

UNIVERSIDAD DE EL SALVADOR - EDUCACIÓN A DISTANCIA
FACULTAD DE INGENIERIA Y ARQUITECTURA
ESCUELA DE INGENIERIA DE SISTEMAS INFORMATICOS
PROGRAMACIÓN PARA DISPOSITIVOS MÓVILES
PDM-115

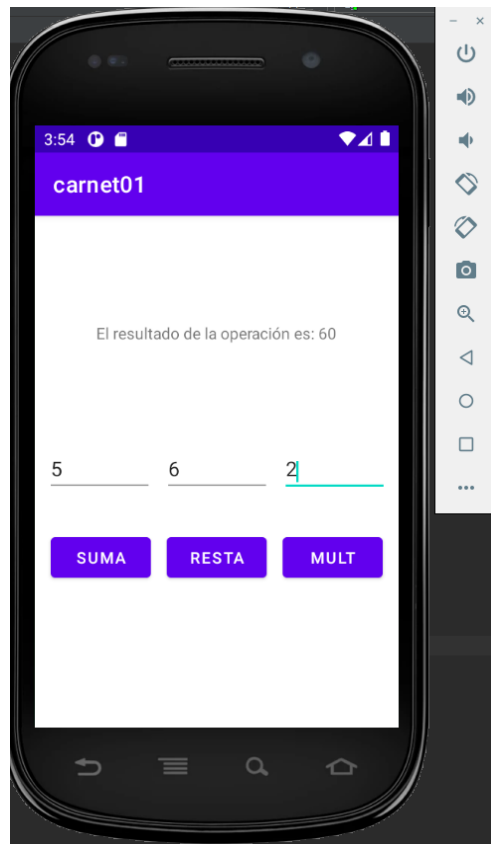


Guía de Laboratorio 1 Kotlin

Objetivos

- Conocer el entorno de programación del IDE Android Studio para programación bajo el sistema Android.
- Hacer una nueva aplicación en Android que utilice los controles: TextView, EditText y Button, interactuando con la interfaz gráfica mediante métodos y eventos.
- Ejecutar la aplicación mediante un AVD

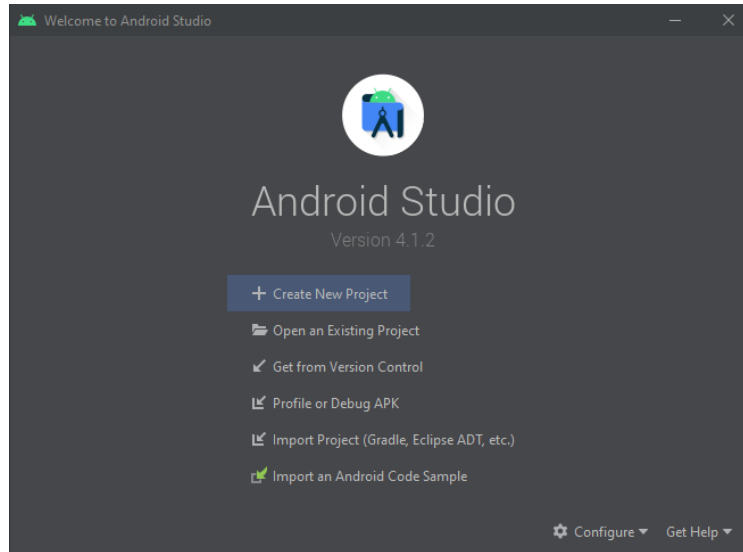
Kotlin



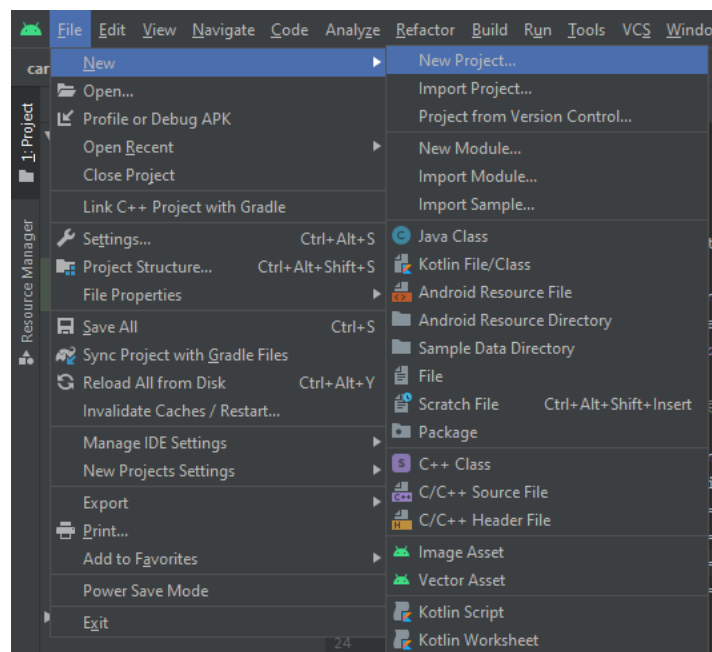
En esta guía, crearemos nuestra primera aplicación en Android utilizando Kotlin como lenguaje de programación; aunque el objetivo principal de la guía es brindar un recorrido por el IDE Android Studio y comprender algunos conceptos básicos.

Iniciemos abriendo Android Studio, en caso que no lo tengamos instalado, podemos seguir la **Guía de instalación** que está en el aula virtual.

Si no tenemos ningún proyecto activo, Android Studio nos levantara la siguiente pantalla, seleccionamos Crear Nuevo Proyecto (Create New Project).

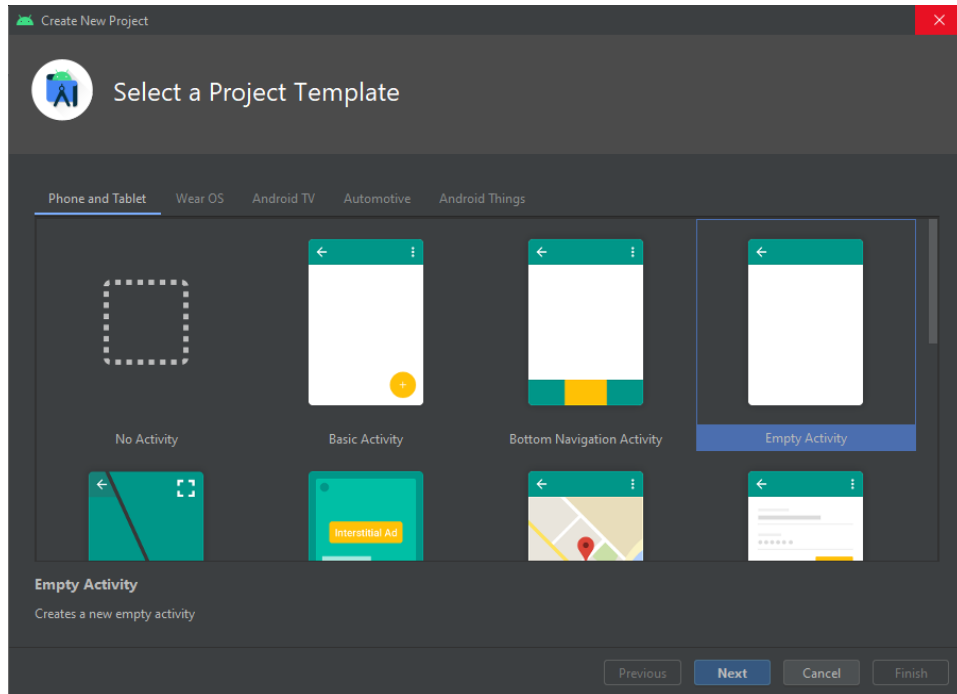


En caso que ya tengamos algún proyecto activo podemos crear uno nuevo yéndonos a **Archivo -> Nuevo -> Nuevo Proyecto (File -> New -> New Project)**

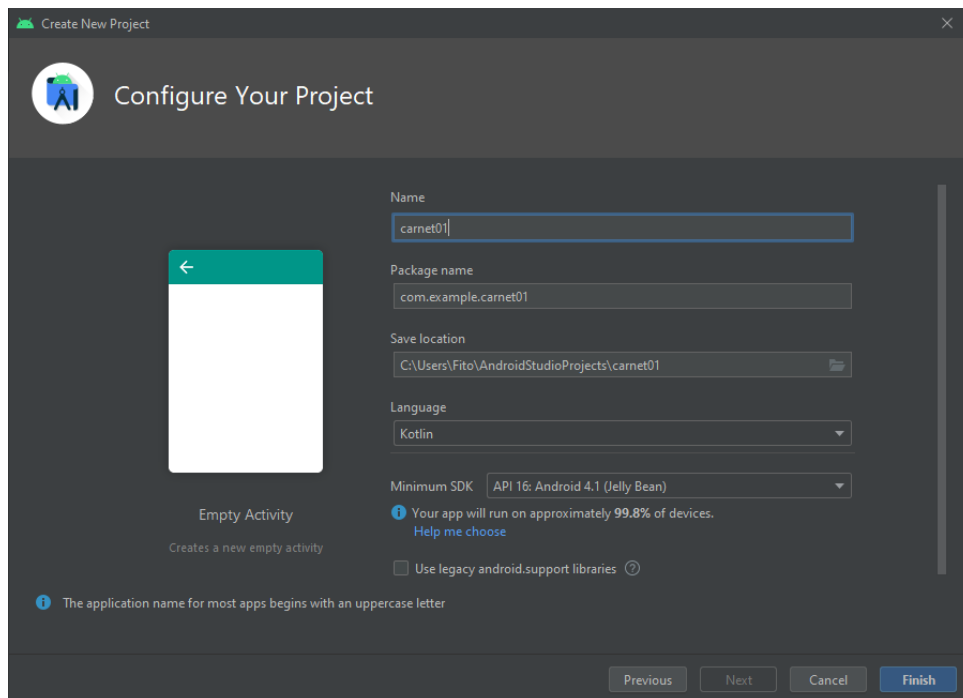


Android Studio nos ofrece algunas plantillas de Activity prediseñadas, para agilizar el desarrollo.

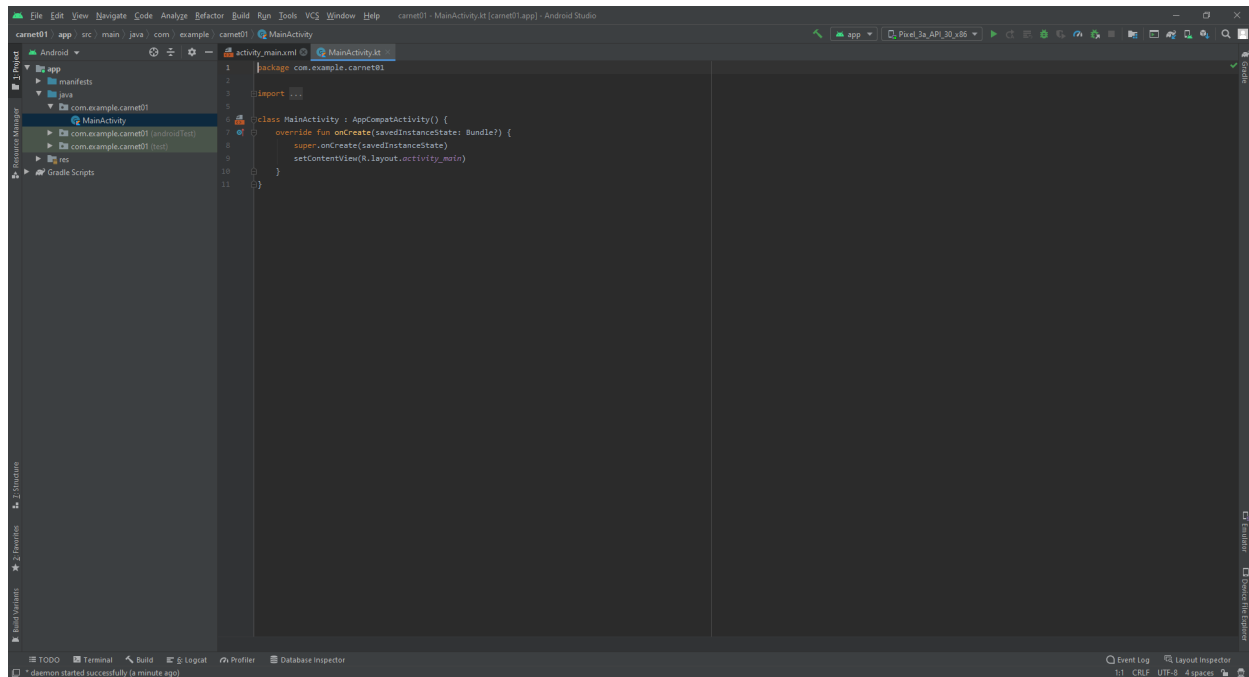
Para este laboratorio seleccionaremos una Activity Vacía (Empty Activity). Damos clic en Siguiente (Next)



Como nombre de aplicación pon tu carnet, y un correlativo si es necesario, en lenguaje selecciona Kotlin y da click en Finalizar.

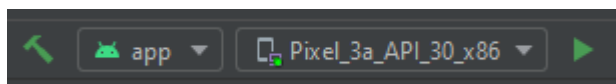


Espera a que Android Studio cree y configure tu aplicación, debe de verse más o menos de esta forma.



