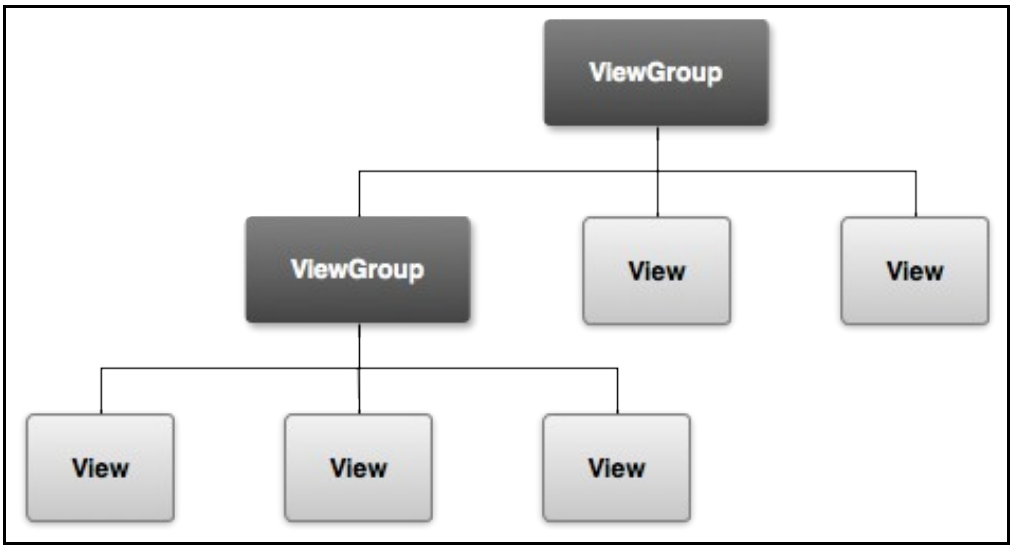
Interface gráfica (UI - User Interface). Vistas (Views). Atributos básicos.

Sumario

Introducción

- A **Interface de Usuario (UI)** é calquera obxecto que o usuario pode ver e interactuar con el.
- Unha **Vista (View)** é un obxecto que debuxa algo na pantalla (botón, etiqueta, radio, casilla de verificación, etc). O usuario pode interactuar con ese obxecto.
- Un **ViewGroup** é unha vista especial invisible que pode conter Vistas e outros ViewGroups. Serve para organizar a posición das Vistas/ViewGroups na pantalla.



- Tanto as vistas como os ViewGroups poden ser creados como obxectos en Java ou facendo uso dos ficheiros XML.

- Exemplo dun ficheiro 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>
```

- Nas liñas marcadas temos o comezo de definición de cada un dos **elementos** que compoñen a pantalla.
- <LinearLayout>, <TextView> e <Button> son Views (Vistas) que definen ese tipo de obxectos visuais.
- Para cada **<elemento>** dun ficheiro XML temos a unha clase Java asociada que permite manipular ese elemento ou crear un novo en tempo de execución.

- O elemento **<TextView>** de XML ten asociada a clase Java **TextView**. Un TextView equivale a unha etiqueta (label) noutras linguaxes de programación.
- O elemento **<LinearLayout>** XML crea o ViewGroup en Java **LinearLayout**. Un LinearLayout é unha extensión dun ViewGroup.

- En Java crearíase un obxecto TextView da seguinte maneira:

```
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);
}
```

- As vantaxes de usar XML:
  - Permite separar as capas de presentación (UI) da de programación.
  - As modificacións da UI poden ser realizadas sen tocar o código Java

- Como xa se viu no apartado anterior:
  - Os ficheiros XML decláranse dentro de **/res**
  - Os recursos, neste caso XML, son accedidos dende o código fonte Java a través da clase Java R.

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

Atributos dos elementos visuais

