	UNIVERSIDAD DON BOSCO **FACULTAD DE INGENIERÍA** **ESCUELA DE COMPUTACIÓN**	
CICLO: I/2021	**GUIA DE LABORATORIO #06**	
	Nombre de la Práctica:	Layouts
	MATERIA:	Desarrollo de Software para Móviles

I. OBJETIVOS

1. Conocer el concepto de Layout.
2. Conocer los layouts más utilizados para construir una aplicación Android.
3. Crear una aplicación Android y construir su interfaz gráfica haciendo uso de Layouts.

II. INTRODUCCION TEORICA

En la actualidad las aplicaciones (en general no solamente Android) pueden poseer diferentes tipos de interfaz con las cuales el usuario puede interactuar e indicar las acciones que desea realizar. Este tipo de interfaces se limitan actualmente a: las interfaces puramente de texto (aplicaciones de terminal) e interfaces gráficas (que son las mayormente utilizadas hoy en día).

Android como sistema operativo cuenta con aplicaciones que hacen uso de interfaces gráficas y para facilitar la vida de sus desarrolladores creó un API especial llamada Layouts.

Un Layout es **un contenedor o cuadrícula que sirve para dividir espacios en nuestras vistas** (view) dentro de una actividad. De esta manera, se facilita la distribución de elementos como textos, gráficos y demás en una aplicación. Permitiendo así una mejor interacción del usuario con la interfaz (UI).

Los layouts se pueden escribir en XML o por medio del Asistente Drag and Drop de la interfaz gráfica. Sin embargo, **lo recomendable es escribirlos en XML** porque así tendrás un mayor control de las clases y subclases de la vista y cada uno de sus elementos.

Al tratarse de elementos XML estos poseen una serie de características y atributos que permiten describir el funcionamiento del layout y la forma en que se mostrará en pantalla.

Por ejemplo: **android:layout_height="match_parent" –** aplica la dimensión de su layout contenedor. **android:layout_width="20dp"** – unidad de medida (dp) – Densidad de

píxeles independientes. **android:layout_alignParentBottom="true"** – indica al widget que su borde superior deberá estar alineado con el borde superior del contenedor.

Numerar o explicar todos los atributos que puede tener un layout se escapa del objetivo de esta guía así que se invita al estudiante a documentarse con la lectura de las guías de desarrollo oficiales de Google.

Tipos de Layout

FrameLayout

Un FrameLayout es un view group creado para mostrar un solo elemento en pantalla.

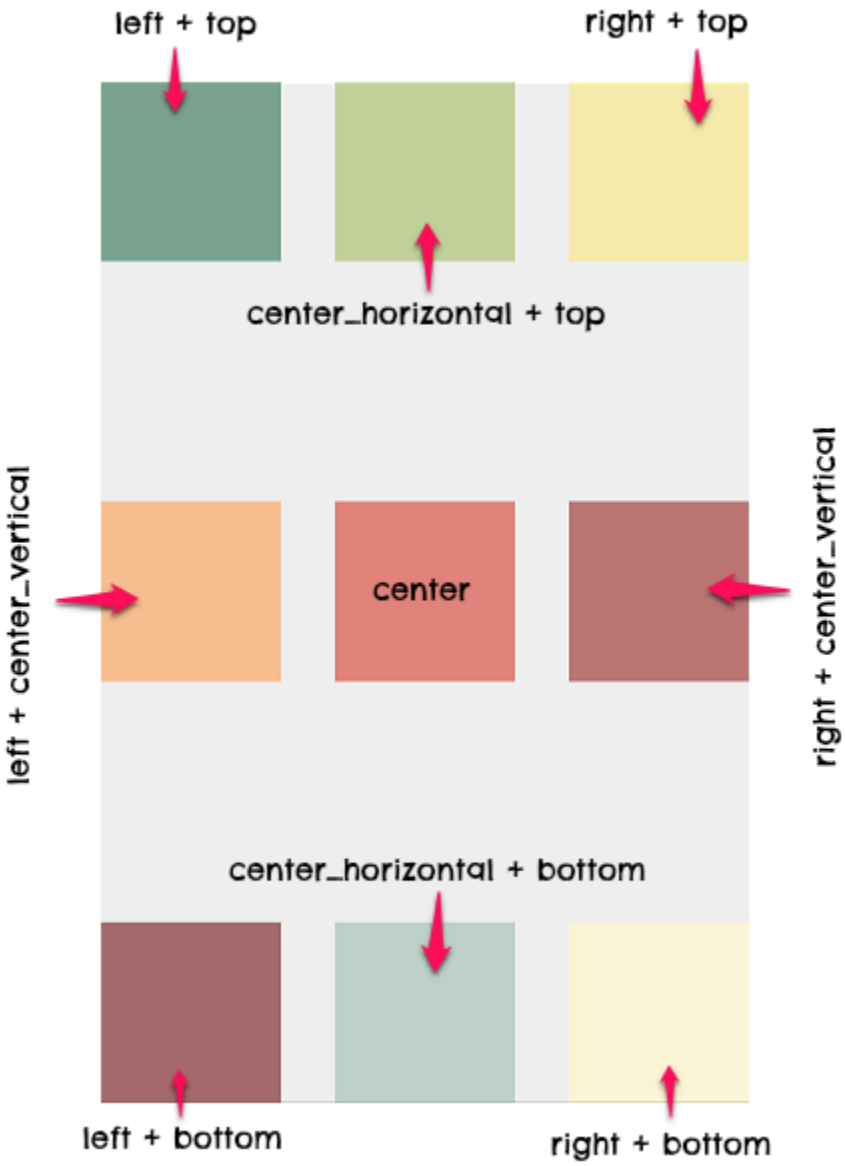
Sin embargo puedes añadir varios hijos con el fin de superponerlos, donde el último hijo agregado, es el que se muestra en la parte superior y el resto se pone por debajo en forma de pila.

