
 GOBIERNO DEL ESTADO DE MÉXICO	<b>MANUAL DE PRÁCTICAS</b>  FO-TESJI-11100-12		 <b>TESJI</b> TECNOLÓGICO DE ESTUDIOS SUPERIORES <b>JILOTEPEC</b>	
<b>NOMBRE DE LA PRÁCTICA:</b>	Creación de una app en Android Studio capaz de consumir una api local		<b>No.</b>	2
<b>ASIGNATURA:</b>	Tópicos Avanzados De Programación	<b>CARRERA:</b>	Ingeniería en Sistemas Computacionales	<b>Unidad:</b> I
<b>ALUMNO:</b>	Jesús Navarrete Martínez			

**II. Lugar de realización de la práctica (laboratorio, taller, aula u otro): Aula**

**III. Material empleado:**

- Laptop
- Software Android Studio

**ENCUADRE CACEI**

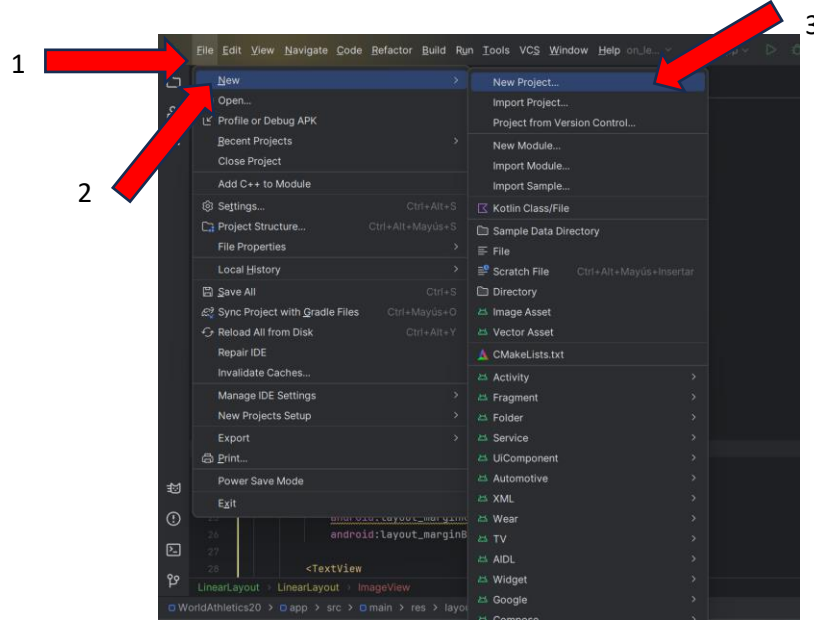
No. atributo	Atributos de egreso del PE que impactan en la asignatura	No.Criterio	Criterios de desempeño	No. Indicador	Indicadores
2	El estudiante diseñará esquemas de trabajo y procesos, usando metodologías congruentes en la resolución de problemas de ingeniería en sistemas computacionales	CD2	Diseña soluciones a problemas, empleando metodologías apropiadas al área	I1	Uso de metodologías para el modelado de la solución de sistemas y aplicaciones
				I2	Diseño algorítmico
3	El estudiante plantea soluciones basadas en tecnologías empleando su juicio ingenieril para valorar necesidades, recursos y resultados esperados.	CD1	Emplea los conocimientos adquiridos para el desarrollo de soluciones	I1	Elección de metodologías, técnicas y/o herramientas para el desarrollo de soluciones
				I2	Uso de metodologías adecuadas para el desarrollo de proyectos
				I3	Generación de productos y/o proyectos
		CD2	Analiza y comprueba resultados	I2	Documentar información de las pruebas realizadas y los resultados



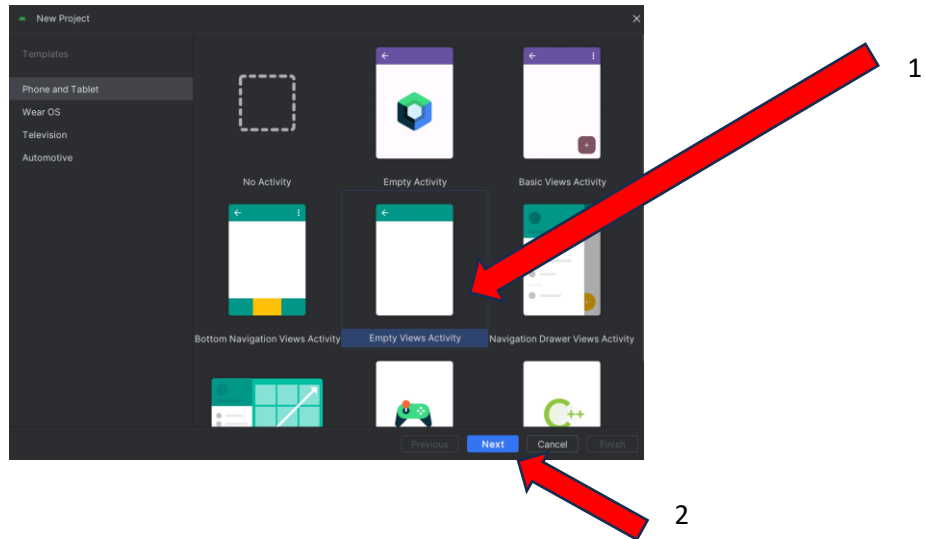
## IV. Desarrollo de la práctica:

### Creación del proyecto:

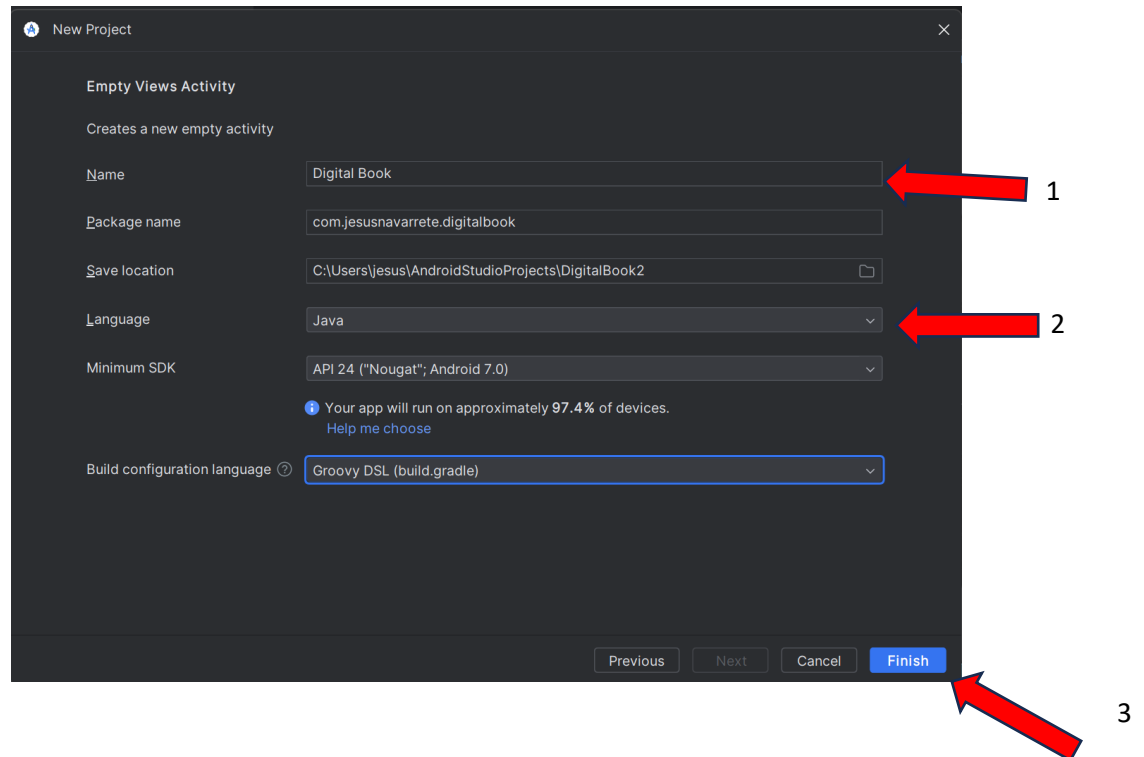
1. Abrimos la app de Android Studio y creamos un nuevo proyecto, para ello nos dirigimos a la parte superior Izquierda de la aplicación y damos click en el apartado **“File”**, posteriormente en **“New”** y por último en **“New Project”**.



2. Ahora se nos desplegará la siguiente ventana en la cual deberemos seleccionar el tipo de proyecto que deseamos desarrollar, en esta ocasión seleccionaremos la opción **“Empty Views Activity”**, para posteriormente dar click en **“Next”**.

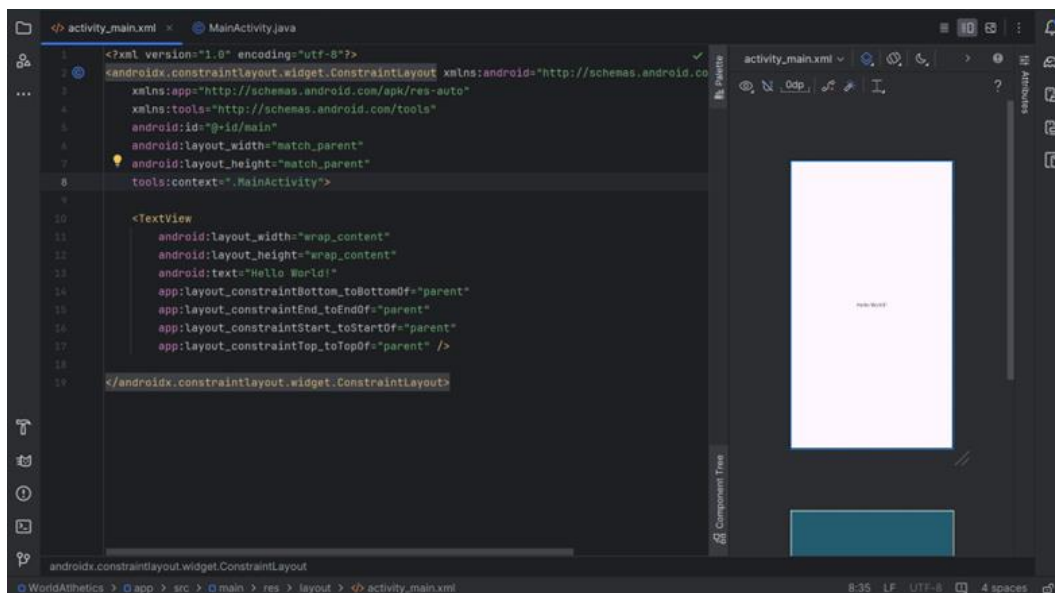


3. Posteriormente nos encontraremos con la siguiente ventana en la cual debemos indicar cual será el nombre de nuestra aplicación, en mi caso la llamare **“Digital Book”**, también deberemos indicar en que lenguaje de programación deseamos desarrollar nuestra app, en este caso será realizada usando **“Java”**, después daremos click en **“Finish”**.



