Layouts (1)

El *layout* o diseño establece la forma en que se organizan los diferentes componentes de una interfaz.

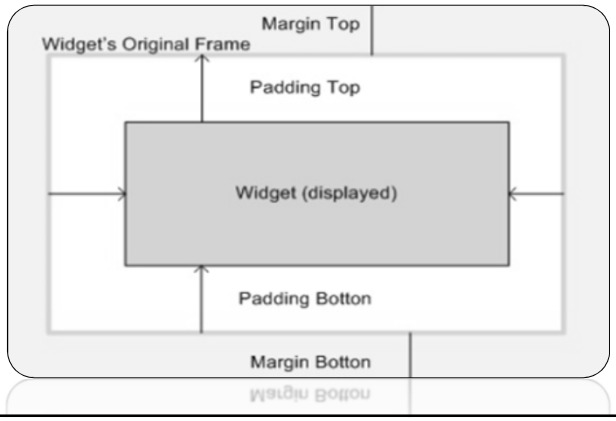
Los *layout* de Android se especifican mediante archivos XML en los que se establecen de manera jerárquica los elementos (contenedores o *widgets*) que se quieren mostrar en pantalla.

Los archivos de *layout* se almacenan en la carpeta **res/layout** del proyecto Android.

13

Layouts (2)

- **Contenedores**: pueden ser visuales o no y sirven para almacenar y organizar otros contenedores o *widgets*.

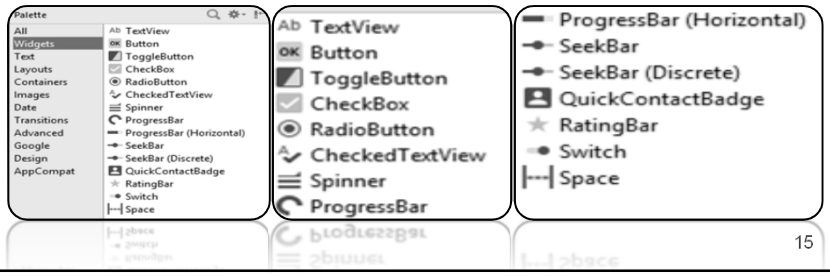


14

Layouts (3)

- ***Widgets***: son cada uno de los elementos que podemos añadir a nuestra interfaz, y casi todos son visuales.

  No hay que confundir los *widgets* de interfaz (UI) con pequeñas aplicaciones Android también denominadas widget.



15

Layouts (4)

La forma en que se comportan y muestran contenedores y *widgets* viene dado por sus atributos que también estarán definidos en el archivo XML.

```
android:atributo="valor"
```

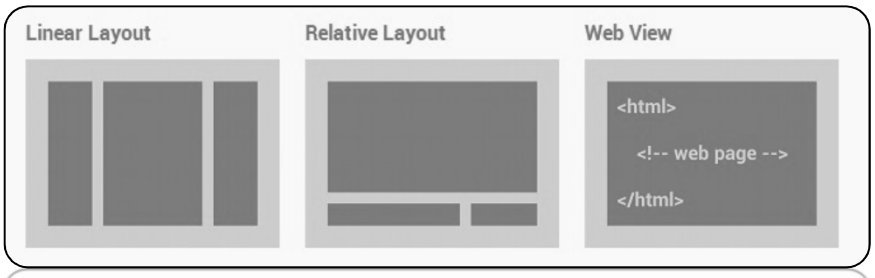
Cada tipo de elemento tendrá sus propios atributos, siendo algunos obligatorios y otros opcionales.

16

Layouts (5)

Tipos de Layout:

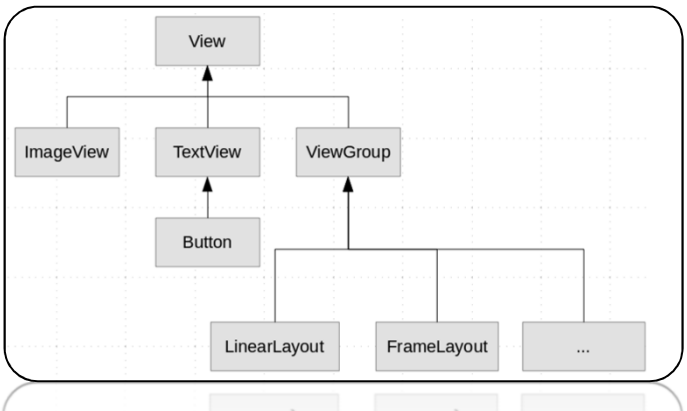
- LinearLayout
- TableLayout
- AbsoluteLayout
- RelativeLayout
- FrameLayout
- GridLayout



17

Layouts (6)

Todos los *layout* de Android heredan de la clase ***ViewGroup***, que a su vez hereda de la clase ***View***.

