

UF 1 - Tarjeta felicitación



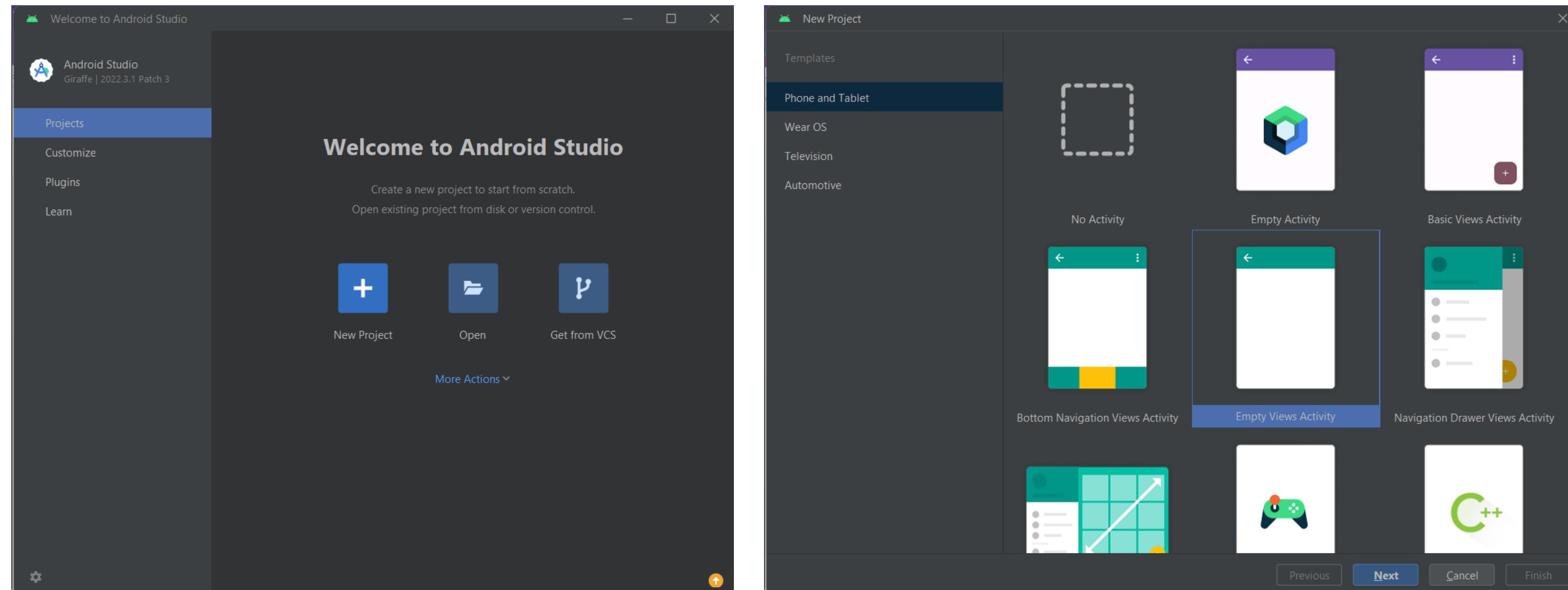
Contenido

- Instalación
- Ejecución en dispositivo físico
- Tarjeta de felicitación 1
- Tarjeta de felicitación 2
- Tarjeta de felicitación 3 - fuentes
- Tarjeta de felicitación 4 -música
- Animaciones básicas
- Tarjeta de felicitación 5 - animaciones
- Tarjeta de felicitación 6 - splash
- Cambio de activity

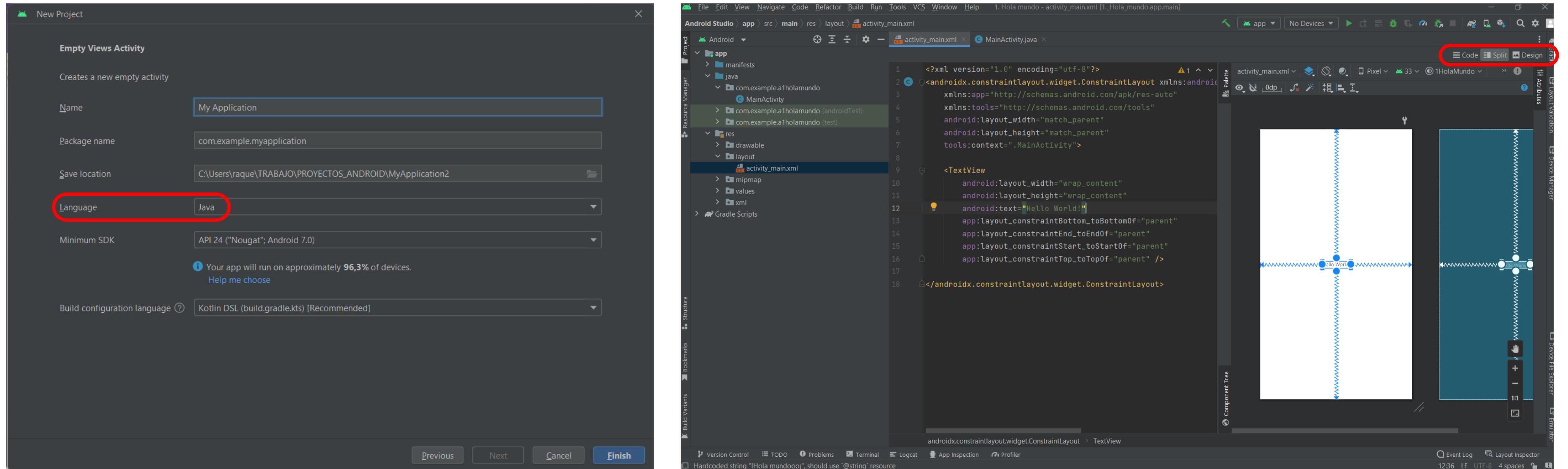
Instalación

Vamos a la web de Android y nos descargamos la aplicación para el SO que necesitemos: <https://developer.android.com/studio>

Una vez descargado crearemos nuestro primer proyecto:



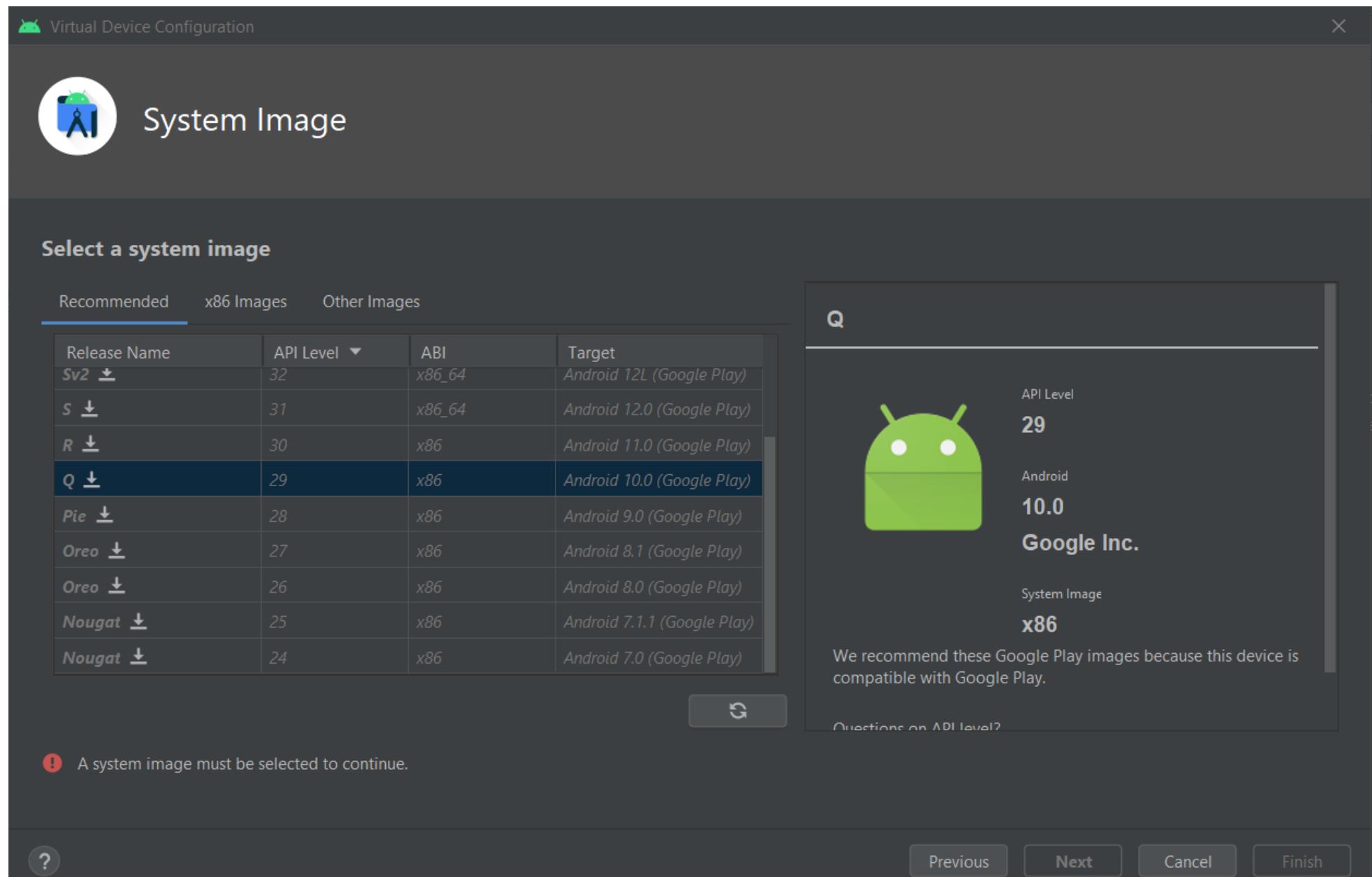
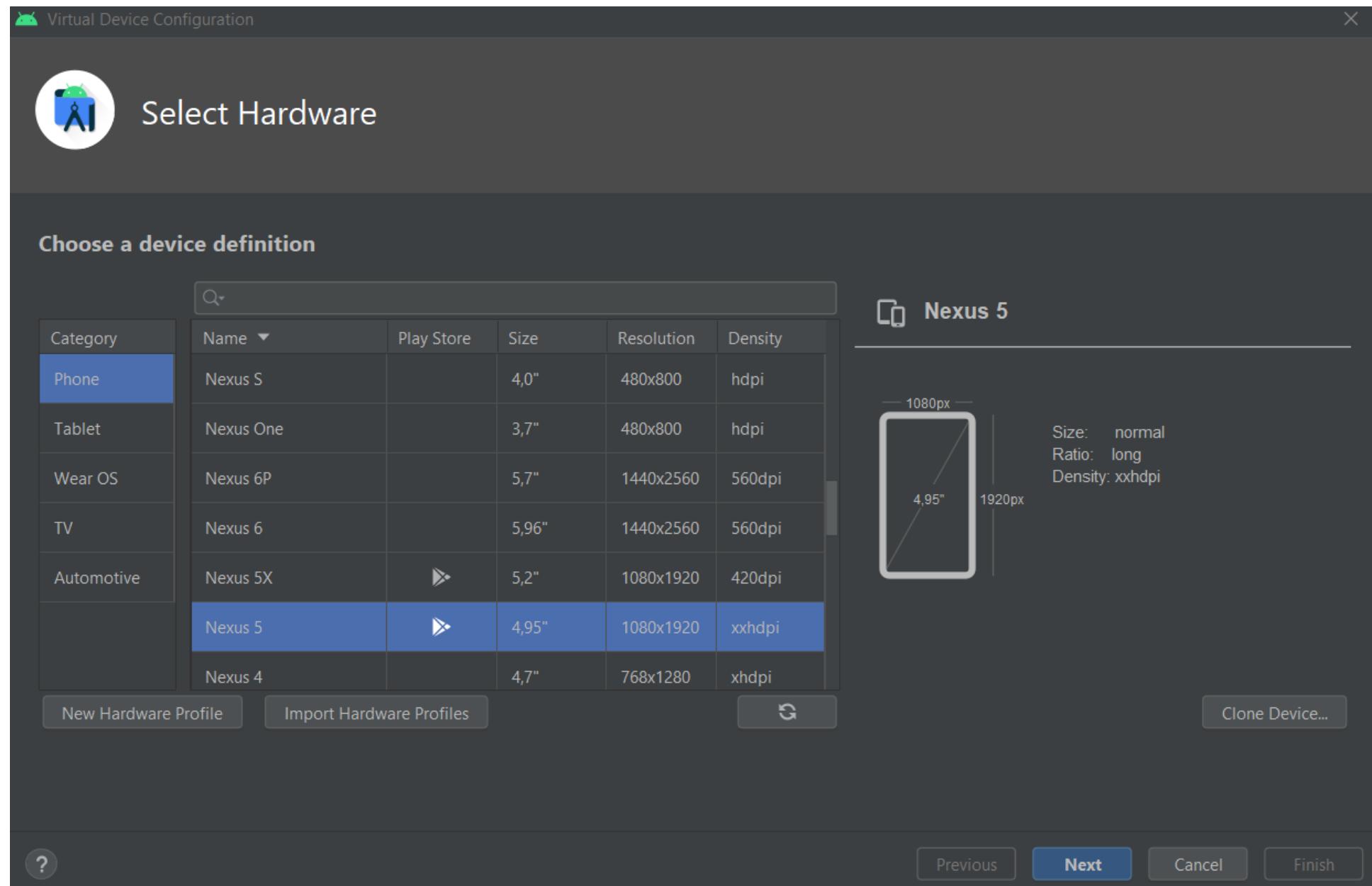
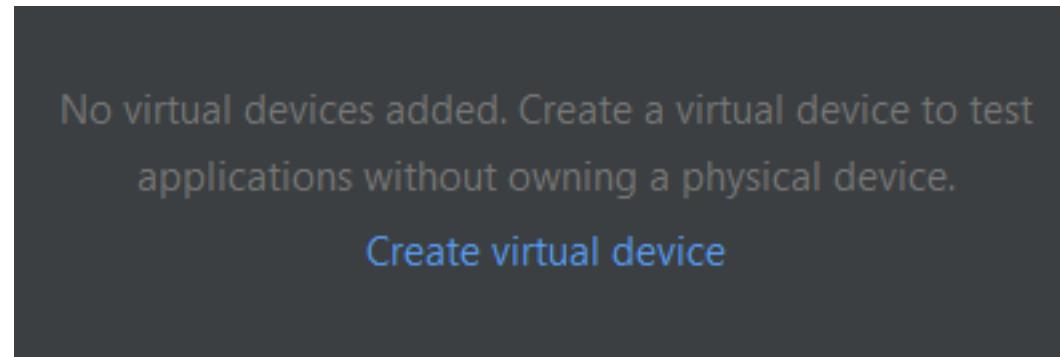
Instalación