Para alinear cada elemento dentro del FrameLayout se usa el parámetro android:layout_gravity.

El parámetro gravitiy se basa en las posiciones comunes de un view dentro del layout. Se describe con constantes de orientación:

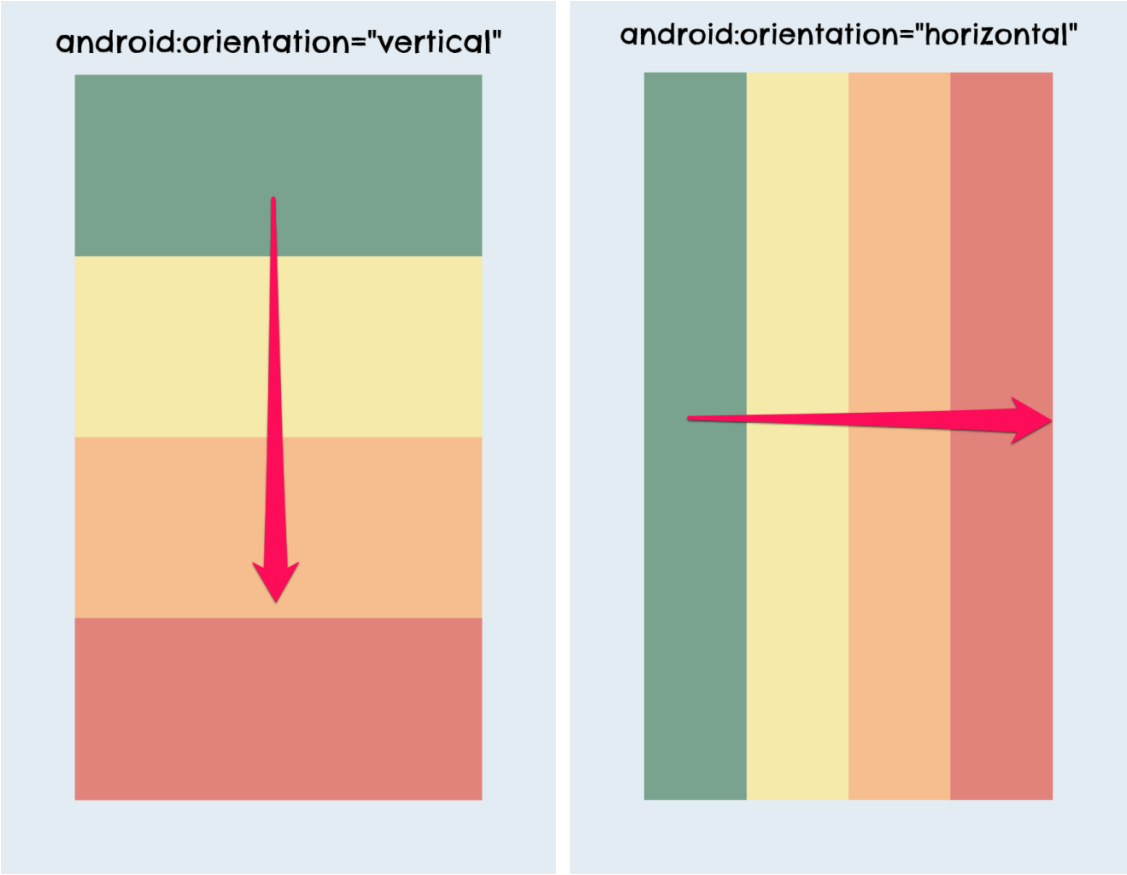
- top: Indica la parte superior del layout.
- left: Indica la parte izquierda del layout.
- right: Se refiere a la parte derecha del layout.
- bottom: Representa el límite inferior del layout.
- center_horizontal: Centro horizontal del layout.
- center_vertical: Alineación al centro vertical del layout.
- center: Es la combinación de center_horizontal y center_vertical.

