FEUD

Fundación de egresados U. Distrital

Construyendo Profesionales

Somos el Centro de Entrenamiento Autorizado por marcas representativas en Gobierno TI y empresa, con el portafolio más amplio en Latinoamérica:

FEUD
Fundación de egresados U. Distrital
Construyendo Profesionales

Desarrollo de aplicaciones utilizando SDK - Android

Diplomado Desarrollo de Software para móviles

Julio César Galeano García

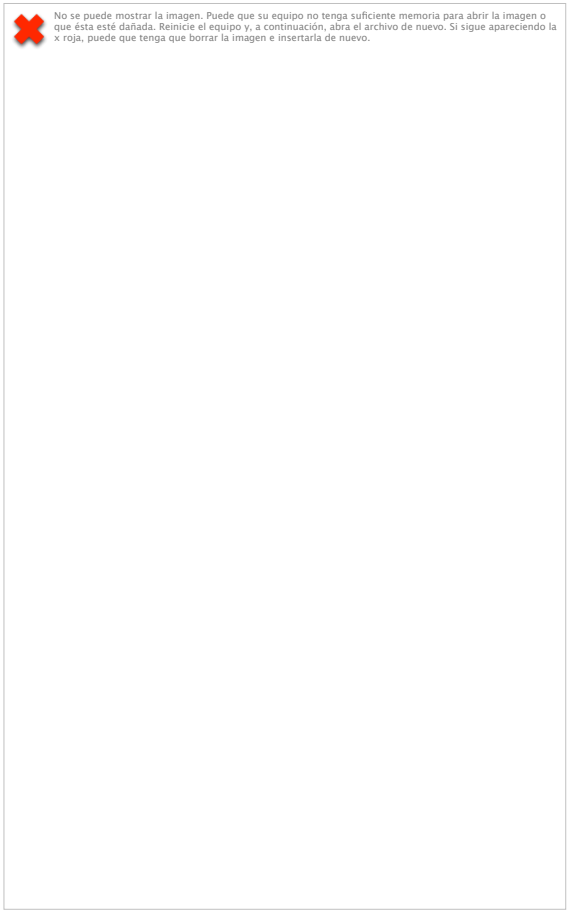
2017

FEUD
Fundación de egresados U. Distrital
Construyendo Profesionales

Android: Actividades

Una actividad puede verse de manera simple como una pantalla que interactúa con el usuario.

Una Activity es un componente de la aplicación que contiene una pantalla con la que los usuarios pueden interactuar para realizar una acción, como marcar un número telefónico, tomar una foto, enviar un correo electrónico o ver un mapa. A cada actividad se le asigna una ventana en la que se puede dibujar su interfaz de usuario. La ventana generalmente abarca toda la pantalla, pero en ocasiones puede ser más pequeña que esta y quedar "flotando" encima de otras ventanas.



Actividades

- Una aplicación puede estar compuesta por varias actividades.
- Debe ser iniciada por un Intent.
  - `Intent i = new Intent(this, ActivityTwo.class);`
  - `startActivity(i);`
- Las actividades comunican datos a traves de los Bundles adjuntados al Intent.

Actividades: Intents

Una Intent es un objeto de acción que puedes usar para solicitar una acción de otro componente de la aplicación. Aunque las intents facilitan la comunicación entre los componentes de muchas maneras, existen tres casos de uso fundamentales:

- Para comenzar una actividad
- Para iniciar un servicio
- Para entregar un mensaje

Actividades: Intents

Existen dos tipos de intents

- **Intents explícitas:** especifican qué componente se debe iniciar mediante su nombre (el nombre de clase completamente calificado).

- **Intents implícitas:** no se nombra el componente específicamente; pero, en cambio, se declara una acción general para realizar, lo que permite que un componente de otra aplicación la maneje. Por ejemplo, por medio de una acción.

Actividades: Usando Bundles

```
//Desde Activity1 (llama a segunda actividad)

Intent intent = new Intent(this,myActivity2.class);
Bundle bundle = new Bundle();
bundle.putString("myValue", myValue);
intent.putExtras(bundle);
navigation.this.startActivity(intent);

//In Activity2 (ejecutada por Actividad1)

Bundle bundle = getIntent().getExtras();
act2MyValue= bundle.getString(“myValue“);
```

Layout

Layout define la estructura visual para una interfaz de usuario, como la IU para una actividad o widget de una app. Puedes declarar un diseño de dos maneras:

- Declarar elementos de la IU en XML.

