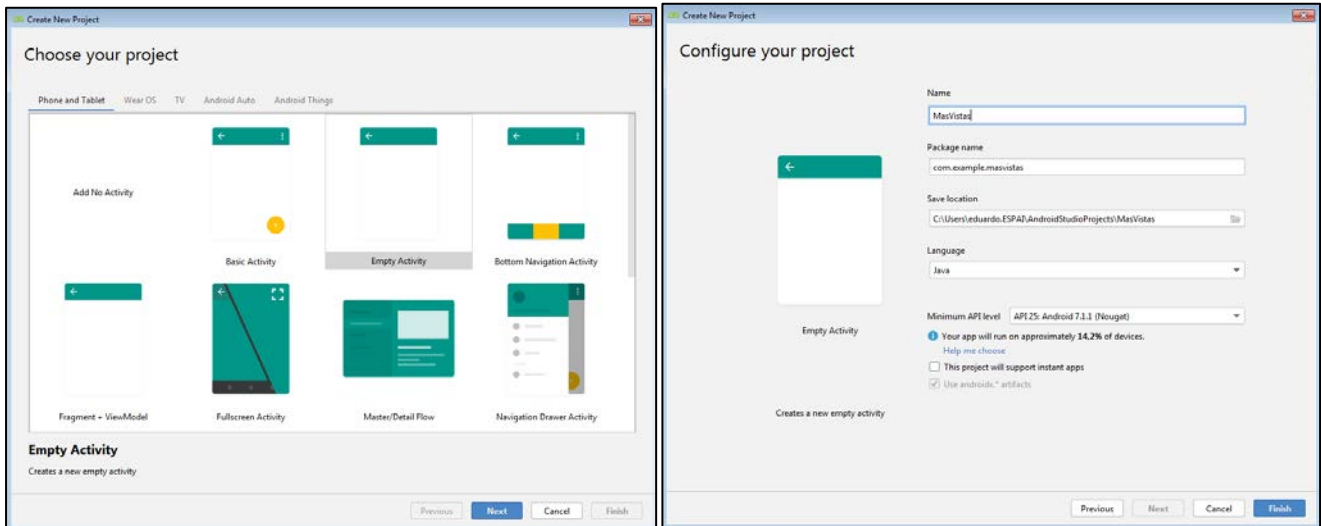


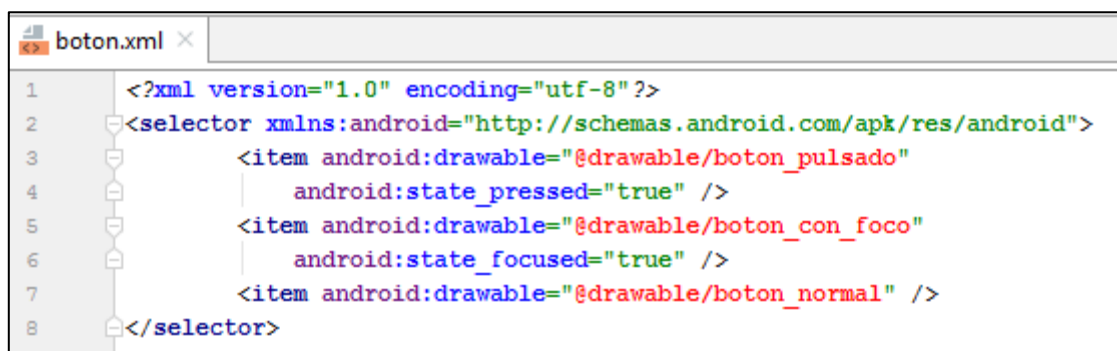
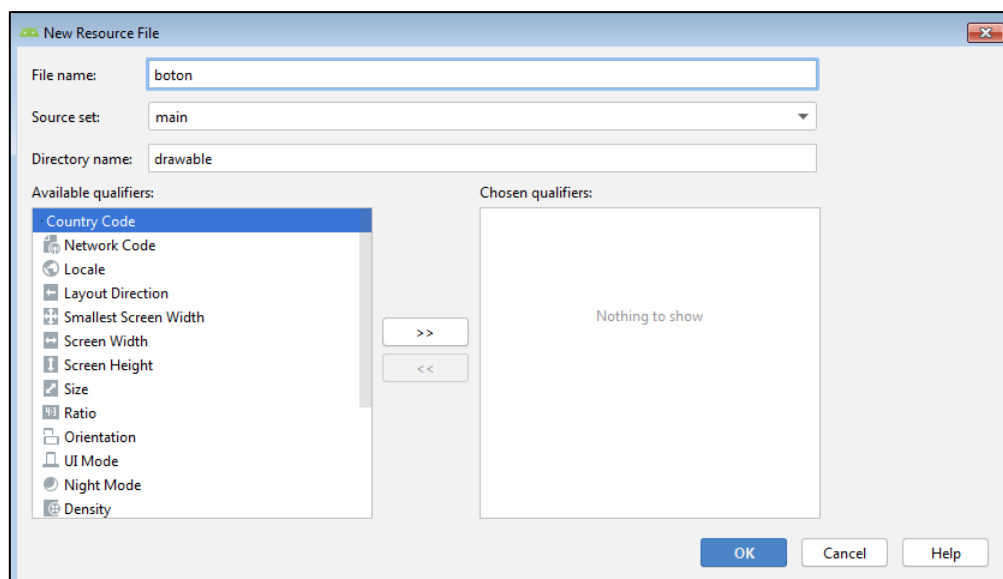
## PRACTICA 5: UN BOTÓN CON GRÁFICOS PERSONALIZADOS

### Parte I: Un botón con gráficos personalizados

**Paso 1.** Crea un nuevo proyecto, de nombre MasVistas. En la tercera ventana selecciona Empty Activity. Puedes dejar el resto de parámetros con los valores por defecto.