Como se muestra en la ilustración de abajo, es posible crear variaciones combinadas, como por ejemplo **right + bottom**.

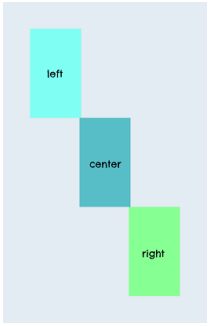


LinearLayout

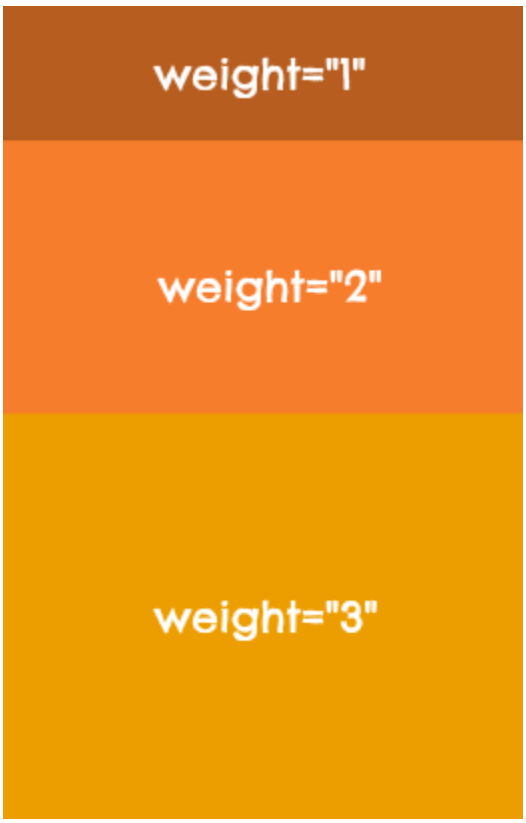
Un LinearLayout es un view group que distribuye sus hijos en una sola dimensión establecida. Es decir, todos organizados en una sola columna (vertical) o en una sola fila (horizontal). La orientación se elige a través del atributo android:orientation.



Al igual que el FrameLayout, el LinearLayout permite asignar una gravedad a cada componente según el espacio que ocupa.



Adicionalmente existe un parámetro llamado android:layout_weight, el cual define la importancia que tiene un view dentro del linearlayout. A mayor importancia, más espacio podrá ocupar.

