

Instalación Android-Studio.

- Windows

<https://developer.android.com/studio>

- Ubuntu:

Instalar previamente: lib32z1 lib32ncurses5 lib32stdc++6 qemu-kvm libvirt-bin

<https://developer.android.com/studio>

Descarga y descomprimir.

Ejecutar desde el terminal

```
administrador@administrador-VirtualBox: ~/Descargas/android-studio/bin
administrador@administrador-VirtualBox:~$ cd Descargas/
administrador@administrador-VirtualBox:~/Descargas$ cd android-studio/
administrador@administrador-VirtualBox:~/Descargas/android-studio$ cd bin
administrador@administrador-VirtualBox:~/Descargas/android-studio/bin$ ./studio.sh
```

cd Descargas/-----Carpeta donde hemos guardado la descarga de Android-Studio

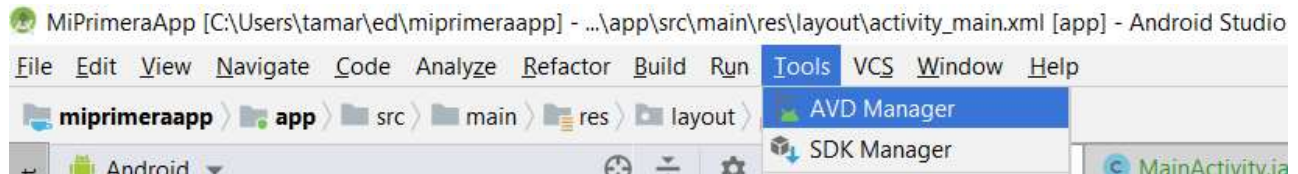
cd android-studio

cd bin

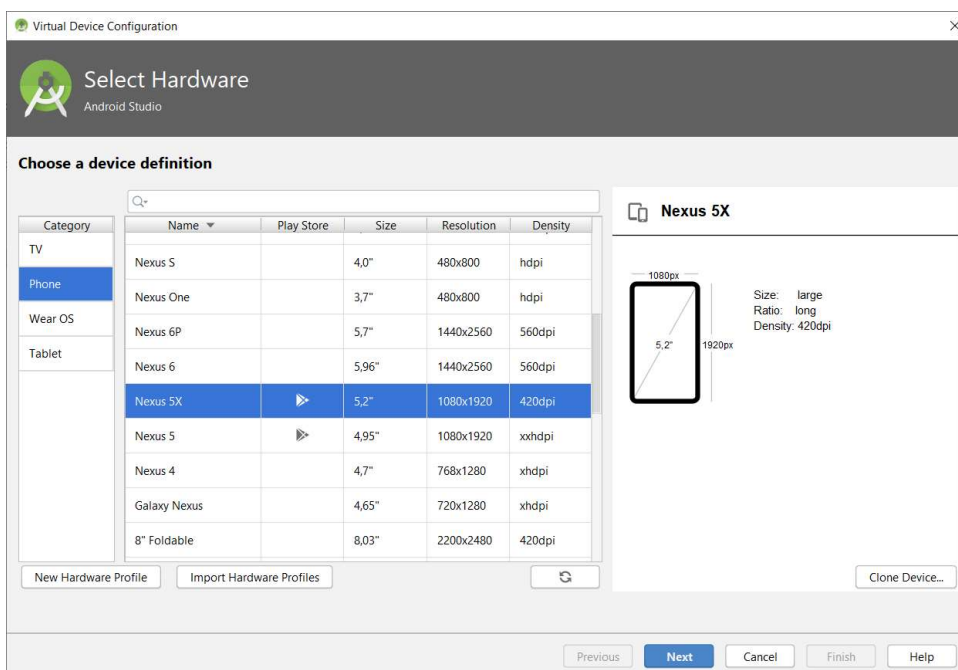
./studio.sh

Crear Terminal Virtual:

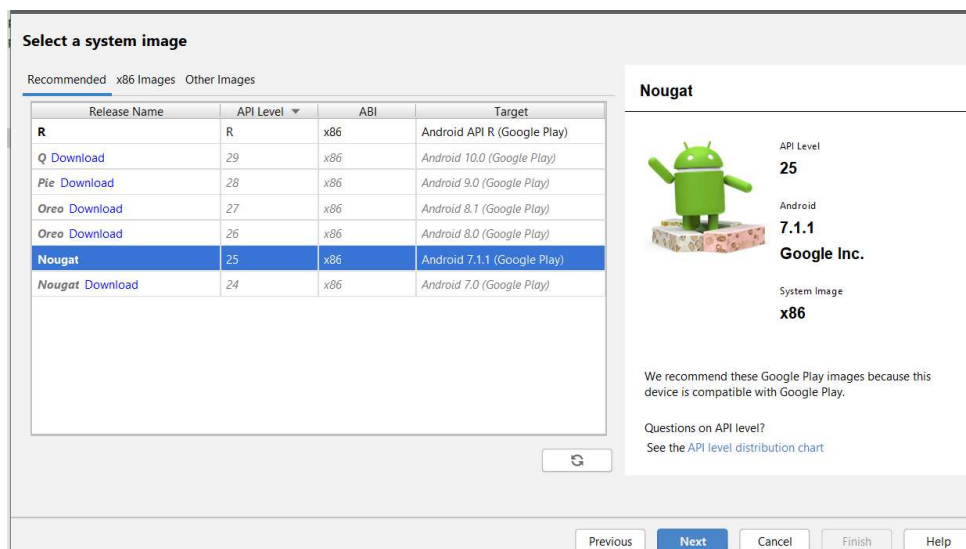
La creación de un terminal nos va a servir para poder probar nuestra aplicación.



