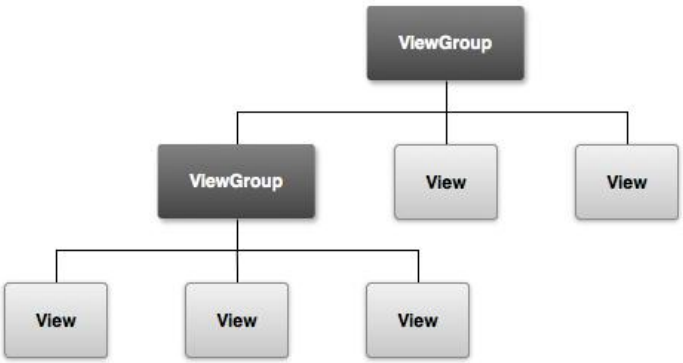
INDICE

1.1. Introducción

- La **interfaz de usuario (UI)** es cualquier objeto que el usuario pueda ver e interactuar.
- Una **vista (View)** 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.

Creación de una UI usando XML

Veamos el ejemplo de un fichero XML. Corresponde al fichero *res/layout/activity_main.xml* de un proyecto donde tenemos un TextView y un botón.

- 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.

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onclick="onButtonClick"
            android:text="I am a Button"/>
/>
</LinearLayout>
```

- 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 .

Creación de una UI usando código

En Java, un objeto de TextView de la siguiente manera:

```java
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

       // Create the text view
      TextView textView = new TextView(this);
      textView.setTextSize(40);
      textView.setText("Hello");