## Creación de la interfaz para la pantalla SPLASH:

1. Ahora podremos visualizar la siguiente ventana, la cual nos ubica en la sección “activity\_main.xml” para comenzar con el desarrollo de la interfaz gráfica para nuestro SPLASH.

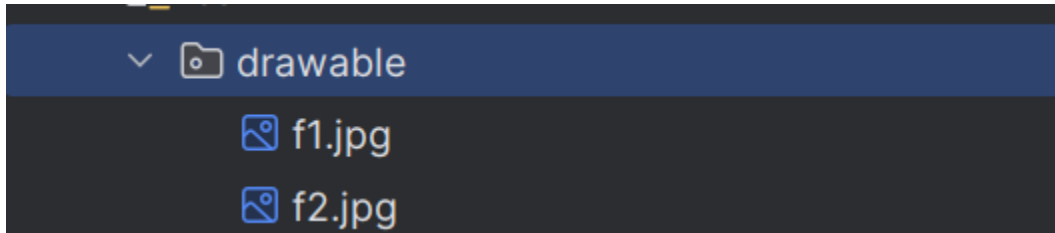




## 2. Lo siguiente que vamos a realizar será modificar el Layout que va a contener todos nuestros demás

elementos visuales para nuestra interfaz, en este Layout vamos a establecer una imagen de fondo para darle una presentación un poco mejor.

**Nota:** Todas las imágenes que vayamos a utilizar deben estar dentro de la carpeta **drawable** de nuestro proyecto.



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical"
    android:background="@drawable/f2">
```

Propiedad para agregar



## 3. Ahora agregaremos un nuevo Layout el cual contendrá una animación JSON relacionada al logo de nuestra aplicación para darle una mejor estética.

**Nota:** todas las animaciones JSON deberán estar dentro de la carpeta raw de nuestro proyecto.

```
<LinearLayout
    android:layout_marginTop="80dp"
    android:layout_marginLeft="50dp"
    android:layout_width="match_parent"
    android:layout_height="200dp">

    <com.airbnb.lottie.LottieAnimationView
        android:id="@+id/animation_view2"
        android:layout_width="326dp"
        android:layout_height="350dp"

        android:layout_gravity="center"
        app:lottie_autoPlay="true"
        app:lottie_loop="true"
        app:lottie_rawRes="@raw/libro7"

        tools:ignore="MissingConstraints" />

</LinearLayout>
```



4. Posteriormente incluiremos un **TextView** para poder visualizar el nombre de nuestra app, el tamaño del texto, la tipografía, el color y otras propiedades puedes modificarlas según tus gustos y necesidades.

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Digital Books"
    android:layout_marginLeft="95dp"
    android:textColor="#FFF"
    android:layout_marginTop="20dp"

    android:textSize="65sp"
    android:fontFamily="@font/cursiva3"

/>
```

5. Ahora agregaremos un nuevo Layout que alojara una nueva animación JSON para simular un Loading... de la aplicación.

```
<LinearLayout
    android:layout_marginLeft="110dp"
    android:layout_width="match_parent"
    android:layout_height="150dp"
    android:layout_marginBottom="180dp">
    <com.airbnb.lottie.LottieAnimationView
        android:id="@+id/animation_view3"
        android:layout_width="226dp"
        android:layout_height="150dp"

        android:layout_gravity="center"
        app:lottie_autoPlay="true"
        app:lottie_loop="true"
        app:lottie_rawRes="@raw/carga6"

        tools:ignore="MissingConstraints" />
    </LinearLayout>
```



## Código Completo:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    android:orientation="vertical"
    android:background="@drawable/f2">
    <LinearLayout
        android:layout_marginTop="80dp"
        android:layout_marginLeft="50dp"
        android:layout_width="match_parent"
        android:layout_height="200dp">
        <com.airbnb.lottie.LottieAnimationView
            android:id="@+id/animation_view2"
            android:layout_width="326dp"
            android:layout_height="350dp"
            android:layout_gravity="center"
            app:lottie_autoPlay="true"
            app:lottie_loop="true"
            app:lottie_rawRes="@raw/libro7"
            tools:ignore="MissingConstraints" />

    </LinearLayout>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/titulo"
        android:layout_marginLeft="95dp"
        android:textColor="#FFF"
        android:layout_marginTop="20dp"
        android:textSize="65sp"
        android:fontFamily="@font/cursiva3"/>

    <LinearLayout
        android:layout_marginLeft="110dp"
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:layout_marginBottom="180dp">
        <com.airbnb.lottie.LottieAnimationView
            android:id="@+id/animation_view3"
            android:layout_width="226dp"
            android:layout_height="150dp"
            android:layout_gravity="center"
            app:lottie_autoPlay="true"
            app:lottie_loop="true"
            app:lottie_rawRes="@raw/carga6"
            tools:ignore="MissingConstraints" />

    </LinearLayout>

</LinearLayout>
```

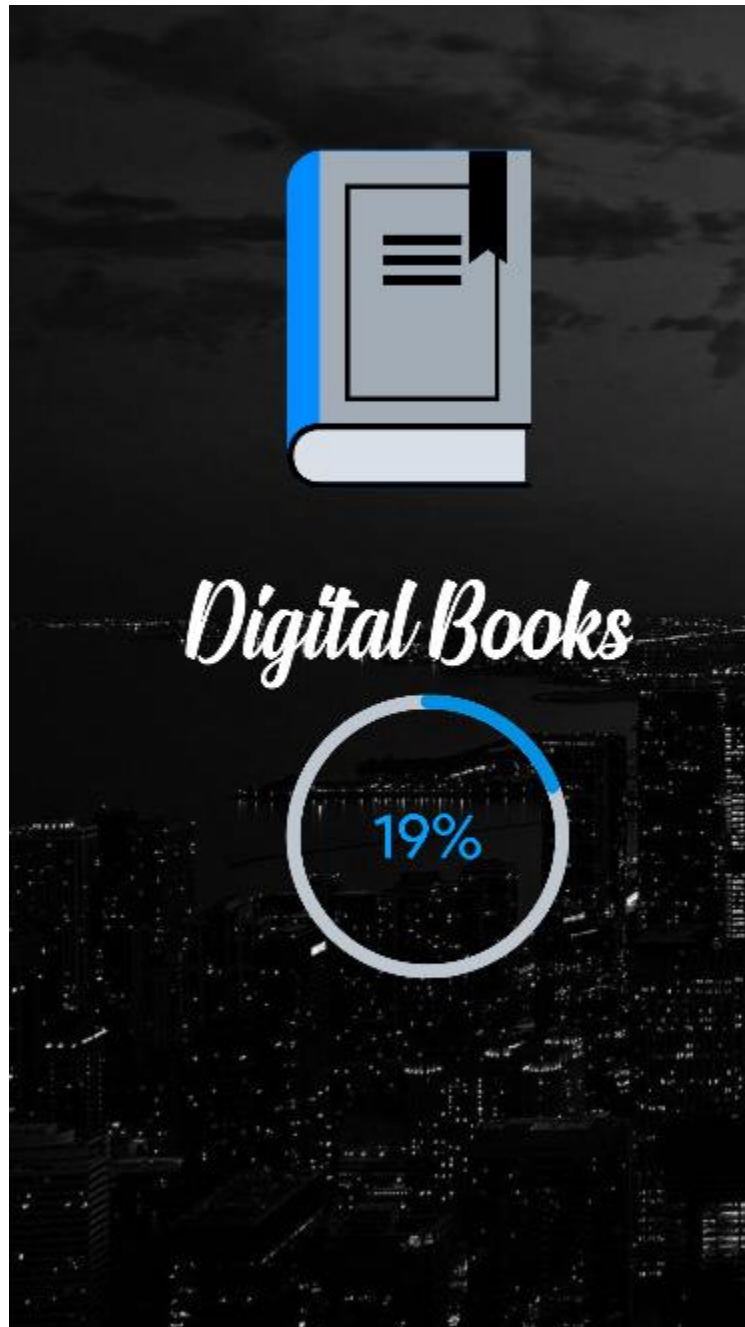


GOBIERNO DEL  
ESTADO DE MÉXICO

## MANUAL DE PRACTICAS



Previsualización del diseño creado:





6. Ahora nos dirigiremos a nuestra clase java llamada MainActivity, en la cual colocaremos el siguiente código, este código programa una tarea para que se ejecute después de 10 segundos. Cuando se ejecuta la tarea, se inicia una nueva actividad llamada `main_home`, ( la cual crearemos más adelante, así que por el momento marcara un error en esa sección, pero enseguida arreglaremos eso) y se cierra la actividad actual. Con esto se va mostrar una pantalla de inicio o una pantalla de bienvenida ( la cual fue la interfaz que acabamos de crear) durante unos segundos antes de redirigir al usuario a la pantalla principal de la aplicación.

```
new Handler().postDelayed(new Runnable() {  
    @Override  
    public void run() {  
        Intent abrirHome= new Intent(getApplicationContext(),main_home.class);  
        startActivity(abrirHome);  
        finish();  
    }  
}, delayMillis: 10000);
```

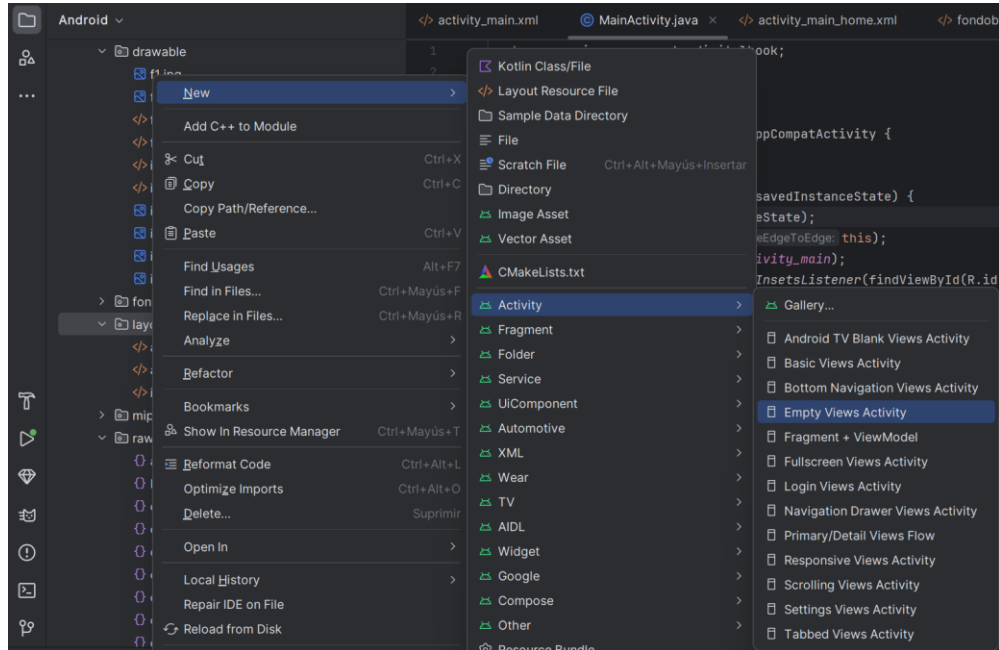
Código completo:

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        EdgeToEdge.enable(this);  
        setContentView(R.layout.activity_main);  
        ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main), (v,  
insets) -> {  
            Insets systemBars =  
insets.getInsets(WindowInsetsCompat.Type.systemBars());  
            v.setPadding(systemBars.left, systemBars.top, systemBars.right,  
systemBars.bottom);  
            return insets;  
        }));  
        new Handler().postDelayed(new Runnable() {  
            @Override  
            public void run() {  
                Intent abrirHome= new  
Intent(getApplicationContext(),main_home.class);  
                startActivity(abrirHome);  
                finish();  
            }  
        },10000);  
    }  
}
```