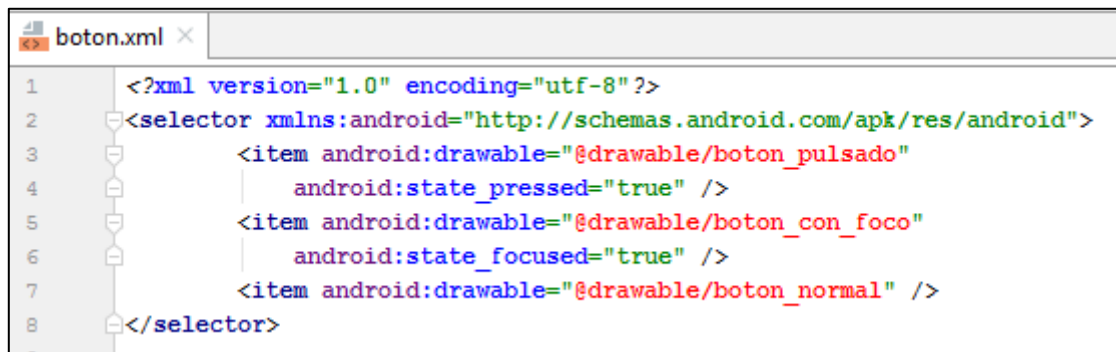
**Paso 2.** Crea el fichero boton.xml en la carpeta res/drawable/. Para ello puedes utilizar el menú File >New > Drawable Resource File. Introduce en el campo File name: <<boton>> Reemplaza el código por el siguiente:



Este XML define un recurso único gráfico (drawable) que cambiará en función del estado del botón. El primer ítem define la imagen usada cuando se pulsa el botón, el segundo ítem define la imagen usada cuando el botón tiene el foco (cuando el botón está seleccionado con la rueda de desplazamiento o las teclas de dirección) y el tercero, la imagen en estado normal. Los gráficos, y en concreto los drawables, se estudiarán en el capítulo 4.

**NOTA:** El orden de los elementos <item> es importante. Cuando se va a dibujar se recorren los ítems en orden hasta que se cumpla una condición. Debido a que "boton\_normal" es el último, sólo se aplica cuando las condiciones state\_pressed y state\_focused no se cumplen.

**Paso 3.** Crea el fichero **boton.xml** en la carpeta **res/drawable/**. Para ello puedes utilizar el menú **Archivo/Nuevo/Android XML File** y pon en File: "botón" y selecciona en tipo Drawable. Reemplaza el código por el siguiente:



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <selector xmlns:android="http://schemas.android.com/apk/res/android">
3     <item android:drawable="@drawable/boton_pulsado"
4         android:state_pressed="true" />
5     <item android:drawable="@drawable/boton_con_foco"
6         android:state_focused="true" />
7     <item android:drawable="@drawable/boton_normal" />
8 </selector>
```

**Paso 4.** Descarga de <http://www.androidcurso.com/index.php/119> las tres imágenes que aparecen a continuación. Para bajar cada imagen, pulsa sobre los nombres. Guárdalos con el nombre de fichero se indica a continuación:

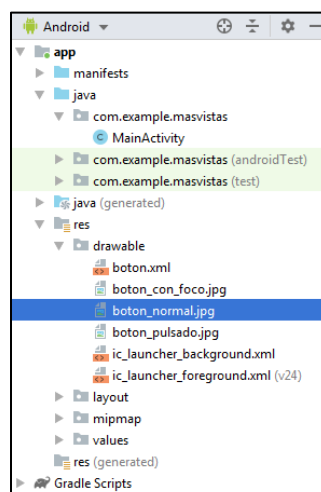


[boton\\_normal.jpg](#)

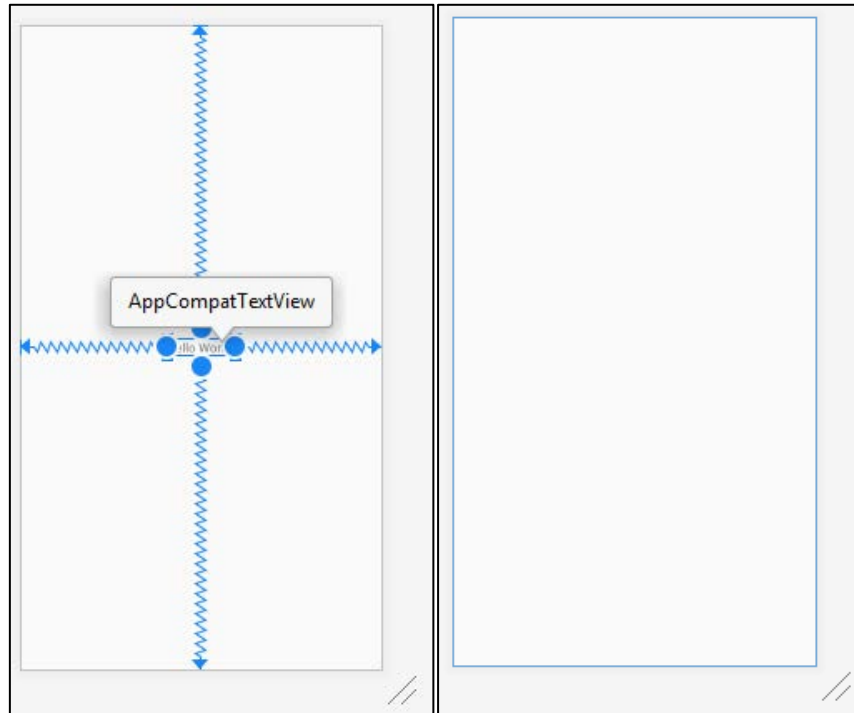
[boton\\_con\\_foco.jpg](#)

[boton\\_pulsado.jpg](#)

**Paso 5.** Selecciona los tres ficheros y cópialos en el portapapeles (Ctrl-C), selecciona la carpeta **res/drawable/** del proyecto y pega los ficheros (Ctrl-V). Te preguntará si quieres copiarlos a la carpeta de recursos por defecto a alguna de recursos alternativos. Selecciona la primera opción.

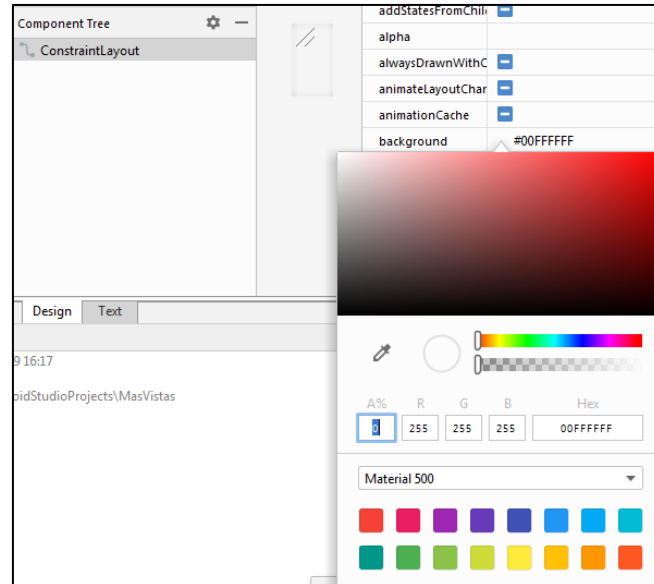


**Paso 6.** Abre el fichero res/layout/activity\_main.xml.



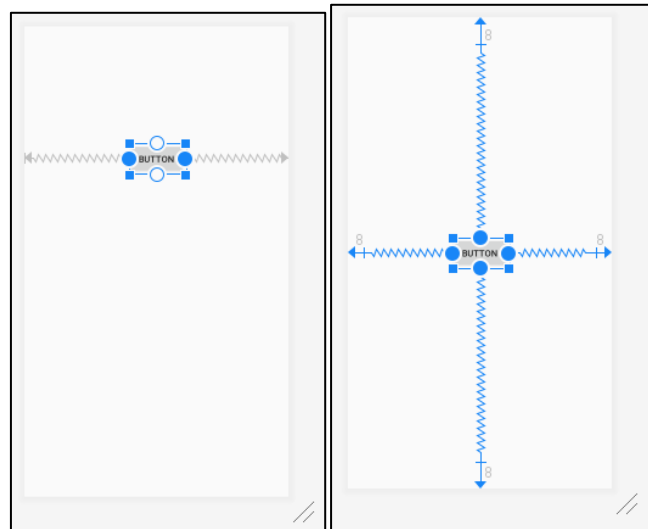
**Paso 7.** En la ventana Component Tree, elimina el TextView que encontrarás dentro del ConstraintLayout.

**Paso 8.** Selecciona el ConstraintLayout. En la ventana de Attributes busca el atributo Background. Pulsa en el icono de la herramienta y selecciona el recurso Color/android/White (#FFFFFF)



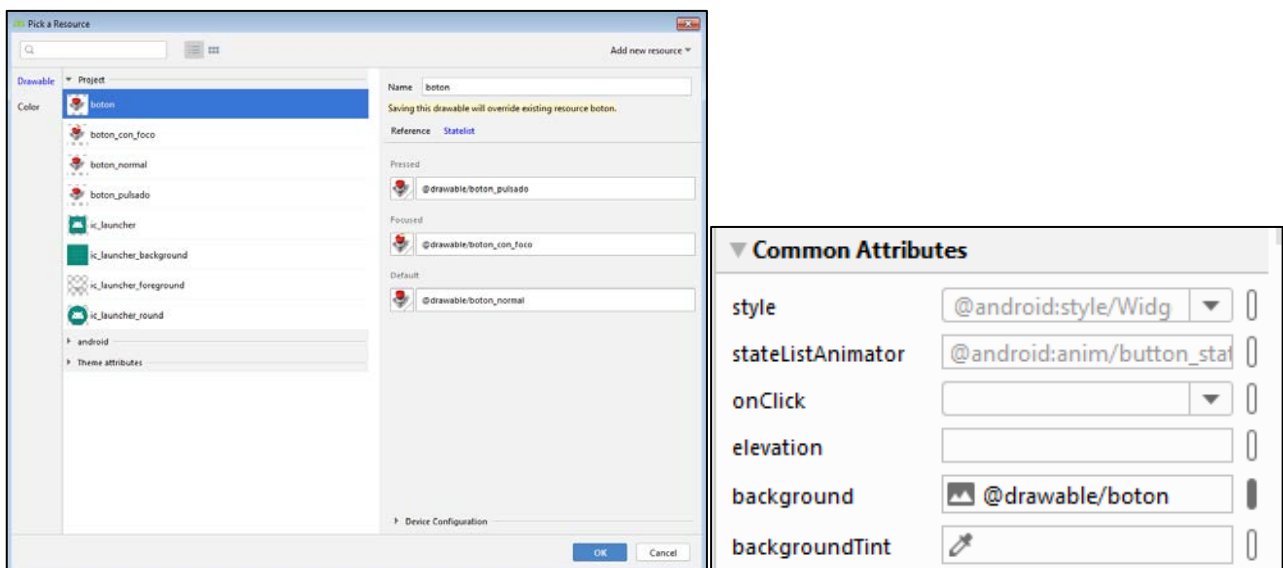
```
activity_main.xml x
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      android:background="#00FFFFFF"