      // Set the text view as the activity layout
      setContentView(textView);
}
```

➢ 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

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/my_text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="I am a TextView" />
    <Button android:id="@+id/my_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="I am a Button"/>
/>
</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.

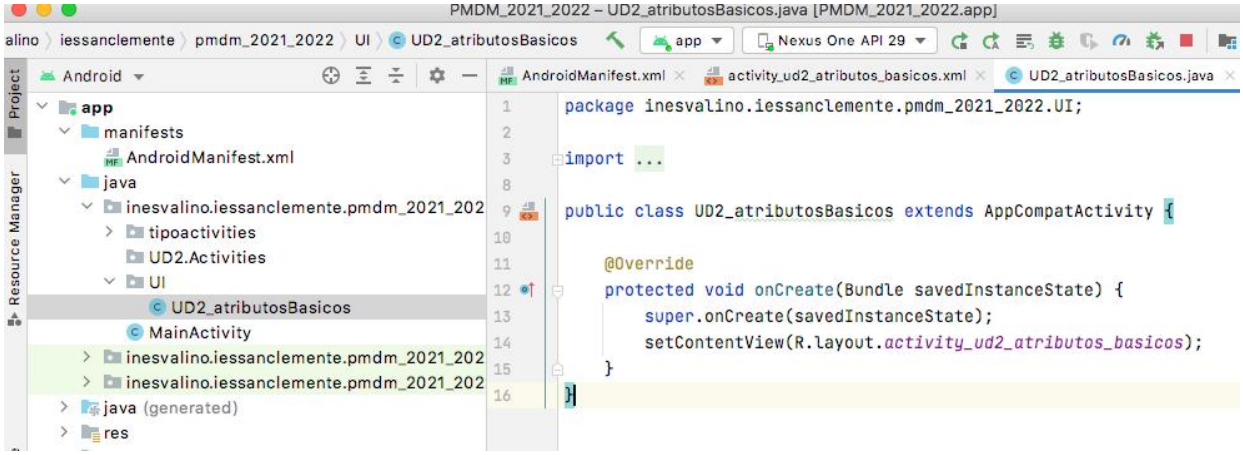
1.3. Ejemplos de atributos de los elementos visuales

1.3.1. Diseño pantalla en modo gráfico

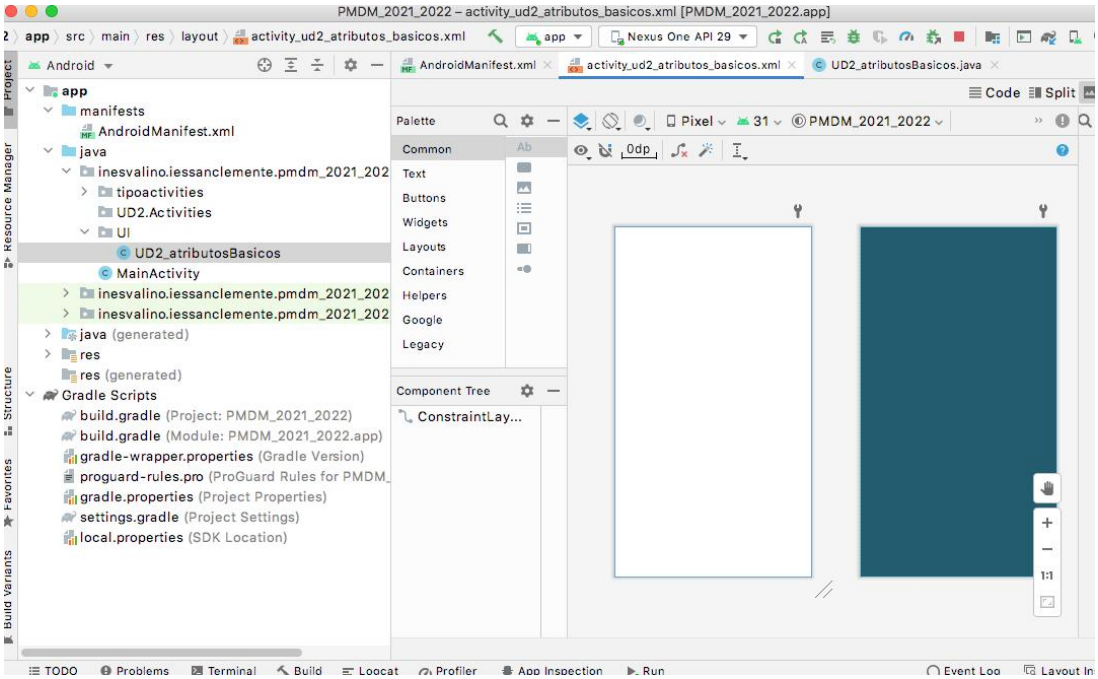
- Partiremos del proyecto anterior.
- Comenzaremos creando un nuevo paquete llamado: **UI**.
- Dentro de este paquete cree una nueva actividad llamada: **UD2_atributosBasicos**
- Recordad configurar esta actividad como Launcher Activity.
- Esta vez usaremos inicialmente un diseño llamado **ConstraintLayout** 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

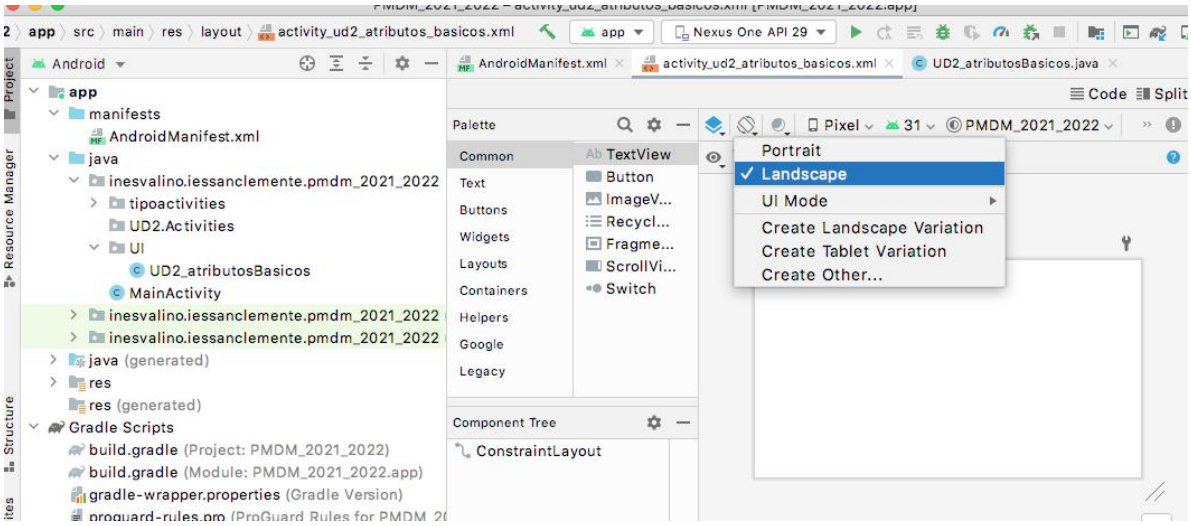
Creamos el paquete y la actividad



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



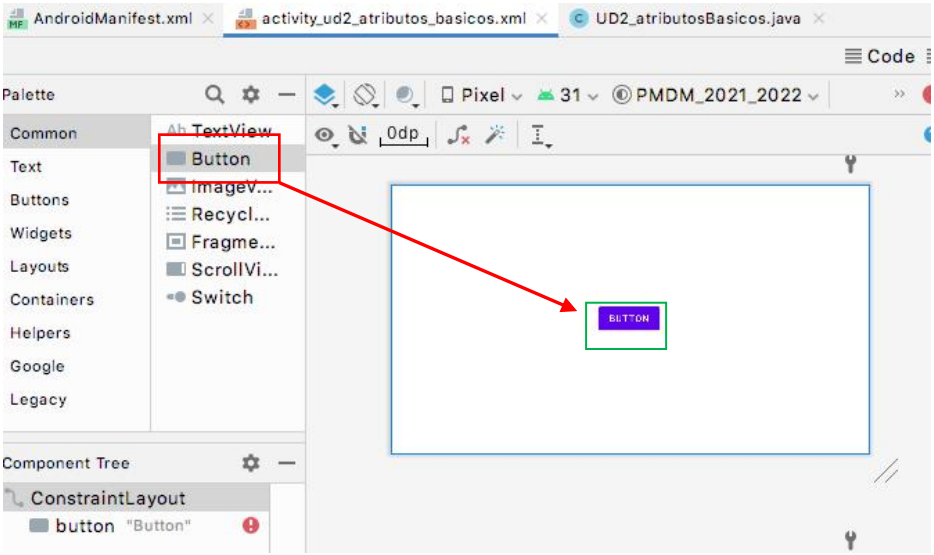
El cual podemos manipular, por ejemplo para poner en apaisado



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

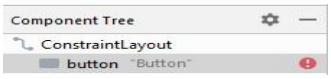
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".UI.UD2_atributosBasicos">
8
9  </androidx.constraintlayout.widget.ConstraintLayout>
```

