Layouts

PROCEDIMIENTO GUIADO

Crea la aplicación **App_UT2_02**

Borra el TextView y añade un LinearLayout que ocupe la mitad superior de la pantalla y un TableLayout que ocupe la mitad inferior. Crear las restricciones necesarias para ello.

Se facilita parte del código:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
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"
    tools:context=".MainActivity">



    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        app:layout_constraintHeight_percent="0.5"
        android:background="@color/purple_200"
        android:orientation="horizontal"
>

    </LinearLayout>

    <TableLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:background="@color/teal_200"
        app:layout_constraintBottom_toBottomOf="parent"
>

    </TableLayout>
</androidx.constraintlayout.widget.ConstraintLayout>
```

En la parte superior (LinearLayout) añade 2 botones y un texto. Intenta realizar un diseño similar al siguiente:

Establece las propiedades necesarias para que los botones ocupen todo el ancho.
Abre el fichero **colors.xml** dentro de res → values y observa su contenido. Vamos a añadir 3 colores:

```
<color name="gris">#9B9B9B</color>
<color name="azul_marino">#252850</color>
<color name="amarillo">#DED560</color>
```

Establece el color azul_marino a los botones y el botón amarillo al layout. Por último, establece el tamaño del texto del TextView en 100 sp.

En la parte inferior (TableLayout) intenta realizar un diseño similar al de una calculadora. Para ello necesitaremos añadir 5 TableRow y dentro de cada uno 3 o 4 botones:

Vemos que todos los botones tienen el mismo estilo. Para no tener que establecer varias propiedades en cada uno de los botones vamos a definir un estilo.
Para ello crea un fichero **styles.xml** dentro de la carpeta res → values. El contenido puede ser el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <style name="BotonCalculadora" parent="Widget.AppCompat.Button">
      <item name="android:layout_height">match_parent</item>
      <item name="backgroundTint">@color/gris</item>
      <item name="android:textSize">20sp</item>
   </style>
</resources>
```

Vemos que acabamos de crear un estilo llamado BotonCalculadora que hereda los estilos de Widget.AppCompat.Button y en el que redefinimos varios estilos para hacer que ocupen el mismo alto que su padre (el TableRow), ponerle el color gris definido antes y establecer 20sp como tamaño de texto.

Ahora, en cada botón simplemente tenemos que indicarle que usen ese estilo:

```
<Button android:id="@+id/botonBorrar"
   style="@style/BotonCalculadora"
   android:layout_span="2"
   android:text="Borrar" />
```

FUNCIONALIDAD

Aunque en esta unidad únicamente vamos a trabajar con los diseños vamos a ir adelantando trabajo y establecer algo de funcionalidad.

Únicamente programaremos la funcionalidad del LinearLayout. Si hacemos clic en el botón Mostrar Toast mostraremos un Toast corto en el que se muestre el valor del TextView.

Crea un atributo contador directamente en la clase MainActivity.
Recuerda buscar cada componente con el método findViewById. Por ejemplo:

```
val botonContar = findViewById<Button>(R.id.botonContar)
```

Establece un método cuando se produzca el evento OnClick:

```
botonContar.setOnClickListener { contar(texto) }
```

Y crea dos métodos con la funcionalidad para incrementar en uno el contador y modificar el texto por un lado y mostrar el Toast por otro.