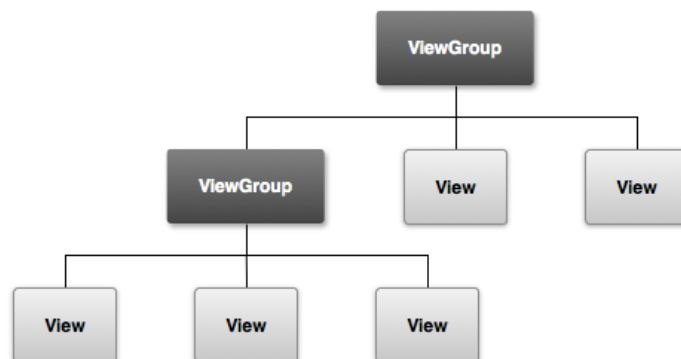


INDICE

<u>1.1.</u>	<u>INTRODUCCIÓN</u>	<u>1</u>
<u>1.2.</u>	<u>ATRIBUTOS DE LOS ELEMENTOS VISUALES</u>	<u>3</u>
<u>1.3.</u>	<u>EJEMPLOS DE ATRIBUTOS DE LOS ELEMENTOS VISUALES</u>	<u>4</u>
<u>1.3.1.</u>	<u>DISEÑO PANTALLA EN MODO GRÁFICO</u>	<u>4</u>
<u>1.3.2.</u>	<u>DISEÑO DE PANTALLA EN XML</u>	<u>7</u>
<u>1.3.3.</u>	<u>ATRIBUTOS DE ESTILO, COLOR, TAMAÑO,...</u>	<u>13</u>

1.1. Introducción

- La **interfaz de usuario (UI)** es cualquier objeto que el usuario pueda ver e interactuar.
- Una **vista** es un objeto que dibuja algo en la pantalla (botón, etiqueta, radio, casilla de verificación, etc.). El usuario puede interactuar con ese objeto.
- Un **ViewGroup** es una vista invisible especial que puede contener Vistas y otros ViewGroups. Se utiliza para organizar la posición de las vistas / grupos de vistas en la pantalla.



- Tanto las vistas como los grupos de vistas se pueden crear como objetos en Java o utilizando archivos XML.
- Ejemplo de un archivo XML.

```

1 <? xml version = "1.0" encoding = "utf-8"?>
2 <LinearLayout xmlns: android = "http://schemas.android.com/apk/res/android"
3     android: layout_width = "fill_parent"
4     android: layout_height = "fill_parent"
5     android: orientación = "vertical" >
6     <TextView android: id = "@ + id / text"
7         android: layout_width = "wrap_content"
8         android: layout_height = "wrap_content"
9         android: text = "Soy un TextView" />
10    <Botón de android: id = "@ + id / button"
11        android: layout_width = "wrap_content"
12        android: layout_height = "wrap_content"
13        android: text = "Soy un botón" />
14 />
15 </LinearLayout>

```

- En las líneas marcadas tenemos el inicio de definición de cada uno de los **elementos** que componen la pantalla.
- <LinearLayout>, <TextView> y <Button> son Views (Vistas) que definen ese tipo de objetos visuales.
- Para cada **<elemento>** de un archivo XML tenemos una clase Java asociada que nos permite manipular ese elemento o crear uno nuevo en tiempo de ejecución.
 - El elemento XML **<TextView>** está asociado con la clase Java **TextView**. Un TextView es equivalente a una etiqueta en otros lenguajes de programación.
 - El elemento XML **<LinearLayout>** crea el ViewGroup en Java **LinearLayout**. Un LinearLayout es una extensión de un ViewGroup.
- Las ventajas de usar XML:
 - Le permite separar las capas de presentación (UI) de las capas de programación.
 - Se pueden realizar modificaciones en la interfaz de usuario sin tocar el código Java
- Como ya se vio anteriormente:
 - Los archivos XML se declaran dentro de **/res**
 - Recursos, en este caso, XML, se accede desde el código fuente de Java a través de la [clase Java R](#).
 - Por ejemplo, este es el código que genera Android Studio cuando creamos una nueva actividad con un diseño asociado:

```

1 @Override
2 protected void onCreate(Bundle savedInstanceState) {
3     super.onCreate(savedInstanceState);
4     setContentView(R.layout.activity_ud02_01_a2_activities);
5 }

```

- Referencias:
 - Interface gráfica: <http://developer.android.com/guide/topics/ui/overview.html>
 - Un buen diseño: <http://developer.android.com/design/get-started/principles.html>
 - Vistas: <http://developer.android.com/reference/android/view/View.html>