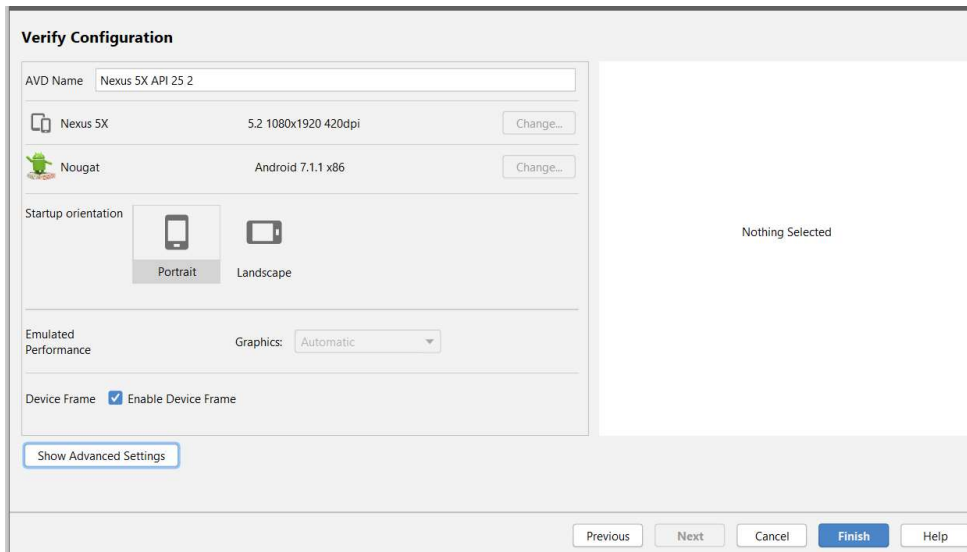
En nuestro caso elegimos el Nexus 5X.



En el apartado de recomendaciones nos descargamos la Nougat 25.

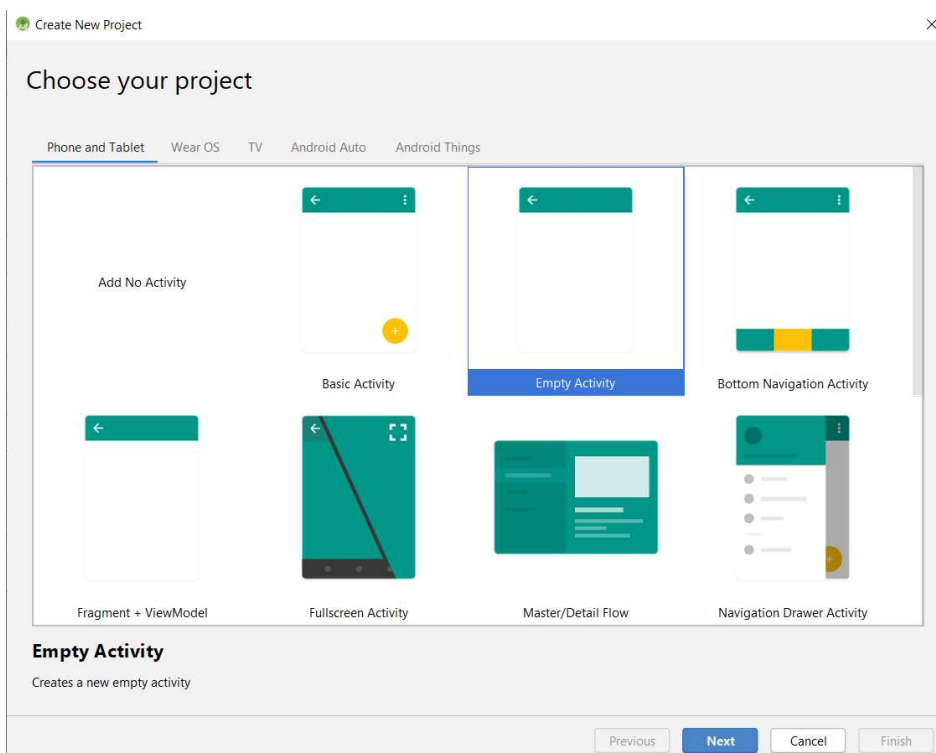


Y elegiremos Enable Dive Frame para que nos muestre la carcasa de nuestro dispositivo.



Crear Proyecto Nuevo:

Elegiremos Empty Activity, hay que saber que una **activity** es cada pantalla que van a formar parte de nuestra aplicación.



En la siguiente pantalla rellenaremos los siguientes campos:

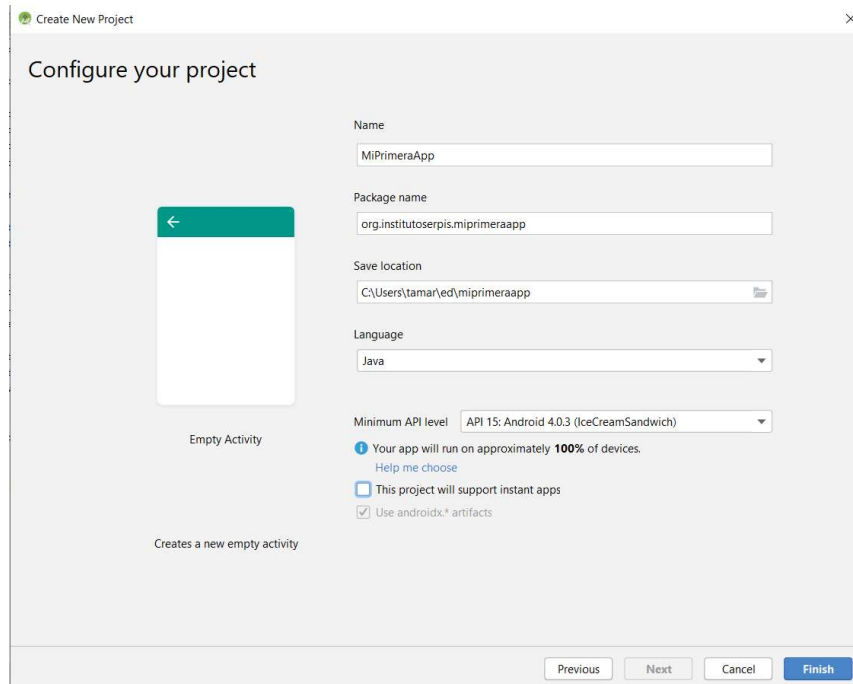
Name: [MiPrimeraApp](#) (Nombre de nuestra Aplicación)

Package name: [org.institutserpis.ed.miprimeraapp](#)

Save Location: Ruta donde vamos a guardar nuestro proyecto

Language:Java (Language de programación a utilizar)