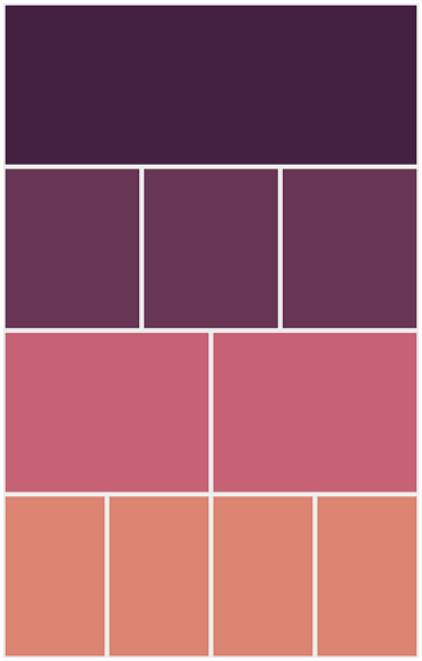


La anterior ilustración muestra tres views con pesos de 1, 2 y 3 respectivamente. La magnitud de sus alturas corresponde a su preponderancia. Matemáticamente, el espacio disponible total sería la suma de las alturas (6), por lo que 3 representa el 50%, 2 el 33,33% y 1 el 16,66%.

TableLayout

Permite distribuir los elementos en una cuadrícula de forma tabular. De esta manera, podemos definir filas y columnas y, por supuesto, la posición de cualquier elemento dentro de la tabla. Este layout es una distinción de **LinearLayout vertical** y **TableRow** será siempre un hijo de esta. El ancho de cada columna será igual al ancho del mayor elemento adicionado a la vista. De la misma manera, podemos cambiar este comportamiento utilizando las siguientes propiedades:

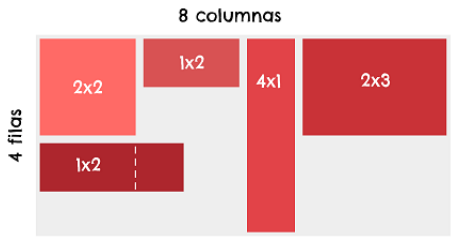
- **android:stretchColumns**: las columnas se expanden logrando absorber el espacio libre que no usan las otras columnas
- **android:shrinkColumns**: las columnas se contraen para lograr espacio entre otras columnas
- **android:collapseColums**: permite ocultar las tablas



Estas propiedades las podemos aplicar a todas las columnas o sólo a las que necesitemos. Por ejemplo, si necesitamos aplicarlo a todas las columnas debemos terminar el comando con un asterisco: **android:shrinkColumns="*"**.

GridLayout

Un GridLayout es un ViewGroup que alinea sus elementos hijos en una cuadrícula (grilla ó grid). Nace con el fin de evitar anidar linear layouts para crear diseños complejos.



Su funcionamiento se basa en un sistema de índices con inicio en cero. Es decir, la primera columna (o fila) tiene asignado el índice 0, la segunda el 1, la tercera el 2, etc.

Los atributos más importantes del GridLayout son:

- columnCount: Cantidad de columnas que tendrá la grilla.
- rowCount: Cantidad de filas de la cuadrícula.
- useDefaultMargins: Si asignas el valor de true para establecer márgenes

predeterminados entre los ítems.

En cuanto a sus parámetros, es posible especificar la cantidad de filas y columnas que ocupará una celda a través de los atributos android:layout_rowSpan y android:layout_columnSpan. Esta característica nos posibilita crear diseños irregulares que un TableLayout no permitiría.

En la ilustración mostrada al inicio, existe una cuadrícula de 8×4 con 5 views. Sus atributos span se encuentran escritos en forma **axb**.

También puedes especificar a qué fila y columna pertenece cada view con los atributos android:layout_row y android:layout_column.

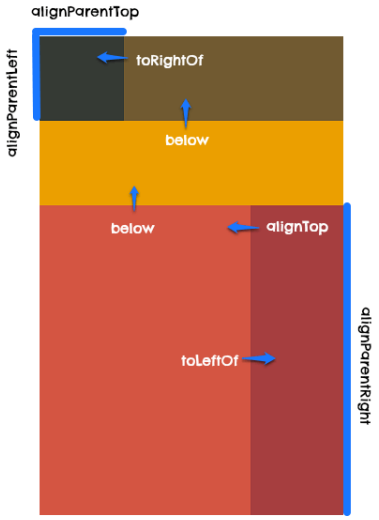
RelativeLayout

Este layout permite **especificar la posición de cada elemento utilizando ubicaciones relativas** en relación a otro elemento padre o hermano.

Este elemento es el más flexible y elaborado de todos los view groups que veremos. El RelativeLayout permite alinear cada hijo con referencias subjetivas de cada hermano.