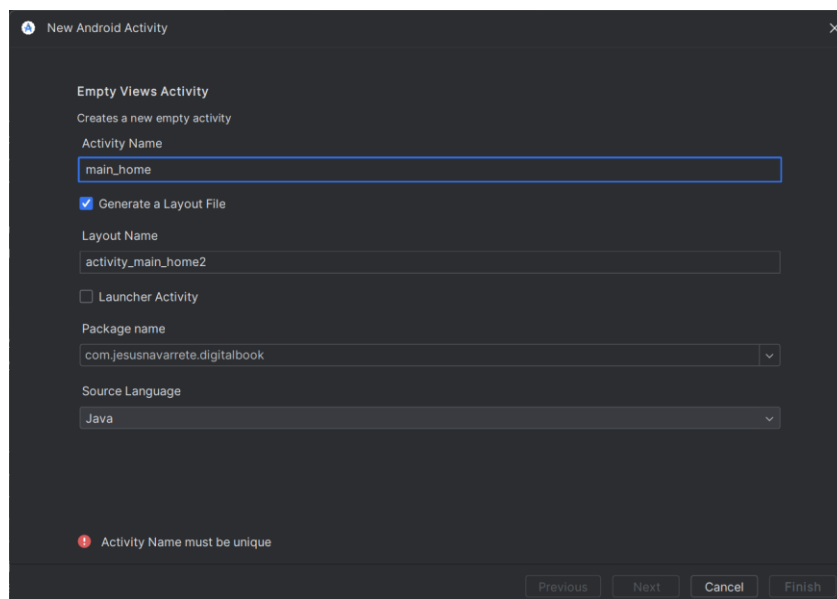




7. Para poder crear la actividad **main\_home** nos vamos a dirigir a la carpeta de nuestro proyecto llamada **layout**, le daremos click derecho y se desplegara una lista de opciones de la cual vamos a seleccionar la opción llamada **New**, posteriormente en la siguiente lista seleccionaremos la opción **Activity** y por ultimo seleccionaremos la opción **Empty Views Activity**.



8. Se desplegara la siguiente ventana en la cual escribiremos el nombre de nuestra nueva actividad la cual será **main\_home**, posteriormente daremos click en finish.





9. Al hacer esto se creara un archivo xml llamado **Activity\_main\_home.xml** y una clase en la carpeta java llamada **main\_home.java** , ahora sobre el archivo xml vamos a trabajar el diseño de la pantalla principal de nuestra aplicación.
10. Comenzaremos a diseñar la interfaz de nuestra pantalla principal, lo primero que haremos será configurar al Layout que contendrá todos nuestros elementos visuales.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".main_home"
    android:orientation="vertical"
    android:background="@drawable/f2">
```

11. Ahora vamos a crear un nuevo Layout, el cual va contener el nombre de nuestra aplicación utilizando un TextView y una animación JSON para hacer más interactiva la interfaz.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="160dp"
    >

    <TextView
        android:layout_width="270dp"
        android:layout_height="match_parent"
        android:layout_marginTop="50dp"
        android:fontFamily="@font/cursiva3"
        android:text="@string/titulo"
        android:textColor="#FFF"
        android:textSize="70sp"
        android:textStyle="bold|italic" />

    <com.airbnb.lottie.LottieAnimationView
        android:id="@+id/animation_view3"
        android:layout_width="120dp"
        android:layout_height="150dp"
        android:layout_gravity="center"
        app:lottie_autoPlay="true"
        app:lottie_loop="true"
        app:lottie_rawRes="@raw/logo3"
        tools:ignore="MissingConstraints" />

</LinearLayout>
```



12. Ahora agregaremos un nuevo Layout con las propiedades mostradas en la imagen, el cual contendrá el slogan de nuestra aplicación.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="40dp"
        android:text="Miles de historias en un solo lugar"
        android:textColor="#fff"
        android:textAlignment="center"
        android:textSize="25sp"
        android:fontFamily="@font/lilita">
    </TextView>
</LinearLayout>
```

13. Posteriormente agregaremos un nuevo Layout que contendrá mas animaciones JSON y el nombre de nuestra aplicación para darle un aspecto mas agradable a la interfaz.

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="40dp"
    android:orientation="horizontal">
    <com.airbnb.lottie.LottieAnimationView
        android:id="@+id/animation_view5"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:layout_marginLeft="70dp"
        android:layout_marginTop="10dp"
        android:layout_gravity="center"
        app:lottie_autoPlay="true"
        app:lottie_loop="true"
        app:lottie_rawRes="@raw/libro5"
    />
    <TextView
        android:layout_marginTop="10dp"
        android:layout_width="150dp"
        android:layout_height="50dp"
        android:text="@string/logo"
        android:textColor="#808080"
        android:textAlignment="center"
        android:textSize="30sp"
        android:fontFamily="@font/cursiva3">
    </TextView>
    <com.airbnb.lottie.LottieAnimationView
        android:id="@+id/animation_view4"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:layout_marginTop="10dp"
        android:layout_gravity="center"
        app:lottie_autoPlay="true"
        app:lottie_loop="true"
        app:lottie_rawRes="@raw/libro5"
    />
</LinearLayout>
```

14. Nuevamente agregaremos un Layout con un TextView para un nuevo slogan de nuestra aplicación.

```
<LinearLayout
    android:id="@+id/prueba"
    android:layout_width="360dp"
    android:layout_height="30dp"
    android:layout_gravity="center"
    >
    <TextView
        android:layout_marginTop="10dp"
        android:layout_width="350dp"
        android:layout_height="30dp"
        android:text="Tus libros favoritos al alcance de un click"
        android:textColor="#fff"
        android:textAlignment="center"
        android:textSize="15sp"
        android:fontFamily="@font/maguina"
    >
    </TextView>
</LinearLayout>
```

15. Ahora en el siguiente Layout vamos a establecer un botón con las siguientes propiedades, el cual tendrá como funcionalidad desplegar la lista de libros almacenados.

```
<LinearLayout
    android:layout_width="365dp"
    android:layout_height="60dp"
    android:layout_marginLeft="15dp"
    android:id="@+id/hola">
    <Button
        android:layout_marginTop="10dp"
        android:id="@+id/button_get_books"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_weight="1"
        android:onClick="btnGetInfoOnClick"
        android:text="Ver Catalogo De Libros"
        android:textColor="@color/black"
        android:background="@drawable/fondoboton"
        android:drawableLeft="@drawable/icono1"
        android:paddingLeft="20dp"

    />
</LinearLayout>
```



16. En el siguiente Layout definiremos el botón que se encargara de desplegar una sección para poder realizar la búsqueda de un libro mediante su id.

```
<LinearLayout
    android:layout_width="365dp"
    android:layout_height="68dp"
    android:layout_marginLeft="15dp"
    android:id="@+id/hola2">
    <Button
        android:layout_marginTop="18dp"
        android:id="@+id/buscar"
        android:layout_width="58dp"
        android:layout_height="58dp"
        android:layout_weight="1"
        android:onClick="btnGetInfoOnClick"
        android:text="Buscar Libros"
        android:textColor="@color/black"
        android:background="@drawable/fondoboton"
        android:drawableLeft="@drawable/icono1"
        android:paddingLeft="28dp"
    />
</LinearLayout>
```

17. Ahora debemos diseñar dentro de un nuevo Layout la interfaz de la sección que ayudara al usuario a realizar la búsqueda de los libros, la cual contendrá un EditText para que el usuario pueda ingresar el id y un botón para comenzar la búsqueda.

Nota: todos los Layout que contienen el diseño de las secciones a desplegar tendrán la propiedad `android:visibility="gone"` para ocultar la visibilidad hasta que el usuario de click en el botón que abrirá esta sección .

```
<LinearLayout
    android:layout_marginLeft="50dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/buscarlibro"
    android:orientation="horizontal"
    android:visibility="gone">
    <EditText
        android:id="@+id/edit_text_book_id"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="Ingresa ID del libro"
        android:inputType="number"
        android:layout_marginTop="16dp"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true" />
    <Button
        android:id="@+id/button_search_book"
        android:layout_width="120dp"
        android:layout_height="wrap_content"
        android:text="Buscar Libro"
        android:layout_below="@id/edit_text_book_id"
        android:layout_marginTop="16dp"
        android:layout_centerHorizontal="true"
        android:textColor="@color/black"
        android:background="@drawable/fondoboton"
    />
</LinearLayout>
```



18. Ahora en otro Layout definiremos un nuevo botón el cual ayudara a desplegar la seccion que le dará al usuario la opción de poder agregar un libro nuevo al catálogo.

```
<LinearLayout
    android:layout_marginLeft="15dp"
    android:layout_width="365dp"
    android:layout_height="wrap_content"
    android:id="@+id/hola3">
    <Button
        android:layout_marginTop="10dp"
        android:id="@+id/agregar"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_weight="1"
        android:onClick="btnGetInfoOnClick"
        android:text="Agregar Un Nuevo Libro"
        android:textColor="@color/black"
        android:background="@drawable/fondoboton"
        android:drawableLeft="@drawable/icono1"
        android:paddingLeft="20dp"
    />
</LinearLayout>
```