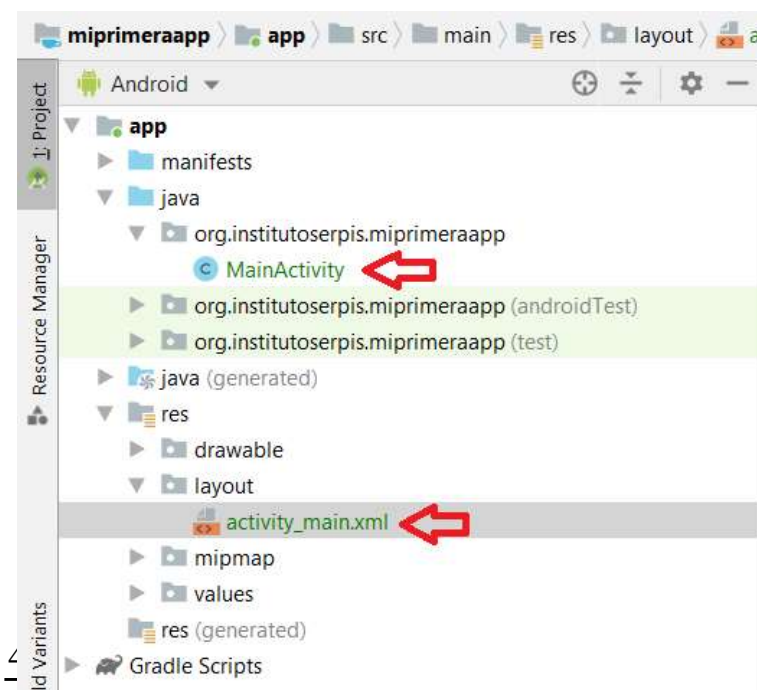
SDK:API 15:Android 4.0.3 (iceCreamSandwich).En la parte inferior aparece el porcentaje de compatibilidad.



Una vez creado nuestro proyecto debemos tener claro que hay dos partes bien diferenciadas.

1. **La parte gráfica** que corresponde al archivo **“activity_main.xml”**.
2. **La parte lógica** que corresponde al archivo **“MainActivity.java”**.

Estos archivos los podemos encontrar en la siguientes rutas:



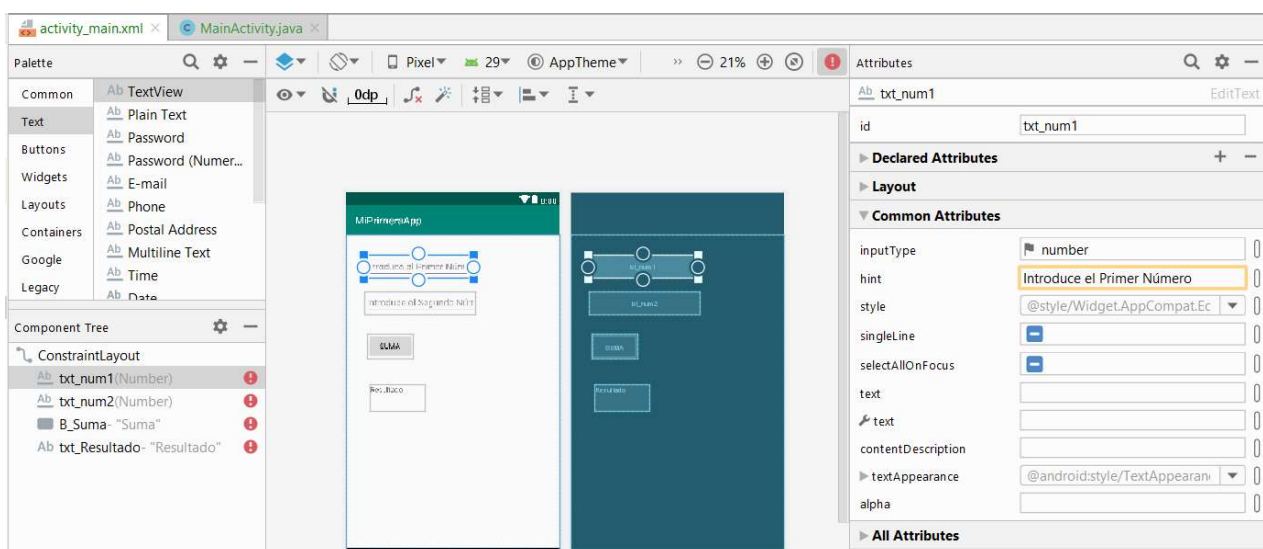
Parte grafica corresponde a la parte de diseño de nuestra APP y se distribuye en cuatro partes:

1-Palette: Parte donde nos aparecen todos y cada uno de los componentes que podemos agregar a nuestra activity.

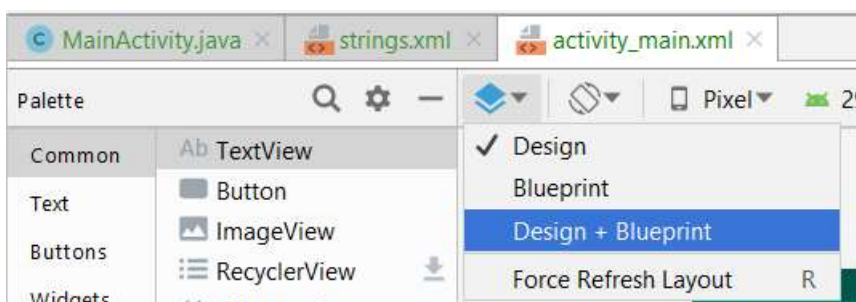
2-Component Tree: Muestra todos los componentes que hemos añadido a nuestra aplicación.

3-Attributes: Aparecen todos los atributos relacionados con los objetos que hay en nuestra APP.

4-En la parte central podemos ver las dos diferentes vistas (Design y Blueprint) de nuestra aplicación. Además podemos acceder a la parte de código de xml, para también poder modificar la configuración de nuestra APP

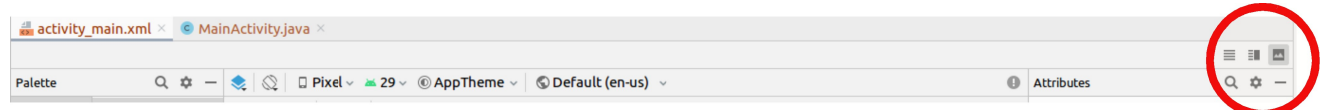


Para cambiar de vista:

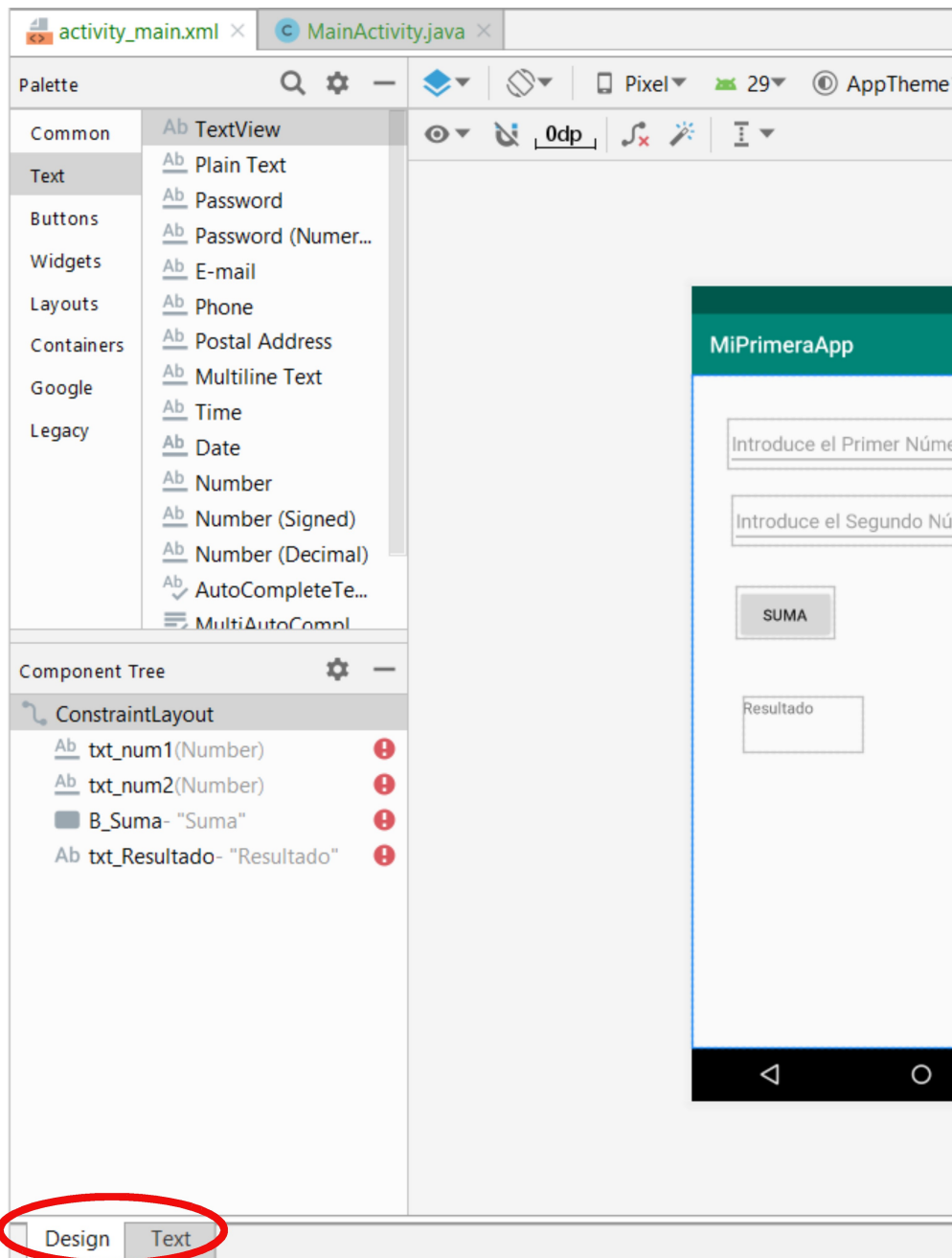


Para acceder a la parte de xml:

Desde Ubuntu:



Desde Windows nos aparecen dos pestañas en la parte inferior del archivo “*activity_main.xml*”



Modificar Atributos:

id: Nombre con el que referenciaremos nuestro objeto en la parte lógica (código). Es único e irrepetible

text: Texto a mostrar

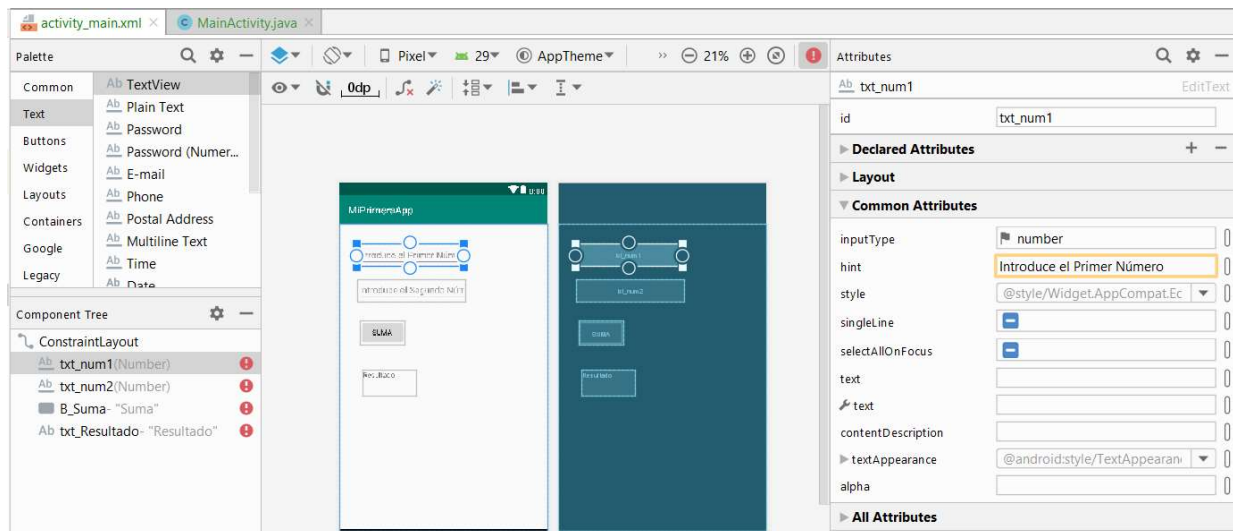
textsize: Tamaño del texto

textcolor: Color del texto

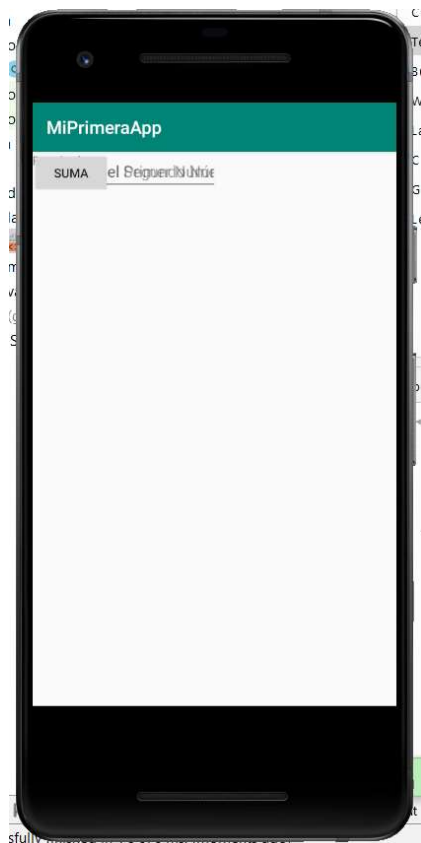
hint: Texto a mostrar en los textView que desaparece cuando escribimos o mostramos información en ellos.


gravity: Indicamos la posición del texto dentro de las textViews. Si lo que queremos es que el texto aparezca en la parte derecha, es mejor seleccionar “end”.