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…

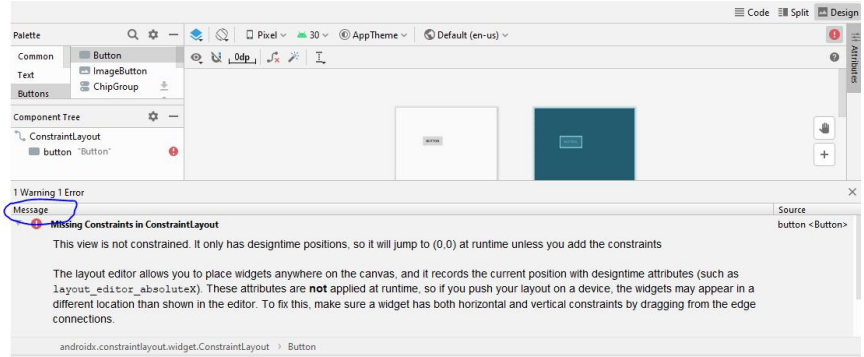


- El archivo xml se vería así: Las líneas correspondientes al botón son las marcadas:

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:



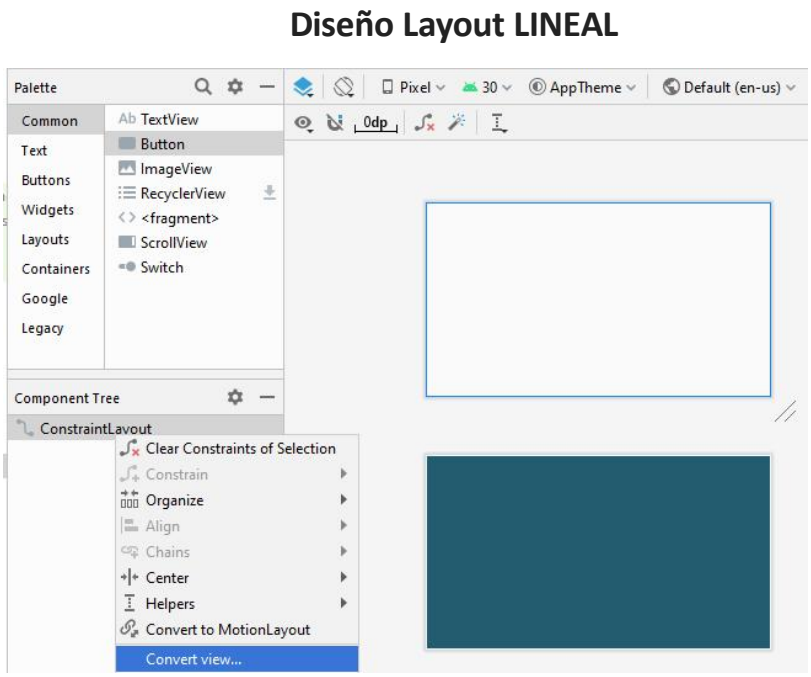
En el caso de que no se añadan las restricciones, el control, en este caso, el botón se colocará en la posición (0,0). Se pueden añadir manualmente o existe una **varita mágica** que lo hace por nosotros añadiendo las restricciones necesarias.

