DISEÑO DE LA INTERFAZ DE USUARIO

Págs 42 y sgts. del Manual de SGOliver

1. DISEÑAR LA INTERFAZ DE USUARIO MEDIANTE VISTAS

- Android permite desarrollar interfaces de usuario usando archivos de diseño XML.

- La interfaz de usuario de Android se compone de **vistas** (Views). Una vista es un objeto como un botón, una imagen, una etiqueta de texto, etc. Cada uno de estos objetos se hereda de la clase principal **View**.

- Los **Layout** son elementos no visibles que establecen cómo se distribuyen en la interfaz del usuario los componentes (*vistas*) que incluyamos en su interior. Son como paneles donde vamos incorporando, de forma diseñada, los componentes con los que interacciona el usuario. Un Layout deriva de la clase **ViewGroup**.

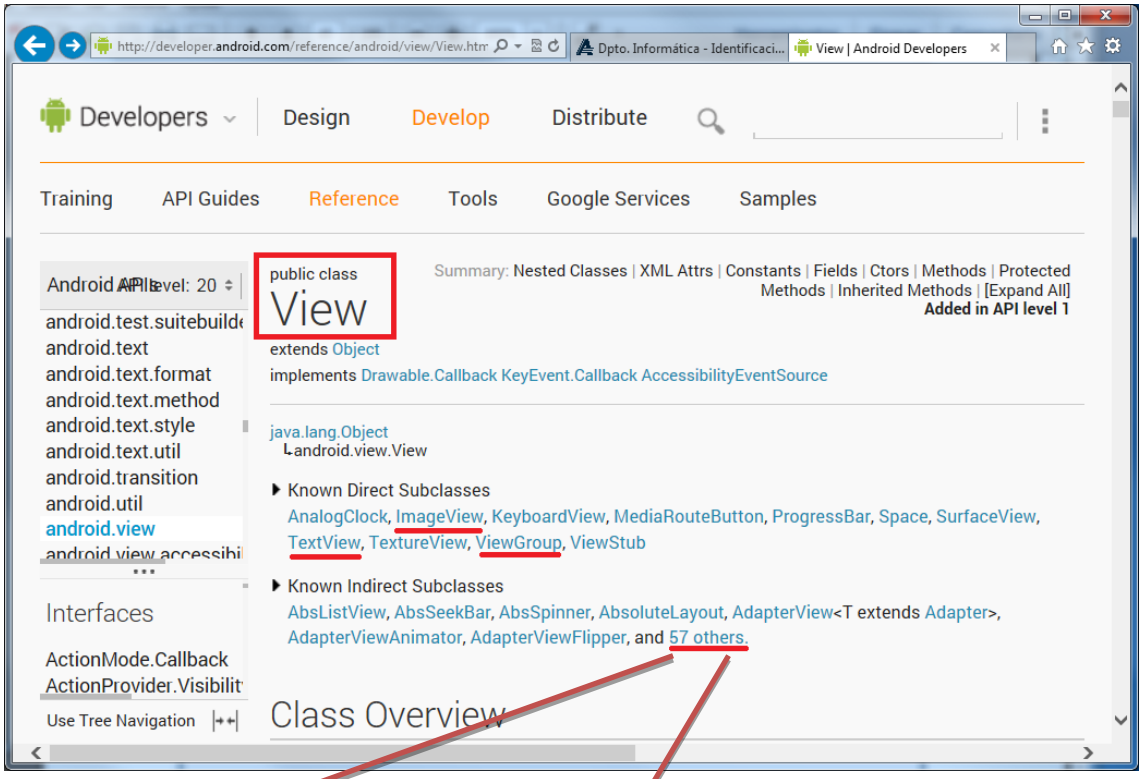
- En resumen: las Vistas visibles deben situarse dentro de otro tipo de vista denominada Layout (Panel de diseño).

- Cada fichero XML asociado a una pantalla/layout debe contener un elemento raíz y, dentro de él, se podrán añadir más layouts y componentes hijos hasta construir una jerarquía de Vistas (Views) que definiran la pantalla/layout.