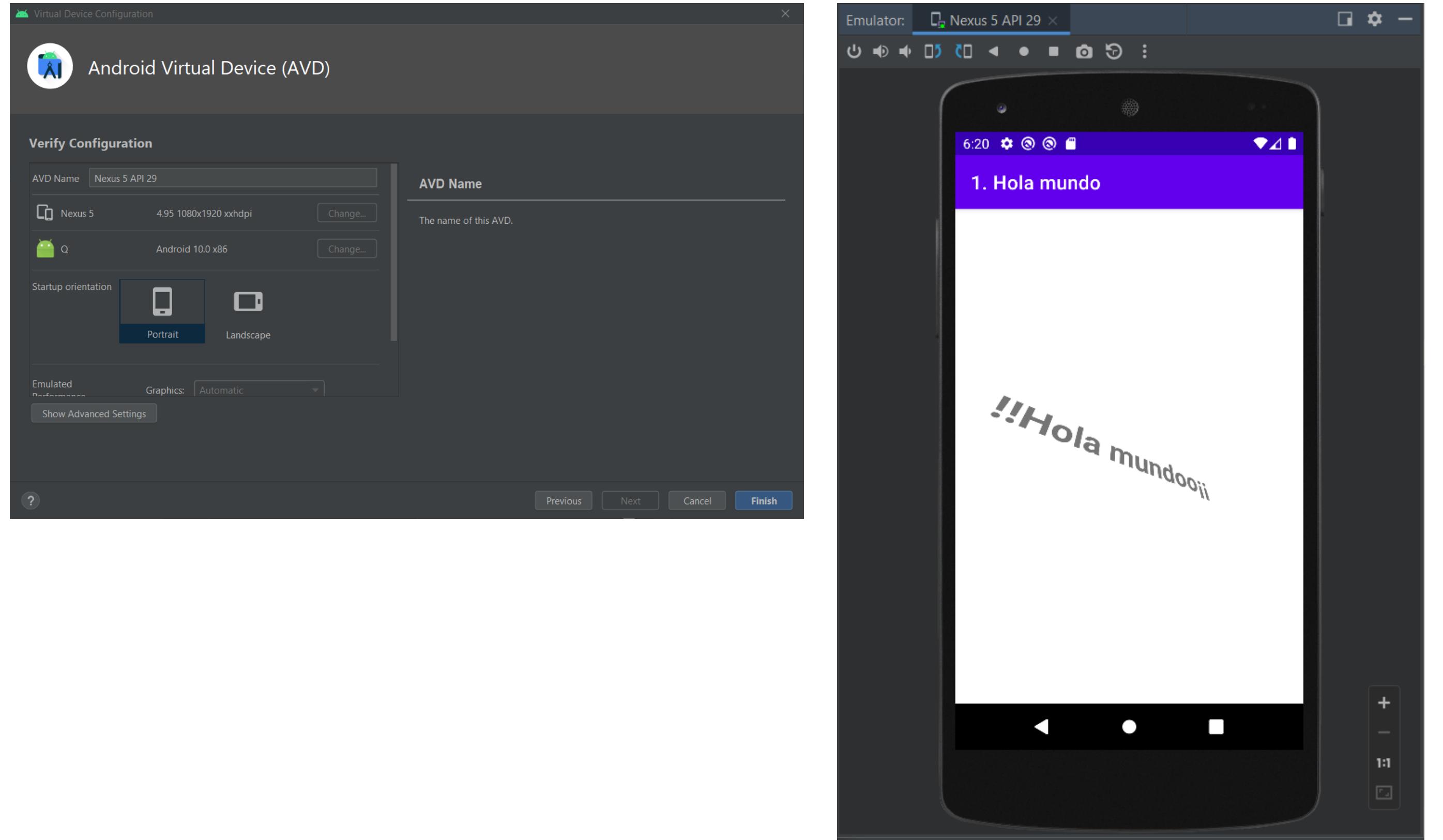
.xml

```
android:text="!!Hola mundo!!"
android:rotationX="38"
android:rotationY="33"
android:textSize="34sp"
android:textStyle="bold"
```