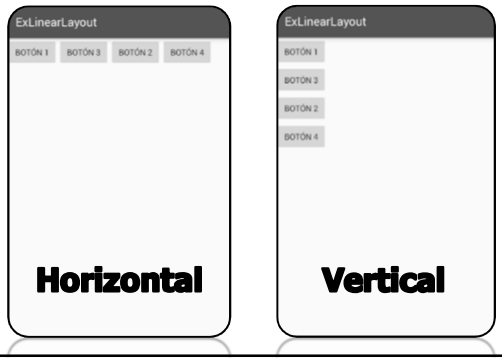


18

Layouts (7)

- ***LinearLayout***

Coloca los widgets uno detrás de otro según se declaren en el XML, pudiendo ser colocados vertical u horizontalmente.



19

Layouts (8)

- ***LinearLayout***

Orientación:

- **orientation**: especifica la orientación del *layout*: horizontal o vertical

Posicionamiento de elementos:

- **layout_gravity**: establece la posición de un elemento en su contenedor: left, right...
- **gravity**: establece la posición del contenido de un elemento: left, right…

20

Layouts (9)

- ***LinearLayout***

Peso de un elemento:

- **weightSum**: esta propiedad del *layout* establece un peso total para el conjunto de los elementos que comparten espacio horizontal o vertical.
- **weight**: permite establecer el espacio vacío que ocupará un elemento, por defecto 0. Con el valor "*match_content*" los pesos funcionan al revés.

21

Layouts (10)

- ***TableLayout***

Coloca los elementos en disposición de filas y columnas, para lo que suele tener como hijos elementos de tipo *TableRow*.

22

Layouts (11)

- ***TableLayout***

Cada *TableRow* representa una fila de la tabla y cada elemento dentro del *TableRow* una celda de esa fila.

El efecto es similar al de un *LinearLayout* vertical con varios *LinearLayout* horizontales en su interior.

Permite crear de forma muy rápida diseños sencillos que se ajustan muy bien al girar la pantalla.

23

Layouts (12)

- ***TableLayout***

En el *TableLayout* la propiedad *orientation* carece de sentido, ya que la distribución es en forma de tabla.

Las dimensiones de los *TableRow* que se sitúen dentro de un *TableLayout* van a ser siempre *match_parent* para el ancho y *wrap_content* para el alto.

24

Layouts (13)

- ***TableLayout***

Estirar y encoger columnas:

Podemos establecer las columnas que se podrán estirar o encoger para que el conjunto se ajuste al ancho total.

Para ello las columnas se numeran partiendo de 0, y para especificar todas se utiliza *****.

- **stretchColumns**: estirar.
- **shrinkColumns**: encoger.

25

Layouts (14)

- ***TableLayout***

Estirar y encoger columnas:

android:stretchColumns="*"

android:shrinkColumns="0,2"

Se pueden aplicar las dos propiedades a la misma columna de forma simultánea para que se estire o encoja según convenga.

26

Layouts (15)

- ***TableLayout***

Estirar celdas:

- **layout_span**: permite estirar el ancho de un elemento para que ocupe varias celdas.

```
android:layout_span="2"
```

27

Layouts (16)

- ***TableLayout***

Colocar celdas:

- **layout_column**: permite especificar en qué columna se desea colocar un elemento en lugar de la que le corresponde.

```
android:layout_column="1"
```

28

Layouts (17)

- ***AbsoluteLayout***

La posición de los elementos se establece directamente mediante sus coordenadas: *layout_x* y *layout_y*.

29

Layouts (18)

- ***AbsoluteLayout***

La creación de este *layout* es muy sencilla. Basta con seleccionar los elementos en la vista de diseño y arrastrarlos a la posición de la pantalla que deseemos.

El principal inconveniente es que al establecer la posición de los elementos mediante sus coordenadas, la vista variará en función del tamaño de la pantalla y de la orientación.

30

Layouts (19)

- ***AbsoluteLayout***

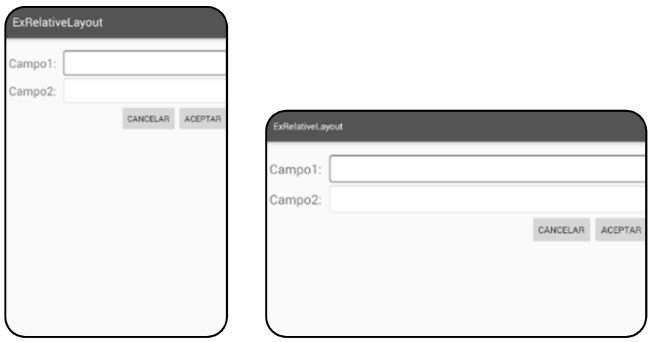
Dados sus problemas, este tipo de diseño esta obsoleto (*deprecated*) y se recomienda no utilizarlo.