- Ejemplo:

```
<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="com.example.saludo.MainActivity" >

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

</RelativeLayout>
```

Documentación en http://developer.android.com/reference/android/view/View.html



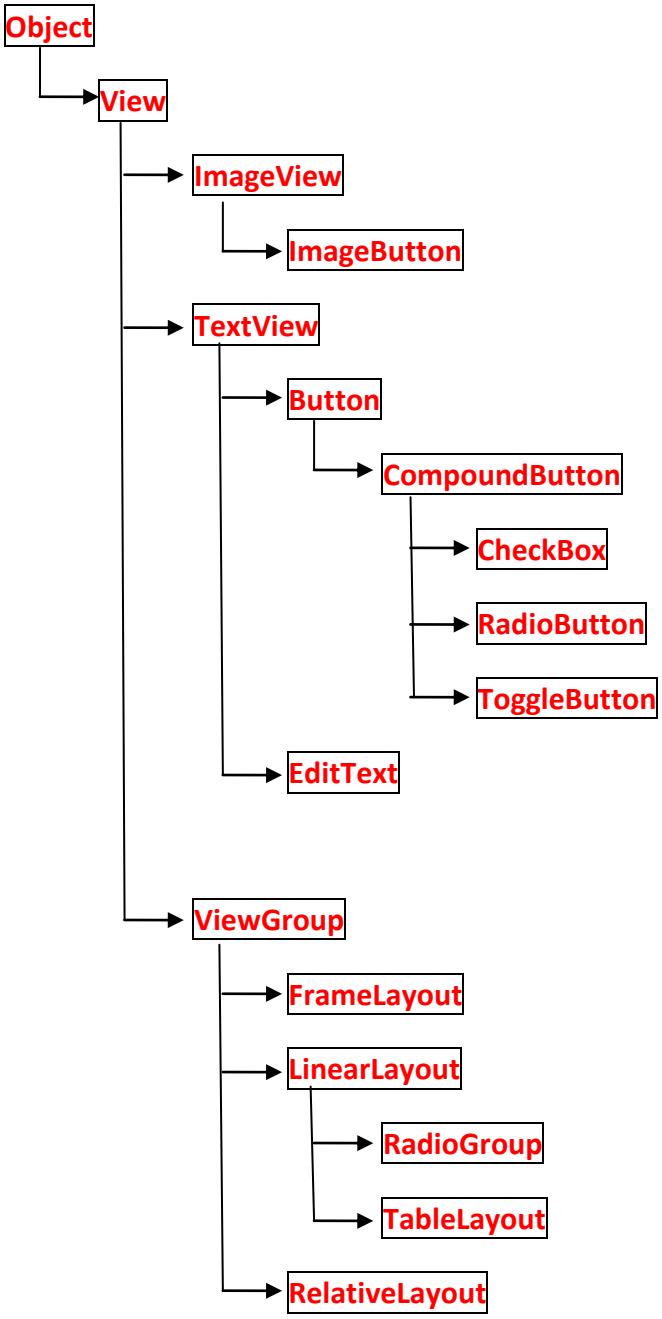
Button	Represents a push-button widget.
CheckBox	A checkbox is a specific type of two-states button that can be either checked or unchecked.
EditText	EditText is a thin veneer over TextView that configures itself to be editable.

...

FrameLayout	FrameLayout is designed to block out an area on the screen to display a single item.
LinearLayout	A Layout that arranges its children in a single column or a single row.
RelativeLayout	A Layout where the positions of the children can be described in relation to each other or to the parent.
TableLayout	A layout that arranges its children into rows and columns.

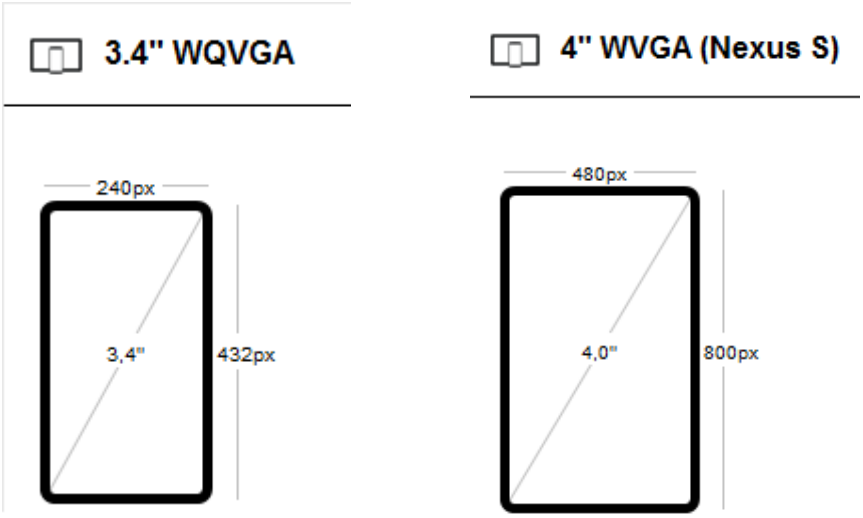
...