9      tools:context=".MainActivity" />
```

**Paso 9.** Arrastra una vista de tipo Button dentro del ConstraintLayout.



**Paso 10.** Sitúa el botón en el centro. Para ello selecciona cada uno de sus puntos de anclaje arrastrando hasta el borde al que mira. El resultado ha de ser:

**Paso 11.** Selecciona el atributo Background y pulsa en el botón selector de recurso (con puntos suspensivos). Selecciona Drawable/boton.



**Paso 12.** Modifica el atributo Text para que no tenga ningún valor.

tag		
targetApi		
text		
textAlignment		
textAllCaps	<input checked="" type="checkbox"/>	true

**Paso 13.** Introduce en el atributo onClick el valor sePulsa.

nextFocusUp		
numeric	<input checked="" type="checkbox"/>	
onClick	sePulsa	
orientation		
outlineAmbientSha		

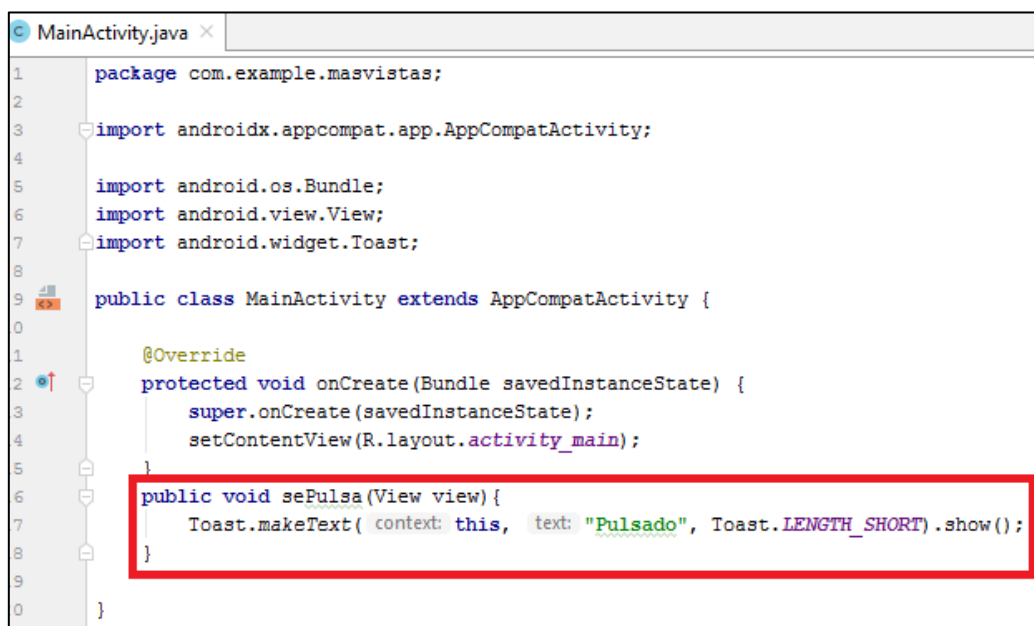
A continuación se muestra el código resultante para activity\_main.xml:



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     android:background="#00FFFFFF"
9     tools:context=".MainActivity" >
10
11     <Button
12         android:id="@+id/button"
13         android:layout_width="wrap_content"
14         android:layout_height="wrap_content"
15         android:layout_marginStart="8dp"
16         android:layout_marginTop="8dp"
17         android:layout_marginEnd="8dp"
18         android:layout_marginBottom="8dp"
19         android:background="@drawable/boton"
20         android:onClick="sePulsa"
21         app:layout_constraintBottom_toBottomOf="parent"
22         app:layout_constraintEnd_toEndOf="parent"
23         app:layout_constraintStart_toStartOf="parent"
24         app:layout_constraintTop_toTopOf="parent" />
25 </androidx.constraintlayout.widget.ConstraintLayout>
```

