdonD ram = retrofit.create(RamdonD.class);
    Call<List<ramdonD>> call = ram.getRamdons();
    call.enqueue(new Callback<List<ramdonD>>() {
        @Override
        public void onResponse(Call<List<ramdonD>> call, Response<List<ramdonD>> response) {
            if(!response.isSuccessful()){
                ramdonText.setText("Codigo: "+response.code());
                return;
            }
            List<ramdonD> ramdonDList = response.body();
            for (ramdonD ran: ramdonDList ) {
                String content = "";
                content += "userId:"+ ran.getUserId() + "\n";
                content += "id:"+ ran.getId() + "\n";
                content += "title:"+ ran.getTitle() + "\n";
                content += "body:"+ ran.getBody() + "\n\n";
               ramdonText.append (content);
            }
        }

        @Override
        public void onFailure(Call<List<ramdonD>> call, Throwable t) {
            ramdonText.setText(t.getMessage());

        }
    });
```

9. Agregamos el metodo al OnCreate y quedaria asi:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    ramdonText = findViewById(R.id.ramdonText);
    getRamdon();
}
```

10. Por ultimo ejecutamos y se veria de la siguiente manera:


8:07
App_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
voluptatem occaecati omnis eligendi aut ad
voluptatem doloribus vel accusantium quis pariatur
molestiae porro eius odio et labore et velit aut
userId:1
id:4
title:eum et est occaecati
body:ullam et saepe reiciendis voluptatem adipisci
sit amet autem assumenda provident rerum culpa
quis hic commodi nesciunt rem tenetur doloremque ipsam iure
quis sunt voluptatem rerum illo velit
userId:1
id:5
title:nesciunt quas odio
body:repudiandae veniam quaerat sunt sed
alias aut fugiat sit autem sed est
voluptatem omnis possimus esse voluptatibus quis
est aut tenetur dolor neque
userId:1