```
<?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:onclick="onButtonClick"
            android:text="I am a Button"/>
/>
</LinearLayout>
```

- **xmlns:** indica cal vai ser o espacio de nomes usado para definir os atributos xml: http://es.wikipedia.org/wiki/Espacio_de_nombres_XML

- **ID**: os elementos poden ter asignado un número enteiro, que é creado en compilación (Clase Java R). Por ese ID é polo que se accede tanto en Java como noutras referencias en XML a ese elemento.
- **android:id**. ID do elemento ou control. Fórmase da seguinte maneira **"@+id/cadea de texto**:

- @: indica que o que vén a continuación é un recurso.
- **+**: indica que o ID non existe e que o cree (lembrar na **clase Java R**)
- **tipo recurso**: neste caso **id**, pero podería ser: string, drawable, layout, etc, como xa vimos no apartado anterior.
- **cadea de texto**: é o nome que se dá ao identificador.
- Exemplo: **android:id="@+id/my_button"**
- Para facer referencia a ese recurso dende calquera outro recurso é sen o "+": **android:id="@id/my_button"**.

- **android:text**. Texto do elemento. Pódese especificar:
  - **directamente**: android:text="I am a Button"
  - **a través dun recurso**: android:text="@string/button_text". Neste caso **button_text** estará definido noutro ficheiro XML, por exemplo strings.xml.

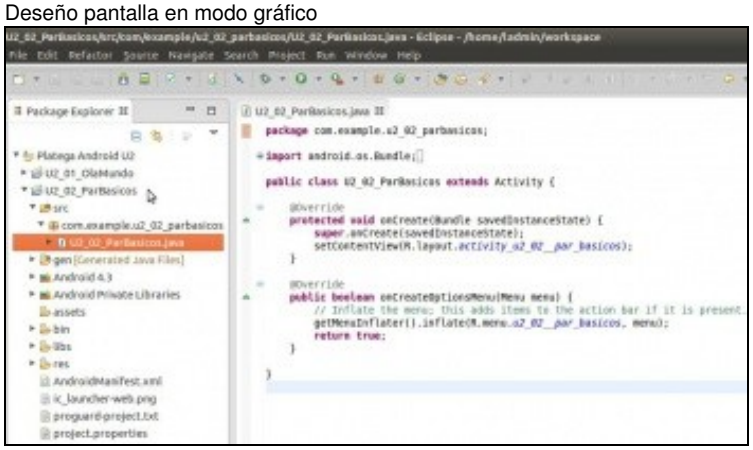
- **android:layout_height** e **android:layout_width**. Especifican as dimensións do elemento con respecto ao Layout que as contén ou ao contido do elemento. Pode tomar os seguintes valores:
  - **valor fixo**: en dp
  - **"match_parent"**: o alto ou ancho do elemento axustarase ao alto ou ancho do elemento que o contén.
  - **"wrap_content"**: o alto ou ancho do elemento axustarase ao alto ou ancho do contido do mesmo elemento.

- Hai máis atributos comúns aos elementos visuais. Imos a continuación amosar máis con exemplos.

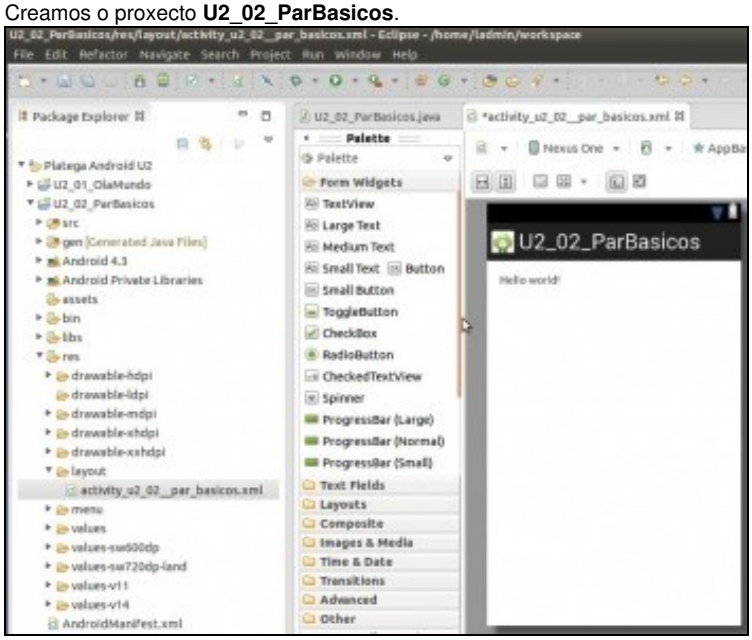
Exemplos de atributos dos elementos visuais

Deseño pantalla en modo gráfico

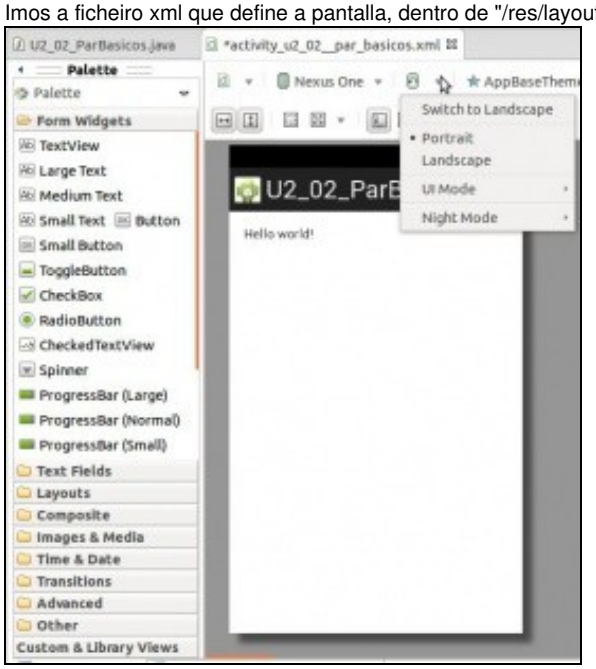
- Comezaremos creando un novo proxecto: **U2_02_ParBasicos**.
- Nesta ocasión vaise usar inicialmente un Layout chamado **RelativeLayout** para dispor os elementos visuais dentro da pantalla, logo cambiaremos a un layout chamado **LinearLayout**. Os Layouts veranse no seguinte apartado.

