Aplicaciones Telemáticas

Tema 2.2.- Interfaz de Usuario (UI) y recursos en Android

J. E. López Patiño, F. J. Martínez Zaldívar

ETSIT-UPV

DEPARTAMENTO DE COMUNICACIONES

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

1. Interfaz de usuario
   - Definición
2. Vistas: objetos de la clase `View`
   - Vistas y layouts
   - Atributos XML `layout` y métodos de objetos `View`
3. `Layout`
   - Qué son y tipos
   - LinearLayout
   - TableLayout
   - FrameLayout
   - RelativeLayout
   - ConstraintLayout
4. Recursos
   - Concepto
   - Configuraciones alternativas de dispositivo
   - Alias de recursos
   - Estilos

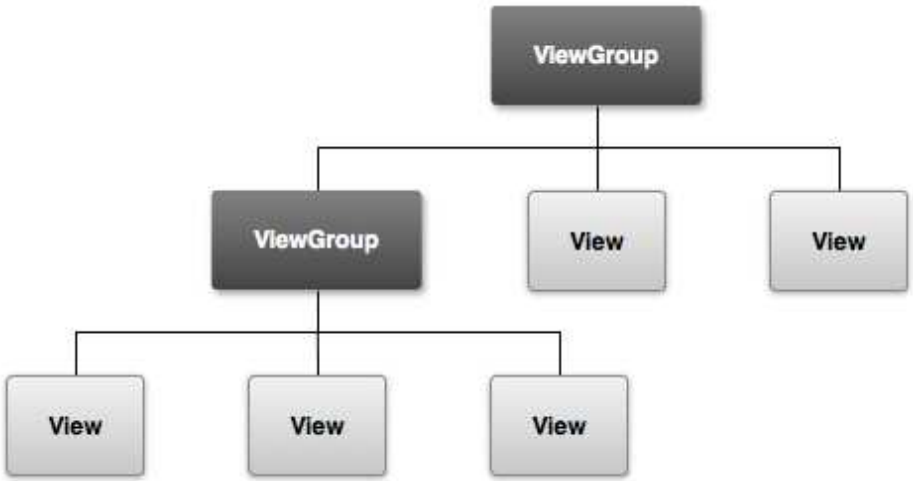
Índice

1 Interfaz de usuario
- Definición

Todo aquéllo que...

...el usuario puede ver y con lo que puede interactuar

- Todos los elementos de UI son objetos `View` y `ViewGroup`
  - `View`: clase de objetos que dibujan algo en la pantalla y el usuario puede interactuar con él.
  - `Viewgroup`: clase de objetos que contiene objetos `View` y otros objetos `Viewgroup`.



- `Android` proporciona subclases de `View` y `Viewgroup` con controles de entrada y disposición *(layout)*

Índice

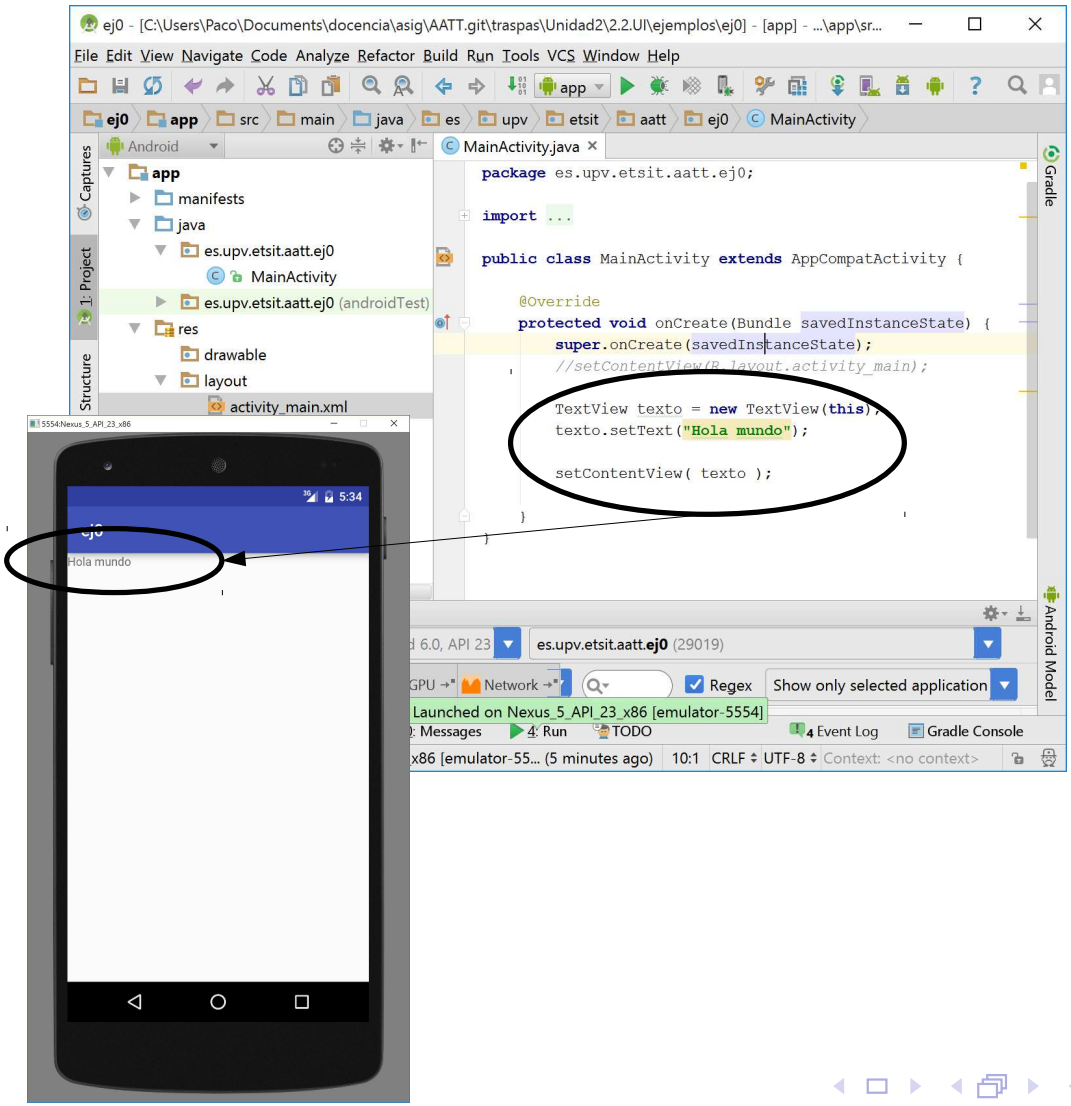
2 Vistas: objetos de la clase `View`
- Vistas y layouts
- Atributos XML `layout` y métodos de objetos `View`

¿Qué son?

- Son objetos de la clase `View` y permiten interactuar con el usuario
- Se pueden añadir *vistas* desde el código o desde uno o más ficheros XML
- Existen subclases de `View` especializadas:
  - Actúan como controles
  - Pueden visualizar texto, imágenes u otro contenido
- *Layouts*: disposición geométrica estructurada de objetos de la clase `View`.
  - Propiedades: dependerán del la subclase instanciada
  - Foco: forzado con `requestFocus()`
  - Registrar *listeners*: programar acciones frente a eventos
  - Visibilidad: mostrar u ocultar visibilidad de vistas