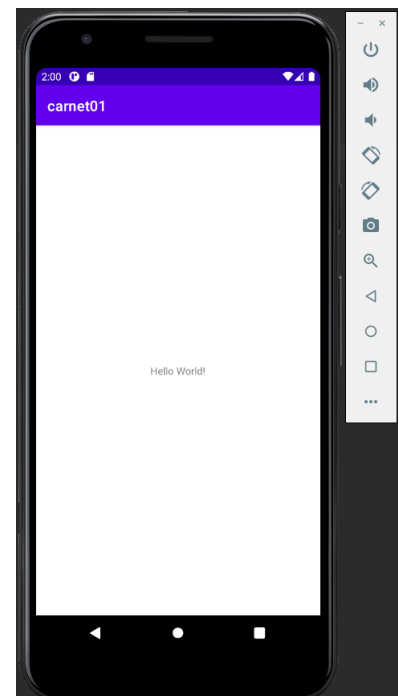
Prueba a correr tu aplicación dando clic en el triángulo verde.

Asegúrate de tener instalado un emulador. Fijate en el Dropbox justo a la izquierda del triángulo, en caso que no tengas uno, sigue la **Guía de creación de un AVD**; puedes encontrarla en el aula virtual.



Se tardará un tiempo en iniciar el emulador; una vez iniciado, se instalará y abrirá la aplicación, mostrando el mensaje de “Hello World”, como se ve en la imagen.

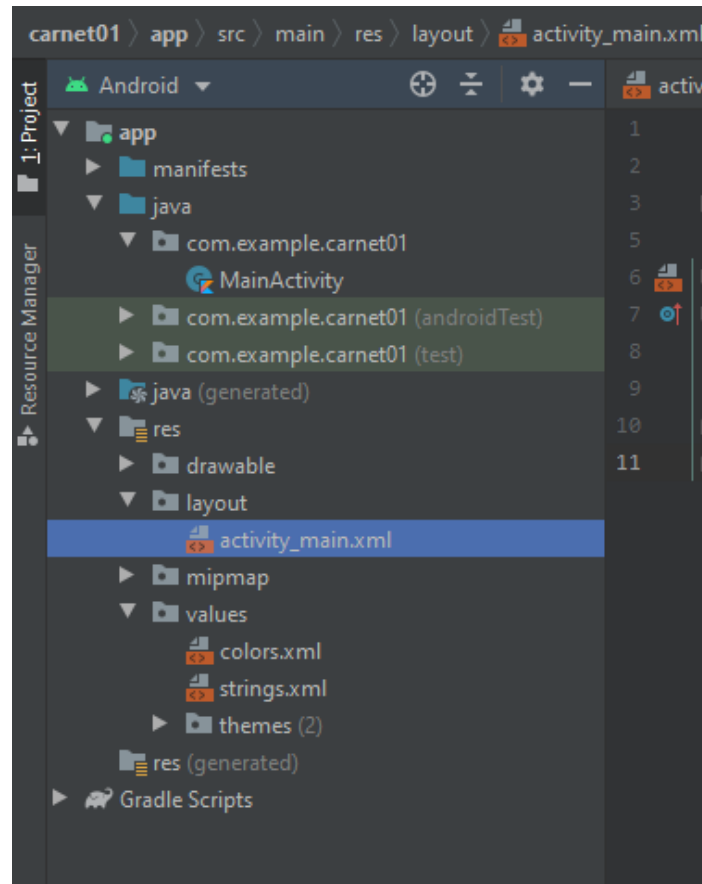
Luego volvemos a Android Studio y detenemos nuestra aplicación.



Reconociendo la estructura de nuestro proyecto:

En esta guía trabajaremos tanto con el MainActivity, su layout y los recursos de string.

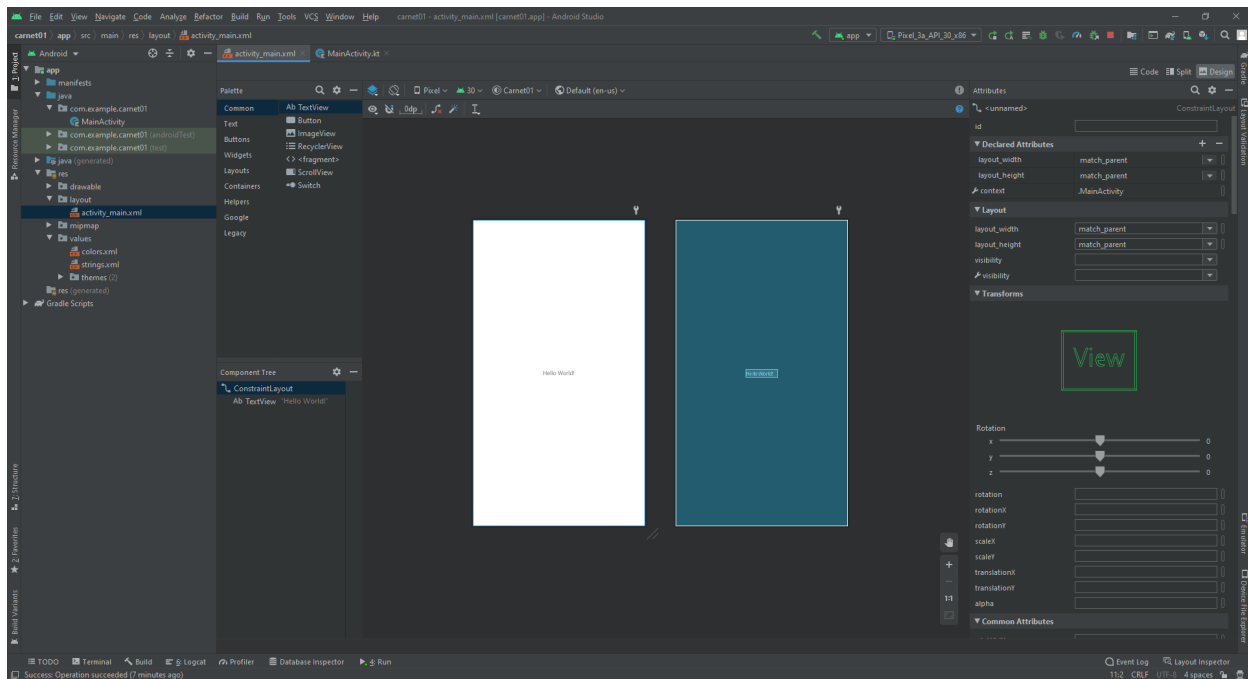
1. MainActivity.kt: Cómo su nombre lo indica, es la Activity principal, es la que se llamará cuando la aplicación se ejecute. Es donde se encuentra el código.
2. activity_main.xml: En este archivo definiremos la composición de las vistas, cada uno de los archivos bajo el folder de layout, están relacionados con una Activity. Se podría decir que son los archivos de maquetado de UI.
3. string.xml: En este archivo se definen todas las cadenas de texto que se ocuparan en la aplicación, la razón de hacerlo de esta manera es para facilitar la localización de las aplicaciones.



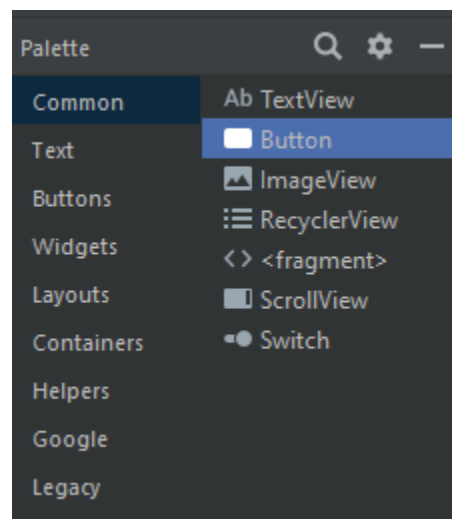
Para empezar a modificar nuestra aplicación, abriremos el archivo activity_main.xml, en el cual agregaremos los componentes de UI que vamos a utilizar. Al abrirlo se debe de ver como la siguiente imagen.

Visual Studio nos da la opción de ver este archivo de tres formas, la vista de **Diseño (Design)**, que es como se muestra en la imagen de abajo, la vista de **Código (Code)** en la que veríamos el xml puro y la vista **Dividida (Split)** en la que veríamos en la mitad de la pantalla el xml y en la otra mitad el diseño.

En la vista de Diseño, vemos que nos salen dos “pantallas”, una blanca y la otra azul. La blanca es una representación aproximada de cómo se va a ver la aplicación cuando ya esté corriendo y la azul se le llama vista de plano (Blueprint).

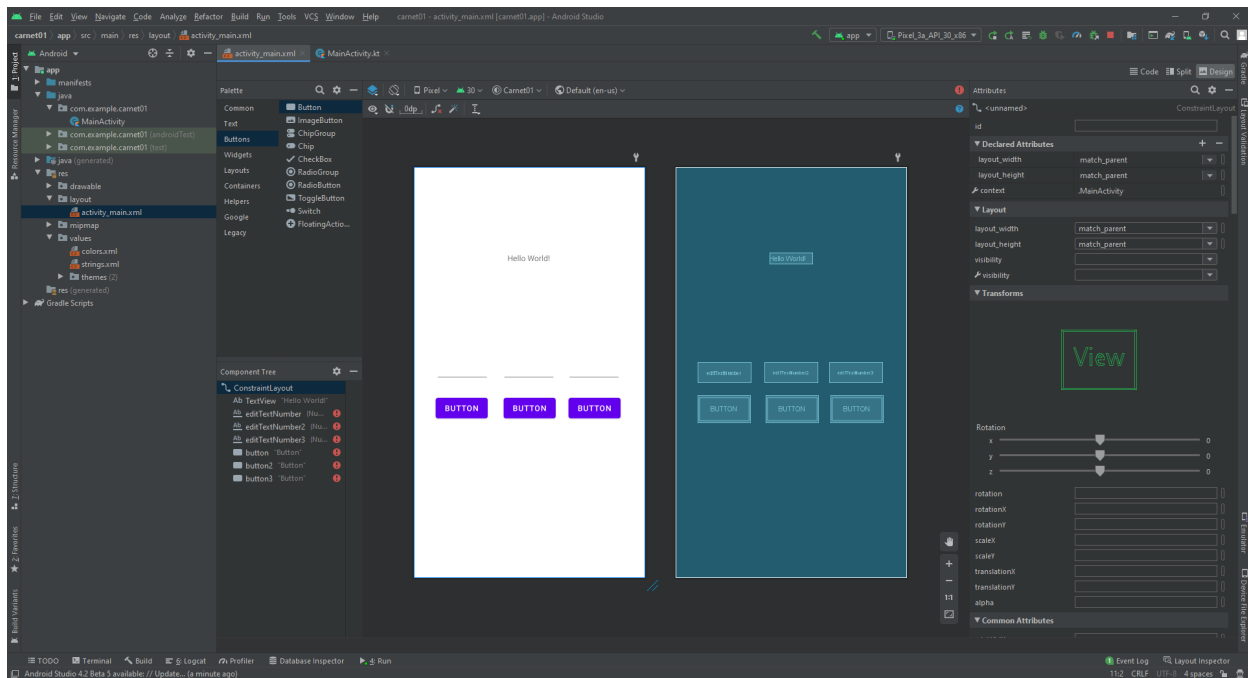


Desde la paleta de componentes o vistas, vamos a agregar **3 Botones (Buttons->Button)** y **3 Campos de texto numérico (Text->Number)**.



Para agregarlos arrastra el componente que deseas desde la paleta hacia la vista.

Trata de que queden más o menos como en la imagen siguiente, esto para diferenciarlos, no te preocupes que no queden exactamente igual, porque lo siguiente que haremos es posicionar cada componente correctamente dentro de la vista.

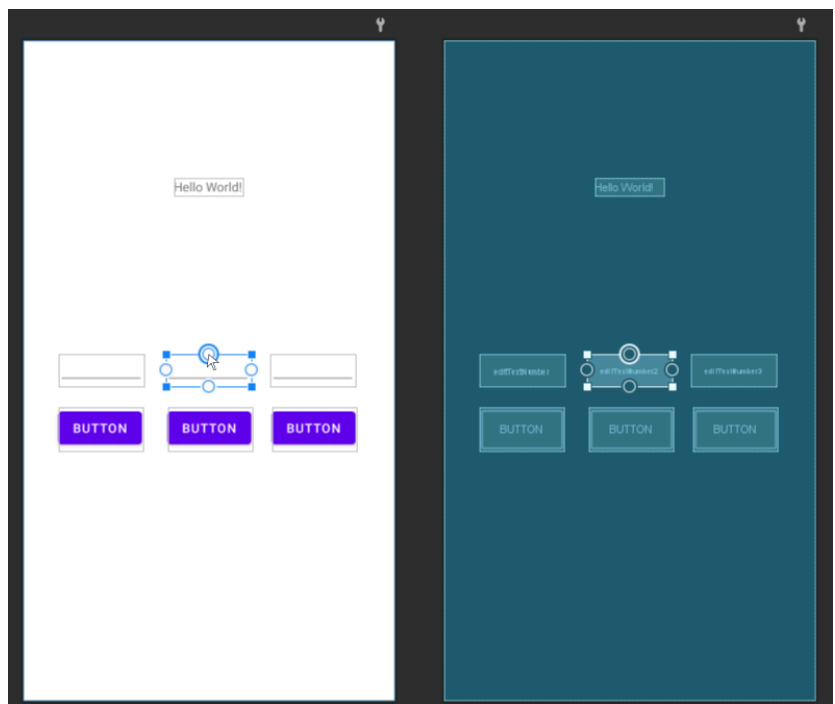


Por el momento cada componente que agregamos, tanto los botones como los campos de texto, están como “volando” dentro de nuestra vista, ya que no tienen ninguna restricción en su posicionamiento.

