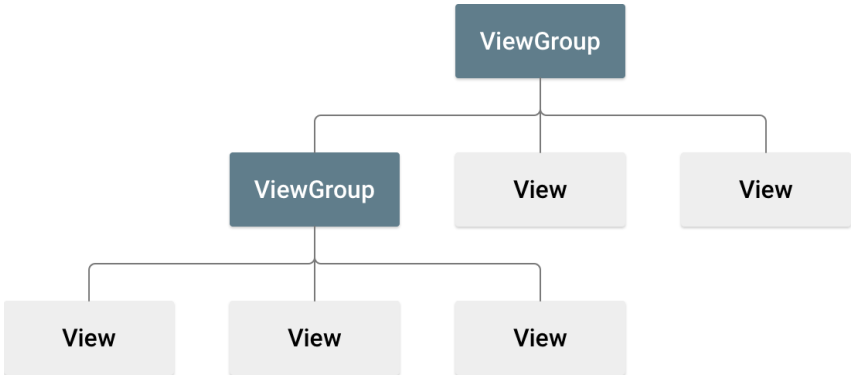
SESIÓN 2

- Interfaz de usuario de una actividad
- Manejo de eventos
- Cambio de orientación

Interfaz de usuario (IU) de una actividad

Se puede crear de dos formas:

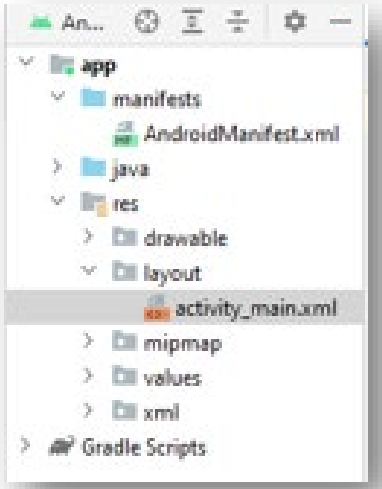
1. Instanciando los elementos durante el tiempo de ejecución
2. Declarando los elementos en ficheros XML.



Crear la IU en tiempo de ejecución

```
public class MainActivity extends AppCompatActivity {
 protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   //setContentView(R.layout.activity_main);
   TextView tv = new TextView(this);
   tv.setText("Hola Mundo");
   tv.setTextSize(50);
   tv.setX(200);
   tv.setY(900);
   setContentView(tv);
 }
}
```

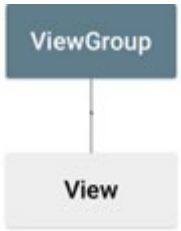
Crear la IU en ficheros XML



- Los ficheros de diseño se almacenan en `res/layout`
- Cada elemento describe una *clase java* de una vista (`View`).
- Permite separar la presentación del código que controla su comportamiento
- Se instancia y muestra en la actividad con `setContentView(R.layout.nombre_fichero_xml)`
- R es una clase autogenerada con todos los recursos de la aplicación.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/a
  xmlns:tools="http://schemas.android.com/tool
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  tools:context=".MainActivity">
  <TextView
    android:id="@+id/txt_Hola"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Hola DAM"
    android:textAlignment="center"
    android:textSize="25sp"
    android:textStyle="bold" />
</LinearLayout>
```

Fichero de diseño XML




```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="horizontal"
  tools:context=".MainActivity">
  <TextView
    android:id="@+id/txt_Hola"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:text="Hola DAM"
    android:textSize="25sp"
    android:textStyle="bold"
    android:gravity="center" />
</LinearLayout>
```

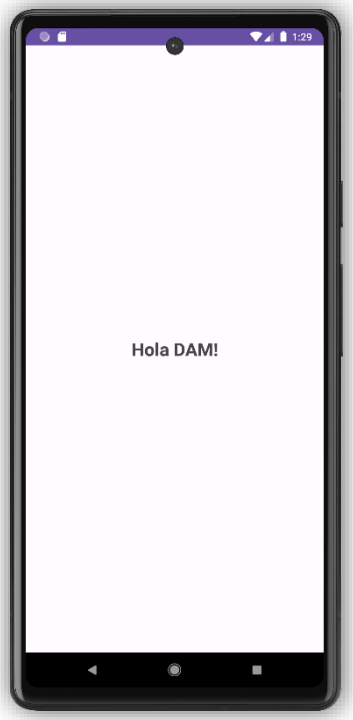
Tipo de layout a usar (ViewGroup raíz)

Declaración de espacio de nombres

Características del layout

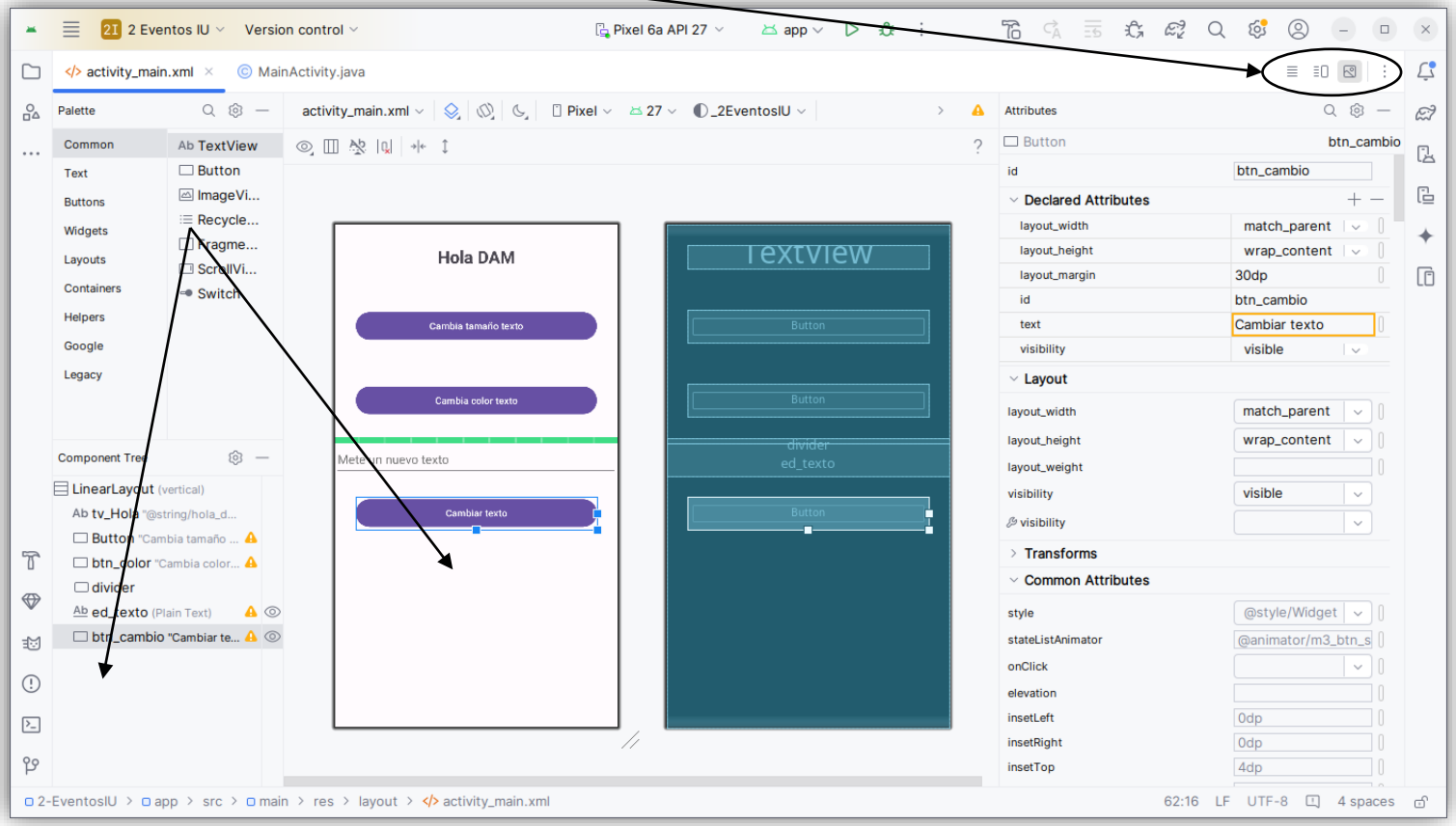
Actividad asociada al layout

Definición de un objeto View



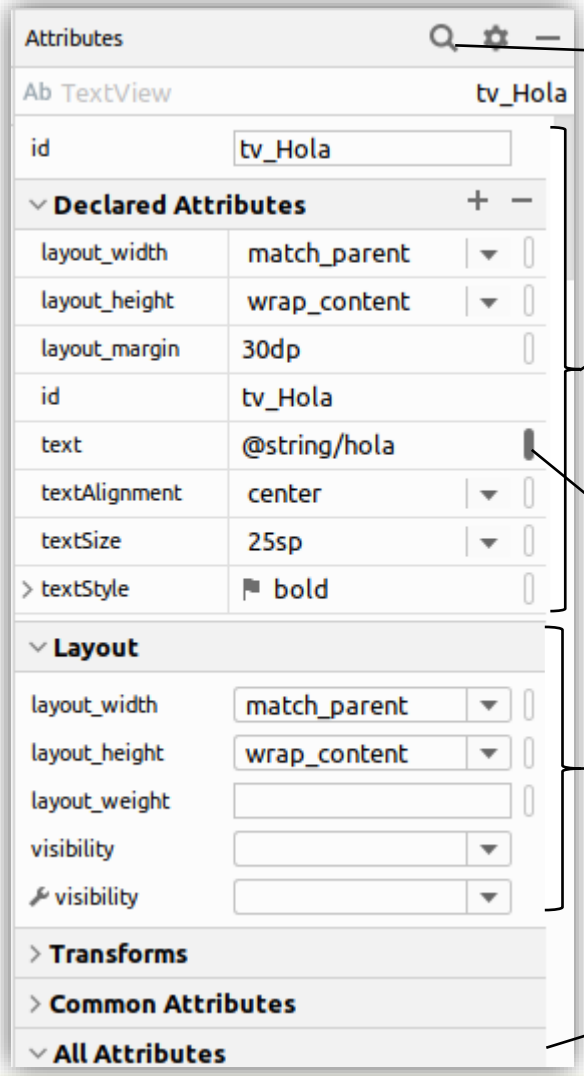
Layout Editor

- Permite crear los ficheros XML de la interfaz de usuario de una actividad
- Dispone de 3 vistas: *Code* (para escribir el texto XML), *Design* (para realizar el diseño de forma visual) y *Split* (una mezcla de las anteriores)



https://developer.android.com/studio/write/layout-editor#intro

Atributos



Búsqueda de atributo

Atributos especificados del objeto