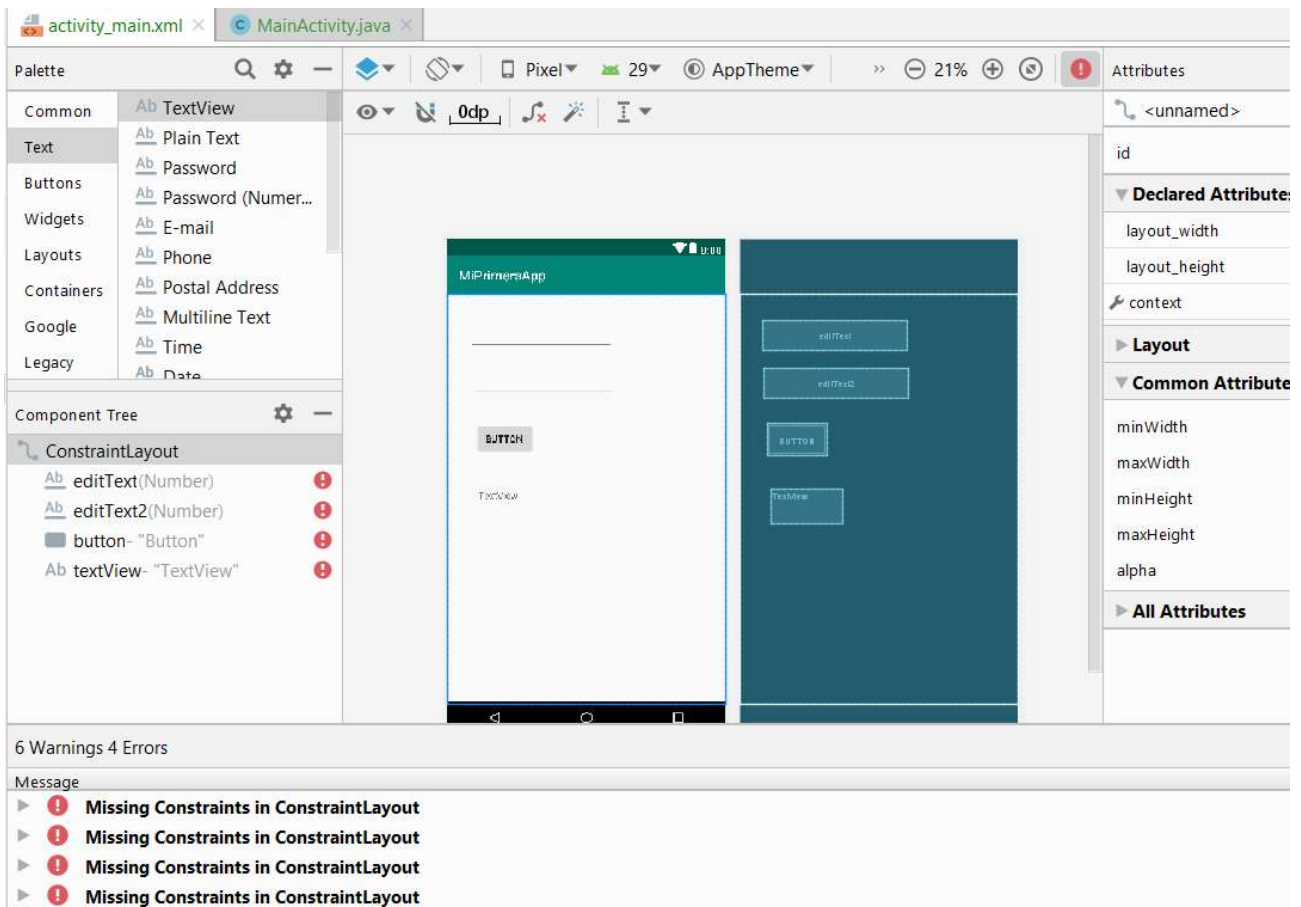
OJO!! para que aparezca las opciones debemos pulsar al mismo tiempo “ctrl+barra espaciadora”, una vez elegido la opción presionamos INTRO.



Si únicamente arrastramos los objetos que van a formar parte de nuestra aplicación y ejecutamos nuestro programa, veremos en nuestro terminal virtual que aparecen todos los componentes amontonados.

