

## 2. APP CODEBUTTON

---

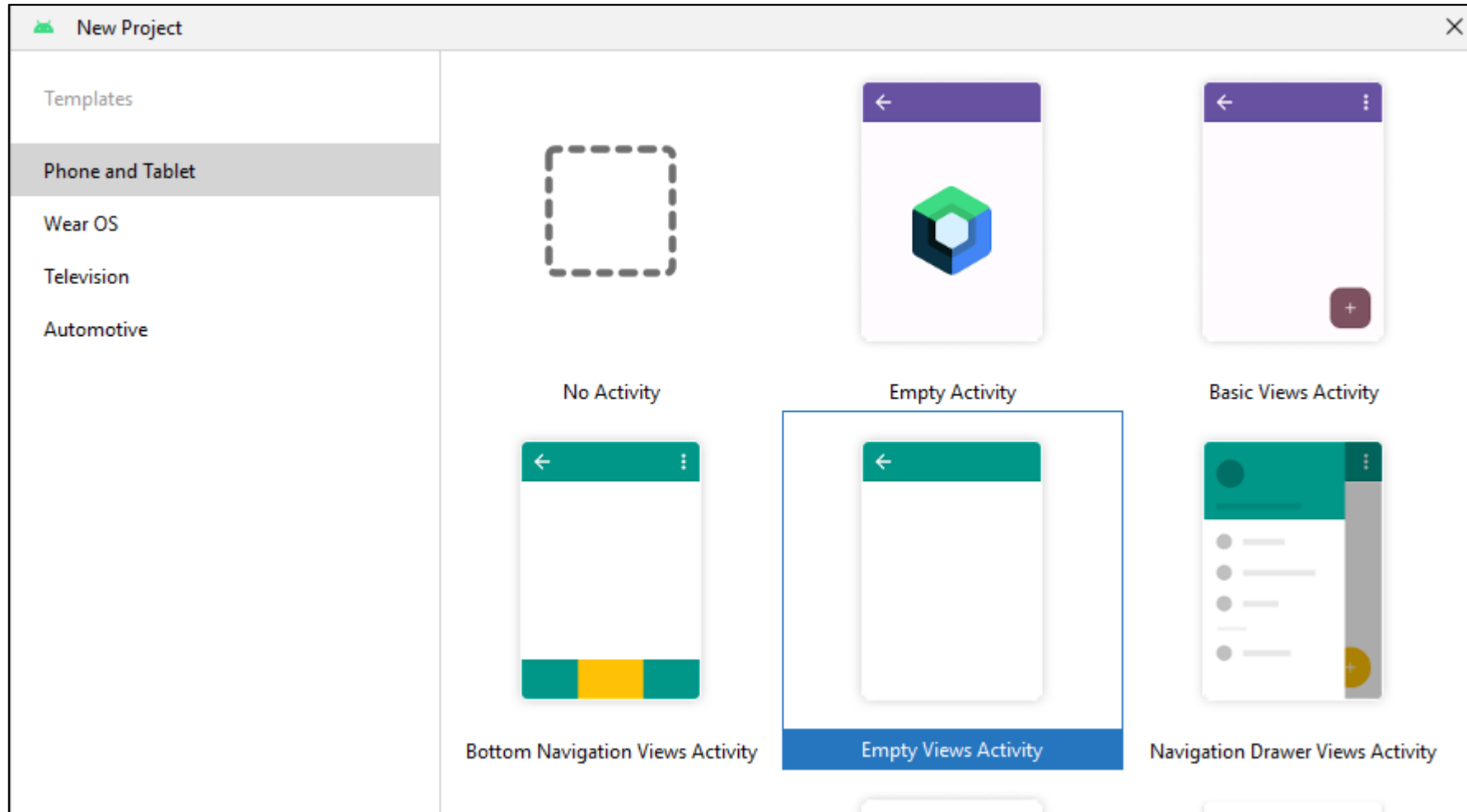
### Objetivos

En esta primera aplicación veremos las diferentes formas que presenta Android de asociar el código del evento clic de un botón. Existen dos formas básicas, pero se pueden extender a 4, en función de donde se implemente la clase Listener que recoge el clic:

- 1) Mediante la definición de un objeto anónimo de la clase OnClickListener, y asociando el objeto con el botón dentro de onCreate de MainActivity
- 2) Implementando la interfaz OnClickListener en el MainActivity.java.
- 3) Creando una clase aparte que implemente el listener OnClickListener
- 4) Enlazando una función con argumentos View a través de la propiedad onClick XML del botón

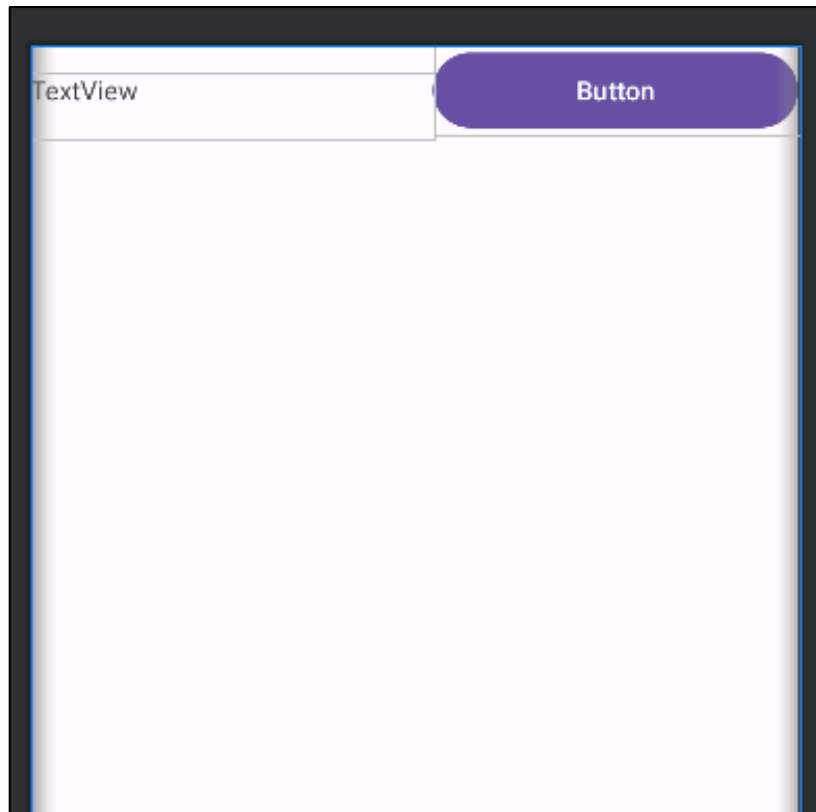
## 2. APP CODEBUTTON

**Paso 1.** Creamos una nueva aplicación Android, mediante la opción Empty Views Activity



## 2. APP CODEBUTTON

**Paso 2.** En el activity\_main.xml primero pasamos el layout principal de ConstraintLayout (layout por defecto) a LinearLayout, con orientación horizontal (por facilidad). A continuación insertamos un textview y un button.



```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:orientation="horizontal"
8      tools:context=".MainActivity">
9
10     <TextView
11         android:id="@+id/textView"
12         android:layout_width="111dp"
13         android:layout_height="35dp"
14         android:layout_weight="1"
15         android:text="TextView" />
16
17     <Button
18         android:id="@+id/button"
19         android:layout_width="wrap_content"
20         android:layout_height="wrap_content"
21         android:layout_weight="1"
22         android:text="Button" />
23 </LinearLayout>
24
```