Ejecución

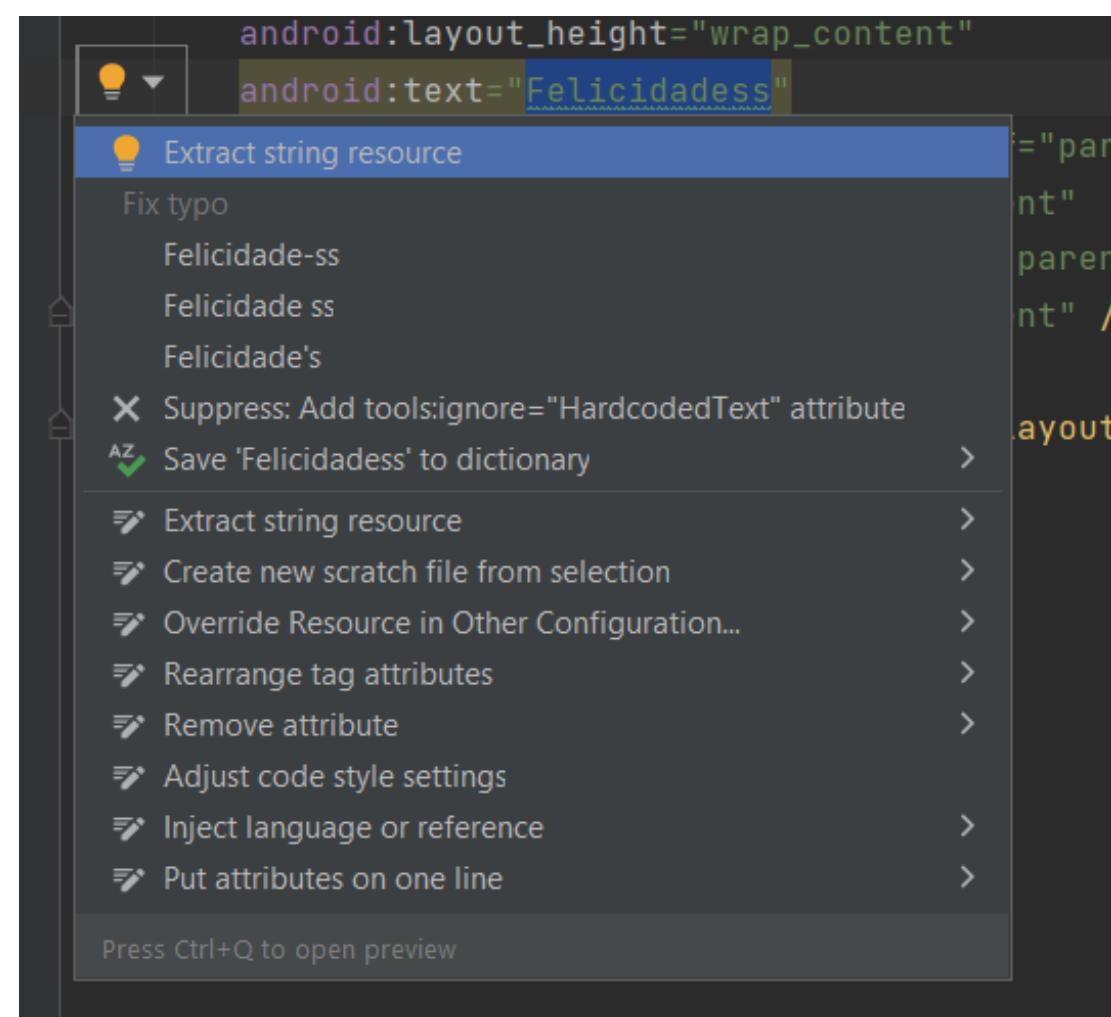
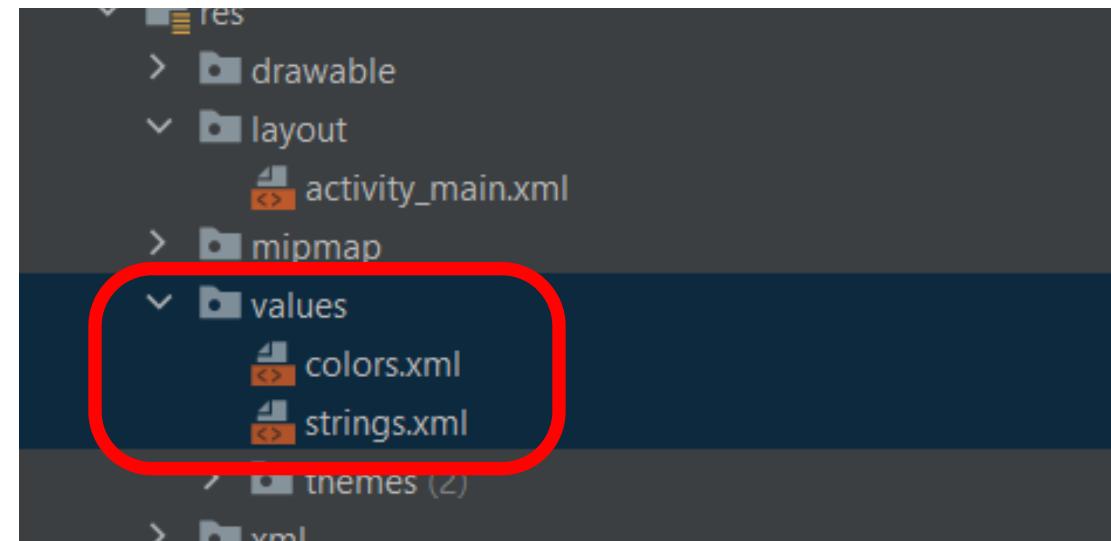


Ejecución

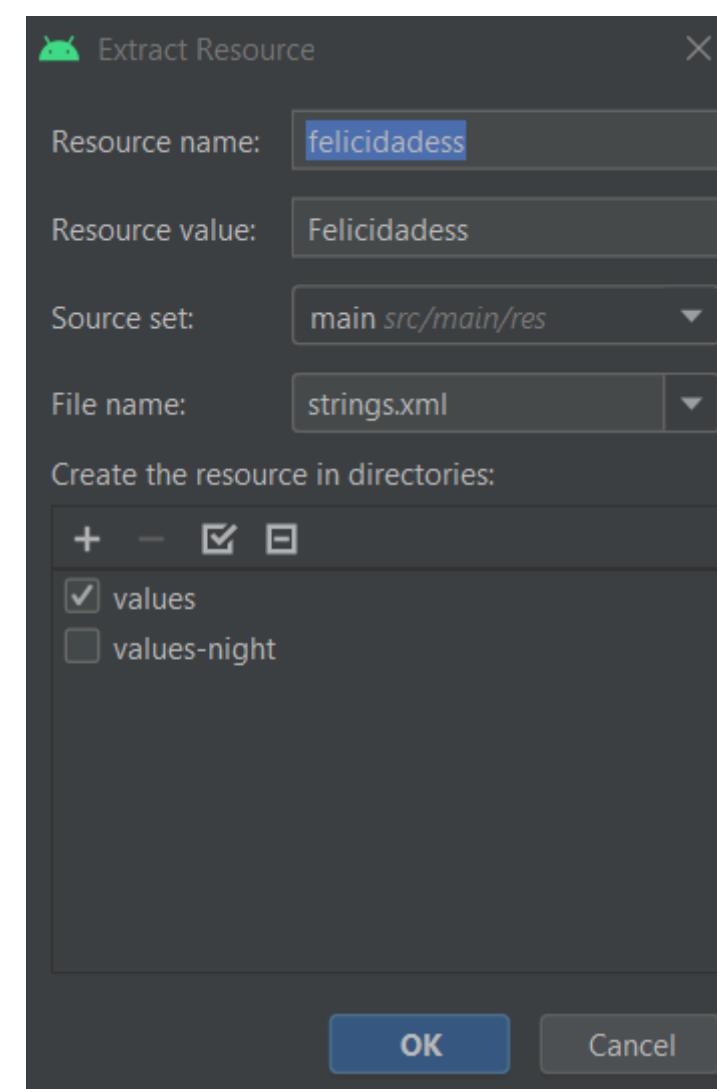


Tarjeta de felicitación 1

Android maneja tanto los colores como los strings de la siguiente manera:



Podemos encontrar 2 archivos en los que se localizarán los textos y colores que utilizará nuestra app, es una centralización ya que hace mas rápido los cambios de textos/colores.

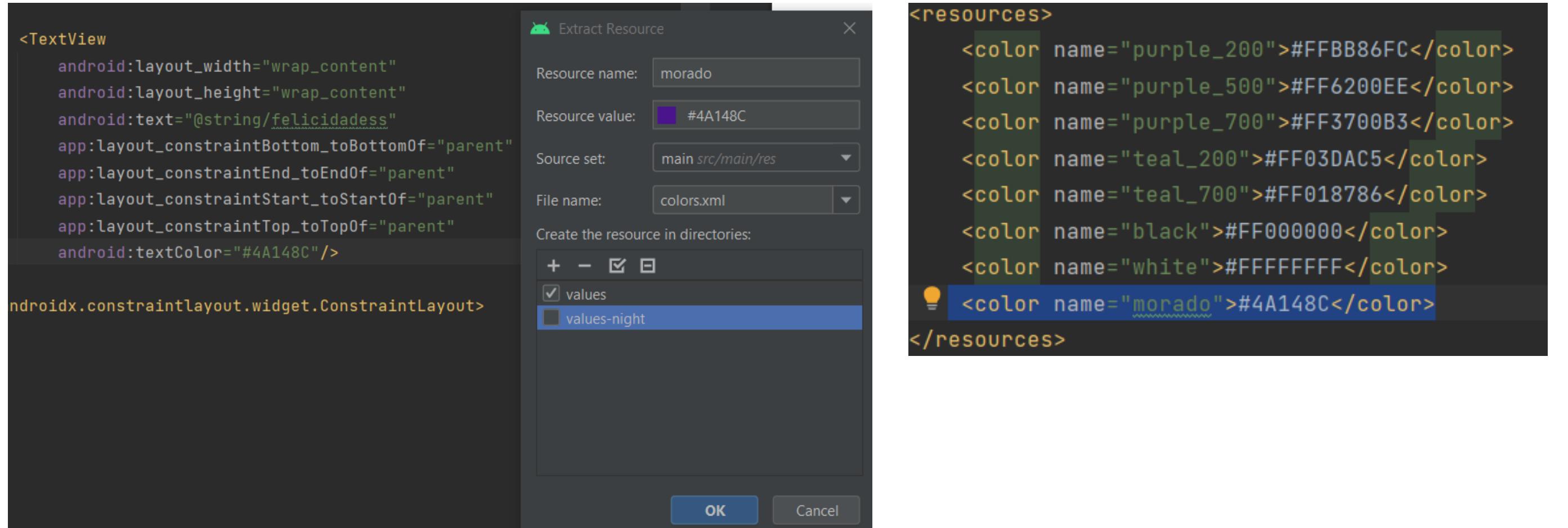


```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/felicidadess"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:layout_constraintTop_toTopOf="parent" />  
  
<resources>  
    <string name="app_name">Felicion Navidad</string>  
    <string name="felicidadess">Felicidadess</string>  
</resources>
```

Cuando le damos a ok lo que va a ocurrir es que donde antes estaba el texto ahora tendremos una referencia al valor de la variable que se localizará en el archivo **strings.xml**

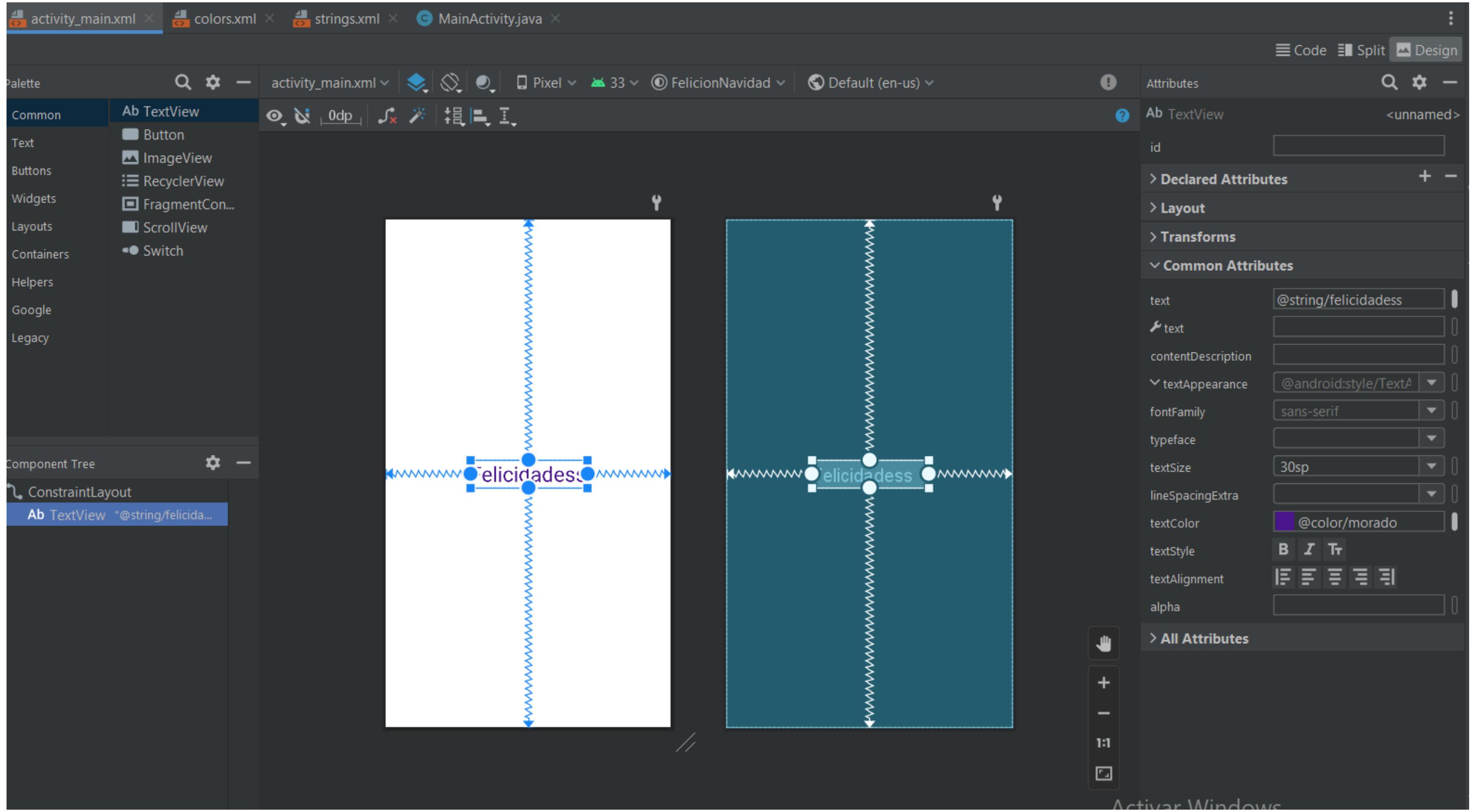
Tarjeta de felicitación 1

Pasa lo mismo con los colores:



Tarjeta de felicitación 1

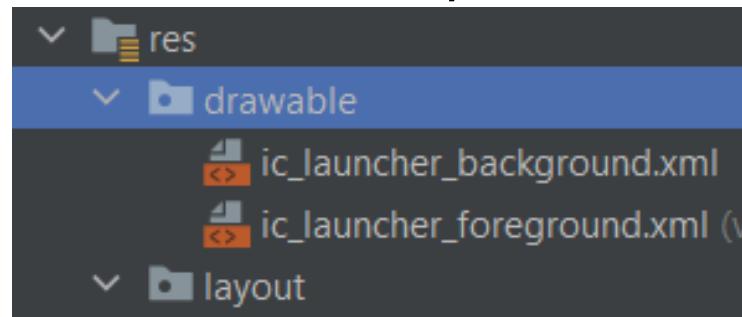
También podemos cambiar el tamaño, posición... podemos hacerlo desde la ventana de diseño:



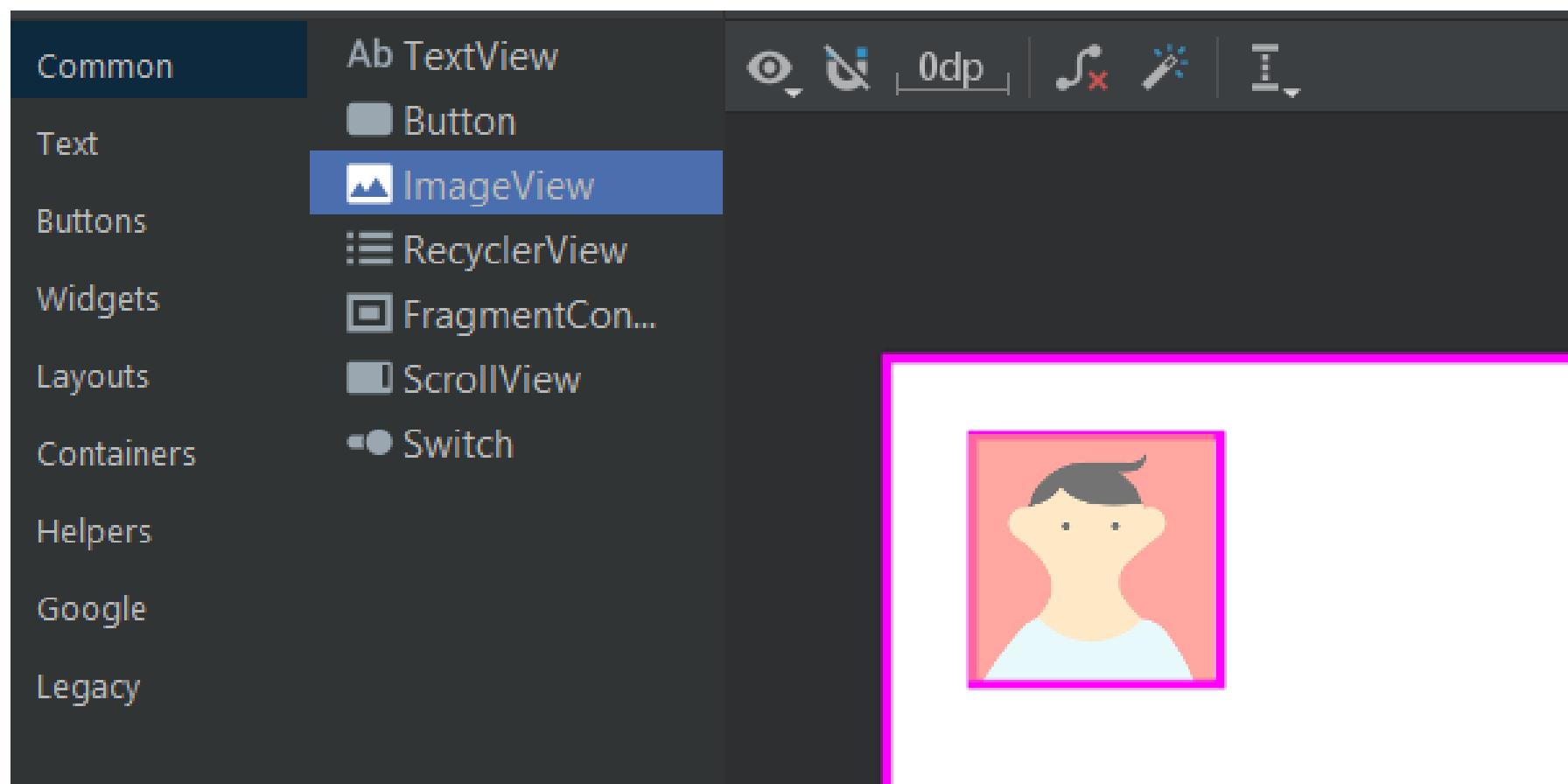
Tarjeta de felicitación 2

Ahora vamos a agregar un “fondo” a la aplicación, para eso necesitaremos una imagen.

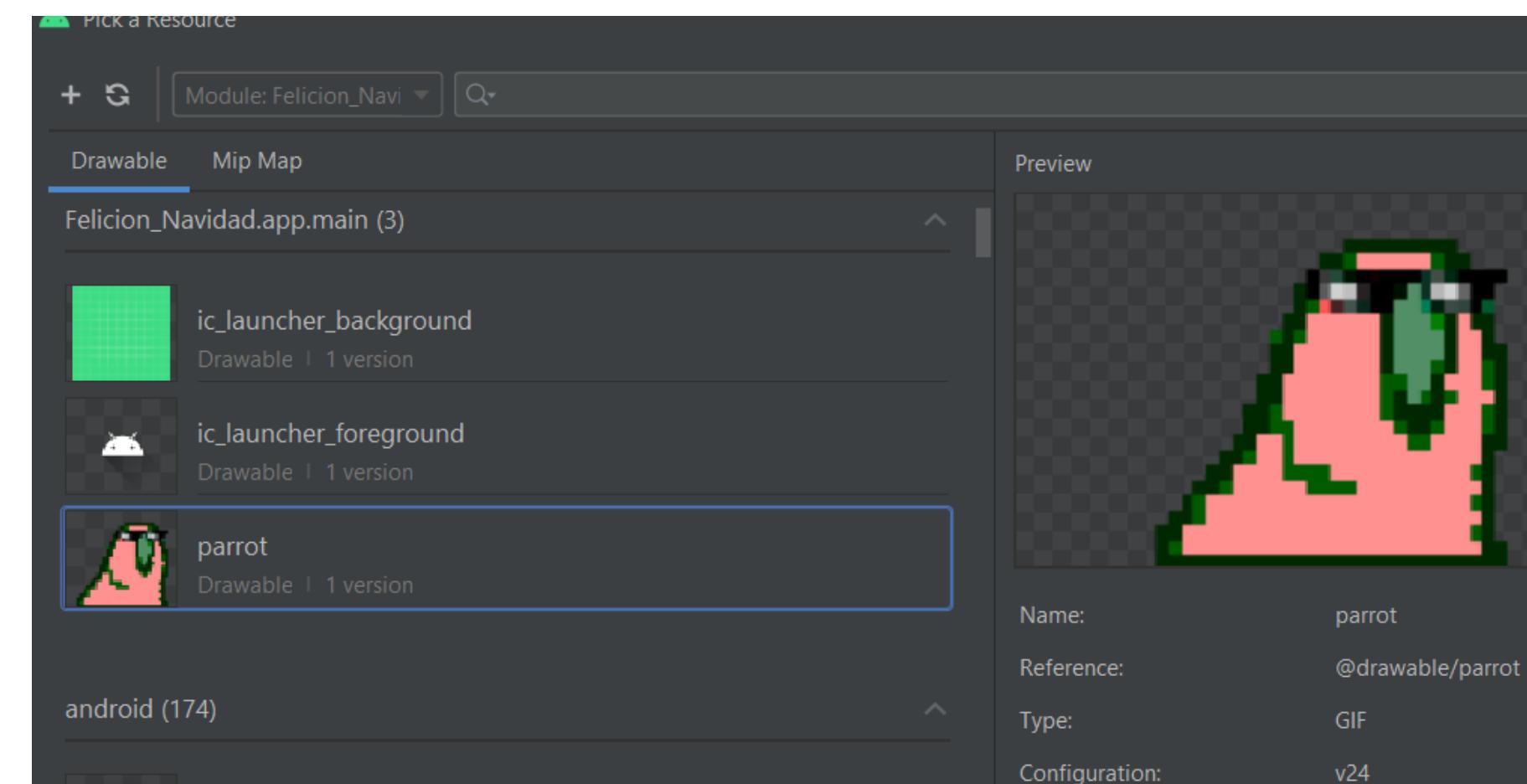
1. Vamos a la carpeta **drawable** y colocamos la foto.