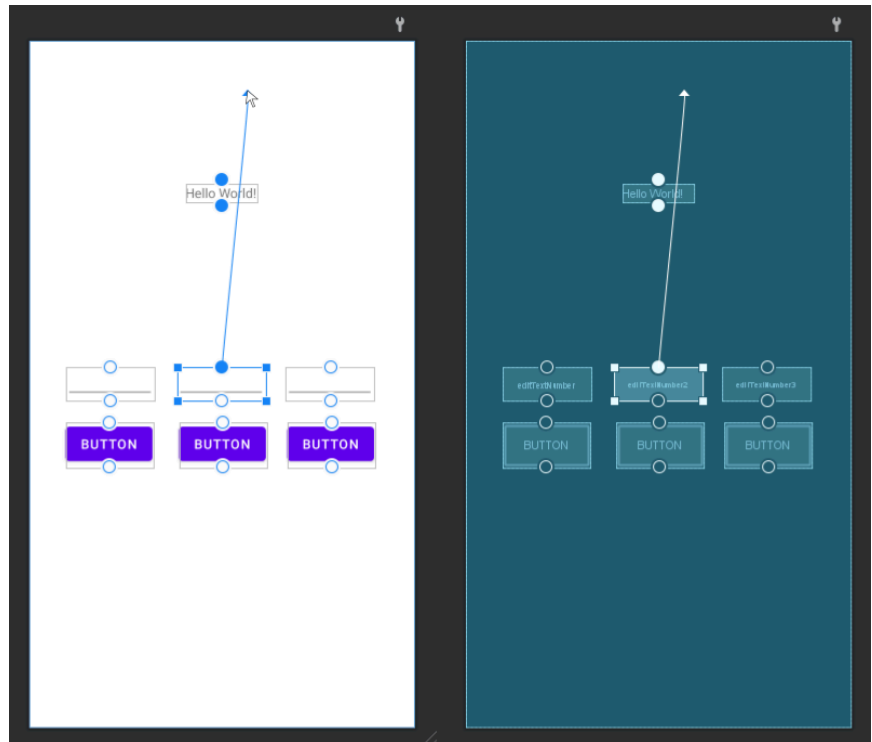
Las restricciones de posicionamiento le dicen a cada componente como, donde y con respecto a que se debe posicionar.

El primer componente que vamos a ubicar es **editTextNumber2**, que debería de ser el campo de texto del medio.

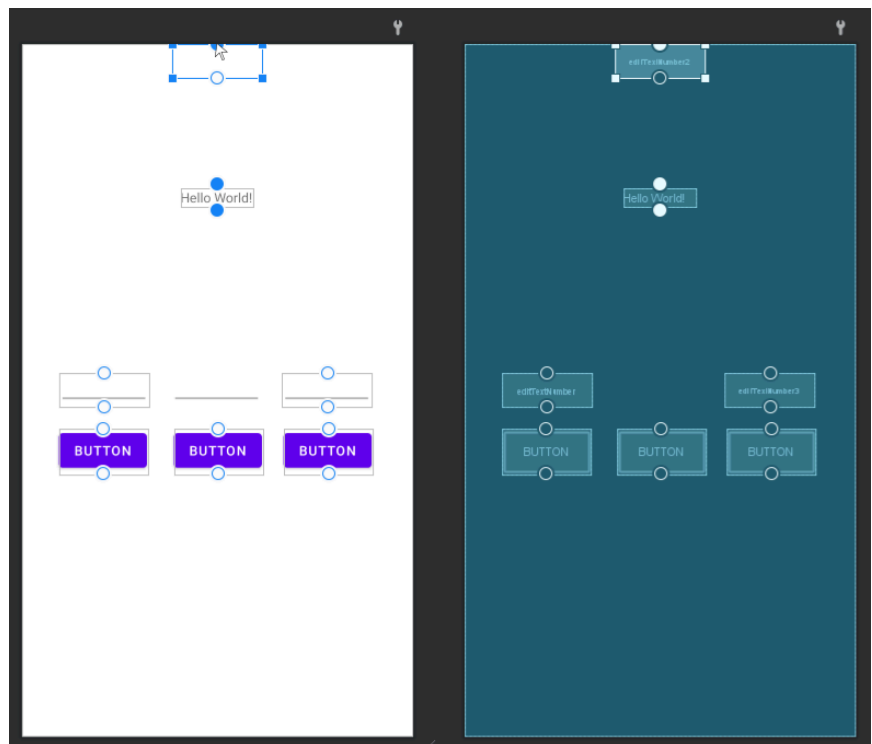
Seleccionamos el componente y mantenemos clic sobre el círculo que está en la parte superior.



Con el clic presionado
movemos el puntero hacia la
parte superior de nuestra vista.

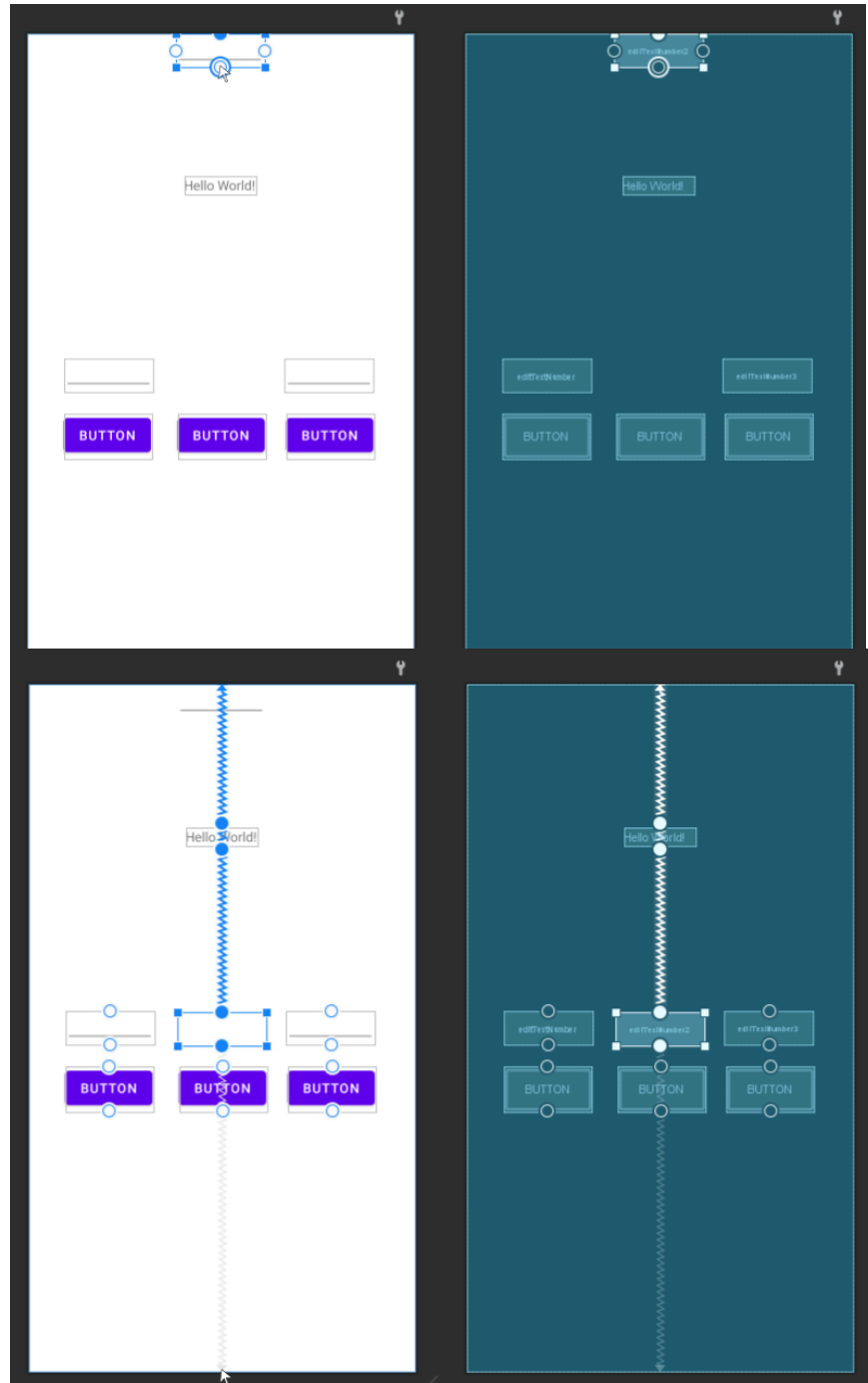


Al llegar al tope, verá como el
componente se mueve hacia
arriba, en ese momento ya
puede soltar el clic.



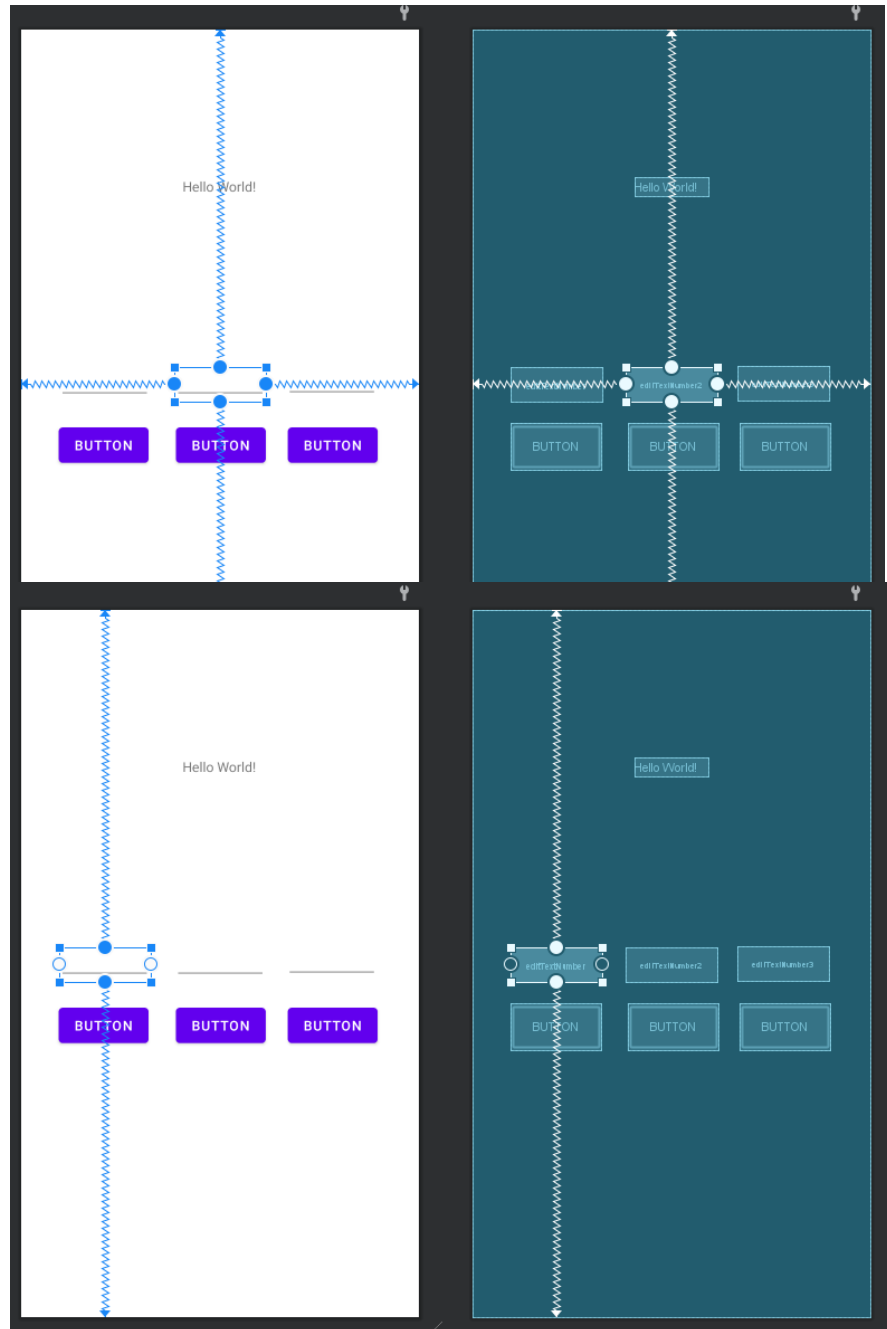
Ahora agregaremos la restricción para la parte de abajo, igualmente presionamos clic sobre el círculo inferior y lo arrastramos hasta el fondo.

Al llegar al fondo verás cómo el campo de texto se centra verticalmente. Esto es porque la posición del componente tiene restricciones, se podría decir que está atado tanto a la parte de arriba de la pantalla como la de abajo.