31

Layouts (20)

- ***RelativeLayout***

La posición de los elementos se establece en función de otros, ya sea su contenedor u otro elemento de la vista.



32

Layouts (21)

- ***RelativeLayout***

Dado que la posición se establece en relación a otros elementos es muy importante que los elementos tengan un identificador para poder referirse a ellos.

Cuando se maneja con soltura permite realizar diseños que serían muy complicados mediante *LinearLayout*.

33

Layouts (22)

- ***RelativeLayout***

Posicionamiento relativo al contenedor:

Podemos posicionar un elemento en función de los cuatro bordes del contenedor (arriba, abajo, izquierda y derecha) estableciendo el valor *true* o *false* en uno o varios de los siguientes atributos:

- **layout_alignParentTop**: alinear el borde superior del elemento con el borde superior del contenedor.

34

Layouts (23)

- ***RelativeLayout***

Posicionamiento relativo al contenedor:

- **layout_alignParentBottom**: alinear el borde inferior del elemento con el borde inferior del contenedor.
- **layout_alignParentLeft**: alinear el borde izquierdo del elemento con el borde izquierdo del contenedor.
- **layout_alignParentRight**: alinear el borde derecho del elemento con el borde derecho del contenedor.

35

Layouts (24)

- ***RelativeLayout***

Posicionamiento relativo al contenedor:

- **layout_alignParentStart**: alinear el comienzo del elemento con el comienzo del contenedor.
- **layout_alignParentEnd**: alinear el final del elemento con el final del contenedor.

```
android:layout_alignParentLeft="true"
```

36

Layouts (25)

- ***RelativeLayout***

Centrar elementos:

- **layout_centerHorizontal**: centra el elemento en el contenedor horizontal-mente.
- **layout_centerVertical**: centra el elemento en el contenedor verticalmente.
- **layout_centerInParent**: centra el elemento en el contenedor tanto horizontal como verticalmente.

37

Layouts (26)

- ***RelativeLayout***

Posicionamiento relativo a otros elementos:

Podemos posicionar un elemento en función de otro estableciendo en las siguientes propiedades el ***id*** del elemento de referencia Hemos de utilizar **@+id** si el elemento de referencia aún no se ha declarado.

- **layout_above**: sitúa el borde inferior del elemento sobre el elemento de referencia.
- **layout_below**: sitúa el borde superior del elemento bajo el elemento de referencia. 38

Layouts (27)

- ***RelativeLayout***

Posicionamiento relativo a otros elementos:

- **layout_ toLeftOf** : sitúa el borde derecho del elemento a la izquierda del elemento de referencia.
- **layout_toRightOf**: sitúa el borde izquierdo del elemento a la derecha del elemento de referencia.
- **layout_alignBaseLine**: la línea de base del elemento coincide con la del elemento de referencia.

39

Layouts (28)

- ***RelativeLayout***

Posicionamiento relativo a otros elementos:

- **layout_alignTop**: el borde superior del elemento coincide con el de referencia.
- **layout_alignBottom**: el borde inferior del elemento coincide con el de referencia.
- **layout_alignLeft**: el borde izquierdo del elemento coincide con el de referencia.
- **layout_alignRight**: el borde derecho del elemento coincide con el de referencia.

40

Layouts (29)

- ***RelativeLayout***

Posicionamiento relativo a otros elementos:

Si el elemento de referencia ya ha sido creado anteriormente:

```
android:layout_below="@id/ele1"
```

Si el elemento de referencia aún no ha sido creado:

```
android:layout_below="@+id/ele1"
```

41

Layouts (30)

- ***FrameLayout***

Este *layout* se comporta como una pila de elementos donde cada nuevo elemento se sitúa encima de los anteriores.

42

Layouts (31)

- ***FrameLayout***

En la imagen anterior puede verse como el primer botón "Aceptar y Enviar" ha sido parcialmente ocultado por el segundo "Borrar".

Si el primer botón hubiera tenido un tamaño inferior al del segundo habría sido ocultado por completo.

43

Layouts (32)

- ***FrameLayout***

FrameLayout esta diseñado para bloquear un área de la pantalla para mostrar un único elemento.