**Nota:** Una forma alternativa puede ser definir  
android:background="@android:color/transparent"  
android:drawableEnd="@drawable/botón"

**Paso 14.** Abre el fichero MainActivity.java e introduce al final, antes de la última llave, el código:

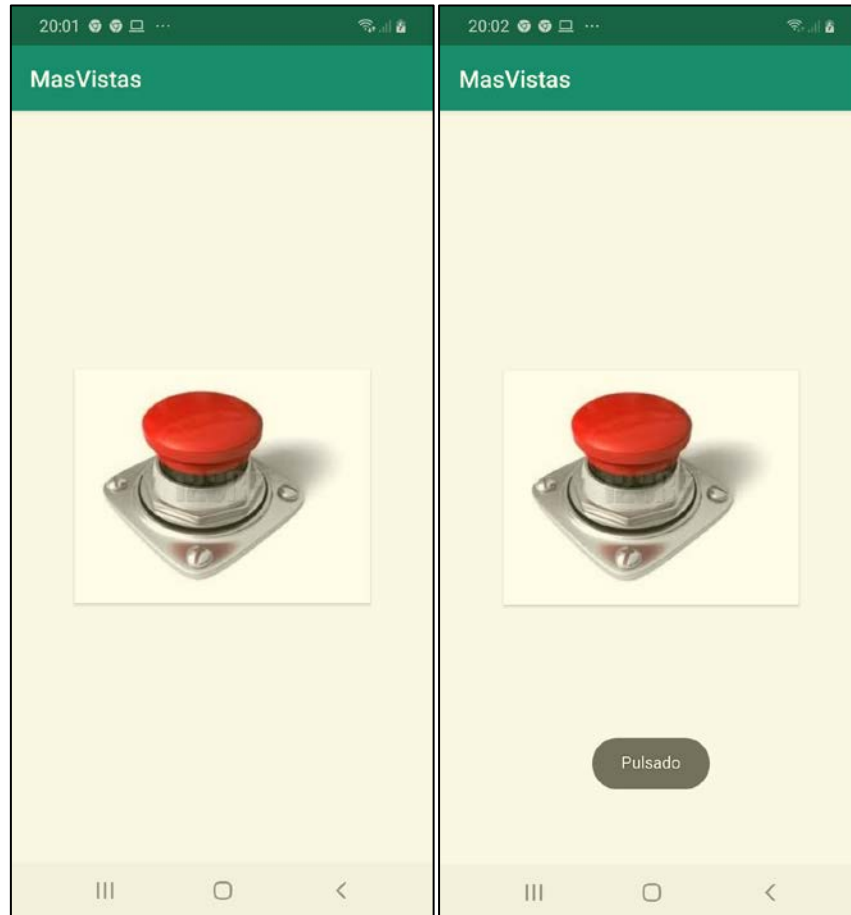


```
1 package com.example.masvistas;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.Toast;
8
9 public class MainActivity extends AppCompatActivity {
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_main);
15     }
16
17     public void sePulsa(View view) {
18         Toast.makeText( context: this, text: "Pulsado", Toast.LENGTH_SHORT).show();
19     }
20 }
```

**NOTA:** Pulsa Alt-Intro en Android Studio para que se añadan automáticamente los paquetes que faltan en la sección import.

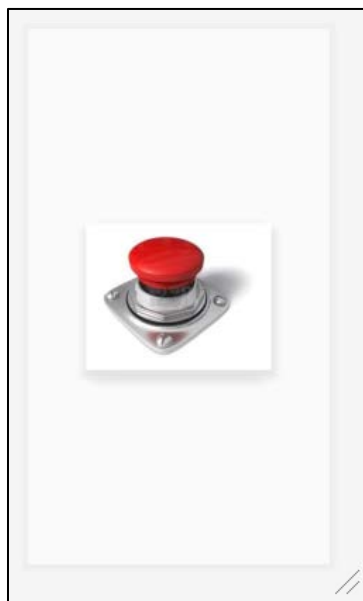
El método anterior se ejecutará cuando se pulse el botón. A este tipo de métodos se los conoce como escuchadores de eventos (listeners). Este método se limita a lanzar un toast, es decir, un aviso que permanece cierto tiempo sobre la pantalla y luego desaparece. Los tres parámetros son: el contexto utilizado, que coincide con la actividad, el texto a mostrar y el tiempo que permanecerá este texto. Los conceptos de actividad y contexto se desarrollarán en el siguiente capítulo.