Agrega las restricciones horizontales, para que ese campo permanezca al centro de la pantalla siempre.

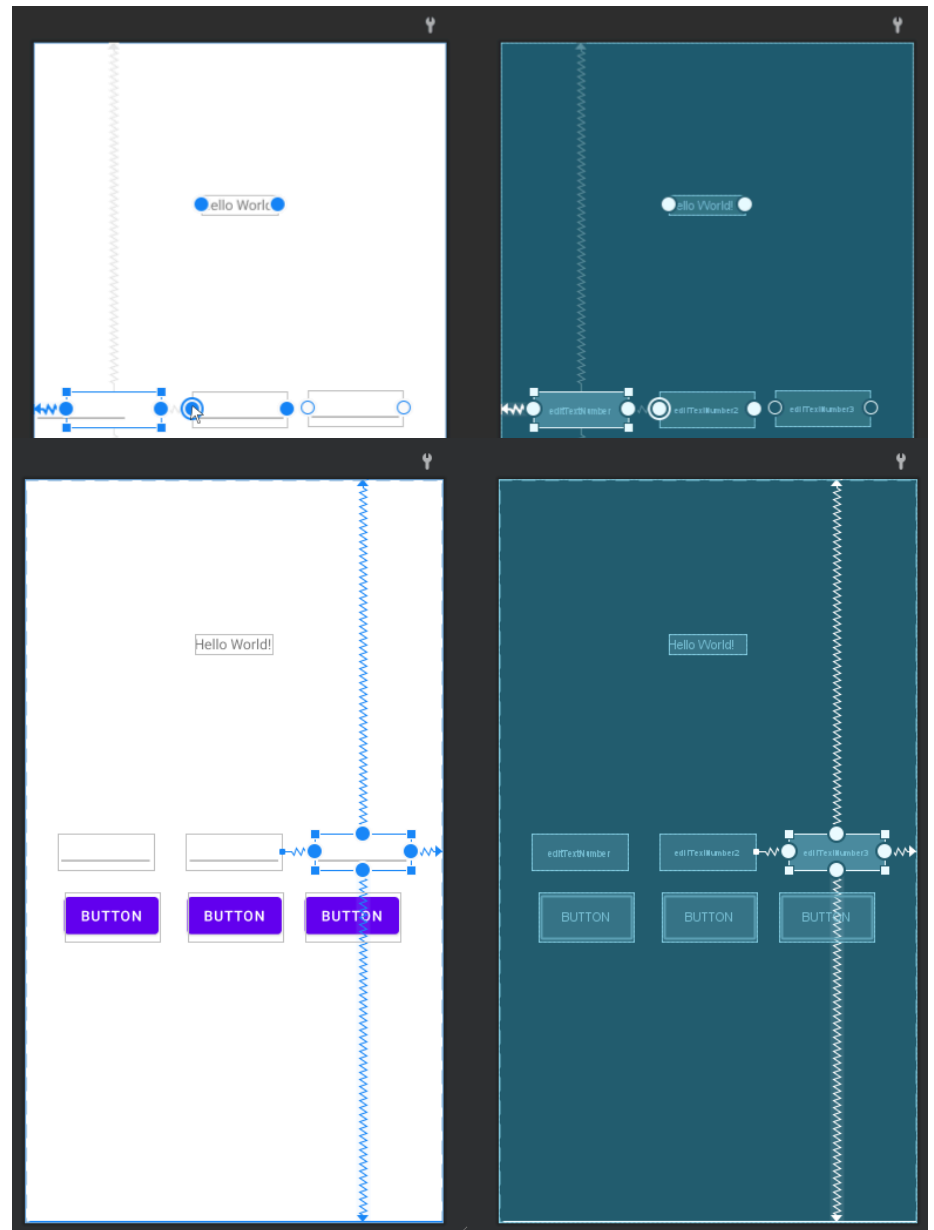
Agrega las restricciones verticales al campo de texto de la izquierda, déjalo verticalmente centrado.



Las restricciones horizontales serán en este caso un poco diferentes. A la izquierda tendrá su restricción hacia el lado izquierdo del teléfono, pero a la derecha se conectará con el lado

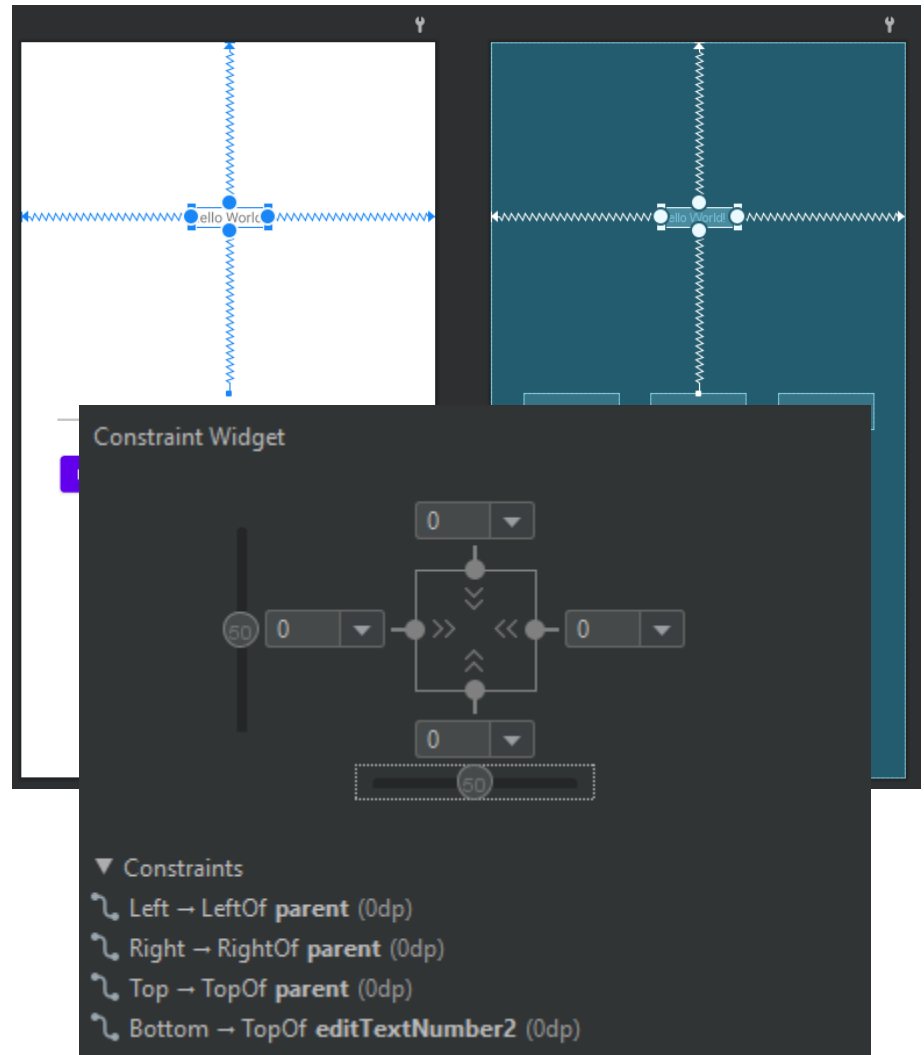
izquierdo del campo de texto del centro. De esta manera, el campo de texto quedará centrado en el espacio izquierdo.

Agrega las restricciones al campo de texto que falta, de la manera que muestra la imagen.

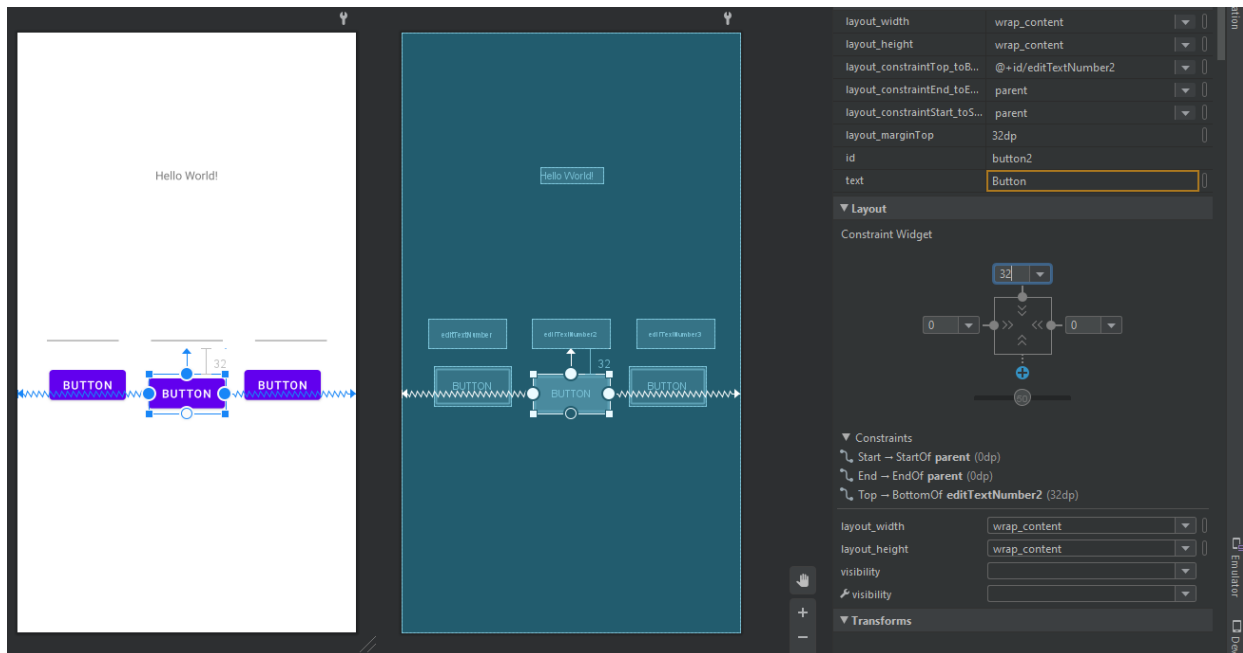


Agrega las restricciones a la etiqueta de texto.

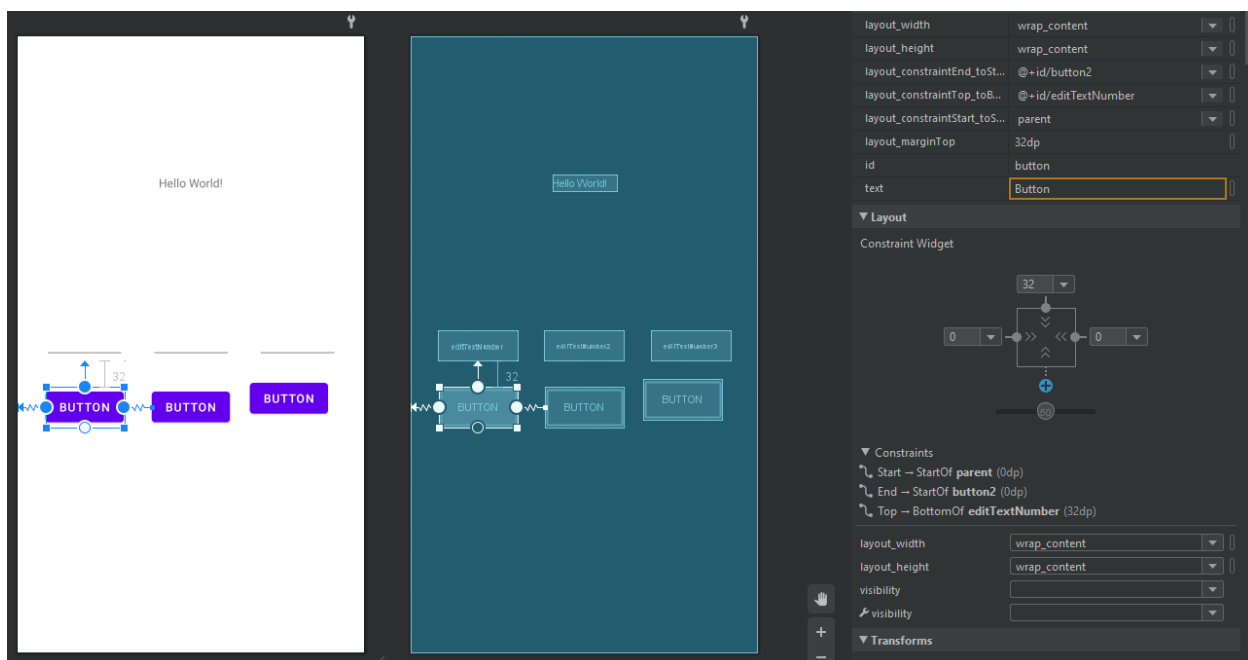
Para que quede centrado, asegúrate que los sesgos estén al 50% y todos los rellenos tengan 0 como valor.