```
<TextView
    android:id="@+id/tv_Hola"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="30dp"
    android:text="@string/hola_dam"
    android:textAlignment="center"
    android:textSize="25sp"
    android:textStyle="bold" />
```

Si el valor es una referencia o no

Tamaño y restricciones

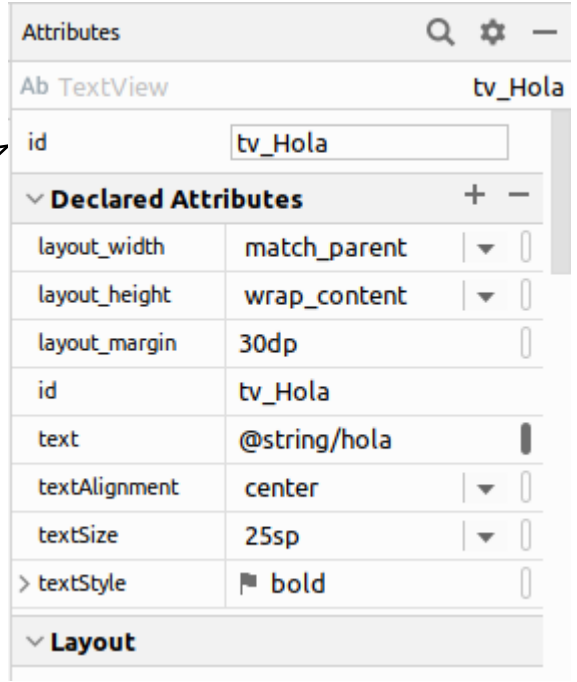
Resto de atributos elegibles para la vista

https://developer.android.com/studio/write/layout-editor#edit-properties

Atributos: identificador

- id: nº entero que identifica el objeto. En el XML se usa un *string*. Debe ser único en el mismo árbol
- Ejemplo:

```
<TextView
    android:id="@+id/tv_Hola"
    android:text="Hola DAM"
    ... />
```



- Referencia desde java:

```
TextView texto = findViewById (R.id.tv_Hola);
```

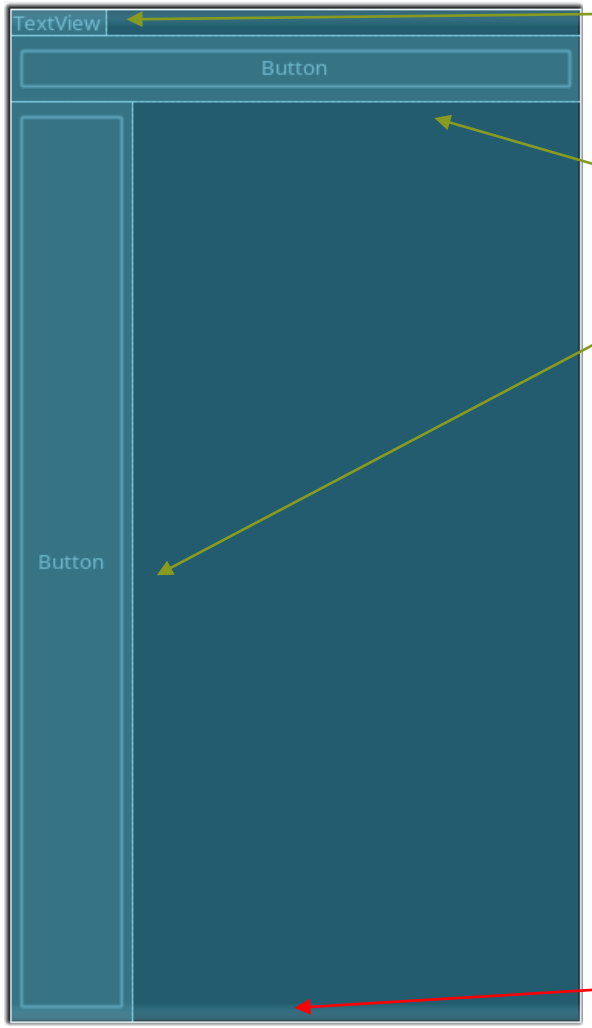
https://developer.android.com/guide/topics/ui/declaring-layout.html#id

Atributos: tamaño de la vista

- Todo objeto View es un rectángulo con un ancho (`layout_width`) y un alto (`layout_height`).
- Generalmente se usan los valores:
  - `wrap_content`: adaptarse al contenido
  - `match_parent`: agrandarse lo que permita la vista que lo contiene
  - Ocasionalmente, un número en *dp* (pixel independiente de densidad)

https://developer.android.com/guide/topics/ui/declaring-layout.html#layout-params

Atributos: tamaño de la vista



TextView

layout_width	wrap_content
layout_height	wrap_content

Ancho: adaptarse al contenido
Alto: adaptarse al contenido

Botón 1:

layout_width	match_parent
layout_height	wrap_content

Ancho: el ancho de la vista padre
Alto: adaptarse al contenido

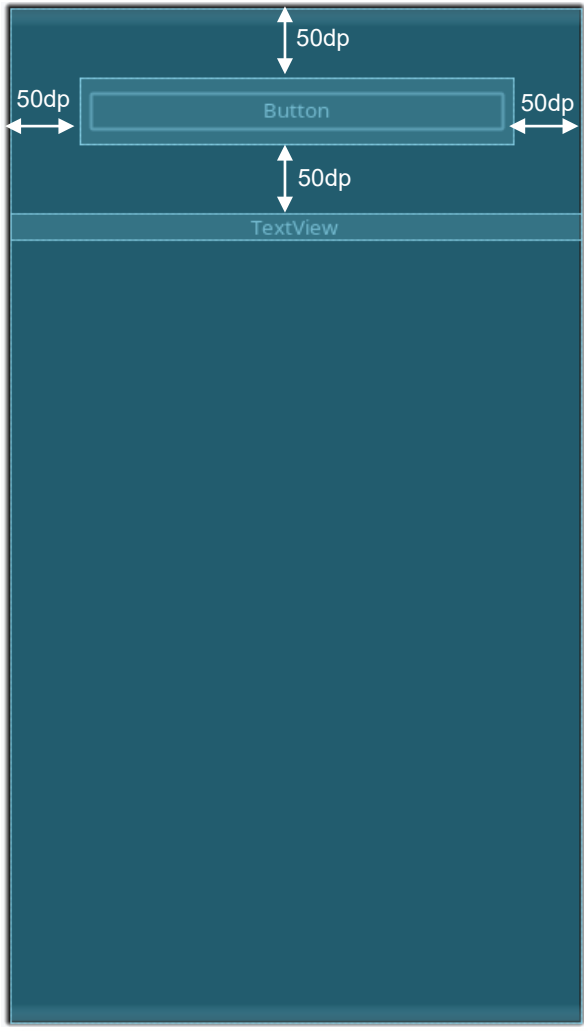
Botón 2:

layout_width	wrap_content
layout_height	match_parent

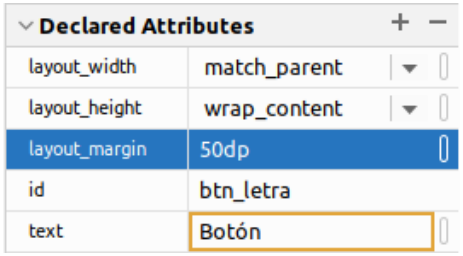
Ancho: adaptarse al contenido
Alto: el ancho de la vista padre

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"/>
<Button
    android:id="@+id/Boton1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
<Button
    android:id="@+id/Boton2"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"/>
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
```

Atributos: margen de la vista



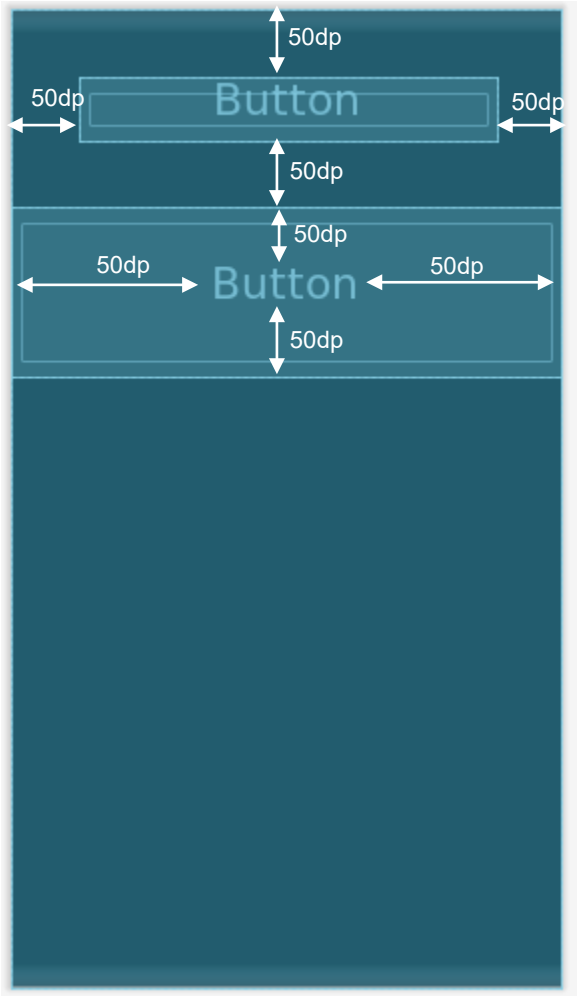
Button con un *layout_margin* a 50dp, que es la distancia que le separa del resto de *Views.*



