

# 3 – LAYOUTS y VISTAS Básicas



# Layout - Concepto

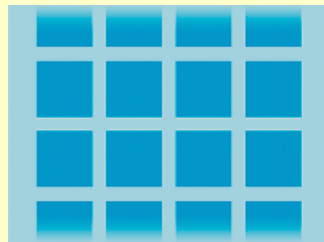
Un Layout (plano o diseño del inglés) es un elemento no visual, dedicado a controlar la distribución, posición y dimensiones de los controles que se insertan en su interior (ViewGroup)

# Layout - Concepto

El layout puedo definirlo:

- Tiempo de compilación (XML)
- Tiempo de ejecución

# Layout - Tipos



ANDROID

VMG

# Layout - Tipos

**FrameLayout:** El más sencillo. Sus hijos, estarán superpuestos en la esquina superior izquierda

**LinearLayout:** Distribuye los controles uno tras otro, en horizontal o en vertical.

Orientación `android:orientation:vertical` | **horizontal**

Atributo proporcional `android:layout_weight="1"`

# Layout - Tipos

**RelativeLayout:** Definimos la posición relativa a su contenedor o elemento hermano

**android:layout\_alignParentTop:**

pegado al padre [true, false]

**android:layout\_below:**

indico un ID del elemento del que irá a continuación

# Layout - Tipos

## RelativeLayout:

Listado completo de atributos

<https://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html>

# Layout - Tipos

**TableLayout:** Distribuimos las subvistas en filas y columnas. Cada fila se define con un elemento TableRow y en su interior, cada elemento visual

El número de columnas queda determinado por el mayor número de elementos en una fila



# Layout - Tipos

**TableLayout:** Por lo general, el ancho de un elemento viene determinado por el máximo del elemento de esa columna

**android:layout\_span:** N (combina n celdas)

**android:stretchColumns=** "\*" (todas las columnas aprovechan el máximo ancho)

# Layout - Tipos

**GridLayout:** Parecido a `TableLayout`, pero se indica el número de filas y columnas con los atributos `rowCount` y `columnCount`

**`android:orientation:`** indico si el relleno se hace vertical u horizontalmente

# Layout - Tipos

**ConstraintLayout:** Similar a RelativeLayout, pero se compone desde el editor gráfico.

Para cada View, es imprescindible definir una restricción / referencia horizontal y otra vertical