De esta forma quedaría el código:

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http
    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.UD2_atributosBasicos">

    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Como veremos posteriormente cada layout tendrá unos parámetros específicos.

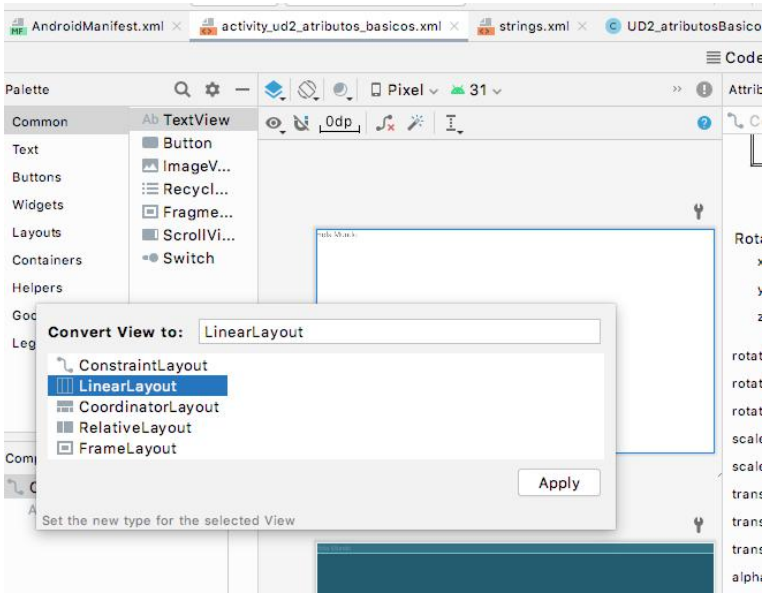
1.3.2. Diseño de pantalla en xml



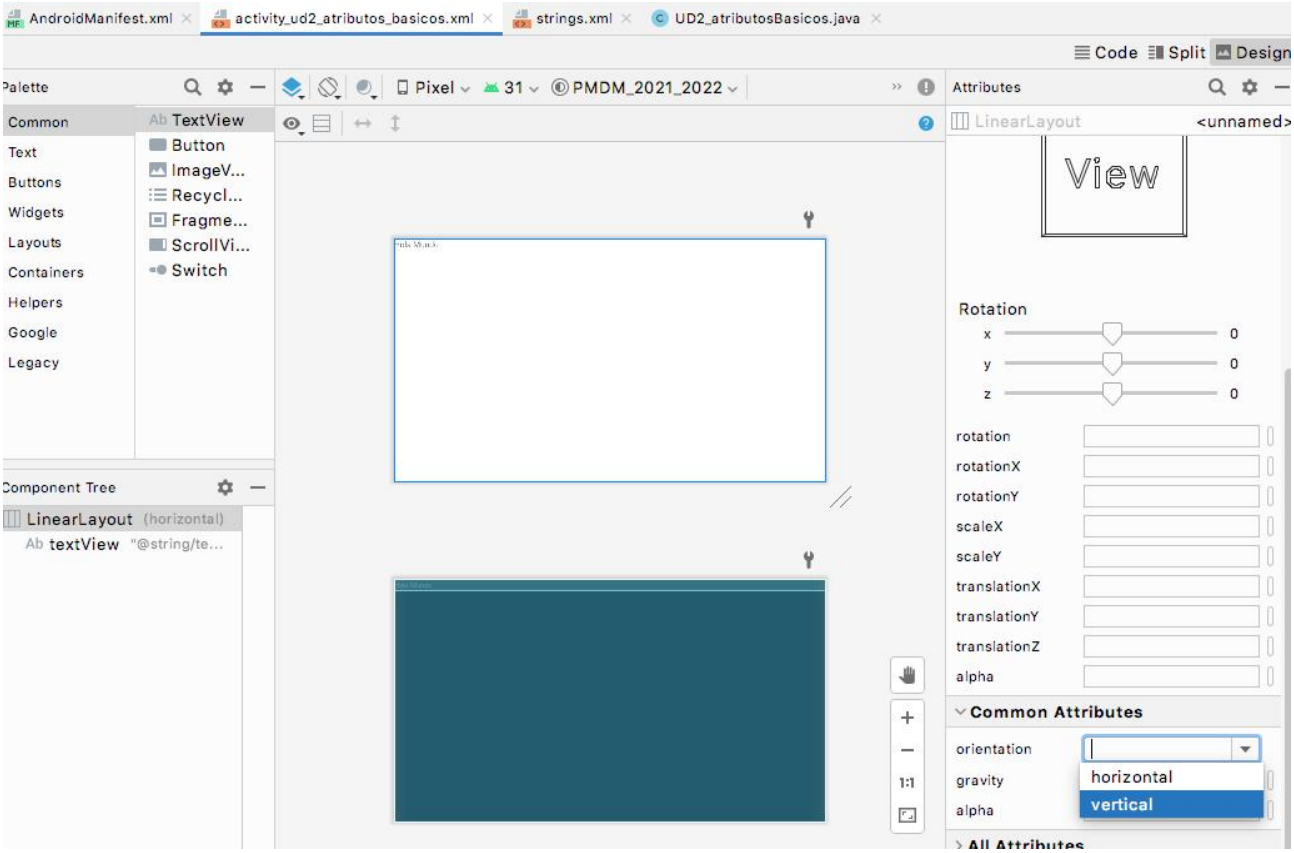
Eliminamos el botón agregado y agregamos algo de texto si aún no lo trae por defecto. Cambiamos el tipo de diseño. Para hacer esto, haga clic con el botón derecho en *'ConstraintLayout'* y elegir la opción **Convert View**.