19. Lo siguiente en definir el diseño de la seccion que contendra los elementos para poder agregar un libro, esta contendra 4 EditText los cuales permitirán añadir el título, autor, edición y editorial del libro, también contendra un botón el cual ayudara para la funcionalidad de guardar los datos ingresados por el usuario.

```
<LinearLayout
    android:layout_width="365dp"
    android:layout_marginLeft="15dp"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:id="@+id/agregarLibros"
    android:visibility="gone">
    <EditText
        android:layout_marginTop="10dp"
        android:id="@+id/edit_text_titulo"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Título"
        android:layout_below="@id/recycler_view"
    />
    <EditText
        android:id="@+id/edit_text_autores"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Autores"
        android:layout_below="@id/edit_text_titulo"
        android:layout_marginTop="16dp" />
```



```
<EditText
    android:id="@+id/edit_text_editorial"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Editorial"
    android:layout_below="@id/edit_text_autores"
    android:layout_marginTop="16dp" />

<EditText
    android:id="@+id/edit_text_edicion"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Edición"
    android:layout_below="@id/edit_text_editorial"
    android:layout_marginTop="16dp" />

<Button
    android:id="@+id/button_save_book"
    android:layout_width="120dp"
    android:layout_height="wrap_content"
    android:text="Guardar Libro"
    android:layout_below="@id/edit_text_edicion"
    android:layout_marginTop="16dp"
    android:layout_centerHorizontal="true"
    android:textColor="@color/black"
    android:background="@drawable/fondoboton"
    android:layout_marginLeft="150dp"
    />

</LinearLayout>
```

20. Ahora definiremos el botón que se encargará de desplegar la sección que le permitirá al usuario poder eliminar un libro mediante su id.

```
<LinearLayout
    android:layout_marginLeft="15dp"
    android:layout_width="365dp"
    android:layout_height="wrap_content"
    android:id="@+id/hola4">
    <Button
        android:layout_marginTop="10dp"
        android:id="@+id/eliminar"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_weight="1"
        android:onClick="btnGetInfoOnClick"
        android:text="Eliminar Un Libro"
        android:textColor="@color/black"
        android:background="@drawable/fondoboton"
        android:drawableLeft="@drawable/icono1"
        android:paddingLeft="20dp"
        />
</LinearLayout>
```



21. Posteriormente diseñamos la sección que va contener un EditText para que el usuario ingrese el id del libro y un botón para lograr la funcionalidad de eliminar el libro.

```
<LinearLayout
    android:layout_marginTop="10dp"
    android:layout_marginLeft="50dp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/eliminarLibro"
    android:visibility="gone">

    <EditText
        android:id="@+id/edit_text_delete_book_id"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:hint="ID del libro a eliminar" />

    <Button
        android:id="@+id/button_delete_book"
        android:layout_width="122dp"
        android:layout_height="wrap_content"
        android:text="Eliminar Libro"
        android:textColor="@color/black"
        android:background="@drawable/fondoboton"
        />

</LinearLayout>
```

22. Después de ello definiremos el botón que desplegará una sección que le permitirá al usuario poder Actualizar los datos de algún libro.

```
<LinearLayout
    android:layout_marginLeft="15dp"
    android:layout_width="365dp"
    android:layout_height="wrap_content"
    android:id="@+id/libAc">

    <Button
        android:layout_marginTop="10dp"
        android:id="@+id/actualizar"
        android:layout_width="50dp"
        android:layout_height="50dp"
        android:layout_weight="1"
        android:text="Actualizar Datos De Un Libro"
        android:textColor="@color/black"
        android:background="@drawable/fondoboton"
        android:drawableLeft="@drawable/icono1"
        android:paddingLeft="20dp"
        android:onClick="btnGetInfoOnClick"
        />

</LinearLayout>
```





23. Ahora diseñaremos la sección que contendrá todos los campos a llenar utilizando EditText para poder ingresar los datos y un botón que ayudará para la funcionalidad de actualizar los datos del libro.

```
<LinearLayout
    android:layout_width="365dp"
    android:layout_marginLeft="15dp"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:id="@+id/actualizacion"
    android:visibility="gone">

    <EditText
        android:id="@+id/edit_text_book_id2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="ID del Libro" />

    <EditText
        android:id="@+id/edit_text_titulo2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Título" />

    <EditText
        android:id="@+id/edit_text_autores2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Autores" />

    <EditText
        android:id="@+id/edit_text_editorial2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Editorial" />

    <EditText
        android:id="@+id/edit_text_edicion2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Edición" />

    <Button
        android:layout_marginLeft="150dp"
        android:id="@+id/button_update_book"
        android:layout_width="150dp"
        android:layout_height="wrap_content"
        android:text="Actualizar Libro"
        android:layout_below="@id/button_save_book"
        android:layout_marginTop="16dp"
        android:layout_centerHorizontal="true"
        android:textColor="@color/black"
        android:background="@drawable/fondoboton"/>

</LinearLayout>
```



24. Por último vamos a definir el RecyclerView en el cual se visualizarán los datos de los libros solicitados por el usuario.

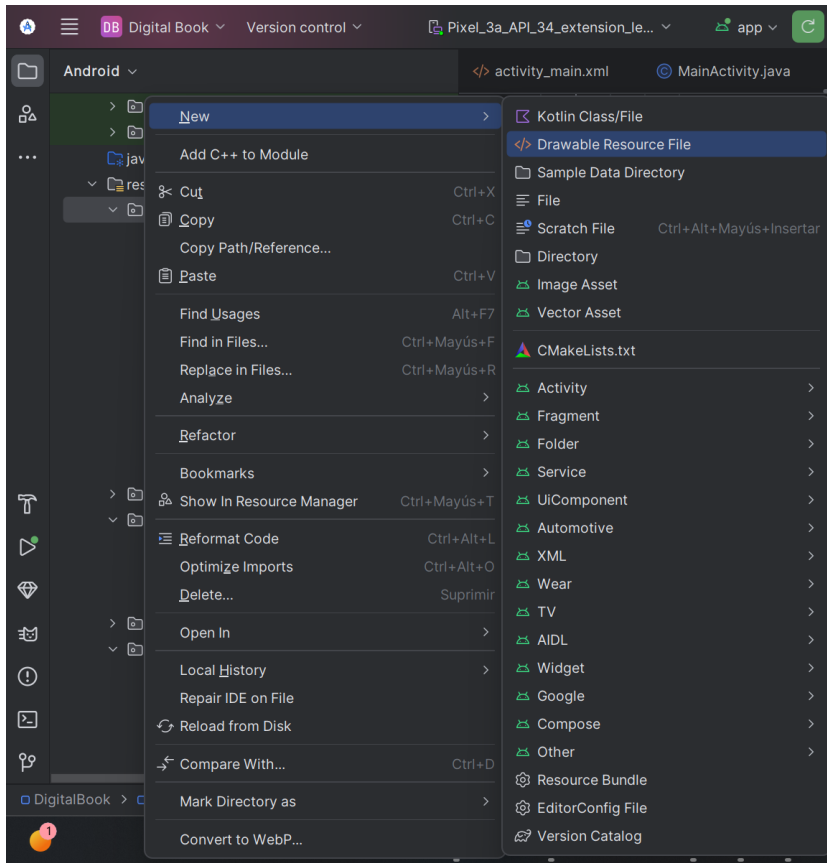
```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/libros"
    >
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/recycler_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"

    />
</LinearLayout>
```

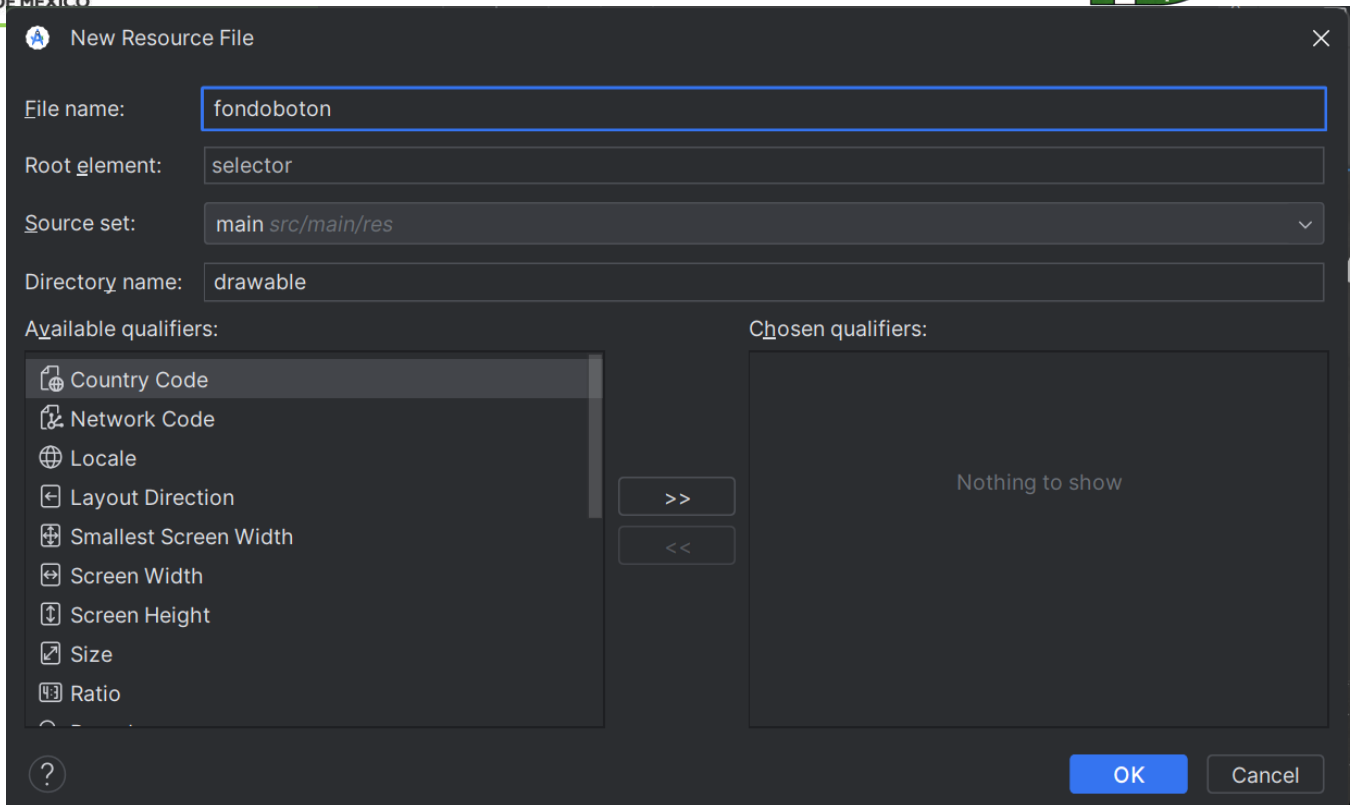
Previsualizacion del diseño creado:



25. Para poder darle un diseño mejor a la interfaz de cada uno de nuestros botones debemos crear un nuevo archivo xml en la carpeta drawable.