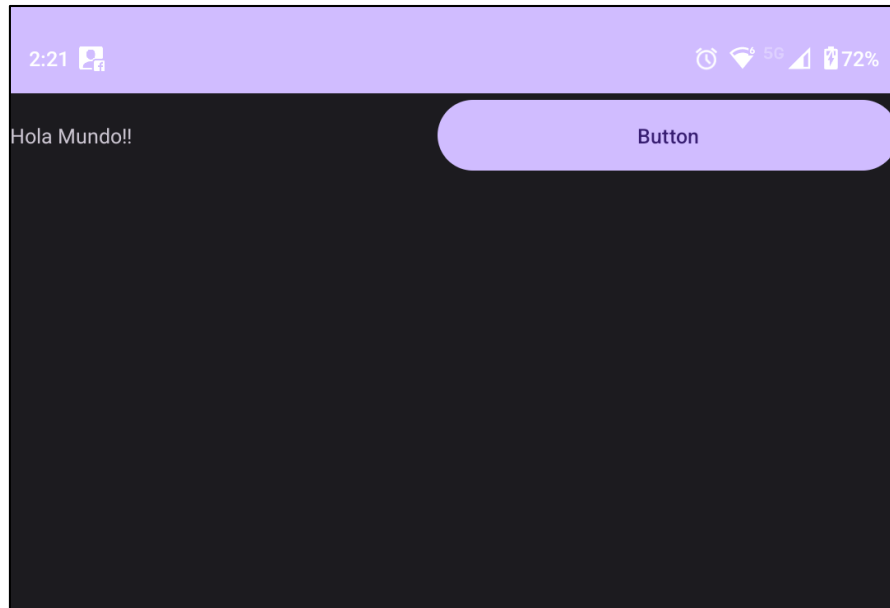
## 2. APP CODEBUTTON

**Paso 3.** En MainActivity .java definimos dos atributos de la clase, un Button y un TextView y los enlazamos con los elementos xml creados anteriormente mediante la función findViewById, dentro OnCreate de MainActivity

```
1 package com.example.code_boton;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6 import android.widget.Button;
7 import android.widget.TextView;
8
9 2 usages
10 public class MainActivity extends AppCompatActivity {
11
12     1 usage
13     private Button mybutton;
14     1 usage
15     private TextView mytext;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.activity_main);
21         this.mybutton=(Button)findViewById(R.id.button);
22         this.mytext=(TextView)findViewById(R.id.textView);
23     }
24 }
```

## 2. APP CODEBUTTON

**Paso 4. Primera forma definición de código del botón.** Mediante la definición de un objeto anónimo de la clase OnClickListener y asociación con su botón correspondiente dentro de onCreate

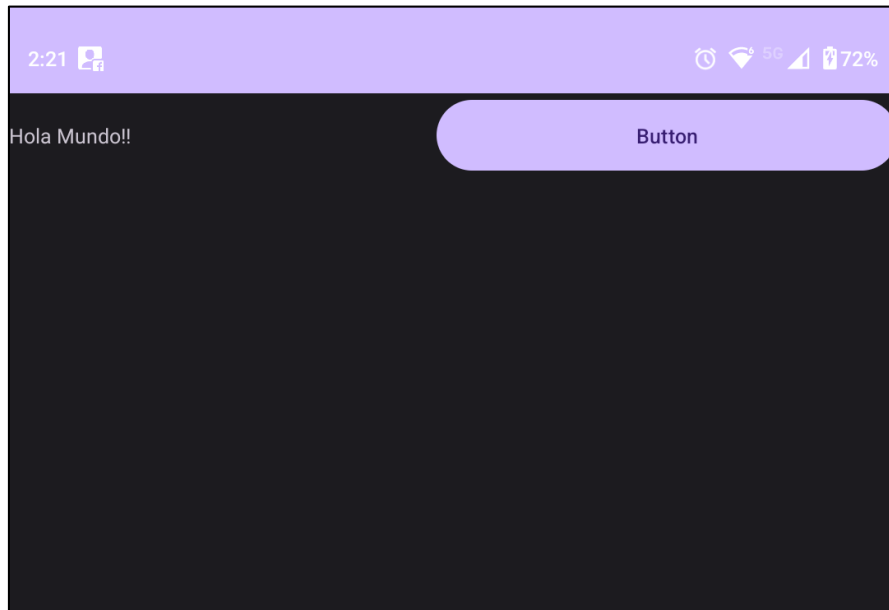


Forma mas usada, pero que tiene un consumo mayor de memoria

```
1 package com.example.codebutton;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.annotation.SuppressLint;
6 import android.os.Bundle;
7 import android.view.View;
8 import android.widget.Button;
9 import android.widget.TextView;
10
11 2 usages
12 public class MainActivity extends AppCompatActivity {
13     2 usages
14     private Button mybutton;
15     2 usages
16     private TextView mytextView;
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.activity_main);
21         this.mybutton = (Button)findViewById(R.id.button);
22         this.mytextView = (TextView)findViewById(R.id.textView);
23         this.mybutton.setOnClickListener(new View.OnClickListener(){
24             @Override public void onClick(View view) { mytextView.setText("Hola Mundo!!"); }
25         });
26     }
27 }
```