Para hacer un diseño sencillo cambiaremos el tipo de layout a LinearLayout (lineal).

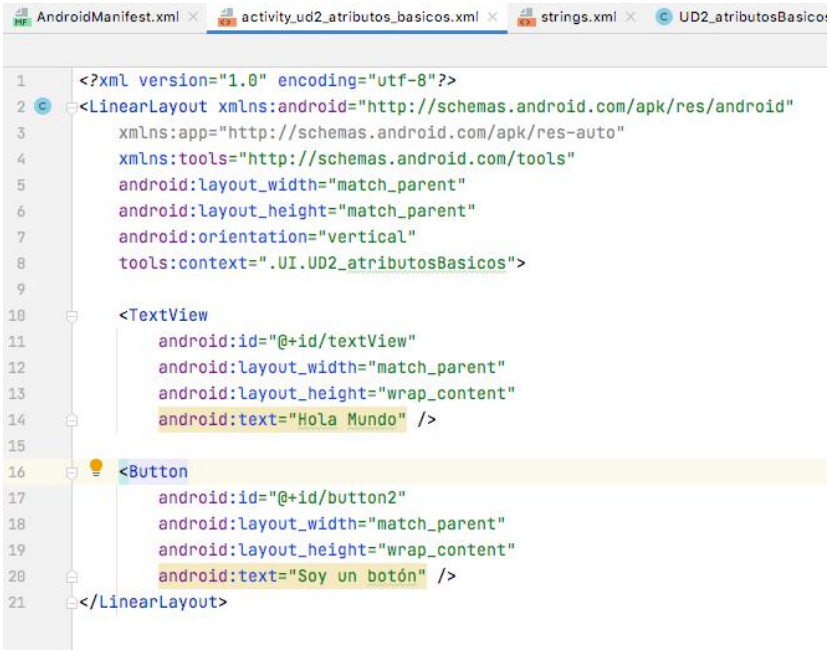
Y agregaremos el botón nuevamente.



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

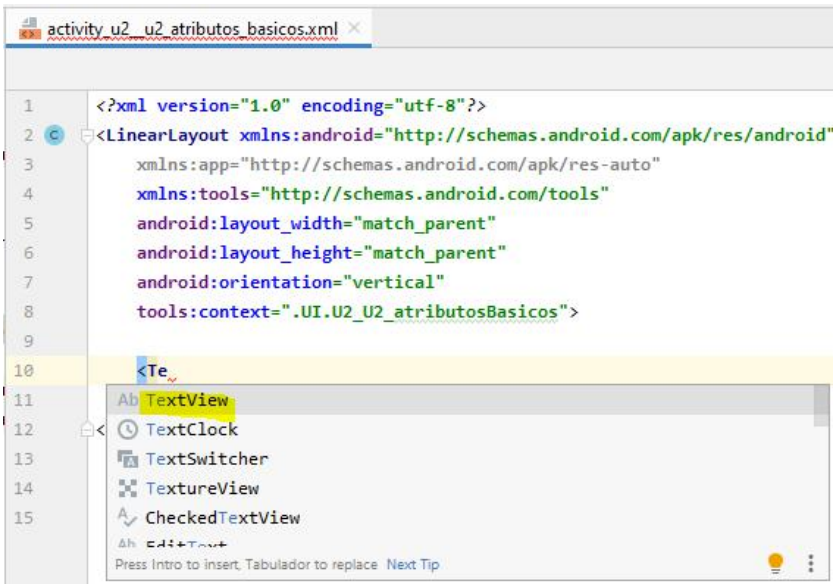


Podéis volver a añadir un botón si queréis, para ver cómo se vería.