- Deseño pantalla en modo gráfico
- 

Creamos o proxecto **U2_02_ParBasicos**.

- 

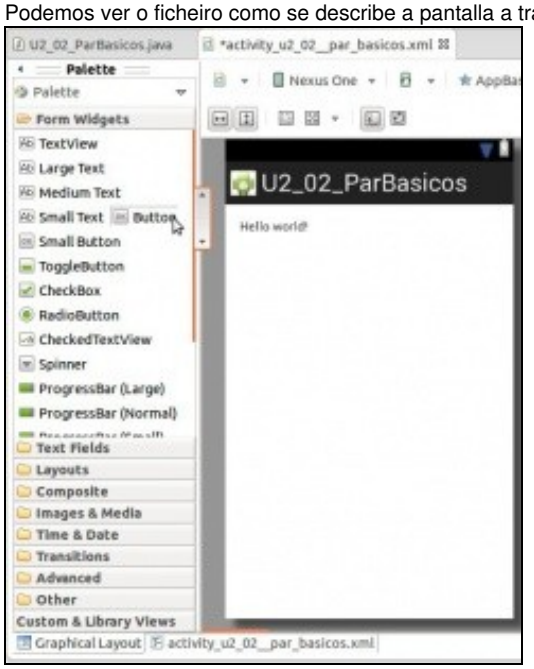
Imos a ficheiro xml que define a pantalla, dentro de "/res/layout". Preséntase nunha pantalla que simula un dispositivo.

- 

A cal podemos manipular, por exemplo para poñer en apaisado.

- 

Podemos ver o ficheiro como se describe a pantalla a través de elementos/controis XML.

- 

Podemos definir a pantalla de modo gráfico ou en xml. En calquera caso o realizado nun dos casos ten a súa correspondente tradución ao outro. Por exemplo se se arrastra un botón en modo gráfico á pantalla.

- 

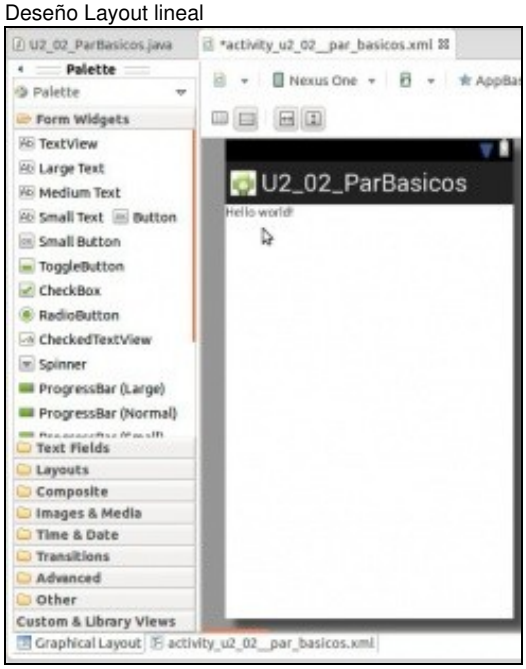
E o situamos onde desexamos ..

- O ficheiro xml quedaría como segue: As liñas correspondentes ao botón son as que está marcadas:

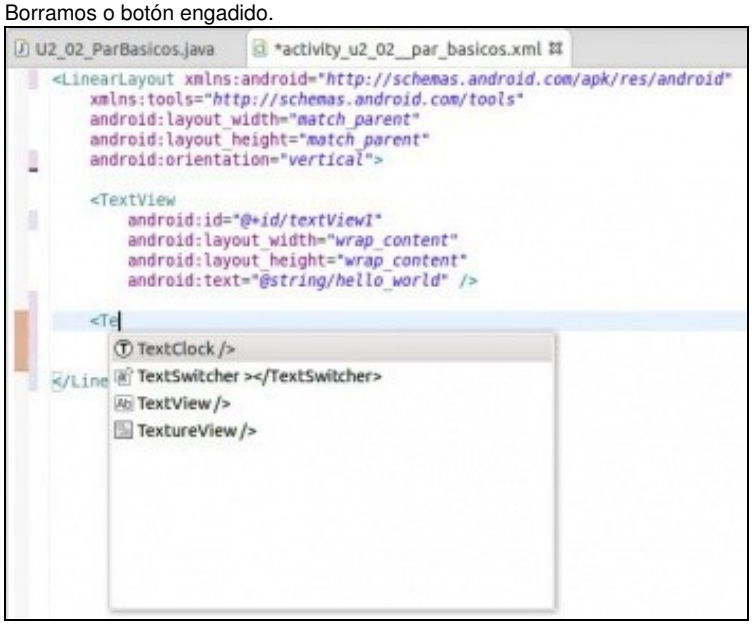
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".U2_02_ParBasicos" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/textView1"
        android:layout_marginTop="99dp"
        android:layout_toRightOf="@id/textView1"
        android:text="Button" />

</RelativeLayout>
```