Con el RelativeLayout pensaremos en cómo alinear los bordes de cada view con otros. Imagina en una sentencia como «el botón estará por debajo del texto» o «la imagen se encuentra a la derecha de la descripción».

En ninguno de los casos nos referimos a una posición absoluta o un espacio determinado. Simplemente describimos la ubicación y el framework de Android computará el resultado final.



El ejemplo anterior ilustra cómo una serie de views forman un diseño irregular. Esto es posible gracias a unos parámetros que determinan cómo se juntan los bordes de cada uno y en qué alineación.

Cada referencia es indicada usando el identificador de cada view. Por ejemplo, el siguiente botón está por debajo de un view con id "editor_nombre" (se indica con el parámetro layout_below).

Algunos de los parámetros del RelativeLayout para definir posiciones:

- android:layout_above: Posiciona el borde inferior del elemento actual con el borde superior del view referenciado con el id indicado.
- android:layout_centerHorizontal: Usa true para indicar que el view será centrado horizontalmente con respecto al padre.
- android:layout_alignParentBottom: Usa true para alinear el borde inferior de este view con el borde inferior del padre.
- android:layout_alignStart: Alinea el borde inicial de este elemento con el borde inicial del view referido por id.

III. PROCEDIMIENTO

Para realizar la práctica debe haber completado con éxito la guía anterior **Introducción a Android**.

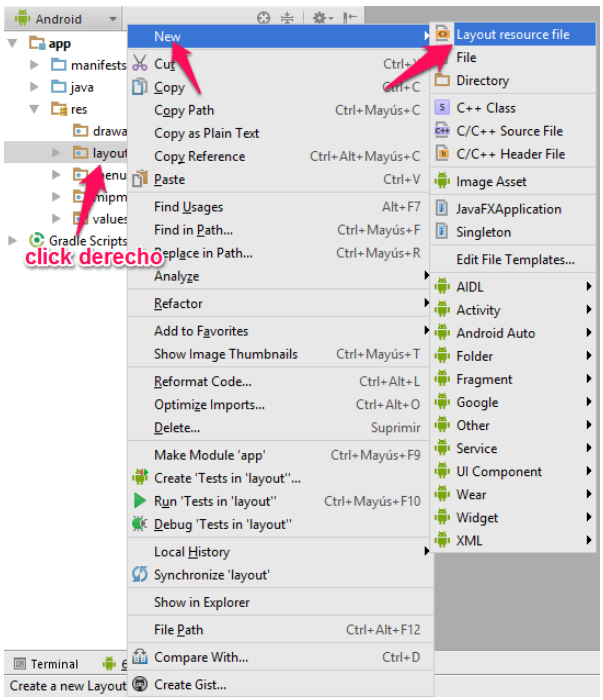
1. Ejecuta Android Studio y crea un nuevo proyecto. En **Project Template** selecciona **Empty Activity** y presiona **Next**.
2. Como **Name** coloca **Guia06App,** en **Package name** escribe **sv.edu.udb.guia06app** en **Save location** escoge la carpeta de tu preferencia, en Language elige **Java** y el **Minimum SDK** selecciona **API 16: Android 4.1 (Jelly Bean)**

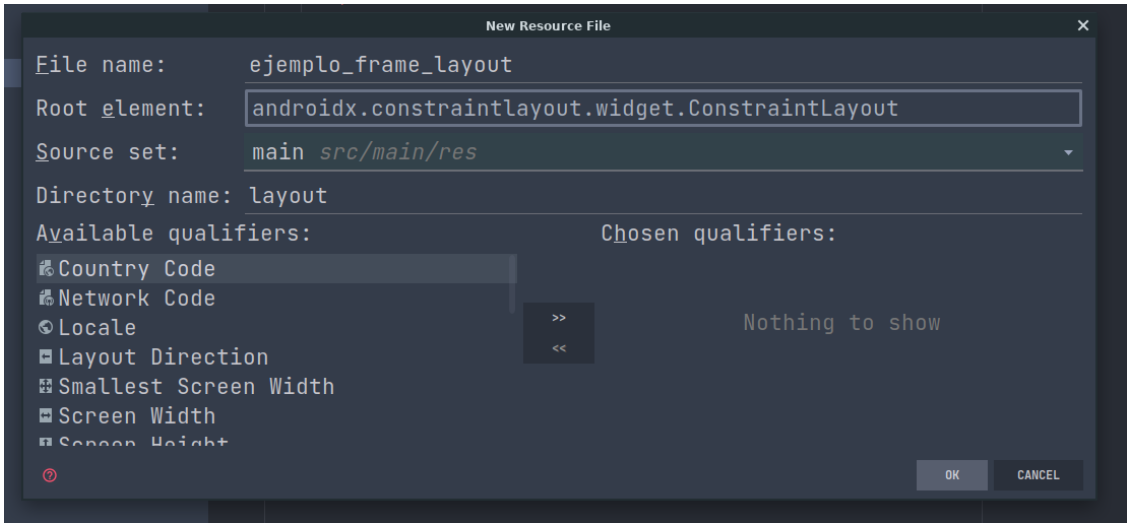
Nota: En esta práctica se hará uso del lenguaje de programación Java pero, se invita al estudiante a realizarla también utilizando Kotlin.