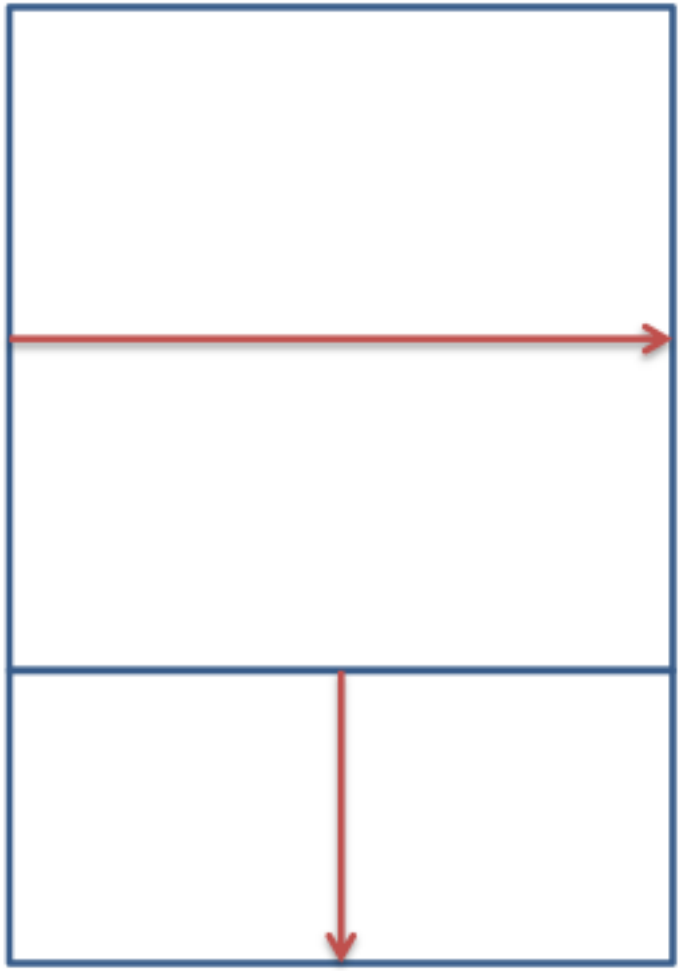
- Crear una instancia de elementos del diseño en tiempo de ejecución. Tu aplicación puede crear objetos View y ViewGroup (y manipular sus propiedades) programáticamente.

Tipos más recientes de Layout

- Linear Layout
- Relative Layout
- Table Layout
- Frame layout
- Constraint Layout

Linear Layout

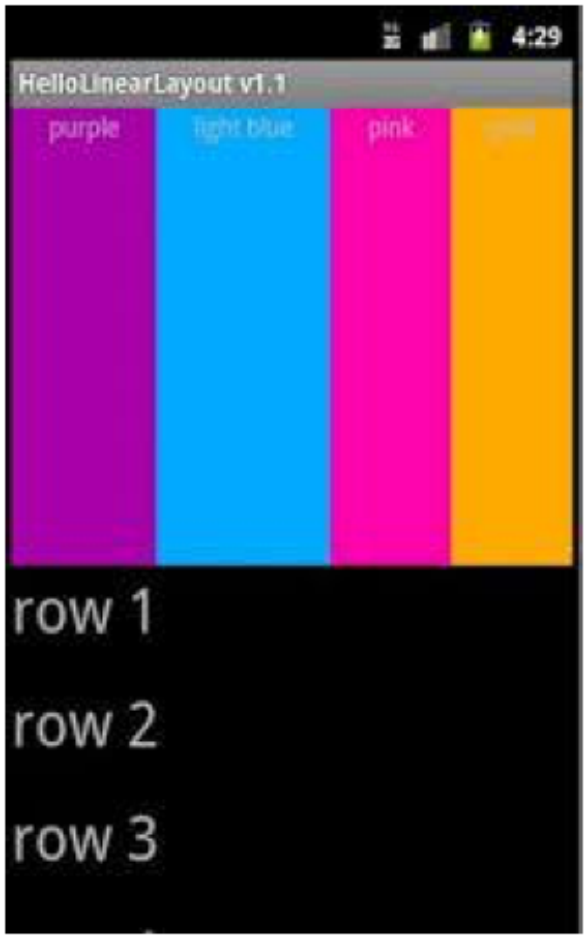
Es uno de los Layout más utilizado en la práctica. Distribuye los elementos uno detrás de otro, bien de forma horizontal o vertical.