- Observar como <TextView> se crea un ID para o elemento que é usado en <Button> para indicar que este elemento horizontalmente comeza á dereita do <TextView>.
- Observar como se indica, neste caso, a posición vertical: "layout_marginTop"

Deseño pantalla en xml

- Deseño Layout lineal
- 

Borramos o botón engadido.

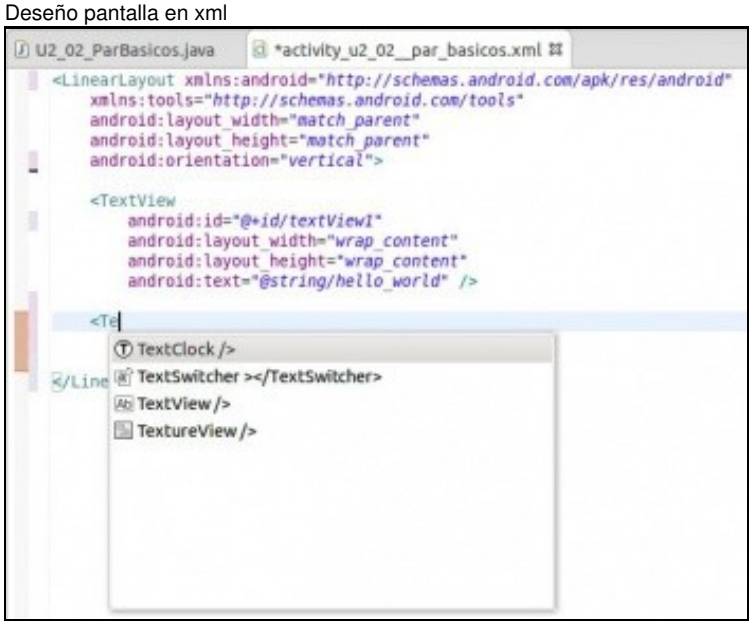
- 

para facer un deseño sinxelo imos cambiar o tipo de layout de Relativo a Lineal (Estes estudaranse máis adiante) e borrar as opcións de recheo (**padding**) do layout

```
<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">


    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</LinearLayout>
```

- Observar como se cambian as liñas marcadas. No caso da liña 5 indicamos ao layout lineal que os elementos que conteña os dispoña en modo vertical, un a continuación do outro.

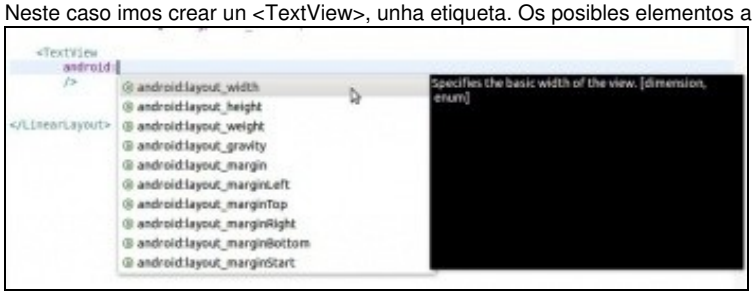
- Deseño pantalla en xml

- 

Comezamos a crear un elemento visual e se prememos **CTRL+barra espaciadora** o IDE ofrece os posibles elementos que comezan por ese nome.

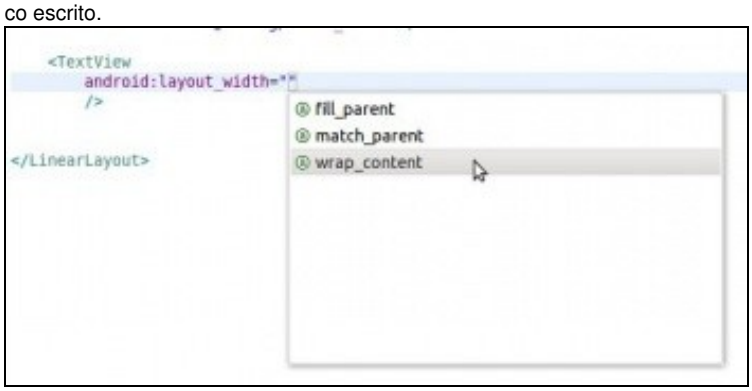
- 

Neste caso imos crear un <TextView>, unha etiqueta. Os posibles elementos a crear veranse nun apartado posterior.