1.2. Atributos de los elementos visuales

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="fill_parent"
4     android:layout_height="fill_parent"
5     android:orientation="vertical" >
6     <TextView android:id="@+id/my_text"
7         android:layout_width="wrap_content"
8         android:layout_height="wrap_content"
9         android:text="I am a TextView" />
10    <Button android:id="@+id/my_button"
11        android:layout_width="wrap_content"
12        android:layout_height="wrap_content"
13        android:text="I am a Button" />
14 />
15 </LinearLayout>

```

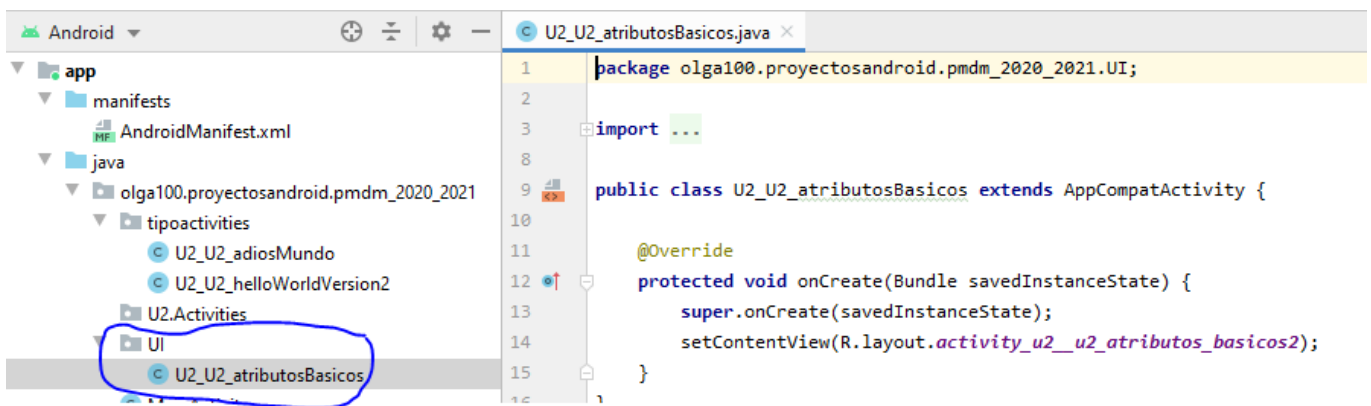
- **xmlns:** indica el espacio de nombres utilizado para definir los atributos xml:
http://es.wikipedia.org/wiki/Espacio_de_nombres_XML
- **ID:** a los elementos se les puede asignar un número entero, que se crea en la compilación (Java Class R). Esa ID es la razón por la que se accede tanto en Java como en otras referencias XML a ese elemento.
- **android: id.** ID de artículo o control. Tiene el siguiente **formato "@ + id / cadena de texto :**
 - @: indica que lo que sigue es un recurso.
 - + : indica que el ID no existe y lo crea (recuerde en **la clase Java R**)
 - **tipo de recurso:** en este caso **id**, pero podría ser: string, drawable, layout, etc.
 - **cadena de texto:** es el nombre que se le da al identificador.
 - Ejemplo: **android: id = "@ + id / my_button"**
 - Para hacer referencia a ese recurso desde cualquier otro recurso es sin el "+": **android: id = "@ id / my_button"**.
- **android: texto.** Texto del elemento. Puede especificar:
 - **directamente :** android: text = "Soy un botón"
 - **a través de un recurso:** android: text = "@ string / button_text". En este caso, **button_text** se definirá en otro archivo XML, por ejemplo strings.xml.
- **android: layout_height** y **android: layout_width.** Especifican las dimensiones del elemento con respecto al Layout que las contiene o el contenido del elemento. Puede tomar los siguientes valores:
 - **valor fijo :** en dp
 - **"match_parent":** la altura o el ancho del elemento coincidirá con la altura o el ancho del elemento que lo contiene.
 - **"wrap_content":** la altura o el ancho del elemento se ajustará a la altura o el ancho del contenido del mismo elemento.
- Hay más atributos comunes a los elementos visuales. Luego mostraremos más con ejemplos.

1.3. Ejemplos de atributos de los elementos visuales

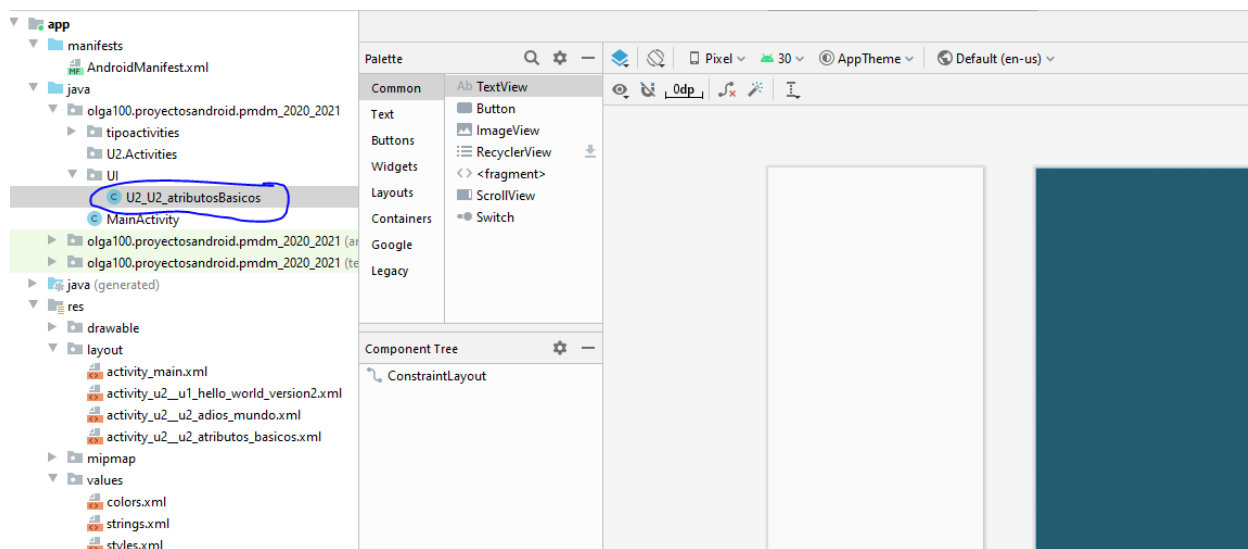
1.3.1. Diseño pantalla en modo gráfico

- Partiremos del proyecto anterior.
- Comenzaremos creando un nuevo paquete llamado: **UI**.
- Dentro de este paquete cree una nueva actividad llamada: **UD2_2_atributosBasicos**
- Esta vez usaremos inicialmente un diseño llamado **ConstraintLayout** para organizar los elementos visuales dentro de la pantalla, luego cambiaremos a un diseño llamado **LinearLayout**. Los Layouts se verán en el siguiente tema.

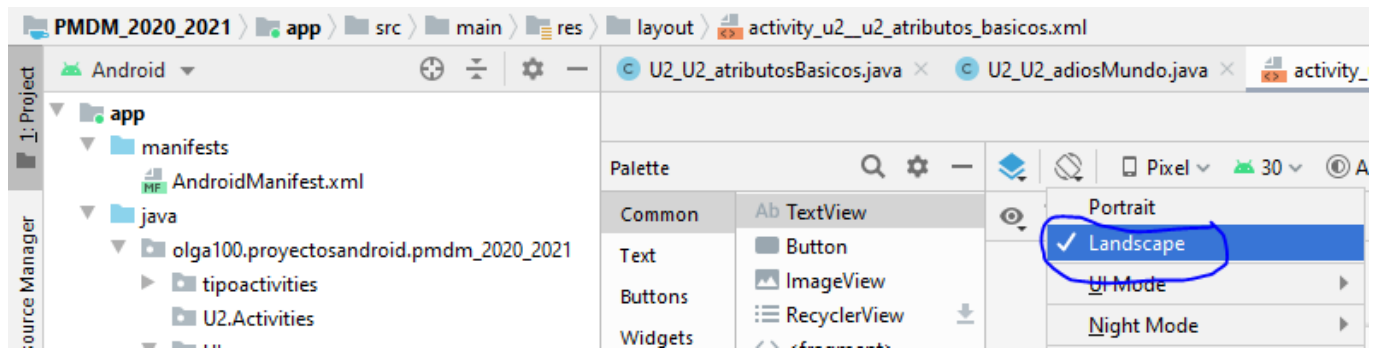
Diseño pantalla en modo gráfico



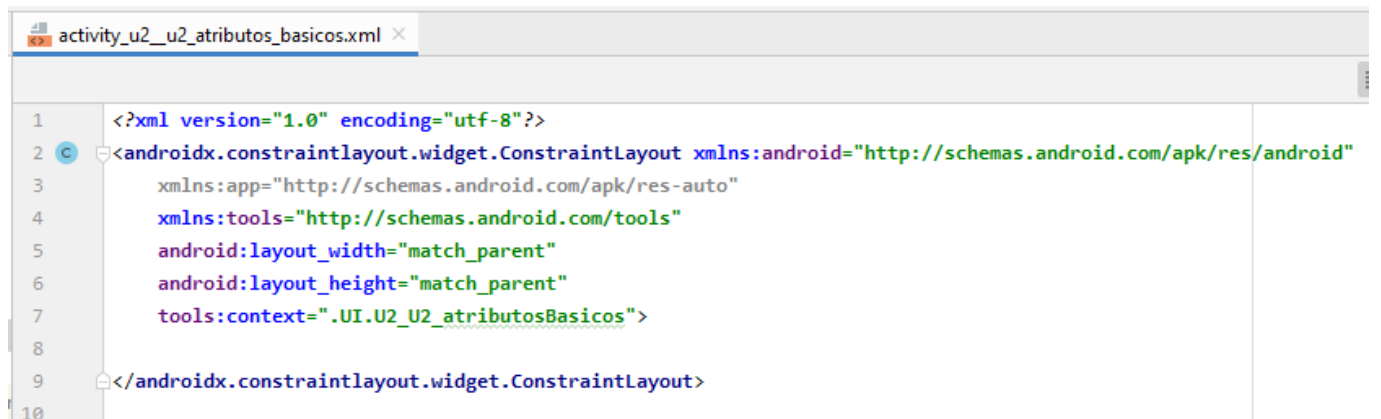
Creamos el paquete y la actividad



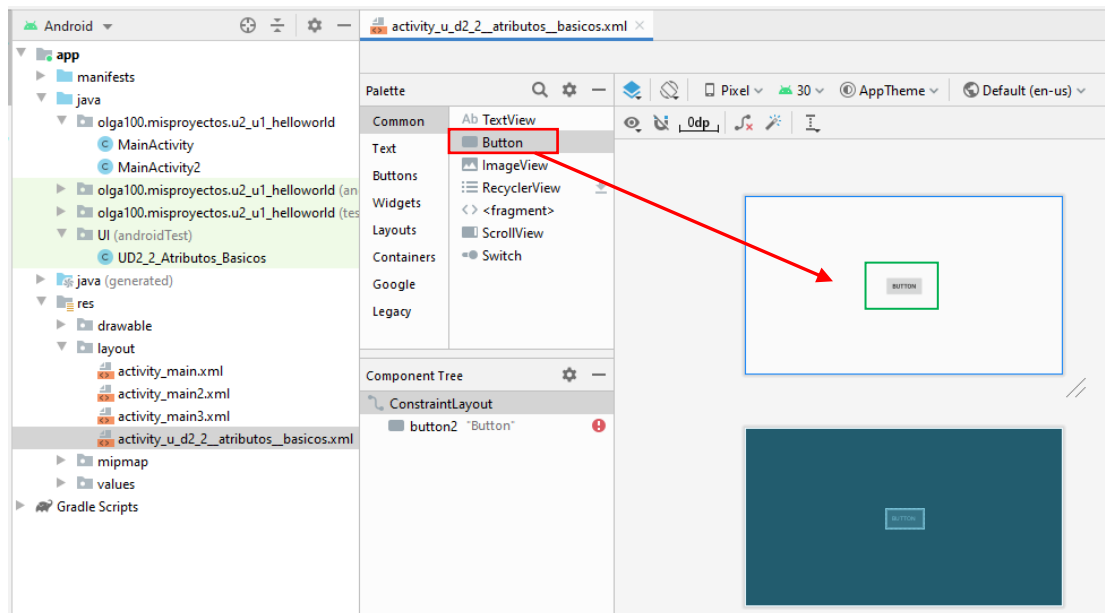
Vayamos al archivo xml que define la pantalla, dentro de **"/res/layout"**. Se presenta en una pantalla que simula un dispositivo. Podemos intercambiar entre el modo gráfico y el modo texto en las pestañas superior derecha



El cual podemos manipular, por ejemplo para poner en apaisado



Podemos ver el archivo como se describe la pantalla mediante elementos/controles XML



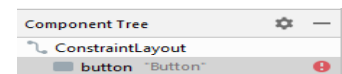
Podemos definir la pantalla de modo gráfico o en xml. En cualquier caso lo realizado en uno de los casos tiene su correspondiente traducción en el otro. Por ejemplo si se arrastra un botón en modo gráfico a la pantalla y lo situamos donde deseemos...

- El archivo xml se vería así: Las líneas correspondientes al botón son las marcadas:

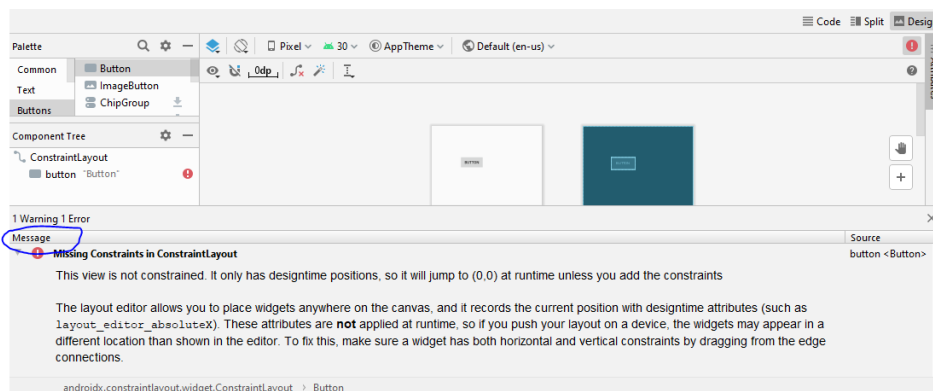
```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout 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      tools:context="UI.UD2_2_Atributos_Basicos">