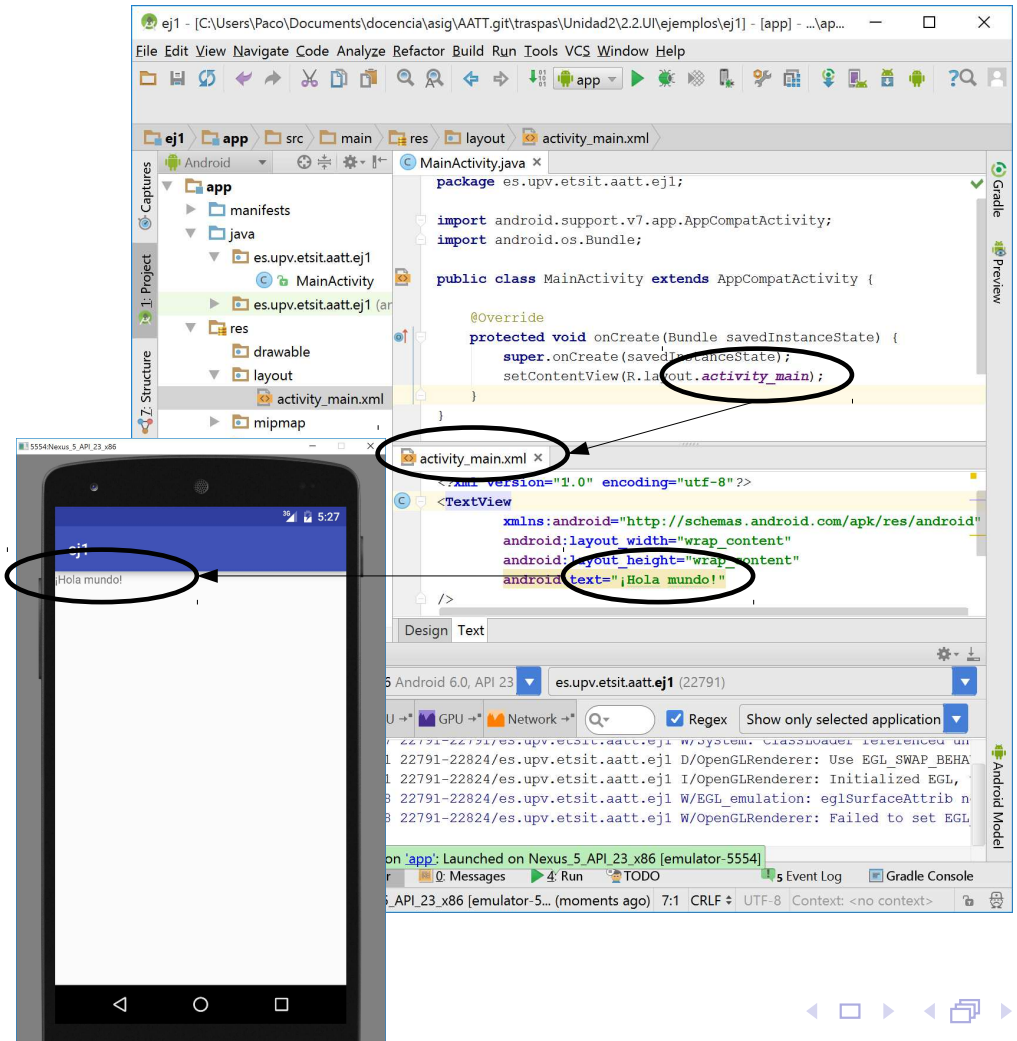
Vistas definidas desde Java

- Ejemplo `TextView`: `https://github.com/AATT-ETSIT/U2T2-Ej1-Vistas.git (v1)`



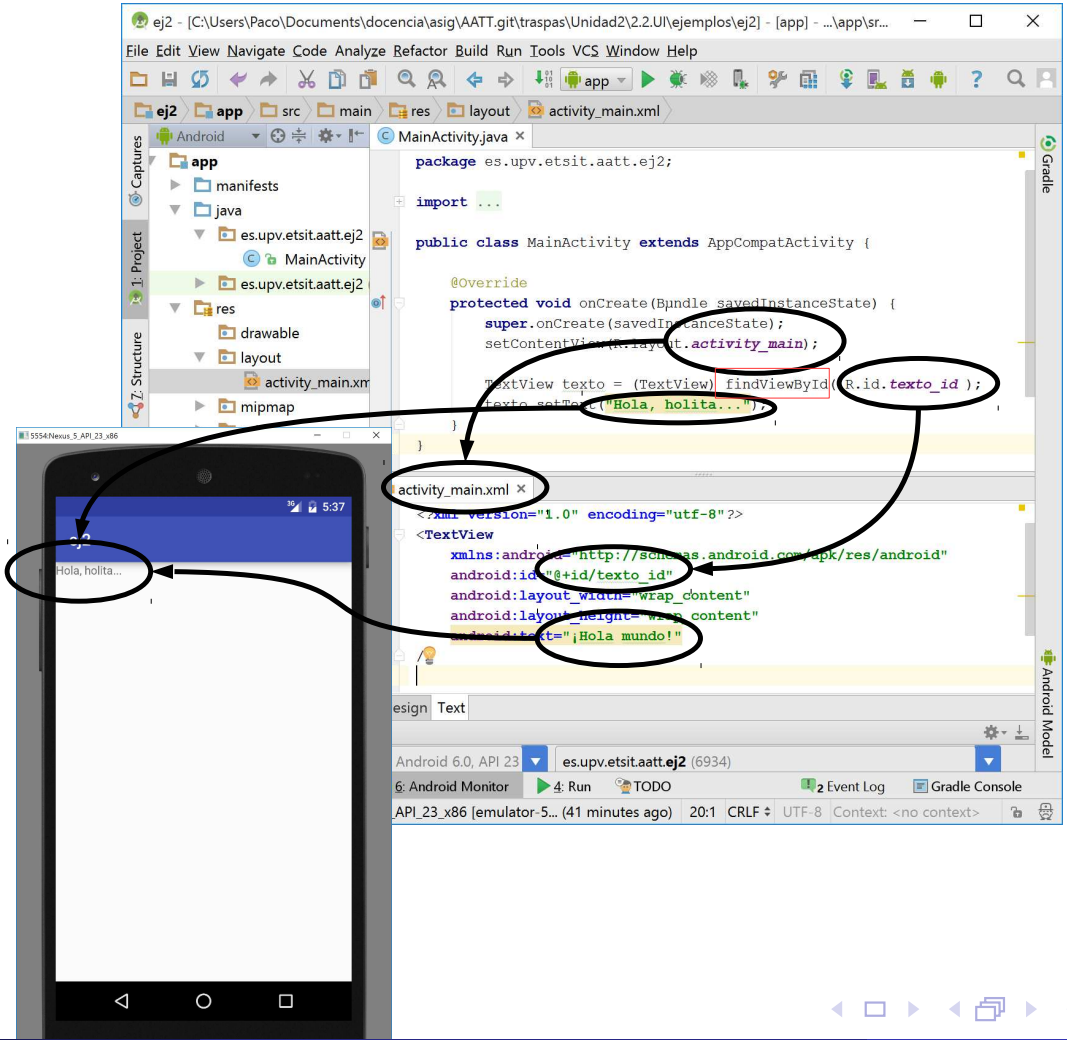
Alternativa: vistas definidas en fichero XML

- Ejemplo `TextView`: `https://github.com/AATT-ETSIT/U2T2-Ej1-Vistas.git (v2)`
- Mayor comodidad y flexibilidad



Alternativa: vistas definidas en fichero XML (cont.)

- Ejemplo `TextView`: `https://github.com/AATT-ETSIT/U2T2-Ej1-Vistas.git (v3)`
- Mayor comodidad y flexibilidad



Atributos de objetos `View`

Algunos implícitos o con valores por defecto

- `id`: identificación
- dimensiones: anchura y altura
- posición: relativa al *view* padre (izquierda y arriba)
- relleno y márgenes: huecos intra- y extra-*vistas*

id

- Cualquier objeto `View` puede tener un entero que lo identifique unívocamente.
- Sintaxis en fichero xml:

atributo

```
android:id="@+id/IDeNTiFiCaDoR"
```

- Sucesivas referencias (posteriores en el fichero), sin +:

atributo

```
android:layout_below="@id/IDeNTiFiCaDoR"
```

Sólo la que aparezca primero debe llevar el signo +. (Se admite que lleve siempre + pero puede provocar problemas de compatibilidad futura)

`id` (cont.)

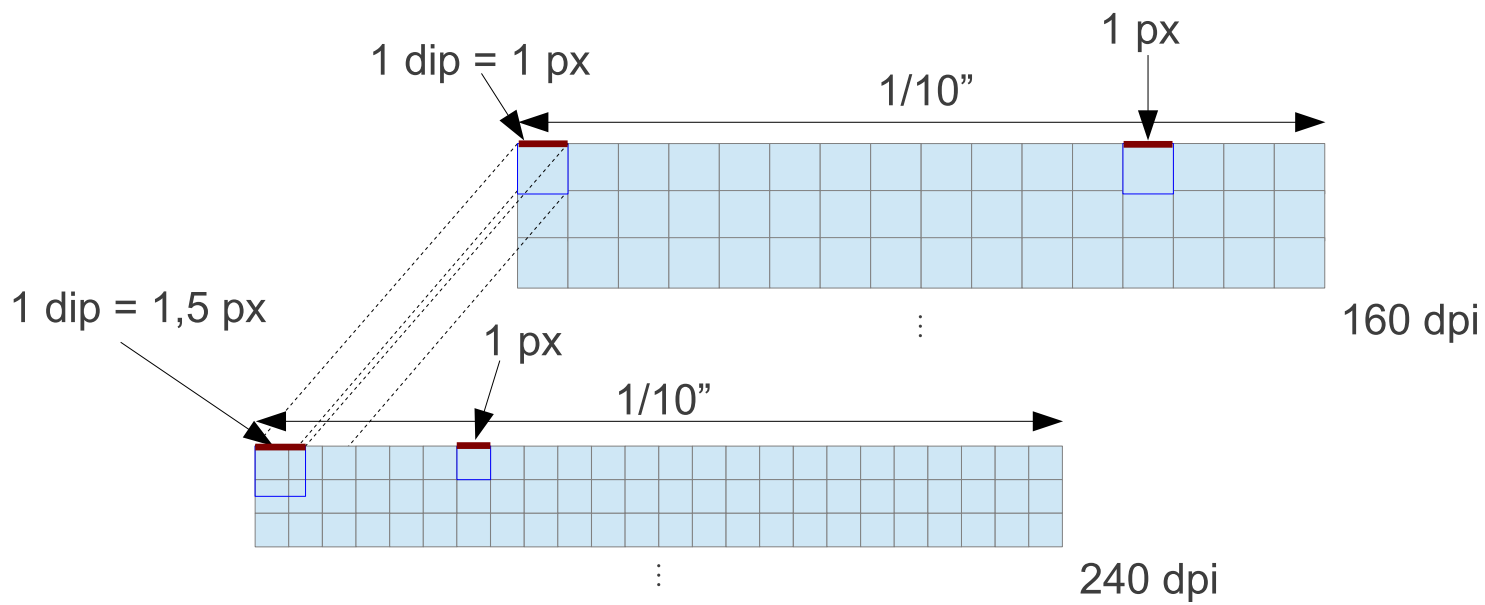
- El IDE/plugin se encarga de asignar/obtener el identificador numérico para cada objeto `View` especificado en ficheros XML
- No se debe modificar (autogenerado por `aapt`)
- Contiene los identificadores numéricos

Dimensiones

- Atributros con prefijo `layout_`: información al contenedor *padre*
- Ancho y alto:
  - `android:layout_width`
  - `android:layout_height`
- Especificación:
  - Constantes
    - `wrap_content`: dimensiones requeridas por su contenido
    - `match_parent` o `fill_parent` (obsoleto): tan grande como su padre
    - `match_constraint`: tan grande como permita la restricción (sólo en `ConstraintLayout`)
  - Valores y unidades
    - `px`: *pixels*
    - `in`: pulgadas
    - `mm`: milímetros
    - `pt`: puntos (1/72 pulgada)
    - `dp` ó `dip`: *density-independent pixels*, relativas a una pantalla de 160 dpi *(dots per inch)*
    - `sp`: *scale-independent pixels*, para fuentes de texto
    - ...

`dpi`, `px` y `dip(dp)`

- Tamaño de pantalla: longitud de diagonal en pulgadas
- Densidad de pantalla: `dpi` *(dots per inch)*, o `ppi` *(pixels per inch)*
- Resolución: número de pixels en cada dimensión
  - `px` *(pixels)*: pixel físico
  - `dip` o `dp` *(density independent pixel)*: pixel virtual relativo a densidad de 160 dpi

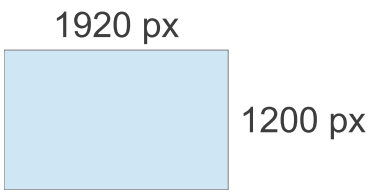


$\texttt{px} = \texttt{dip} \times (\texttt{dpi}/160)$ $\texttt{dip} = \texttt{px} \times (160/\texttt{dpi})$

Dimensiones pantalla

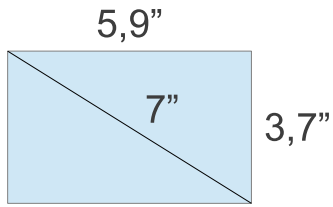
Ejemplo: 1920 × 1200 pixels a 323 dpi.