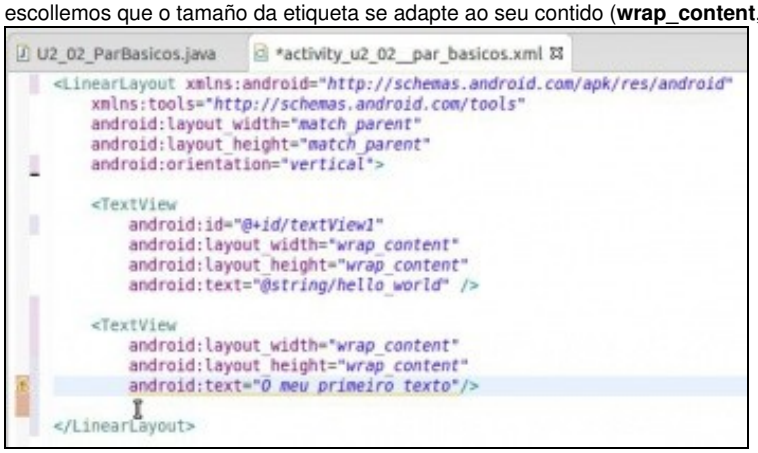
- 

Comezamos indicando o ancho... Tecleamos android:**CTRL+barra espaciadora**, ou simplemente prememos **CTRL+barra espaciadora** e ofrécenos os posibles atributos. Cantas máis letras escribamos nós do atributo máis se afina a busca. Neste caso seleccionamos **android:layout_width**

Tamén podemos escribir as letras dun atributo (sen android:) e premer **CTRL+barra espaciadora** e xa nos ofrece os atributos que coinciden co escrito.

- 

Se dentro das "" hai xa posibles valores predefinidos para ese atributo ímolo saber premendo **CTRL+barra espaciadora**. Neste caso escollemos que o tamaño da etiqueta se adapte ao seu contido (**wrap_content**, envolver o contido).

- 

Definimos o alto (**android:layout_height**) do elemento, así como o seu contido **android:text**.

- 

Cando salvemos os cambios, vai aparecer unha advertencia que nos indica que é mellor que o contido de texto estea definido a través dun recurso **@string** e non directamente.

Isto é, o que hai que facer sempre, porque así permite a internacionalización da aplicación e o reuso de recursos, pero en exemplos sucesivos non se vai facer, porque así o contido pode axudarnos a entender que fai o elemento sen ter que ir mirar outro recurso para ver cal é o seu valor.

- 

A pantalla gráfica amosa o novo elemento creado dende xml.

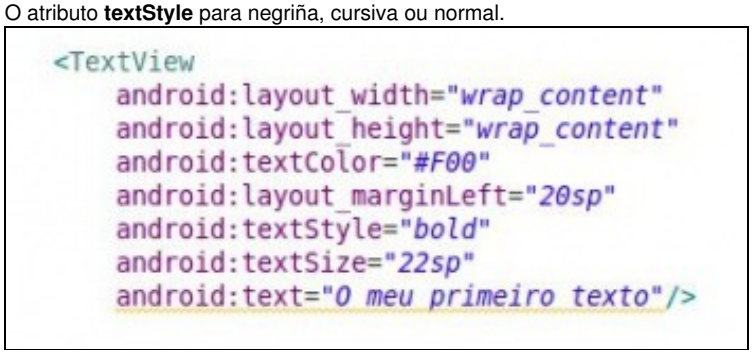
- A continuación o código xml asociado ao layout:

```
<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">


    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="O meu primeiro texto"/>

</LinearLayout>
```

Atributos de estilo, cor, tamaño, ...

- A continuación vanse amosar outros atributos comúns á maioría dos distintos elementos visuais.

- Deseño pantalla: cor, tamaño e estilo en xml
- 

O atributo **textStyle** para negriña, cursiva ou normal.
- 

O atributo **textColor** en formato **RGB** (Ver ao final as referencias). O tamaño (**textSize**) do texto en **sp**. Tamén se deixa unha marxe á esquerda do layout **layout_marginLeft** de 20 sp, que é o mesmo que 20 dp.
- 

O resultado anterior en modo gráfico.

- A continuación engadir 2 <TextView>s máis: Os que están marcados.

```
<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">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#F00"
        android:layout_marginLeft="20sp"
        android:textStyle="bold"
        android:textSize="22sp"
        android:text="O meu primeiro texto"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#000"
        android:textColor="#FFF"
        android:textStyle="bold|italic"
        android:typeface="serif"
        android:text="Negriña, cursiva e serif"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#00F"
        android:textColor="#FFF"
        android:text="Ocupando o ancho do pai"/>

</LinearLayout>
```

- A pantalla amosa eses dous novos elementos. Observar o valor de **layout_width="match_parent"** do último elemento, como fai que a etiqueta ocupe todo o ancho do elemento pai (o LinearLayout).