2. Una vez añadida nos vamos a la pestaña de diseño y arrastramos la opción imageview:

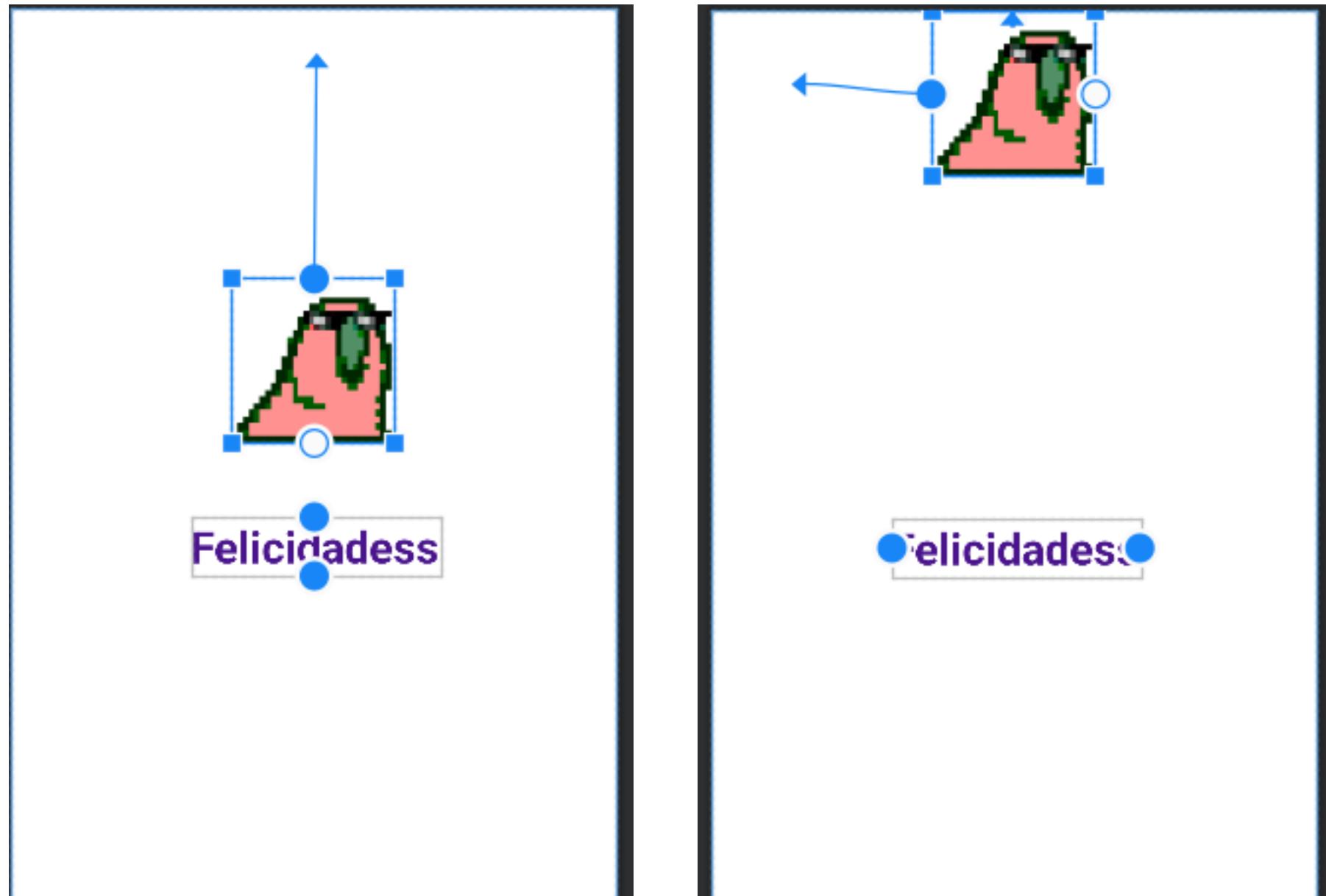


E indicamos cual queremos:

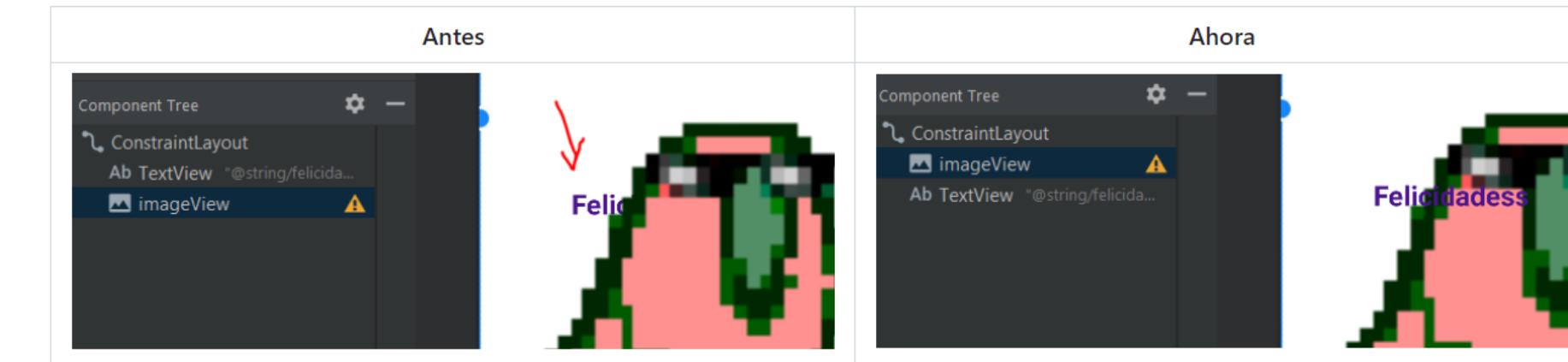


Tarjeta de felicitación 2

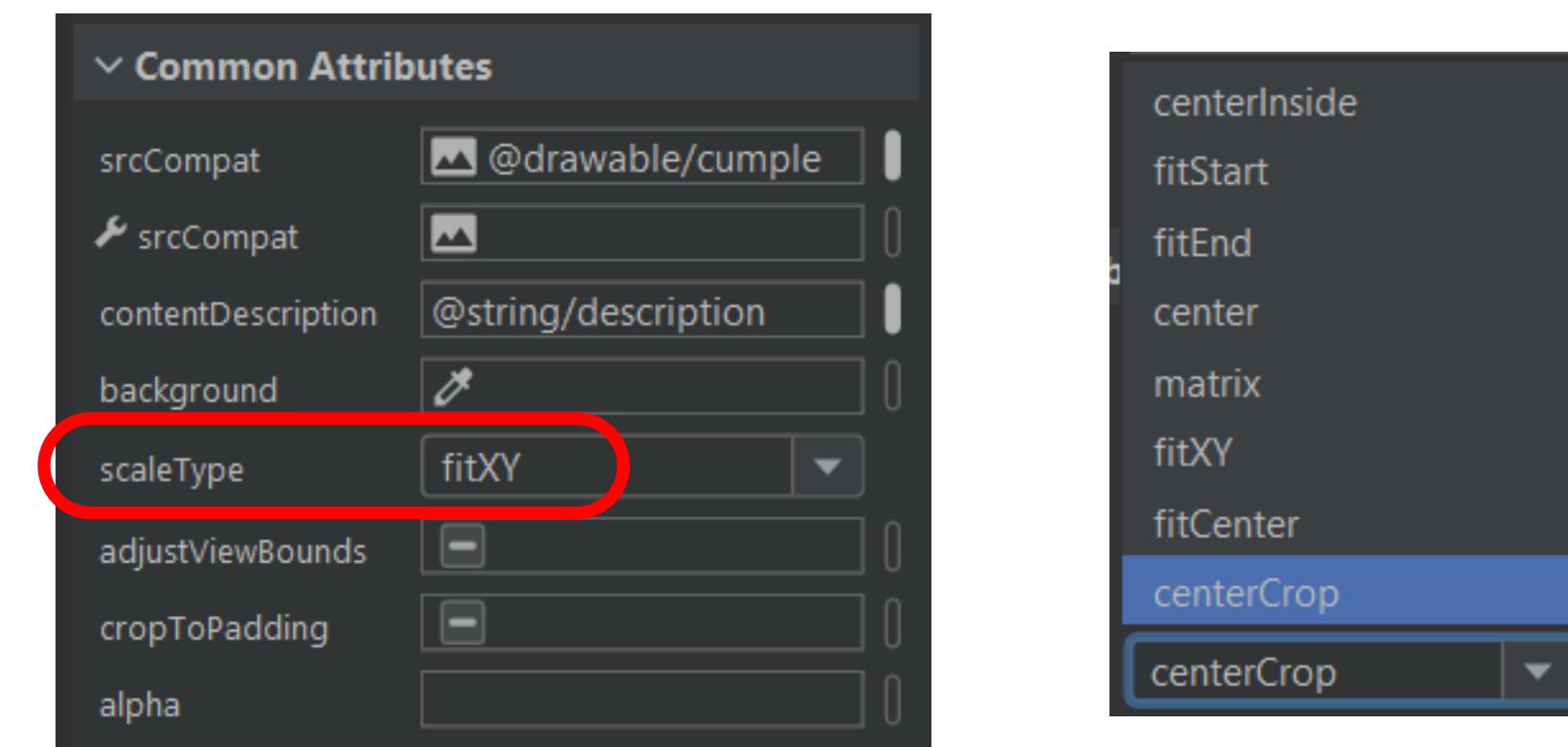
3. Una vez añadido tendremos que fijar 2 puntos de "referencia": uno en eje vertical y otro en horizontal:



Seguramente nuestro texto no se verá, tendremos que moverlo de posición!⚠



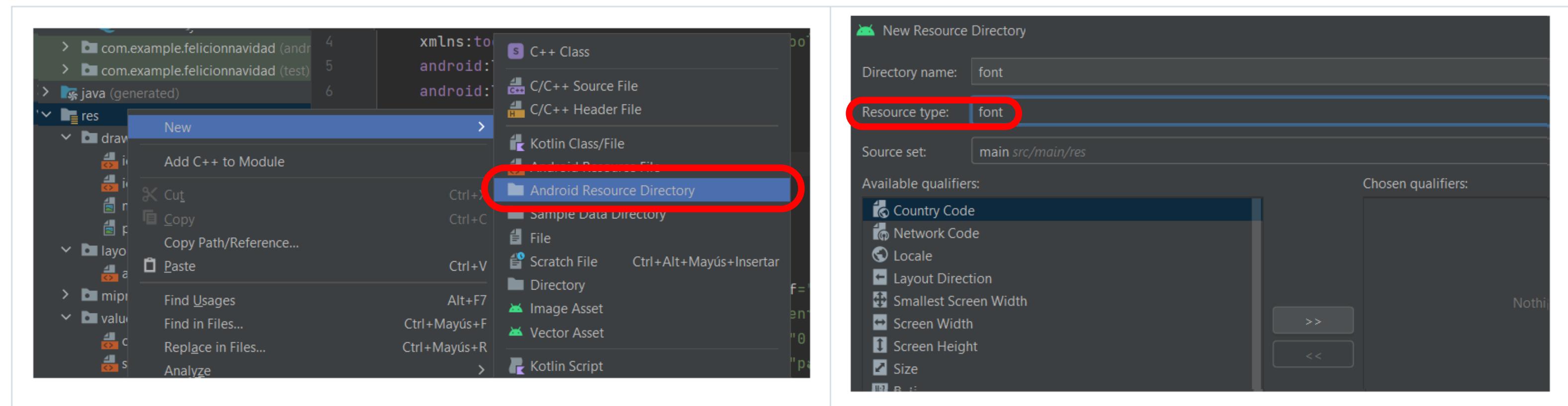
4. Por ultimo podemos utilizar estas opciones para posicionar nuestra foto:



Tarjeta de felicitación 3 - fuentes

En esta web: [freepik](#), hay muchos recursos como fotos que pueden ser útiles para crear la app, también se puede usar [dafont](#), para fuentes.