# Layout – Atributos

**android:id** Identificador (@R - int id)

**android:layout\_width:** Ancho

**android:layout\_height:** Alto

**android:gravity:** posición del contenido

**android:layout\_gravity:** posición respecto del continente

# Layout – Atributos

## MARGEN EXTERIOR (margin)

`android:layout_margin`

`android:layout_marginBottom`

`android:layout_marginTop`

`android:layout_marginLeft`

`android:layout_marginRight`

# Layout – Atributos

## MARGEN INTERIOR (padding)

`android:padding`

`android:paddingBottom`

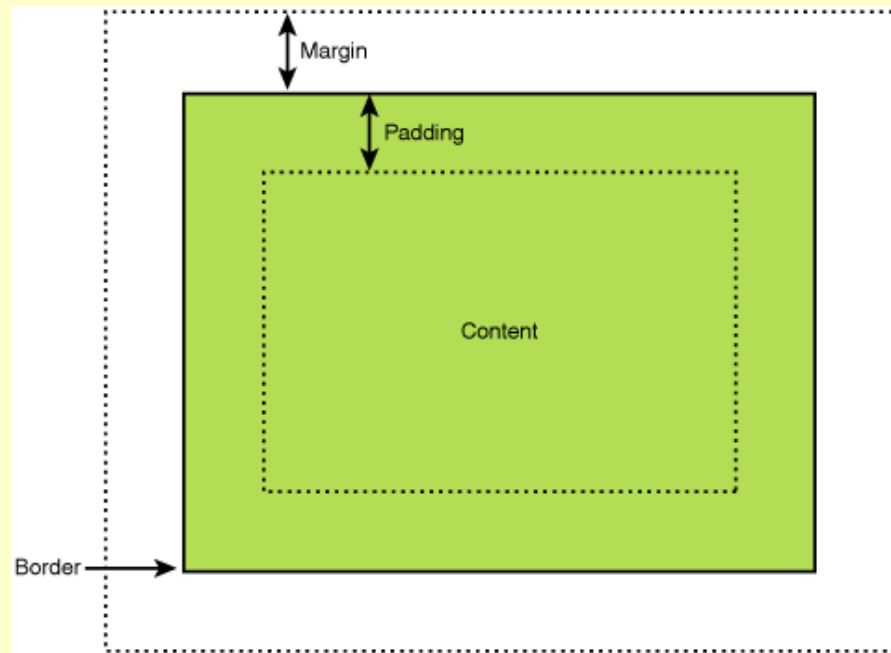
`android:paddingTop`

`android:paddingLeft`

`android:paddingRight`

# Layout – Atributos

## PADDING vs MARGIN



# Unidades absolutas

**px** Píxeles

**in** Pulgadas (25,4 mm)

**mm** Milímetros

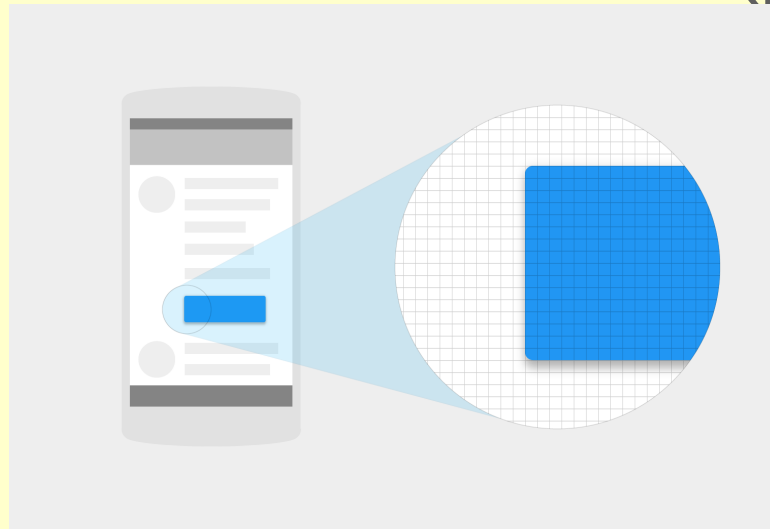
**pt** Puntos (1/72) pulgadas



# Layout – Densidad DPI

¿Cuántos pixels caben en una pulgada?

Densidad pantalla = ancho/alto (pulgadas)

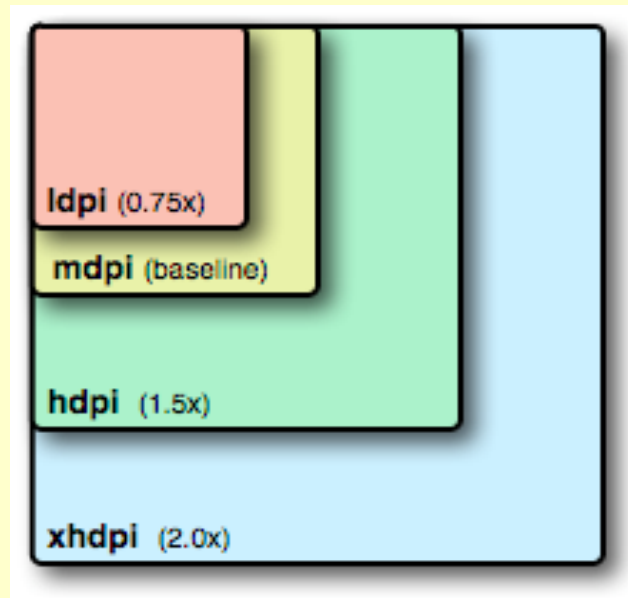


ANDROID

VMG

# Layout – Densidad DPI

La Densidad de referencia es 160 dpi (1)