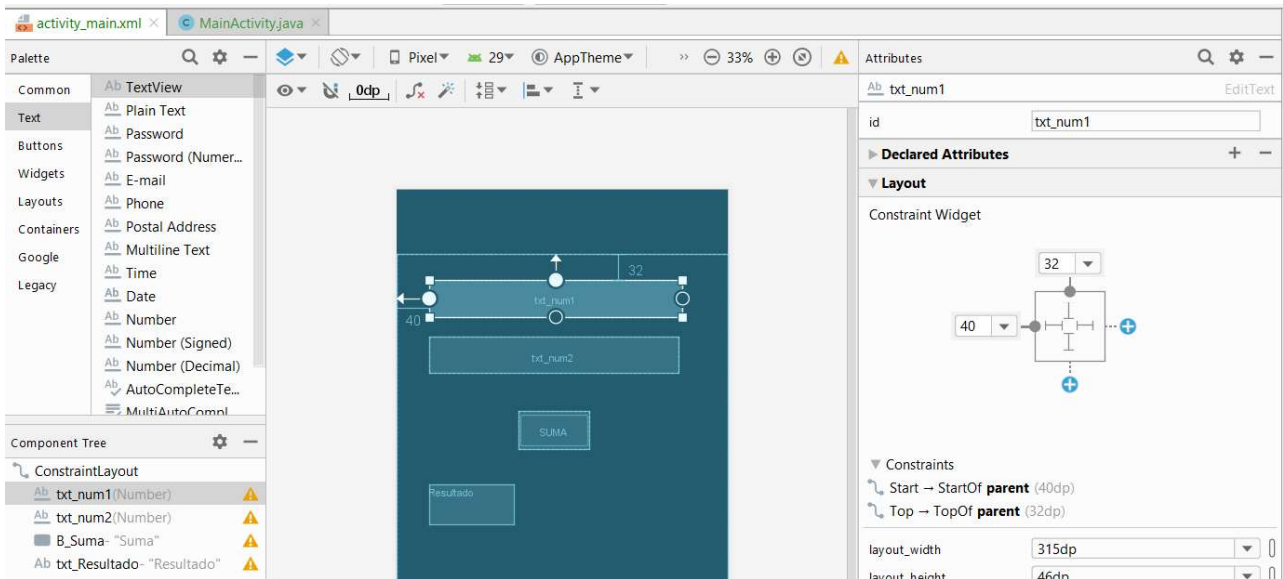


Además si nos fijamos bien, en la parte de diseño podemos ver el siguiente símbolo de error/advertencia . Si pulsamos sobre el, nos aparece en la parte inferior los **message**.



Para eliminar los errores de **“Missing Constraints in CosntraintLayout”** es necesario que a los objetos que van a formar parte de nuestra APP les asignamos puntos de referencia (mínimo dos). Para ellos pulsaremos en los círculos que aparecen y arrastremos hasta uno de los márgenes. También podemos modificar estos márgenes en la parte de **layout** en attributes.

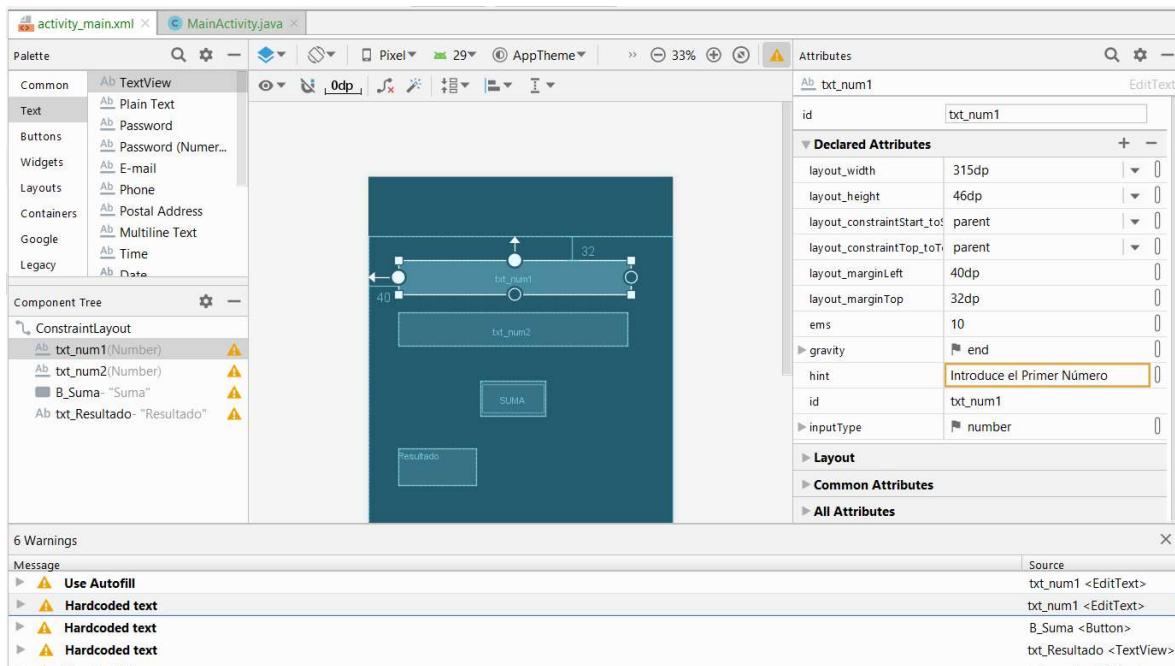
Entornos de Desarrollo: Android-Studio



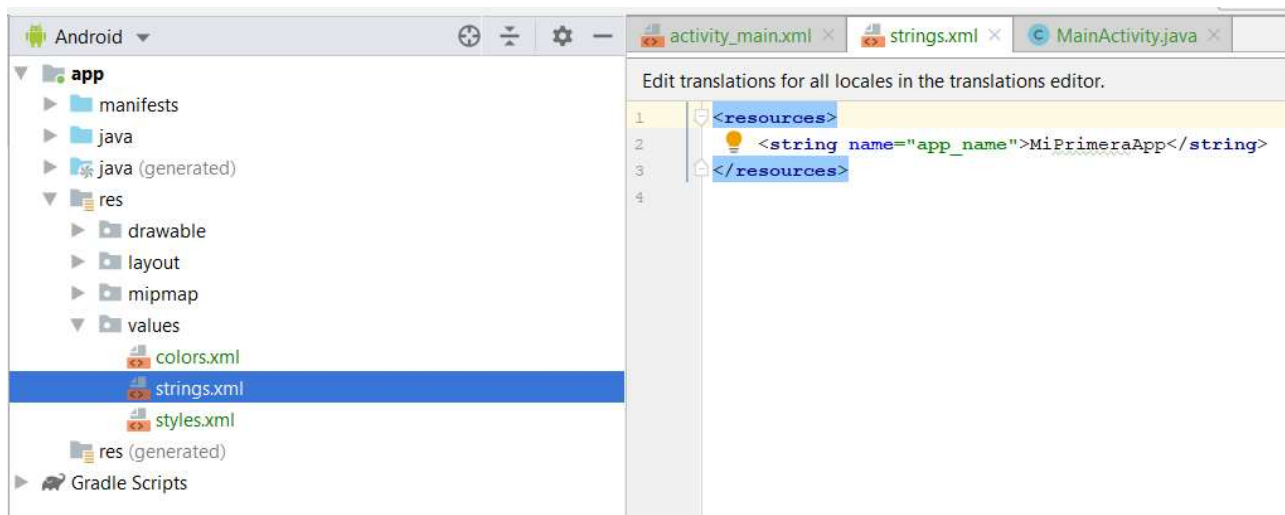
Ahora cuando ejecutemos nuestra aplicación, la distribución de los objetos que forman parte de nuestra APP ya es correcta.