RESUMEN DE LA JERARQUÍA DE LA CLASE VIEW



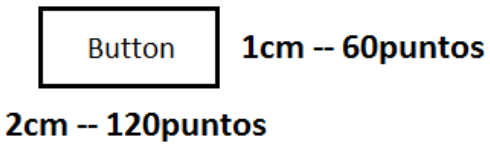
UNIDADES DE MEDIDA

- Respecto al **tamaño** y **resolución** de las pantallas es muy importante comprender cómo funcionan las unidades de medida.
- El **tamaño** de una pantalla se referencia por la **longitud de la diagonal** medida en **pulgadas**.
- La **resolución** de una pantalla se mide en **pixels**. La resolución en pixels es el **número de puntos reales** que tiene la pantalla, en horizontal y en vertical.



- La densidad de una pantalla se mide en:
  - **Puntos-por-pulgada** (**ppp**) o su equivalente
  - **Dots-per-inch** (**dpi**)
- En Android, la **densidad normal** tiene un valor de 160 puntos por pulgada o **160 dpi**. De esta forma, y sabiendo que 1 pulgada equivale a 2’54cm. aproximadamente, podemos calcular que 1cm equivale a 63 puntos en una pantalla de densidad normal (lo que se denomina **mdpi**)

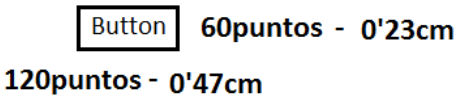
  Por ejemplo, un botón de 2cm*1cm tendría una medida en pixels de aproximadamente 120*60 pixels en una pantalla de densidad media o mdpi.



- Pero existen otros valores de densidades…

120 dpi	**ldpi**	*0.75
160 dpi	**mdpi**	*1
240 dpi	**hdpi**	*1.5
320 dpi	**xhdpi**	*2
480 dpi	**xxhdpi**	*3
640 dpi	**xxxhdpi**	*4

- El botón del ejemplo anterior, con la misma medida de pixels (120*60 pixels), ya no se vería igual en pantallas de otras densidades. Por ejemplo, en una pantalla xxxhdpi (densidad de 640 dpi) pasaría a medir 0.47*0.23 cm., aproximadamente



- Evitamos esta diferencia usando otra unidad: **puntos adimensionales** o **puntos independientes de la densidad**:

  Density-independient-pixel o **dip** (también se abrevia a **dp**)

  Entonces, 1 dip = 1 dp equivale a 1 pixel en una pantalla de densidad media (mdpi). Y en 1 cm de dicha pantalla "entrarían" 63 pixels o 63 dp.

- Android "escala" el valor en dp (puntos independientes de la densidad) para calcular el número real en pixels que se visualizarán en pantallas de otras densidades según la fórmula

  Nº pixels = nº dp * (valor dpi/160)

  Por ejemplo:

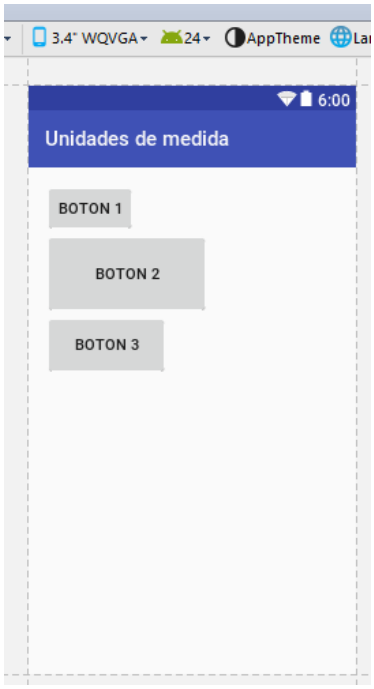
  120dp 120 * (160/160) = **120 pixels en pantalla mdpi**

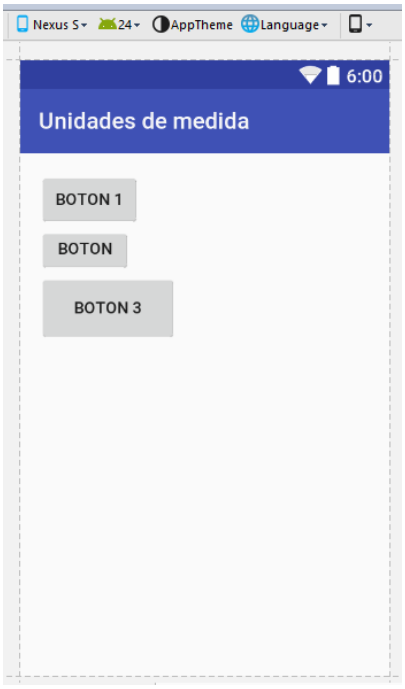
  120dp 120 * (320/160) = **240 pixels en pantalla xhdpi**