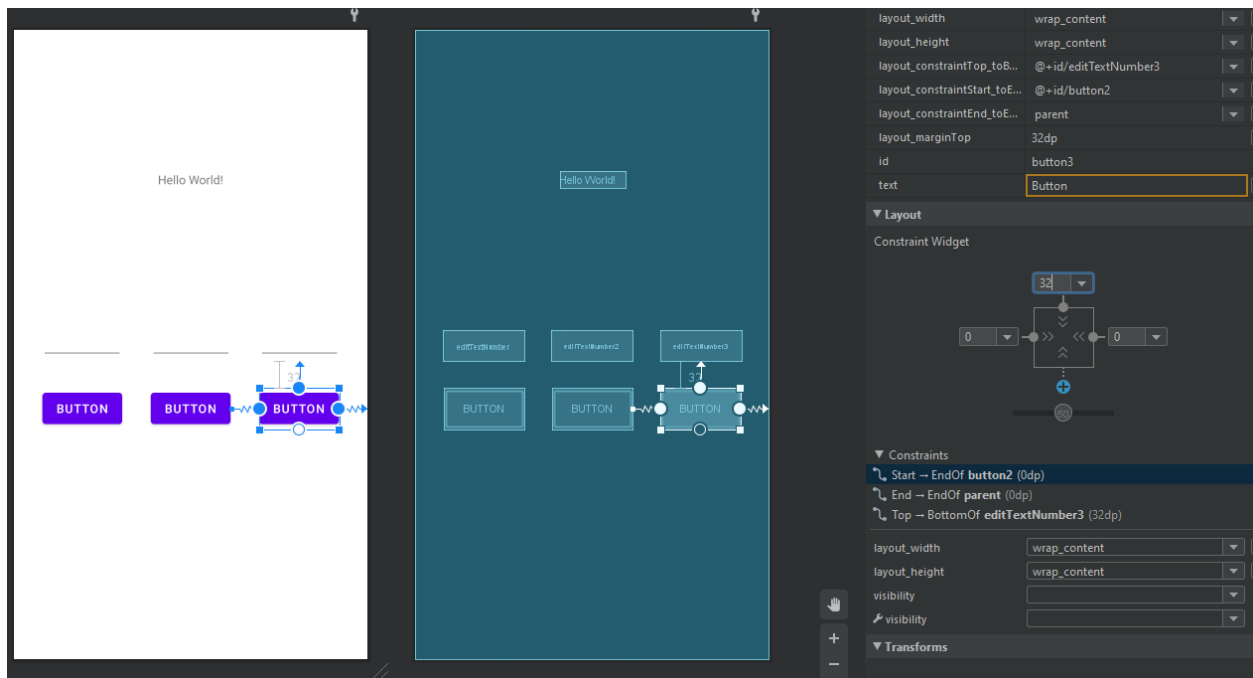


Ahora solo falta agregar las restricciones a los botones, en este caso a los botones no les agregaremos una restricción hacia abajo, pero le agregaremos un relleno de 32 en la parte de arriba.

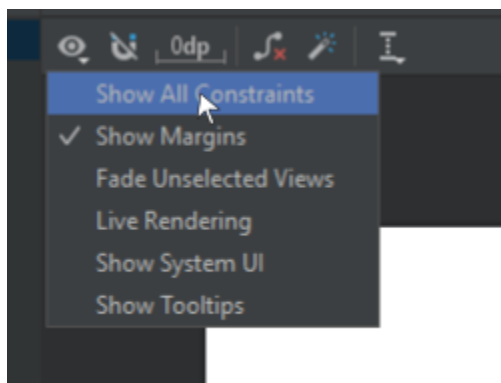


Agrega las restricciones a los botones faltantes.

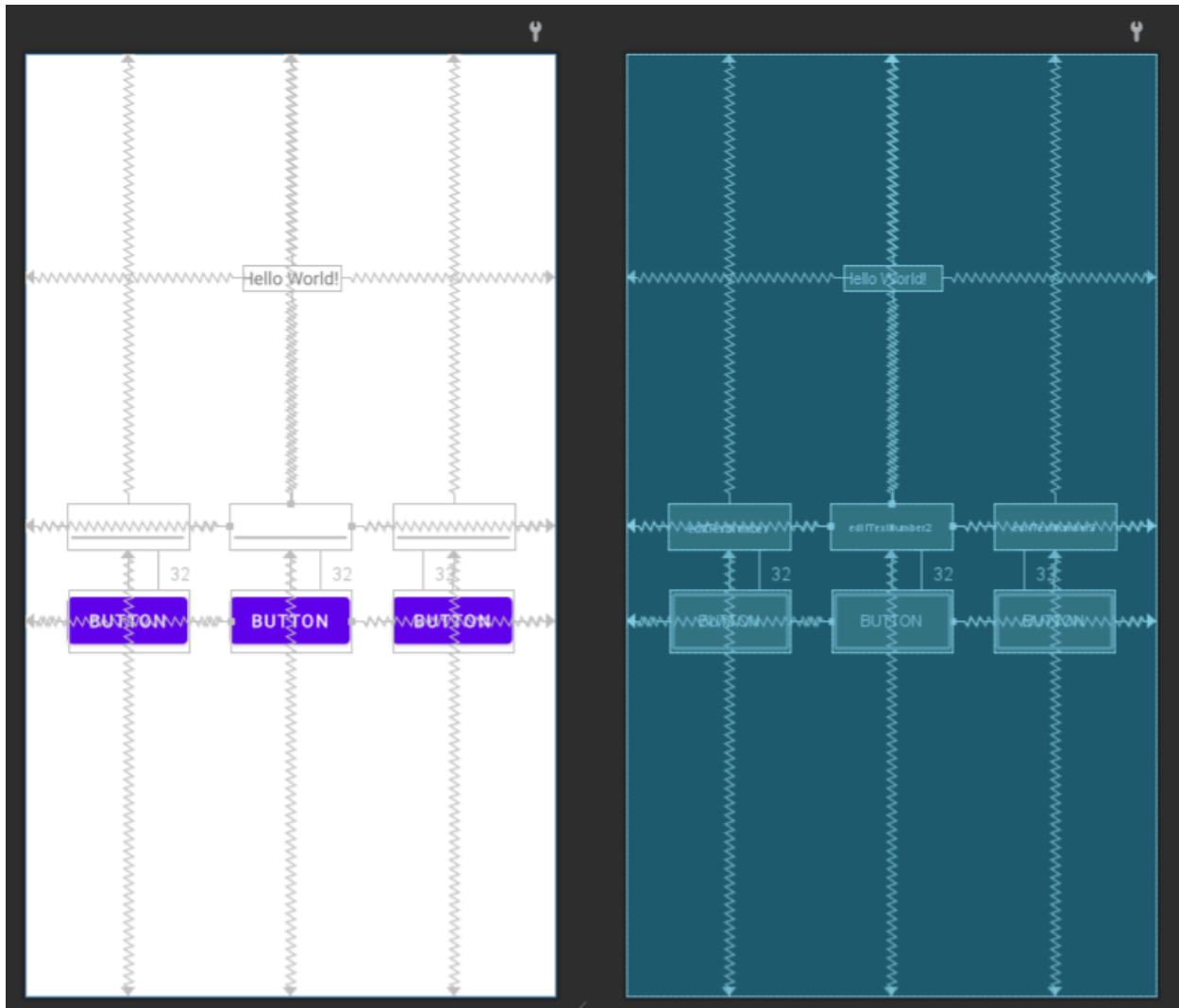




Habilita mostrar todas las restricciones (Show All Constraints)

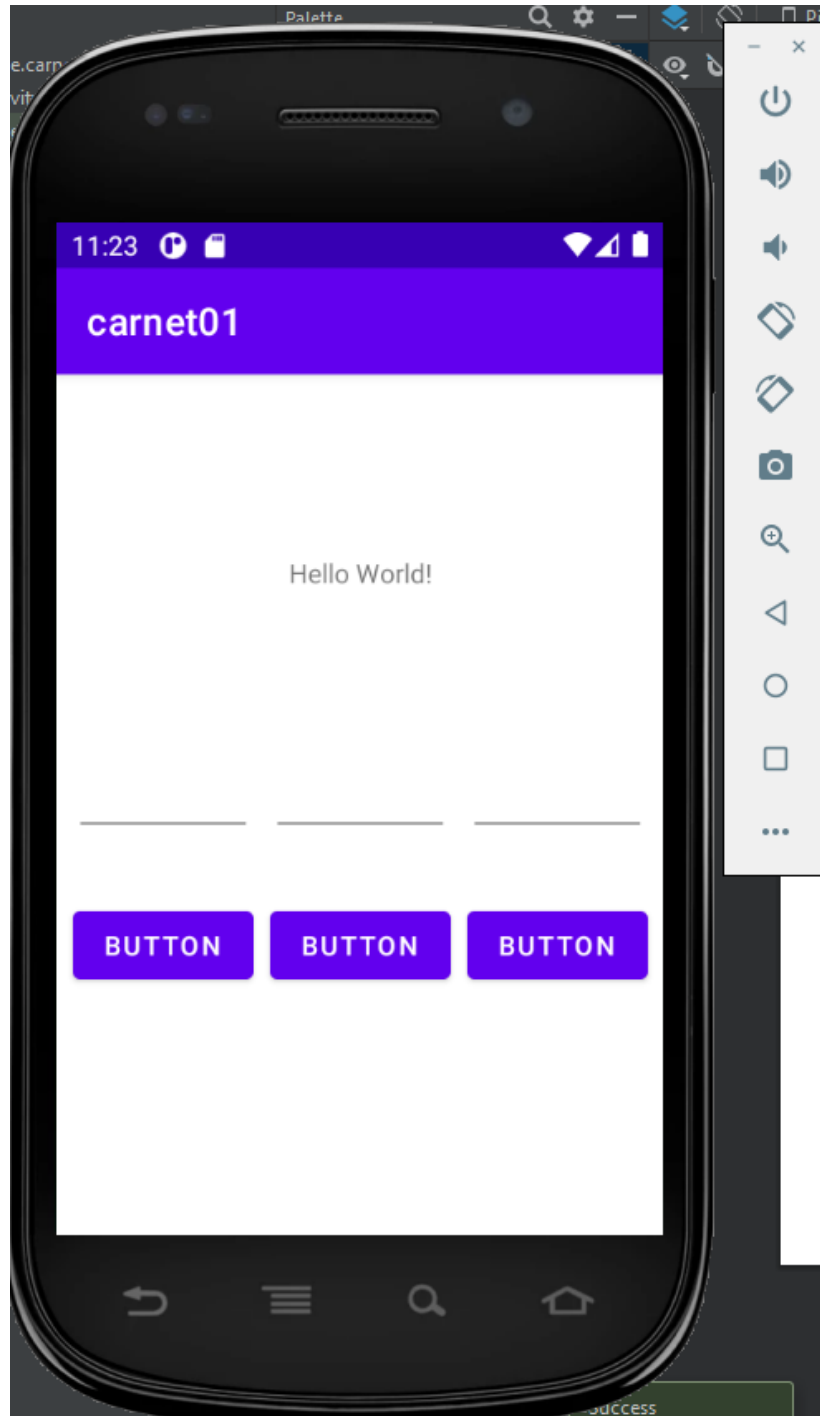


Las restricciones deberían de verse como lo muestran la imagen.

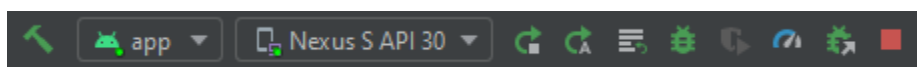


Con esto terminamos de configurar el posicionamiento de todos los componentes.

Corre la aplicación para confirmar que todo ha quedado posicionado correctamente.

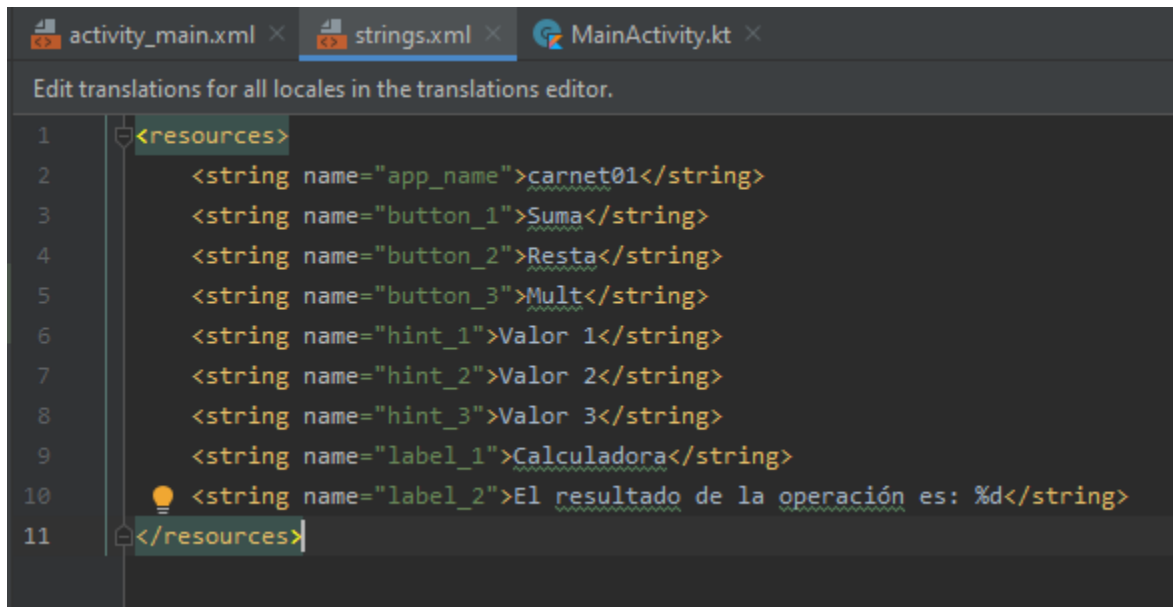


Recuerda detener la aplicación, para no consumir memoria RAM innecesaria, dando clic en el cuadro rojo.