Se puede especificar el espacio en los 4 márgenes o concretar un margen específico

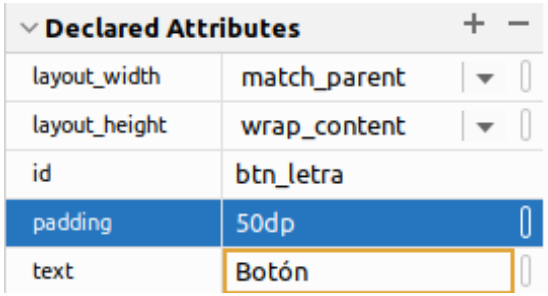

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="50dp" />
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
```

Atributos: relleno de la vista

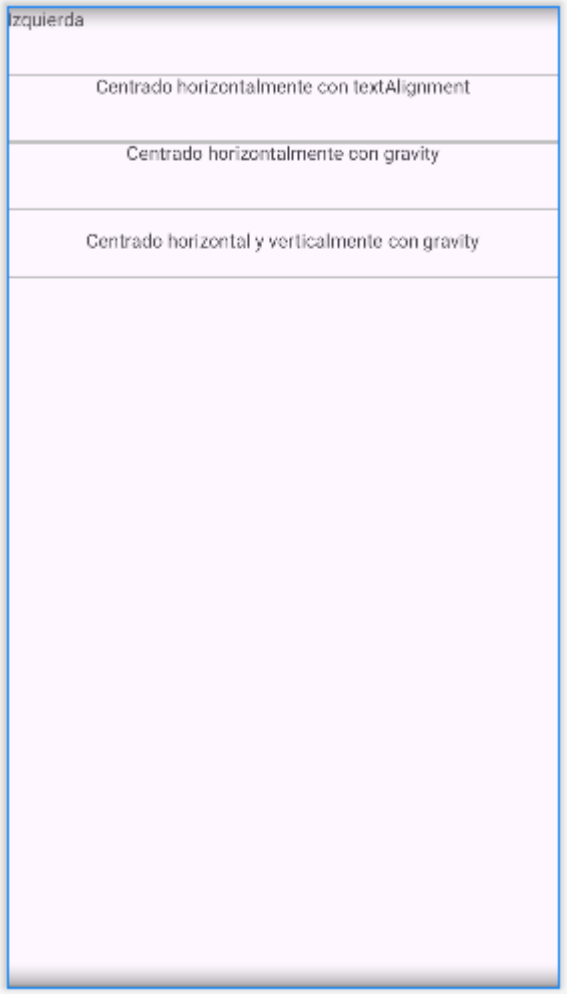


El segundo *Button* tiene un *padding* de 50dp, que especifica el espacio entre los bordes del rectángulo del objeto y el contenido.

Se puede especificar el espacio en los 4 bordes o concretar un borde específico

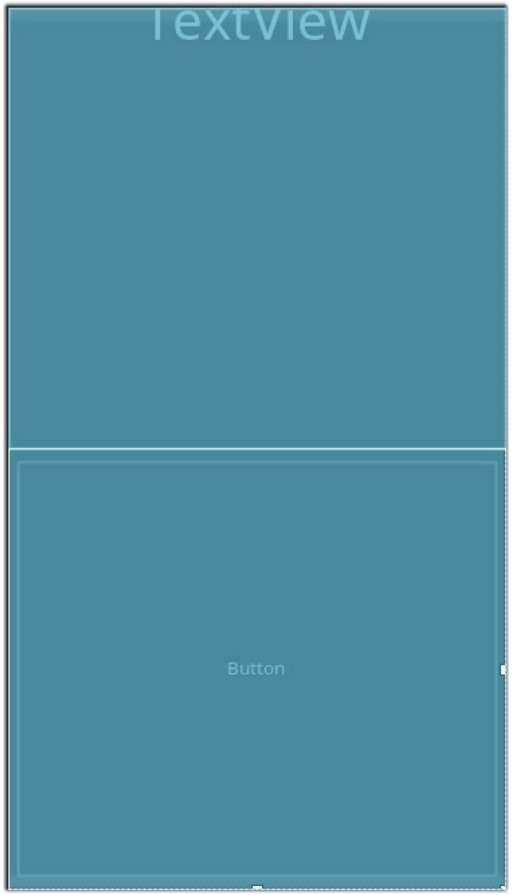


Atributos: alineación




```
<TextView
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:text="Izquierda" />

<TextView
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:text="Centrado horizontalmente con textAlignment"
    android:textAlignment="center" />

<TextView
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:gravity="center_horizontal"
    android:text="Centrado horizontalmente con gravity" />

<TextView
    android:layout_width="match_parent"
    android:layout_height="50dp"
    android:gravity="center_horizontal|center_vertical"
    android:text="Centrado horizontal y verticalmente con gravity" />
```

Distribución equitativa



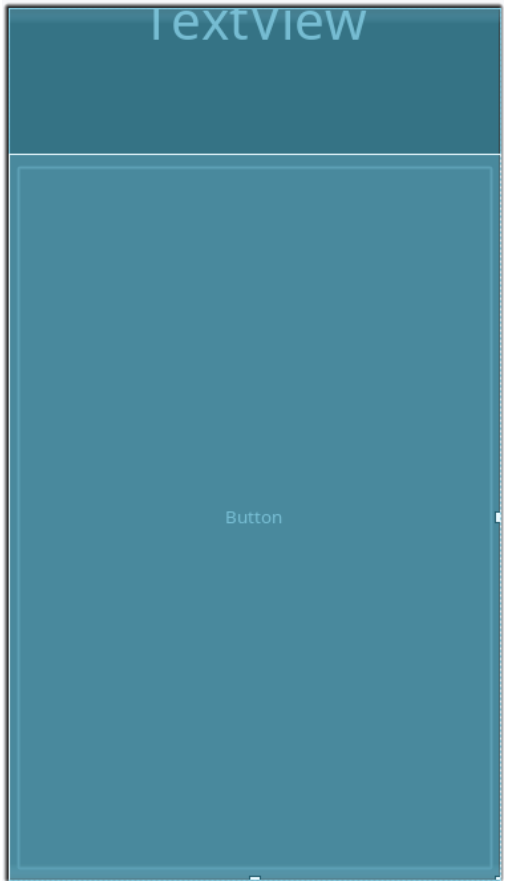
Si se quiere que todos los elementos usen la misma cantidad de espacio en la pantalla, en cada *View* hay que definir el peso (*weight*)

Ejemplo con orientación vertical:

```
<TextView
  android:id="@+id/tv_Hola"
  android:layout_width="match_parent"
  android:layout_height="0dp"
  android:layout_weight="1"
  android:text="@string/hola"
  android:textAlignment="center"
  android:textSize="25sp"
  android:textStyle="bold" />
<Button
  android:id="@+id/btn_letra"
  android:layout_width="match_parent"
  android:layout_height="0dp"
  android:layout_weight="1"
  android:text="Botón" />
```

https://developer.android.com/guide/topics/ui/layout/linear?#Weight

Distribución no equitativa



Si se quiere que algún elemento use más espacio en la pantalla, en cada View hay que definir un peso diferente

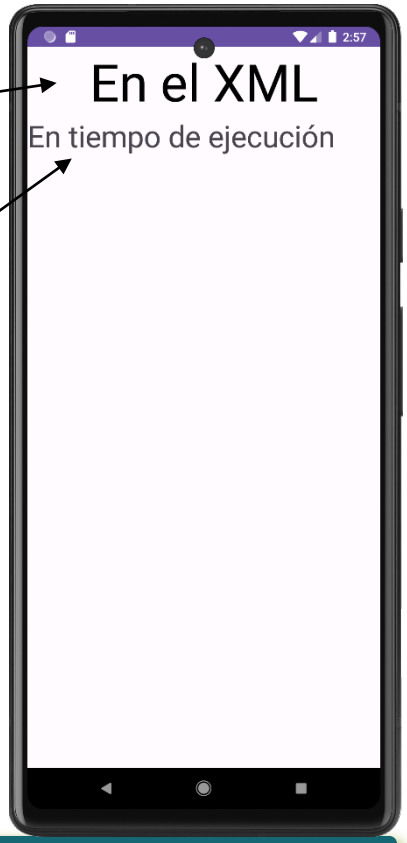
Ejemplo con orientación vertical:

```
<TextView
  android:id="@+id/tv_Hola"
  android:layout_width="match_parent"
  android:layout_height="0dp"
  android:layout_weight="1"
  android:text="@string/hola"
  android:textAlignment="center"
  android:textSize="25sp"
  android:textStyle="bold" />
<Button
  android:id="@+id/btn_letra"
  android:layout_width="match_parent"
  android:layout_height="0dp"
  android:layout_weight="5"
  android:text="Botón" />
```

Mas peso

Añadir a una IU creada en XML una vista creada en tiempo de ejecución

```
public class MainActivity extends AppCompatActivity {
 protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   TextView tv = new TextView(this);
   tv.setText("En tiempo de ejecución");
   tv.setTextSize(35);
   //tv.setX(60);
   //tv.setY(900);
   //setContentView(tv);
    LinearLayout miDiseño = (LinearLayout)
             findViewById(R.id.id_linearLayout);
   miDiseño.addView(tv);
 }
}
```



EditText (1)

Componente de la interfaz de usuario para introducir o modificar texto