  120dp 120 * (640/160) = **480 pixels en pantalla xxxhdpi**

- Podemos ver esto con las capturas de un proyecto cuyo layout consta de 3 botones, dimensionados con diferentes unidades de medida, y visualizado en dos terminales de diferente densidad:

```
<?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:id="@+id/activity_main"
    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"
    android:orientation="vertical"
    tools:context="com.example.user.muchosbotones.MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="BOTON 1" />
    <Button
        android:layout_width="120px"
        android:layout_height="60px"
        android:text="BOTON 2" />
    <Button
        android:layout_width="120dp"
        android:layout_height="60dp"
        android:text="BOTON 3" />

</LinearLayout>
```





TIPOS DE PANELES (LAYOUT)

Panel Marco (FrameLayout)

- Es el panel más sencillo.
- Coloca todos sus componentes hijos pegados a su esquina superior izquierda de forma que cada componente nuevo añadido oculta el componente anterior.
- Se suele utilizar para mostrar un único control en su interior.
- Propiedades:
    - **android:layout_width**. Anchura. Valores posibles:
        - **match_parent.** El componente hijo tiene la dimensión del layout que lo contiene. (En API inferior a 8, el equivalente es **fill_parent**).
        - **wrap_content.** El componente hijo ocupa el tamaño de su contenido.
    - **android:layout_height**. Altura. Mismos valores.

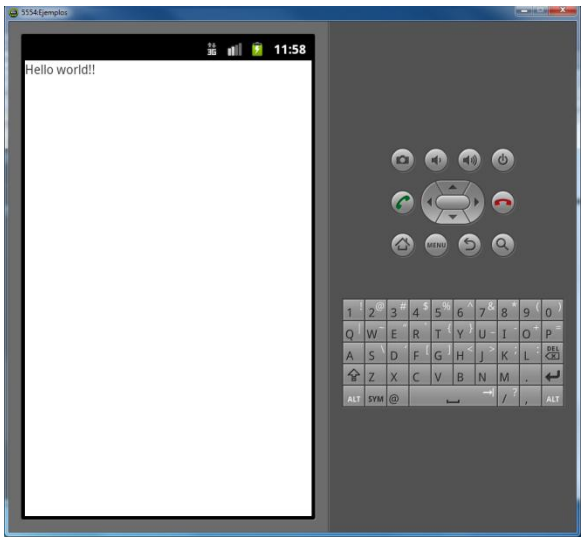
EJEMPLO:

Creamos un proyecto de nombre "**Interface_FrameLayout**", con una etiqueta de texto a modo de saludo.

Archivo **activity_main.xml**

```
<FrameLayout
       xmlns:android="http://schemas.android.com/apk/res/android"
       android:layout_width="match_parent"
       android:layout_height="match_parent">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />
</FrameLayout>
```

Ejecución en emulador



Vamos a probar la superposición de las vistas añadiendo otra etiqueta.

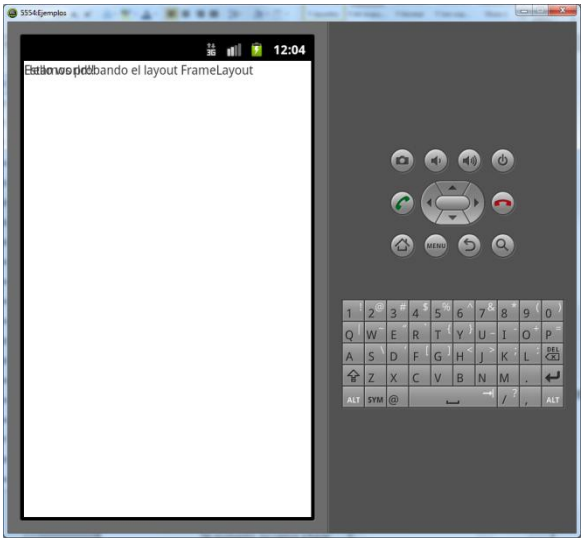
Archivo **activity_main.xml**

```
<FrameLayout
       xmlns:android="http://schemas.android.com/apk/res/android"
       android:layout_width="match_parent"
       android:layout_height="match_parent">

    <TextView
        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="@string/frame" />
</FrameLayout>
```

Las cadenas de texto se han definido en el archivo de recursos **strings.xml**:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Interface_FrameLayout</string>
    <string name="hello_world">Hello world!!</string>
    <string name="frame">Estamos probando el layout FrameLayout</string>