3. Presiona **Finish.**

Ejemplo de FrameLayout

1. Crea un nuevo Layout llamado **ejemplo_frame_layout**.





2. Abre el archivo recién creado y reemplaza el código generado por el IDE por el siguiente:https://gist.github.com/hugovalenciadev/93ad33910e89047a01431e688ab11edf

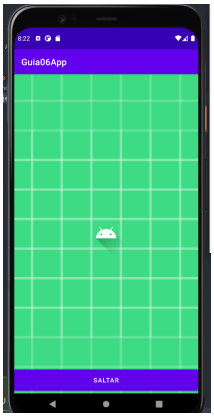
3. Abre MainActivity.java y edita la siguiente línea:

   setContentView(R.layout.activity_main);

   y en su lugar coloca lo siguiente:

setContentView(R.layout.ejemplo_frame_layout);

4. Presiona **Run 'app'.**



Ejemplo de LinearLayout

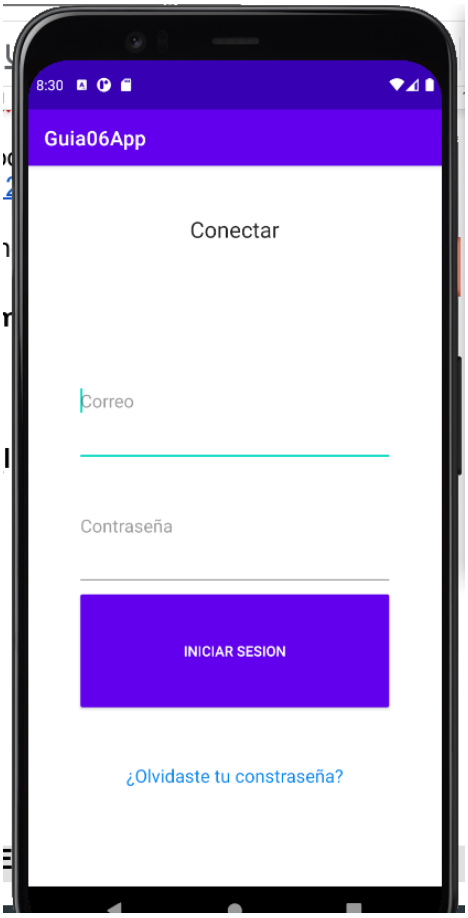
1. Crea un nuevo layout llamado **ejemplo_linear_layout**.
2. Edita el archivo recién creado y coloca el código del siguiente enlace: https://gist.github.com/hugovalenciadev/6b1205101815d44fd4c080de0353ff8c.
3. Abre MainActivity.java y edita la siguiente línea:

   setContentView(R.layout.activity_main);

   y en su lugar coloca lo siguiente:

   setContentView(R.layout.ejemplo_linear_layout);