```
<EditText
   android:id="@+id/identificador"
   android:layout_height="wrap_content"
   android:layout_width="match_parent"
   android:inputType="text"
   android:hint="Mete un nuevo texto"
   android:text= "Texto por defecto"
/>
```

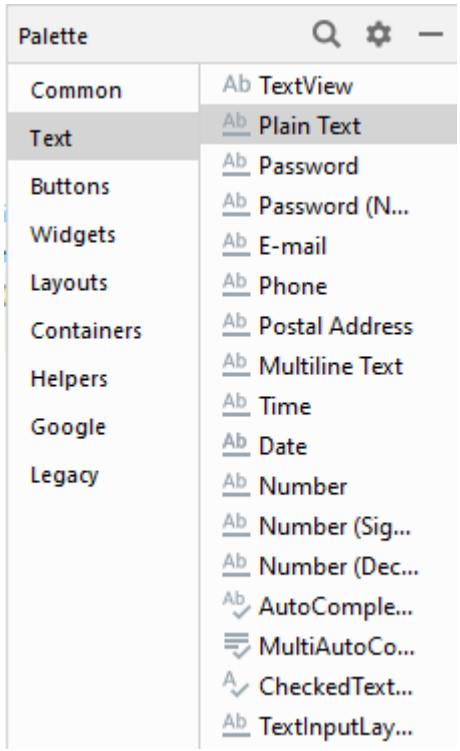
Imprescindible el identificador para poder recoger el texto

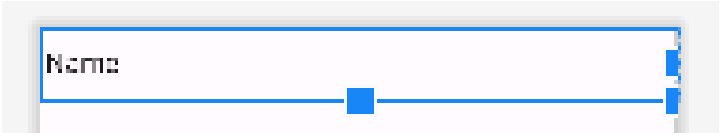
Permite indicar algunas características del texto a introducir: texto, números, contraseña, fecha, ….

Muestra un texto para informar al usuario de lo que tiene que escribir

Texto por defecto. Si el usuario lo borra se mostraría el texto del atributo hint (si existe)

EditText (2)



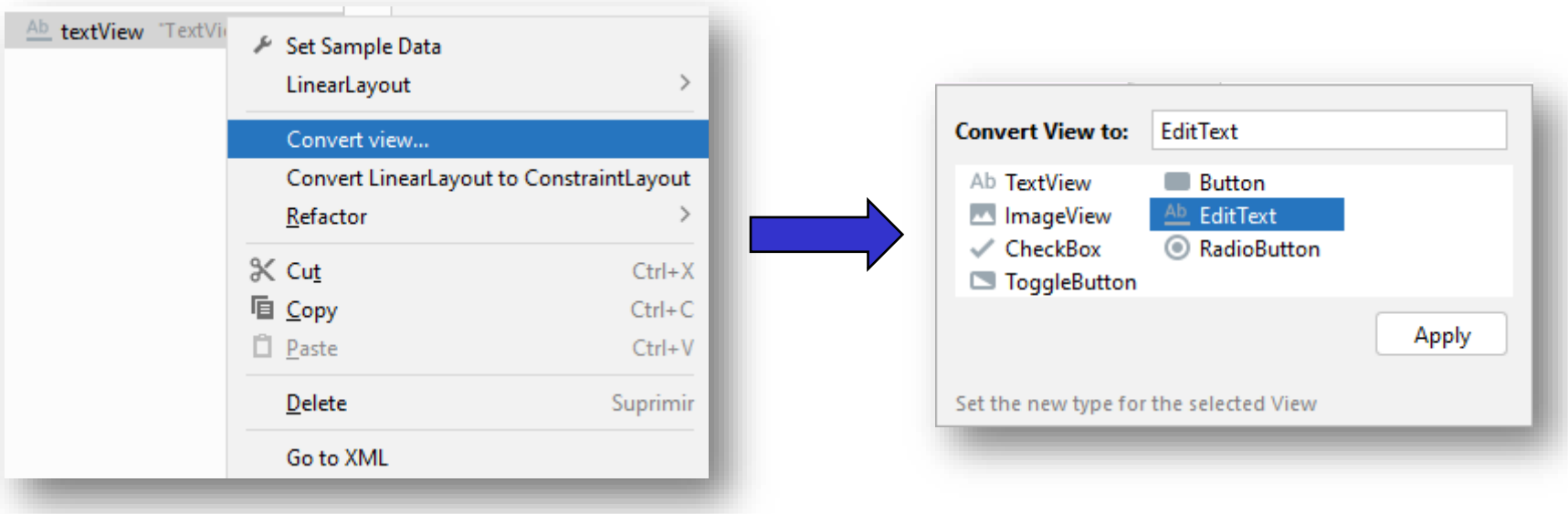


- En *Palette* del editor de diseño no se muestra directamente el componente *EditText*
- Se muestran algunos tipos *EditText* con un valor del atributo *inputType* concreto y otros atributos con valores por defecto

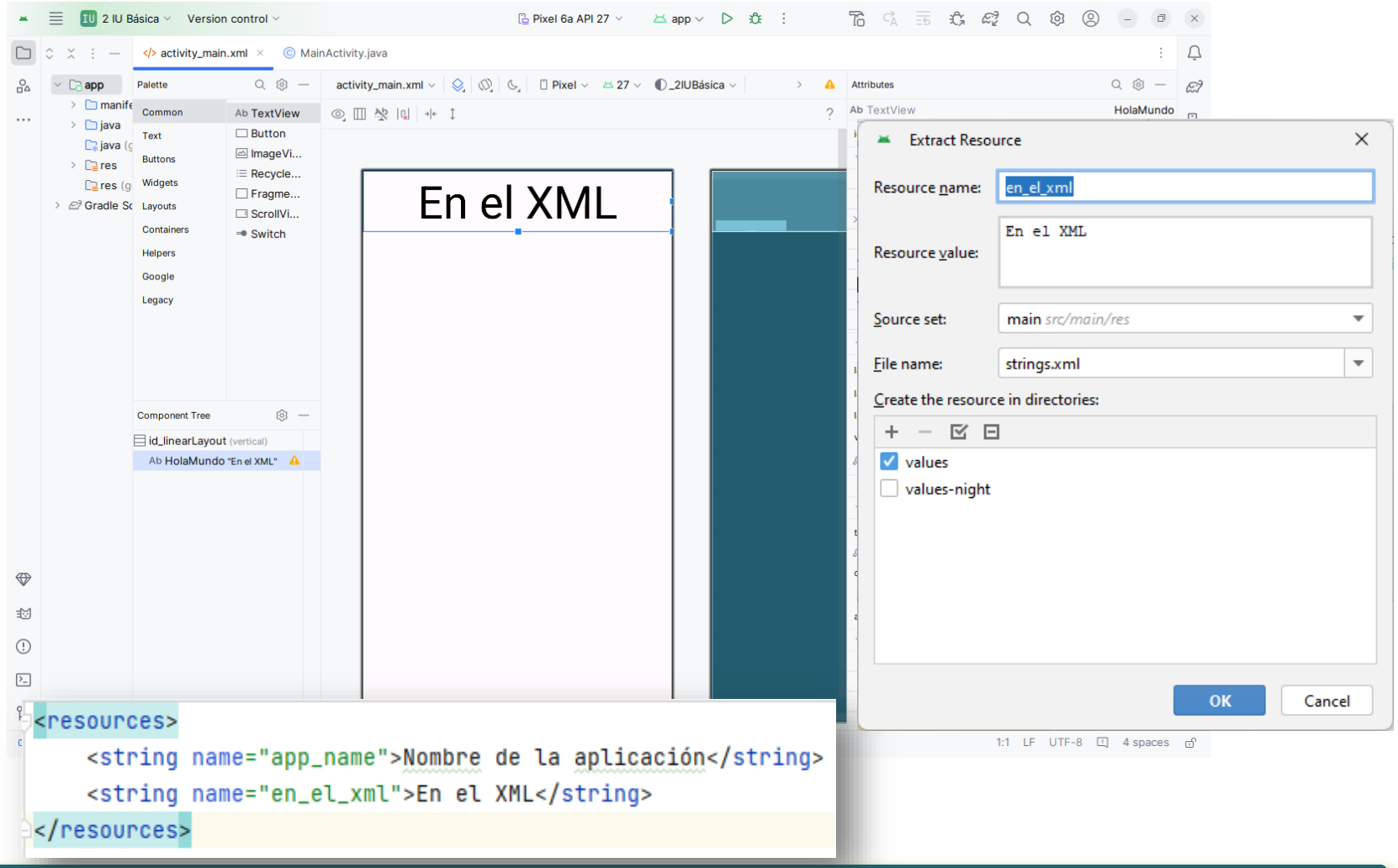
```
<EditText
    android:id="@+id/editTextText"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10"
    android:inputType="text"
    android:text="Name" />
```

EditText (3)

Una forma de añadir *EditText* genérico es insertar un *TextView* y convertirlo a *EditText*



Corrección de errores



Ejercicio 1 (1)

Crea un proyecto nuevo, con las siguientes características:

- Plantilla: *Empty Views Activity*
- Nombre: 2 IU Básica
- Paquete: dam.iubasica
- Directorio: 2-IUBasica

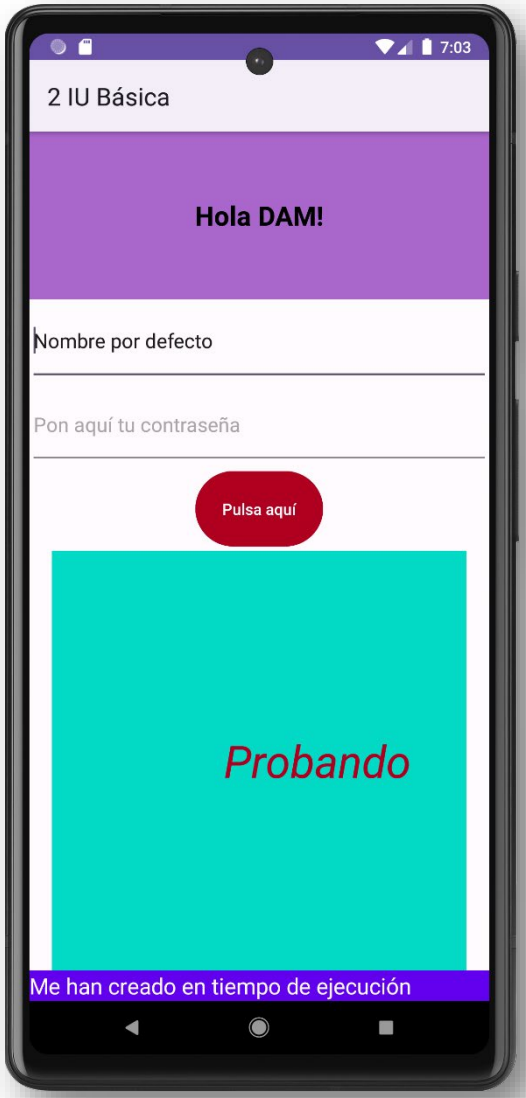
Ejecútalo en un terminal, y después:

1. Cambia el tema para que se vea la barra de título
2. Comenta o borra las dos instrucciones que extienden la interfaz a toda la pantalla
3. Cambia el *Layout* general del diseño, que es del tipo `ConstraintLayout`, por uno de tipo `LinearLayout` vertical
4. Modifica el `TexView` para poner el texto “Hola DAM” en negro, centrado horizontal y verticalmente, tamaño 24sp, negrita y fondo morado
5. Usando una distribución no equitativa, añade un `TexView` de mayor tamaño que `TexView` el anterior y fondo verde-azulado, con el texto “Probando” centrado verticalmente, alineado a la derecha, de tamaño 40sp, en cursiva, de color rojo. El layout debe estar separado del borde de la pantalla 20dp a la izquierda y 40dp a la derecha

Pistas: usa los atributos `layout_weight`, `gravity`, `layout_margin`, `textSize`, `textColor`, `textStyle` y `background` (color de fondo del contenedor)

Ejercicio 1 (2)

6. Modifica el `TexView` inferior (el del texto “Probando”) para añadir un margen derecho frente a su contenedor de 30dp (atributo `padding`)
7. Añade un par de `EditText`, uno para introducir un nombre de persona y otro para una contraseña. En el del nombre pon un texto por defecto y en el de la contraseña un texto como pista
8. Añade un `Button` con fondo de color rojo, centrado en el contenedor, con un relleno a izquierda y derecha de 80dp . Usa los atributos `layout_gravity` y `backgroundTint`
9. Crea un `TexView` en tiempo de ejecución con el texto “Me han creado en ejecución”. Sitúalo en la parte inferior de la pantalla, con el fondo azul, texto blanco y de tamaño 20. En esta página tienes los métodos públicos de un `TexView` y en esta de `View`.



Manejo de las vistas (*Views*) del diseño XML

Cuando se necesite consultar o modificar los componentes de la interfaz:

1. Guardar en una variable una referencia al componente con el método `findViewById()`:

   ```
   TextView tv = findViewById (R.id.identificador);
   ```

2. Manipular el componente a través de sus métodos. Ejemplo:

   ```
   tv.setText ("Nuevo TEXTO");
   ```