</resources>
```

Ejecución en emulador



Se observa como las etiquetas se superponen desde la esquina superior izquierda.

Panel Lineal (*LinearLayout*)

- Apila todos sus componentes hijos de forma horizontal o vertical, según se establezca la propiedad **android:orientation** con el valor "**vertical**" u "**horizontal**".
- Propiedades: igual que el FrameLayout, y también:
    - **android:layout_weight**. Permite establecer las dimensiones de los componentes hijos proporcionales entre ellos. Hay que tener en cuenta lo siguiente:
        - **Dirección**: el reparto proporcional sólo se realizará en la misma dirección en la que se haya definido el layout (vertical u horizontal).
        - **Tamaño**: los elementos que se vayan a dimensionar de forma proporcional deben tener un tamaño de 0dp en la dirección elegida (width=0dp para horizontal; height=0dp para vertical).
        - **Valor**: el tamaño elegido se realizará proporcionalmente a la suma de todos los pesos (o también al total: **android:weightSum**, si se ha especificado en el contenedor).

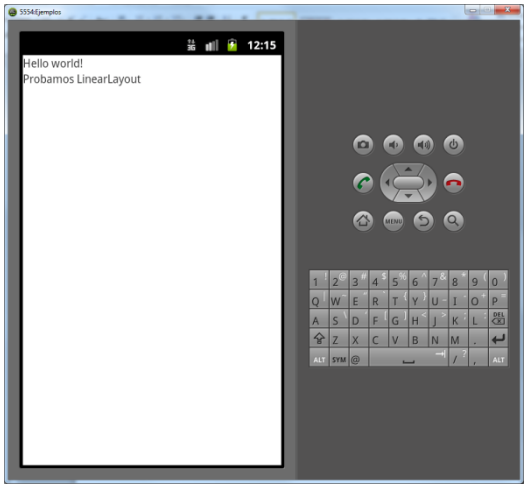
EJEMPLO:

Creamos un proyecto de nombre "**Interface_LinearLayout**", con dos etiquetas de texto como en el ejemplo anterior.

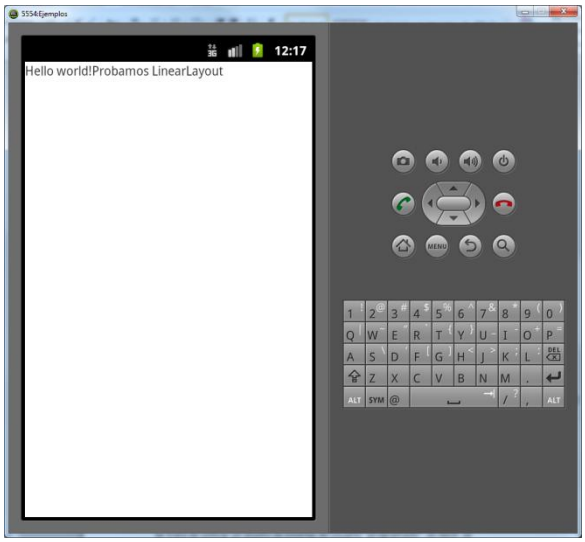
Archivo **activity_main.xml**

