	UNIVERSIDAD DON BOSCO **FACULTAD DE INGENIERÍA** **ESCUELA DE COMPUTACIÓN**	
CICLO: I/2022	**GUIA DE LABORATORIO #3**	
	Nombre de la Práctica:	Guía #3- Recursos y Layouts
	MATERIA:	Desarrollo de Software para Móviles

I. OBJETIVOS

1 Conocer e incorporar diferentes tipos y recursos en Android.
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

Recursos

Los recursos son los archivos adicionales y el contenido estático que usa tu código, como mapas de bits, definiciones de diseño, strings de interfaz de usuario, instrucciones de animación, etc.

Siempre debes externalizar los recursos para aplicaciones, como imágenes y strings de tu código, para que puedas mantenerlos de forma independiente.
También debes proporcionar recursos alternativos para configuraciones de dispositivos específicos, agrupándolos en directorios de recursos con un nombre especial.

En tiempo de ejecución, Android utiliza el recurso adecuado según la configuración actual. Por ejemplo, puedes proporcionar un diseño de interfaz de usuario (IU) diferente según el tamaño de la pantalla o strings diferentes según la configuración de idioma.

Una vez que externalizas los recursos para tu aplicación, puedes acceder a ellos mediante los ID de recursos que se generan en la clase **R** de tu proyecto.

En este documento, se muestra cómo puedes agrupar los recursos en tu proyecto de Android, proporcionar recursos alternativos para configuraciones de dispositivos específicos y acceder a ellos desde el código de tu aplicación u otros archivos XML posteriormente.

Tipos de recursos.
Android posee 2 grandes tipos de recursos:

- **Recursos del proyecto.**
  Son recursos que son exclusivos del proyecto donde residen (recursos alternativos)

- **Recursos del sistema.**
  Son parte del sistema operativo y mantienen compatibilidad con las diferentes versiones del mismo.

Recursos del proyecto.
Debes colocar cada tipo de recurso en un subdirectorio específico del directorio `res/` de tu proyecto. Por ejemplo, esta es la jerarquía de archivos de un proyecto simple:

```
MyProject/
    src/
        MyActivity.java
    res/
        drawable/
            graphic.png
        layout/
            main.xml
            info.xml
        mipmap/
            icon.png
        values/
            strings.xml
```

El directorio **res/** contiene todos los recursos (en subdirectorios): un recurso de imagen, dos recursos de diseño, directorios **mipmap/** para los íconos del selector y un archivo de recursos de strings.

Los nombres del directorio de recursos son importantes y se describen la siguiente tabla.

Carpeta (identificador con clase R)	Descripción breve
res/drawable R.drawable	Archivos de mapas de bits (.png, .9.png, .jpg, .gif) o archivos XML que se han compilado en los siguientes subtipos de recursos de elemento de diseño: Archivos de mapas de bits, Nine patchs (mapas de bits reajustables), **Listas** de estados, Formas, Elementos de diseño de animaciones, Otros elementos de diseño
res/layout R.layout	Archivos XML que contienen layouts.
res/menu R.menu	Archvios XML con descriptores de menús que podemos usar en nuestra aplicación.
res/anim R.anim	Archivos XML que permite definir una animación con interpolación de movimientos.
res/animator R.animator	Archivos XML que permiten modificar las propiedades de objetos a lo largo del tiempo.

res/xml R.xml	Otros archivos XML como archivos de preferencias.
res/raw R.raw	Archivos de Audio, video, texto, etc.

Archivos (identificador con clase R)	**Descripción breve**
res/values	Archivos XML que definen un determinado valor para referencias un color, estilo, cadena.

Tipos de recursos en la carpeta values:

Archivos (identificador)	**Descripción breve**
strings.xml R.string	Identifica cadenas de caracteres **<string name="hola">Hola mundo</string>**
colors.xml R.color	Colores definidos en formato ARGV Los valores se indican en hexadecimal en los formatos #RGB, #ARGB, #RRGGBB ó #AARRGGBB **<color name="verde_opaco">#0f0</color>** **<color name="Rojo_translucido">#80ff0000</color>**
dimensions.xml R.dimen	Un número seguido de una unidad de medida **px – pixeles, mm – milímetros, in – pulgadas, pt – puntos** **dp – pixeles independientes de la densidad.** **<dimen name="alto">2.2mm</dimen>** **<dimen name="tamaño_fuente">16dp</dimen>**

Archivos (identificador)	**Descripción breve**
styles.xml R.styles	Serie de atributos que pueden ser aplicados a una vista o actividad. Si se aplican a una actividad se conocen como temas. **<style name="TextoGrande" parent="@style/Text">** **<item name="android:textSize">20pt</item>** **<item name="android:textColor">#000080</item>** **</style>**

R.int	Define un valor entero **<integer name= “edad”>20</integer>**
R.bool	Define un valor booleano **<bool name= “continuar”>true</bool>**
R.id	Define un identificador único de recurso. **<item type=“id” name=“boton_ok” />** **<item type=“id” name=“txtNombre” />**

Archivos (identificador)	**Descripción breve**
R.array	Una serie ordenada de elementos. Pueden ser strings, enteros o de recursos (TypedArray) **<string-array name= “dias_semana”>** **<item>lunes</item>** **<item>martes</item>** **</string-array>** **<integer-array name= “meses”>** **<item>1</item> <item>2</item><item>3</item>** **</integer-array>** **<array name= “fotos”>** **<item>@drawable/foto1</item>** **<item>@drawable/foto2</item>** **</array>**

Layouts

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.