# Layout – Densidad DPI

## Densidad – DPI / Categoría

0.75 - ldpi - 120 dpi /LOW - baja

1.0 - mdpi - 160 dpi /MEDIUM - media

1.5 - hdpi - 240 dpi /High - ALTA

2.0 - xhdpi - 320 dpi / Extra High- Extra Alta

3.0 - xxhdpi - 480 dpi /2 Extra

4.0 - xxxhdpi - 640 dpi /3 Extra

# Layout – Densidad DPI

```
public String obtenerDensidad (Context context)
{
    String dens = "";
    switch (context.getResources().getDisplayMetrics().densityDpi) {
        case DisplayMetrics.DENSITY_LOW: dens = "ldpi"; break;
        case DisplayMetrics.DENSITY_MEDIUM: dens = "mdpi"; break;
        case DisplayMetrics.DENSITY_HIGH: dens = "hdpi"; break;
        case DisplayMetrics.DENSITY_XHIGH: dens = "xhdpi"; break;
        case DisplayMetrics.DENSITY_XXHIGH: dens = "xxhdpi"; break;
        case DisplayMetrics.DENSITY_XXXHIGH: dens = "xxxhdpi"; }

    return densidad; }
}
```

# Unidades relativas

**DP** o DIP (density independent pixels)

Es una medida relativa al número de pixels disponibles en una pantalla.

Es el tamaño a utilizar en todos los elementos Android (menos en las letras)

# Unidades relativas

**SP** o SIP (scale independent pixels)

Idéntico al DP, pero además, se multiplica por los ajustes del tamaño de letra definido por el usuario

# Unidades relativas

**Match\_parent (fill\_parent)** Ocupa todo el espacio del contenedor

**Wrap\_content** Ocupa lo que necesita el contenido

# Enfoque DP

1 DP = 1 pixel en 160 dpi

30 DP = 30 pixels en 160 dpi

...

N DP = N Pixels en 160 dpi

¿Cuántos pixels serán 30 DP en una pantalla de 180 dpi?



# Enfoque – DP

DP - PX - DPI

30    30    160

30    ?    180

$? = (180 * 30) / 160 = 33.75$  pixels

Pixels = DPI \* DP / 160 en general

Al caber más en la pantalla, mayor es el número de pixels

# Botones

Los tipos de botones en Android son:

Button

ToggleButton

Switch

ImageButton

# Texto e Imágenes

Los tipos principales en Android son:

EditText

TextView

ImageView

# Selectores

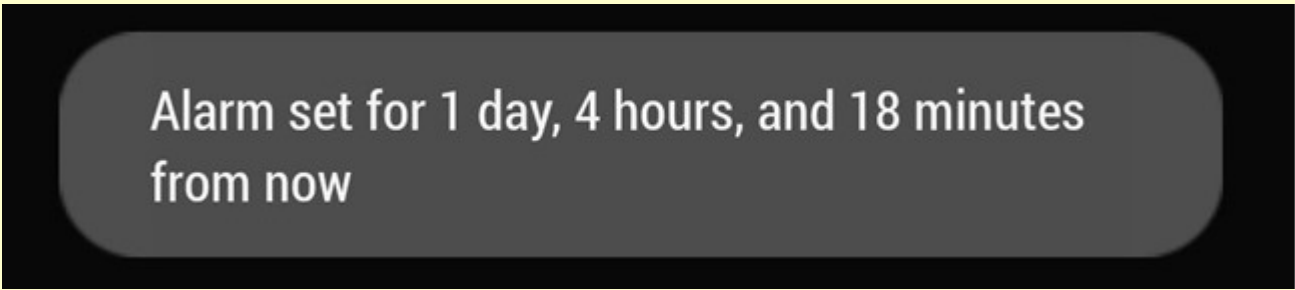
Para representar opciones alternativas:

Checkbox (una opción)

Radiobutton (múltiples opciones)

# Toast

Toast es un texto en la pantalla, que aparece durante un periodo pequeño de tiempo, para informar al usuario