Linear Layout

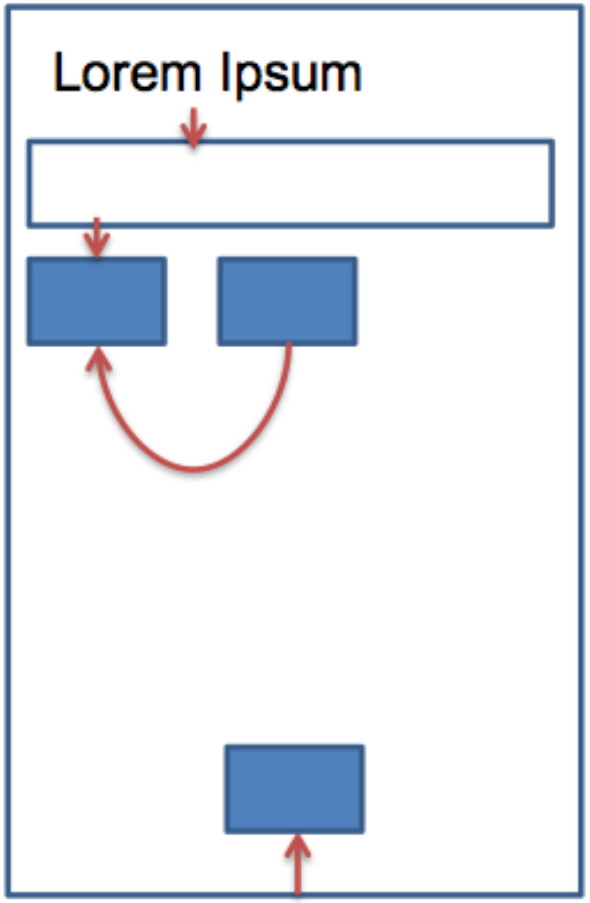
```
<?xml version="1.0" encoding="utf-8"?>
<!-- Orientacion Horizontal -->
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

<!-- Orientacion Vertical -->
  <LinearLayout android:layout_width="fill_parent"
                android:layout_height="wrap_content"
          android:orientation="horizontal"
android:background="#2a2a2a"
          android:layout_marginTop="25dip"
  </LinearLayout>
</LinearLayout>
```



Relative Layout

Permite comenzar a situar los elementos en cualquiera de los cuatro lados del contenedor e ir añadiendo nuevos elementos pegados a estos.



Relative Layout

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

   <TextView android:id="@+id/label" android:layout_width="fill_parent"
          android:layout_height="wrap_content" android:text="@string/email" />

   <EditText android:id="@+id/inputEmail" android:layout_width="fill_parent"
          android:layout_height="wrap_content" android:layout_below="@id/label" />

   <Button android:id="@+id/btnLogin" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:layout_below="@id/inputEmail"
        android:layout_alignParentLeft="true" android:layout_marginRight="10px"
        android:text="@string/login" />

   <Button android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_toRightOf="@id/btnLogin"
        android:layout_alignTop="@id/btnLogin"  android:text="@string/cancel" />

   <Button android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_alignParentBottom="true" android:text="@string/register"
        android:layout_centerHorizontal="true"/>
</RelativeLayout>
```

Table Layout

Distribuye los elementos de forma tabular. Se utiliza la etiqueta <TableRow> cada vez que queremos insertar una nueva línea.

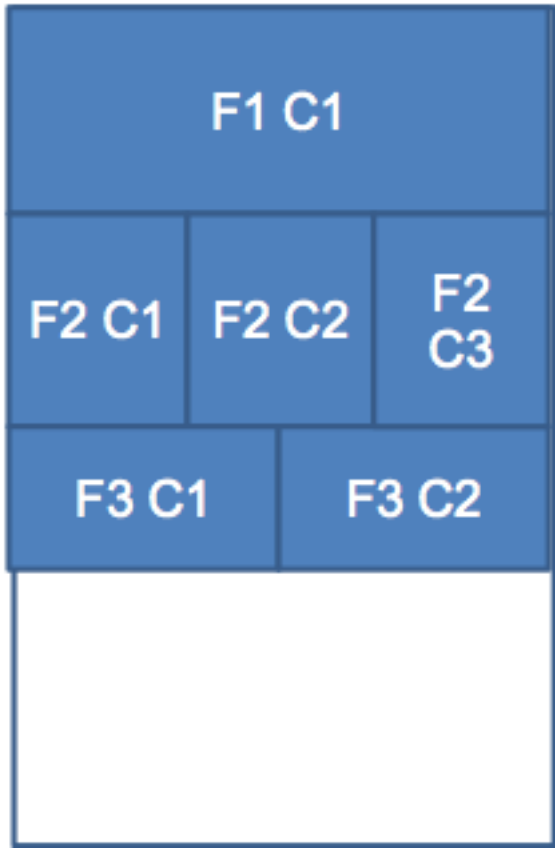


Table Layout