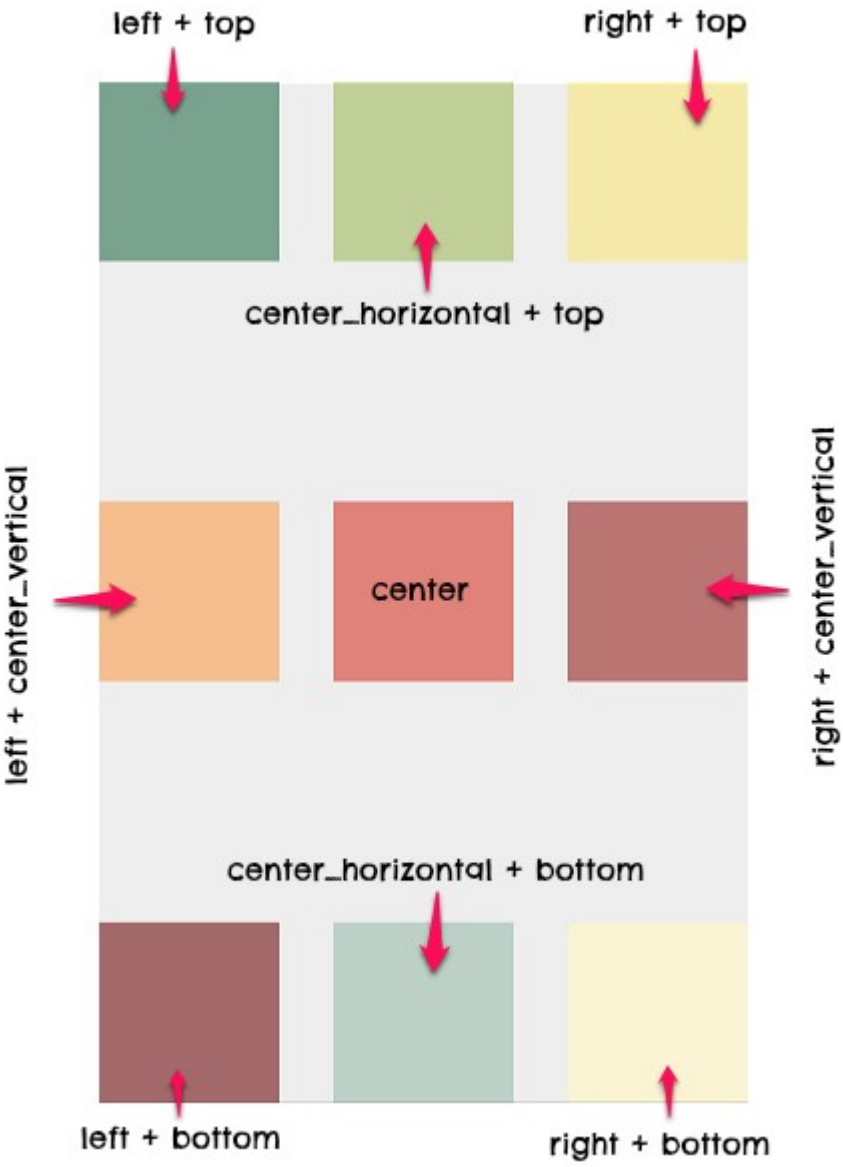
Sin embargo puedes añadir varios hijos con el fin de superponerlos, donde el último hijo agregado, es el que se muestra en la parte superior y el resto se pone por debajo en forma de pila.

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:

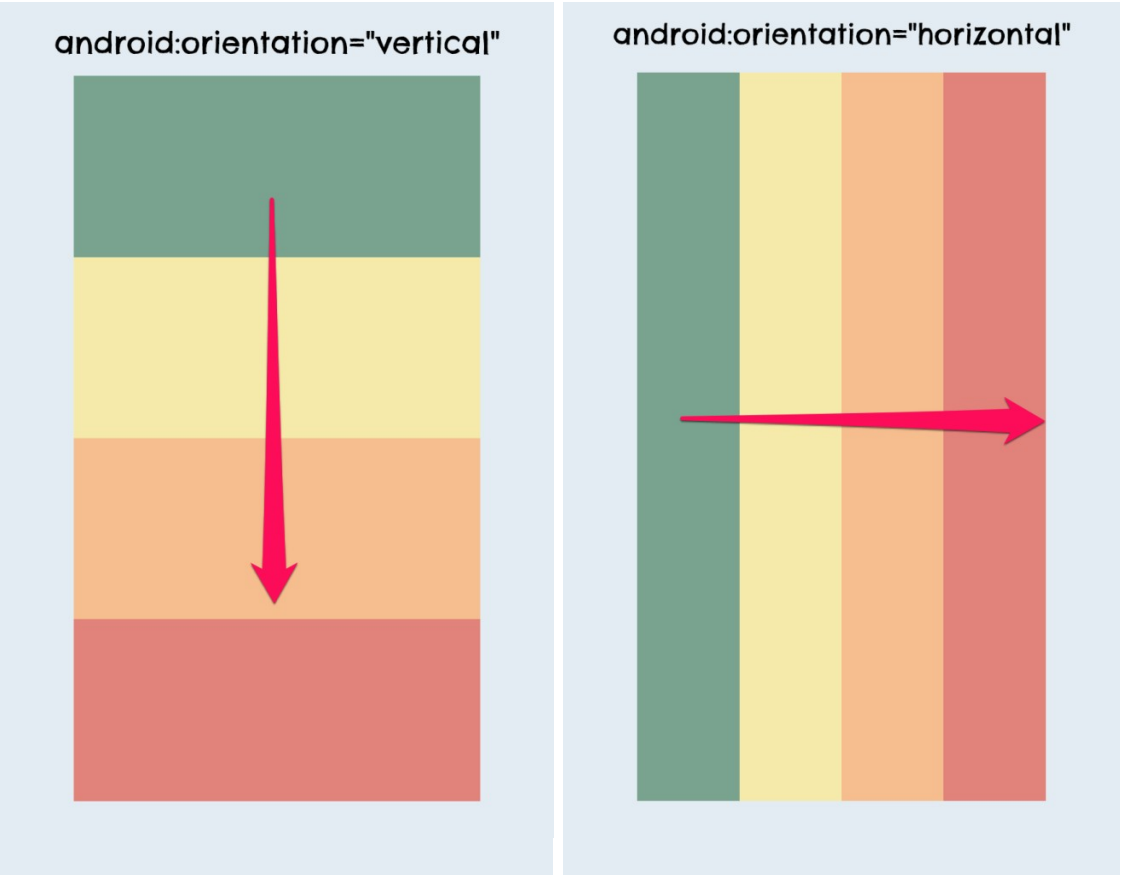
- top: Indica la parte superior del layout.
- left: Indica la parte izquierda del layout.
- right: Se refiere a la parte derecha del layout.
- bottom: Representa el límite inferior del layout.
- center_horizontal: Centro horizontal del layout.
- center_vertical: Alineación al centro vertical del layout.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