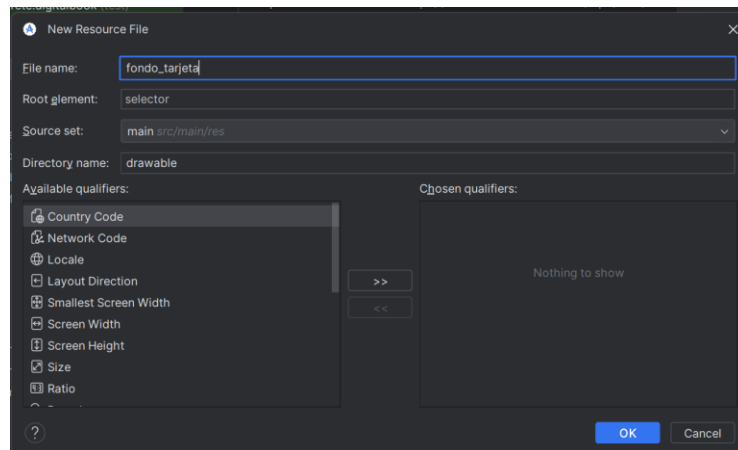
26. Se desplegará la siguiente ventana y pondremos el nombre para el archivo con el diseño



27. Una vez creado el archivo contendrá lo siguiente para definir los colores.

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
  <!--<solid android:color="#4DB6AC"/>-->
  <gradient
    android:startColor="#43ddfd"
    android:endColor="#05849f"/>
  <stroke android:width="2dp"
    android:color="#FFF"/>
  <corners android:radius="20dp"/>
</shape>
```

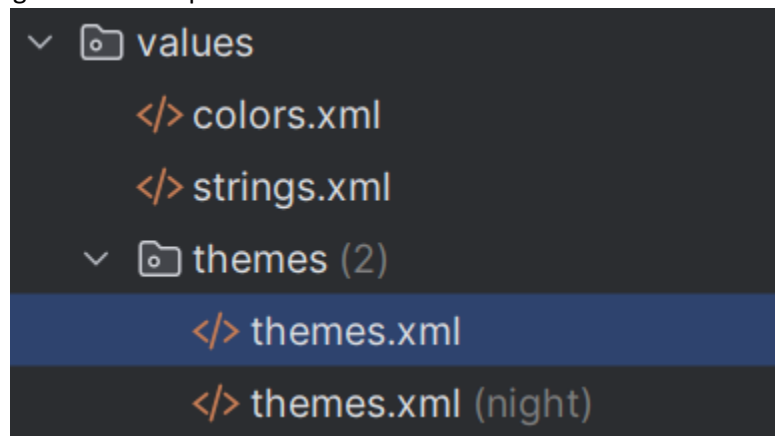
28. Volveremos a realizar los pasos 25 y 26 para crear un nuevo archivo que dará diseño a los Layout que van a contener los datos de los libros, en esta ocasión llamaremos al archivo fondo\_tarjeta.



29. El archivo contendrá lo siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
  <!--<solid android:color="#4DB6AC"/>-->
  <gradient
    android:startColor="#1258b6"
    android:endColor="#4b888d"/>
  <stroke android:width="2dp"
    android:color="#FFF"/>
  <corners android:radius="20dp"/>
</shape>
```

**Nota:** para que estos estilos puedan ser aplicados correctamente y sin problemas debemos dirigirnos a la carpeta **values** y luego **themes** después abrir el archivo llamado **themes.xml**



Una vez que se abra el archivo pondremos la siguiente instrucción subrayada:



```
<resources xmlns:tools="http://schemas.android.com/tools">
  ⚡ <!-- Base application theme. -->
  <style name="Base.Theme.DigitalBook" parent="Theme.AppCompat">
    <!-- Customize your light theme here. -->
    <!-- <item name="colorPrimary">@color/my_light_primary</item> -->
  </style>

  <style name="Theme.DigitalBook" parent="Base.Theme.DigitalBook" />
</resources>
```

30. Ahora debemos crear el archivo xml contendrá los datos de los libros, crearemos el archivo llamado **item\_libro.xml**, dentro de la carpeta **layout** el cual contendrá el siguiente diseño.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    android:padding="16dp">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="126dp"
        android:layout_marginLeft="10dp"
        android:layout_marginTop="10dp"
        android:background="@drawable/fondo_tarjeta"
        android:orientation="vertical">
        <LinearLayout
            android:layout_marginTop="10dp"
            android:layout_marginLeft="10dp"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">
            <TextView
                android:id="@+id/text_view_titulo"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Título"
                android:textColor="@color/white"
                android:textSize="18sp"
                android:textStyle="bold" />
            <TextView
                android:id="@+id/text_view_autores"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:text="Autores"
```

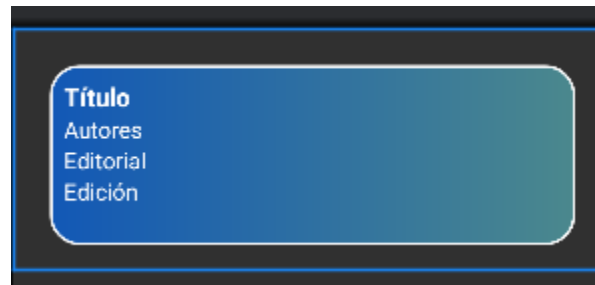


```

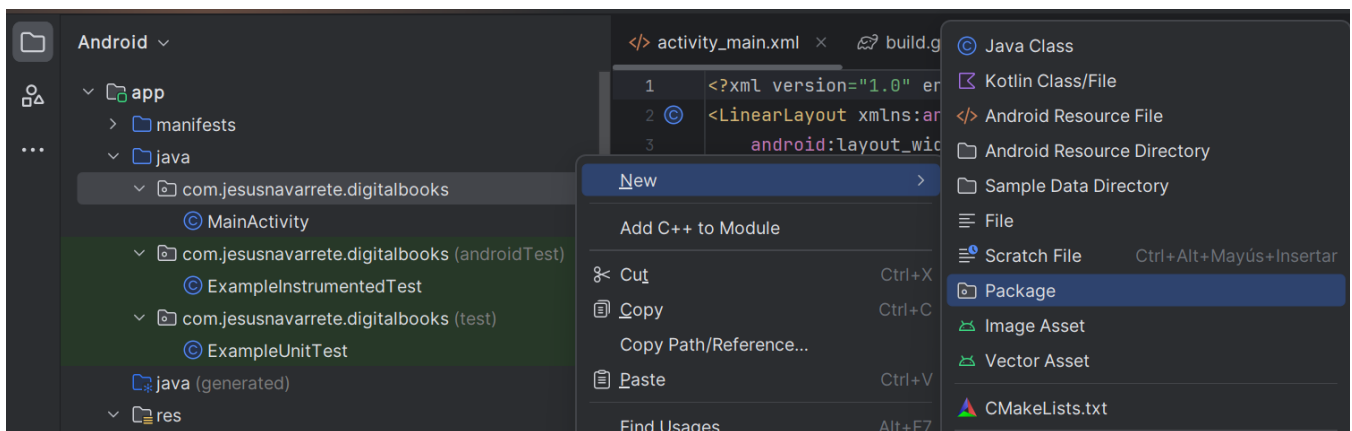
        android:textColor="@color/white"
        android:textSize="16sp" />
    <TextView
        android:id="@+id/text_view_editorial"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Editorial"
        android:textColor="@color/white"
        android:textSize="16sp" />
    <TextView
        android:id="@+id/text_view_edicion"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Edición"
        android:textColor="@color/white"
        android:textSize="16sp" />
</LinearLayout>
</LinearLayout>
</RelativeLayout>

```

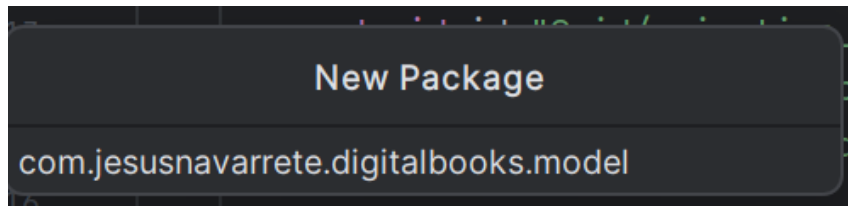
Previsualización del diseño:



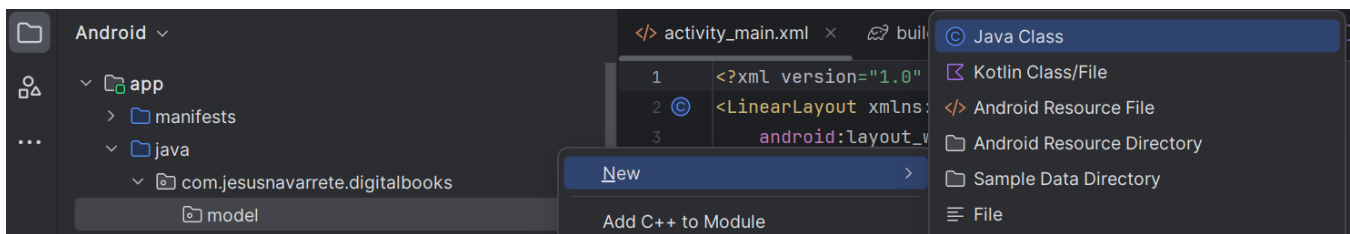
31. Ahora vamos a crear nuestro directorio **model** en el que después crearemos clases para el uso y conexión de la API, lo primero que haremos será dar click derecho en la primer carpeta del directorio java, después seleccionaremos **New** y por ultimo **Package** .



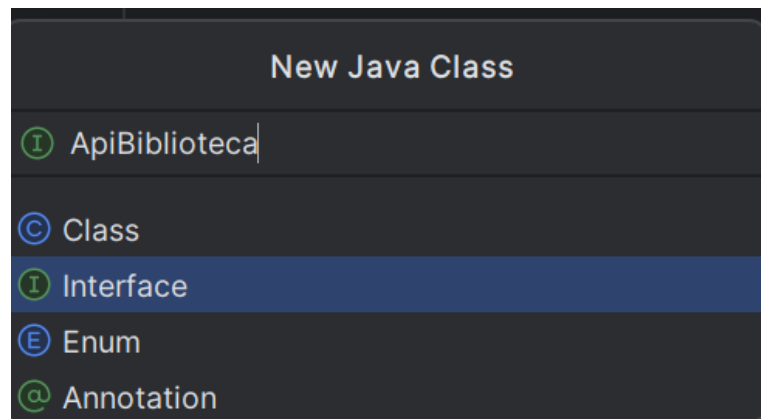
32. Se abrirá la siguiente ventana y escribiremos al final **model**, después damos **ENTER**.



33. Ahora en nuestra nueva carpeta **model** , nuevamente daremos click derecho , después **New** y ahora seleccionaremos la opción **Java Class**.



34. Se abrirá la siguiente ventana y escribiremos **ApiBiblioteca**, seleccionaremos la opción **interface** y daremos **ENTER**.







## Explicación de su creación y contenido:

Esta clase es el controlador de la API REST. Define los endpoints que gestionan las peticiones HTTP para realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre los libros.