**Paso 15.** Ejecuta el proyecto y verifica el resultado.

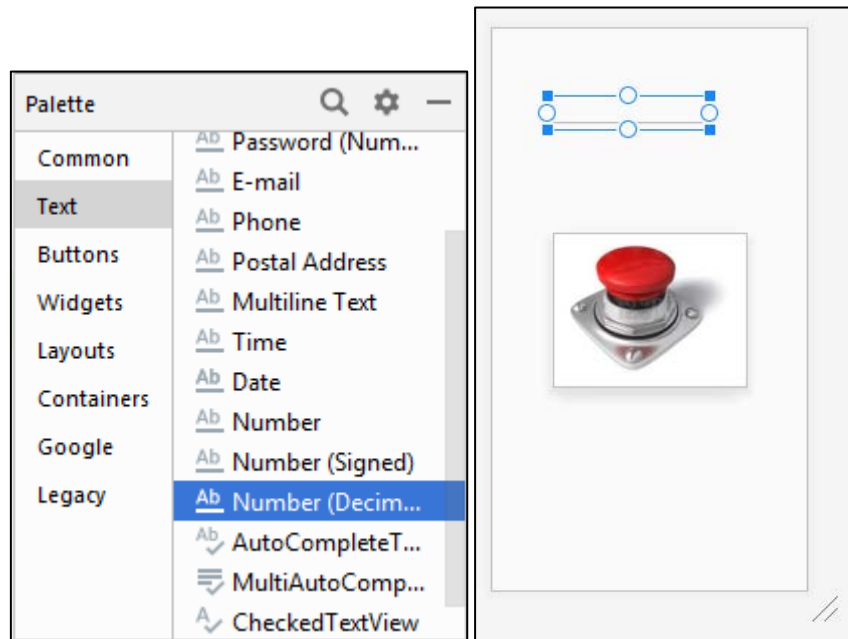


## **Parte II: Acceder y modificar las propiedades de las vistas por código**

**Paso 1.** Abre el *Layout main.xml* creado en la práctica anterior.

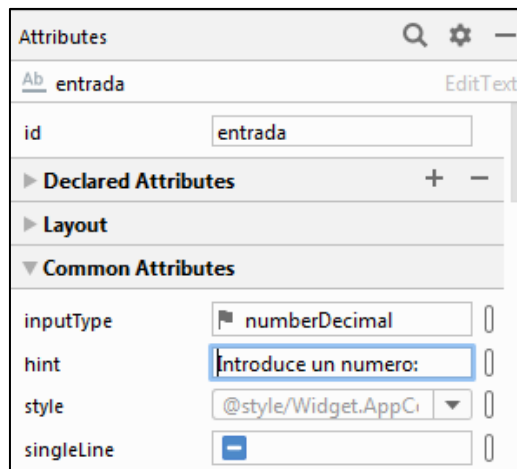


**Paso 2.** En la paleta de vistas, dentro de *Text Fields*, busca *Number (Decimal)* y arrástralo encima del botón rojo.

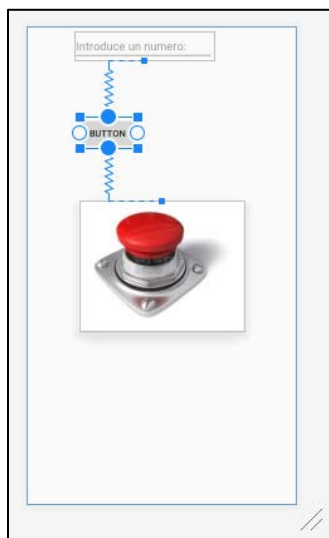


**Paso 3.** Modifica algunos atributos de esta vista:

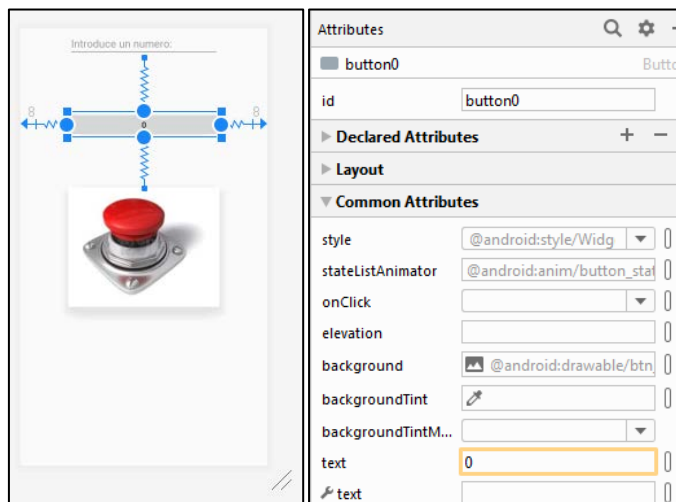
*Hint* = "Introduce un número", *id* = "@+id/entrada"



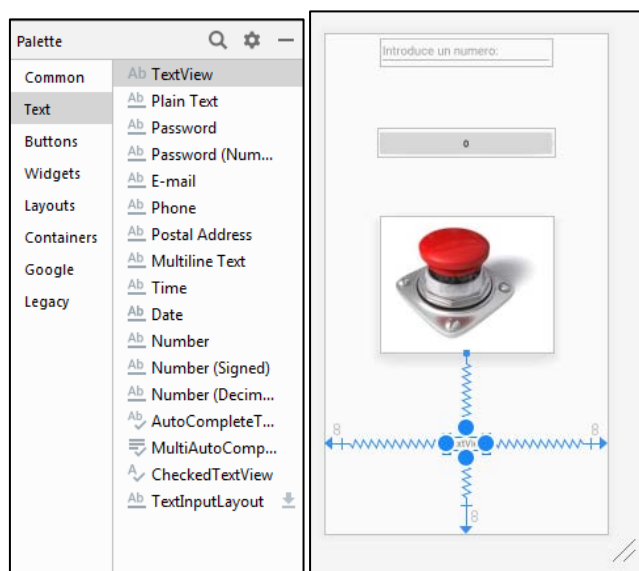
**Paso 4.** En la paleta de vistas, dentro de *Buttons*, busca *Button* y arrástralo encima del botón rojo.



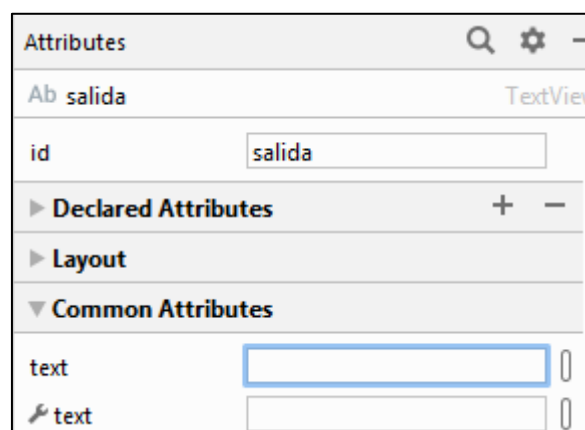
**Paso 5.** Modifica algunos atributos de esta vista: Haz que su ancho ocupe toda la pantalla, que su texto sea “0” y que su id sea “boton0”.