## 2. APP CODEBUTTON

**Paso 5. Segunda forma definición de código del botón.** Implementando la interface OnClickListener en el MainActivity.java



El problema de este método es que comparte el método onClick con el código de todos los botones del layout. Se debe de separar mediante view.getId()

```
1 package com.example.codebutton;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.annotation.SuppressLint;
6 import android.os.Bundle;
7 import android.view.View;
8 import android.widget.Button;
9 import android.widget.TextView;
10
11 2 usages
12 public class MainActivity extends AppCompatActivity implements View.OnClickListener{
13     2 usages
14     private Button mybutton;
15     2 usages
16     private TextView mytextView;
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.activity_main);
21         this.mybutton = (Button) findViewById(R.id.button);
22         this.mytextView = (TextView) findViewById(R.id.textView);
23         this.mybutton.setOnClickListener(this);
24         //this.mysecondbutton.setOnClickListener(this);
25     }
26
27     @Override
28     public void onClick(View view){
29         if(view.getId()== R.id.button) {
30             mytextView.setText("Hola Mundo!!");
31         }
32         //else if(view.getId()==R.id.mysecondbutton){
33         //    mytextView.setText("Hola second button!!");
34         //}
35     }
36 }
```

## 2. APP CODEBUTTON

**Paso 6. Tercera forma definición del código del botón.** Creando una clase aparte que implemente OnClickListener

```
1 package com.example.codebutton;
2
3 import android.view.View;
4 import android.widget.TextView;
5
6 public class MyCodeButton implements View.OnClickListener {
7     TextView textview;
8
9     public MyCodeButton(TextView mytextview){
10         this.textview=mytextview;
11     }
12     @Override
13     public void onClick(View view){
14         textview.setText("HoLa Mundo!!");
15     }
16 }
```

La única pega es que se debería generar tantas clases que implemente el listener como botones tenga el layout.

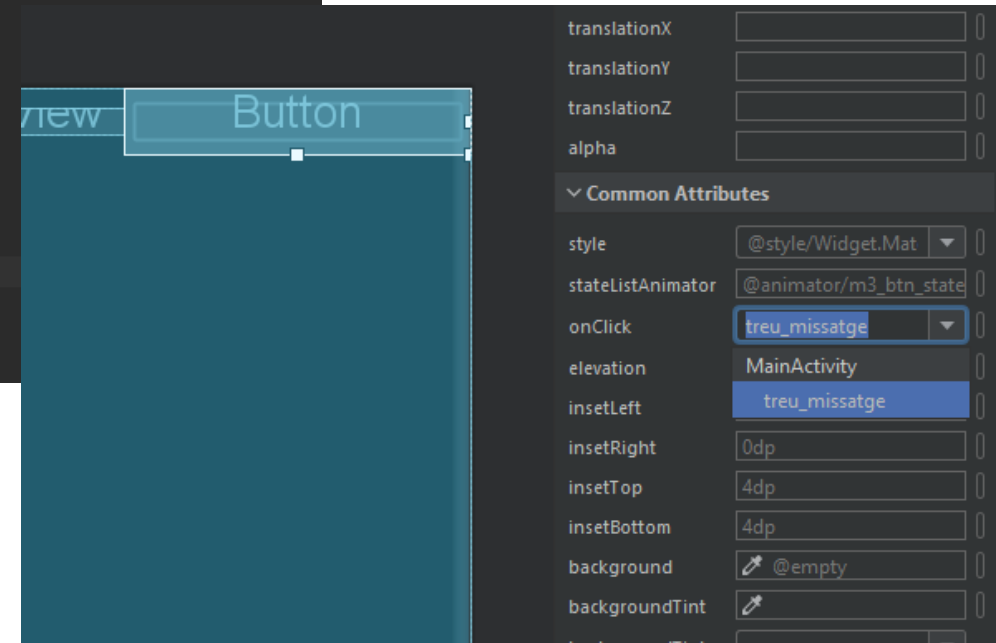
```
1 package com.example.codebutton;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.annotation.SuppressLint;
6 import android.os.Bundle;
7 import android.view.View;
8 import android.widget.Button;
9 import android.widget.TextView;
10
11 public class MainActivity extends AppCompatActivity {
12     private Button mybutton;
13     private TextView mytextview;
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_main);
18         this.mybutton = (Button) findViewById(R.id.button);
19         this.mytextview = (TextView) findViewById(R.id.textView);
20         this.mybutton.setOnClickListener(new MyCodeButton(mytextview));
21     }
22 }
```