```
<TableLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:shrinkColumns="*"  android:stretchColumns="*"
android:background="#ffffff">
   <!-- Fila 1 con una sola columna -->
   <TableRow
      android:layout_height="wrap_content"
      android:layout_width="fill_parent"
      android:gravity="center_horizontal">
      <TextView
         android:layout_width="match_parent"
android:layout_height="wrap_content"
         android:textSize="18dp" android:text="@string/row1"
android:layout_span="3"
         android:padding="18dip" android:background="#b0b0b0"
         android:textColor="#000"/>
   </TableRow>
```

Table Layout

```
<!-- Fila 2 con 3 columnas -->
<TableRow
   android:id="@+id/tableRow1"
   android:layout_height="wrap_content"
   android:layout_width="match_parent">
   <TextView
      android:id="@+id/TextView04" android:text="@string/row2col1"
      android:layout_weight="1" android:background="#dcdcdc"
      android:textColor="#000000"
      android:padding="20dip" android:gravity="center"/>
   <TextView
      android:id="@+id/TextView04" android:text="@string/row2col2"
      android:layout_weight="1" android:background="#d3d3d3"
      android:textColor="#000000"
      android:padding="20dip" android:gravity="center"/>
   <TextView
      android:id="@+id/TextView04" android:text="@string/row2col3"
      android:layout_weight="1" android:background="#cac9c9"
      android:textColor="#000000"
      android:padding="20dip" android:gravity="center"/>
</TableRow>
```

Table Layout

```
<!-- Fila 3 con 2 columnas -->
<TableRow
   android:layout_height="wrap_content"
   android:layout_width="fill_parent"
   android:gravity="center_horizontal">
   <TextView
      android:id="@+id/TextView04" android:text="@string/row3col1"
      android:layout_weight="1" android:background="#b0b0b0"
      android:textColor="#000000"
      android:padding="20dip" android:gravity="center"/>

   <TextView
      android:id="@+id/TextView04" android:text="@string/row3col2"
      android:layout_weight="1" android:background="#a09f9f"
      android:textColor="#000000"
      android:padding="20dip" android:gravity="center"/>
</TableRow>

</TableLayout>
```

Frame Layout

Posiciona las vistas usando todo el contenedor, sin distribuirlas espacialmente. Este Layout suele utilizarse cuando queremos que varias vistas ocupen un mismo lugar. Podemos hacer que solo una sea visible, o superponerlas. Para modificar la visibilidad de un elemento utilizaremos la propiedad visibility.

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">
        <AnalogClock
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"/>
        <CheckBox
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Un checkBox"/>
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Un botón"
            android:visibility="invisible"/>
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Un texto cualquiera"
            android:visibility="invisible"/>
</FrameLayout>
```

Constraint Layout

Constraint Layout es la nueva tecnología de Android Studio para crear interfaces de usuario en distintos tamaños y resoluciones. Ahora bastará crear una vista e implementar los constraints para crear una interfaz de usuario de todos los tamaños de pantalla que existen en el mercado.

ConstraitLayout nos permitirá simplificar las interfaces en anidamiento, para hacerlas lo más complejas posibles a nivel de diseño. Este layout, similar al RelativeLayout nos permitirá establecer relaciones entre todos los elementos y la propia vista padre, permitiendo así ser mucho más flexible que los demás.

Con un estilo muy visual, podremos desde Android Studio gestionar todas las relaciones que establezcamos, de un modo muy sencillo, al más puro estilo drag-and-drop, en lugar de necesitar utilizar el fichero XML.