8
9      <Button
10         android:id="@+id/button2"
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:text="Botón"
14         tools:layout_editor_absoluteX="324dp"
15         tools:layout_editor_absoluteY="185dp" />
16 </androidx.constraintlayout.widget.ConstraintLayout>
  
```

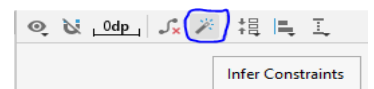
Observar que tanto en la vista 'Design' como en la vista 'Code' se muestra una **advertencia** "This view is not constrained. It only has designtime positions, so it will jump to (0,0) at runtime unless you add the constraints" lo que viene a decir, es que faltan por definir restricciones.



Para ver más información sobre el warning, hacer clic sobre el icono:



En el caso de que no se añadan las restricciones, el control, en este caso, el botón se colocará en la posición (0,0). Se pueden añadir manualmente o existe una varita mágica que lo hace por nosotros añadiendo las restricciones necesarias.



De esta forma quedaría el 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=".UI.U2_U2_atributosBasicos">

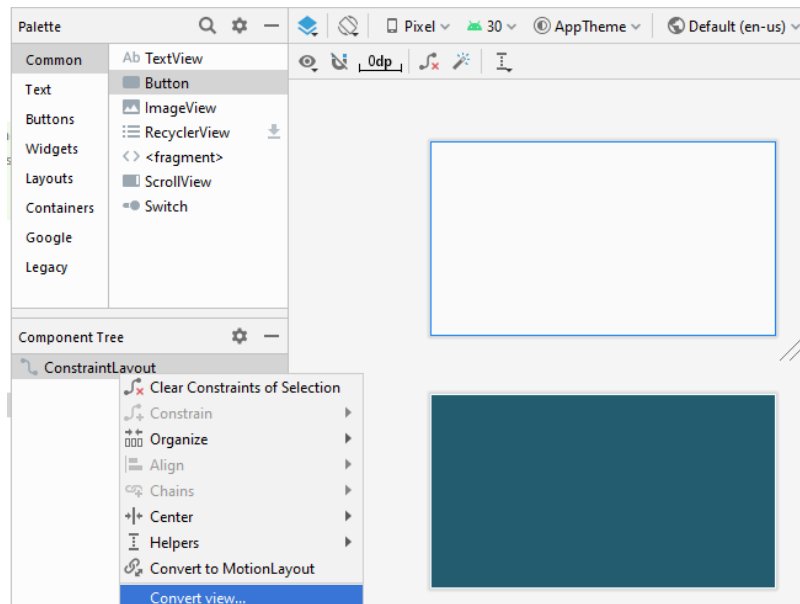
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="106dp"
        android:layout_marginTop="113dp"
        android:text="Button"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

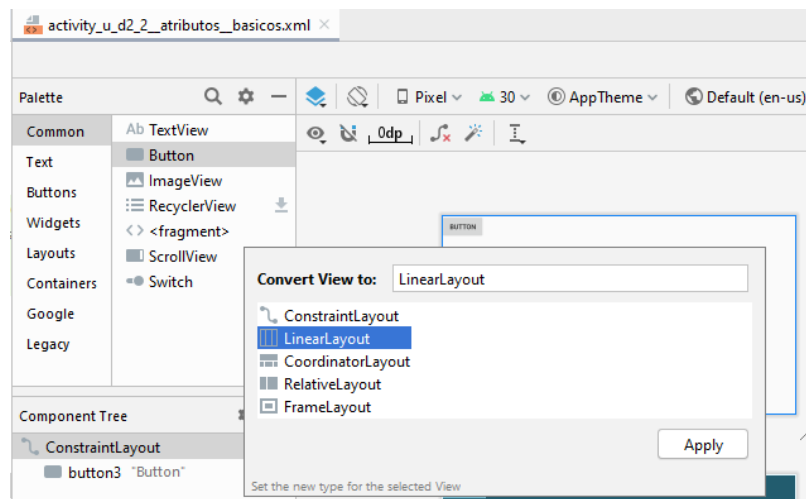
Como veremos posteriormente cada layout tendrá unos parámetros específicos.

1.3.2.Diseño de pantalla en xml

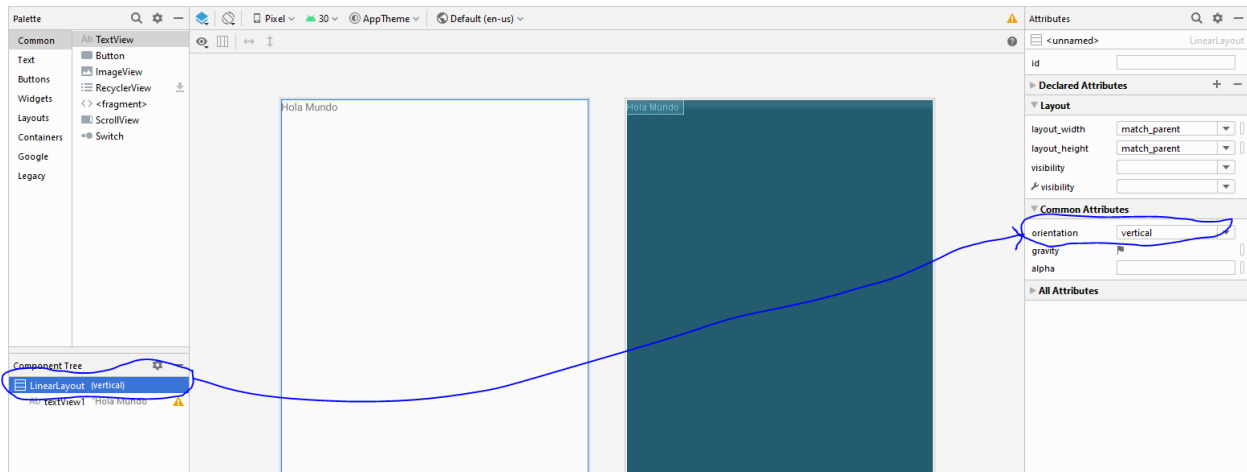
Diseño Layout LINEAL



Eliminamos el botón agregado y agregamos algo de texto si aún no lo trae por defecto. Cambiamos el tipo de diseño. Para hacer esto, haga clic con el botón derecho en 'ConstraintLayout' y elegir la opción **Convert View**.



Para hacer un diseño sencillo cambiaremos el tipo de layout a Lineal (Estos se estudiarán más adelante). Agregaremos el botón nuevamente.



Veremos más adelante que este tipo de layouts tiene dos posibles orientaciones. Cambiaremos a *Vertical*.