```
package com.jesusnavarrete.digitalbook.model;

import com.jesusnavarrete.digitalbook.model.DatosBiblioApi;
import java.util.List;

import retrofit2.Call;
import retrofit2.http.Body;
import retrofit2.http.DELETE;
import retrofit2.http.GET;
import retrofit2.http.POST;
import retrofit2.http.PUT;
import retrofit2.http.Path;

public interface ApiBiblioteca {
    @GET("biblioapi/ver-todoslibros")
    Call<List<DatosBiblioApi>> getVerTodosLibros();

    @GET("biblioapi/ver-libro/{idLibro}")
    Call<DatosBiblioApi> getVerLibro(@Path("idLibro") long id);

    @POST("biblioapi/guardar-libro")
    Call<DatosBiblioApi> postGuardarLibro(@Body DatosBiblioApi book);

    @DELETE("biblioapi/eliminar-libro/{idLibro}")
    Call<Void> deleteBorraLibro(@Path("idLibro") long id);

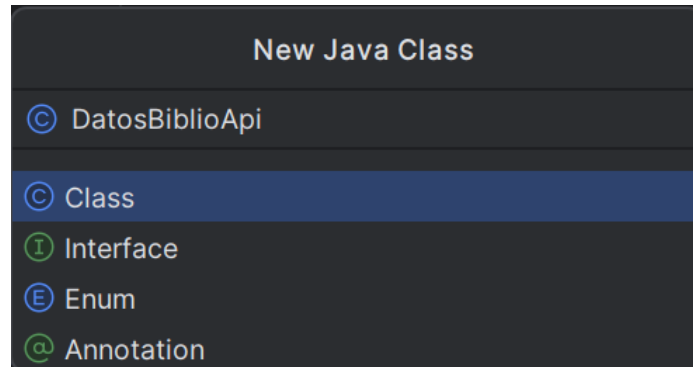
    @PUT("biblioapi/actualizar-libro/{idLibro}")
    Call<DatosBiblioApi> actualizarLibro(@Path("idLibro") long idLibro, @Body
    DatosBiblioApi libro);
}
```

## Aspectos importantes:

- **@RestController:** Indica que esta clase es un controlador de REST y que sus métodos manejan solicitudes web.
- **@RequestMapping("biblioapi"):** Define la URL base para los endpoints.
- **@Autowired:** Inyecta la dependencia BiblioApiService para usar sus métodos.
- **Métodos @GetMapping, @PostMapping, @PutMapping, @DeleteMapping:** Definen los endpoints para las operaciones CRUD.



35. Nuevamente crearemos una nueva clase dentro de nuestra carpeta **model**, sin embargo ya no será de tipo interfaz y le llamaremos **DatosBiblioApi**:



### Explicación de su creación y contenido:

Esta clase representa el modelo de datos para un libro en la biblioteca. Es una entidad que se utilizará para mapear los datos de la base de datos.

```
package com.jesunavarrete.digitalbook.model;

public class DatosBiblioApi {
    private long id;
    private String titulo;
    private String autores;
    private String editorial;
    private String edicion;

    public DatosBiblioApi(String titulo, String autores, String editorial, String
edicion) {
        this.titulo = titulo;
        this.autores = autores;
        this.editorial = editorial;
        this.edicion = edicion;
    }

    // Getters y Setters
    public long getId() {
        return id;
    }

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

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }
}
```

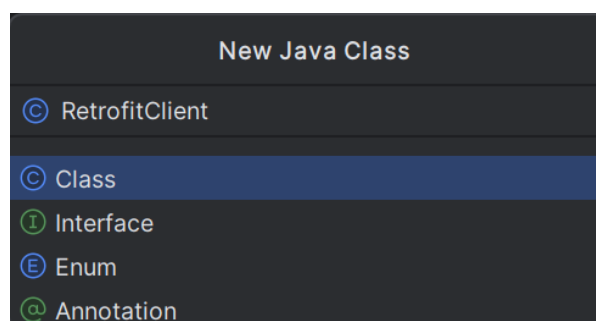


```
}  
  
public String getAutores() {  
    return autores;  
}  
  
public void setAutores(String autores) {  
    this.autores = autores;  
}  
  
public String getEditorial() {  
    return editorial;  
}  
  
public void setEditorial(String editorial) {  
    this.editorial = editorial;  
}  
  
public String getEdicion() {  
    return edicion;  
}  
  
public void setEdicion(String edicion) {  
    this.edicion = edicion;  
}  
}
```

## Aspectos importantes:

- **@Entity:** Anotación que marca esta clase como una entidad de JPA (Java Persistence API).
- **@Id:** Marca el campo id como la clave primaria.
- **@GeneratedValue(strategy = GenerationType.IDENTITY):** Indica que el valor de id se genera automáticamente.
- Los atributos (id, titulo, autores, editorial, edicion) representan las columnas de la tabla en la base de datos.
- Los métodos getter y setter permiten acceder y modificar los atributos.

36. Ahora debemos crear nuevamente una clase llamada RetrofitClient dentro de la carpeta model al igual que como lo hicimos con la clase anterior.



## Explicación de su creación y contenido:

Esta clase proporciona una instancia de Retrofit configurada para realizar solicitudes HTTP.

```
package com.jesusnavarrete.digitalbook.model;

import retrofit2.Retrofit;
import retrofit2.converter.gson.GsonConverterFactory;

2 usages
public class RetrofitClient {
    3 usages
    private static Retrofit retrofit = null;

    1 usage
    private static final String BASE_URL = "http://10.0.2.2:8080/";

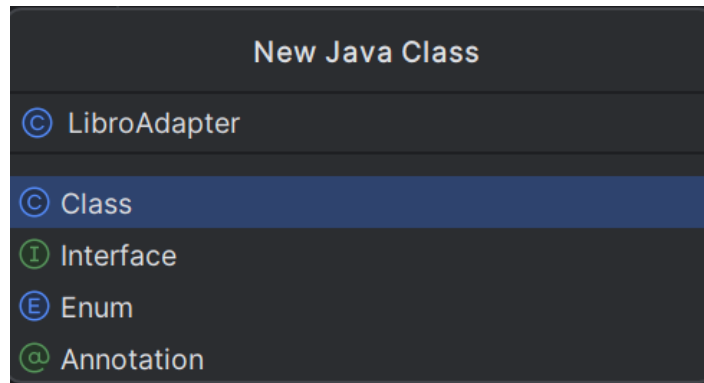
    1 usage
    public static Retrofit getClient(String url) {
        if (retrofit == null) {
            retrofit = new Retrofit.Builder()
                .baseUrl(BASE_URL)
                .addConverterFactory(GsonConverterFactory.create())
                .build();
        }
        return retrofit;
    }
}
```

## Aspectos importantes:

- **getClient(String baseUrl):** Método para obtener una instancia de Retrofit con la URL base proporcionada.



37. Nuevamente crearemos una nueva clase esta vez fuera de la carpeta model , y se llamara LibroAdapter.



### Explicación de su creación y contenido:

Esta clase será el adaptador para el RecyclerView, que maneja la visualización de la lista de libros.

```
package com.jesusnavarrete.digitalbook;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.RecyclerView;

import com.jesusnavarrete.digitalbook.model.DatosBiblioApi;

import java.util.List;

public class LibroAdapter extends
RecyclerView.Adapter<LibroAdapter.LibroViewHolder> {

    private List<DatosBiblioApi> libros;

    public LibroAdapter(List<DatosBiblioApi> libros) {
        this.libros = libros;
    }

    @NonNull
    @Override
    public LibroViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        View view =
LayoutInflater.from(parent.getContext()).inflate(R.layout.item_libro, parent,
false);
        return new LibroViewHolder(view);
    }

    @Override
```



```
public void onBindViewHolder(@NonNull LibroViewHolder holder, int position) {
    DatosBiblioApi libro = libros.get(position);

    holder.textViewTitulo.setText(libro.getTitulo());
    holder.textViewAutores.setText(libro.getAutores());
    holder.textViewEditorial.setText(libro.getEditorial());
    holder.textViewEdicion.setText(libro.getEdicion());
}

@Override
public int getItemCount() {
    return libros.size();
}

public void setLibros(List<DatosBiblioApi> libros) {
    this.libros = libros;
    notifyDataSetChanged();
}

public static class LibroViewHolder extends RecyclerView.ViewHolder {
    TextView textViewTitulo, textViewAutores, textViewEditorial,
textViewEdicion;

    public LibroViewHolder(@NonNull View itemView) {
        super(itemView);
        textViewTitulo = itemView.findViewById(R.id.text_view_titulo);
        textViewAutores = itemView.findViewById(R.id.text_view_autores);
        textViewEditorial = itemView.findViewById(R.id.text_view_editorial);
        textViewEdicion = itemView.findViewById(R.id.text_view_edicion);
    }
}
```

Aspectos importantes:

**LibroAdapter:** Extiende RecyclerView.Adapter y maneja la creación y vinculación de las vistas para cada libro.

**onCreateViewHolder:** Infla la vista del elemento del libro.

**onBindViewHolder:** Vincula los datos del libro a las vistas.

**getItemCount:** Devuelve el número de libros en la lista.

**LibroViewHolder:** Contiene las vistas individuales para cada atributo del libro.



38. Ahora en la clase que habíamos creado mucho antes llamada `main_home` añadiremos código para manejar la interfaz de usuario y realizar las operaciones CRUD utilizando Retrofit para interactuar con la API REST.

