Tecnologías y Desarrollo en Dispositivos Móviles

Apartado 6: Layouts

Autores:

Víctor M. Rivas Santos / Miguel Á. García Cumbreras (Antonio Rueda Ruiz)

Definición

- Los layout permiten definir la estructura de la interfaz de usuario de la app.
- Suelen declararse en un fichero XML
 - También pueden construirse en tiempo de ejecución mediante código (=programáticamente)

Tipos

Layouts

ConstraintLayout

I---I Guideline (horizontal)

Guideline (vertical)

LinearLayout (horizontal)

LinearLayout (vertical)

FrameLayout

TableLayout

TableRow

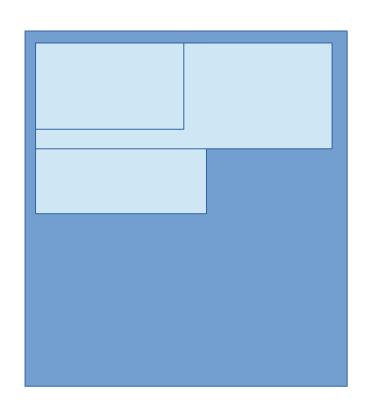
I---I Space

Legacy

■ GridLayout

□ ListView
□ TabHost
■ RelativeLayout
■ GridView

Frame Layout



Coloca los controles hijos alineados en la esquina superior izquierda, dejando cada control oculto por el siguiente control.

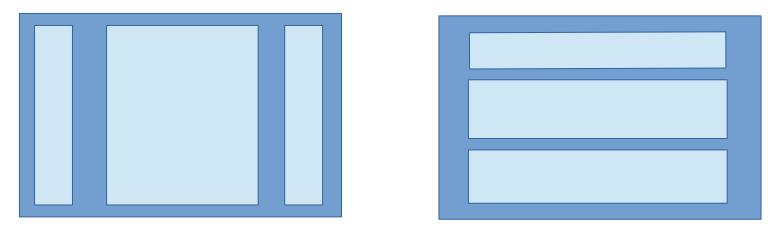
Frame Layout: características

Un layout muy simple que superpone todos los widgets

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"</p>
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:id="@+id/activity main"
  android:layout width="match parent"
  android:layout height="match parent"
  tools:context="com.example.vrivas.framelayout imagenes.MainActivity">
  <lmageView android:layout width="wrap content"</pre>
    android:layout height="wrap content"
    app:srcCompat="@drawable/victor"
    android:id="@+id/imageView" />
  <lmageView app:srcCompat="@drawable/verruga"</pre>
    android:id="@+id/imageView3"
    android:layout_marginTop="160dp"
    android:layout_marginLeft="210dp"
    android:layout_width="59dp"
    android:layout_height="41dp" /> <!-- Control de tamaño y posición -->
  <lmageView android:layout width="wrap content"</pre>
    android:layout height="wrap content"
    app:srcCompat="@drawable/victor2"
    android:id="@+id/imageView4" />
</FrameLayout>
```

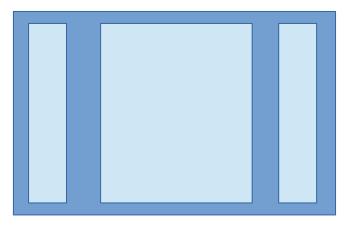


Linear Layout



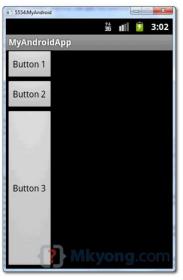
Posiciona los componentes visuales uno junto al otro, según la propiedad android:orientation

Linear Layout









Linear Layout: características

Posiciona los elementos uno a continuación de otro

Existen versiones horizontal y vertical controladas por el

LocalidadesCercanas

Boton 2

atributo orientation