```
<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: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="@string/linear"/>
</LinearLayout>
```

Ejecución en emulador

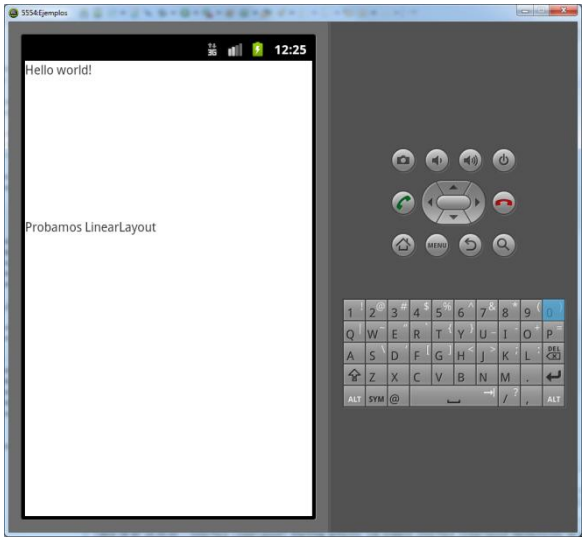


Propuesta: Variar el valor de la orientación y probar ejecución

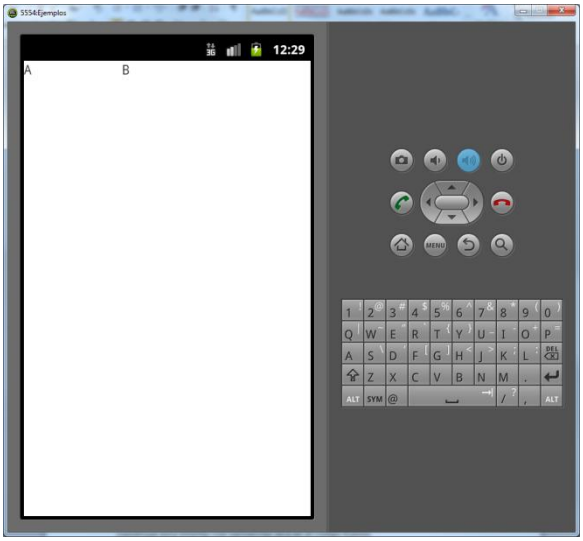


Propuesta: Averiguar cuál es el valor de la orientación por defecto.

Propuesta: Probar la propiedad **layout_weight** en el layout con horientación **vertical**, con los valores 1 para el primer componente TextView y 2 para el segundo.



Propuesta: Idem en el layout **horizontal**. Para ver mejor la proporción, en este caso, las etiquetas contendrán sólo una letra cada una (p. ej. A y B, respectivamente).



Panel Tabla (TableLayout)

- Permite distribuir todos los componentes hijos como si se tratara de una tabla mediante filas y columnas.
- La estructura de la tabla se define como en HTML indicando las filas mediante objetos **TableRow**.
- No existe un objeto especial para definir una columna (similar a *TableColumn*). Los controles necesarios se insertan directamente dentro del TableRow y cada uno de ellos se corresponderá con una columna de la tabla. Es decir, el número de filas de la tabla se corresponde con el número de elementos TableRow, y el número de columnas queda determinado por el número de componentes de la fila que más componentes contenga.
- El ancho de cada columna corresponde, en general, al ancho del mayor componente de dicha columna. Pero existen una serie de propiedades para modificar esto:
- Propiedades:
    - **android:layout_span**: una celda ocupa el espacio de varias columnas de la tabla (similar al atributo colspan de HTML).
    - **android:stretchColumns**: indica las columnas que se pueden expandir para ocupar el espacio libre que queda a la derecha de la tabla.
    - **android:shrinkColumns:** indica las columnas que se pueden contraer para dejar espacio al lado derecho de la tabla.
    - **android:collapseColumns:** indica las columnas de la tabla que se quieren ocultar completamente.

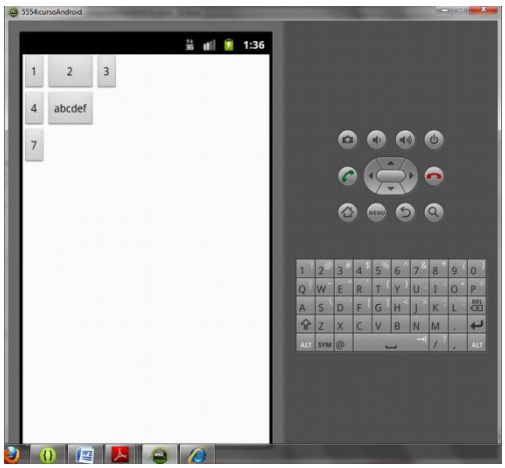
  Estas tres últimas propiedades se indican con el/los índices de las columnas separados por coma, o bien con el símbolo asterisco (*) para hacer referencia a todas las columnas. El índice de la primera columna tiene el valor 0.

EJEMPLO:

Creamos un proyecto de nombre "**Interface_TableLayout**". En lugar de etiquetas de texto, emplearemos botones para facilitar la visualización.

Archivo **activity_main.xml**

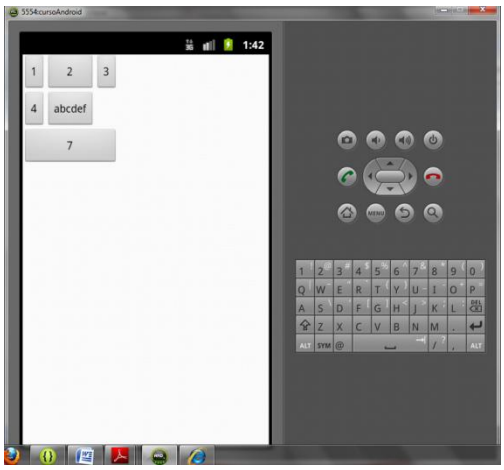
```
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent">