- O seguinte código engade un último elemento:

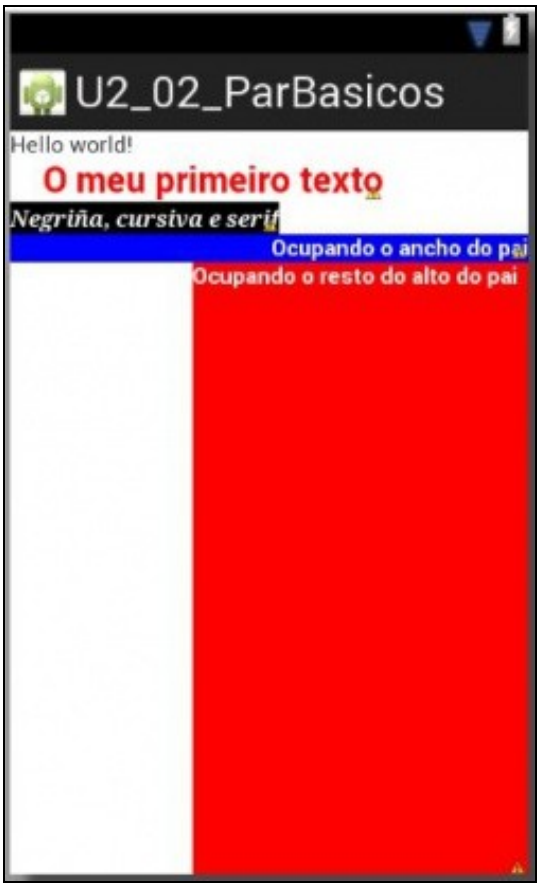
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#F00"
        android:layout_marginLeft="20sp"
        android:textStyle="bold"
        android:textSize="22sp"
        android:text="O meu primeiro texto"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#000"
        android:textColor="#FFF"
        android:textStyle="bold|italic"
        android:typeface="serif"
        android:text="Negriña, cursiva e serif"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#00F"
        android:textColor="#FFF"
```

```
        android:textStyle="bold"
        android:text="Ocupando o ancho do pai"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:background="#F00"
        android:textColor="#FFF"
        android:textStyle="bold"
        android:text="Ocupando o resto do alto do pai  "/>

</LinearLayout>
```

- A imaxe amosa o resultado. Neste caso prestar atención ao atributo **android:layout_height="match_parent"**. Observar como o último elemento ocupa o que resta do alto do pai (o LinearLayout)

- Finalmente imos introducir 2 atributos relacionados co peso:
    - **android:gravity**, que serve para situar o contido do elemento dentro deste. Por exemplo, centrar o texto dentro dun TextView.
    - **android:layout_gravity**, que serve para situar un elemento dentro dun contedor. Por exemplo, aliñar á dereita dun Layout un TextView enteiro.

```
<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">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="#F00"
        android:layout_marginLeft="20sp"
        android:textStyle="bold"
        android:textSize="22sp"
        android:text="O meu primeiro texto"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:background="#000"
        android:textColor="#FFF"
        android:textStyle="bold|italic"
        android:typeface="serif"
        android:text="Negriña, cursiva e serif"/>

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="right"
        android:background="#00F"
        android:textColor="#FFF"
        android:textStyle="bold"
        android:text="Ocupando o ancho do pai"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:layout_gravity="right"
        android:background="#F00"
        android:textColor="#FFF"
        android:textStyle="bold"
        android:text="Ocupando o resto do alto do pai  "/>

</LinearLayout>
```



- Observar a diferenza entre as liñas 34 e 43:
  - Liña 34: O texto sitúase á dereita dentro do TextView, non é o TextView que sofre algún desprazamento.
  - Liña 43: É o TextView enteiro quen se sitúa á dereita dentro do layout que o contén.

- Referencias:
  - Tipografías: http://developer.android.com/design/style/typography.html
  - Cores: http://developer.android.com/guide/topics/resources/more-resources.html#Color

-- Ángel D. Fernández González e Carlos Carrión Álvarez -- (2015).