```
package com.jesusnavarrete.digitalbook;

import androidx.appcompat.app.AppCompatActivity;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.Toast;

import com.jesusnavarrete.digitalbook.model.ApiBiblioteca;
import com.jesusnavarrete.digitalbook.model.RetrofitClient;
import com.jesusnavarrete.digitalbook.model.DatosBiblioApi;

import java.util.ArrayList;
import java.util.List;

import retrofit2.Call;
import retrofit2.Callback;
import retrofit2.Response;
import retrofit2.Retrofit;

public class main_home extends AppCompatActivity {
    private ApiBiblioteca apiService;
    private RecyclerView recyclerView;
    private LibroAdapter libroAdapter;
    private EditText editTextBookId, editTextTitulo, editTextAutores,
    editTextEditorial, editTextEdicion, editTextBookIdDelete;
    private List<DatosBiblioApi> libroList;
    private EditText editTextBookId2, editTextTitulo2, editTextAutores2,
    editTextEditorial2, editTextEdicion2;

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

        recyclerView = findViewById(R.id.recycler_view);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
        libroList = new ArrayList<>();
        libroAdapter = new LibroAdapter(libroList);
        recyclerView.setAdapter(libroAdapter);

        editTextBookId = findViewById(R.id.edit_text_book_id);
        editTextBookIdDelete = findViewById(R.id.edit_text_delete_book_id);
    }
}
```



```
editTextTitulo = findViewById(R.id.edit_text_titulo);
editTextAutores = findViewById(R.id.edit_text_autores);
editTextEditorial = findViewById(R.id.edit_text_editorial);
editTextEdicion = findViewById(R.id.edit_text_edicion);

Button buttonSearchBook = findViewById(R.id.button_search_book);
Button buttonGetBooks = findViewById(R.id.button_get_books);
Button buttonSaveBook = findViewById(R.id.button_save_book);
Button buttonDeleteBook= findViewById(R.id.button_delete_book);

Button buttonUpdateBook = findViewById(R.id.button_update_book);

LinearLayout myLayout = findViewById(R.id.libros);

Button buscarLibs= findViewById(R.id.buscar);
LinearLayout myLayout2 = findViewById(R.id.buscarlibro);
LinearLayout myLayout5 = findViewById(R.id.hola);
LinearLayout myLayout8 = findViewById(R.id.hola2);
LinearLayout myLayout6 = findViewById(R.id.hola3);
LinearLayout myLayout7 = findViewById(R.id.hola4);
LinearLayout myLayout9 = findViewById(R.id.libAc);

Button agregarLibs =findViewById(R.id.agregar);
LinearLayout myLayout3 = findViewById(R.id.agregarLibros);

Button eliminarLibs =findViewById(R.id.eliminar);
LinearLayout myLayout4 = findViewById(R.id.eliminarLibro);

Button actualizarLibros =findViewById(R.id.actualizar);
LinearLayout myLayoutAc = findViewById(R.id.actualizacion);

editTextBookId2 = findViewById(R.id.edit_text_book_id2);
editTextTitulo2 = findViewById(R.id.edit_text_titulo2);
editTextAutores2 = findViewById(R.id.edit_text_autores2);
editTextEditorial2 = findViewById(R.id.edit_text_editorial2);
editTextEdicion2 = findViewById(R.id.edit_text_edicion2);

Retrofit retrofit = RetrofitClient.getClient("http://10.0.2.2:8080/");
apiService = retrofit.create(ApiBiblioteca.class);
buttonGetBooks.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        // Cambia la visibilidad del layout cuando el botón sea presionado

        if (myLayout8.getVisibility() == View.GONE) {
            myLayout8.setVisibility(View.VISIBLE);
        }
    }
});
```





```
    } else {
        myLayout8.setVisibility(View.GONE);
    }

    if (myLayout6.getVisibility() == View.GONE) {
        myLayout6.setVisibility(View.VISIBLE);
    } else {
        myLayout6.setVisibility(View.GONE);
    }

    if (myLayout7.getVisibility() == View.GONE) {
        myLayout7.setVisibility(View.VISIBLE);
    } else {
        myLayout7.setVisibility(View.GONE);
    }

    if (myLayout9.getVisibility() == View.GONE) {
        myLayout9.setVisibility(View.VISIBLE);
    } else {
        myLayout9.setVisibility(View.GONE);
    }
    getBooks();
}

});

buscarLibs.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        // Cambia la visibilidad del layout cuando el botón sea presionado

        if (myLayout2.getVisibility() == View.GONE) {
            myLayout2.setVisibility(View.VISIBLE);
        } else {
            myLayout2.setVisibility(View.GONE);
        }

        if (myLayout5.getVisibility() == View.GONE) {
            myLayout5.setVisibility(View.VISIBLE);
        } else {
            myLayout5.setVisibility(View.GONE);
        }

        if (myLayout6.getVisibility() == View.GONE) {
            myLayout6.setVisibility(View.VISIBLE);
        } else {
            myLayout6.setVisibility(View.GONE);
        }

        if (myLayout7.getVisibility() == View.GONE) {
            myLayout7.setVisibility(View.VISIBLE);
        } else {
            myLayout7.setVisibility(View.GONE);
        }

        if (myLayout9.getVisibility() == View.GONE) {
            myLayout9.setVisibility(View.VISIBLE);
        } else {
            myLayout9.setVisibility(View.GONE);
        }
    }
});
```



```
}

}

});

agregarLibs.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        // Cambia la visibilidad del layout cuando el botón sea presionado
        if (myLayout3.getVisibility() == View.GONE) {
            myLayout3.setVisibility(View.VISIBLE);
        } else {
            myLayout3.setVisibility(View.GONE);
        }

        if (myLayout8.getVisibility() == View.GONE) {
            myLayout8.setVisibility(View.VISIBLE);
        } else {
            myLayout8.setVisibility(View.GONE);
        }

        if (myLayout5.getVisibility() == View.GONE) {
            myLayout5.setVisibility(View.VISIBLE);
        } else {
            myLayout5.setVisibility(View.GONE);
        }

        if (myLayout7.getVisibility() == View.GONE) {
            myLayout7.setVisibility(View.VISIBLE);
        } else {
            myLayout7.setVisibility(View.GONE);
        }

        if (myLayout9.getVisibility() == View.GONE) {
            myLayout9.setVisibility(View.VISIBLE);
        } else {
            myLayout9.setVisibility(View.GONE);
        }
    }
});

eliminarLibs.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        // Cambia la visibilidad del layout cuando el botón sea presionado
        if (myLayout4.getVisibility() == View.GONE) {
            myLayout4.setVisibility(View.VISIBLE);
        } else {
            myLayout4.setVisibility(View.GONE);
        }

        if (myLayout5.getVisibility() == View.GONE) {
            myLayout5.setVisibility(View.VISIBLE);
        }
    }
});
```



```
    } else {
        myLayout5.setVisibility(View.GONE);
    }

    if (myLayout8.setVisibility() == View.GONE) {
        myLayout8.setVisibility(View.VISIBLE);
    } else {
        myLayout8.setVisibility(View.GONE);
    }

    if (myLayout6.setVisibility() == View.GONE) {
        myLayout6.setVisibility(View.VISIBLE);
    } else {
        myLayout6.setVisibility(View.GONE);
    }

    if (myLayout9.setVisibility() == View.GONE) {
        myLayout9.setVisibility(View.VISIBLE);
    } else {
        myLayout9.setVisibility(View.GONE);
    }
}

});

actualizarLibros.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        // Cambia la visibilidad del layout cuando el botón sea presionado
        if (myLayoutAc.setVisibility() == View.GONE) {
            myLayoutAc.setVisibility(View.VISIBLE);
        } else {
            myLayoutAc.setVisibility(View.GONE);
        }

        if (myLayout4.setVisibility() == View.GONE) {
            myLayout4.setVisibility(View.VISIBLE);
        } else {
            myLayout4.setVisibility(View.GONE);
        }

        if (myLayout5.setVisibility() == View.GONE) {
            myLayout5.setVisibility(View.VISIBLE);
        } else {
            myLayout5.setVisibility(View.GONE);
        }

        if (myLayout8.setVisibility() == View.GONE) {
            myLayout8.setVisibility(View.VISIBLE);
        } else {
            myLayout8.setVisibility(View.GONE);
        }

        if (myLayout6.setVisibility() == View.GONE) {
            myLayout6.setVisibility(View.VISIBLE);
        } else {
            myLayout6.setVisibility(View.GONE);
        }
    }
});
```



```
        if (myLayout7.getVisibility() == View.GONE) {
            myLayout7.setVisibility(View.VISIBLE);
        } else {
            myLayout7.setVisibility(View.GONE);
        }

        if (myLayout4.getVisibility() == View.GONE) {
            myLayout4.setVisibility(View.VISIBLE);
        } else {
            myLayout4.setVisibility(View.GONE);
        }
    }
});

buttonSearchBook.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String bookIdStr = editTextBookId.getText().toString();
        if (!bookIdStr.isEmpty()) {
            long bookId = Long.parseLong(bookIdStr);
            getBookById(bookId);
        } else {
            Toast.makeText(main_home.this, "Ingrese un ID válido",
Toast.LENGTH_SHORT).show();
        }
    }
});

buttonSaveBook.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String titulo = editTextTitulo.getText().toString();
        String autores = editTextAutores.getText().toString();
        String editorial = editTextEditorial.getText().toString();
        String edicion = editTextEdicion.getText().toString();

        if (!titulo.isEmpty() && !autores.isEmpty() && !editorial.isEmpty()
&& !edicion.isEmpty()) {
            DatosBiblioApi newBook = new DatosBiblioApi(titulo, autores,
editorial, edicion);
            saveBook(newBook);
        } else {
            Toast.makeText(main_home.this, "Complete todos los campos",
Toast.LENGTH_SHORT).show();
        }
    }
});

buttonDeleteBook.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String bookIdStr = editTextBookIdDelete.getText().toString();
        if (!bookIdStr.isEmpty()) {
            long bookId = Long.parseLong(bookIdStr);
```



```
        deleteBook(bookId);
    } else {
        Toast.makeText(main_home.this, "Ingrese un ID válido",
Toast.LENGTH_SHORT).show();
    }
}
});

buttonUpdateBook.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String bookIdStr = editTextBookId2.getText().toString().trim();
        String titulo = editTextTitulo2.getText().toString().trim();
        String autores = editTextAutores2.getText().toString().trim();
        String editorial = editTextEditorial2.getText().toString().trim();
        String edicion = editTextEdicion2.getText().toString().trim();

        if (!bookIdStr.isEmpty() && !titulo.isEmpty() && !autores.isEmpty()
&& !editorial.isEmpty() && !edicion.isEmpty()) {
            long bookId = Long.parseLong(bookIdStr);
            DatosBiblioApi updatedBook = new DatosBiblioApi(titulo,
autores, editorial, edicion);
            updateBook(bookId, updatedBook);
        } else {
            Toast.makeText(main_home.this, "Complete todos los campos",
Toast.LENGTH_SHORT).show();
        }
    }
});

}

private void getBooks() {
    Call<List<DatosBiblioApi>> call = apiService.getVerTodosLibros();

    call.enqueue(new Callback<List<DatosBiblioApi>>() {
        @Override
        public void onResponse(Call<List<DatosBiblioApi>> call,
Response<List<DatosBiblioApi>> response) {
            if (!response.isSuccessful()) {
                Toast.makeText(main_home.this, "Code: " + response.code(),
Toast.LENGTH_SHORT).show();
                return;
            }

            List<DatosBiblioApi> books = response.body();
            libroList.clear();
            libroList.addAll(books);
            libroAdapter.notifyDataSetChanged();
        }

        @Override
        public void onFailure(Call<List<DatosBiblioApi>> call, Throwable t) {
            Toast.makeText(main_home.this, t.getMessage(),
Toast.LENGTH_SHORT).show();
        }
    });
}
```



```
    });  
}  
  
private void getBookById(long id) {  
    Call<DatosBiblioApi> call = apiService.getVerLibro(id);  
  
    call.enqueue(new Callback<DatosBiblioApi>() {  
        @Override  
        public void onResponse(Call<DatosBiblioApi> call,  
Response<DatosBiblioApi> response) {  
            if (!response.isSuccessful()) {  
                Toast.makeText(main_home.this, "Code: " + response.code(),  
Toast.LENGTH_SHORT).show();  
                return;  
            }  
  