```
public class MainActivity extends AppCompatActivity {
 protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_main);
   TextView tv = findViewById (R.id.identificador);
   tv.setText("Te cambio el texto");
```

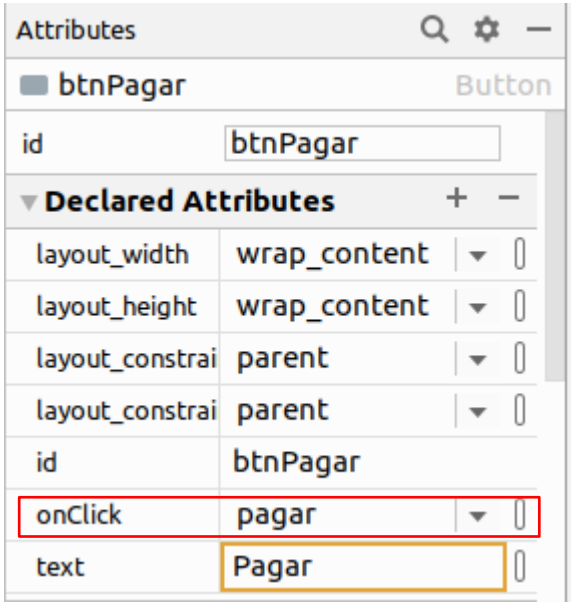
Manejadores de eventos (1)

- Hay componentes que generan eventos, por ejemplo el **`Button`**. Se les llama controles de interfaz (`widgets`)
- Existen dos mecanismos para asociar a un evento el código de procesamiento:
  1. **Especificando en el XML del control el evento a capturar y el nombre del método que lo procesará. Este método deberá estar implementado en la Actividad**

```
<Button
    android:id="@+id/btnPagar"
    android:layout_width="153px"
    android:layout_height="wrap_content"
    android:onClick="pagar"
    android:text="Pagar"
    android:textStyle="bold" >
</Button>
```

Evento *onClick*

método pagar()

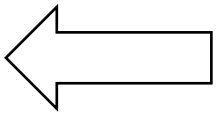


https://developer.android.com/guide/topics/ui/controls/button

Manejadores de eventos (2)

- Existen dos mecanismos para asociar a un evento el código de procesamiento:
  1. Especificando en el XML de la vista el evento a capturar y el nombre del método que lo procesará. Este método deberá estar implementado en la Actividad
  2. **Mediante código: asocia al control el manejador *(listener)* que procesará el evento**

```
<Button
    android:id="@+id/btnBuscar"
    android:layout_width="153px"
    android:layout_height="wrap_content"
    android:text="Buscar"
    android:textStyle="bold"
</Button>
```



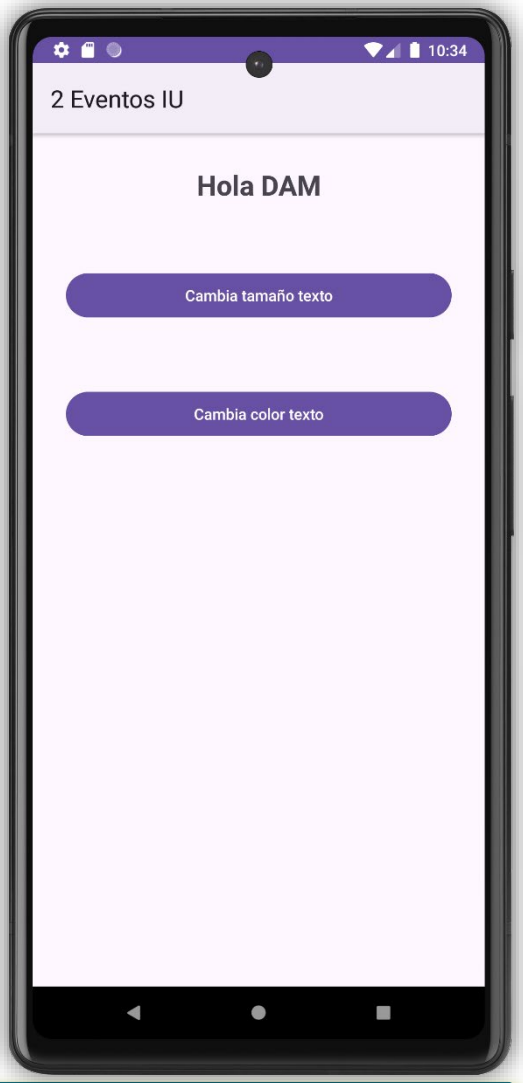
NO se especifica manejador de evento. Se definirá mediante código

```
// Obtener referencia al control btn_color
Button btnColor = findViewById (R.id.btn_color);

// Asignar manejador al control
btnColor.setOnClickListener(new ManejadorColorTexto());
```

https://developer.android.com/guide/topics/ui/controls/button

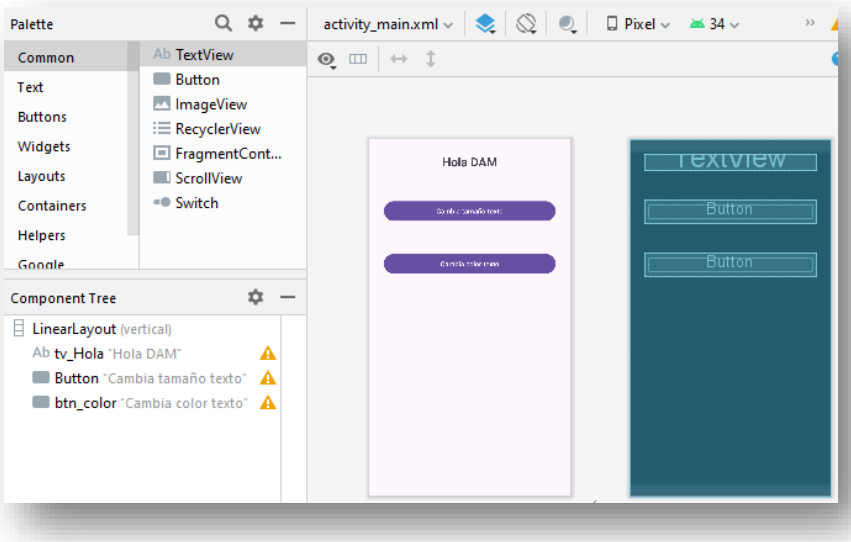
Ejemplo de manejo de eventos

- Este es un ejemplo de APP que maneja los eventos de dos botones que modifican el estilo de presentación de un texto.
- El código de los manejadores de los eventos modifica los atributos de presentación de un `TextView` con el texto inicial “Hola DAM”.
- Un manejador se define en el XML y el otro mediante código (con una clase interna).
- El tipo de diseño de la APP es un `LinearLayout` vertical.



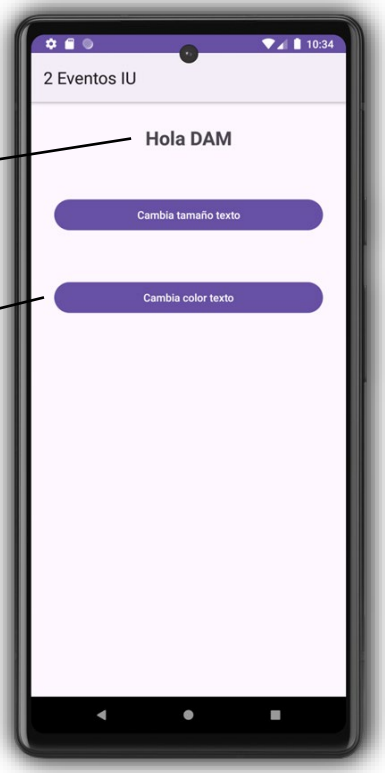
Layout de la APP

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/tv_Hola"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        android:text="Hola DAM"
        android:textAlignment="center"
        android:textSize="25sp"
        android:textStyle="bold" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        android:onClick="manejadorTamTexto"
        android:text="Cambia tamaño texto" />
```




```
    <Button
        android:id="@+id/btn_color"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="30dp"
        android:text="Cambia color texto" />

</LinearLayout>
```