```

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="vertical"
8      tools:context=".UI.U2_u2_atributosBasicos">
9
10
11     <TextView
12         android:id="@+id/textView2"
13         android:layout_width="match_parent"
14         android:layout_height="wrap_content"
15         android:text="Hola Mundo" />
16
17     <Button
18         android:id="@+id/button2"
19         android:layout_width="match_parent"
20         android:layout_height="wrap_content"
21         android:text="Soy un botón" />
22 </LinearLayout>

```

- Observar cómo se cambian las líneas marcadas. En el caso de la línea 2 le indicamos al layout lineal que los elementos que contiene están dispuestos en modo vertical (línea 7), uno tras otro.

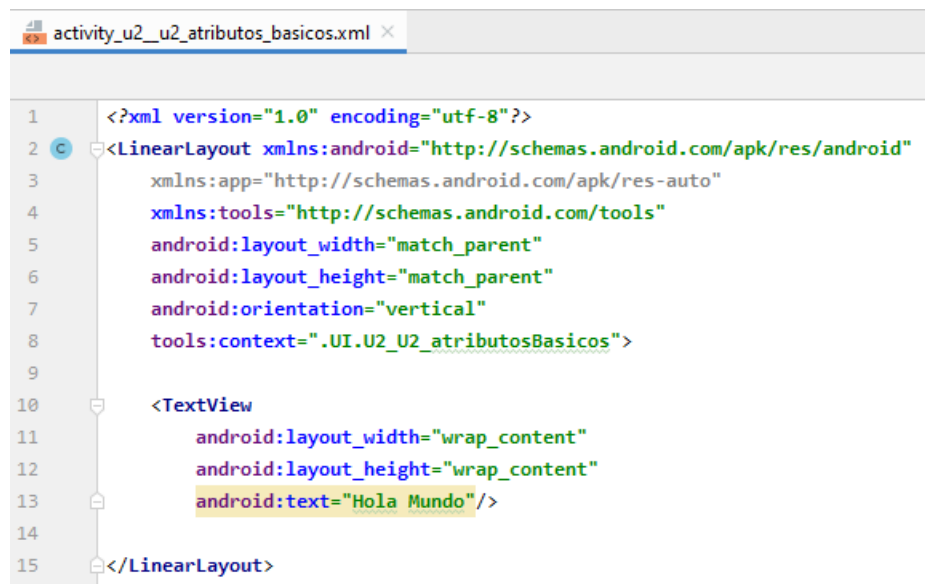
Diseño pantalla en xml

```

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="vertical"
8      tools:context=".UI.U2_u2_atributosBasicos">
9
10     <Te
11     Ab TextView
12     TextClock
13     TextSwitcher
14     TextureView
15     CheckedTextView

```

Eliminamos los controles anteriores (TextView Button) y comenzamos creando un elemento visual y si presionamos **CTRL + BARRA ESPACIADORA** el IDE ofrece los posibles elementos que comienzan con ese nombre.



```

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="vertical"
8      tools:context=".UI.U2_U2_atributosBasicos">
9
10     <TextView
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:text="Hola Mundo"/>
14
15 </LinearLayout>

```

En este caso vamos a crear un `<TextView>`, una etiqueta. Los posibles elementos a crear se verán en un tema posterior.



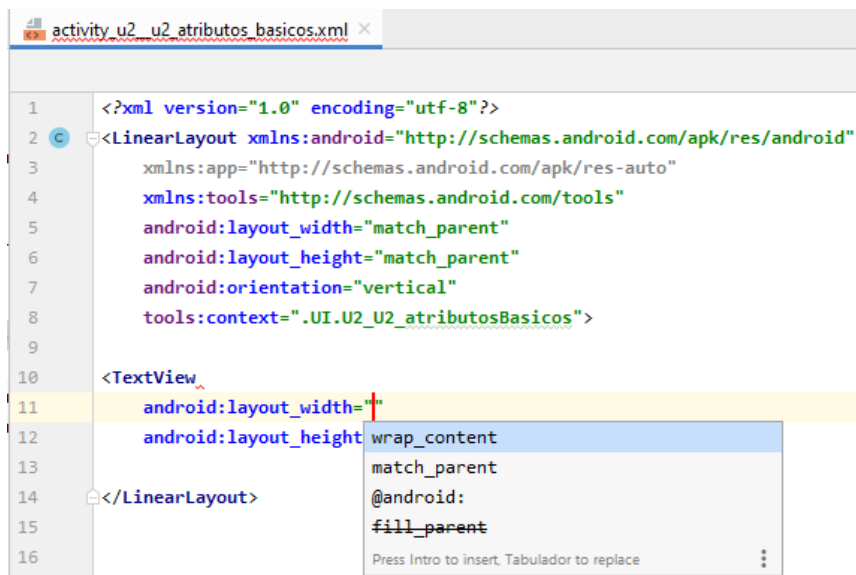
```