- Dimensiones físicas:



$$1920\ \text{px} \Rightarrow \frac{1920\ \text{px}}{323\ \text{dpi}} = \frac{1920\ \text{pixels}}{323\ \text{pixels/pulgada}} \approx 5{,}9\ \text{pulgadas}$$

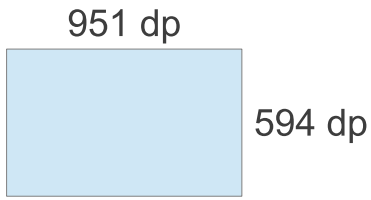
$$1200\ \text{px} \Rightarrow \frac{1200\ \text{px}}{323\ \text{dpi}} = \frac{1200\ \text{pixels}}{323\ \text{pixels/pulgada}} \approx 3{,}7\ \text{pulgadas}$$



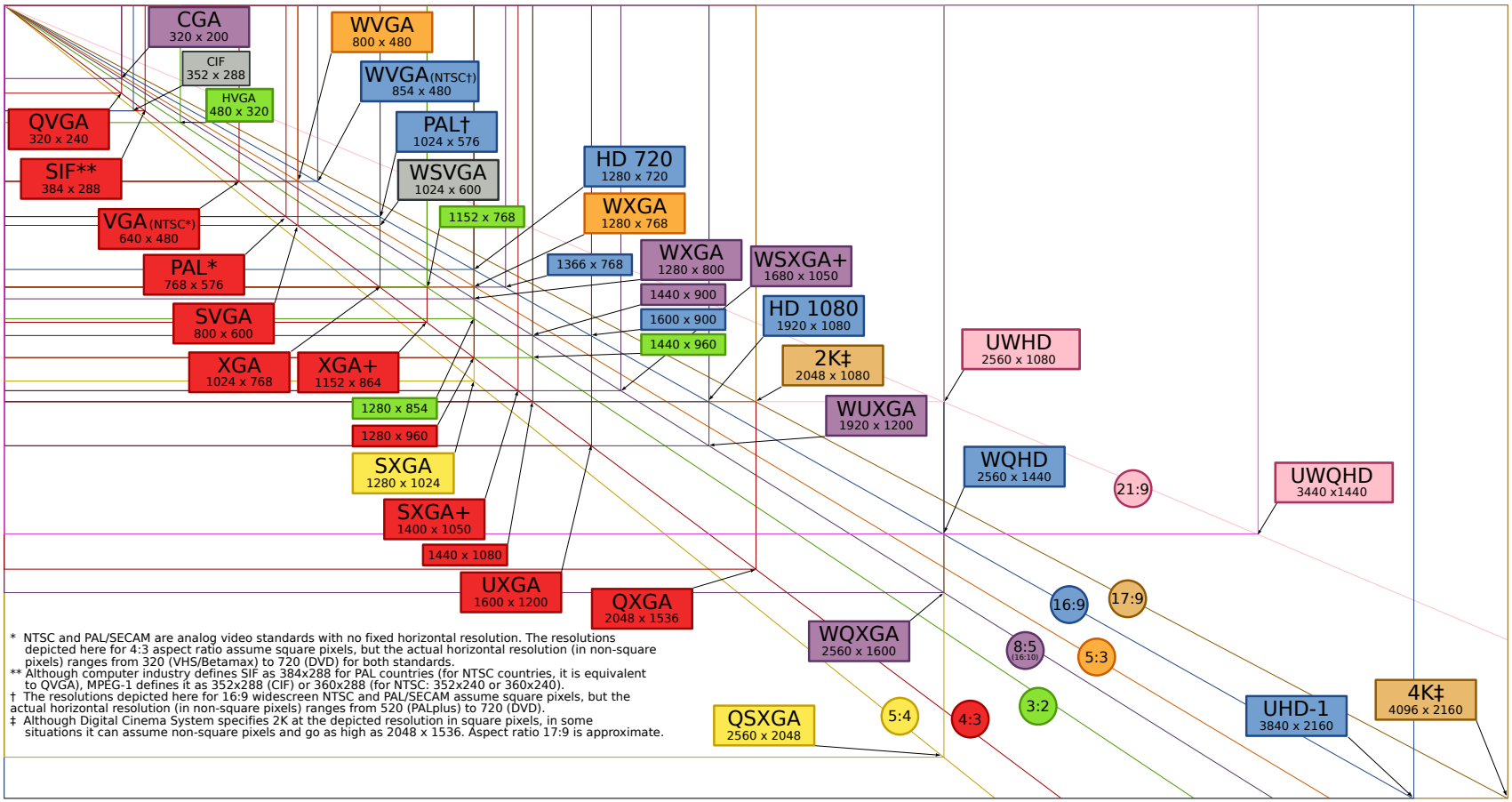
- Dimensiones relativas:

$$1920\ \text{px} \Rightarrow 1920\ \text{pixels} \cdot \frac{160\ \text{pixels/pulgada}}{323\ \text{pixels/pulgada}} \approx 951\ \text{pixels relativos a 160 dpi} \approx 951\ \text{dip} \approx 951\ \text{dp}$$

$$1200\ \text{px} \Rightarrow 1200\ \text{pixels} \cdot \frac{160\ \text{pixels/pulgada}}{323\ \text{pixels/pulgada}} \approx 594\ \text{pixels relativos a 160 dpi} \approx 594\ \text{dip} \approx 594\ \text{dp}$$



Estándares de vídeo



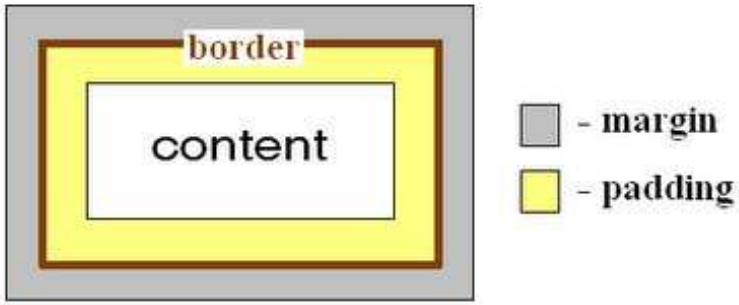
Densidades

Nombre	Calificación	Densidad aprox.	Factor relativo a 160 dpi
ldpi	(low)	≈120dpi	x0,75
mdpi	(medium)	≈160dpi	x1
hdpi	(high)	≈240dpi	x1,5
xhdpi	(extra-high)	≈320dpi	x2
xxhdpi	(extra-extra-high)	≈480dpi	x3
xxxhdpi	(extra-extra-extra-high)	≈640dpi	x4

Dimensiones: ejemplos

```
<TextView android:id="@+id/texto1"
          android:text="Este es el texto"
          android:layout_width="wrap_content"
          android:layout_height="50px"
          android:textColor="#FFFFFF"
/>

<Button android:id="@+id/Boton1"
        android:text="Boton"
        android:layout_width="25dip"
        android:layout_height="match_parent"
/>

<Button android:id="@+id/Boton2"
android:text="Boton"
android:layout_width="wrap_content"
android:layout_height="10dp"
/>
```

Modelo de caja: relleno *(padding)*



- *Padding*: relleno entre el borde del `View` y el contenido del `View` (interno al objeto `View`)
  - Atributos XML:
    - `android:padding`
    - `android:paddingBottom`
    - `android:paddingLeft`
    - `android:paddingRight`
    - `android:paddingTop`
  - Métodos:
    - `setPadding(int left, int top, int right, int bottom)`
    - `getPaddingLeft()`, `getPaddingRight()`, `getPaddingTop()`, `getPaddingBottom()`

Modelo de caja: márgenes *(margins)*

- *Margin*: espacio entre el borde y otro borde de otros objetos `View` contiguos (externo al objeto `View`).
  - Atributos XML:
    - `android:layout_margin`
    - `android:layout_marginBottom`
    - `android:layout_marginTop`
    - `android:layout_marginLeft`
    - `android:layout_marginRight`
  - Métodos:
    - setMargins(int left, int top, int right, int bottom)
    - ...

Índice

3 Layout
- Qué son y tipos
- LinearLayout
- TableLayout
- FrameLayout
- RelativeLayout
- ConstraintLayout

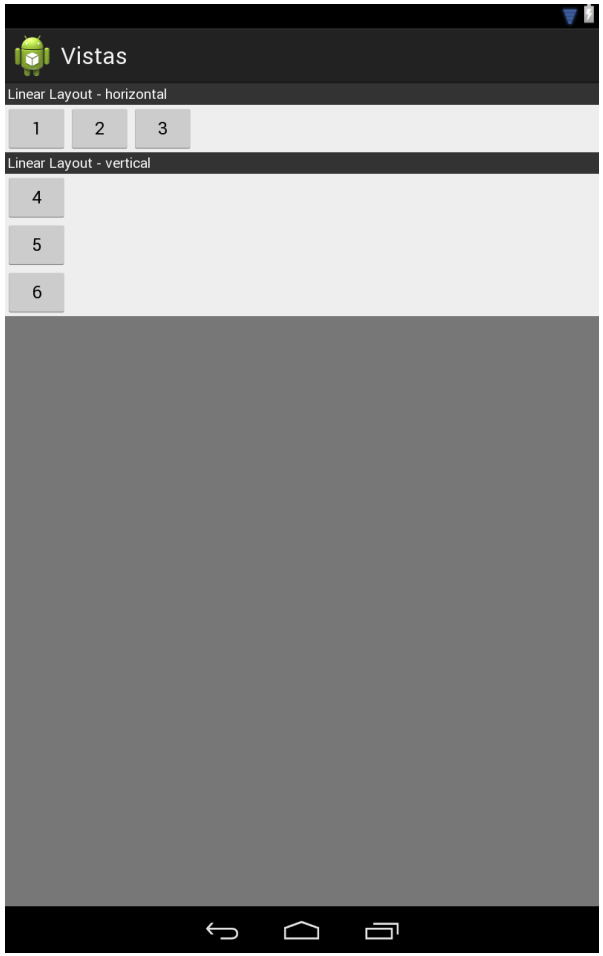
Layouts: contenedores de View

- Agrupación estructurada de objetos `View` *(ViewGroups)*
  - `LinearLayout`
  - `TableLayout`
  - `FrameLayout`
  - `RelativeLayout`
  - `ConstraintLayout`
  - `...`

Ejemplo:

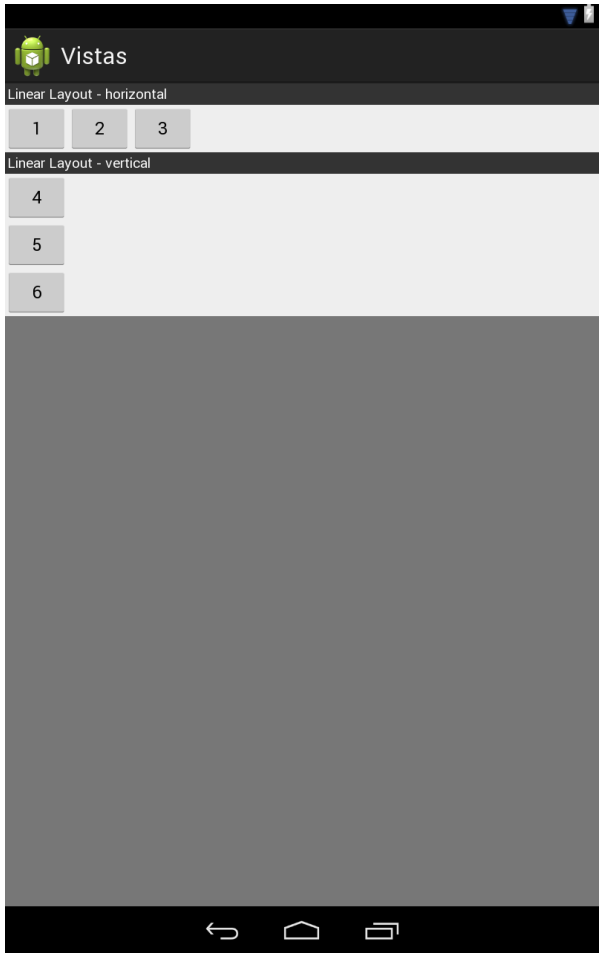
(https://github.com/AATT-ETSIT/U2T2-Ej2-LinearLayouts (v1))

```
<?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" android:background="#666666">

<TextView android:text="Linear Layout - horizontal"
android:layout_width="fill_parent" android:layout_height="wrap_content"
android:textColor="#FFFFFF" android:padding="2dip" />

<LinearLayout android:orientation="horizontal" android:background="#EEEEEE"
android:layout_width="fill_parent" android:layout_height="wrap_content">

<Button android:id="@+id/Button01"  android:text="1"
android:layout_width="wrap_content" android:layout_height="wrap_content"/>

<Button android:id="@+id/Button02"  android:text="2"
android:layout_width="wrap_content" android:layout_height="wrap_content"/>

<Button android:id="@+id/Button03"  android:text="3"
android:layout_width="wrap_content" android:layout_height="wrap_content"/>

</LinearLayout>
...
```