- 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

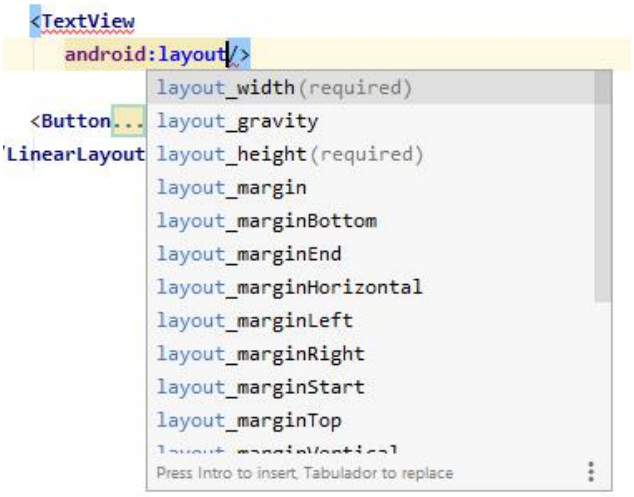


Eliminamos los controles anteriores (TextView Button) y comenzamos creando un elemento visual y si presionamos **CTRL + BARRA ESPACIADORA** el IDE ofrece los posibles elementos que comienzan con ese nombre. En algunas versiones en el momento de empezar a escribir ya aparece el listado de sugerencias.

activity_u2__u2_atributos_basicos.xml

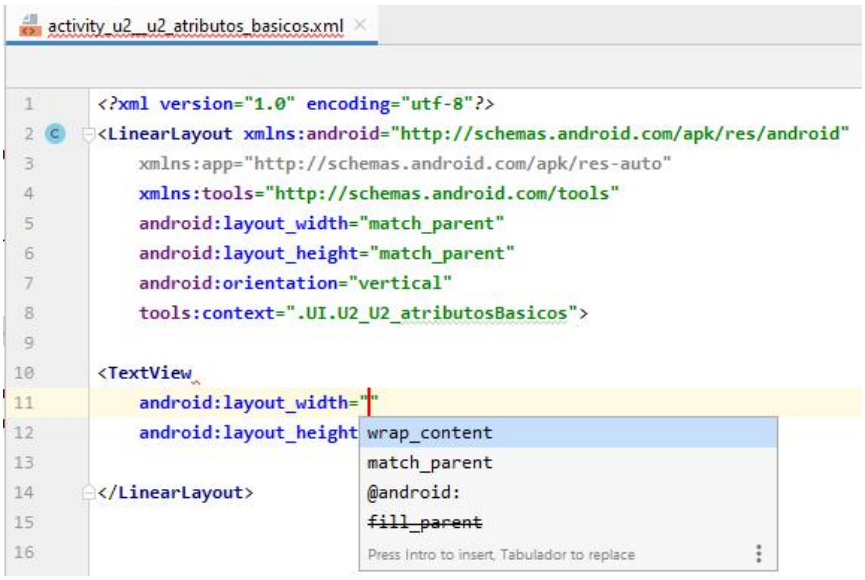
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".UI.U2_U2_atributosBasicos">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hola Mundo"/>