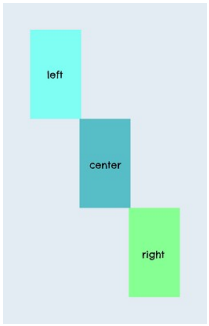


LinearLayout

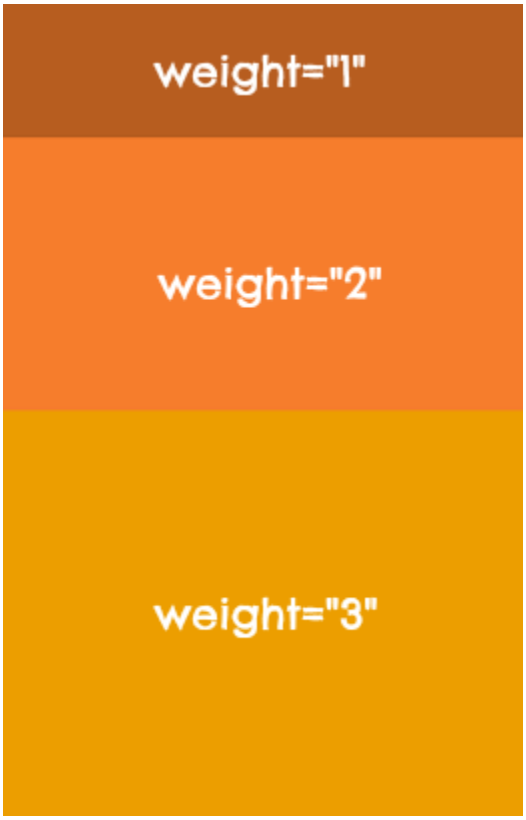
Un LinearLayout es un view group que distribuye sus hijos en una sola dimensión establecida. Es decir, todos organizados en una sola columna (vertical) o en una sola fila (horizontal). La orientación se elige a través del atributo android:orientation.



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.

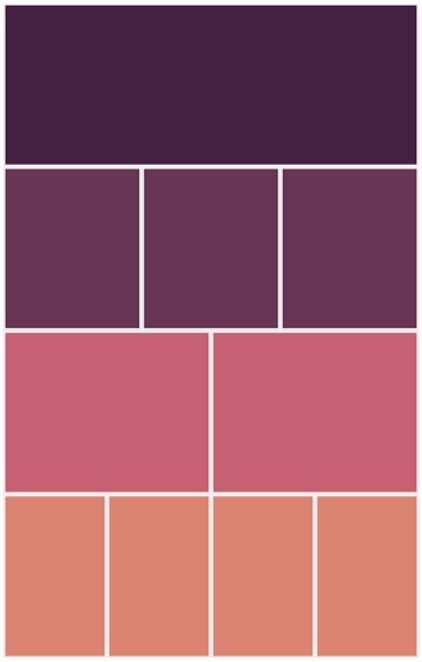


La anterior ilustración muestra tres views con pesos de 1, 2 y 3 respectivamente. La magnitud de sus alturas corresponde a su preponderancia. Matemáticamente, el espacio disponible total sería la suma de las alturas (6), por lo que 3 representa el 50%, 2 el 33,33% y 1 el 16,66%.

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:

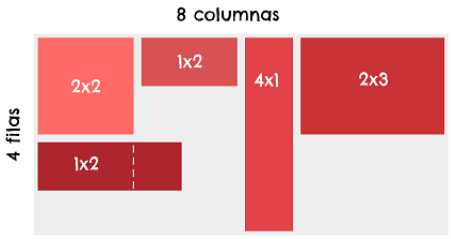
- **android:stretchColumns**: las columnas se expanden logrando absorber el espacio libre que no usan las otras columnas
- **android:shrinkColumns**: las columnas se contraen para lograr espacio entre otras columnas
- **android:collapseColums**: permite ocultar las tablas



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`.

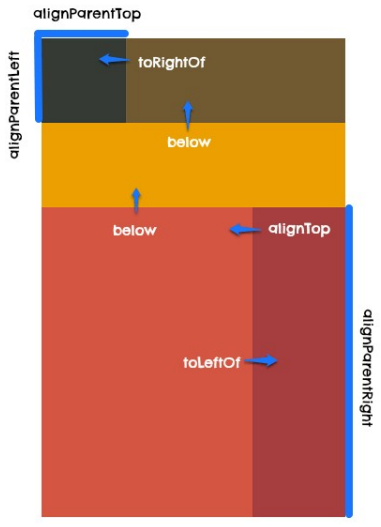
RelativeLayout

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:

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

IV. PROCEDIMIENTO

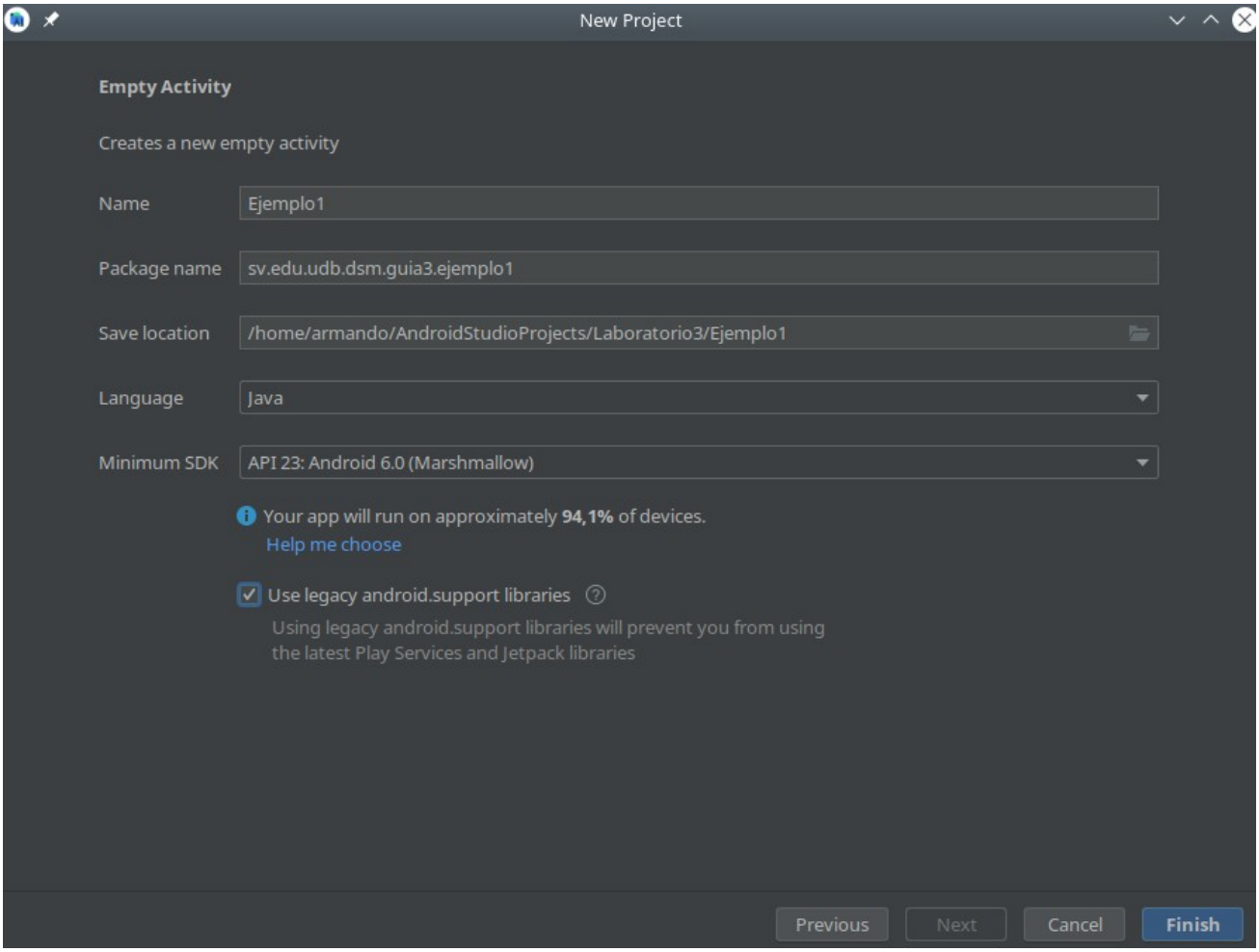
Ejemplo 1:

Incorporando recursos alternos al proyecto.
A continuación, vamos a incorporar recursos a nuestro proyecto para brindar soporte a una aplicación que permita mostrar 2 idiomas en sus opciones: español e inglés.

Esto dependerá de la configuración que el usuario tenga en su dispositivo móvil a través de las opciones **Administración general->Idioma y entrada->Idioma.**

1 Ejecuta Android Studio y crea un nuevo proyecto. En Project Template selecciona Empty Activity y presiona **Next**.

2 Como nombre **Ejemplo1**, en Package name digita **sv.edu.udb.dsm.guia3.ejemplo1** en **Save location** escoge la carpeta de tu preferencia, en Language elige Java y el Minimum SDK selecciona API 23: Android 6.0 (Marshmallow) y Presiona **Finish**.

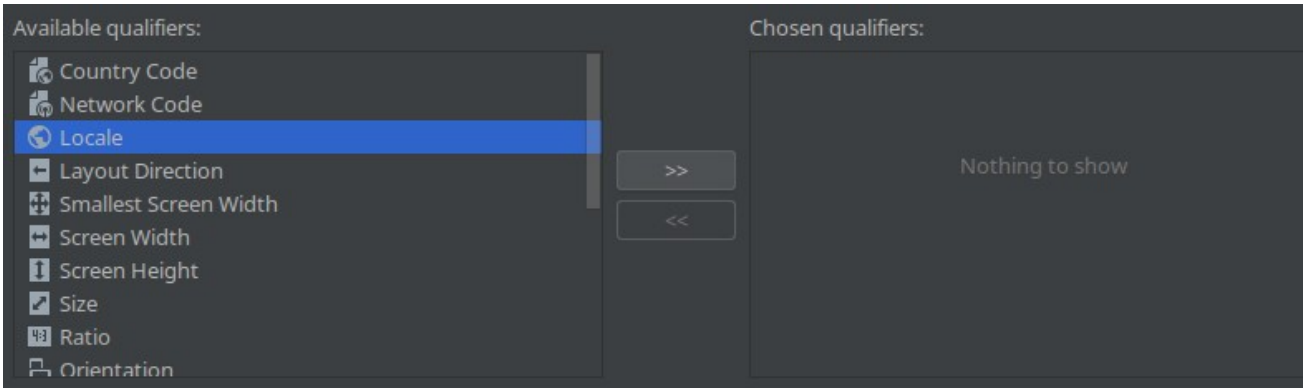


3 Haz **clic derecho** sobre la carpeta **res** y selecciona **new->Android Resource File**

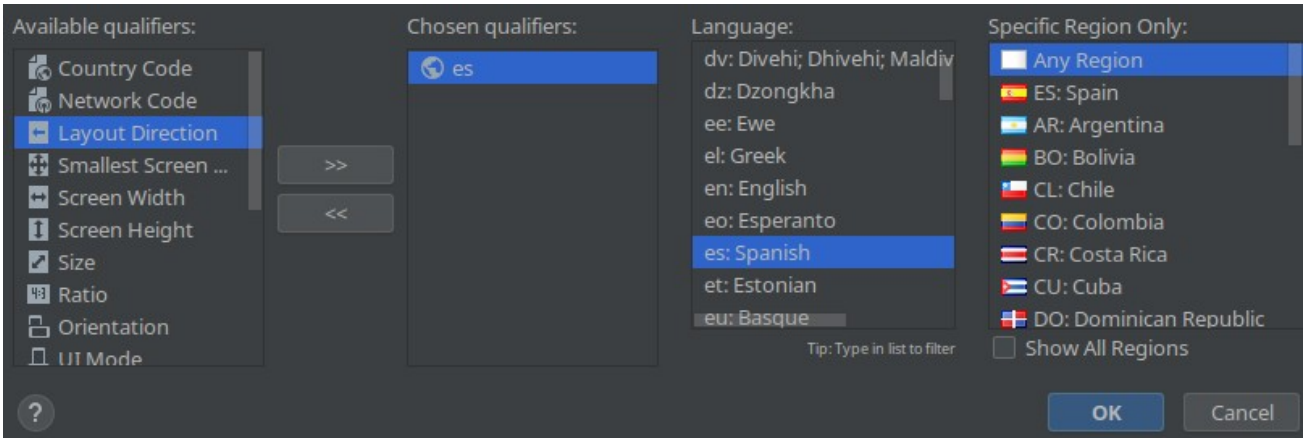
4 En la ventana que aparece a continuación establece los valores como aparece la siguiente imagen:

Debes digitar **strings** en minúscula en File Name)
Asegúrate que quede seleccionado **Values** en la opción **Resource Type**

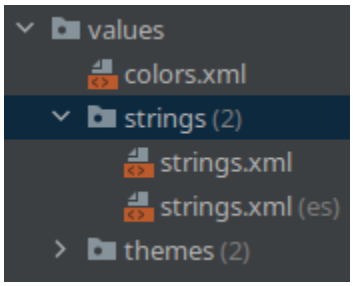
5 En la parte baja selecciona **Locale** haciendo doble clic sobre dicha opción o clic en el botón **>>**.



