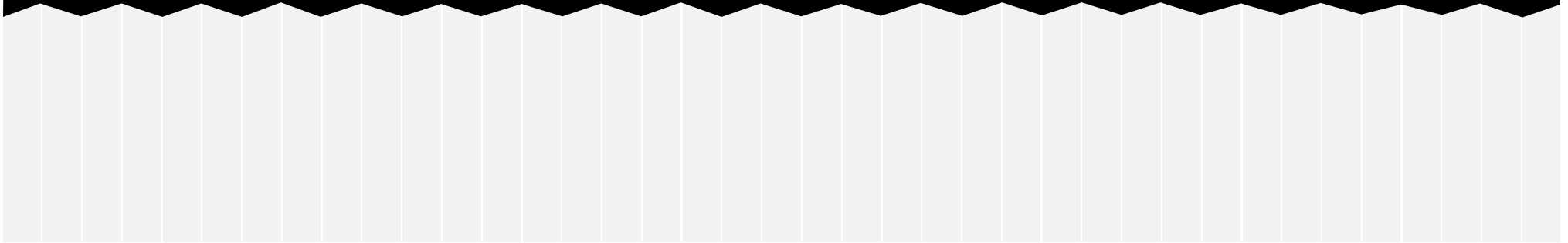


# Tipos de Layouts

A horizontal dotted line in a light green color, positioned directly below the title text.

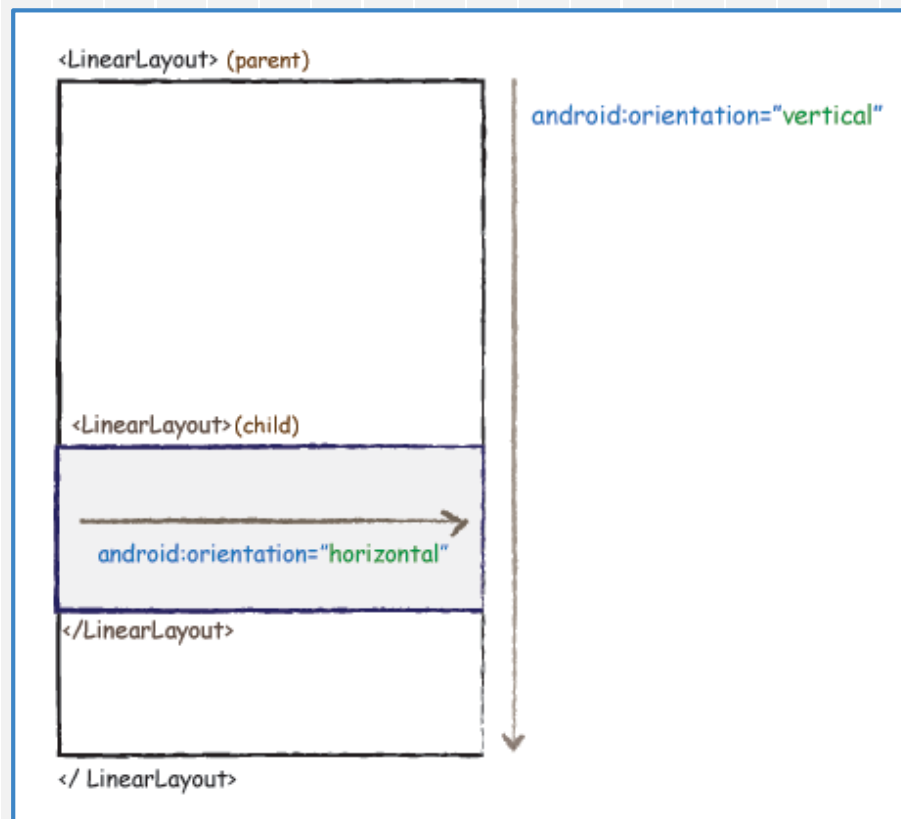
# Tipos de Layout

---

Se pueden definir los siguientes Diseños:

1. `LinearLayout`
2. `RelativeLayout`
3. `GridLayout` (partir de Android 4.0, reemplaza a `TableLayout`)
4. `FrameLayout`
5. `ScrollView`

# LinearLayout



Los atributos principales de este tipo de Layout es

`android:orientation = "vertical"` ó

`android:orientation = "horizontal"`

También atributo `android:weight` (peso) que define cómo se distribuirán los objetos en el Activity.