## 2. APP CODEBUTTON

**Paso 7. Cuarta forma definición del código del botón.** Enlazando una función con argumento View a través de la propiedad onClick del XML del botón

```
1 package com.example.codebutton;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.annotation.SuppressLint;
6 import android.os.Bundle;
7 import android.view.View;
8 import android.widget.Button;
9 import android.widget.TextView;
10
11 2 usages
12 public class MainActivity extends AppCompatActivity {
13     1 usage
14     private Button mybutton;
15     2 usages
16     private TextView mytextView;
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.activity_main);
21         this.mybutton = (Button) findViewById(R.id.button);
22         this.mytextView = (TextView) findViewById(R.id.textview);
23     }
24     no usages
25     public void treumissatge(View View){
26         mytextView.setText("Hello world!!");
27     }
28 }
```

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     android:orientation="horizontal"
8     tools:context=".MainActivity">
9
10     <TextView
11         android:id="@+id/textView"
12         android:layout_width="wrap_content"
13         android:layout_height="wrap_content"
14         android:layout_weight="1"/>
15
16     <Button
17         android:id="@+id/button"
18         android:layout_width="wrap_content"
19         android:layout_height="wrap_content"
20         android:layout_weight="1"
21         android:onClick="treumissatge"
22         android:text="Button" />
23 </LinearLayout>
```



Opción elegida junto con la primera



## 2. APP IMC

---

### Objetivos

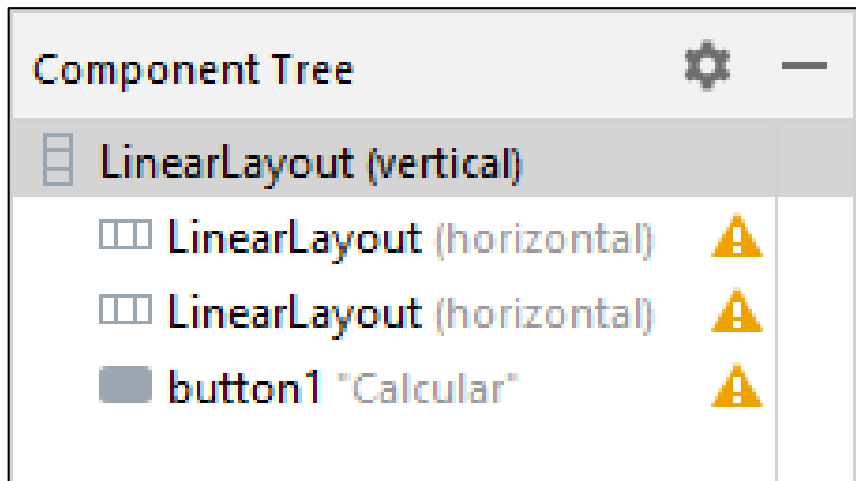
En esta segunda aplicación haremos la típica aplicación IMC Índice de Masa Corporal, que con los datos de la altura y el peso nos indica si estamos en línea o un poco gorditos

Aquí empezaremos a trabajar las diferentes formas de layout, en concreto veremos el funcionamiento del tipo LinearLayout. Veremos como enlazar dos layouts de tipo LinearLayout, uno con orientación vertical y el otro con orientación horizontal, de manera que nos permita el despliegue de nuestros controles de la forma deseada.

También introduciremos la estructura Toast que nos permite ver un mensaje en la parte media-baja de la pantalla que dura unos segundos

## 2. APP IMC

**Paso 1.** Declararemos el layout principal de tipo LinearLayout con orientación vertical. Agregaremos 2 LinearLayout con orientación horizontal y un botón, que se dispondrán verticalmente