Ahora modificaremos el archivo **strings.xml** para agregar todas las cadenas de texto que utilizaremos en nuestra aplicación.

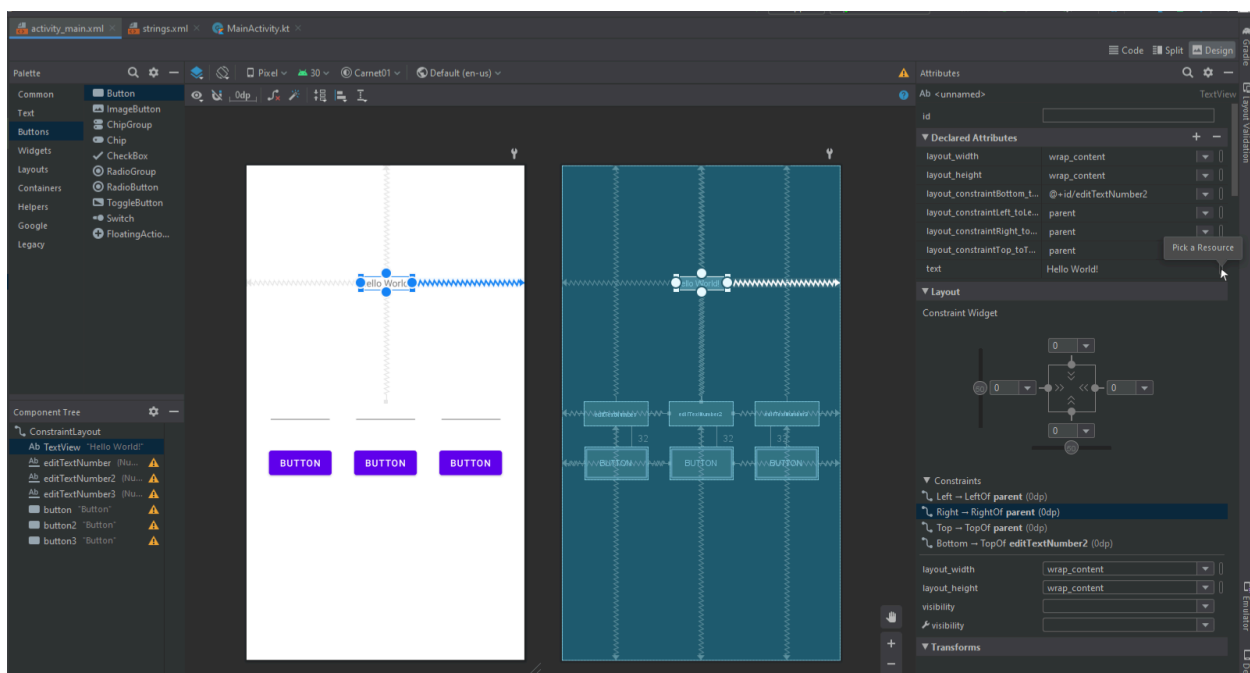
Modifica el archivo de manera que quede como en la imagen siguiente.



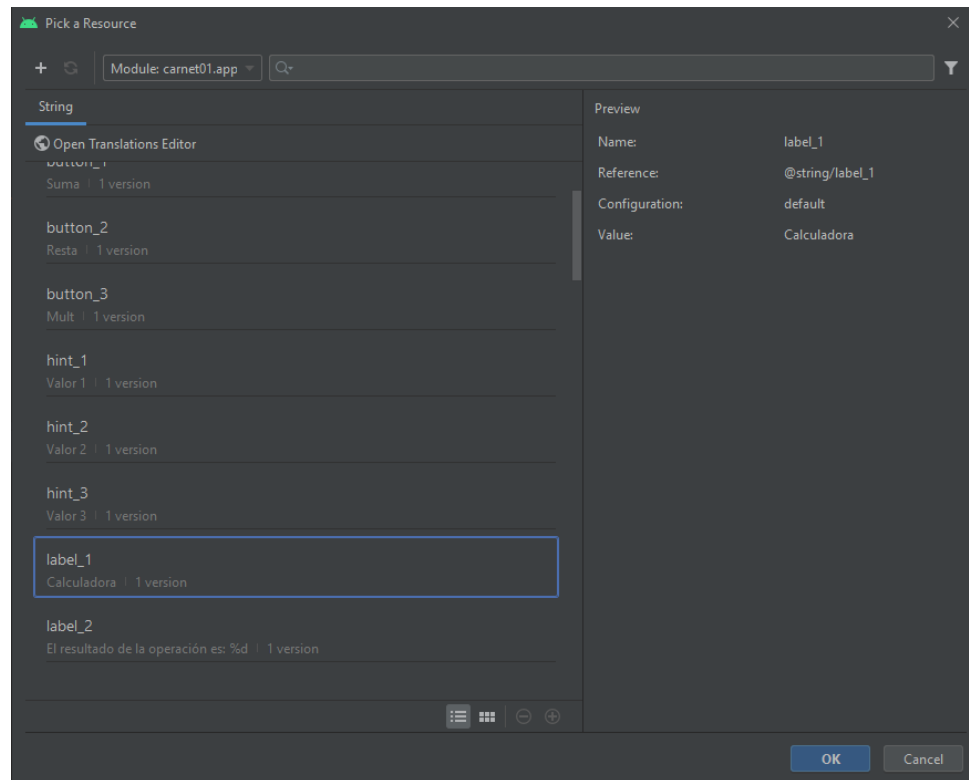
```

1  <resources>
2      <string name="app_name">carnet01</string>
3      <string name="button_1">Suma</string>
4      <string name="button_2">Resta</string>
5      <string name="button_3">Mult</string>
6      <string name="hint_1">Valor 1</string>
7      <string name="hint_2">Valor 2</string>
8      <string name="hint_3">Valor 3</string>
9      <string name="label_1">Calculadora</string>
10     <string name="label_2">El resultado de la operación es: %d</string>
11 </resources>
  
```

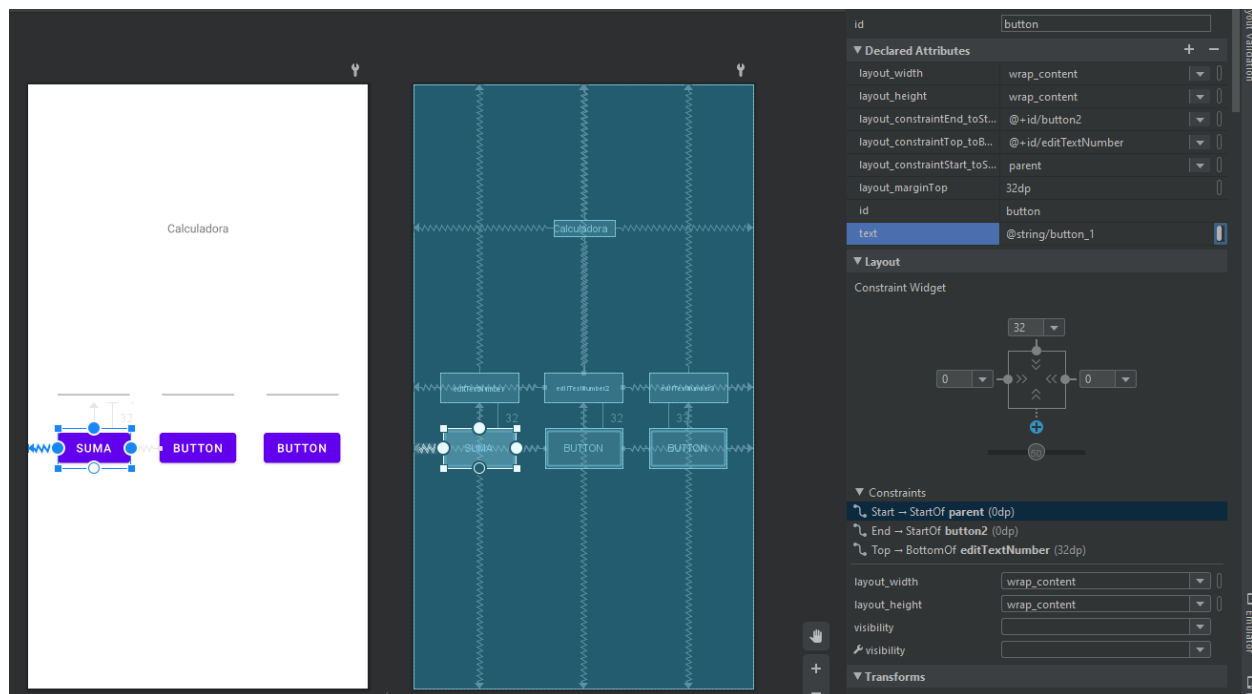
Para asignar la cadena de texto al componente regresamos al archivo de **activity_main.xml**, seleccionamos el componente de texto y en su propiedad **text**, damos clic en el botón rectangular a la derecha del campo de la propiedad.



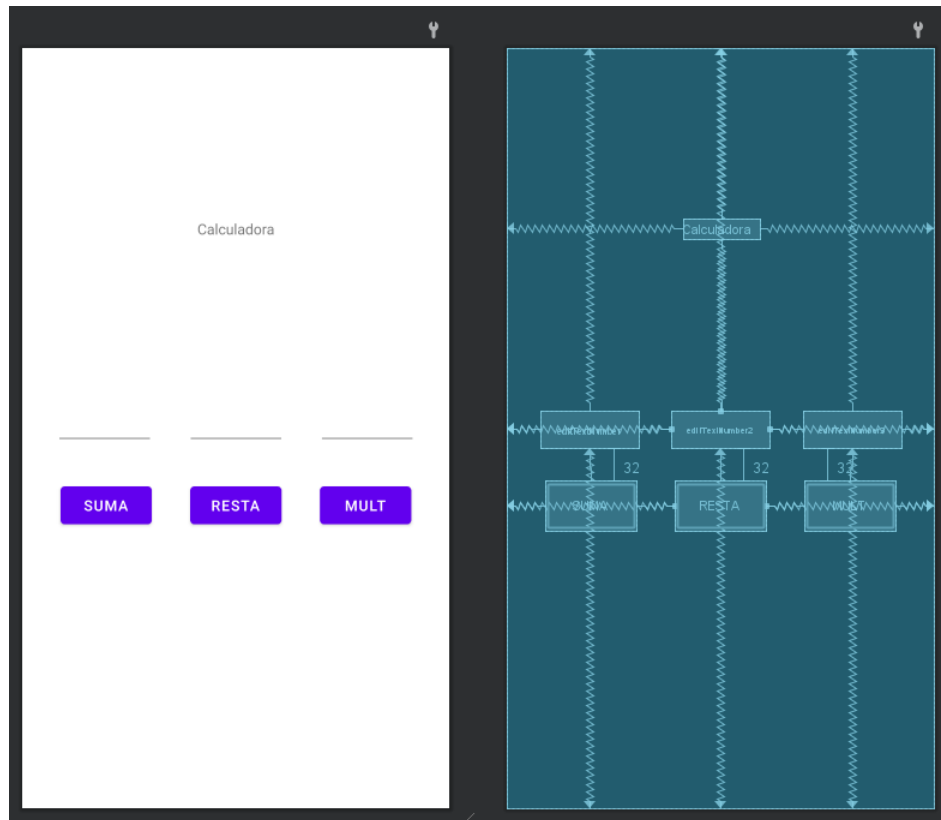
Se levantará la siguiente ventana, selecciona el string **label_1** y da clic en **OK**



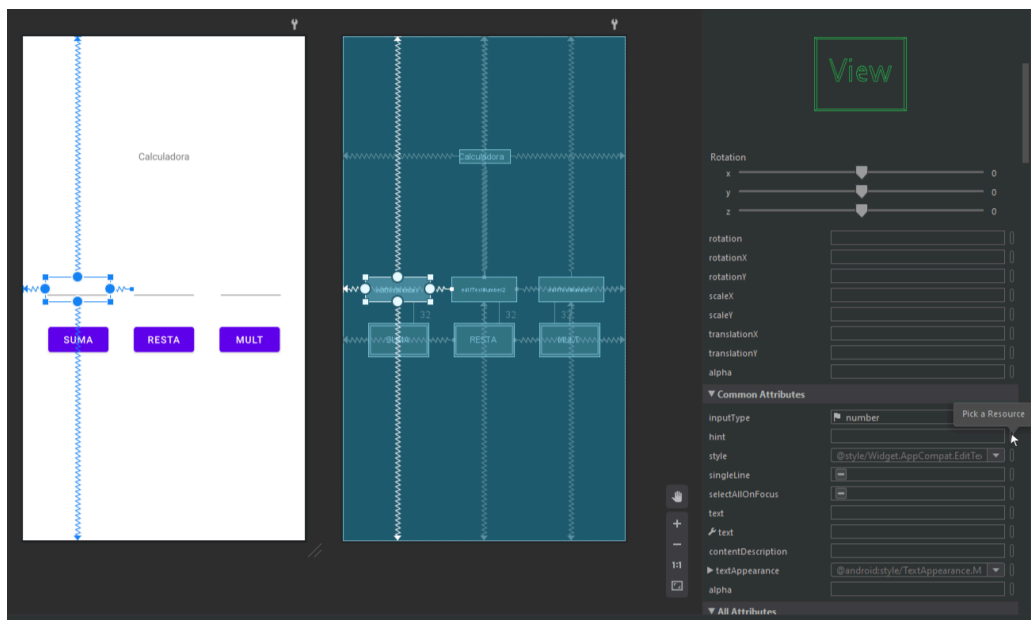
Asigna los string **button_1**, **button_2** y **button_3** respectivamente a la propiedad **text** de los botones.