```
<LinearLayout
 android:layout width="match parent"
 android:layout height="match parent"
 android:orientation="vertical">
 <Button
  android:layout width="wrap content"
  android:layout height="wrap content"
  android:text="Botón 1"/>
 <Button
   android:layout width="wrap content"
   android:layout height="wrap content"
   android:text="Botón 2"/>
</LinearLayout>
```

Linear Layout: dimensiones

- Los atributos width y height controlan el ajuste de las dimensiones del layout:
 - wrap_content → ajustar al contenido del interior
 - match_parent → expandir al layout superior que lo contiene



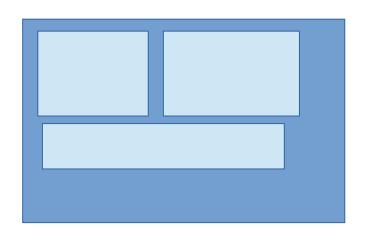


Linear Layout: gravedad

 El atributo gravity define la forma en que se agrupan los elementos en el interior del layout



Relative Layout



Permite especificar la posición de cada elemento de forma relativa a cualquier otro elemento incluido en el propio layout.

Relative Layout: características

 Organiza los widgets estableciendo posiciones relativas al layout que los contiene y a otros widgets

Complejo: pensado para ser usado mediante

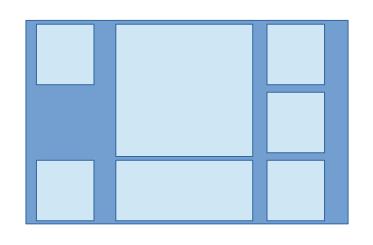
diseñadores

```
<RelativeLayout
       android:layout width="match parent"
       android:layout height="match parent"
       android:orientation="vertical"
       android:id="@+id/layout"
       android:baselineAligned="true">
       <TextView
               android:layout width="wrap content"
               android:layout height="wrap content"
               android:textAppearance="?android:attr/textAppearanceMedium"
               android:text="Nombre:"
               android:id="@+id/textView"
               android:layout alignParentTop="true"
               android:layout alignParentLeft="true" />
       <EditText
               android:layout width="fill parent"
               android:layout height="wrap content"
               android:id="@+id/editText"
               android:layout below="@+id/textView"
               android:layout alignParentLeft="true" />
```

── LocalidadesCercanas

Nombre:

Table Layout



Agrupa componentes en filas y columnas (TableRow).

Table Layout: características

- Organiza los *widgets* en forma de tabla (similar a tablas HTML)
- Cada fila contiene varios *widgets* que son asignados a diferentes columnas
- Las columnas pueden expandirse o estrecharse y unirse entre ellas

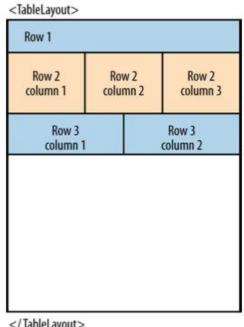
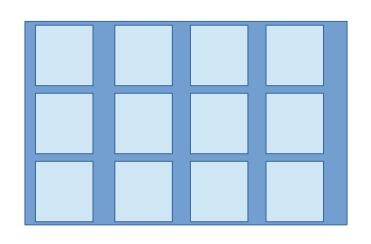


Table Layout: código

```
<TableLayout
       android:stretchColumns="1"> <!-- La columna 1 se expande hasta el final -->
       <TableRow>
              <TextView android:text="Usuario"/> <!-- Columna 0 -->
              <EditText
                     android:id="@+id/etUsuario"
                     android:inputType="text" /> <!-- Columna 1 -->
       </TableRow>
       <TableRow>
              <TextView android:text="Clave"/> <!-- Columna 0 -->
              <EditText
                     android:inputType="textPassword"
                     android:ems="10"
                     android:id="@+id/etClave" /><!-- Columna 1 -->
       </TableRow>
       <TableRow
              android:paddingTop="25dp">
              <Button
                     android:text="Validar"
                     android:id="@+id/bValidar"
                     android:layout span="1"
                     android:layout gravity="center horizontal" />
              <Button
                     android:text="Cncelar"
                     android:id="@+id/bCancelar"
                     android:layout_span="1"
                     android:layout_gravity="center_horizontal" />
                  <!--Centrar botón en columna-->
       </TableRow>
</TableLayout>
```



Grid Layout



Distribuye los elementos de forma tabular, en filas y columnas (rowCount y columnCount).

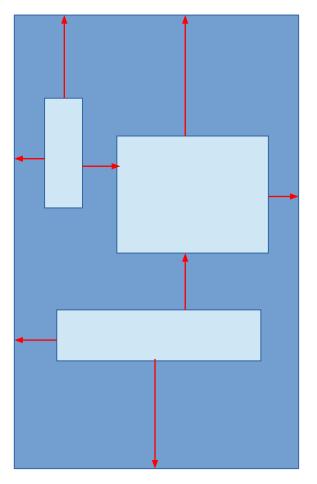
Grid Layout: características

- Similar a TableLayout aunque no requiere especificar fila por fila
- columnCount y rowCount indican el tamaño de la rejilla