Para guardar las fuentes en nuestra app tendremos que crear una carpeta que se localice dentro de **res**:



⚠ El nombre del archivo .ttf que se añadirá a esta carpeta tiene que estar en minúsculas y sin espacios ⚠

Una vez añadido este archivo a la carpeta, en el xml, en el texto que deseamos cambiar, añadimos lo siguiente:

```
.xml
    android:fontFamily="@font/beautica"

    <!-- *****
        El @font hace referencia a la carpeta que acabamos de crear
        Y el beautica al nombre de la fuente (que se tiene que encontrar ahí)
    -->
```



Tarjeta de felicitación 2

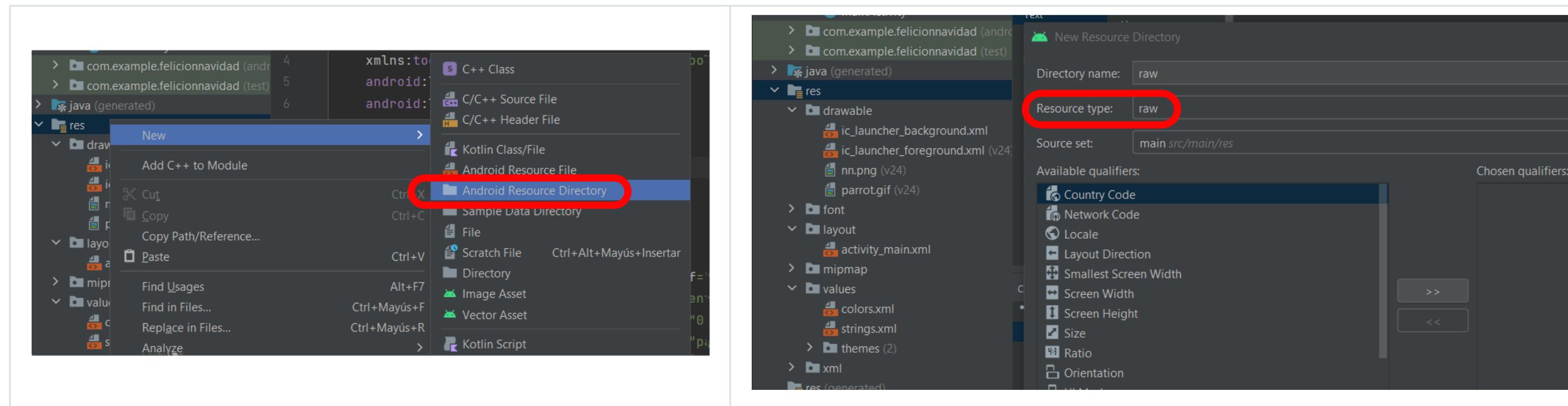
Resultado final:



Tarjeta de felicitación 4 - música

En caso de dudas aquí hay una guía donde se explica como añadir el mediaplayer en Andorid: [Developer Android](#).

Empezamos creando una carpeta en el directorio **res**, es importante que tenga el mismo nombre y tipo.



Añadimos un archivo .mp3 a la carpeta raw, el archivo tiene que estar en minúsculas y sin espacios.

En el archivo MainActivity.java creamos el objeto MediaPlayer:

```
.java  
  
MediaPlayer mediaPlayer = MediaPlayer.create(this, R.raw.nokia_arabe_tono);
```

⚠️ Explicación de la URI! ⚠️

1. La clase R es la que tiene todos los recursos de nuestro proyecto.
2. Raw es la carpeta que contiene los .mp3.
3. nokia_arabe_tono es el archivo al que vamos a hacer referencia sin extensión.

Ahora arrancaremos la canción:

```
.java  
  
mediaPlayer.start();
```

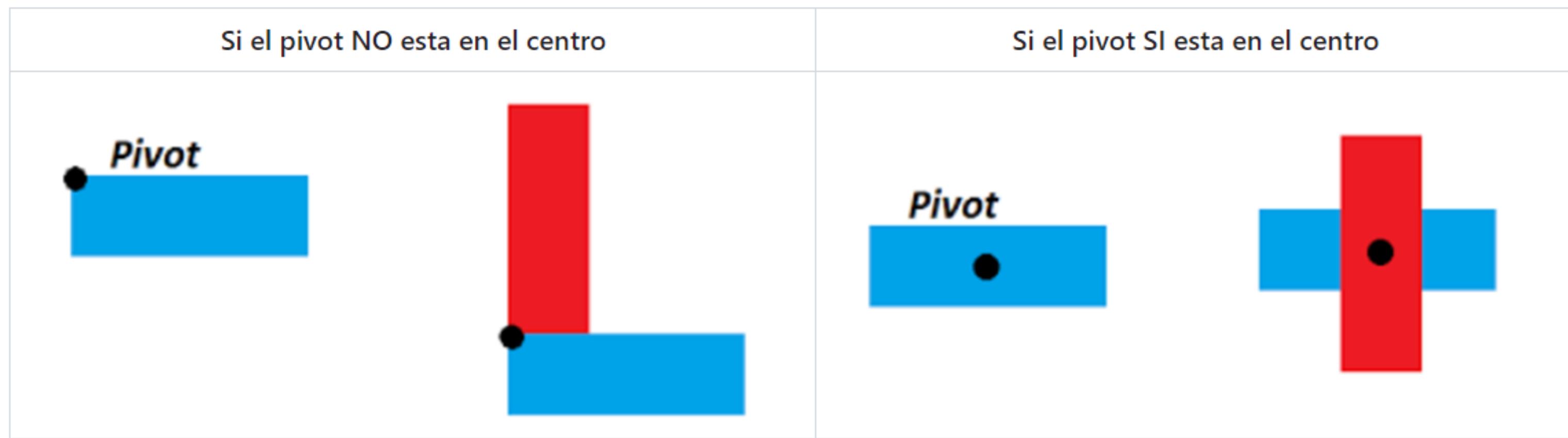
Animaciones básicas

Xml de rotación

1. **duration** → 1000 = 1 segundo
2. **fillAfter** → al finalizar el tiempo indicado ésta regresa a su estado inicial.
3. **fromDegrees** → desde qué grado iniciaremos la animación, en este caso 0.
4. **pivotX** → pivote del eje X, al indicar 50% estamos diciendo que se localizará en el centro de ese eje.
5. **pivotY** → pivote del eje Y, al indicar 50% estamos diciendo que se localizará en el centro de ese eje.
6. **toDegrees** → hasta qué grado queremos llegar para terminar la animación.

```
.xml  
<rotate xmlns:android="http://schemas.android.com/apk/res/android"  
        android:duration="3000"  
        android:fillAfter="true"  
        android:fromDegrees="0"  
  
        android:pivotX="50%"  
        android:pivotY="50%"  
        android:toDegrees="-90" />
```

Si empezamos en 0 y terminamos en -90 significa que girara en sentido contrario a las agujas del reloj



Animaciones básicas

Xml de traslación

1. duration → 1000 = 1 segundo	
2. fillAfter → al finalizar el tiempo indicado esta regresa a su estado inicial.	
3. fromXDelta → punto X de inicio para la animación	
4. fromYDelta → punto Y de inicio para la animación	
5. toXDelta → punto X de fin para la animación	
6. toYDelta → punto Y de fin para la animación	

.xml

```
<translate xmlns:android="http://schemas.android.com/apk/res/android"  
    android:duration="3000"  
    android:fillAfter="true"  
    android:fromXDelta="0"  
    android:toXDelta="200"  
    android:fromYDelta="0"  
    android:toYDelta="200" />
```

codetomg.com

Animaciones básicas

Xml de escala

1. **duration** → 1000 = 1 segundo
 2. **fillAfter** → al finalizar el tiempo indicado ésta regresa a su estado inicial.
 3. **fromXScale** → valor inicial en X para la escala.
 4. **fromYScale** → valor inicial en Y para la escala.
 5. **toXScale** → valor final en X.
 6. **toYScale** → valor final en Y.
- Si toXScale es mayor que fromXScale el objeto se hace mas grande.
 - Si toXScale es menor que fromXScale el objeto se hace mas pequeño.

```
.xml  
  
<scale xmlns:android="http://schemas.android.com/apk/res/android"  
        android:duration="3000"  
        android:fillAfter="true"  
        android:fromXScale="0"  
        android:toXScale="2"  
        android:fromYScale="0"  
        android:toYScale="2"  
        android:pivotX="50%"  
        android:pivotY="50%" />
```

codetomsg.com

Animaciones básicas

¿Cómo arrancar las animaciones?

1. Necesitamos crear la animacion en xml.
2. Crearnos en la interfaz el view que tendrá asociado esa animación.
3. Hacer una referencia a la ubicación de ese view, en el ejmplo una imagen.