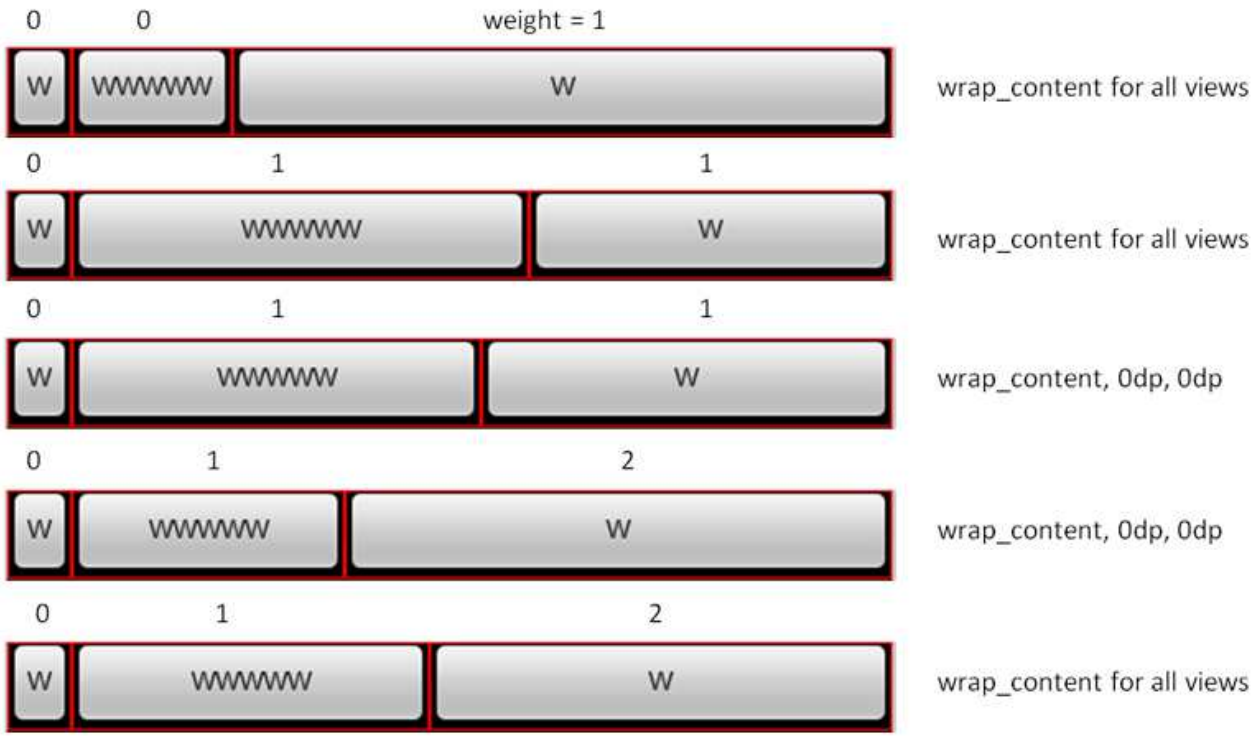
Ejemplo (cont.)

```
...
<TextView android:text="Linear Layout - vertical"
android:layout_width="fill_parent" android:layout_height="wrap_content"
android:textColor="#FFFFFF" android:padding="2dip" />

<LinearLayout android:orientation="vertical" android:background="#DDDDDD"
android:layout_width="fill_parent" android:layout_height="wrap_content">

<Button android:id="@+id/Button04"  android:text="4"
android:layout_width="wrap_content" android:layout_height="wrap_content"/>

<Button android:id="@+id/Button05"  android:text="5"
android:layout_width="wrap_content" android:layout_height="wrap_content"/>

<Button android:id="@+id/Button06"  android:text="6"
android:layout_width="wrap_content" android:layout_height="wrap_content"/>

</LinearLayout>

</LinearLayout>
```



Peso *(weight)*

- Peso relativo de una vista respecto a las restantes contenidas en un contenedor `Layout`: `android:layout_weight`
- Casuística:



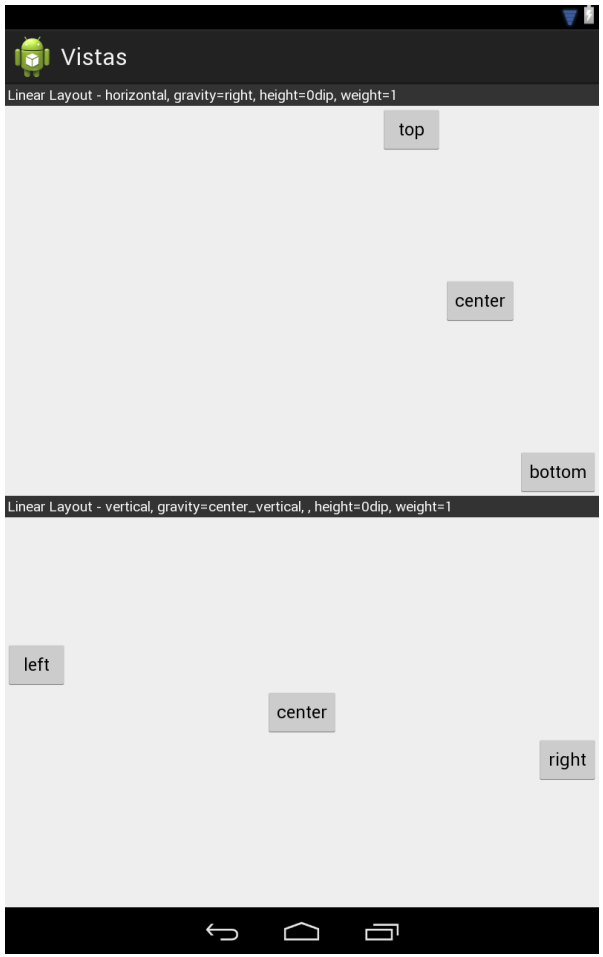
Gravedad *(gravity)*

- Gravity: justificación horizontal o vertical
  - Horizontal: *top*, *center_horizontal*, *bottom*, *fill_horizontal*, ...
  - Vertical: *left*, *center_vertical*, *right*, *fill_vertical*, ...
  - Horizontal y vertical: *center*, *fill*, ...
- Pueden combinarse mediante operador |
- Diferencia:
  - `android:layout_gravity`: justificación para que la tenga en cuenta el *padre*
  - `android:gravity`: justificación del *contenido* del `View`

Gravedad *(gravity)*

(https://github.com/AATT-ETSIT/U2T2-Ej2-LinearLayouts (v2))

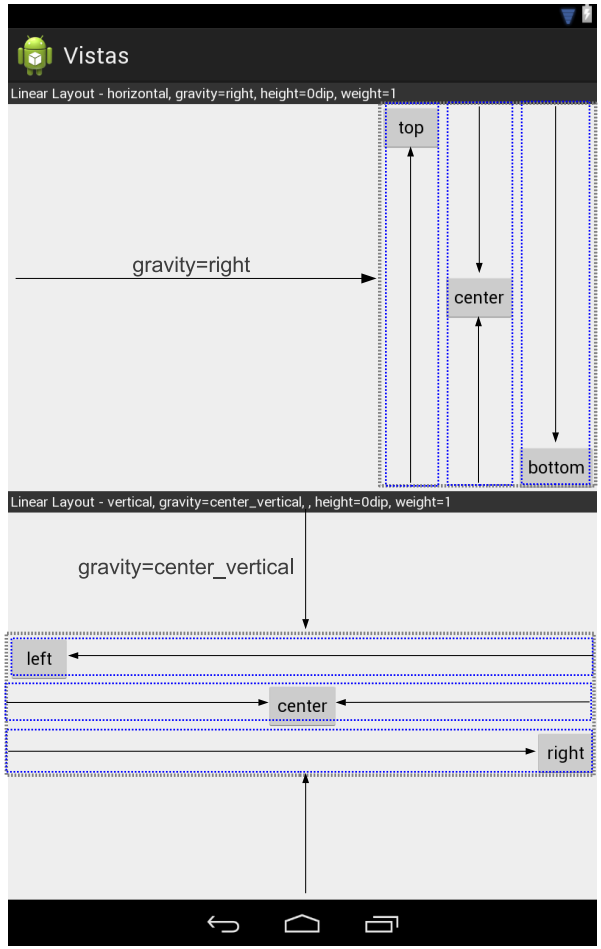
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent" android:layout_height="fill_parent"
android:orientation="vertical" android:background="#777777">

<TextView android:text="Linear Layout - horizontal, gravity=right,
height=0dip, weight=1"
android:layout_width="fill_parent" android:layout_height="wrap_content"
android:textColor="#FFFFFF" android:background="#333333"
android:padding="2dip" />

<LinearLayout android:orientation="horizontal" android:background="#EEEEEE"
android:layout_width="fill_parent"
android:layout_height="0dip" android:layout_weight="1"
android:gravity="right" >

<Button android:id="@+id/Button01"  android:text="top"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="top" />

<Button android:id="@+id/Button02"  android:text="center"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="center" />

<Button android:id="@+id/Button03"  android:text="bottom"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="bottom" />

</LinearLayout>
...
```



Gravedad *(gravity)*

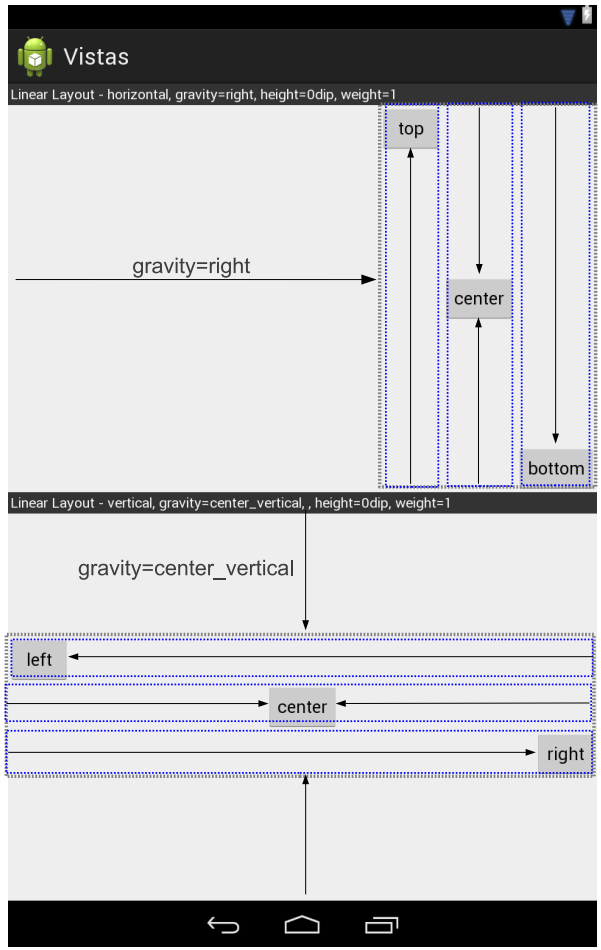
(https://github.com/AATT-ETSIT/U2T2-Ej2-LinearLayouts (v2))