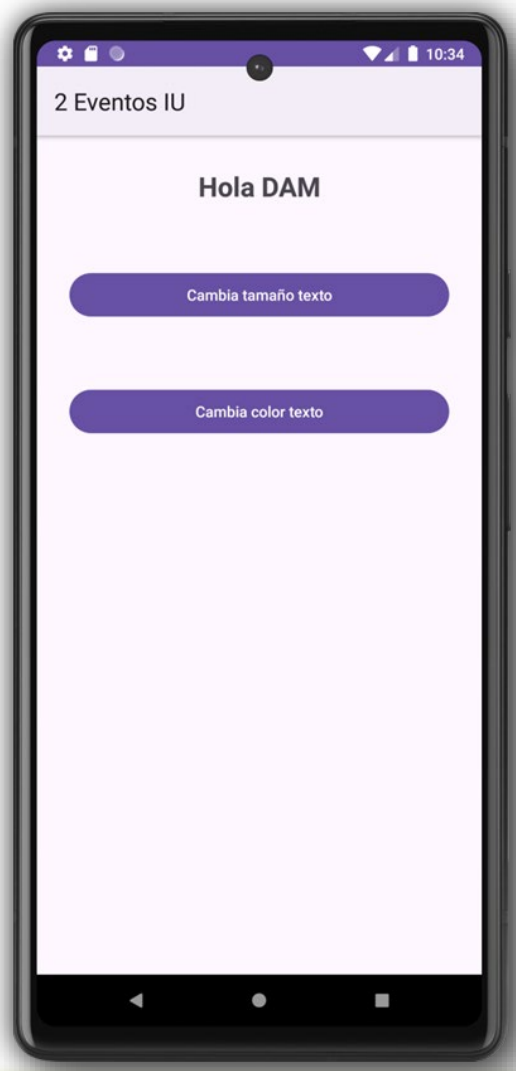
Código de aplicación (1)




```
package dam.eventosIU;

import ...

public class MainActivity extends AppCompatActivity {

    // Atributos de la clase
    private TextView texto;
    private float fuente=30;
    private int color=1;

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

        // Obtener referencia a los componentes de la interfaz
        texto = findViewById (R.id.tv_Hola);    // TextView "Hola DAM"
        final Button btnColor = findViewById (R.id.btn_color);    // Button btn_color

        // Asignar manejador al botón de cambio de color
        btnColor.setOnClickListener(new ManejadorColorTexto());
    }
```

Código de aplicación (2)

```
public void manejadorTamTexto(View v) {
    texto.setTextSize(fuente);
    fuente += 5;
    if (fuente == 50) fuente = 30;
}

class ManejadorColorTexto implements View.OnClickListener {
  @Override
  public void onClick (View v) {
    switch (color) {
        case 1: texto.setTextColor(Color.RED); break;
        case 2: texto.setTextColor(Color.GREEN); break;
        case 3: texto.setTextColor(Color.BLUE); break;
        case 4: texto.setTextColor(Color.CYAN); break;
        case 5: texto.setTextColor(Color.YELLOW); break;
        case 6: texto.setTextColor(Color.MAGENTA); break;
     }
     color++;
     if (color == 7) color = 1;
   } // onClick
} // Class ManejadorColorTexto


} // Class MainActivity
```



Ejercicio 2

- Crear un proyecto nuevo, con las siguientes características:
  - Nombre: 2 Eventos IU
  - Paquete: dam.eventosIU
  - Directorio: 2-EventosIU
- Implementar la aplicación del ejemplo anterior
- Añadir una entrada de texto (`EditText`) y un tercer botón que permita cambiar el texto del `TextView` superior
- Pistas:
  - La línea verde se puede conseguir con un `View` fijando una altura pequeña y un color de fondo. O desde el *Layout Editor* se puede obtener desde Palette/Widgets/Horizontal Divider
  - El código del manejador del nuevo botón leerá el contenido del `EditText` y lo escribirá en el `TexView`
  - El método `getText()` permite obtener el texto de un `EditText`
  - El método `setText` asigna un nuevo valor a un `TextView`

https://developer.android.com/reference/android/widget/EditText

Orientación del terminal

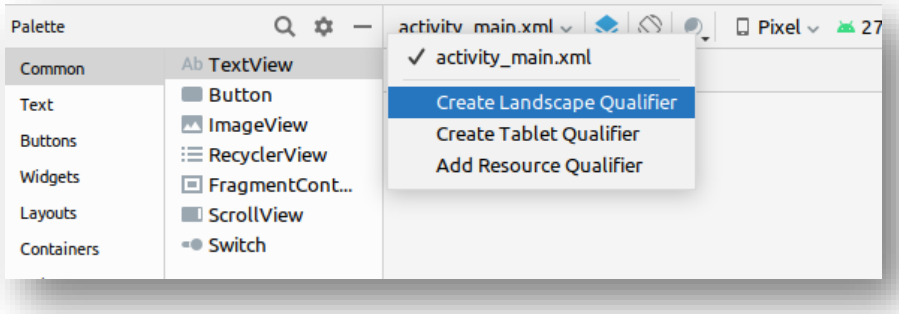
- Una actividad se puede visualizar con el terminal en modo vertical (retrato/*portraint*) o bien en modo horizontal (paisaje/*landscape*).
- Por defecto, al cambiar la orientación de la pantalla se destruye la actividad actual y se vuelve a construir. Esto se explicará más adelante en el *ciclo de vida de una actividad*
- Se puede indicar que no se modifique el *layout* al girar el terminal mediante el atributo *screenOrientation* del elemento `activity` en el fichero de manifest

```
<activity ... android:screenOrientation="portrait" ... >
```

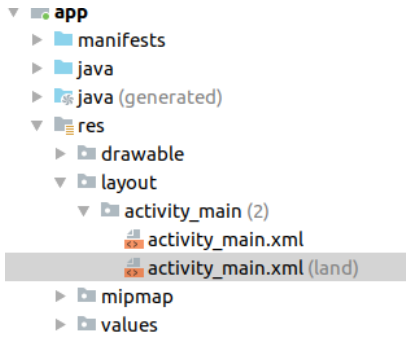
- También es posible definir *layouts* diferentes para cada orientación de la pantalla
- Según la orientación en que se encuentre el terminal, Android busca el fichero de *layout* en un directorio `<nombre_layout>-port` o `<nombre_layout>-land` dentro de `/res/layout`. Si no lo encuentra usa el que está en `/res/layout`.

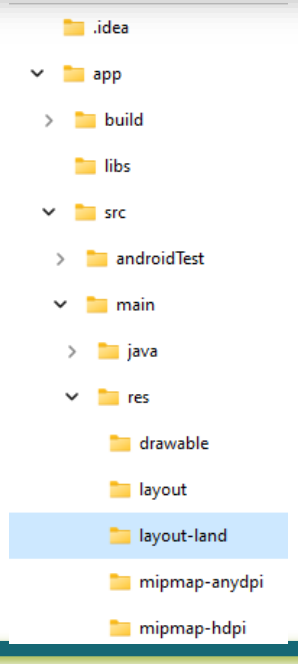
Crear un layout horizontal

- Dentro del *Layout Editor* (en vista *Desing* o *Split*), en la barra de herramientas, desplegar el menú con el nombre del fichero y elegir *Create Landscape Qualifier*



- Se creará un directorio llamado `layout-land` donde meterá una copia del fichero XML en edición

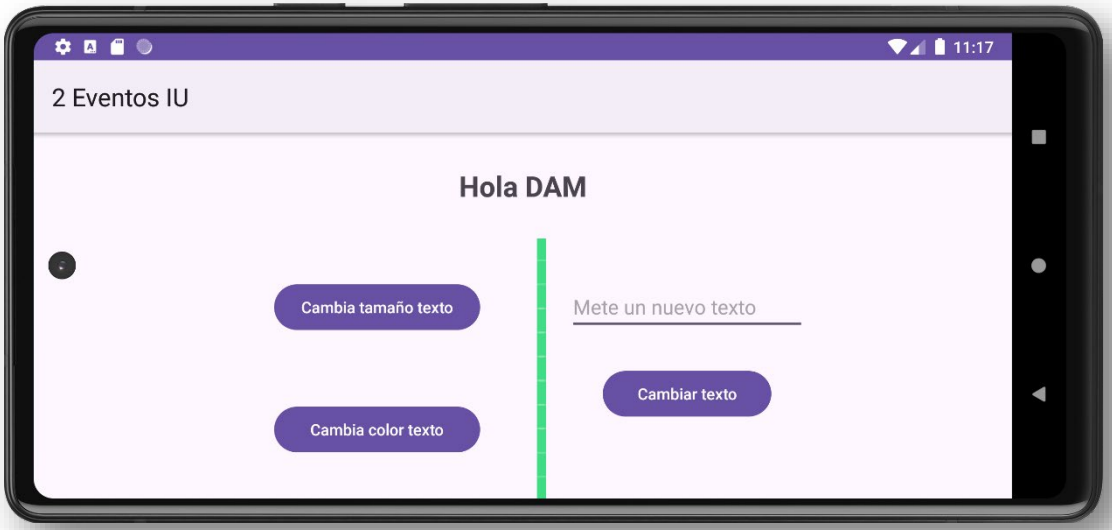


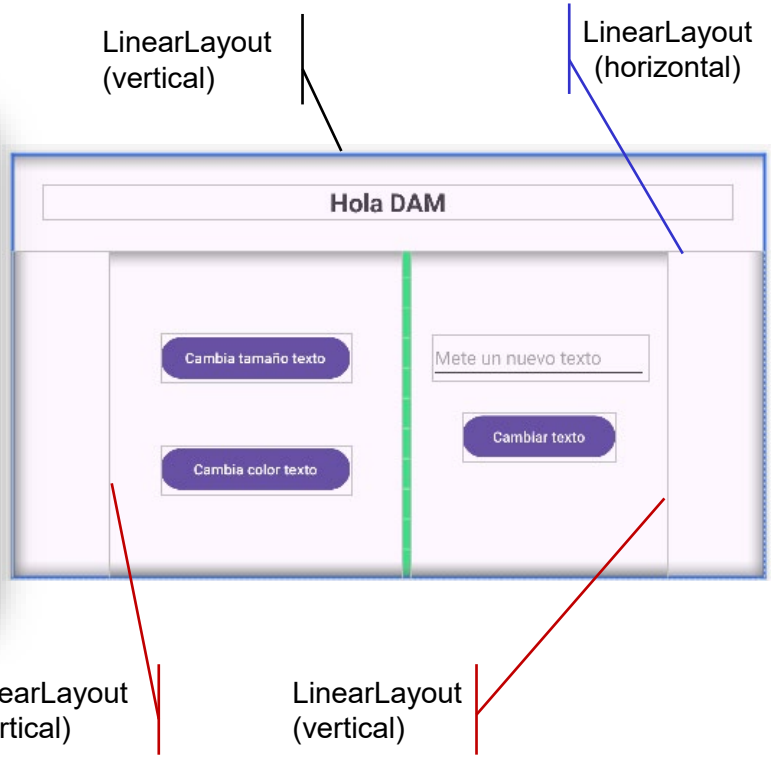


https://developer.android.com/studio/write/layout-editor#create-variant

Ejercicio 3 (1)