</LinearLayout>
```

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

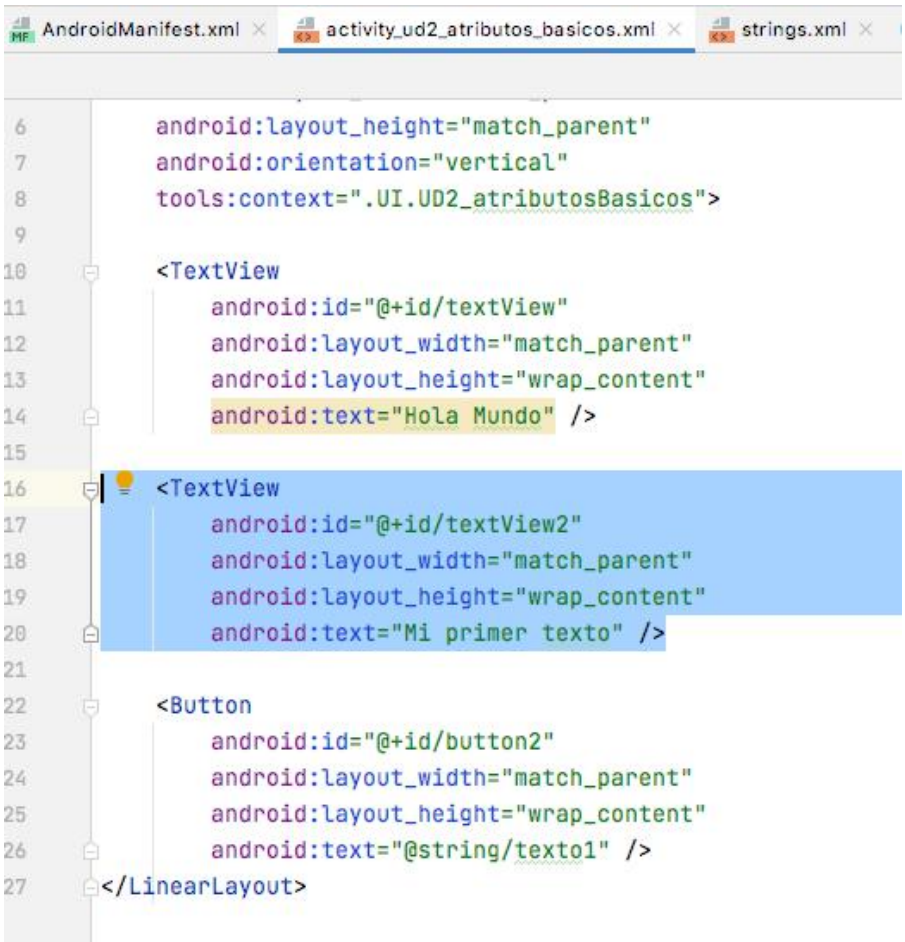


Empezamos indicando el ancho... Escribimos android: **CTRL + barra espaciadora**, o simplemente presionamos **CTRL + barra espaciadora** (en algunas versiones ya basta con simplemente empezar a escribir las primeras letras del atributo) y nos ofrece los posibles atributos. Cuantas más letras escribamos 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.



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).



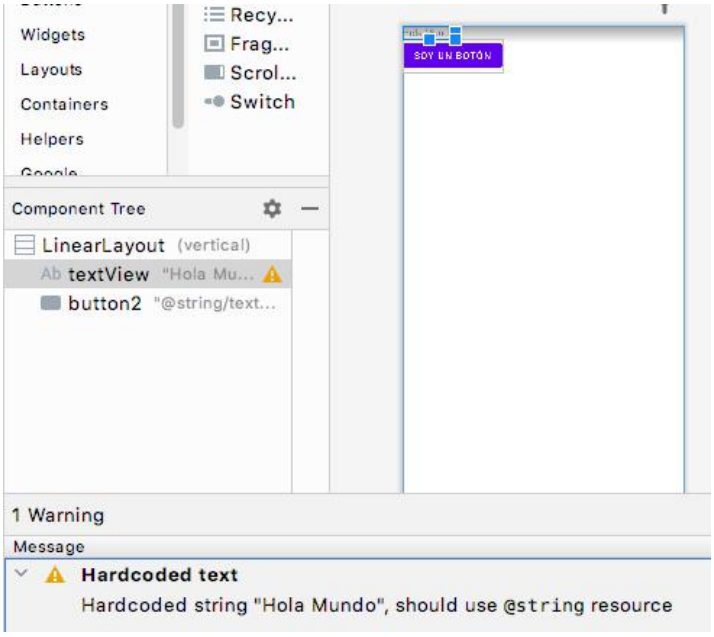
Definimos la altura (android**: layout_height**) del elemento que se adapte al **match_parent**, así como su contenido de **android: text**, y el **android:id**.

La clase base LayoutParams describe qué tan grande se quiere la vista tanto en altura como en anchura. Para cada dimensión se puede elegir una de las siguiente opciones:

- match_parent: la vista será tan grande como su "padre" (menos el *padding*, el relleno/espacio que deja el contenido de una vista respecto sus lados), es decir, se expanderá lo más posible dentro de la vista principal.
- wrap_content: el tamaño de la vista se ajusta al contenido
- también se le podría dar un número exacto.

Nota: Los parámetros de un layout se verán en el siguiente tema de esta unidad.

Desde la vista '*Design'* en el *Component Tree*, veremos unas advertencias (**Hardcoded string "Hola Mundo", should use @string resource**) en los dos componentes TextView y Button (en el ejemplo superior el *warning* está aplicado para el botón), dichas advertencias nos indican que es mejor que el contenido del texto se establezca a través de un recurso **@string** y no directamente.

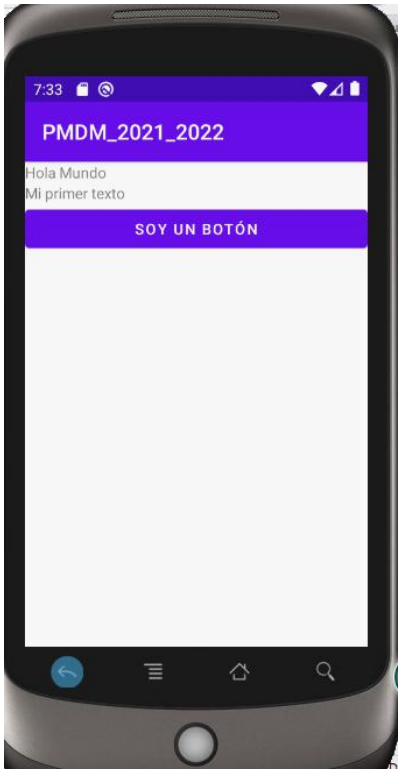


En concreto se trataría de añadir en res/values/strings.xml como por ejemplo he hecho para el botón:

```
<resources>
    <string name="app_name">PMDM_2021_2022</string>
    <string name="texto1">Soy un botón</string>
</resources>
```

Y luego en el campo "text" de los atributos del botón escribir: android:text="@string/texto1"

Esto se debe hacer siempre, porque permite la internacionalización de la aplicación y la reutilización de recursos. Sin embargo, en los ejemplos que veremos a continuación no se hará por simplicidad, solo con el fin de que el contenido pueda ayudarnos a entender qué hace el elemento sin tengamos que ir a buscar elrecurso 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:

```
<?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=".UI.UD2_atributosBasicos">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
```

```
        android:text="Hola Mundo" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Mi primer texto" />

    <Button
        android:id="@+id/button2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/texto1" />
</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

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

En este ejemplo:

```
<TextView
    android:id="@+id/textView2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Mi primer texto"
    android:textStyle="bold"
    android:textSize="22sp"
```

```
android:textColor="#FF0000"
android:layout_marginLeft="20sp"/>
```

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

➢ Eliminad el botón y agregad 2 <TextView> más, como los que están marcados en la imagen inferior.

```
        android:layout_height="wrap_content"
        android:text="Mi primer texto"
        android:textStyle="bold"
        android:textSize="22sp"
        android:textColor="#FF00"
        android:layout_marginLeft="20sp"/>

    <TextView
        android:id="@+id/textView3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Negrita, cursiva y Serif"
        android:textStyle="bold|italic"
        android:textColor="#FFF"
        android:background="#000"
        android:typeface="serif"
        android:layout_marginLeft="20sp"/>

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="otro ajustado al contenido"
        android:textStyle="bold|italic"
        android:textColor="#FFF"
        android:background="#00F"
        android:typeface="serif"
        android:layout_marginLeft="20sp"/>
```

- La pantalla muestra estos dos nuevos elementos. Observad que el valor de **layout_width = "match_parent"** del penúltimo elemento hace que la etiqueta ocupe todo el ancho del elemento principal (*LinearLayout*) menos el margen, mientras que el **layout_width = "wrap_content"** del último hace que la etiqueta se ajuste al propio contenido.

- El siguiente código agrega un último elemento:

```
<TextView
    android:id="@+id/textView5"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:text="ocupando el resto del padre"
    android:textStyle="bold"
    android:textColor="#FFF"
    android:background="#FF00"/>
```

- 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.

```
<?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=".UI.UD2_atributosBasicos">

    <TextView
        android:id="@+id/textView"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Hola Mundo" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Mi primer texto"
        android:textStyle="bold"
        android:textSize="22sp"
        android:textColor="#FF00"
        android:layout_marginLeft="20sp"/>

    <TextView
        android:id="@+id/textView3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="right"
        android:text="Negrita, cursiva y Serif"
        android:textStyle="bold|italic"
        android:textColor="#FFF"
        android:background="#000"
        android:typeface="serif"
        android:layout_marginLeft="20sp"/>

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="otro ajustado al contenido"
        android:textStyle="bold|italic"
```

```
        android:textColor="#FFF"
        android:background="#00F"
        android:typeface="serif"
        android:layout_marginLeft="20sp"/>

    <TextView
        android:id="@+id/textView5"
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="right"
        android:text="ocupando el resto del padre"
        android:textStyle="bold"
        android:textColor="#FFF"
        android:background="#FF00"/>

</LinearLayout>
```

- Observe las diferencias:
  - android:gravity="right": el texto está ubicado a la derecha dentro del TextView, no es el TextView el que sufre algunos desplazamientos.
  - android:layout_gravity="right": 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


8:29
PMDM_2021_2022
Hola Mundo
Mi primer texto
Negrita, cursiva y Serif
otro ajustado al contenido
ocupando el resto del padre