```
<?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" android:background="#777777">

<TextView android:text="Linear Layout - horizontal, gravity=right,
height=0dip, weight=1"
android:layout_width="fill_parent" android:layout_height="wrap_content"
android:textColor="#FFFFFF" android:background="#333333"
android:padding="2dip" />

<LinearLayout android:orientation="horizontal" android:background="#EEEEEE"
android:layout_width="fill_parent"
android:layout_height="0dip" android:layout_weight="1"
android:gravity="right" >

<Button android:id="@+id/Button01"  android:text="top"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="top" />

<Button android:id="@+id/Button02"  android:text="center"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="center" />

<Button android:id="@+id/Button03"  android:text="bottom"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="bottom" />

</LinearLayout>
...
```



Gravedad *(gravity)*

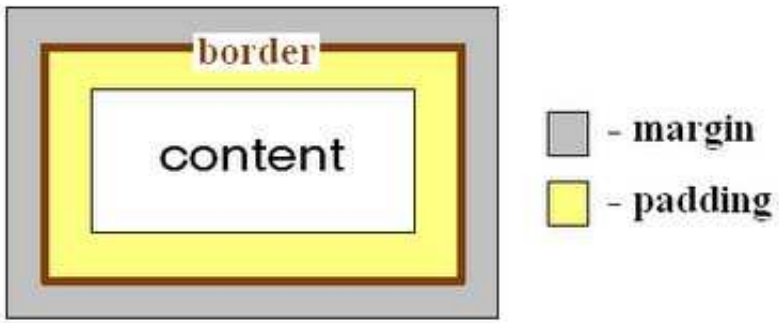
```
...
<TextView android:text="Linear Layout - vertical, gravity=center_vertical,
height=0dip, weight=1"
android:layout_width="fill_parent" android:layout_height="wrap_content"
android:textColor="#FFFFFF" android:background="#333333"
android:padding="2dip" />

<LinearLayout android:orientation="vertical" android:background="#EEEEEE"
android:layout_width="fill_parent"
android:layout_height="0dip" android:layout_weight="1"
android:gravity="center_vertical" >

<Button android:id="@+id/Button01"  android:text="left"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="left" />

<Button android:id="@+id/Button02"  android:text="center"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="center" />

<Button android:id="@+id/Button03"  android:text="right"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="right"/>

</LinearLayout>

</LinearLayout>
```



Relleno *(padding)* y márgenes *(margin)*

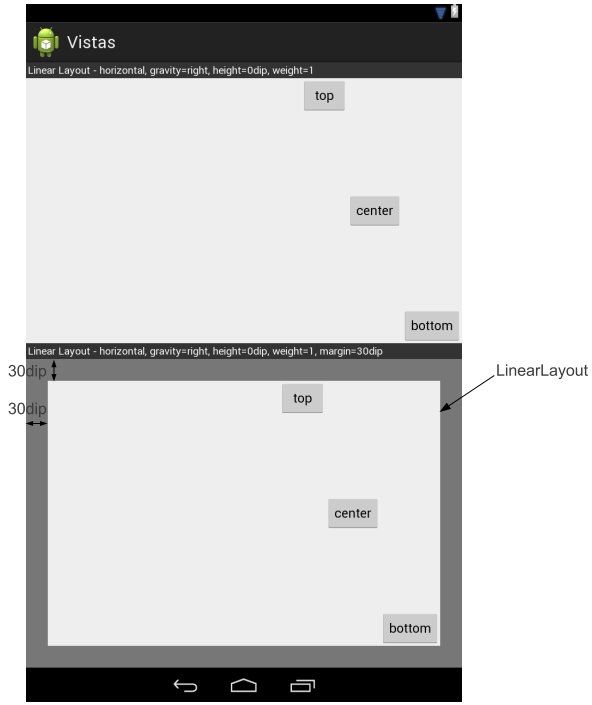
(`https://github.com/AATT-ETSIT/U2T2-Ej2-LinearLayouts (v3)`)

- *Padding*: relleno entre el borde del `View` y el contenido del `View` (interno al objeto `View`). Atributos XML:
  - `android:padding`
  - `android:paddingBottom`
  - `android:paddingLeft`
  - `android:paddingRight`
  - `android:paddingTop`
- *Margin*: espacio entre el borde y otro borde de otros objetos `View` contiguos (externo al objeto `View`). Atributos XML:
  - `android:layout_margin`
  - `android:layout_marginBottom`
  - `android:layout_marginTop`
  - `android:layout_marginLeft`
  - `android:layout_marginRight`



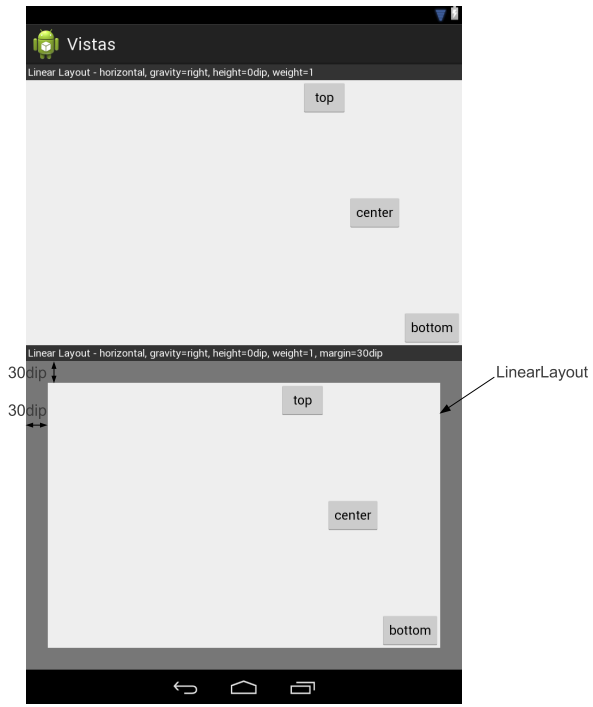
Ejemplo márgenes

```
<?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" android:background="#777777">

<TextView android:text="Linear Layout - horizontal, gravity=right,
height=0dip, weight=1"
android:layout_width="fill_parent" android:layout_height="wrap_content"
android:textColor="#FFFFFF" android:background="#333333"
android:padding="2dip" />

<LinearLayout android:orientation="horizontal" android:background="#EEEEEE"
android:layout_width="fill_parent"
android:layout_height="0dip" android:layout_weight="1"
android:gravity="right" >

<Button android:id="@+id/Button01"  android:text="top"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="top" />

<Button android:id="@+id/Button02"  android:text="center"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="center" />

<Button android:id="@+id/Button03"  android:text="bottom"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="bottom" />

</LinearLayout>
...
```



Ejemplo márgenes

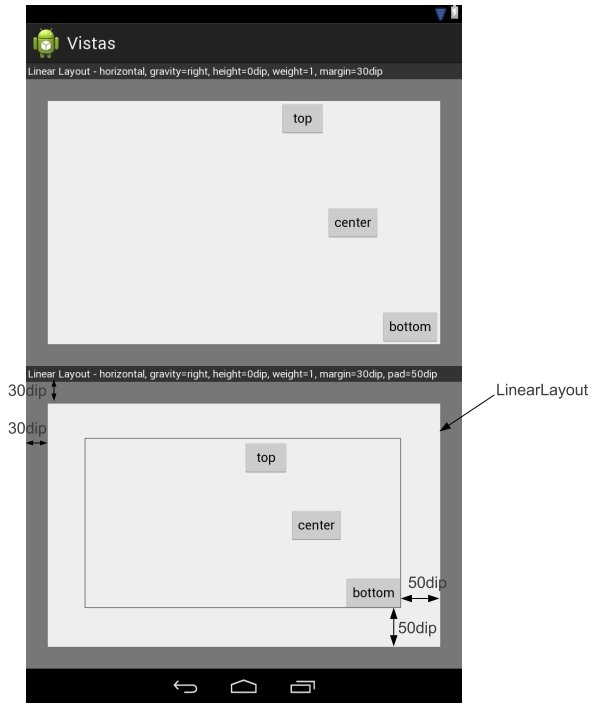
```
...
<TextView android:text="Linear Layout - horizontal, gravity=right,
height=0dip, weight=1, margin=30dip"
android:layout_width="fill_parent" android:layout_height="wrap_content"
android:textColor="#FFFFFF" android:background="#333333"
android:padding="2dip" />

<LinearLayout android:orientation="horizontal" android:background="#EEEEEE"
android:layout_width="fill_parent"
android:layout_height="0dip" android:layout_weight="1"
android:gravity="right" android:layout_margin="30dip" >

<Button android:id="@+id/Button01"  android:text="top"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="top" />

<Button android:id="@+id/Button02"  android:text="center"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="center" />

<Button android:id="@+id/Button03"  android:text="bottom"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="bottom" />

</LinearLayout>

</LinearLayout>
```



Ejemplo relleno (y márgenes)

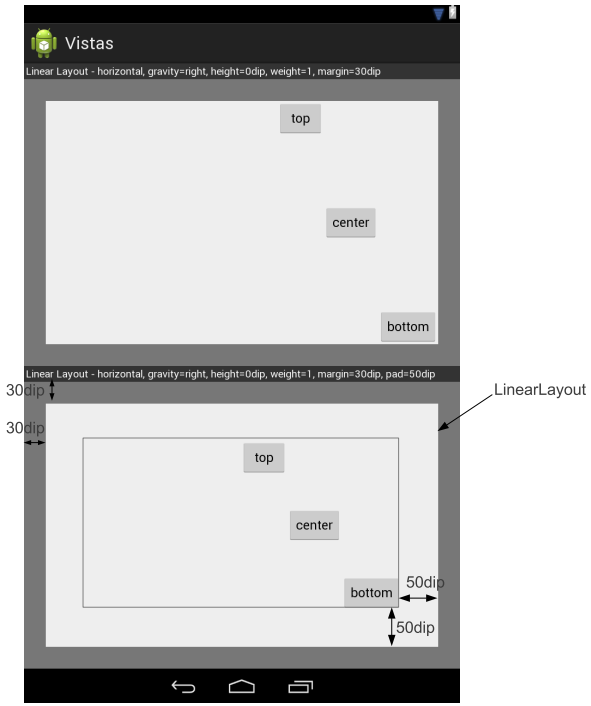
(https://github.com/AATT-ETSIT/U2T2-Ej2-LinearLayouts (v4))

```
<?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" android:background="#777777">


<TextView android:text="Linear Layout - horizontal, gravity=right,
height=0dip, weight=1, margin=30dip"
android:layout_width="fill_parent" android:layout_height="wrap_content"
android:textColor="#FFFFFF" android:background="#333333"
android:padding="2dip" />

<LinearLayout android:orientation="horizontal" android:background="#EEEEEE"
android:layout_width="fill_parent"
android:layout_height="0dip" android:layout_weight="1"
android:gravity="right" android:layout_margin="30dip" >

<Button android:id="@+id/Button01"  android:text="top"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="top" />

<Button android:id="@+id/Button02"  android:text="center"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="center" />