Si hay 2 objetos, 1º peso=2 y 2º peso=1

Se divide en tercios y asigna  $\frac{2}{3}$  1º y  $\frac{1}{3}$  2º

# Ejercitación

1. Crear un proyecto llamarlo LayoutApp.
2. En MainActivity:
  - a. Reemplazar el RelativeLayout por LinearLayout
  - b. Agregar 4 botones, LinearLayout, RelativeLayout, FrameLayout, ScrollView. En cada uno:
    - i. Asignar atributo id
    - ii. Asignar atributo text (crear los string en archivo strings)
    - iii. Asignar atributo onClick = AbrirLinearLayout, etc. (c/u nombre ➡)

# Ejercitación

## 3. Agregar otra activity LinearLayoutActivity y XML asignar (= al Alumni)

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

  <TextView android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Email:"
    android:padding="5dip"/>
  <EditText android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="10dip"/>

  <Button android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Login"/>
```

# Ejercitación

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

```
    <TextView android:layout_width="fill_parent"  
        android:layout_height="wrap_content"  
        android:text="Home"  
        android:padding="15dip"  
        android:layout_weight="1"  
        android:gravity="center"/>
```

```
    <TextView android:layout_width="fill_parent"  
        android:layout_height="wrap_content"  
        android:text="About" android:padding="15dip"  
        android:layout_weight="1"  
        android:gravity="center"/>
```

```
</LinearLayout>
```

```
</LinearLayout>
```

# Ejercitación

4. En MainActivity.java agregar el método asignado a las propiedad OnClick del Boton que llama a la Activity LinearLayout.

```
/*crear el Método que asignamos en el OnClick del Boton:
/*
public void AbrirXXXXXLayout(View view){
    // Declaro el intent que abrirá la Activity
    Intent intent = new Intent(this, ClaseActivityXXXXX.class);
    startActivity(intent);
}
*/
//Así queda el definitivo
public void AbrirLinearLayout() {
    Intent intent = new Intent(this, LinearLayoutActivity.class);
    startActivity(intent);
}
```

# Ejercitación

5. En el AndroidManifest.xml, para que el Layout quede con pantalla negra (Alumni) hay que modificar el Tema de la Activity a Theme.Black y queda así:

```
<application...
```

```
android:theme="@android:style/Theme.Black" >
```

Así es para toda la aplicación

Dentro TAG activity es Solo para esa

```
<activity
```

```
....
```

```
android:theme=
```

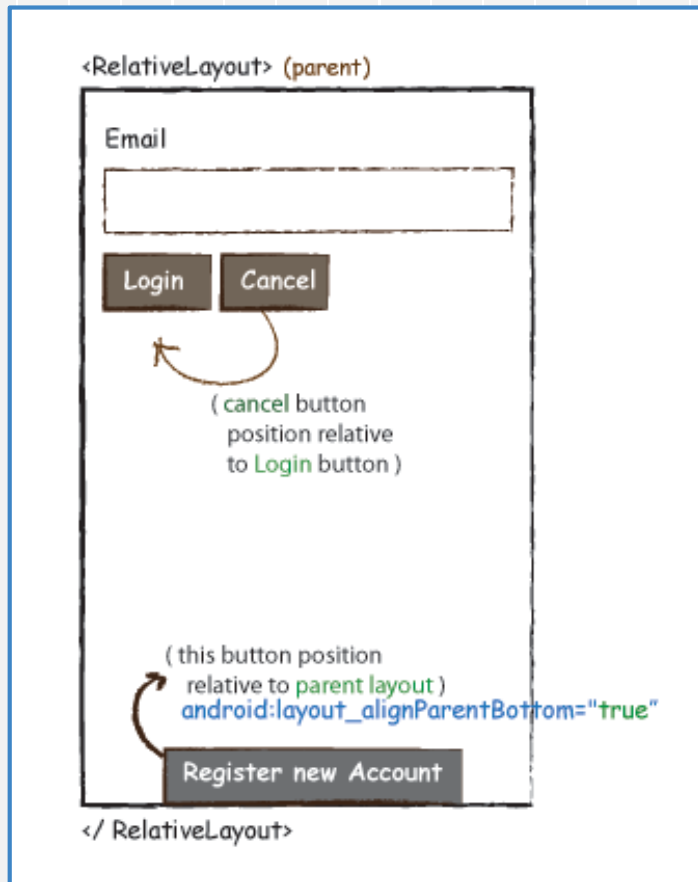
```
"@android:style/Theme.Black"
```