	<TableRow>
		<Button android:text="1" />
		<Button android:text="2" />
		<Button android:text="3" />
	</TableRow>
	<TableRow>
		<Button android:text="4" />
		<Button android:text="abcdef" />
	</TableRow>
	<TableRow>
		<Button android:text="7" />
	</TableRow>
</TableLayout>
```



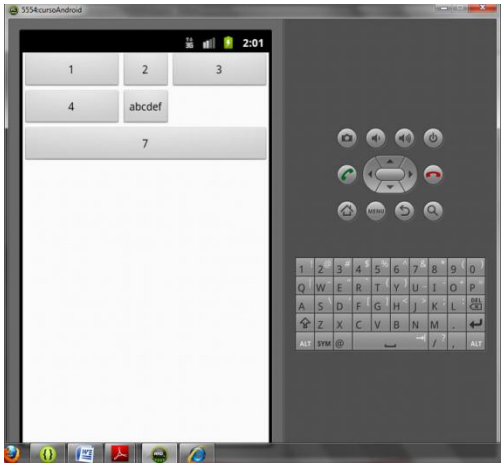
Ejecución en emulador

Cada celda tiene el ancho determinado por su contenido.
El ancho de la segunda columna se ajusta al de la celda de mayor contenido.

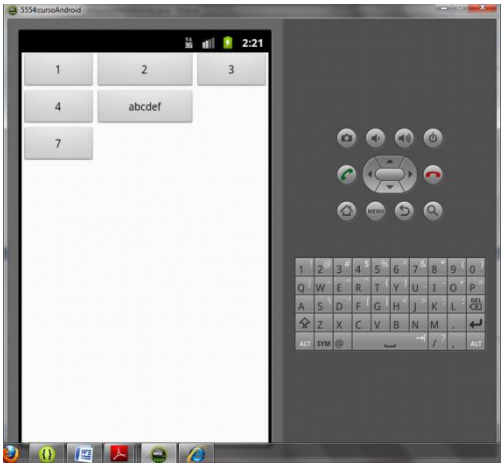
Propuesta: Expandir la última celda para que ocupe el espacio de las tres columnas, mediante `android:layout_span="3"`



Propuesta: expandir las columnas 1 y 3 para que ocupen el ancho que queda libre a la derecha de la pantalla, mediante `android:stretchColumns="0,2"`



Propuesta: probar el efecto de la propiedad `android:stretchColumns="*"`



Panel Relativo (RelativeLayout)

- Las vistas que lo componen pueden posicionarse en relación a otras vistas o al layout que las contiene.
- Propiedades al **posicionar una vista respecto a otra** (hace necesario usar un id para la vista que se va a usar como referencia):
  - **android:layout_above**: sitúa la parte inferior encima del id especificado.
  - **android:layout_below:** sitúa la parte superior debajo del id especificado.
  - **android:layout_toLeftOf**: sitúa a la izquierda.
  - **android:layout_toRightOf**: sitúa a la derecha.
  - **android:layout_alignLeft**: alinea el borde izqdo. con el del id especificado.
  - **android:layout_alignRight**: alinea el borde dcho. con el del id especificado.
  - **android:layout_alignTop**: alinea el borde superior con el del id especificado
  - **android:layout_alignBottom**: alinea el borde inferior con el del id especificado.

- Propiedades al **posicionar una vista respecto a su contenedor** (el cual es un RelativeLayout):
    - **android:layout_alignParentLeft**: si true, alinea con el borde izqdo. del padre.
    - **android:layout_alignParentRight**: si true, alinea con el borde dcho. del padre.
    - **android:layout_alignParentTop**: si true, alinea con el borde sup. del padre.
    - **android:layout_alignParentBottom**: si true, alinea con el borde inf. del padre.
    - **android:layout_centerHorizontal**: si true, centra en horiz. respecto al padre.
    - **android:layout_centerVertical**: si true, centra en vertical respecto al padre.
    - **android:layout_centerInParent**: si true, centra en ambos sentidos.
- Si no se referencia la posición, por defecto, todos los componentes se colocan en la parte superior izquierda de su contenedor padre.

EJEMPLO:

Creamos un proyecto de nombre "**Interface_RelativeLayout**", con dos etiquetas de texto como se indica a continuación.

Archivo **activity_main.xml**

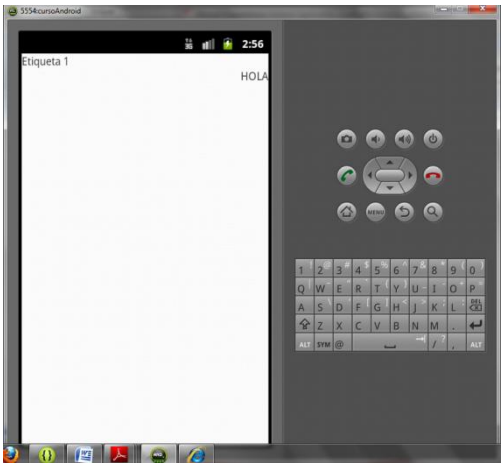
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="${relativePackage}.${activityClass}" >

    <TextView
        android:id="@+id/texto1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Etiqueta 1" />
     <!-- La segunda TextView, con el contenido HOLA, se situará debajo (below)
         de la primera (con id=texto1), y alineada a derecha en el Layout "padre"-->
    <TextView
        android:layout_below="@id/texto1"
        android:layout_alignParentRight="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="HOLA" />
</RelativeLayout>
```