6 Ahora en el grupo **Language** selecciona **es:Spanish** y luego haz clic en el botón **OK**.



Observa como ahora tienes 2 archivos **strings.xml** en **res/values/strings** (Pero uno de ellos tiene un sufijo **es**)



7 Edita el archivo **strings.xml** (el que no tiene sufijo) Este archivo representa el contenido en idioma inglés. (archivo por default)
para que tenga el siguiente contenido:

```xml
<resources>
    <string name="app_name">Ejemplo1</string>
    <string name="saludo1">Hello user</string>
    <string name="saludo2">Welcome</string>
</resources>
```

8 Edita el archivo **strings.xml (es).** Con el siguiente contenido:
(Este archivo representa el contenido en español.)

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Ejemplo1</string>
    <string name="saludo1">Hola usuario</string>
    <string name="saludo2">Bienvenido</string>
</resources>
```

Importante.
Observa como las constantes **saludo1** y **saludo2** tienen exactamente el mismo nombre en ambos archivos. Sin embargo, el contenido de dichas constantes es el que cambia de idioma según el archivo donde se configura.

Tu aplicación puede ajustar su contenido en varios idiomas a mostrar dependiendo de cuantos archivos strings.xml hayas configurado (uno por cada idioma)

Para este ejemplo hemos configurado 2 idiomas: inglés (archivo strings.xml por default) y español (archivo strings.xml **con el sufijo es**).

Existen una serie de sufijos que representan a cada idioma, puedes consultarlos aquí:
https://es.wikipedia.org/wiki/ISO_639-1

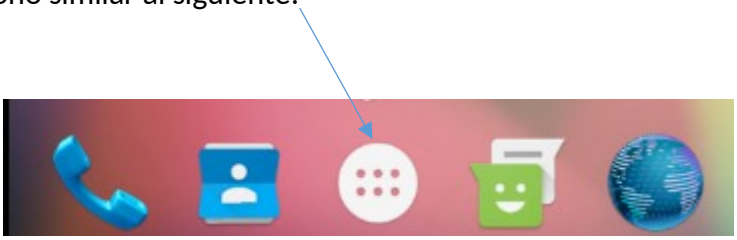
9 Ahora edita tu archivo de layout (en la carpeta **res/layout/activity_mail.xml**) de la siguiente forma:

```xml
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        tools:context=".MainActivity">
```

```
  <TextView
     android:text="@string/saludo1"
     android:textSize="32sp"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"/>
  <TextView
     android:text="@string/saludo2"
     android:textSize="32sp"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"/>