**Paso 6.** En la paleta de vistas, dentro de Text, busca *TextView* y arrástralo debajo del botón rojo.

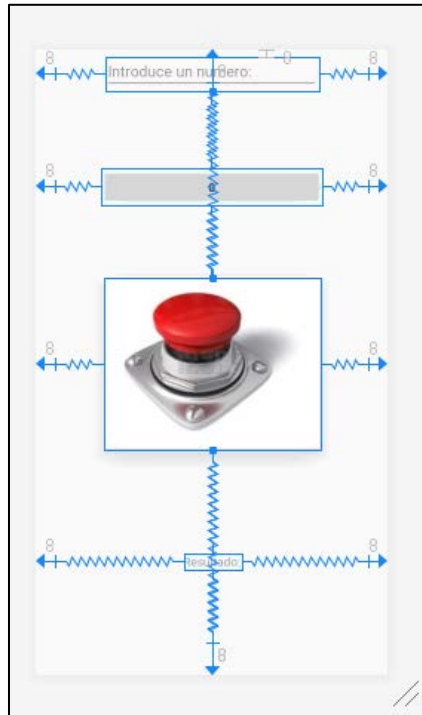


**Paso 7.** Modifica algunos atributos de esta vista: **TextColor** = #0000FF, **Text** = “”, **Hint** = “Resultado”, **id**= “salida”.





**Paso 8.** Ajusta las restricciones de las vistas introducidas.



**Paso 9.** Abre el fichero MainActivity. En Java vamos a añadir dos nuevas propiedades a la clase. Para ello copia el siguiente código al principio de la clase (antes del método onCreate()):

```
private EditText entrada;  
private TextView salida;
```

```
public class MainActivity extends AppCompatActivity {  
    private EditText entrada;  
    private TextView salida;
```

**NOTA:** Recuerda pulsar Alt-Intro para que se añadan los paquetes de las dos nuevas clases utilizadas.

**Paso 10.** En Java copia al final del método onCreate() las siguientes dos líneas:

```
entrada = (EditText) findViewById(R.id.entrada);  
salida = (TextView) findViewById(R.id.salida);
```

Como se explicó al principio del capítulo, las diferentes vistas definidas en activity\_main.xml, son creadas como objetos Java cuando se ejecuta setContentView(R.layout.main). Si queremos manipular algunos de estos objetos hemos de declarar las variables (paso 9) y asignarles la referencia al objeto correspondiente (paso 10). Para ello, hay que introducir el atributo id en XML y utilizar el método findViewById(R.id.valor\_en\_atributo\_id). Este método devuelve un objeto de la clase View.

No obstante los objetos declarados (**entrada** y **salida**) no son exactamente de esta clase por lo que Java no permite una asignación directa. En estos casos hemos de utilizar una conversión de tipo (type cast) para poder hacer la asignación.

```

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

    entrada = findViewById(R.id.entrada);
    salida = findViewById(R.id.salida);
}

```

**Paso 11.** Introduce en el atributo onClick del botón con id boton0 el valor “sePulsa0”. De esta manera, cuando se pulse sobre el botón se ejecutará el método sePulsa0(). Según la jerga de Java, diremos que este método es un escuchador del evento *click* que puede generar el objeto boton0.

Attributes			Q	⚙	—
button0		Button			
nextFocusForward					0
nextFocusLeft					0
nextFocusRight					0
nextFocusUp					0
numeric	<input type="checkbox"/>				0
onClick		sePulsa0			0
orientation					▼

**Paso 12.** Añade el siguiente método al final de la clase MainActivity.

```

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

    entrada = findViewById(R.id.entrada);
    salida = findViewById(R.id.salida);
}

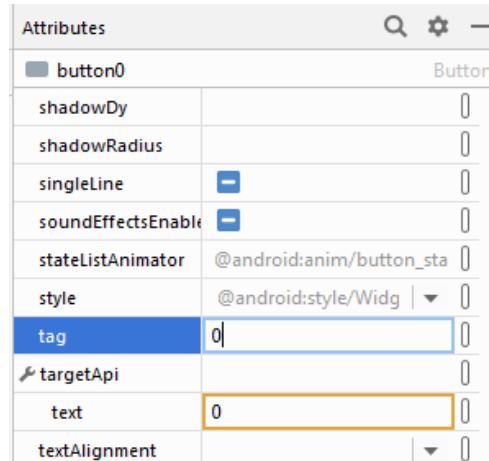
public void sePulsa(View view) {
    Toast.makeText(context: this, text: "Pulsado", Toast.LENGTH_SHORT).show();
}

public void sePulsa0(View view) {
    entrada.setText(entrada.getText()+"0");
}

```

Lo que hace es asignar como textos de entrada el resultado de concatenar al texto de entrada el carácter “0”.