<TextView
    android:layout_width
    layout_width(required)
    layout_gravity
    layout_height(required)
    layout_margin
    layout_marginBottom
    layout_marginEnd
    layout_marginHorizontal
    layout_marginLeft
    layout_marginRight
    layout_marginStart
    layout_marginTop
    layout_marginVertical
    Press Intro to insert, Tabulador to replace

```

Empezamos indicando el ancho... Escribimos android: **CTRL + barra espaciadora**, o simplemente presionamos **CTRL + barra espaciadora** y nos ofrece los posibles atributos. Cuantas más letras escribamos en el atributo, más refinamos la búsqueda. En este caso seleccionamos **android: layout_width**.

También podemos escribir las letras de un atributo (sin android:) y presionar **CTRL + barra espaciadora** y nos da los atributos que coinciden con la escritura.



```

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="vertical"
8      tools:context=".UI.U2_U2_atributosBasicos">
9
10     <TextView
11         android:layout_width="
12         android:layout_height="wrap_content
13         match_parent
14         @android:
15         fill_parent
16

```

Si ya existen posibles valores predefinidos para ese atributo "", se lo informaremos presionando **CTRL + barra espaciadora**. En este caso elegimos que el tamaño de la etiqueta se ajuste a su contenido (**wrap_content**, envolver el contenido).

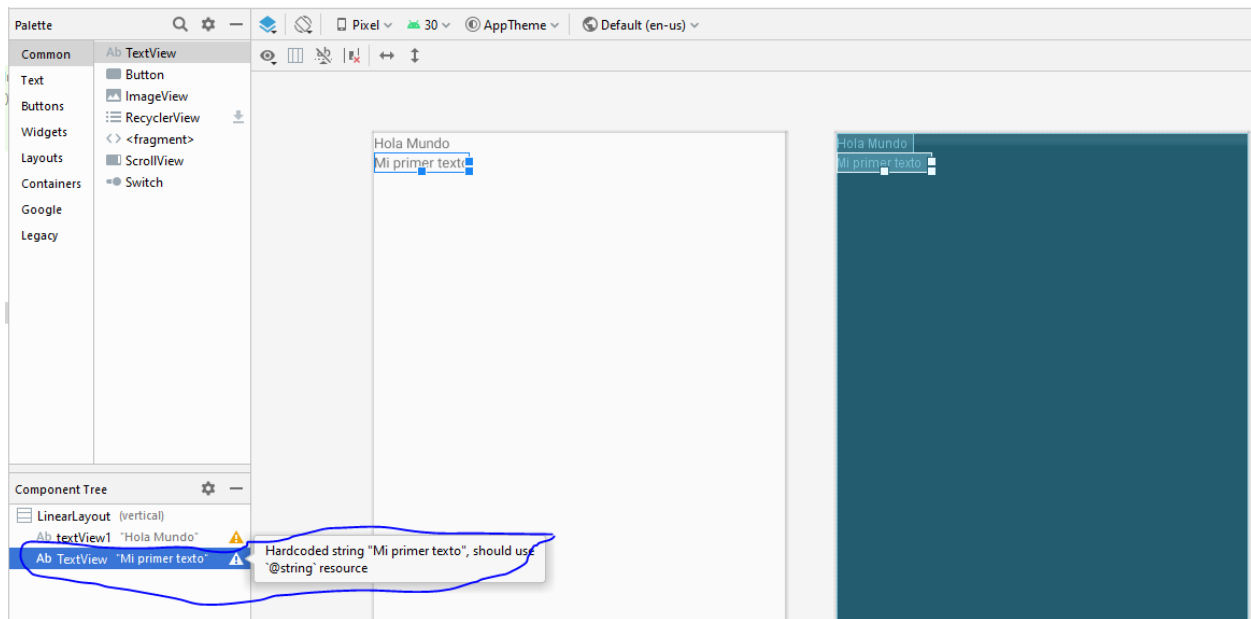


```

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="vertical"
8      tools:context=".UI.U2_U2_atributosBasicos">
9
10     <TextView
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:text="Hola Mundo"/>
14
15     <TextView
16         android:layout_width="wrap_content"
17         android:layout_height="wrap_content"
18         android:text="Mi primer texto"/>
19
20 </LinearLayout>

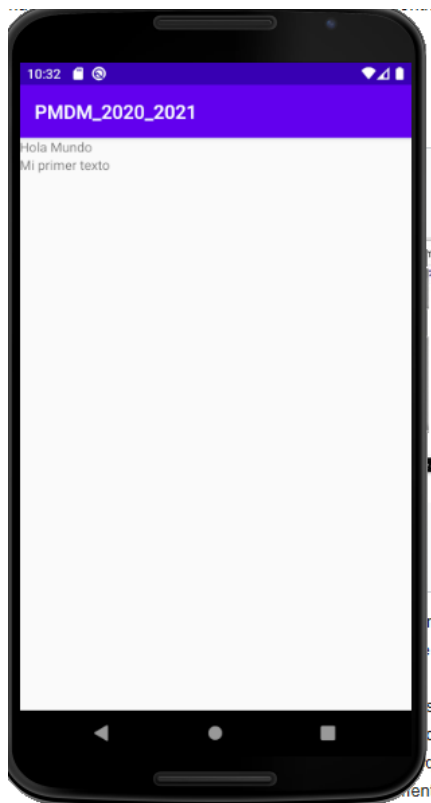
```

Definimos la altura (**android:layout_height**) del elemento, así como su contenido de **android:text**.



Desde la vista 'Design' en el *Component Tree*, veremos unas advertencias (**Hardcoded string "Mi primer texto", should use @string resource**) en los dos componentes `TextView`, dichas advertencias nos indican que es mejor que el contenido del texto se establezca a través de un recurso `@string` y no directamente.

Es decir, lo que se debe hacer siempre, porque permite la internacionalización de la aplicación y la reutilización de recursos, pero en sucesivos ejemplos no se hará, porque el contenido puede ayudarnos a entender qué hace el elemento sin tener que ir a buscar otro recurso para ver cuál es su valor.



La pantalla gráfica muestra el nuevo elemento creado a partir de xml.

- A continuación el código xml asociado al layout:



```

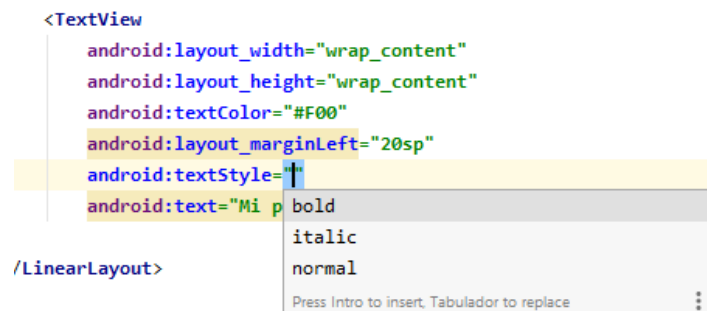
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:orientation="vertical"
8      tools:context=".UI.U2_u2_atributosBasicos">
9
10     <TextView
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:text="Hola Mundo"/>
14
15     <TextView
16         android:layout_width="wrap_content"
17         android:layout_height="wrap_content"
18         android:text="Mi primer texto"/>
19
20 </LinearLayout>

```

1.3.3. Atributos de estilo, color, tamaño,...

- A continuación se muestran otros atributos comunes a la mayoría de los diferentes elementos visuales.

Diseño de pantalla: color, tamaño y estilo en xml



```

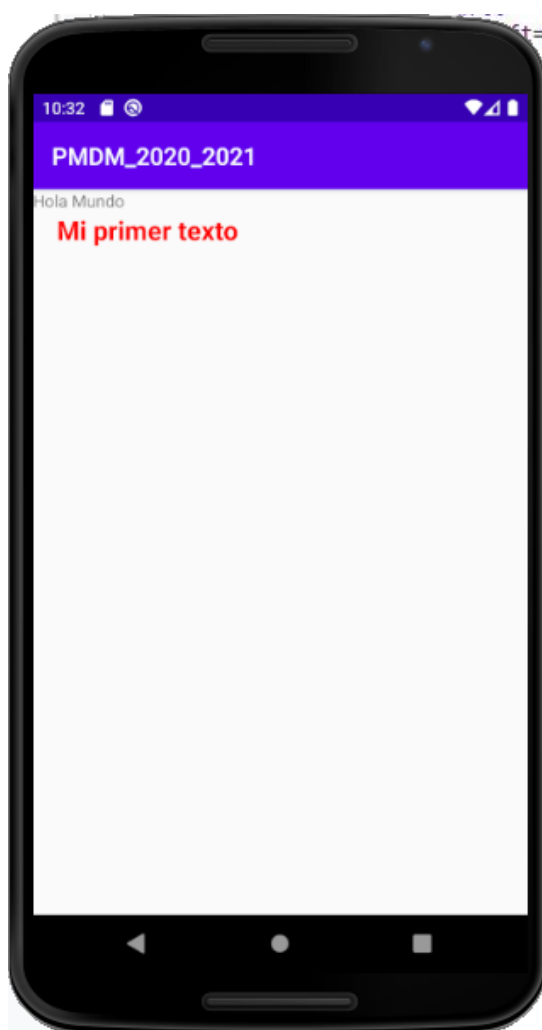
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textColor="#F00"
    android:layout_marginLeft="20sp"
    android:textStyle="bold"
    android:text="Mi p
/LinearLayout>

```

El atributo **textStyle** para negrita, cursiva o normal.

```
<TextView  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Mi primer texto"  
    android:textStyle="bold"  
    android:textSize="22sp"  
    android:textColor="#F00"  
    android:layout_marginLeft="20sp"/>
```

El atributo **textColor** en formato **RGB**. El tamaño (**textSize**) del texto en **sp**. También deja un margen a la izquierda del diseño **layout_marginLeft** de 20 sp, que es lo mismo que 20 dp.



El resultado anterior en modo gráfico.

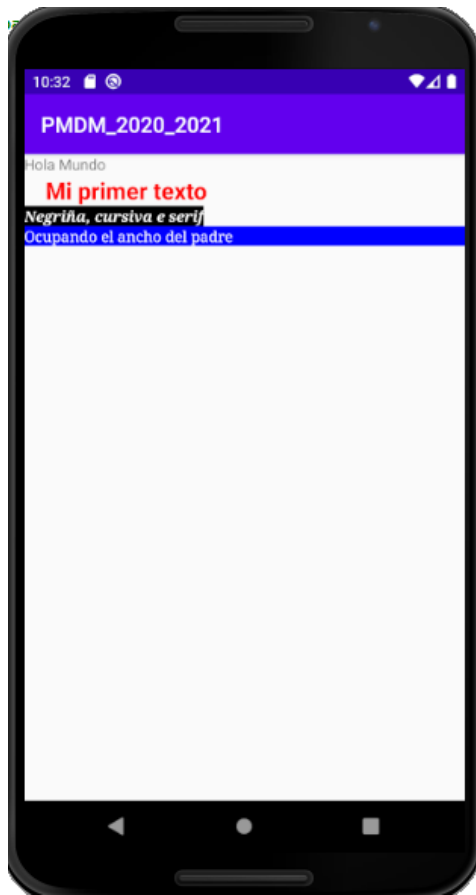
- Agregar 2 <TextView> más: los que están marcados.