  </LinearLayout>

</android.support.constraint.ConstraintLayout>
```

Observa cómo se ha utilizado los recursos @string/saludo1 y @string/saludo2. De esta forma Android decidirá cual recurso utilizar en tiempo de ejecución dependiendo de la configuración que el usuario haya establecido en su dispositivo móvil.

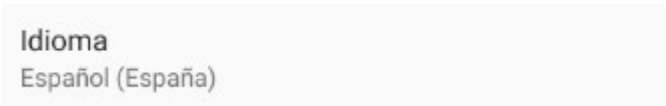
10 Configura el emulador para el idioma Ingles.
Haz clic en un icono similar al siguiente:



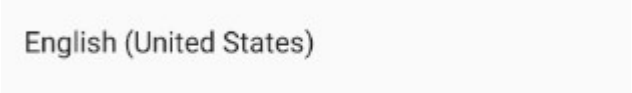
11 Luego en el icono de ajustes:

12 Ahora busca el icono (Idioma e introducción de texto)
(Language & input)
Reconócelo por el icono del mundo.

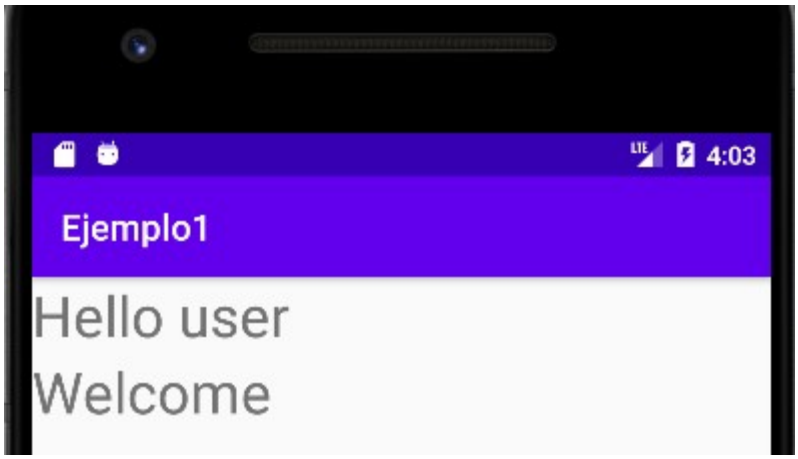
13 Ahora selecciona la opción **Idioma**:



14 Busca y selecciona el idioma English (United States)

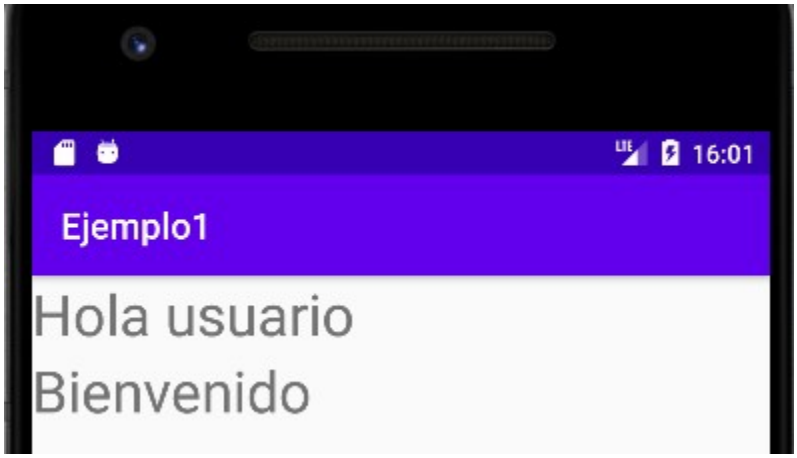


15 Ejecuta la app.
Veras que la aplicación muestra el contenido en ingles.



16 Vuelve a hacer los cambios en el emulador, pero ahora seleccionando el idioma Español (España).

17 Regrese a la aplicación o vuélvela a ejecutar y observaras que cambia su contenido a español.



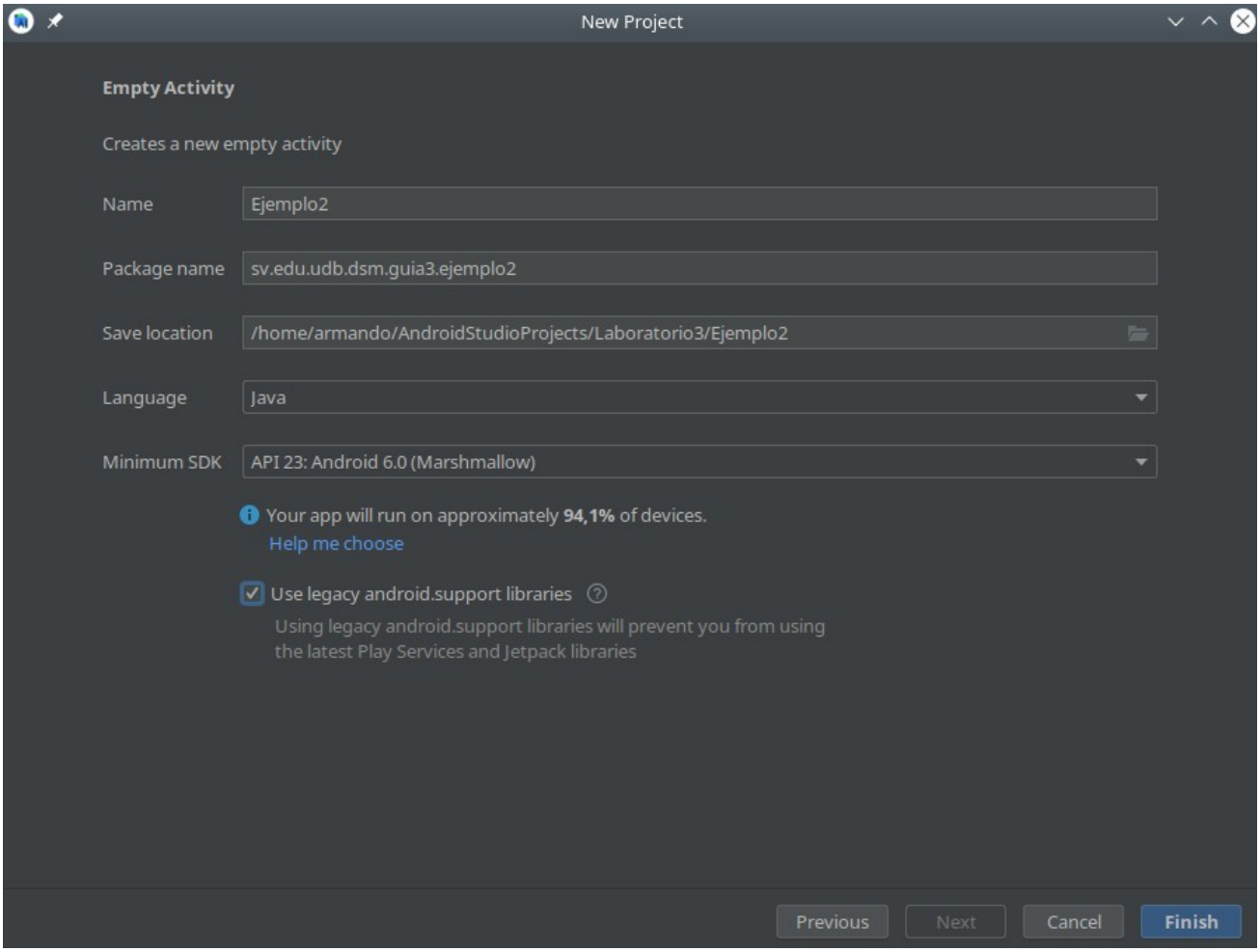
Ejemplo 2:

Trabajando con diferentes Layouts
En este ejemplo, vamos a trabajar con diferentes tipos de Layouts para ubicar elementos dentro de una activity en diferentes posiciones:

1. Ejecuta Android Studio y crea un nuevo proyecto. En Project Template selecciona Empty Activity y presiona **Next**.

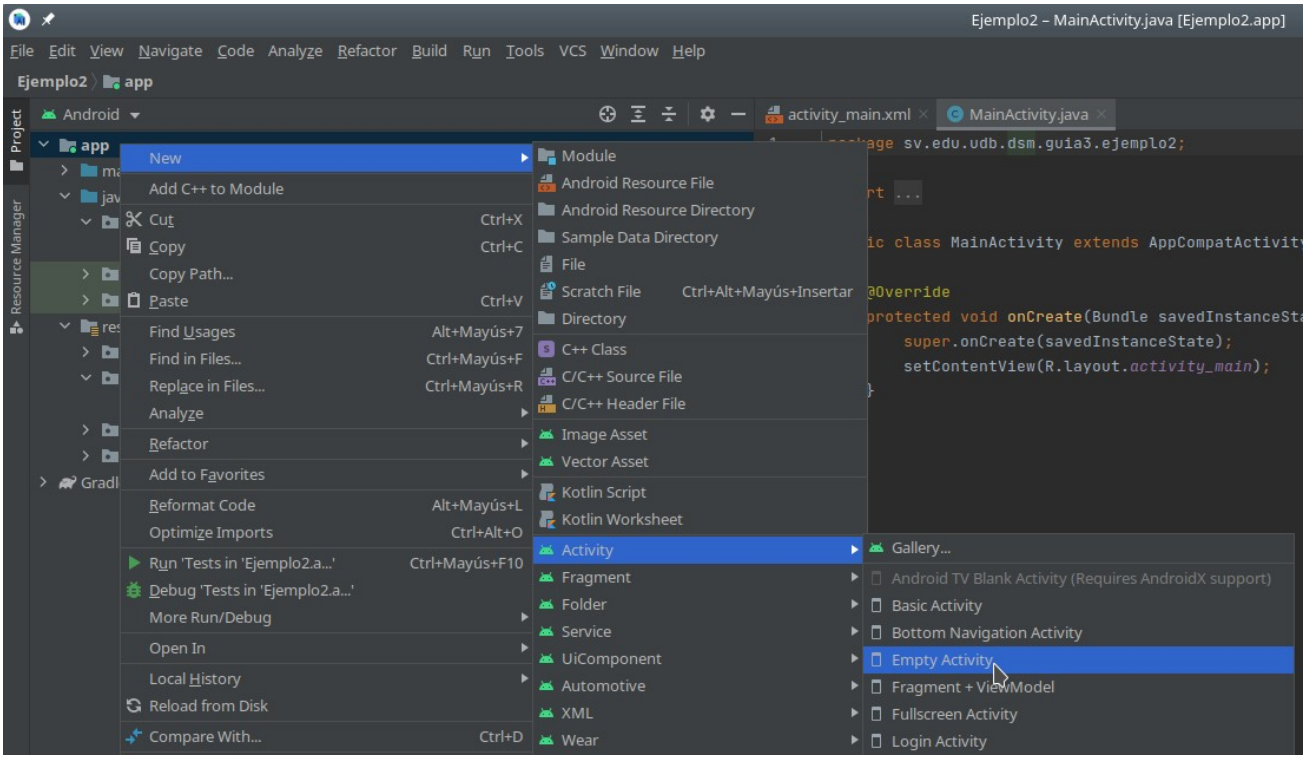
2. Como nombre **Ejemplo2**, en Package name escribe **sv.edu.udb.dsm.guia3.ejemplo2** en **Save location** escoge la carpeta de tu preferencia, en Language elige Java y el Minimum SDK selecciona API 23: Android 6.0 (Marshmallow) y Presiona **Finish**.

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

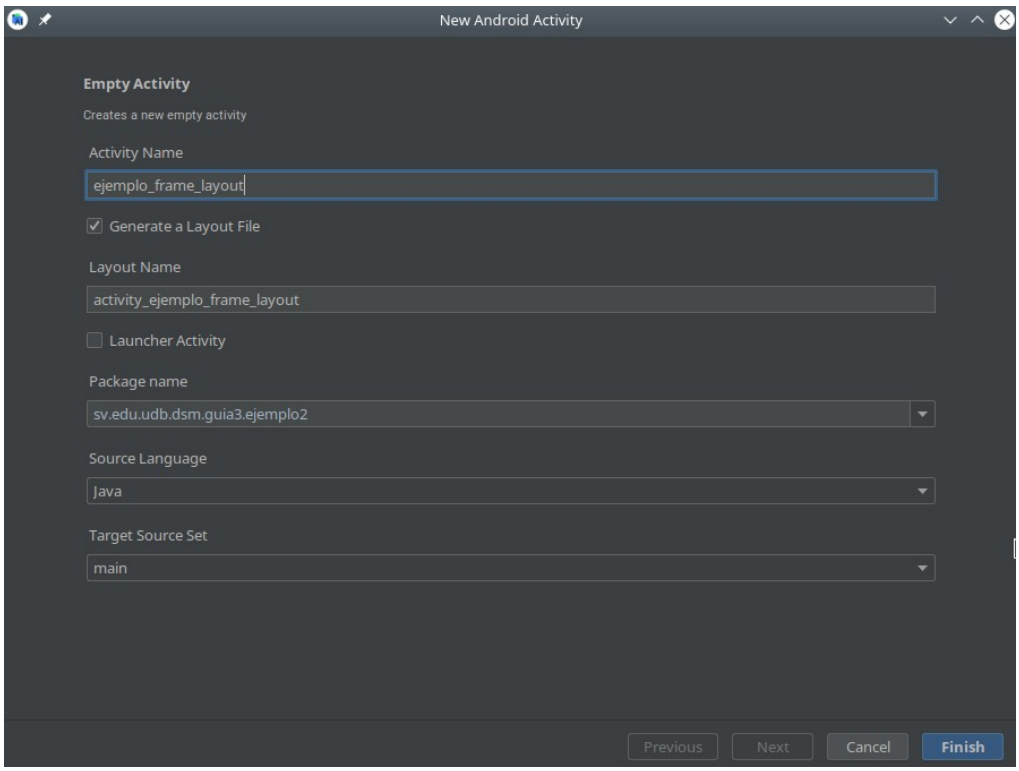


Ejemplo de FrameLayout

1. Agregue una nueva Activity a su aplicación, haciendo clic derecho sobre **App**, luego **New → Activity → Empty Activity**:



2. El nombre será **ejemplo_frame_layout**:



3. Damos clic en **Finish**, lo anterior va a generar una nueva actividad con sus dos componentes: la clase java y el archivo xml.

4. El contenido del archivo **activity_ejemplo_frame_layout.xml** es el siguiente:

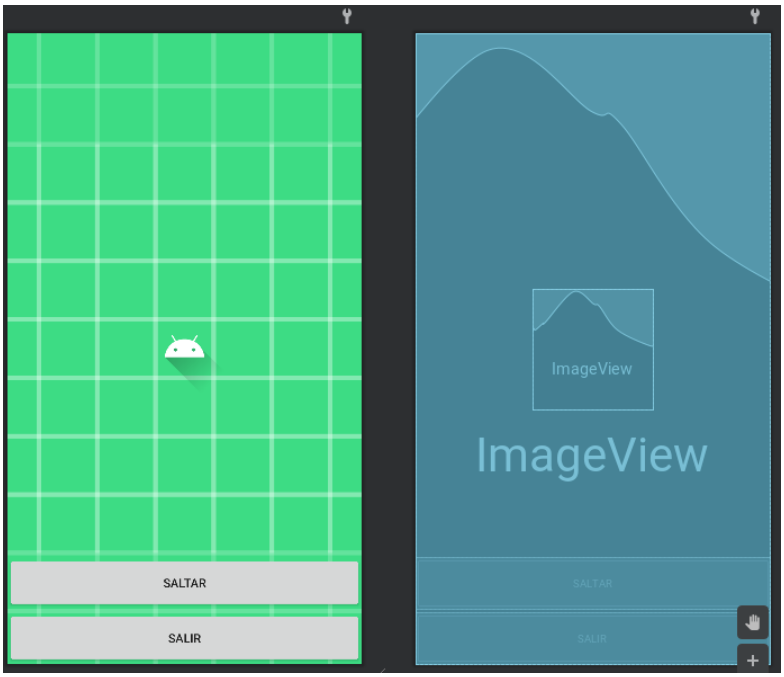
```
<?xml version="1.0" encoding="utf-8"?>
    <android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
       xmlns:app="http://schemas.android.com/apk/res-auto"
       xmlns:tools="http://schemas.android.com/tools"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       tools:context=".ejemplo_frame_layout">