```
>
```





# RelativeLayout



Los atributos principales de este tipo de Layout es `android:layout_alignParentBottom="true"` (posición relativa a la View "Padre", que es Layout)

También atributo `android:weight` (peso)

Si hay 2 objetos, 1° peso=2 y 2° peso=1

Se divide en tercios y asigna  $\frac{2}{3}$  1° y  $\frac{1}{3}$  2°

`layout_centerInParent`,

# RelativeLayout

## Posición relativa a otro control:

- `android:layout_above`
- `android:layout_below`
- `android:layout_toLeftOf`
- `android:layout_toRightOf`
- `android:layout_alignLeft`
- `android:layout_alignRight`
- `android:layout_alignTop`
- `android:layout_alignBottom`
- `android:layout_alignBaseline`

## Posición relativa al layout padre:

- `android:layout_alignParentLeft`
- `android:layout_alignParentRight`
- `android:layout_alignParentTop`
- `android:layout_alignParentBottom`
- `android:layout_centerHorizontal`
- `android:layout_centerVertical`
- `android:layout_centerInParent`

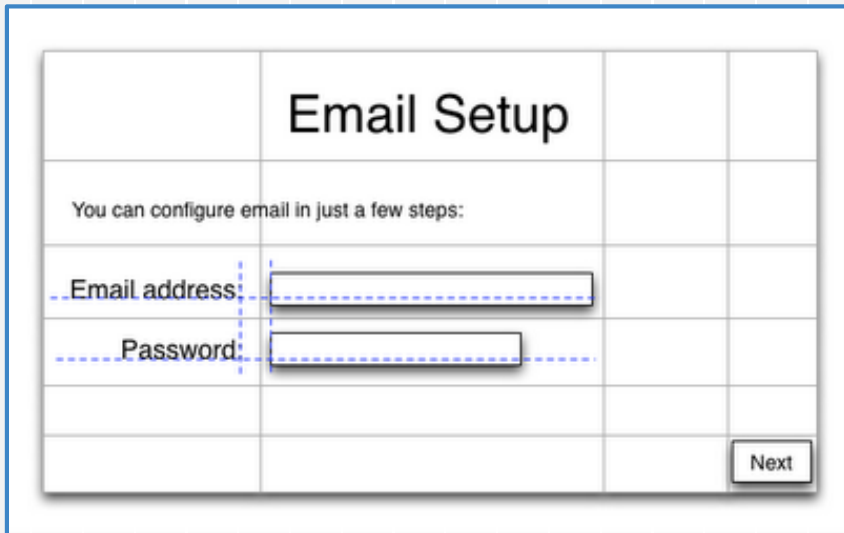
## Opciones de margen exterior:

- `android:layout_margin`
- `android:layout_marginBottom`
- `android:layout_marginTop`
- `android:layout_marginLeft`
- `android:layout_marginRight`

## Opciones de margen interior:

- `android:padding`
- `android:paddingBottom`
- `android:paddingTop`
- `android:paddingLeft`
- `android:paddingRight`

# GridLayout



The image shows a screenshot of an 'Email Setup' form. The form is organized into a grid with 4 columns and 5 rows. The first row contains the title 'Email Setup' in the second column. The second row contains the text 'You can configure email in just a few steps:' in the first column. The third row contains the label 'Email address' in the first column and a text input field in the second column. The fourth row contains the label 'Password' in the first column and a text input field in the second column. The fifth row contains a 'Next' button in the fourth column. Dashed blue lines highlight the grid structure.

	Email Setup		
You can configure email in just a few steps:			
Email address	<input type="text"/>		
Password	<input type="password"/>		
			Next

Los atributos característicos de este tipo de Layout es cantidad de filas y columnas.

`android:columnCount="4"`