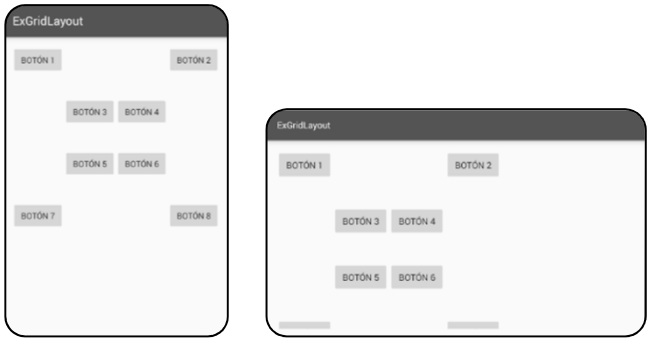
Aunque se permiten varios hijos cuya posición puede ser controlada mediante *layout_gravity*, resulta muy complicado realizar este tipo de diseños y que sean escalables sin que los elementos se superpongan entre sí.

44

Layouts (33)

- ***GridLayout***

Este *layout* muestra los elementos colocados en las celdas de una cuadrícula o matriz de dos dimensiones.



45

Layouts (34)

- ***GridLayout***

La cuadrícula está formada por un conjunto de líneas infinito que separan el área de visualización en celdas referenciadas por los índices que ocupan: fila (*layout_row*) y columna (*layout_column*).

El número de filas y columnas viene dado por el valor de los atributos *rowCount* y *columnCount* del elemento *GridLayout*. Si no se establecen vendrá dado por el índice más alto utilizado entre los elementos.

46

Layouts (35)

- ***GridLayout***

Podemos establecer que un elemento ocupe varias filas (*layout_rowSpan*) o columnas (*layout_Columnspan*), pero se requiere el valor *fill_vertical* o *fill_horizontal* en la propiedad *layout_gravity*.

```
<Button
    android:id="@+id/bot9" android:text="Botón 9"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="0"
    android:layout_column="1"
    android:layout_gravity="fill_horizontal"
    android:layout_columnSpan="2"/>
```

Layouts (36)

- ***GridLayout***

Para aumentar la separación entre los elementos se pueden utilizar los atributos *margin* o *padding*, o añadir elementos de separación *<space>* en las celdas intermedias.

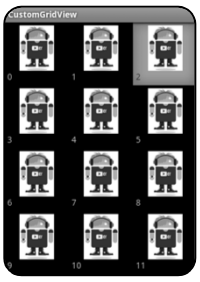
```
<Space
    android:layout_width="90dp"
    android:layout_height="40dp"
    android:layout_row="0"
    android:layout_column="1"/>
```

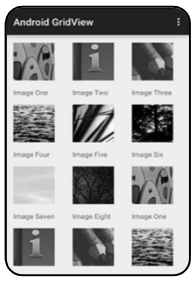
48

Layouts (37)

- ***GridLayout***

No debemos confundir *GridLayout* que es un diseño de pantalla, con *GridView* que es un *ViewGroup* que muestra los elementos del *ListAdapter* asociado.



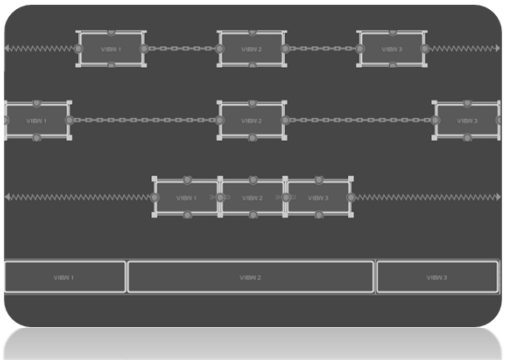


49

Layouts (38)

- ***ConstraintLayout***

Este *layout* establece la posición de los elementos en base a restricciones de posición, dimensiones, visibilidad y margen.



50

Layouts (39)

- ***ConstraintLayout***

Este *layout* está concebido para trabajar en modo de diseño pero evitando los problemas del posicionamiento absoluto, para ello en lugar de utilizar coordenadas utiliza una serie de restricciones en cuanto a las dimensiones de los elementos, posicionamiento relativo a otros elementos, centrado, márgenes, etc.

Se puede utilizar casi en cualquier versión de Android (nivel API mínimo 9).

51

Layouts (40)

- ***ScrollView***

Permite a un contenedor disponer de *scroll* y por tanto, superar el tamaño de la pantalla.

Para crear un diseño con *scroll* hemos de crear un elemento *ScrollView* y en su interior colocar el *layout* al que queremos añadir el *scroll*.

No puede haber más de un hijo dentro del *ScrollView*.

52