Ahora solo nos falta eliminar los Warnings **“Hardcodec text”**



Para ello, hay que editar el archivo **“strings.xml”** que vamos a encontrar en la ruta **“app/res/values/strings.xml”**

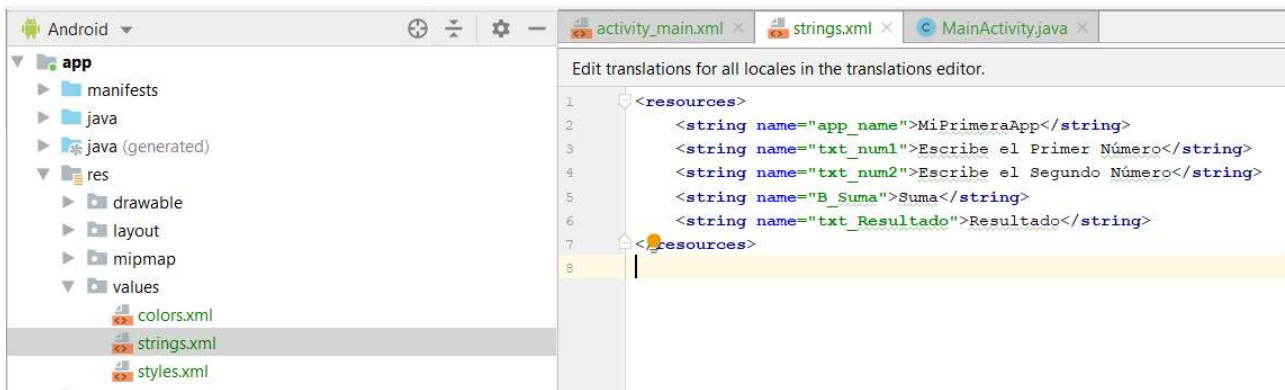


Creamos una entrada para cada uno de nuestros objetos. La estructura de estas entradas es la siguiente:

```
<string name="txt_num2">Escribe el Segundo Número</string>
```

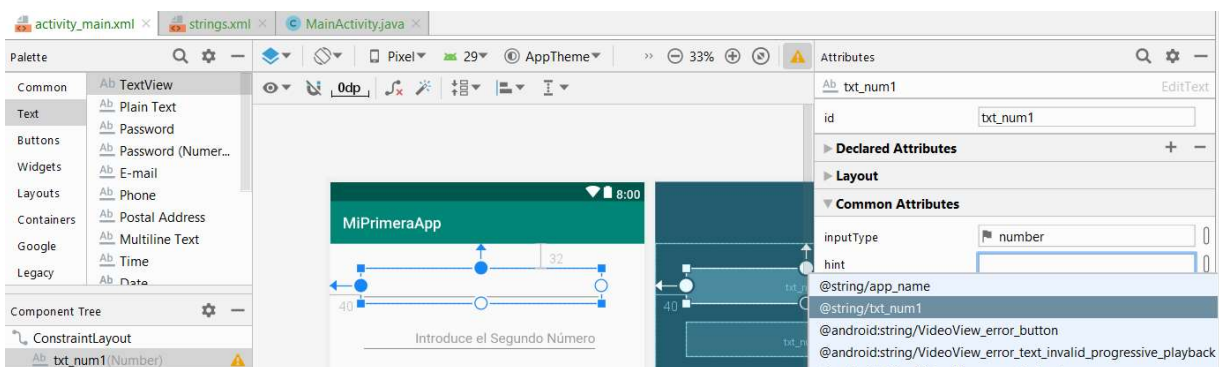
txt_num2= Es el id de nuestro componente.

>Escribe el Segundo Número<= Es el texto que aparecerá en nuestros objetos.



Una vez realizados todos los cambios, solo nos queda asignar a los objetos estas entradas .xml. Para ello volvemos a la parte grafica y en el caso de los textView modificamos los **hint**.

OJO!! para que aparezca las opciones debemos pulsar al mismo tiempo “ctrl+barra espaciadora”, una vez elegido la opción presionamos **INTRO**.

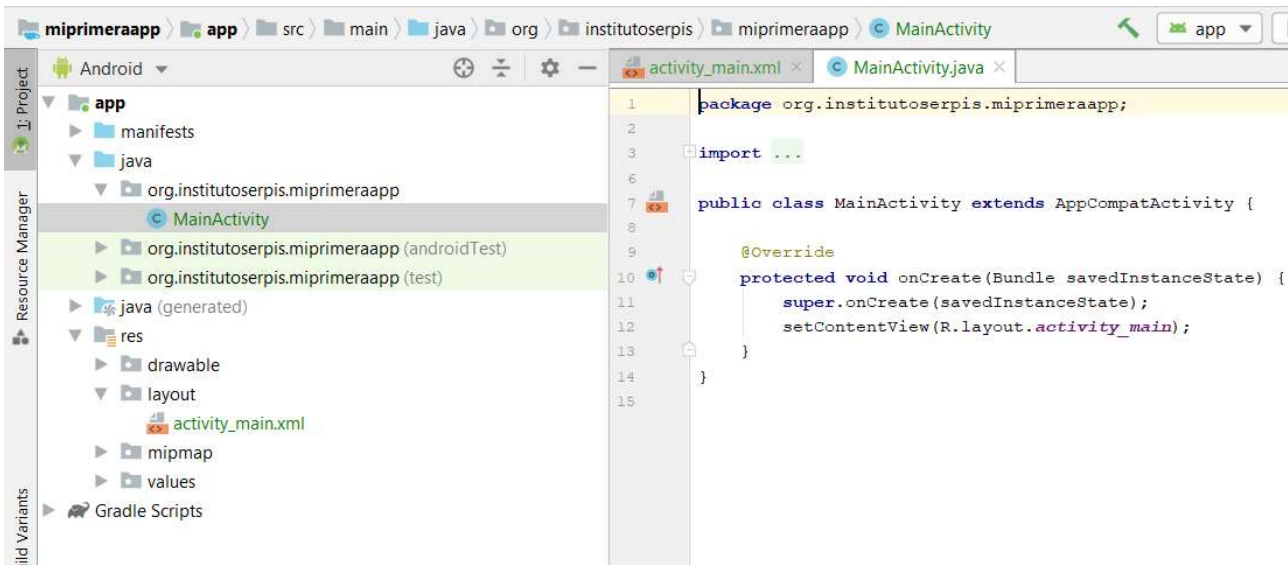


La parte lógica, es la parte destinada al código de nuestras aplicaciones.