.java

```
ImageView imagen;  
imagen = (ImageView) findViewById(R.id.imagen);
```

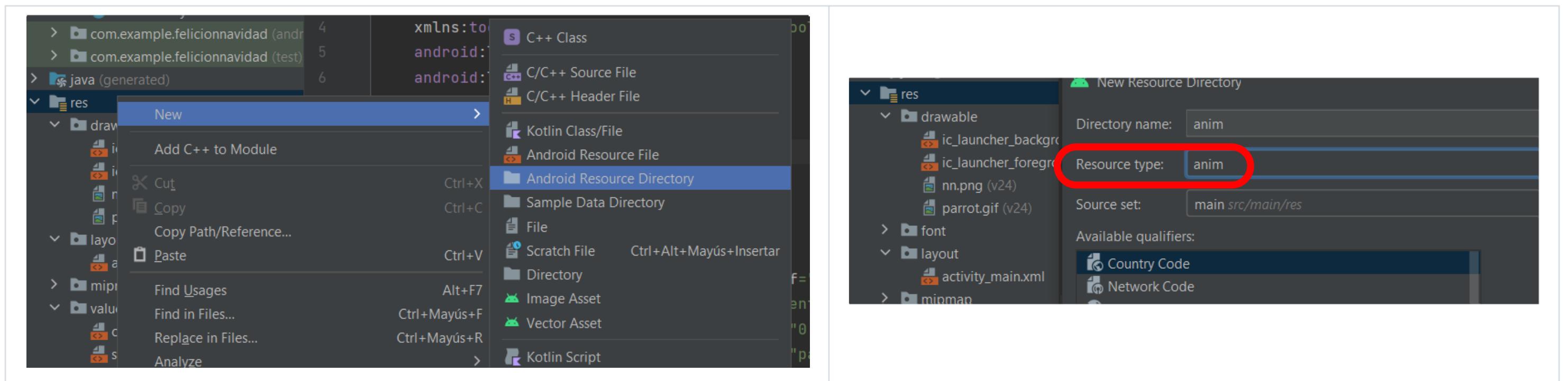
4. Cargar la animación con la clase Animation (tendrá que estar definido el archivo xml):

.java

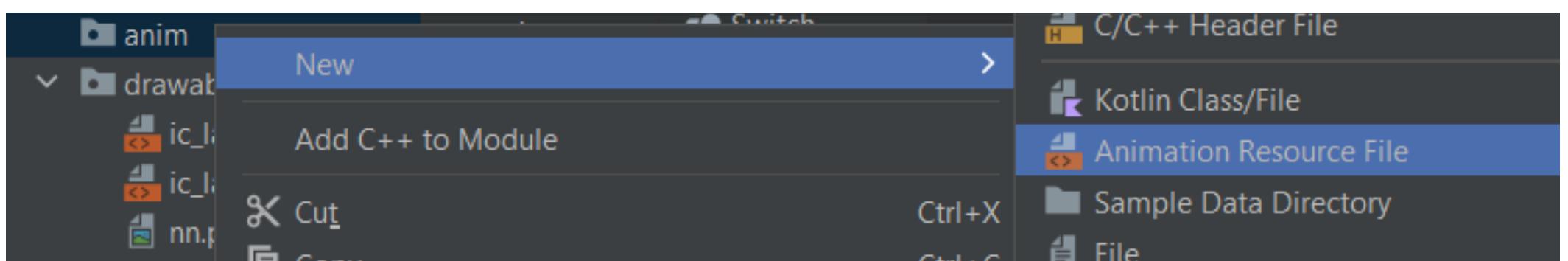
```
Animation rotate = AnimationUtils.loadAnimation(this, R.anim.rotate_animation1);  
imagen.startAnimation(rotate);
```

Tarjeta de felicitación 5 - animaciones

Para crear una animación tendremos que crearnos una carpeta en **res**



Después a la carpeta anim tendremos que añadir un fichero de recursos:



Tarjeta de felicitación 5 - animaciones

.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">

    <!-- Escalado -->
    <scale
        android:duration="3000"
        android:pivotX="50%"
        android:pivotY="50%"
        android:fromXScale="0"
        android:fromYScale="0"
        android:toXScale="1"
        android:toYScale="1" />

    <!-- Rotacion -->
    <rotate
        android:duration="3000"
        android:pivotX="50%"
        android:pivotY="50%"
        android:fromDegrees="0"
        android:toDegrees="360" />

</set>
```

.xml

```
<TextView
    android:id="@+id/titulo" />
```

codetomg.com

.java

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    TextView titulo = (TextView) findViewById(R.id.titulo);
    Animation animTitulo = AnimationUtils.loadAnimation(this, R.anim.anim_titulo);
    titulo.startAnimation(animTitulo);
}
```

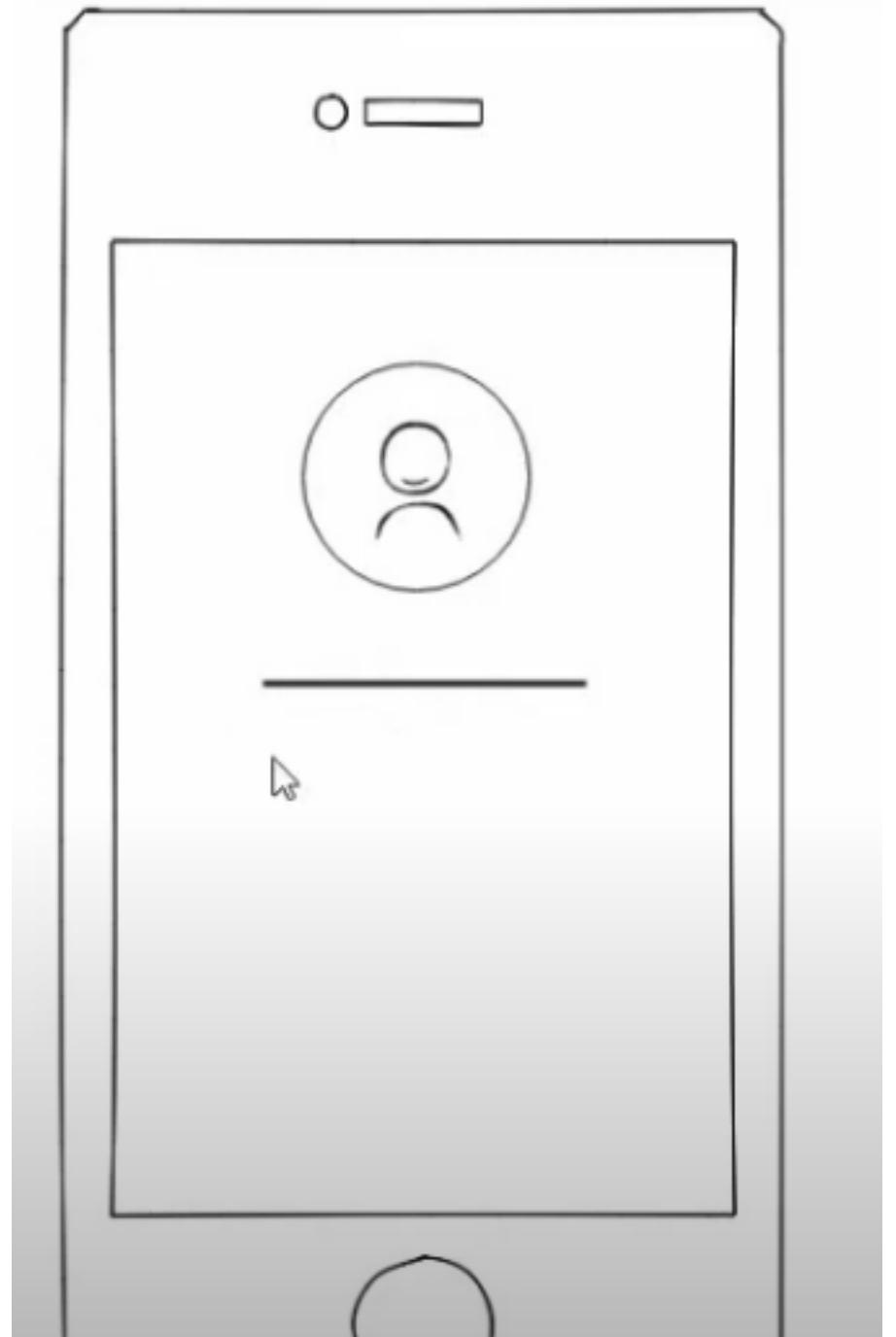
codetomg.com

Tarjeta de felicitación 6 - Splash

Diseño: ¿qué es un wireframe?

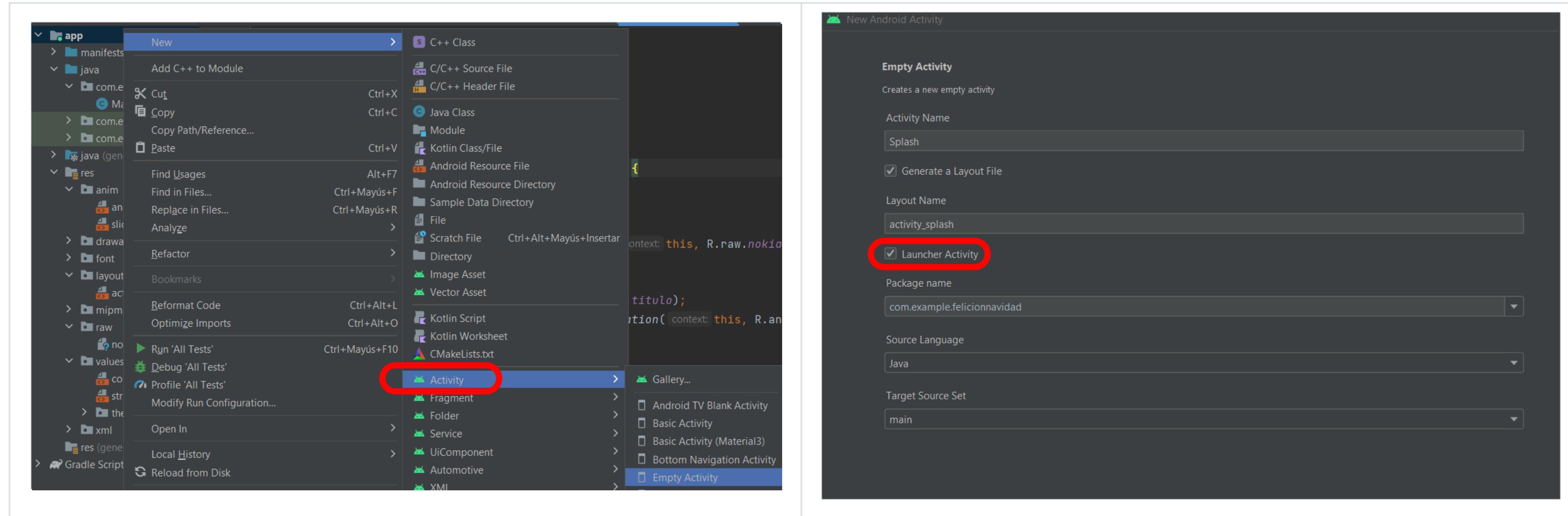
Representación visual en escala de grises de la estructura y funcionalidad de una aplicación.