Modificar la aplicación del ejercicio 2 para adaptar el diseño vertical para cuando la pantalla esté en modo horizontal con el siguiente aspecto:





Ejercicio 3 (2)

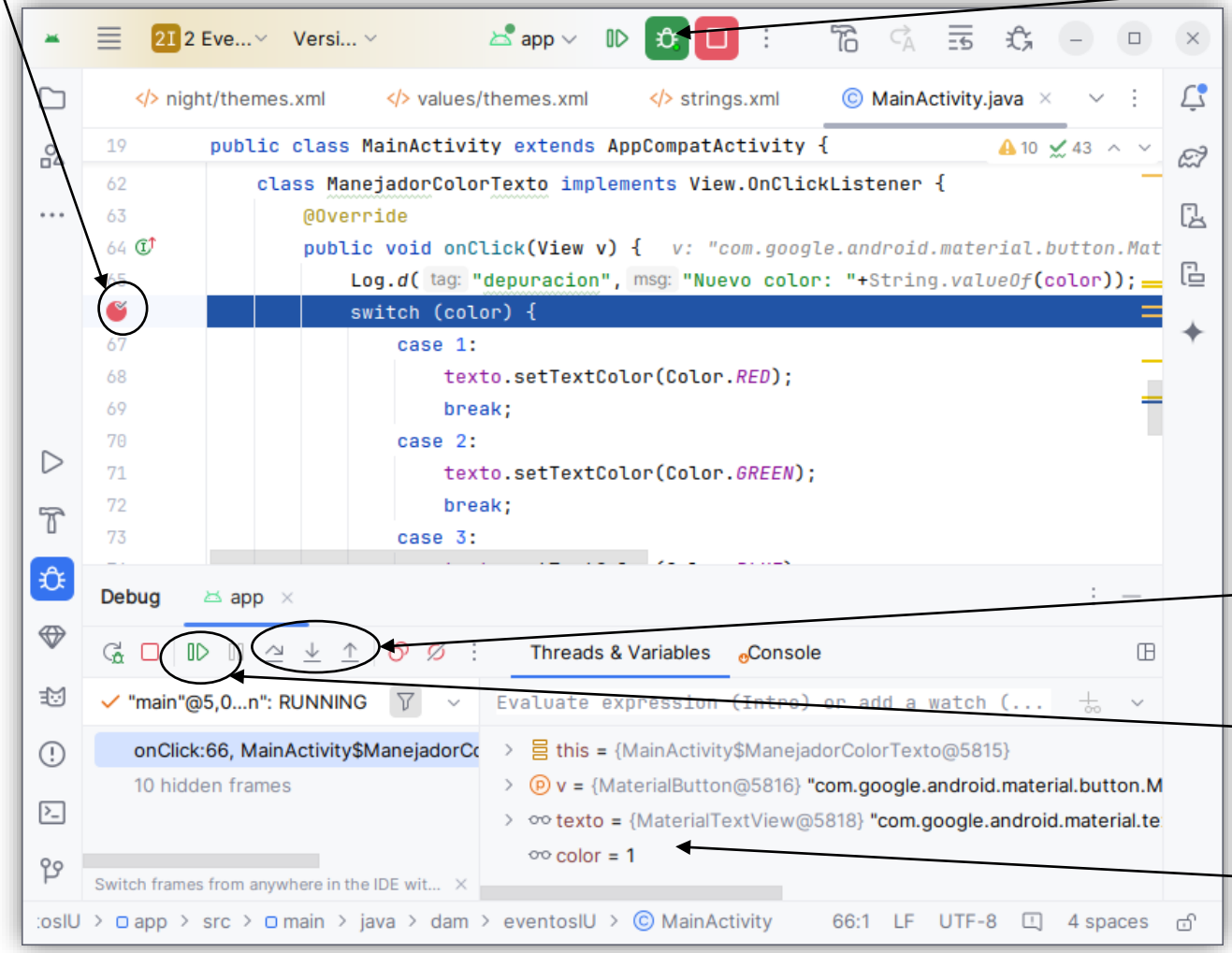
- Cada vez que se cambie la orientación, deber salir un mensaje usando `Toast` indicando en qué orientación está
- Si lo necesitas, puedes detectar la orientación con este código:

```
final int rotacion = getWindowManager().getDefaultDisplay().getRotation();
if (rotacion == Surface.ROTATION_0 || rotacion == Surface.ROTATION_180) {
    orientacionVertical=true;
} else {
    orientacionVertical=false;
}
```