4. Presiona **Run 'app'.**



Ejemplo de RelativeLayout

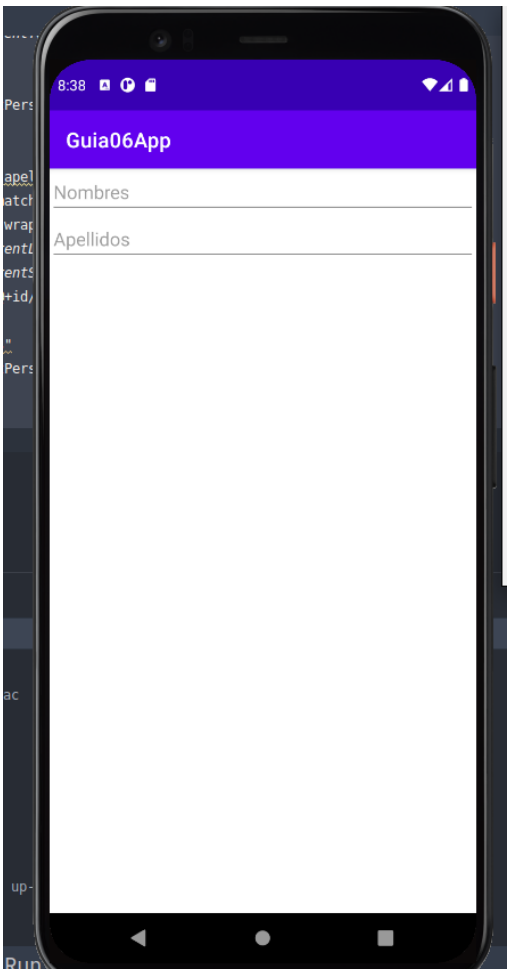
1. Crea un nuevo layout llamado **ejemplo_relative_layout**.
2. Edita el archivo recién creado y coloca el código del siguiente enlace: https://gist.github.com/hugovalenciadev/3ff858a792a1040e8d3ff429d6001b21
3. Abre MainActivity.java y edita la siguiente línea:

   setContentView(R.layout.activity_main);

   y en su lugar coloca lo siguiente:

   setContentView(R.layout.ejemplo_relative_layout);

4. Presiona **Run 'app'.**



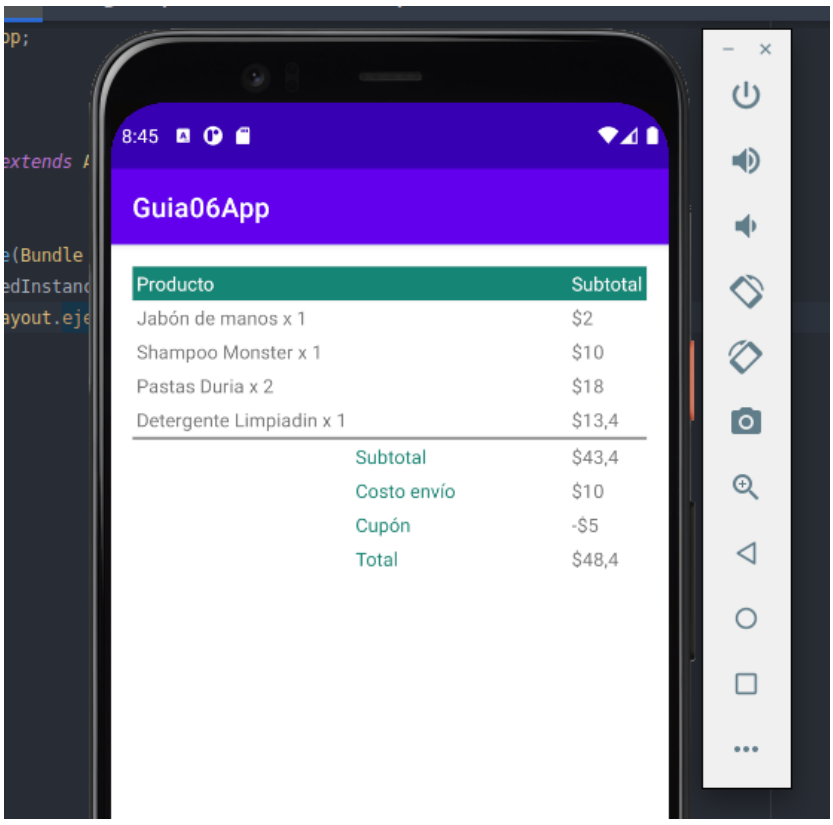
Ejemplo de TableLayout

1. Crea un nuevo layout llamado **ejemplo_table_layout**.
2. Edita el archivo recién creado y coloca el código del siguiente enlace: https://gist.github.com/hugovalenciadev/b3fac38b14f8f6624381e7a1397c0c11
3. Abre MainActivity.java y edita la siguiente línea:

   setContentView(R.layout.activity_main);

   y en su lugar coloca lo siguiente:

   setContentView(R.layout.ejemplo_table_layout);
4. Presiona **Run 'app'.**



Ejemplo de GridLayout

1. Crea un nuevo layout llamado **ejemplo_grid_layout**.

2. Agrega el siguiente código a tu archivo **res/values/themes.xml**

```
 <style name="AppTheme.BotonCalculadora"
parent="TextAppearance.AppCompat.Headline">
        <item name="android:textColor">@android:color/white</item>
        <item name="android:gravity">center</item>
        <item name="android:padding">36dp</item>
        <item name="android:layout_width">wrap_content</item>
        <item name="android:layout_height">wrap_content</item>
    </style>

    <style name="AppTheme.BotonCalculadora.Azul"
parent="AppTheme.BotonCalculadora">
        <item name="android:background">#00537D</item>
```

```
    </style>

    <style name="AppTheme.BotonCalculadora.Rojo"
parent="AppTheme.BotonCalculadora">
        <item name="android:background">#EA4D39</item>
    </style>
```

3. Edita el archivo recién creado y coloca el código del siguiente enlace: https://gist.github.com/hugovalenciadev/9001378104089f52d9ead7e4db8e5bfc

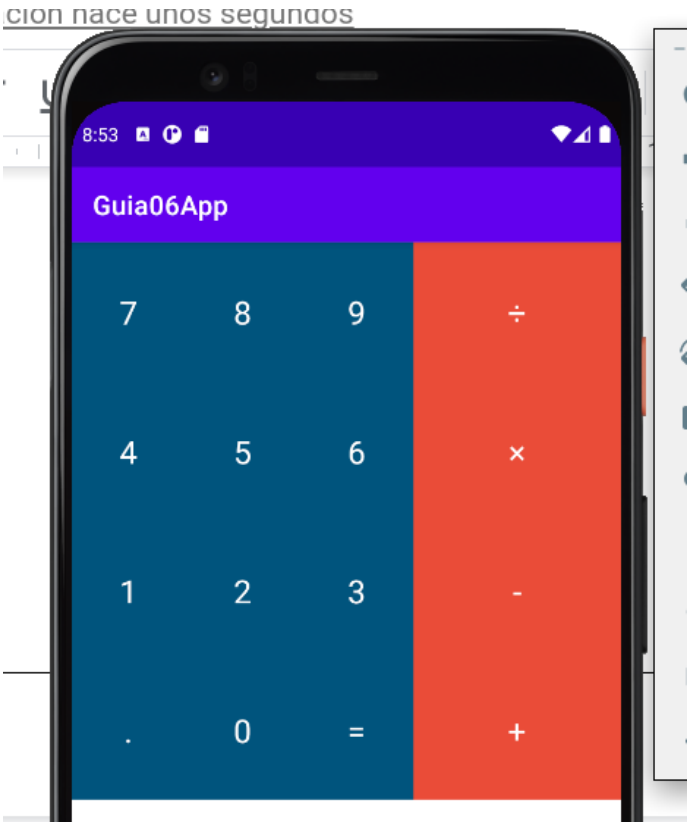
4. Abre MainActivity.java y edita la siguiente línea:

   setContentView(R.layout.activity_main);

   y en su lugar coloca lo siguiente:

   setContentView(R.layout.ejemplo_grid_layout);

5. Presiona **Run 'app'.**



IV. DISCUSIÓN DE RESULTADOS

1. Crea una aplicación Android con un layout como el del ejemplo de LinearLayout (una pantalla de login) solo que en lugar de utilizar LinearLayout deberá hacerse exclusivamente con RelativeLayout.

2. Investigar el uso de ConstraintLayout y realizar un nuevo layout (la misma pantalla de login del numeral anterior) solo que en lugar de utilizar RelativeLayout deberá hacerse exclusivamente con ConstraintLayout.

V. BIBLIOGRAFÍA

- Documentación Oficial Android
  https://developer.android.com/guide/topics/ui/declaring-layout?hl=es-419