<Button android:id="@+id/Button03"  android:text="bottom"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="bottom" />

</LinearLayout>
...
```



Ejemplo relleno (y márgenes)

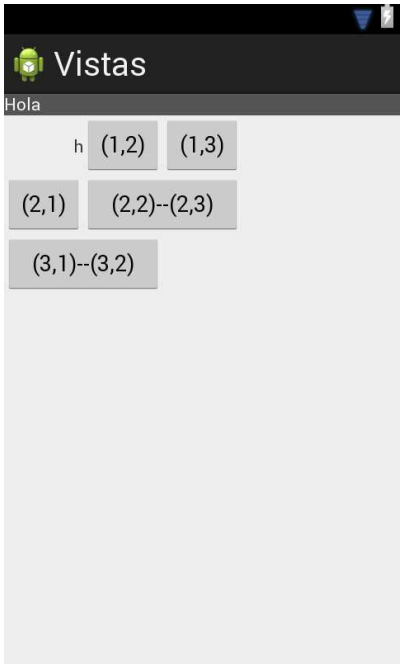
```
...
<TextView android:text="Linear Layout - horizontal, gravity=right,
height=0dip, weight=1, margin=30dip, pad=50dip"
android:layout_width="fill_parent" android:layout_height="wrap_content"
android:textColor="#FFFFFF" android:background="#333333"
android:padding="2dip" />

<LinearLayout android:orientation="horizontal" android:background="#EEEEEE"
android:layout_width="fill_parent"
android:layout_height="0dip" android:layout_weight="1"
android:gravity="right" android:layout_margin="30dip"
android:padding="50dip" >

<Button android:id="@+id/Button01"  android:text="top"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="top" />

<Button android:id="@+id/Button02"  android:text="center"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="center" />

<Button android:id="@+id/Button03"  android:text="bottom"
android:layout_width="wrap_content" android:layout_height="wrap_content"
android:layout_gravity="bottom" />

</LinearLayout>


</LinearLayout>
```



Disposición en forma de filas

(https://github.com/AATT-ETSIT/U2T2-Ej2-OtrosLayouts (v1))

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#EEEEEE">


    <TextView android:text="h"
              android:background="#555555"
              android:textColor="#FFFFFF"/>

    <TableRow>
        <TextView android:text="h" android:gravity="right"/>
        <Button android:text="(1,2)" />
        <Button android:text="(1,3)" />
    </TableRow>

    <TableRow>
        <Button android:text="(2,1)" />
        <Button android:text="(2,2)--(2,3)"
                android:layout_span="2" />
    </TableRow>

    <TableRow>
        <Button android:text="(3,1)--(3,2)"
                android:layout_span="2" />

    </TableRow>
```



Otros atributos:

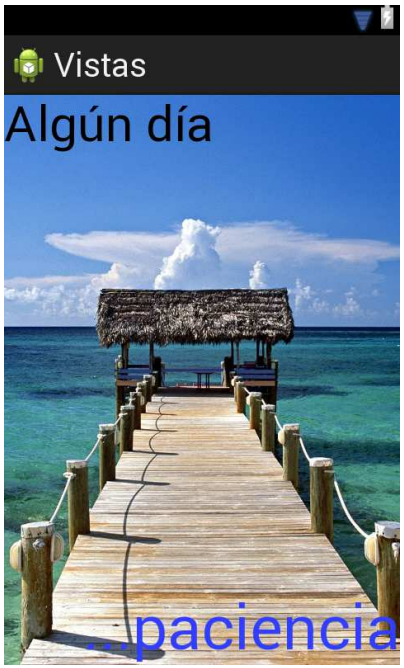
- android:stretchColumnms
- android:shrinkColumns
- android:collapseColumns

En FrameLayout las View se *apilan*

(https://github.com/AATT-ETSIT/U2T2-Ej2-OtrosLayouts (v2))

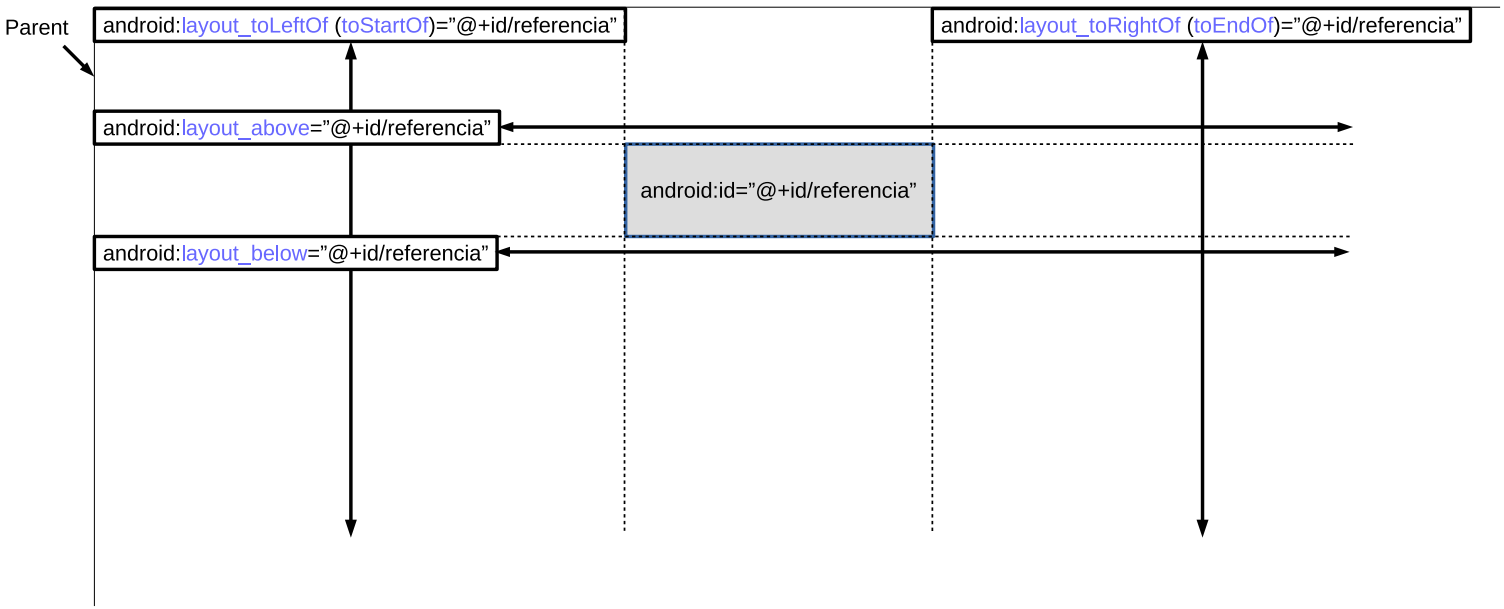
Orden según XML; mecanismos para selección/visualización

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

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

    <ImageView
        android:id="@+id/ImageView01"
        android:layout_width="fill_parent" android:layout_height="fill_parent"
        android:scaleType="centerCrop"
        android:src="@drawable/playa" />

    <TextView
        android:layout_width="fill_parent" android:layout_height="wrap_content"
        android:textColor="#000"
        android:textSize="40dp"
        android:text="Algún día" />

    <TextView
        android:layout_width="fill_parent" android:layout_height="wrap_content"
        android:layout_gravity="bottom"
        android:gravity="right"
        android:text="...paciencia"
        android:textColor="#333fff"
        android:textSize="50dp" />