Vamos a crear una aplicación que tenga como ventana de inicio este diseño:



Tarjeta de felicitación 6 - Splash

Para crear el splash tendremos que hacer lo siguiente:



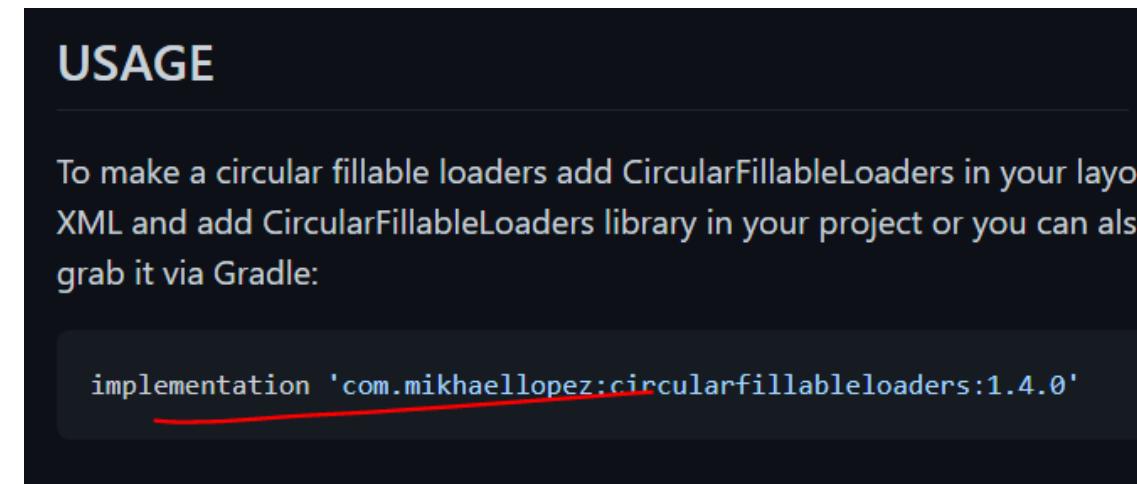
Una vez creada la nueva actividad, configuramos el `AndroidManifest.xml` y en el tramo que le pertenece a `MainActivity` eliminamos esto:

```
AndroidManifest

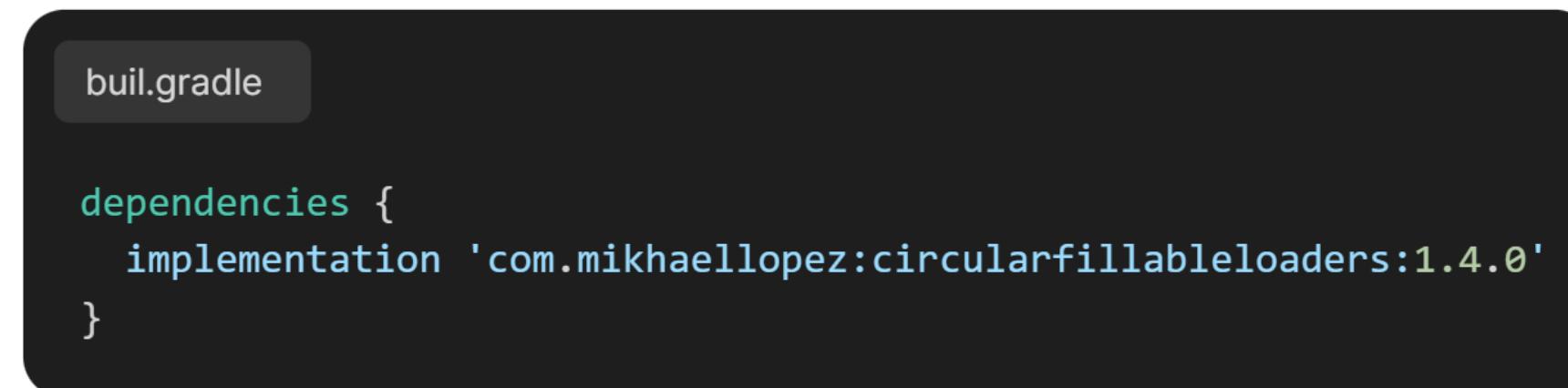
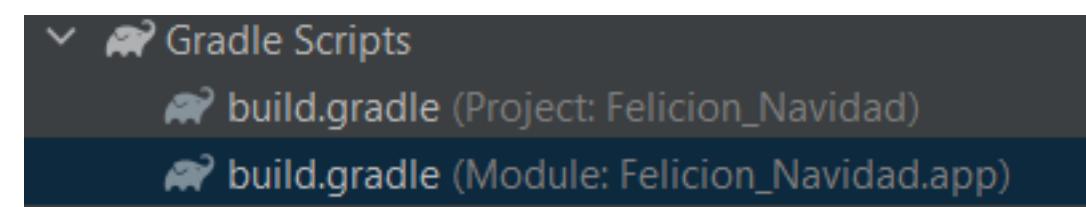
<intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Tarjeta de felicitación 6 - Splash

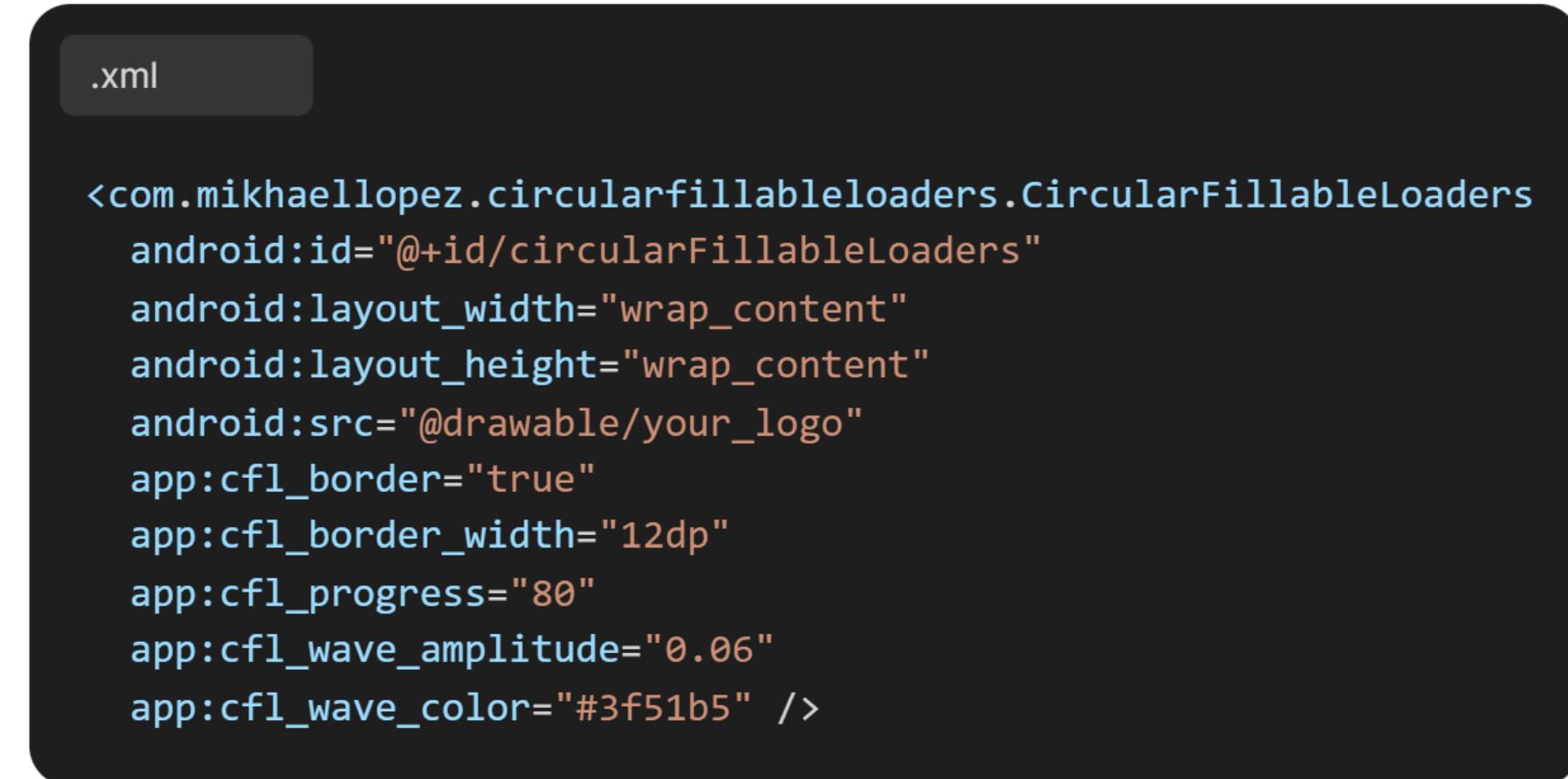
Para cargar una animación, vamos a este repositorio de [github](#) y copiamos el código que aparece aquí:



Ahora vamos a este archivo y añadimos el tramo copiado en el apartado de dependencias.

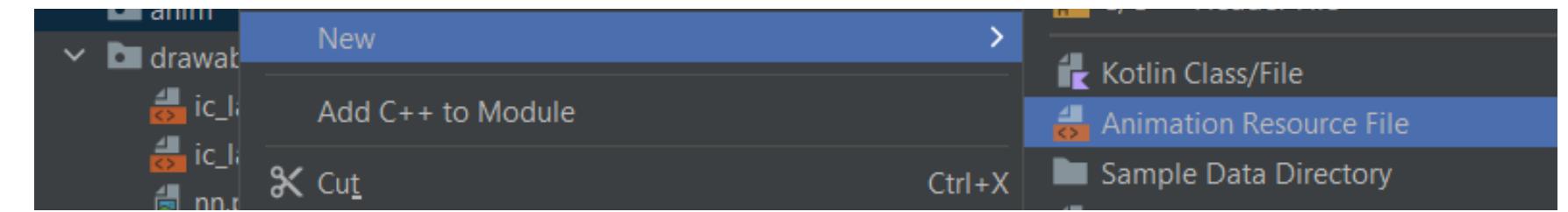


codetomg.com



Al añadirlo nos saldrá un error en todo el tramo nuevo	Esto se debe a que hicimos un cambio en el compilador y cada vez que se hace un cambio tenemos que actualizar el proyecto
 The screenshot shows the Android Studio editor with the XML code from the previous slide. Red squiggly lines are underlined under the entire code block, indicating syntax errors.	 The screenshot shows the Android Studio toolbar with the 'Sync Now' button highlighted with a red underline.

Tarjeta de felicitación 6 - Splash



```
.xml
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:fillAfter="true"><!--que se quede en la posicion esta cuando termine la animacion-->

    <translate
        android:fromXDelta="600"
        android:fromYDelta="0"
        android:toXDelta="0"
        android:toYDelta="0"
        android:duration="3000" />

    <scale
        android:pivotX="0"
        android:pivotY="50%"
        android:fromXScale="1"
        android:fromYScale="1"
        android:toXScale="0"
        android:toYScale="1"

        android:duration="2000"
        android:startOffset="3500"//delay que se esperara para que empiece la animacion />
</set>
```

```
.java
TextView cargando = findViewById(R.id.tituloSplash);
Animation animacion = AnimationUtils.loadAnimation(this, R.anim.splash); //carga la animacion
cargando.startAnimation(animacion); //inicia la animacion
```

Resultado final



Cambio de activity

```
.java  
public class Splash extends AppCompatActivity implements Animation.AnimationListener{  
    ...  
  
    @Override  
    public void onAnimationEnd(Animation animation) {  
        //instanciamos la ventana que queremos abrir posteriormente  
        Intent intent = new Intent(Splash.this, MainActivity.class);  
        startActivity(intent); //abrimos la ventana del menu  
        finish(); //para terminar esa accion y no se pueda acceder a ella  
    }  
}
```

Accionamiento de la función

En el `onCreate` añadimos esto para que esté a la escucha, le pasaremos la misma ventana para que así podamos usar los listeners implementados anteriormente.

```
.java  
animacion.setAnimationListener(this);
```

codetomg.com

unir LA UNIVERSIDAD
EN INTERNET