Constraint Layout

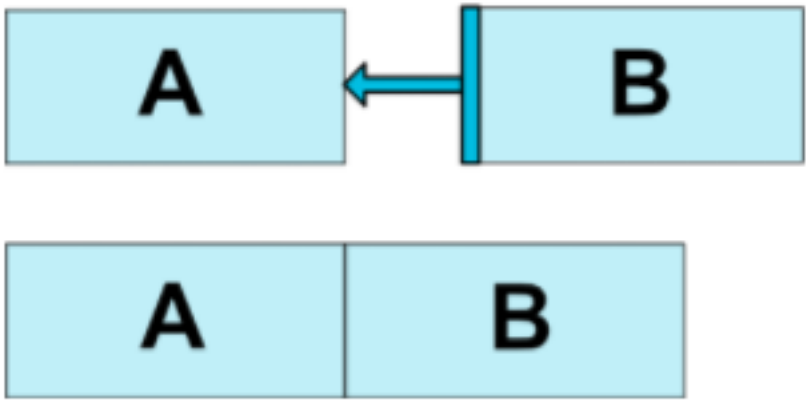
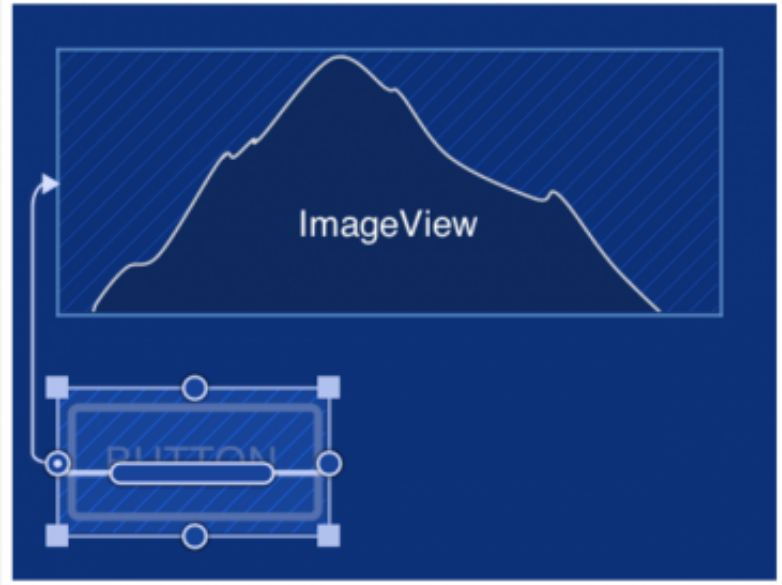
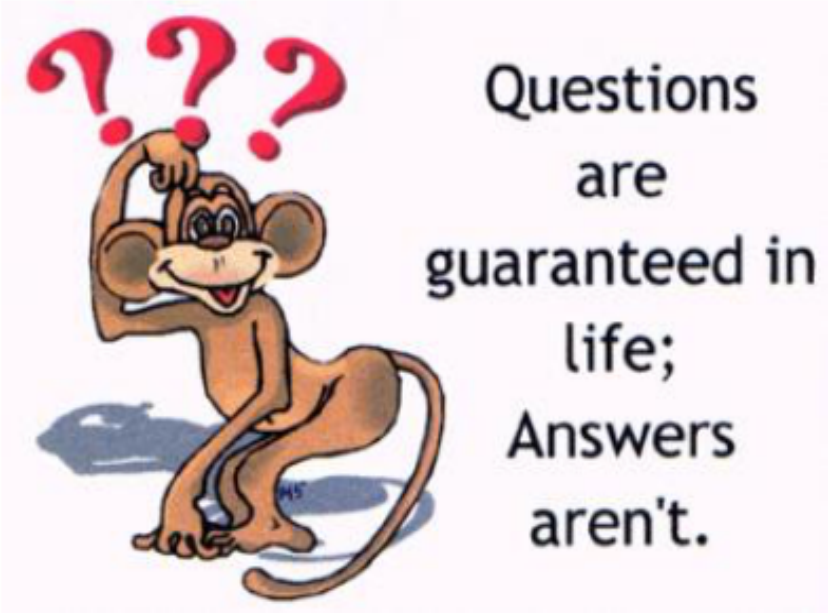


Fig. 1 - Relative Positioning Example




```
<Button android:id="@+id/buttonA" ... />
        <Button android:id="@+id/buttonB" ...
                app:layout_constraintLeft_toRightOf="@+id/buttonA" />
```

https://developer.android.com/reference/android/support/constraint/ConstraintLayout.html
https://obux.wordpress.com/2017/03/17/tutorial-de-constraint-layout-android/
https://elandroidelibre.elespanol.com/2016/10/revolucion-interfaces-android-constraintlayout.html



¿Preguntas?

Julio César Galeano García