```

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="vertical"
8      tools:context=".UI.U2_u2_atributosBasicos">
9      <TextView
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:text="Hola Mundo"/>
13     <TextView
14         android:layout_width="wrap_content"
15         android:layout_height="wrap_content"
16         android:text="Mi primer texto"
17         android:textStyle="bold"
18         android:textSize="22sp"
19         android:textColor="#F00"
20         android:layout_marginLeft="20sp"/>
21     <TextView
22         android:layout_width="wrap_content"
23         android:layout_height="wrap_content"
24         android:text="Negriña, cursiva e serif"
25         android:textStyle="bold|italic"
26         android:textColor="#FFF"
27         android:background="#000"
28         android:typeface="serif"/>
29     <TextView
30         android:layout_width="match_parent"
31         android:layout_height="wrap_content"
32         android:text="Ocupando el ancho del padre"
33         android:textColor="#FFF"
34         android:background="#00F"
35         android:typeface="serif"/>
36 </LinearLayout>

```

- La pantalla muestra estos dos nuevos elementos. Observar el valor de `layout_width = "match_parent"` del último elemento, ya que hace que la etiqueta ocupe todo el ancho del elemento principal (*LinearLayout*).



- El siguiente código agrega un último elemento:

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:text="Ocupando el resto del alto del padre"
    android:background="#F00"
    android:textColor="#FFF"
    android:textStyle="bold"/>
```

- La imagen muestra el resultado. En este caso, preste atención al atributo de **Android: layout_height = "match_parent"**. Observar cómo el último elemento ocupa el resto de la altura del padre (el **LinearLayout**).



- Finalmente introduciremos 2 atributos relacionados con el peso:
 - **android: gravity**, que se usa para colocar el contenido del elemento dentro de él. Por ejemplo, centre el texto dentro de un TextView.
 - **android: layout_gravity**, que se usa para colocar un elemento dentro de un contenedor. Por ejemplo, alinear a la derecha de un Layout un TextView entero.

```

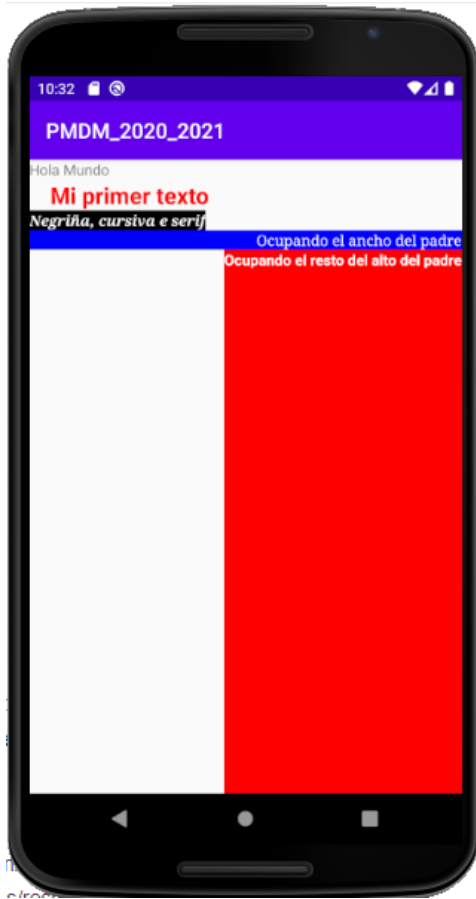
activity_u2_u2_atributos_basicos.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      android:orientation="vertical"
8      tools:context=".UI.U2_U2_atributosBasicos">
9      <TextView
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:text="Hola Mundo"/>
13      <TextView
14         android:layout_width="wrap_content"
15         android:layout_height="wrap_content"
16         android:text="Mi primer texto"
17         android:textStyle="bold"
18         android:textSize="22sp"
19         android:textColor="#F00"
20         android:layout_marginLeft="20sp"/>
21      <TextView
22         android:layout_width="wrap_content"
23         android:layout_height="wrap_content"
24         android:text="Negriña, cursiva e serif"
25         android:textStyle="bold|italic"
26         android:textColor="#FFF"
27         android:background="#000"
28         android:typeface="serif"/>
29      <TextView
30         android:layout_width="match_parent"
31         android:layout_height="wrap_content"
32         android:gravity="right"
33         android:text="Ocupando el ancho del padre"
34         android:textColor="#FFF"
35         android:background="#00F"
36         android:typeface="serif"/>
37      <TextView
38         android:layout_width="wrap_content"
39         android:layout_height="match_parent"
40         android:layout_gravity="right"
41         android:text="Ocupando el resto del alto del padre"
42         android:background="#F00"
43         android:textColor="#FFF"
44         android:textStyle="bold"/>
45  </LinearLayout>
46

```

```

activity_u2_u2_atributos_basicos.xml
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="vertical"
8      tools:context=".UI.U2_U2_atributosBasicos">
9      <TextView
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:text="Hola Mundo"/>
13     <TextView
14         android:layout_width="wrap_content"
15         android:layout_height="wrap_content"
16         android:text="Mi primer texto"
17         android:textStyle="bold"
18         android:textSize="22sp"
19         android:textColor="#F00"
20         android:layout_marginLeft="20sp"/>
21     <TextView
22         android:layout_width="wrap_content"
23         android:layout_height="wrap_content"
24         android:text="Negriña, cursiva e serif"
25         android:textStyle="bold|italic"
26         android:textColor="#FFF"
27         android:background="#000"
28         android:typeface="serif"/>
29     <TextView
30         android:layout_width="match_parent"
31         android:layout_height="wrap_content"
32         android:gravity="right"
33         android:text="Ocupando el ancho del padre"
34         android:textColor="#FFF"
35         android:background="#00F"
36         android:typeface="serif"/>
37     <TextView
38         android:layout_width="wrap_content"
39         android:layout_height="match_parent"
40         android:layout_gravity="right"
41         android:text="Ocupando el resto del alto del padre"
42         android:background="#F00"
43         android:textColor="#FFF"
44         android:textStyle="bold"/>
45 </LinearLayout>
46

```



- Observe la diferencia entre las líneas 32 y 40:
 - Línea 32: El texto está ubicado a la derecha dentro del TextView, no es el TextView el que sufre algunos desplazamientos.
 - Línea 40: Es el TextView entero quien se sitúa dentro del layout que lo contiene.
- Referencias:
 - Tipografías: <http://developer.android.com/design/style/typography.html>
 - Colores: <http://developer.android.com/guide/topics/resources/more-resources.html#Color>