ó `android:rowCount="4"` .

A su vez las View se asignan a la fila, columna donde estará ubicada con el atributo `android:layout_column="3"` y `android:layout_row="5"`, son, por ejemplo, los atributos del botón Next.

# FrameLayout

- Es el layout más simple de todos los de Android.
- Suele utilizarse para mostrar un único control en su interior, a modo de contenedor sencillo para un sólo elemento, por ejemplo una imagen.
- Coloca todos sus controles hijos alineados con su esquina superior izquierda. Cada control quedará oculto por el control siguiente (a menos que éste último tenga transparencia).
- Propiedades **android:layout\_width** y **android:layout\_height**,  
“match\_parent” para que el control hijo tome la dimensión de su layout contenedor  
“wrap\_content” para que el control hijo tome la dimensión de su contenido.

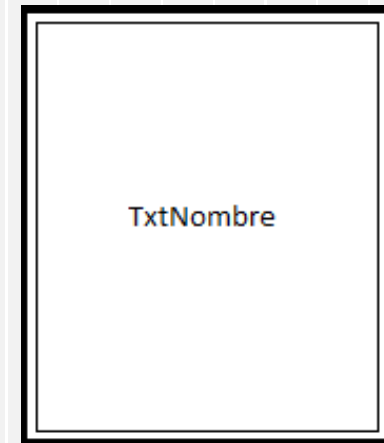
# Ejercitación

- Dentro de una Activity con LinearLayout, agregar este FrameLayout.

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

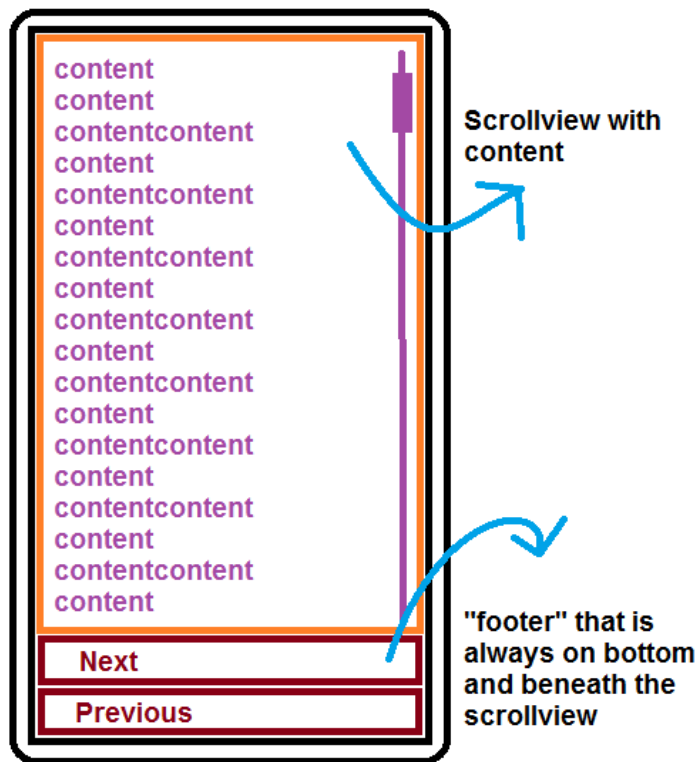
    <EditText
        android:id="@+id/TxtNombre"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:inputType="text" />

</FrameLayout>
```



- Agregar un segundo FrameLayout y asignarle un ImageView.

# ScrollView



Se usa para Activity más grande que el tamaño del dispositivo.

Agrega una barra de desplazamiento horizontal o vertical según se haya definido en el atributo

`android:orientation="vertical"` o `"horizontal"`

El atributo `android:fillViewport="true"` asegura que el ScrollView está en la pantalla.

# Para seguir investigando

Existen otros layouts algo más sofisticados a los que dedicaremos artículos específicos un poco más adelante, como por ejemplo el **DrawerLayout** para añadir menús laterales deslizantes.

Site **developer.android.com**

<http://developer.android.com/intl/es/guide/topics/ui/layout/linear.html>

<http://developer.android.com/intl/es/guide/topics/ui/layout/relative.html>

<http://developer.android.com/intl/es/guide/topics/ui/layout/gridview.html>

<http://developer.android.com/intl/es/reference/android/widget/FrameLayout.html>

<http://developer.android.com/intl/es/reference/android/widget/ScrollView.html>