            DatosBiblioApi book = response.body();  
            if (book != null) {  
                libroList.clear();  
                libroList.add(book);  
                libroAdapter.notifyDataSetChanged();  
            } else {  
                Toast.makeText(main_home.this, "Libro no encontrado",  
Toast.LENGTH_SHORT).show();  
            }  
        }  
  
        @Override  
        public void onFailure(Call<DatosBiblioApi> call, Throwable t) {  
            Toast.makeText(main_home.this, t.getMessage(),  
Toast.LENGTH_SHORT).show();  
        }  
    });  
}  
  
private void saveBook(DatosBiblioApi book) {  
    Call<DatosBiblioApi> call = apiService.postGuardarLibro(book);  
  
    call.enqueue(new Callback<DatosBiblioApi>() {  
        @Override  
        public void onResponse(Call<DatosBiblioApi> call,  
Response<DatosBiblioApi> response) {  
            if (!response.isSuccessful()) {  
                Toast.makeText(main_home.this, "Code: " + response.code(),  
Toast.LENGTH_SHORT).show();  
                return;  
            }  
  
            DatosBiblioApi savedBook = response.body();  
            if (savedBook != null) {  
                libroList.add(savedBook);  
                libroAdapter.notifyDataSetChanged();  
                Toast.makeText(main_home.this, "Libro guardado con éxito",  
Toast.LENGTH_SHORT).show();  
            }  
        }  
    })  
}
```



```
@Override
public void onFailure(Call<DatosBiblioApi> call, Throwable t) {
    Toast.makeText(main_home.this, t.getMessage(),
Toast.LENGTH_SHORT).show();
}

});
}

private void deleteBook(long id) {
    Call<Void> call = apiService.deleteBorraLibro(id);

    call.enqueue(new Callback<Void>() {
        @Override
        public void onResponse(Call<Void> call, Response<Void> response) {
            if (response.isSuccessful()) {
                Toast.makeText(main_home.this, "Libro eliminado con éxito",
Toast.LENGTH_SHORT).show();
                // Actualizar la lista de libros después de la eliminación
                getBooks();
            } else {
                Toast.makeText(main_home.this, "Error al eliminar el libro",
Toast.LENGTH_SHORT).show();
            }
        }

        @Override
        public void onFailure(Call<Void> call, Throwable t) {
            Toast.makeText(main_home.this, t.getMessage(),
Toast.LENGTH_SHORT).show();
        }
    });
}

private void updateBook(long id, DatosBiblioApi book) {
    Call<DatosBiblioApi> call = apiService.actualizarLibro(id, book);

    call.enqueue(new Callback<DatosBiblioApi>() {
        @Override
        public void onResponse(Call<DatosBiblioApi> call,
Response<DatosBiblioApi> response) {
            if (!response.isSuccessful()) {
                Toast.makeText(main_home.this, "Code: " + response.code(),
Toast.LENGTH_SHORT).show();
                return;
            }

            DatosBiblioApi updatedBook = response.body();
            if (updatedBook != null) {
                // Actualizar la lista de libros con los datos del libro
actualizado
                getBooks();
                Toast.makeText(main_home.this, "Libro actualizado con éxito",
Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(main_home.this, "Error al actualizar el libro",
Toast.LENGTH_SHORT).show();
            }
        }
    })
}
```

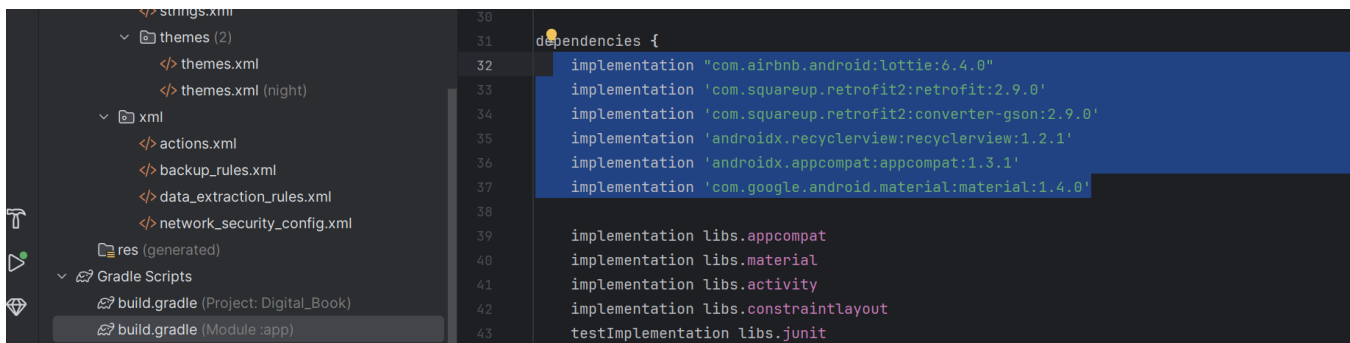


```
@Override
public void onFailure(Call<DatosBiblioApi> call, Throwable t) {
    Toast.makeText(main_home.this, t.getMessage(),
Toast.LENGTH_SHORT).show();
}
});
}
}
```

## Aspectos importantes:

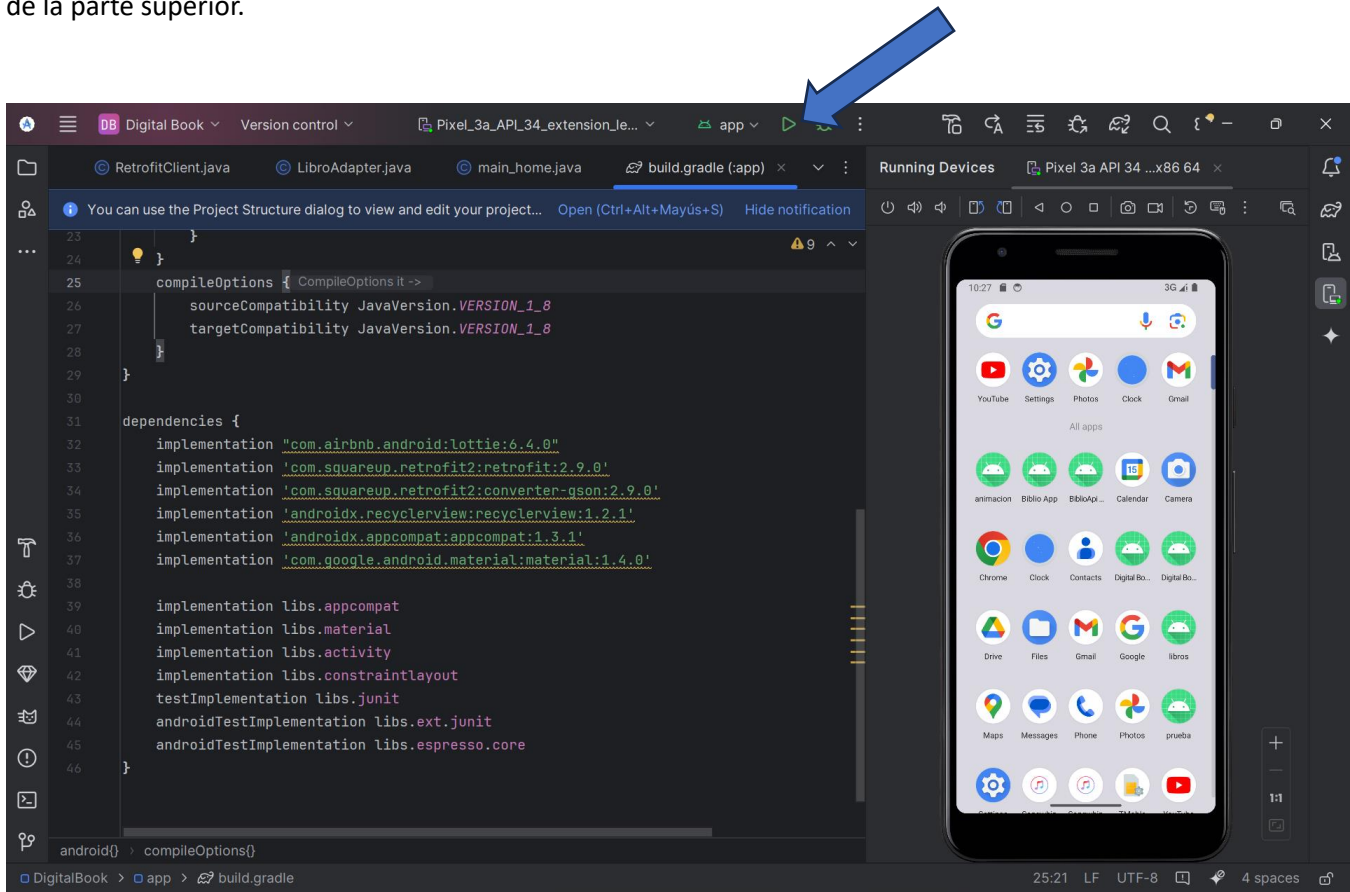
- **onCreate:** Método donde se inicializa la actividad. Se configura el RecyclerView, los EditText y los Button.
- **Métodos para manejar las operaciones CRUD:** getBooks, getBookById, saveBook, updateBook, deleteBook.
- Cada método utiliza Retrofit para realizar llamadas a la API y maneja las respuestas con enqueue para operaciones asíncronas.

39. Ahora nos dirigiremos a la sección **build.gradle (Module :app)** después añadiremos las siguientes dependencias subrayadas las cuales nos ayudaran a la implementación de las animaciones JSON, el poder trabajar con retrofit y con los objetos JSON de nuestra API.





40. Una vez realizado todo lo anterior podremos Compilar nuestra aplicación dando click en el botón de play de la parte superior.



Pantallas resultantes al compilar la aplicación.



Pantalla splash con  
duración de 10 segundos



Pantalla principal



## Funcionamiento:





## Conclusiones:

Trabajar en este proyecto ha sido una experiencia muy enriquecedora, tanto a nivel personal como profesional. He logrado crear mi propia API desde cero y desarrollar una aplicación móvil en Android Studio que la consume, lo cual es un gran logro.

Se comenzo con el diseño y desarrollo de la API, donde defini claramente qué datos y funcionalidades necesitaba. Crear estos endpoints robustos y seguros fue crucial para garantizar una comunicación eficaz entre el servidor y las aplicaciones que lo utilizan.

Luego, me enfoque en la aplicación móvil en Android Studio. Aquí, no solo me asegure de que la app tuviera un diseño atractivo y fácil de usar, sino también de que se integrara perfectamente con nuestra API. Fue un desafío interesante lograr que la app consumiera y presentara los datos de manera dinámica y fluida.