Hay una parte de código sin la cual nuestra aplicación no funcionara, esta parte es la que siempre vendrá implementada.

Este código es el siguiente:

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



Cuando trabajamos con entornos gráficos en Java, es necesario que indiquemos que componentes vamos a utilizar en nuestro programa.

En este ejemplo tenemos cuatro componentes, pero únicamente tenemos tres en los cuales vamos a introducir o mostrar datos. El Button únicamente disparar un evento.

Estos componentes los vamos a declarar de tipo privado, únicamente esta clase va a tener acceso a ellos.

En este ejemplo hemos declarado dos de tipo **EditText** y otro de tipo **TextView**. Los de tipo EditText los hemos llamado **et1** y **et2**. Y el de tipo TextView lo hemos llamado **tv1**. Cuando declaremos estas variables, automáticamente se añaden las librerías pero si no es así, con la combinación de teclas **Alt+Intro** nos las añade.

Una vez ya tenemos declarados nuestros componentes, ahora toca guardar los valores que vamos a introducir en los EditText. La variable que hemos declarada “et1” le asignamos el valor de lo introduciremos en txt_num1, para ello utilizamos la siguiente estructura:

```
et1 = (EditText) findViewById(R.id.txt_num1);
```

donde:

(EditText) : lo utilizamos para convertir en tipo EditText

findViewById: Método para buscar

R.id.txt_num1: R. función que nos permite unir la parte lógica con la parte gráfica de nuestra aplicación. Y con **id.txt_num1** indicamos el id de nuestro componente a buscar.

En cambio para el Text en el que vamos a mostrar el resultado, lo que hacemos es lo siguiente:

```
tv1 = (TextView) findViewById(R.id.txt_Resultado);
```

en lugar de convertir en texto editable lo que hacemos es convertirlo en texto para mostrar.

Ahora solo nos falta implementar el método de nuestra aplicación.

Creamos un método nuevo y le pasamos un objeto de la clase View, en este caso le he puesto el mismo nombre al objeto.

Definimos una variable para guardar el valor que hemos introducido en nuestro txt_num1, en este caso la hemos llamado num1 y la hemos definido de tipo entero, por lo tanto deberemos convertir el valor introducido en txt_num1 que este tipo string.

El valor de txt_num1 ya lo tenemos guardado en la variable et1, ahora solo nos falta convertirlo, por ello utilizamos **Integer.parseInt** y le pasamos el valor que habremos introducido en txt_num1.

Con getText recuperamos el valor que tenemos en el objeto et1, con toString lo que estamos haciendo es decirle que ese valor es de tipo string y finalmente con **Integer.parseInt** lo convertimos en tipo entero para poder operar con el.

```
int num1 = Integer.parseInt(et1.getText().toString());
```

También lo podemos hacer de la siguiente forma:

```
String valor1_String=et1.getText().toString();
int num1=Integer.parseInt(valor1_String);
```

Realizamos la operación que necesitamos, en nuestro caso es una simple suma:

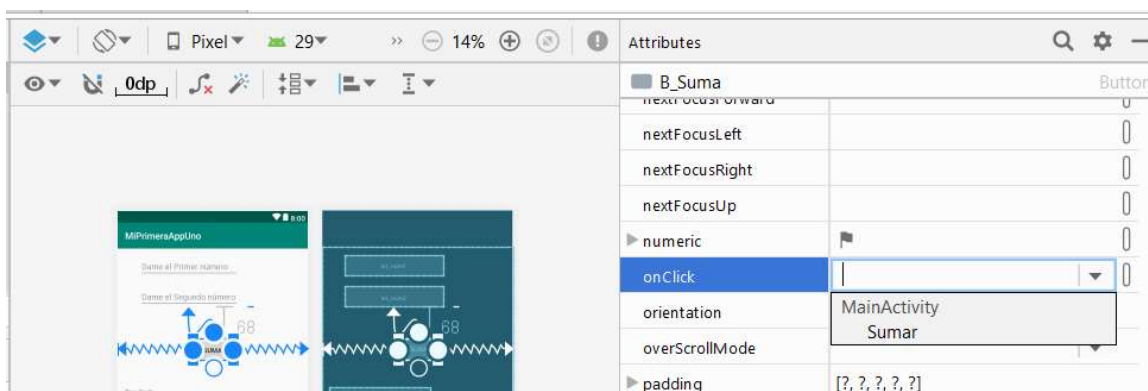
```
int suma = num1 + num2;
```

Y para finalizar la parte de código únicamente nos queda mostrar el resultado de nuestra operación en el TextView.

```
tv1.setText(String.valueOf(suma));
```

Ahora tenemos que revertir el cambio de entero a string para poderlo mostrar en este TextView. Por eso hacemos uso de setText, con lo que mostramos resultado de nuestra operación, convirtiendo antes den String con **String.valueOf**

Una vez creado el método, solo nos falta asignarlo al boton que lo va a lanzar. Para ello volvemos al entorno grafico y dentro de los atributos del Boton buscamos donde pone **onClick** y al pinchar el desplegable nos aparecen todos los métodos creados seleccionamos el que queremos asignar y ya tendremos finalizada nuestra aplicación.



-----Código de MiPrimeraApp-----

```
package org.institutoserpis.miprimeraappuno;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    private EditText et1,et2;
    private TextView tv1;

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

        et1 = (EditText) findViewById(R.id.txt_num1);
        et2 = (EditText) findViewById(R.id.txt_num2);
        tv1 = (TextView) findViewById(R.id.txt_Resultado);
    }

    //Método SUMAR
    public void Sumar(View view) {

        /* String valor1_String=et1.getText().toString();
        String valor2_String=et2.getText().toString();
        int num1=Integer.parseInt(valor1_String);
        int num2=Integer.parseInt(valor2_String);
        int suma=num1+num2;
        String resultado=String.valueOf(suma);
        tv1.setText(resultado);*/

        int num1 = Integer.parseInt(et1.getText().toString());
        int num2 = Integer.parseInt(et2.getText().toString());

        int suma = num1 + num2;

        tv1.setText(String.valueOf(suma));

    }
}
```

EJERCICIOS:

1-Crear APP con la siguiente apariencia Fisica:



Cada botón realizará una operación básica (Suma, Resta, Multiplicación, División).

El Nombre de la aplicación debe ser *ACalculadora* y debéis minimizar al máximo todos los Errores/Warnings que aparezcan en la parte de diseño. Una vez creado subir proyecto al Github.