Depuración

Punto de ruptura

Iniciar depuración

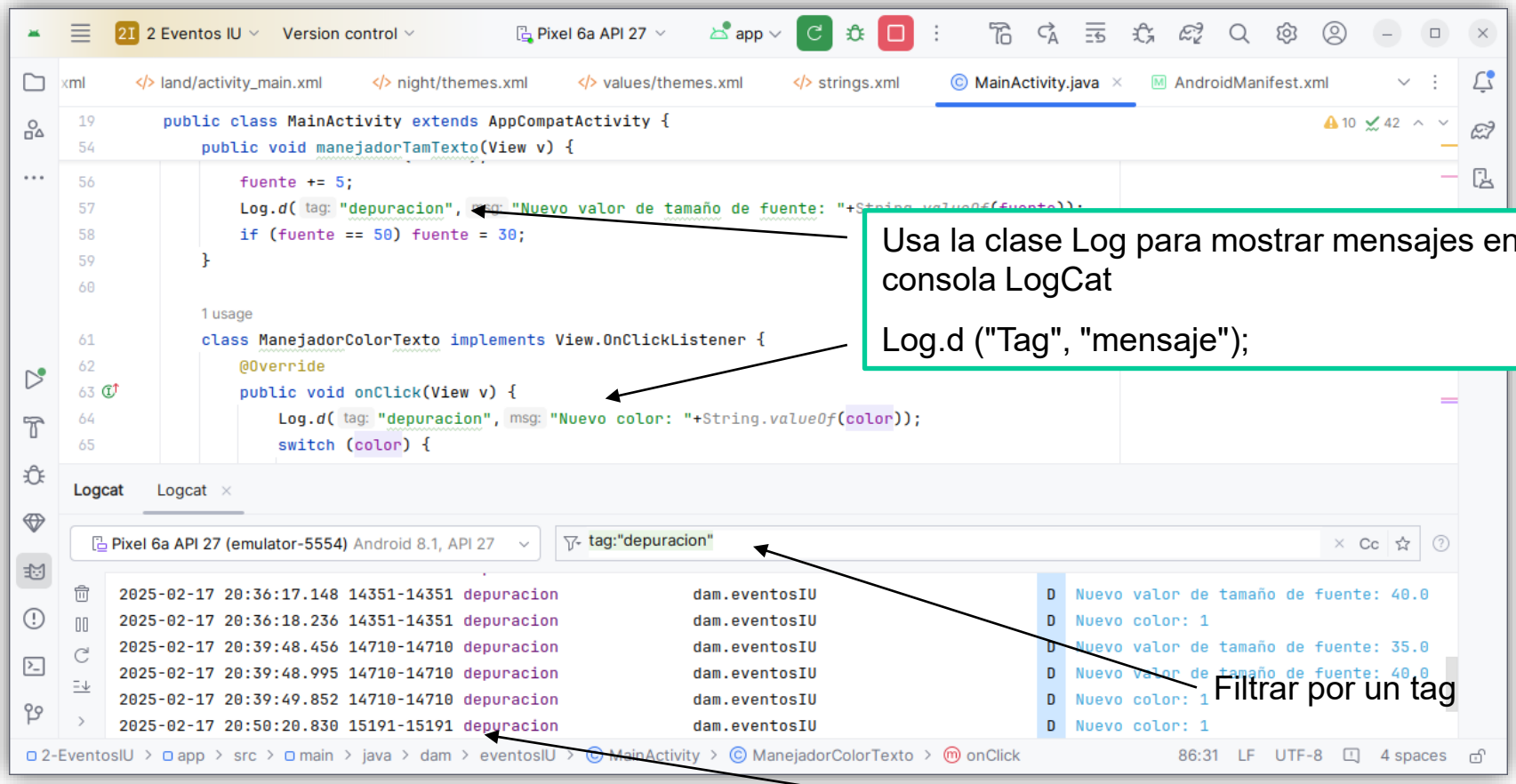


Botones para avanzar

Continuar hasta el siguiente punto de ruptura

Variables y atributos

Depuración con la clase Log



Usa la clase Log para mostrar mensajes en la consola LogCat

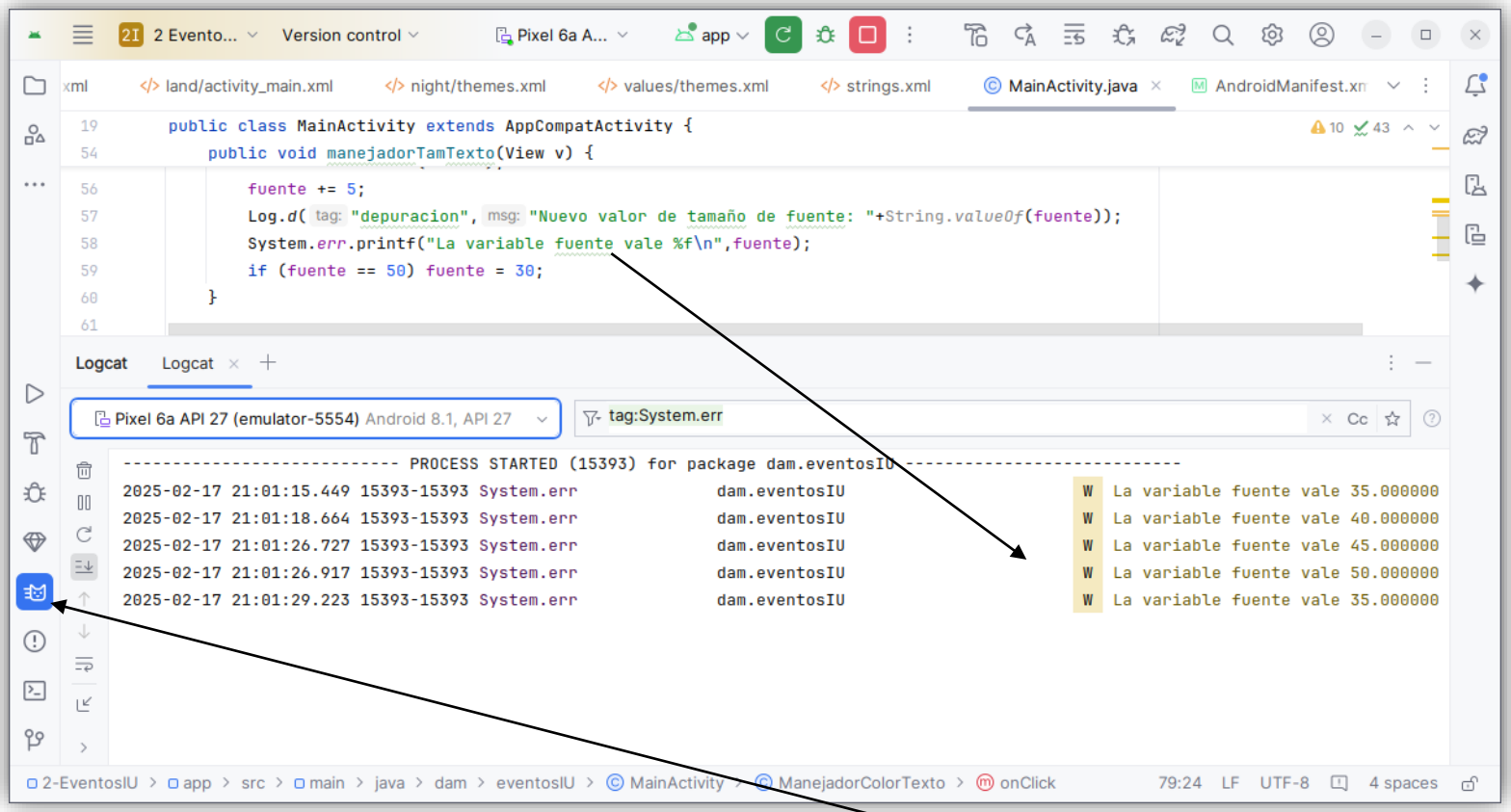
Log.d ("Tag", "mensaje");

Filtrar por un tag

Consola LogCat

View logs with Logcat

Depuración con printf()



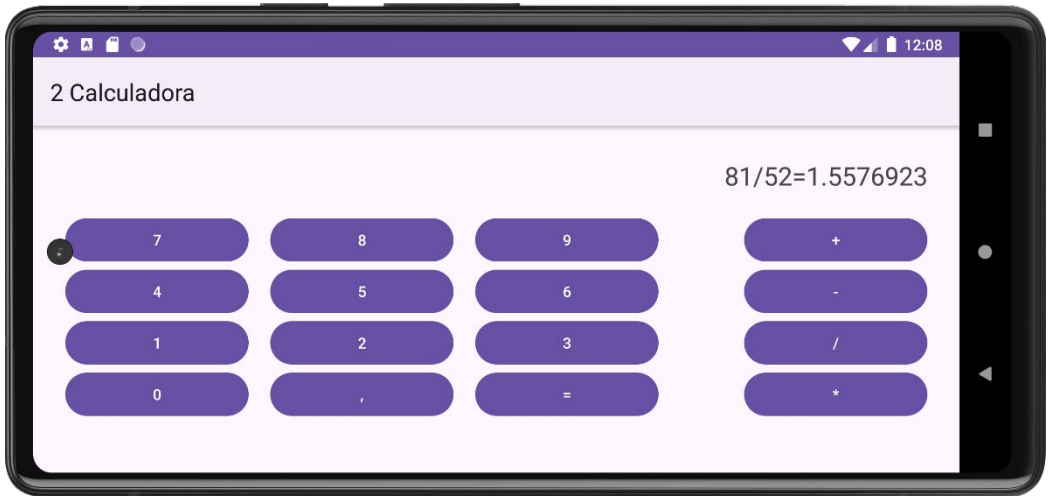
Consola LogCat

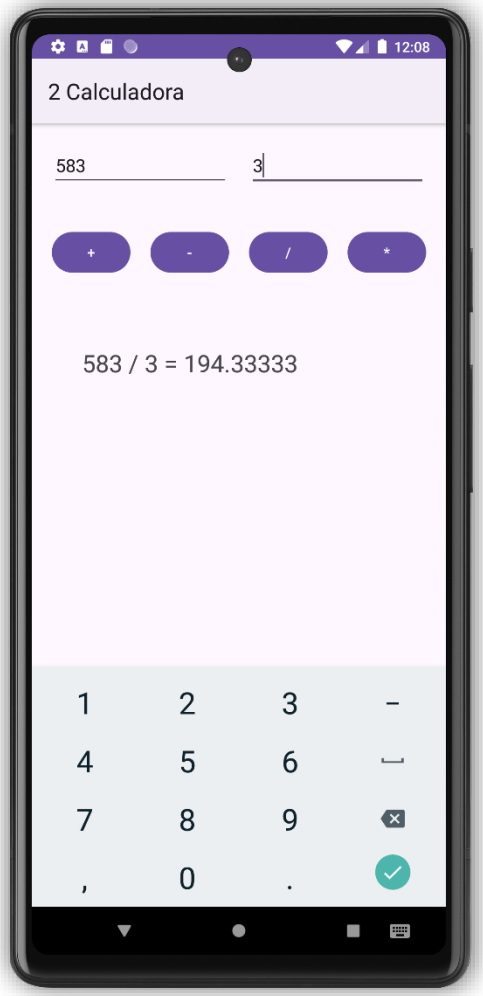
Ejercicio 4 (1)

Crea una aplicación que implemente una calculadora simple como la mostrada en las figuras

Características del proyecto:

- Nombre: 2 Calculadora
- Paquete: dam.calculadora
- Directorio: 2-Calculadora
- **Cambiar el icono por defecto**

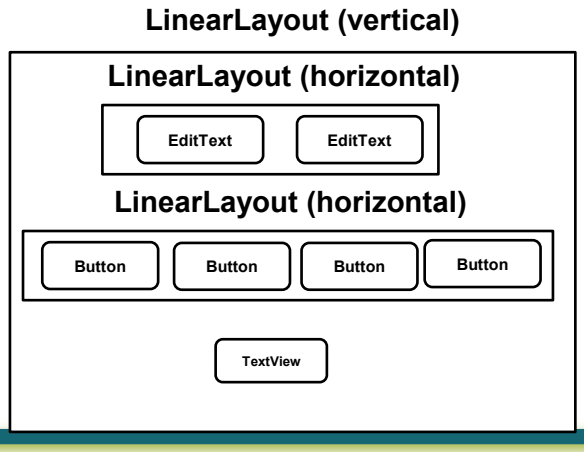




Ejercicio 4 (2)

Pistas para la orientación vertical:

- Usa un `EditText` para la entrada de cada operando
- Usa el atributo `inputType` para limitar la entrada del `EditText` a dígitos
- Usa `getText.toString()` para recuperar el valor de los operandos en los `EditText`
- El manejador asociado a cada botón de operación debe recoger los datos de los operandos, realizar la operación y escribir el resultado en el `TextView` inferior
- `Float.parseFloat` convierte las cadenas obtenidas de los `EditText` a cantidades de tipo `float`
- `String.valueOf()` convierte un `float` a `String`
- Para realizar el diseño puedes usar `LinearLayout` anidados:



Ejercicio 4 (3)

Pistas para la orientación horizontal:

- Usa el `TableLayout` para meter todos los dígitos y operaciones
- Crea una variable `String` donde almacenar cada dígito u operación. Al pulsar los botones de los dígitos, invoca a un manejador que simplemente concatena los datos de esa variable `String` con el nuevo dígito y lo muestra en el `TextView` de resultados.
- Al pulsar el botón = el manejador asociado obtiene los operandos de la variable `String`, realiza la operación y lo muestra en el `TextView` de resultados. De un `String` puede obtenerse los operandos usando una expresión regular mediante `split("[+\\-*/]")`
- Para que todos los botones de los dígitos tengan asociado el mismo manejador, usa el atributo `tag` del botón al que le asocias el valor del dígito. Igualmente, con los botones de las operaciones, para que puedan invocar al mismo manejador, pon en cada botón un atributo `tag` con el carácter de la operación. El atributo `tag` del botón se recoge con `getTag()`
- Restricción: para facilitar el algoritmo, las operaciones tienen que ser únicamente de 2 operandos. De esta forma, en cuanto se muestra el resultado, la siguiente pulsación de un nuevo dígito borrará el contenido del `TextView` de resultados para iniciar una nueva operación. O también puedes implementar el típico botón de borrado [C].