**Paso 13.** Añade al botón con texto "0" el atributo tag = "0".



**Paso 14.** Modifica el método sePulsa0() de la siguiente forma:

```
public void sePulsa0(View view) {  
    //entrada.setText(entrada.getText()+"0");  
    entrada.setText(entrada.getText()+ (String) view.getTag());  
}
```

**NOTA:** El resultado obtenido es equivalente al anterior. En algunos casos será interesante utilizar un mismo método como escuchador de eventos de varias vistas. Podrás averiguar la vista que causó el evento, dado que esta es pasada como parámetro del método. En el ejemplo sabemos que en el atributo *tag* guardamos el carácter a insertar. El atributo tag puede ser usado libremente por el programador para almacenar un objeto de la clase *Object* (*mas info*) (en la práctica podemos usar cualquier tipo de clase, dado que *Object* es la clase raíz de la que heredan todas las clases en Java). En nuestro caso hemos almacenado un objeto *String*, por lo que necesitamos una conversión de tipo. Otro ejemplo de *Polimorfismo*.

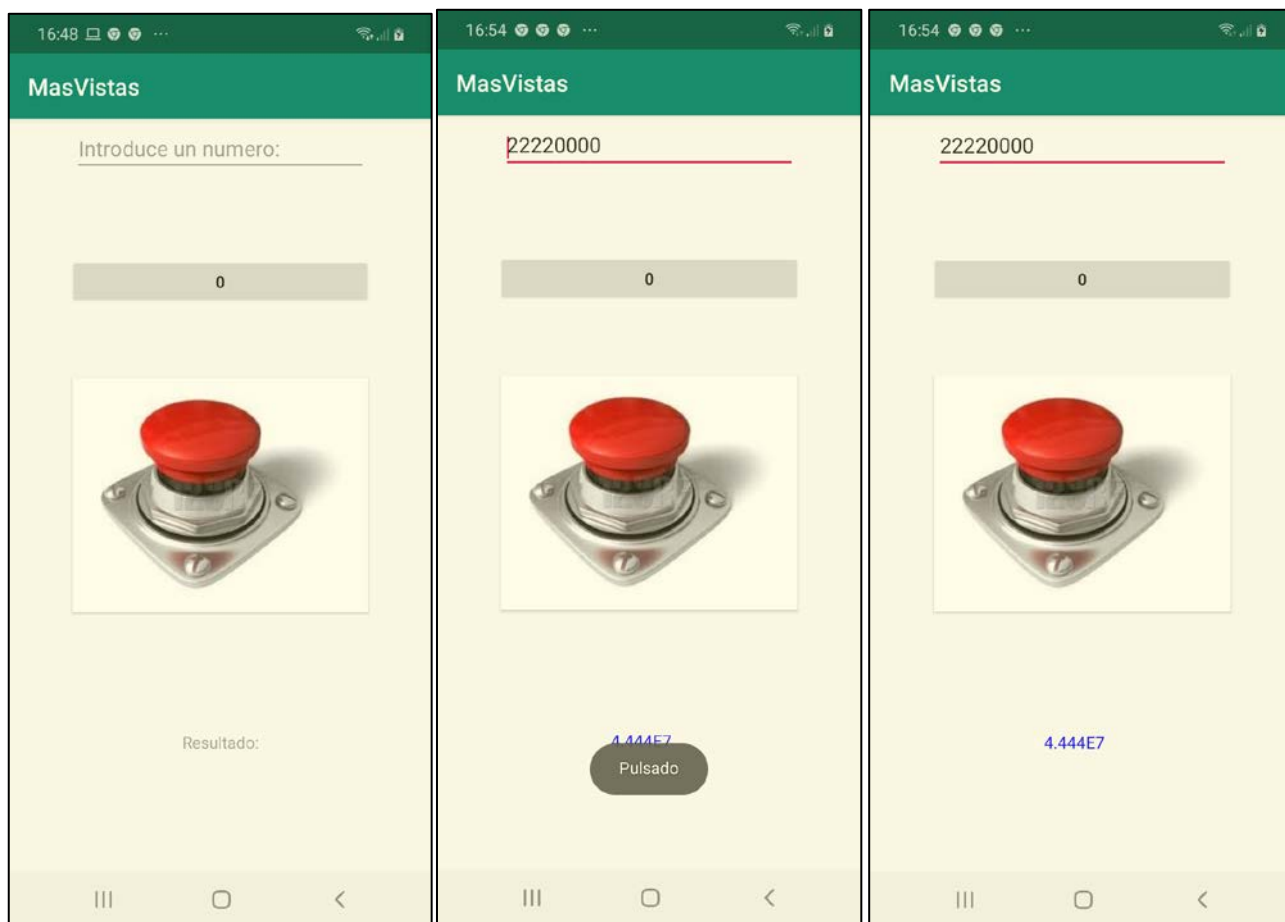
**NOTA:** Utiliza esta forma de trabajar en la práctica para no tener que crear un método *onClick* para cada botón de la calculadora.

**Paso 15.** Modifica el código de sePulsa() con el siguiente código:

```
public void sePulsa(View view) {  
    Toast.makeText(context, this, text: "Pulsado", Toast.LENGTH_SHORT).show();  
    salida.setText(String.valueOf(Float.parseFloat(entrada.getText().toString()) * 2.0));  
}
```

En este código el valor de entrada es convertido en Float, multiplicado por dos y convertido en String para ser asignado a salida.

**Paso 16.** Ejecuta el proyecto y verifica el resultado.



**NOTA:** En este ejercicio no se ha realizado la verificación de que los datos introducidos por el usuario. Has de introducir datos válidos y en el orden adecuado.