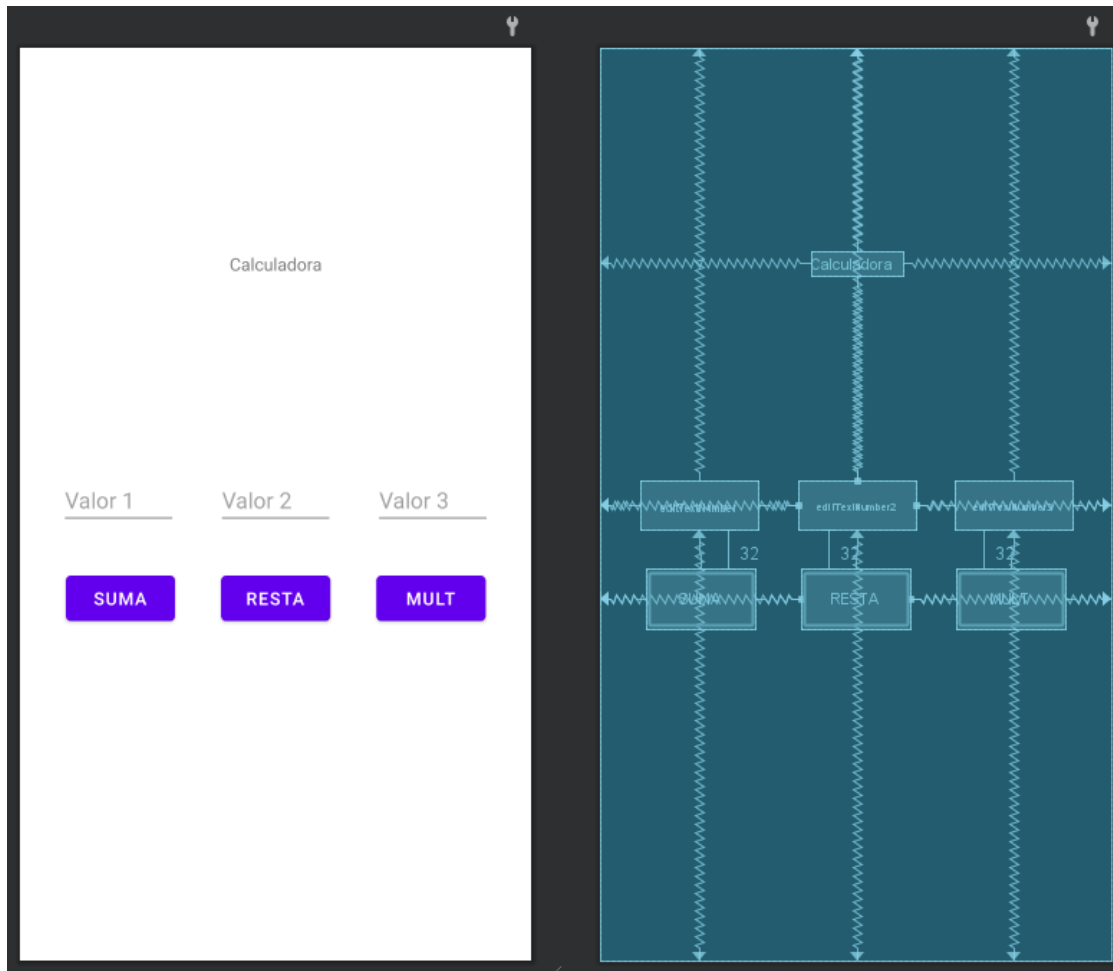
Al asignar los string en los botones, el UI de tu aplicación debería de verse de la siguiente manera



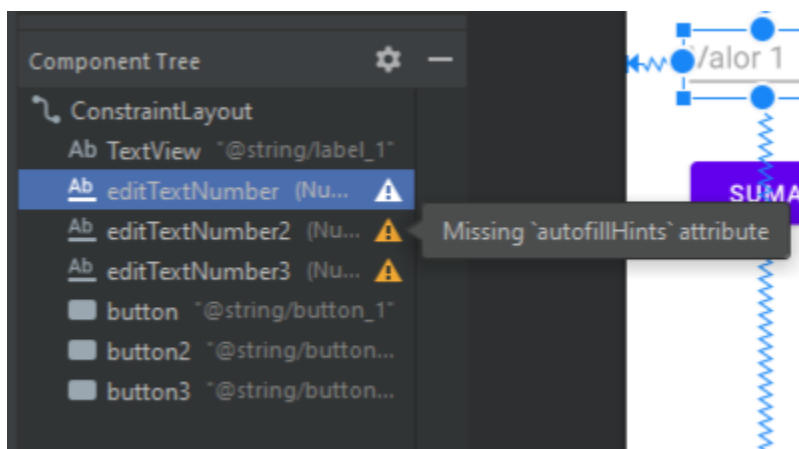
Asigna los string **hint_1**, **hint_2** y **hint_3** respectivamente a la propiedad **hint** de los campos de texto.



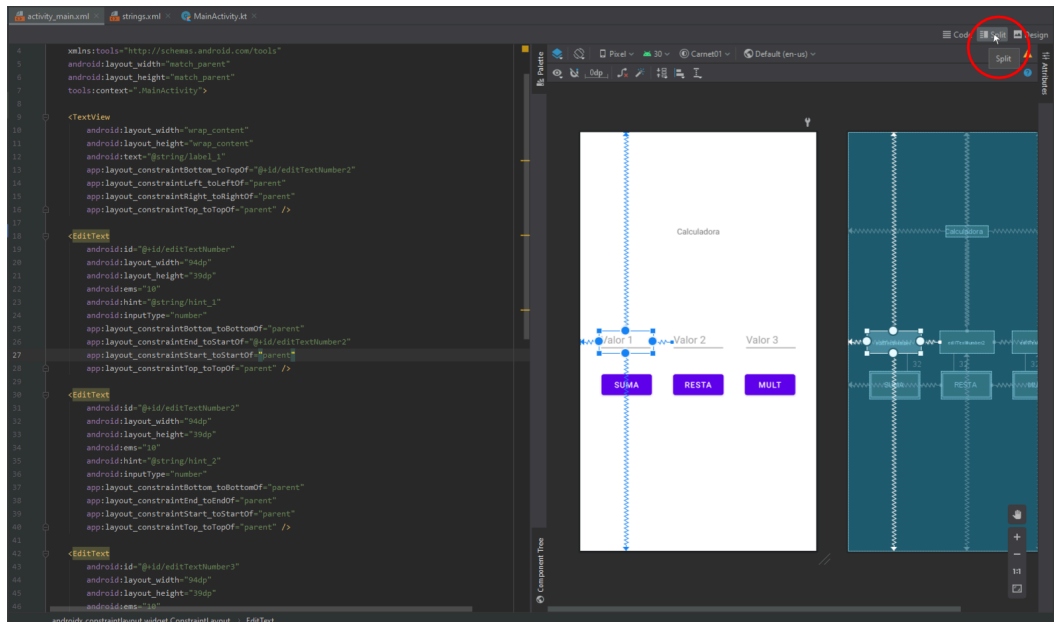
Al asignar los string en la propiedad de pista (hint) de los campos de textos, el UI de tu aplicación debería de verse así.



A pesar de ya haber agregado todos nuestros strings, los campos de texto aún muestran un ícono de advertencia, si colocamos nuestro cursor sobre el ícono, el IDE nos muestra la advertencia asociada.



Para poder solucionar esa advertencia, activaremos el modo **dividido (Split)**

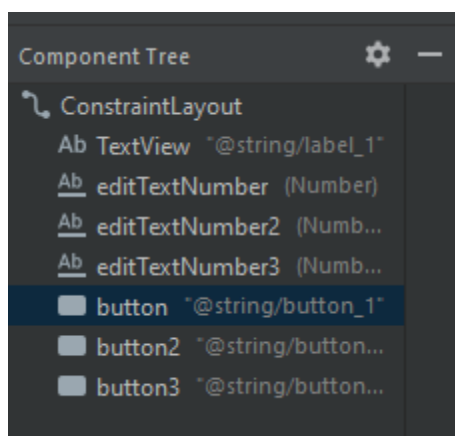


Agrega la siguiente etiqueta dentro de los **EditText**.

`android:importantForAutofill="no"`

Luego de agregarlas debería de quedar justo cómo lo muestra la imagen de la derecha, recuerda que solo se modificarán los **EditText**; los demás componentes quedarán igual de cómo estaban.

Al regresar a la vista de diseño, verifica que el ícono de advertencia ya no aparezca.



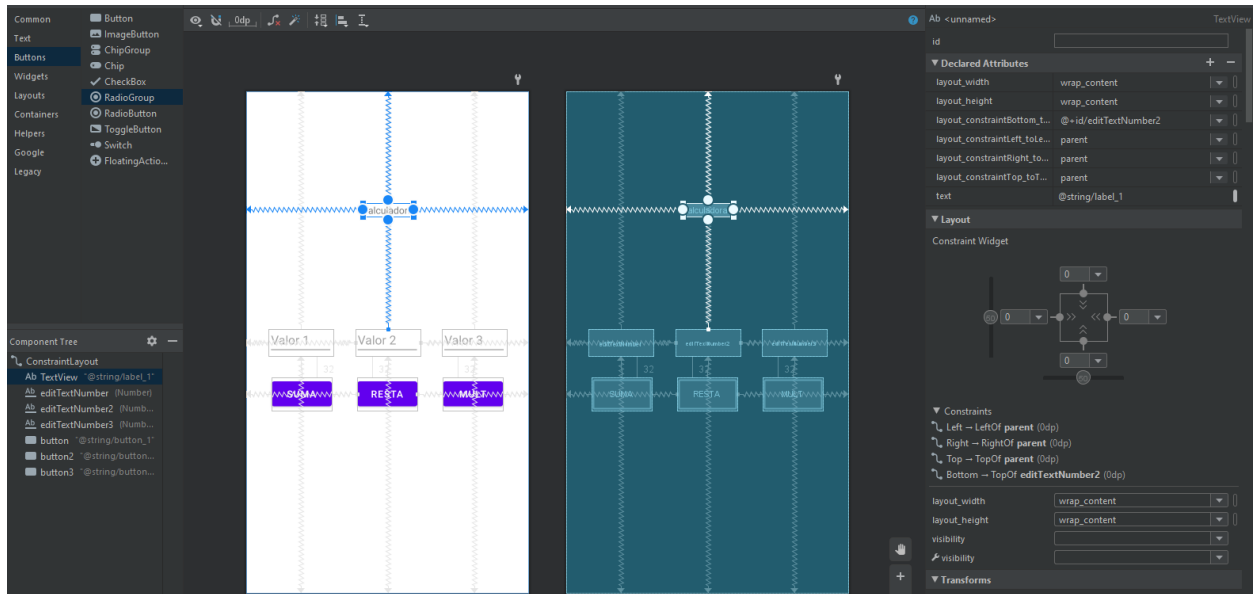
Si

```
<EditText
    android:id="@+id/editTextNumber"
    android:layout_width="94dp"
    android:layout_height="39dp"
    android:ems="10"
    android:hint="@string/hint_1"
    android:inputType="number"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/editTextNumber2"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:importantForAutofill="no" />

<EditText
    android:id="@+id/editTextNumber2"
    android:layout_width="94dp"
    android:layout_height="39dp"
    android:ems="10"
    android:hint="@string/hint_2"
    android:inputType="number"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:importantForAutofill="no" />

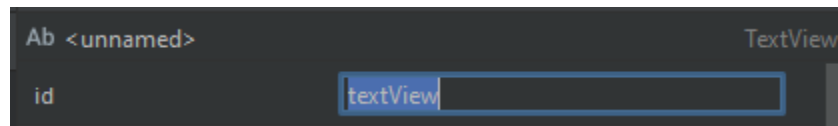
<EditText
    android:id="@+id/editTextNumber3"
    android:layout_width="94dp"
    android:layout_height="39dp"
    android:ems="10"
    android:hint="@string/hint_3"
    android:inputType="number"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@+id/editTextNumber2"
    app:layout_constraintTop_toTopOf="parent"
    android:importantForAutofill="no" />
```

seleccionamos el componente de texto, veremos que no tiene un **id** asociado, a diferencia de los demás componentes que agregamos que ya se les auto asignó un identificador.



Para poder hacer referencia a este componente desde nuestro código, necesitamos que tenga un identificador. Pocos casos serán en los que no se necesite agregar un identificador a los campos.