```
<GridLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
       android:layout width="match parent"
       android:layout_height="match_parent"
       android:columnCount="2"
       android:orientation="horizontal">
       <TextView android:text="Usuario"/>
       <EditText
             android:id="@+id/etUsuario"
             android:inputType="text"
             android:layout gravity="fill horizontal" />
       <TextView android:text="Clave"/>
       <EditText
             android:id="@+id/etClave"
             android:inputType="textPassword"
             android:layout gravity="fill horizontal" />
       <Button
             android:text="Validar"
             android:id="@+id/bValidar"
             android:layout_gravity="center_horizontal"
             android:layout columnSpan="2"
             android:layout marginTop="25dp" />
              <!--Unir las dos columnas, centrar y dar
             un poco de espacio con la fila de arriba-->
</GridLayout>
```

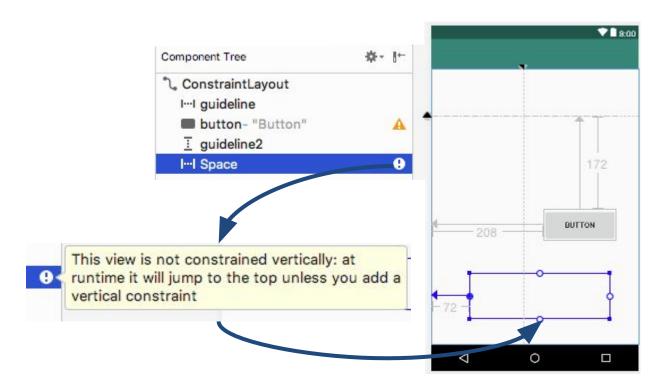


Constraint Layout (2016)

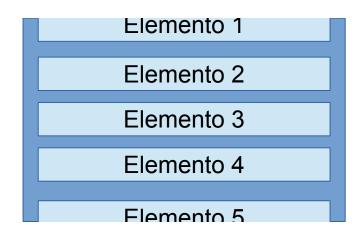


Similar a
RelativeLayout, nos
permite establecer
relaciones entre
todos los
elementos y la vista
padre.

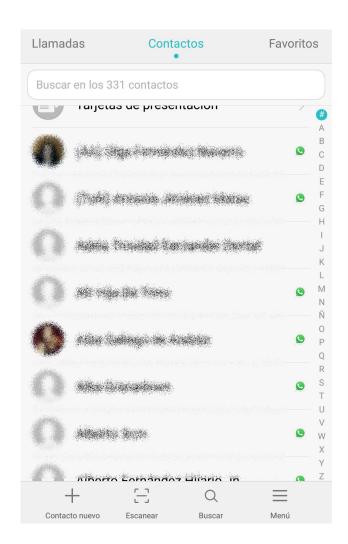
Constraint Layout: helpers



List View



Grupo de vistas con elementos desplazables



EJERCICIONAI)

Realiza el diseño de apps para jugar al tres en raya (tic-tac-toe)

MTV: Trabajar con los distintos layouts

- Utilizando los layouts que prefieras o consideres más adecuados, realiza dos apps distintas para jugar al tres en raya (tic-tac-toe)
 - Cada una de las apps debe usar layouts distinto.
 - Ten en cuenta que no es necesario que se pueda jugar... solo necesitamos la interfaz, esto es, *layout* y widgets que se usarían.
- Sube un breve documento en PDF con capturas de pantalla de tus aplicaciones, explicando brevemente qué layouts y widgets has usado en cada una de ellas.