</FrameLayout>
```

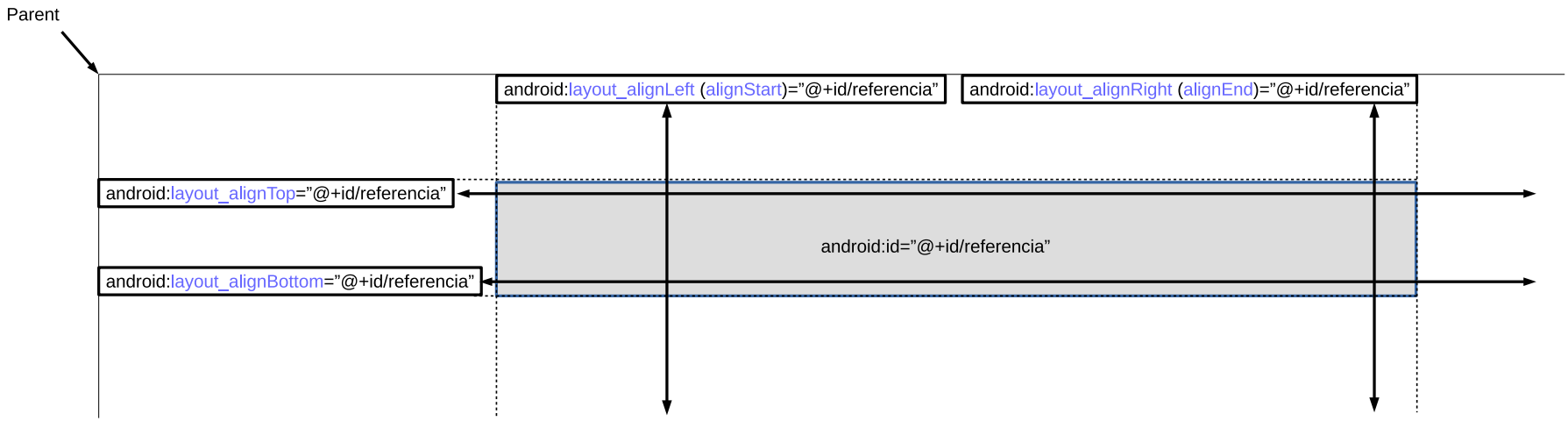


Atributos de `RelativeLayout` respecto a otro elemento (I)

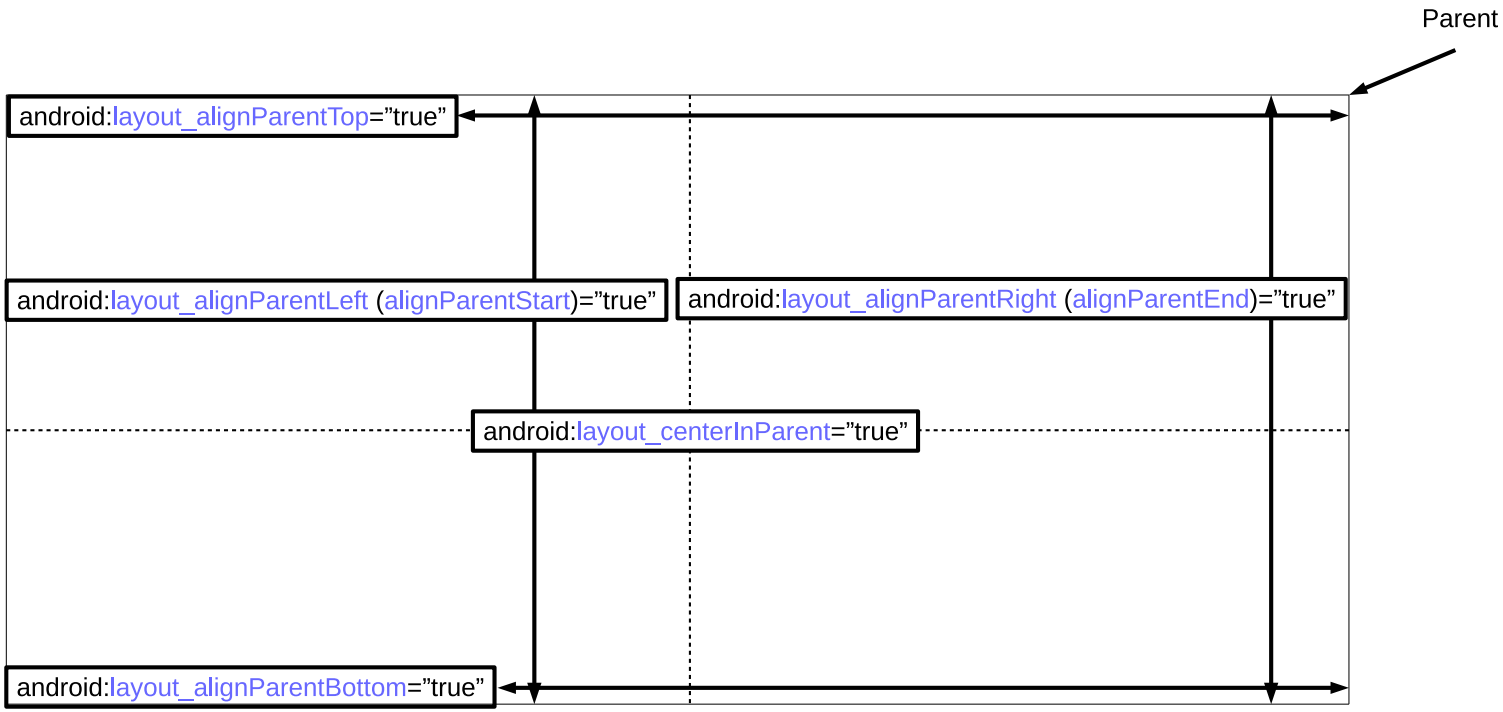
- Por defecto: arriba e izquierda (dentro de contenido de `parent`)
- Combinación de múltiples atributos si son coherentes



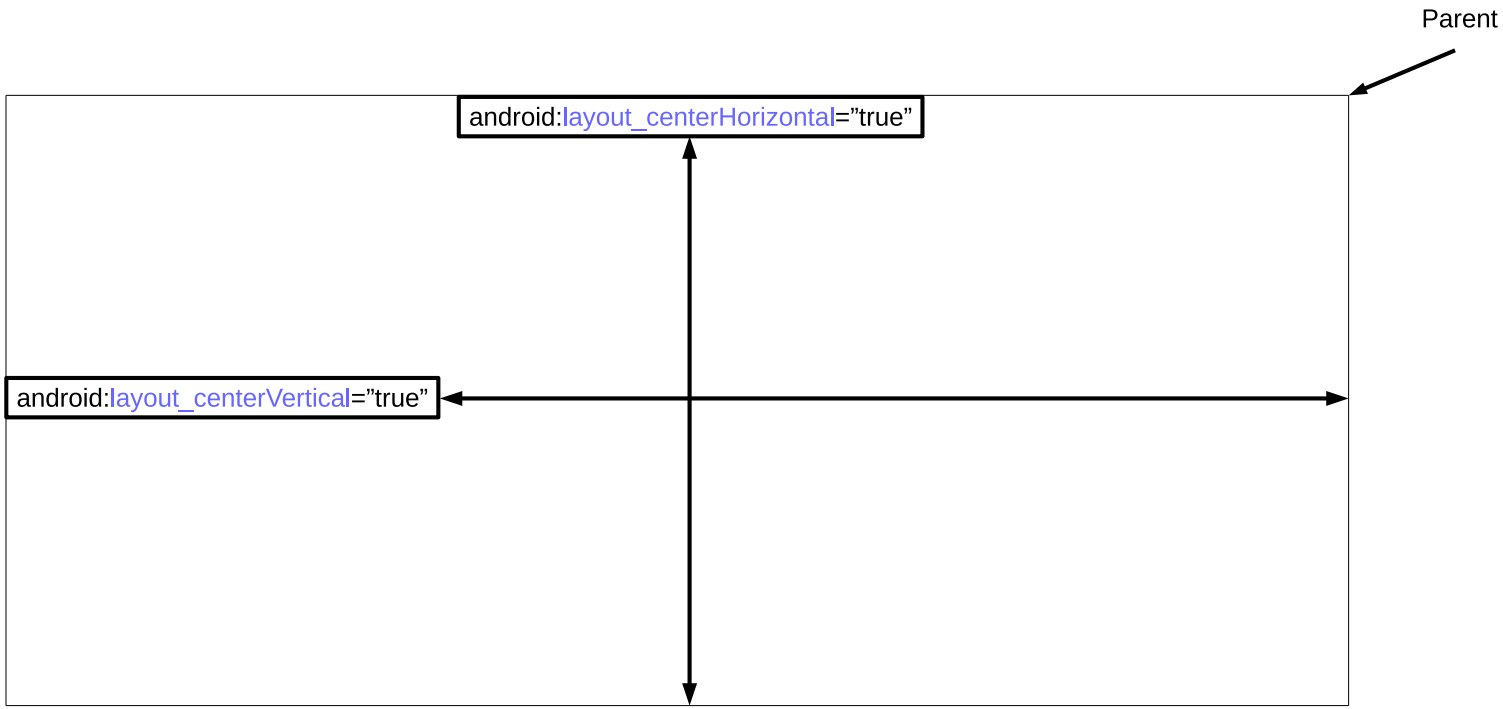
Atributos de RelativeLayout respecto a otro elemento (II)



Atributos de RelativeLayout respecto a parent (I)



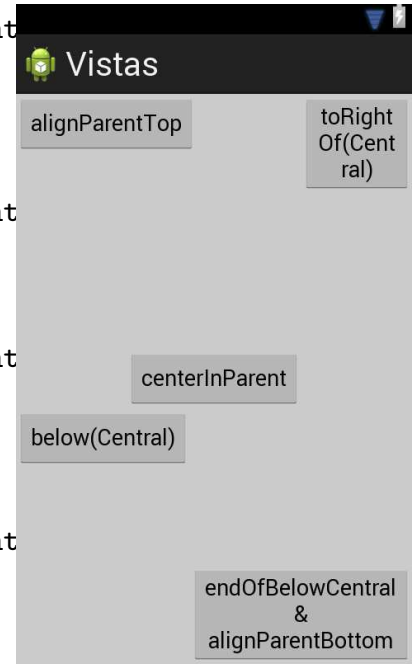
Atributos de RelativeLayout respecto a parent (II)



Ejemplo con RelativeLayout

(https://github.com/AATT-ETSIT/U2T2-Ej2-OtrosLayouts (v3))

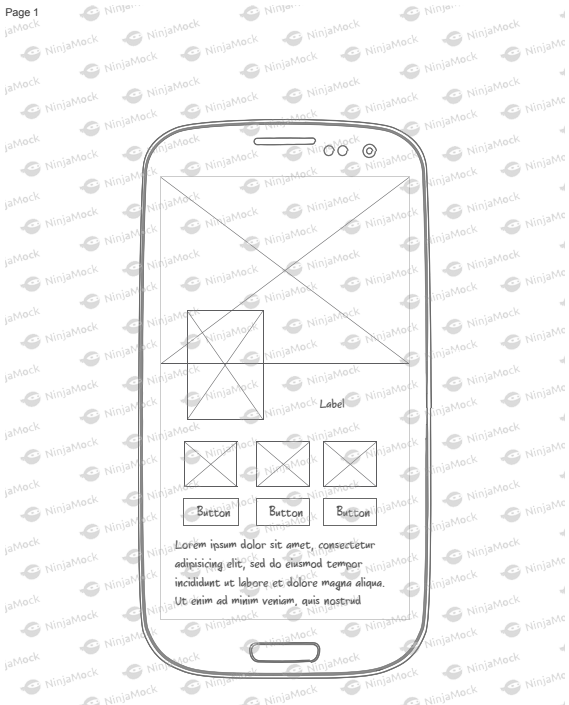
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:background="#CCCCCC" >

    <Button android:text="alignParentTop"
            android:layout_width="wrap_content" android:layout_height="wrap_cont
        android:layout_alignParentTop="true"/>

    <Button android:id="@+id/central"
            android:text="centerInParent"
            android:layout_width="wrap_content" android:layout_height="wrap_cont
        android:layout_centerInParent="true"/>

    <Button android:text="toRightOf(Central)"
            android:layout_width="wrap_content" android:layout_height="wrap_cont
        android:layout_toRightOf="@+id/central"/>

    <Button android:id="@+id/belowCentral"
            android:text="below(Central)"
            android:layout_width="wrap_content" android:layout_height="wrap_cont
        android:layout_below="@+id/central"/>

    <Button android:text="endOfBelowCentral & alignParentBottom"
            android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_toEndOf="@+id/belowCentral"
        android:layout_alignParentBottom="true"/>

</RelativeLayout>
```



android.support.constraint.ConstraintLayout

- Similar a `RelativeLayout` pero con mejor interacción gráfica en el IDE y mejor rendimiento en el terminal
- Referencia:

  `https://developer.android.com/training/constraint-layout`
- Ejemplo:
  - Conviene realizar un borrador, bosquejo, *(mockup)*, **a mano** (`http://ninjamock.com` si no se dispone de lápiz y papel)



Índice

4 Recursos
- Concepto
- Configuraciones alternativas de dispositivo
- Alias de recursos
- Estilos

Organización

Subdirectorios de `res`:

`animator/`	ficheros XML que definen animación de propiedades de objetos en el tiempo
`anim/`	ficheros XML que definen *tween animations* (punto inicial, final, rotación, ...)
`drawable/`	Ficheros de gráficos: `.png`, `.9.png`, `.jpg`, `.gif`
`layout/`	ficheros XML que definen *layouts*
`menu/`	ficheros XML que definen menús de apliación
`raw/`	ficheros en general
`values/`	ficheros XML con valores simples tales como strings, enteros, colores, ...: etiquetas <`colors`>, <`dimens`>, <`strings`>, ..., hijos de etiqueta <`resources`>. Ficheros: - `arrays.xml` - `colors.xml` - `dimens.xml` - `strings.xml` - `styles.xml`
`xml/`	ficheros XML arbitrarios
...	