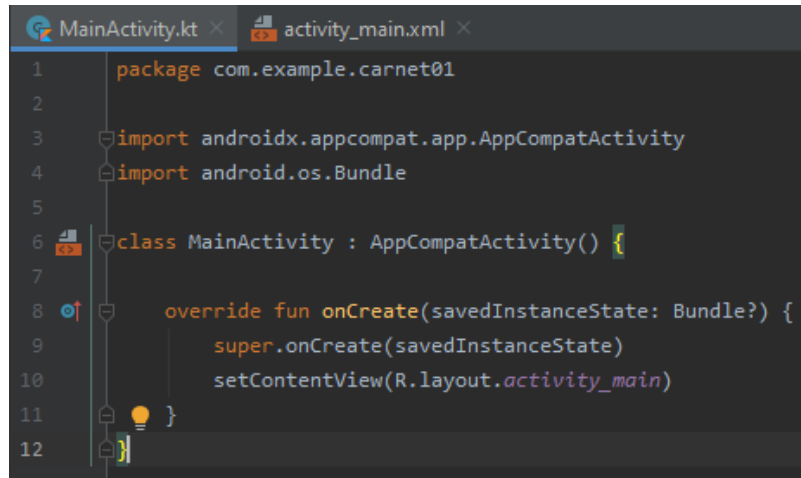
Como identificador para este componente utilizaremos **“textView”**



Asignando este identificador, ya tenemos todo listo para empezar a modificar el código y hacer nuestra aplicación funcional.

Abre el archivo **MainActivity.kt**, puedes ver que ya tiene un poco de código autogenerated.

La función **onCreate** se llama al crearse la actividad, y lo único que hace es cargar la vista de contenido, si te fijas, **activity_main** es la vista que hemos estado modificando.



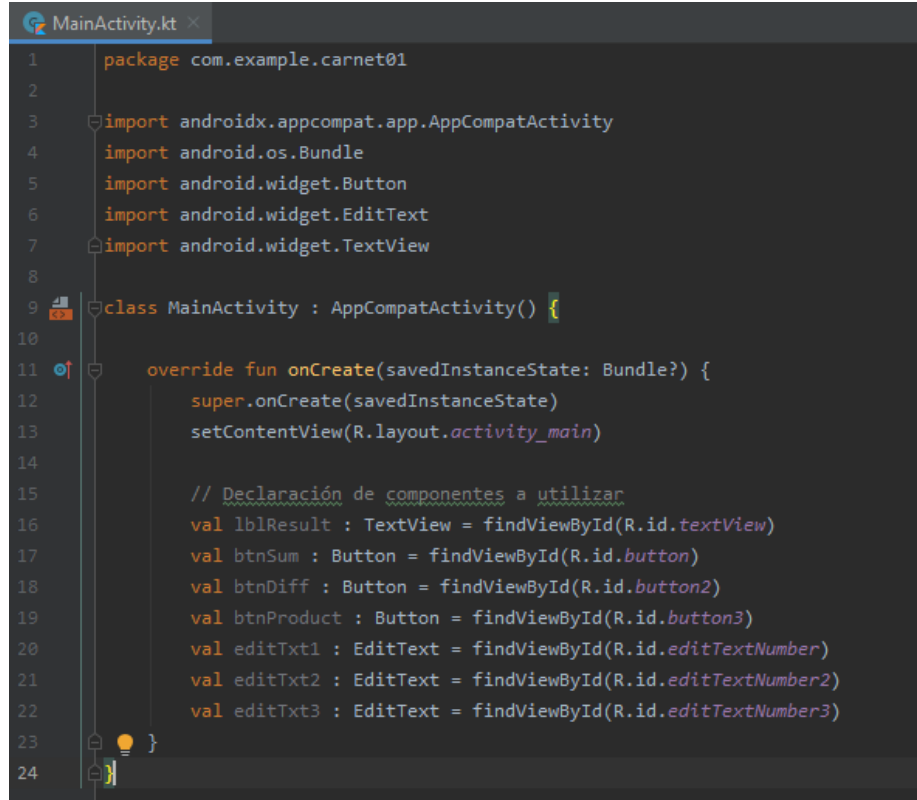
```

1 package com.example.carnet01
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5
6 class MainActivity : AppCompatActivity() {
7
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10        setContentView(R.layout.activity_main)
11    }
12 }
  
```

Primero obtengamos la referencia a cada uno de los componentes que creamos en la vista.

Utilizamos el id de los componentes para poder obtenerlos con la función **findViewById**.

Toma en cuenta que necesitarás importar los paquetes de cada uno de los diferentes componentes que vayas utilizando.



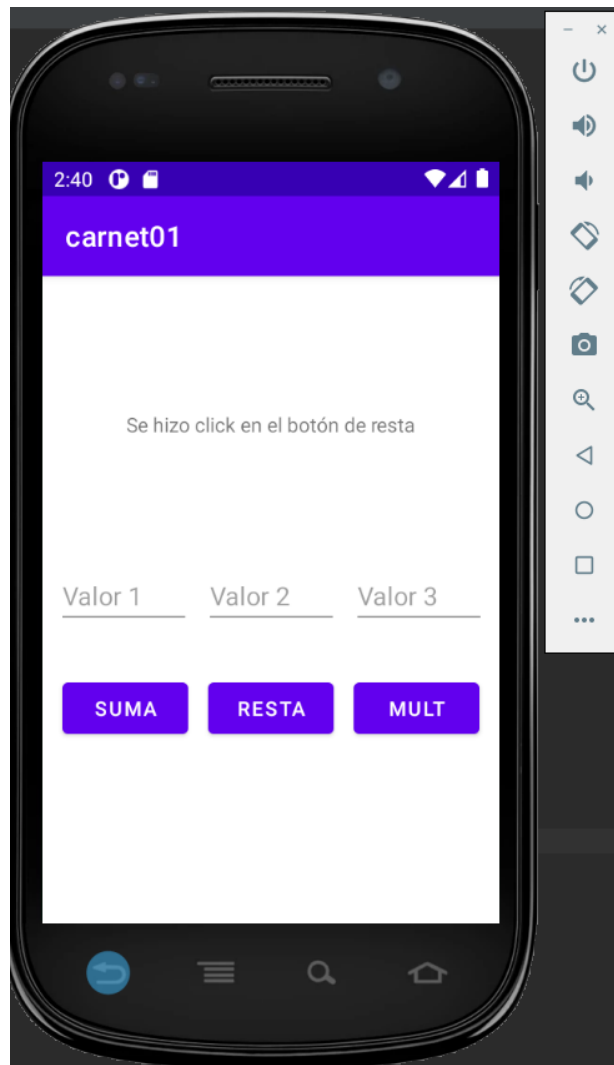
```

1 package com.example.carnet01
2
3 import androidx.appcompat.app.AppCompatActivity
4 import android.os.Bundle
5 import android.widget.Button
6 import android.widget.EditText
7 import android.widget.TextView
8
9 class MainActivity : AppCompatActivity() {
10
11     override fun onCreate(savedInstanceState: Bundle?) {
12         super.onCreate(savedInstanceState)
13         setContentView(R.layout.activity_main)
14
15         // Declaración de componentes a utilizar
16         val lblResult : TextView = findViewById(R.id.textView)
17         val btnSum : Button = findViewById(R.id.button)
18         val btnDiff : Button = findViewById(R.id.button2)
19         val btnProduct : Button = findViewById(R.id.button3)
20         val editTxt1 : EditText = findViewById(R.id.editTextNumber)
21         val editTxt2 : EditText = findViewById(R.id.editTextNumber2)
22         val editTxt3 : EditText = findViewById(R.id.editTextNumber3)
23     }
24 }
  
```

Ahora que ya tenemos la referencia a nuestros botones, asigna un listener al evento de click.

```
23
24 // Definición de eventos de click de los botones
25 btnSum.setOnClickListener { it: View!
26     lblResult.text = "Se hizo click en el botón de suma"
27 }
28
29 btnDiff.setOnClickListener { it: View!
30     lblResult.text = "Se hizo click en el botón de resta"
31 }
32
33 btnProduct.setOnClickListener { it: View!
34     lblResult.text = "Se hizo click en el botón de multiplicacion"
35 }
```

Corre tu aplicación, y prueba hacer clic en cada botón, revisa que el texto se cambie correctamente de acuerdo al botón que presionas.



Ahora modificaremos la función de cada botón, modifica el listener del botón de suma, con el siguiente contenido

```

25 btnSum.setOnClickListener { it: View!
26     val num1 = editTxt1.text.toString().toIntOrNull() ?: 0
27     val num2 = editTxt2.text.toString().toIntOrNull() ?: 0
28     val num3 = editTxt3.text.toString().toIntOrNull() ?: 0
29     lblResult.text = getString(R.string.Label_2, ...formatArgs: num1 + num2 + num3)
30 }

```

Modifica con lo siguiente el listener del clic del botón de resta.

```

32 btnDiff.setOnClickListener { it: View!
33     val num1 = editTxt1.text.toString().toIntOrNull()
34     val num2 = editTxt2.text.toString().toIntOrNull()
35     val num3 = editTxt3.text.toString().toIntOrNull()
36     val diff = when {
37         num1 != null -> {
38             num1 - (num2 ?: 0) - (num3 ?: 0)
39         }
40         num2 != null -> {
41             num2 - (num3 ?: 0)
42         }
43         else -> {
44             num3 ?: 0
45         }
46     }
47     lblResult.text = getString(R.string.Label_2, diff)
48 }

```

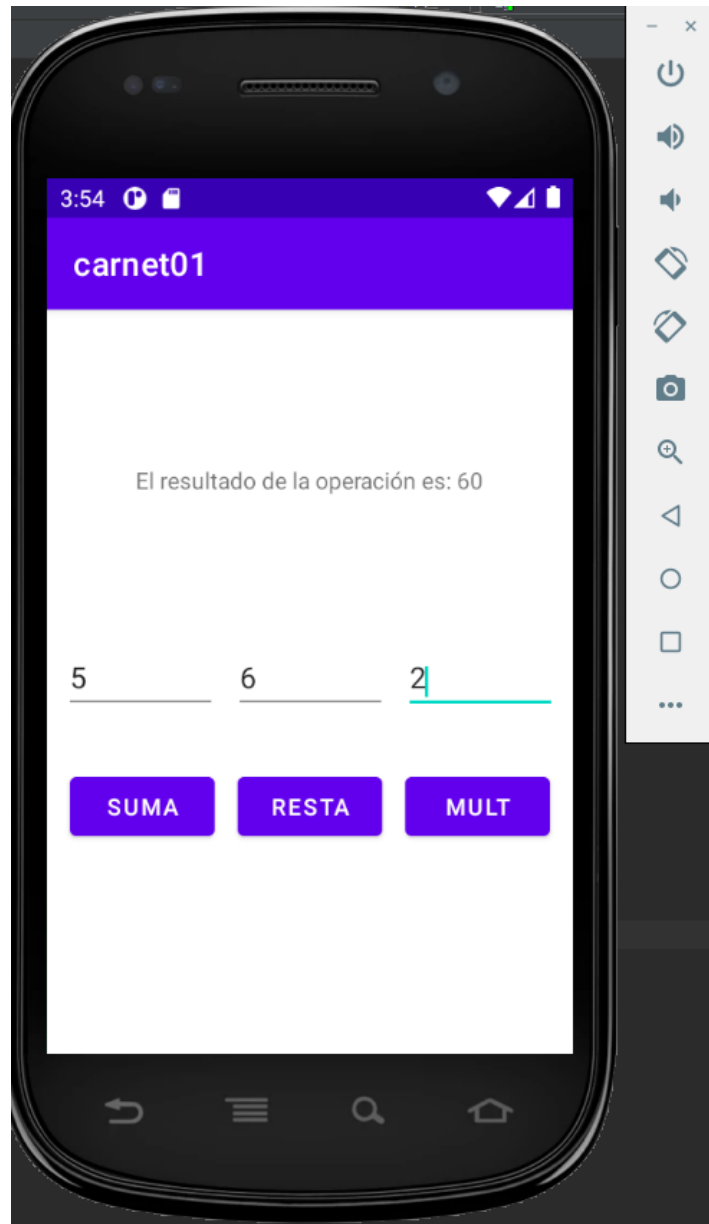
Y por último modifica el listener del clic del botón de multiplicación con el siguiente código.

```

50 btnProduct.setOnClickListener { it: View!
51     val num1 = editTxt1.text.toString().toIntOrNull()
52     val num2 = editTxt2.text.toString().toIntOrNull()
53     val num3 = editTxt3.text.toString().toIntOrNull()
54     if (num1 == null && num2 == null && num3 == null) {
55         lblResult.text = getString(R.string.Label_2, ...formatArgs: 0)
56     } else {
57         lblResult.text = getString(
58             R.string.Label_2,
59             ...formatArgs: (num1 ?: 1) * (num2 ?: 1) * (num3 ?: 1))
60     }
61 }

```

Corre y prueba tu primera aplicación para Android, desarrollada utilizando Kotlin como lenguaje.



Tareas opcionales:

- Agrega un botón más e implementa la funcionalidad de división.
- Agrega soporte para números con punto flotante en tu aplicación.

NOTA:

Cualquier duda o problema consúltalo con tu tutor.