```
<?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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="70dp"
        android:orientation="horizontal"/>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="70dp"
        android:orientation="horizontal" />
    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Calcular" />
</LinearLayout>
```

## 2. APP IMC

**Paso 2.** Insertamos los textViews y los editTexts para completar la aplicación

The screenshot displays the Android Studio IDE with the following components:

- Component Tree (Left):** Shows a hierarchy of `LinearLayout` elements. The first `LinearLayout` (vertical) contains two horizontal `LinearLayout` elements. The first horizontal `LinearLayout` contains `textView1` (labeled "Altura (m):") and `editText1` (labeled "Plain Text"). The second horizontal `LinearLayout` contains `textView2` (labeled "Peso (Kg):") and `editText2` (labeled "Plain Text"). A `button1` (labeled "Calcular") is positioned at the bottom.
- XML Editor (Center):** Displays the XML layout code for the first `LinearLayout` (vertical). The code defines the structure and attributes for the `textView1`, `editText1`, `textView2`, `editText2`, and `button1` elements.
- Preview (Right):** Shows a visual representation of the UI. It features two input fields labeled "Altura (m):" and "Peso (Kg):", each followed by a text input field. A blue button labeled "Calcular" is located at the bottom.

```
<?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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="70dp"
        android:orientation="horizontal" >

        <TextView
            android:id="@+id/textView1"
            android:layout_width="248dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Altura (m):"
            android:textSize="24sp" />

        <EditText
            android:id="@+id/editText1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:ems="10"
            android:inputType="text"
            android:textSize="24sp" />

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="70dp"
        android:orientation="horizontal" >

        <TextView
            android:id="@+id/textView2"
            android:layout_width="258dp"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="Peso (Kg):"
            android:textSize="24sp" />

        <EditText
            android:id="@+id/editText2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:ems="10"
            android:inputType="text"
            android:textSize="24sp" />

    </LinearLayout>

    <Button
        android:id="@+id/button1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Calcular" />

</LinearLayout>
```

## 2. APP IMC

**Paso 3.** En MainActivity.java declaramos las 3 variables (1 botón y 2 edittexts) que enlazaremos con sus respectivos controles xml mediante findViewById en el evento onCreate de MainActivity

```
2 usages
public class MainActivity extends AppCompatActivity {
    1 usage
    private Button button;
    1 usage
    private EditText editText1;
    1 usage
    private EditText editText2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        button = (Button) findViewById(R.id.button1);
        editText1 = (EditText) findViewById(R.id.editText1);
        editText2 = (EditText) findViewById(R.id.editText2);
    }
}
```

## 2. APP IMC

**Paso 4.** Solo nos falta definir el cálculo que hace la aplicación al hacer click en el botón, una vez hemos introducido los datos de la altura y peso. Aquí utilizamos la función Toast, que durante 2 segundos nos da el resultado de nuestro IMC.

```
button.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View view) {
        if (editText1.equals("") || editText1.equals(""))
            Toast.makeText(context: MainActivity.this, text: "Faltan parametros!", Toast.LENGTH_LONG).show();

        double altura = Double.parseDouble(editText1.getText().toString());
        double peso = Double.parseDouble(editText2.getText().toString());
        double imc= peso/(altura*altura);
        if (imc<18.5)
            Toast.makeText(context: MainActivity.this, text: "Bajo peso corporal!", Toast.LENGTH_LONG).show();
        else if (imc<24.9)
            Toast.makeText(context: MainActivity.this, text: "Normal!", Toast.LENGTH_LONG).show();
        else if (imc<29.9)
            Toast.makeText(context: MainActivity.this, text: "Sobrepeso!", Toast.LENGTH_LONG).show();
        else
            Toast.makeText(context: MainActivity.this, text: "Obeso!", Toast.LENGTH_LONG).show();
    });
});
```

## 2. APP IMC

### Paso 5. Ejemplo de ejecución de la aplicación