Comentario

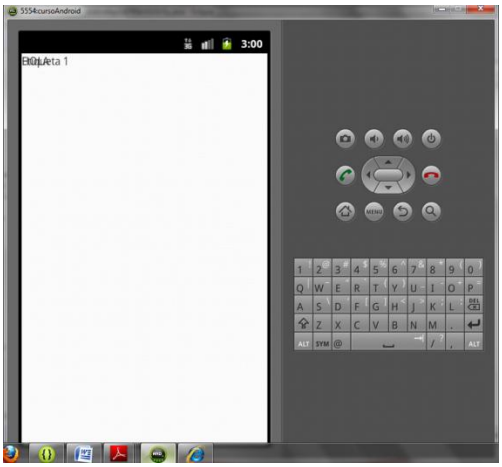
- **"@+id/cadena de texto**":
    - **android:id**. ID del elemento.
    - **@**: indica que lo que va a continuación es un recurso.
    - **+**: indica que el ID no existe y que hay que crearlo.
    - **tipo recurso**: en este caso, **id** (podría ser string, drawable, layout, etc.).
    - **cadena de texto**: es el nombre que se le da al identificador.

- **"@id/cadena de texto**": Para hacer referencia a ese recurso desde cualquier otro. No lleva el signo "+".

Ejecución en emulador

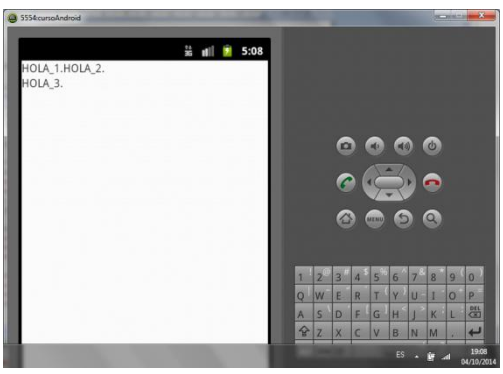


Propuesta: Probar sin hacer referencia a la posición de la segunda vista. Comprobar que ambas vistas se solapan en la esquina superior izquierda



Propuesta: Posicionar dos vistas en relación a otra.

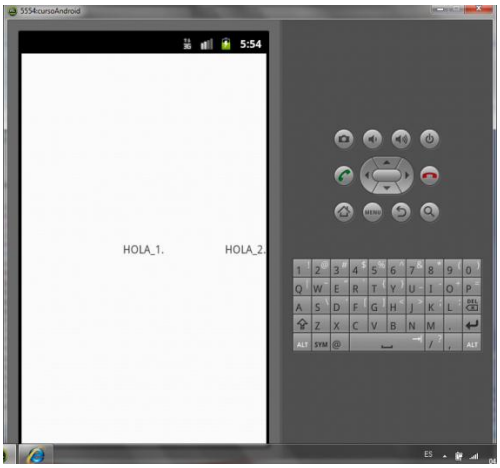
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="${relativePackage}.${activityClass}" >
      <!-- añadimos el identificador a la vista
           que se va a usar como referencia -->
    <TextView
        android:id="@+id/texto1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="HOLA_1." />
    <!--  y hacemos uso de este id en aquellas otras
vistas que
          se van a posicionar en relación a ella -->
```



```
    <TextView
        android:layout_toRightOf="@id/texto1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="HOLA_2." />
    <TextView
        android:layout_below="@id/texto1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="HOLA_3." />
</RelativeLayout>
```

Propuesta: Posicionar varias vistas en relación al layout contenedor y a otras vistas

```
<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"

tools:context="${relativePackage}.${activityClass}"
>
   <!-- centrado en el contenedor padre -->
    <TextView
        android:id="@+id/texto1"
        android:layout_centerInParent="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="HOLA_1." />
    <!-- alineado con la parte inferior de la
etiqueta anterior
        y a la dcha respecto al contenedor padre -->
    <TextView
        android:layout_alignBottom="@id/texto1"
        android:layout_alignParentRight="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="HOLA_2." />
</RelativeLayout>
```



COMPONENTES BASICOS

- Permiten al usuario interaccionar con la aplicación.
- Todos los componentes visuales están en el paquete **android.widget.**
  (Documentación en: https://developer.android.com/reference/android/widget/package-summary.html)