Alarm set for 1 day, 4 hours, and 18 minutes  
from now

# Toast

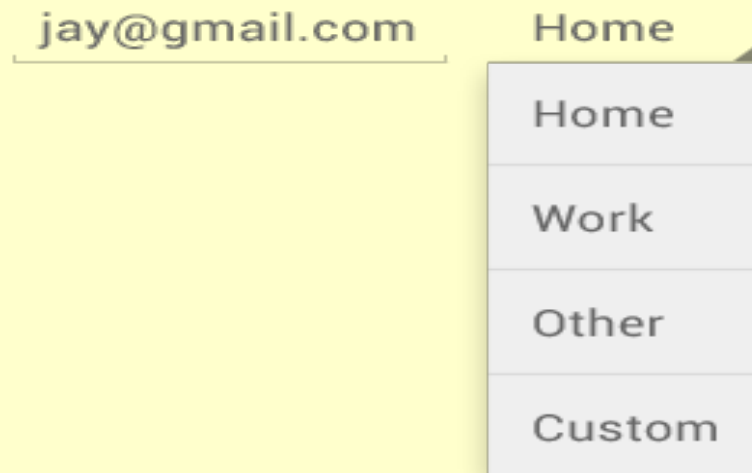
Gravity → Posición relativa

Show() → Mostrar

MakeText() → Para construir el mensaje

# Spinner

Es otra clase que me permite seleccionar entre varios elementos



# Spinner

Los datos, pueden estar predefinidos de forma estática (en tiempo de compilación) o bien, indicandando un Adapter al Spinner (que hace de AdapterView)



# Spinner

También admite un listener, para detectar cuando un elemento del spinner es seleccionado

```
Class A implements OnItemSelectedListener {  
    public void onItemSelected(AdapterView<?> parent, View  
view,    int pos, long id)  
    public void onNothingSelected(AdapterView<?> parent) {  
        // Another interface callback  
    }  
}
```

# TypedArray

Permite crear un Array a partir de valores definidos en un xml de recursos

Después de su uso, hay que invocar al método `recycle` (que evita recrear el array innecesariamente, optimizando el uso de la memoria)

# View.setTag()

El método `setTag()` merece mención especial por su generosidad y potencia de uso.

En general, permite asociar información lógica al elemento físico (Vista) y emplearla cuando sea necesario

# Vistas dinámicas

Para construir vistas en tiempo de ejecución, Android me permite usar un AdapterView.

Un AdapterView usará un Adapter obtener los datos

`AdapterView.setAdapter (Adapter)`

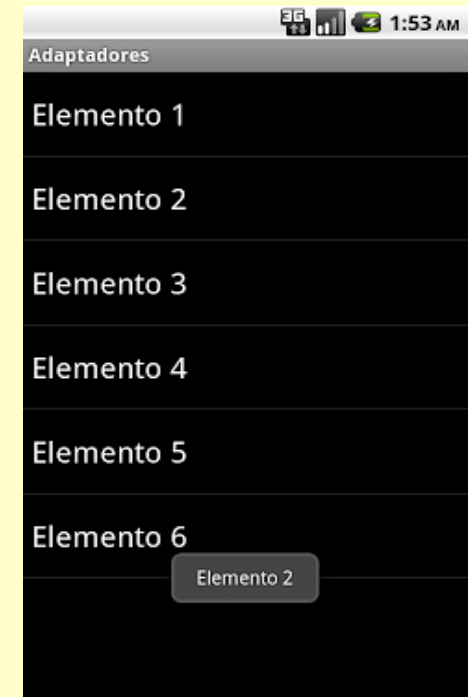
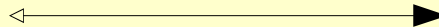
# Adapter

Se puede entender como un intermediario entre una estructura de datos y su representación visual. Al `adapterView` le ofrece:

- Acceso a los datos
- Las vistas infladas

# Adapter

```
String[] array = new  
String[] {  
    "Elemento 1"  
    ,"Elemento 2"  
    ,"Elemento 3"  
    ,"Elemento 4"  
    ,"Elemento 5"  
    ,"Elemento 6"  
};
```



# AdapterView

Un AdapterView será el ViewGroup en cuyo interior se vayan inflando los elementos hijos. Hay muchos tipos de AdapterView. Nos centraremos en dos básicos

ListView

GridView

# ListView

Un ListView es una ViewGroup específica para dibujar listas. Definiremos el ListView en la actividad donde queramos dibujar la lista

```
<ListView
```

```
    android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content"
```

```
    android:id="@+id/lista"/>
```



# GridView

Una GridView es una ViewGroup específica para tablas sobre la marcha.

```
<GridView
```

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

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
```

```
/>
```

# Listener - Item

Para los AdapterView, puedo definir un listener, que sea invocado cuando un elemento se seleccione:

```
AdapterView.setOnItemClickListener  
(new AdapterView.OnItemClickListener()  
    {  
        public void onItemClick(AdapterView<?> parent,  
View v, int position, long id) {  
            Log.d ("DEBUG ", position + " seleccionado");  
        }  
    }  
);
```