Recursos drawable

- Admite ficheros de tipo bitmap: `.png`, `.9.png`, `.jpg`, `.gif`
- Ubicación: `res/drawable/imagen.[png|9.png|jpg|gif]`
- Acceso desde XML: `@[paquete:]drawable/fichero`

```
<ImageView
        android:id="@+id/ImageView01"
        android:layout_width="fill_parent" android:layout_height="fill_parent"
        android:src="@drawable/imagen" />
```

- Acceso desde Java: `R.drawable.imagen`

```
    Drawable d = (Drawable) getResources().getDrawable(R.drawable.imagen, null);
    ImageView img = (ImageView) findViewById(R.id.ImageView01);

    // Una posibilidad
    img.setImageDrawable(d);

    // O alternativamente
    img.setImageResource(R.drawable.imagen);
```

- Recursos drawable de Android:
  - Desde Java: `android.R.drawable.imagen`
  - Desde XML: `@android:drawable/imagen`
  - En la carpeta `<SDK>\platforms\android-<VERSION>\data\res\` pueden observarse todos los recursos disponibles
    En Windows, el directorio `<SDK>` suele estar ubicado en `C:\Users\usuario\AppData\Local\Android\sdk`

Ejemplo de recurso: strings

- Fichero: `res/values/fichero.xml`. El fichero suele denominarse `strings.xml`
- Strings: centralización y multilenguajes (configuraciones. alternativas)
- Acceso desde XML: `@[paquete:]string/nombre_string`

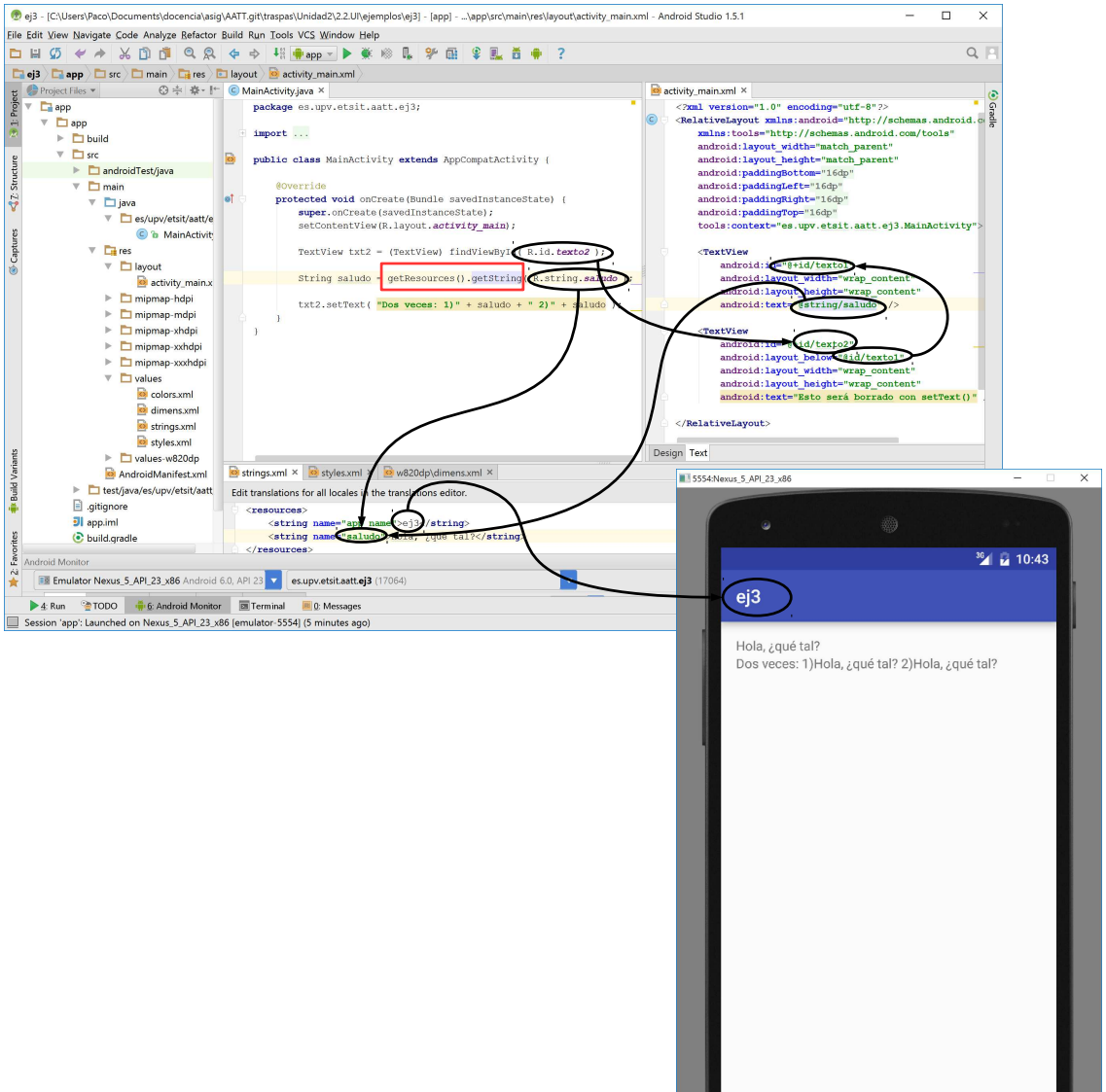
```
<TextView
    android:id="@+id/texto1"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:text="@string/saludo" />
```

- Acceso desde Java:

```
String string = getResources().getString(R.string.saludo);
String [ ] stringArray = getResources().getStringArray(R.array.laborables);
```

- Recursos drawable de Android:
  - Desde Java: `android.R.string.texto`, con `texto=cancel, copy, url, ...`
  - Desde XML: `@android:string/texto`
  - En la carpeta `<SDK>\platforms\android-<VERSION>\data\res\values\strings.xml` pueden observarse todos los recursos string disponibles
    En Windows, el directorio <SDK> suele estar ubicado en `C:\Users\usuario\AppData\Local\Android\sdk`

Ejemplo de recurso: strings (cont.)



strings.xml

```
<resources>
 <string name="app_name">ej3</string>
 <string
   name="saludo">Hola, ¿qué tal?</string>
 <string-array name="laborables">
  <item>Lunes</item>
  <item>Martes</item>
  <item>Miércoles</item>
  <item>Jueves</item>
  <item>Viernes</item>
 </string-array>
</resources>
```

activity_main.xml

```
<TextView
    android:id="@+id/texto1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/saludo" />
```

Otros recursos

- Ubicados en `res/values`
- El nombre del fichero puede ser cualquiera, pero se suelen utilizar los indicados
- Colores:
  - `int c = getResources().getColor(R.color.nombre_color, null);` // código ARGB
  - `@[paquete:]color/nombre_color`

colors.xml

```
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
</resources>
```

- Dimensiones
  - `float b = getResources().getDimension(R.dimen.dimension);` // traducción a pixels
  - `@[paquete:]dimen/dimension`

dimens.xml

```
<resources>
    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="distancia">16dp</dimen>
    <dimen name="tamanyo_fuente">20sp</dimen>
</resources>
```

Otros recursos (cont.)

- Valores lógicos
  - `boolean b = getResources().getBoolean(R.bool.nombre);`
  - `@[paquete:]bool/nombre`

bools.xml

```
<resources>
    <bool name="valor_logico">true</bool>
    <bool name="valor_logico2">false</bool>
</resources>
```

- Enteros:
  - `int i = getResources().getInteger(R.integer.nombre);`
  - `@[paquete:]integer/nombre`

integers.xml

```
<resources>
    <integer name="maximo_valor">76</integer>
    <integer name="minimo_valor">14</integer>
</resources>
```

- Array de enteros:
  - `int [] X = getResources().getIntArray(R.array.nombre);`
  - `@[paquete:]array/nombre`

integers.xml

```
<resources>
    <integer-array name="array-enteros">
       <item>76</item>
       <item>22</item>
       <item>24</item>
     </integer-array>
</resources>
```

Configuraciones alternativas de dispositivo

- Organización de la UI según orientación (*portrait* o *landscape*)
- Contenido de `TextView` en función del idioma escogido en terminal Android
- Gráficos/layouts según tamaño y densidad de pantalla
- ...

Solución: creación de distintos directorios/ficheros con distintas informaciones en función de situación/parámetros de sistema

Nombres de calificadores de configuración

MCC y MNC	Mobile Country Code y Mobile Netowrk Code `mcc214-mnc07`: Movistar, `mcc214-mnc06`: Pepephone, ...
Idioma y región	Dos letras para idioma (ISO 639-1) y opcionalmente 'r' y dos letras para región (ISO 3166-1-alpha-2) `en`, `en-rUS`, `en-rGB`, `es`, `es-rES`, `es-rMX`, ...
Dirección *layout*	Dirección de escritura `ldrtl`: *right to left*, `ldltf`: *left to right*
Anchura mínima	La menor dimensión de la pantalla (anchura o altura): `sw<N>dp` `sw320dp`, `sw600dp`, `sw720dp`, ...
Anchura disponible	`w<N>dp` `w720dp`, `w1024`
Altura disponible	`h<N>dp` `h720dp`, `h1024`
Tamaño de la pantalla	`small` (>= 426dp × 320dp): QVGA low density, VGA high density `normal` (>= 470dp × 320dp): WQVGA low density, HVGA medium density, WVGA high density `large` (>= 640dp × 480dp): VGA y WVGA medium density `xlarge` (>= 960dp × 720dp): mayores que HVGA medium density
Aspecto de la pantalla	Alargada (panorámica) o no `long`: WQVGA, WVGA, FWVGA, ... `notlong`: QVGA, HVGA, VGA, ...
Orientación	`port`: retrato *(portrait)* `land`: paisaje, apaisada *(landscape)*
Modo UI	`car`: coche `desk`: escritorio `television`: televisión `applicance`: aparato, dispositivo (sin display)
Modo noche	`night`: modo noche `nonight`: modo día
Densidad	`ldpi` (≈120 dpi), `mdpi` (≈160 dpi), `hdpi` (≈240 dpi), `xhdpi` (≈320 dpi) `nodpi` (bitmaps sin escalado), `tvdpi` (≈213 dpi)

Nombres de calificadores de configuración (cont.)

- Orden de los sufijos es importante
- Ejemplos:
  - Diferenciación de `res/layout/activity_main.xml` en modo *portrait* o *landscape*:
    - `res/layout-land/activity_main.xml`
    - `res/layout-port/activity_main.xml`
  - Diferenciación del gráfico `res/drawable/icono.png` atendiendo a idioma y orientación:
    - `res/drawable-es-rES-land/icono.png`
    - `res/drawable-es-rES-port/icono.png`
    - `res/drawable-en-rUS-land/icono.png`
    - `res/drawable-en-rUS-port/icono.png`
  - ...
- Se siguen ciertos criterios de elección en función del concepto

strings y valores simples

res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello</string>
    <string name="hi">@string/hello</string>
</resources>
```

R.string.hi es un alias de R.string.hello

res/values/colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="yellow">#f00</color>
    <color name="highlight">@color/yellow</color>
</resources>
```

R.color.highlight es un alias de R.color.yellow

Conjunto de propiedades

- Colección de propiedades que especifican la apariencia y formato de un objeto `View` o una ventana
- Ejemplo estilo (estilo de partida —opcional—: `@android:style/TextAppearance.Medium`):

res/values/miEstilo.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="estiloFuente" parent="@android:style/TextAppearance.Medium">
        <item name="android:layout_width">fill_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#00FF00</item>
        <item name="android:typeface">monospace</item>
    </style>
</resources>
```

Uso:

res/layout/main_activity.xml

```
...
<TextView
    style="@style/estiloFuente"
    android:text="@string/hello" />
...
```

Temas

- Estilo aplicado (a todas las `View` pertenecientes) a una actividad o aplicación entera.
- Indicación en fichero `AndroidManifest.xml`:
  - Tema para aplicación:

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
...
   <application android:theme="@style/estiloFuente">
...
```

  - Tema para actividad:

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
...
  <activity android:theme="@style/estiloFuente">
...
```