       <FrameLayout
          android:layout_width="match_parent"
          android:layout_height="match_parent">

          <Button
             android:id="@+id/btnSaltar"
             android:layout_width="match_parent"
             android:layout_height="60dp"
             android:layout_gravity="bottom"
             android:layout_marginBottom="@android:dimen/notification_large_icon_height"
             android:text="Saltar" />

          <Button
             android:layout_width="match_parent"
             android:layout_height="60dp"
             android:text="Salir"
             android:layout_gravity="bottom"
             android:onClick="finalizarActividad"
             android:id="@+id/btnSalir"  />

          <ImageView
             android:id="@+id/imageView1"
             android:layout_width="match_parent"
             android:layout_height="match_parent"
             android:layout_gravity="top|center"
             android:scaleType="centerCrop"
             android:src="@drawable/ic_launcher_background" />

          <ImageView
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:id="@+id/imageView2"
             android:layout_gravity="center"
             android:src="@mipmap/ic_launcher"
             android:padding="16dp" />
```

```
    </FrameLayout>
</android.support.constraint.ConstraintLayout>
```

5. El código de **ejemplo_frame_layout.java** quedará como el siguiente:

activity_ejemplo_frame_layout.xml | ejemplo_frame_layout.java

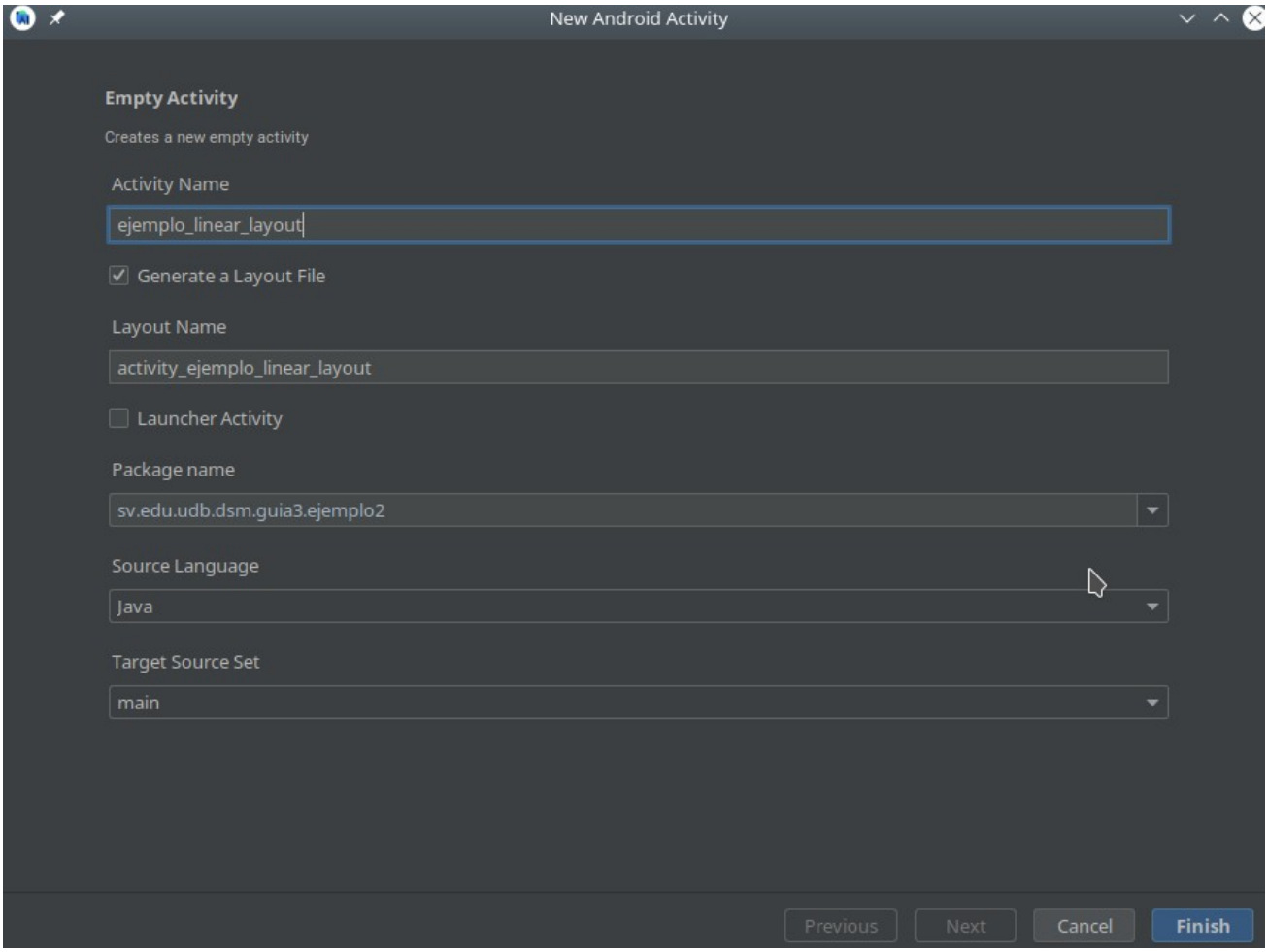
```java
package sv.edu.udb.dsm.guia3.ejemplo2;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;

public class ejemplo_frame_layout extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ejemplo_frame_layout);
    }

    public void finalizarActividad(View v){
        finish();
    }
}
```



Ejemplo de LinearLayout

1. Agregue una nueva Activity a su aplicación, haciendo clic derecho sobre **App**, luego **New → Activity → Empty Activity**:

2. El nombre será **ejemplo_linear_layout**:



3. El contenido del archivo **activity_ejemplo_linear_layout.xml** es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
    <android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
       xmlns:app="http://schemas.android.com/apk/res-auto"
       xmlns:tools="http://schemas.android.com/tools"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       tools:context=".ejemplo_linear_layout">
```

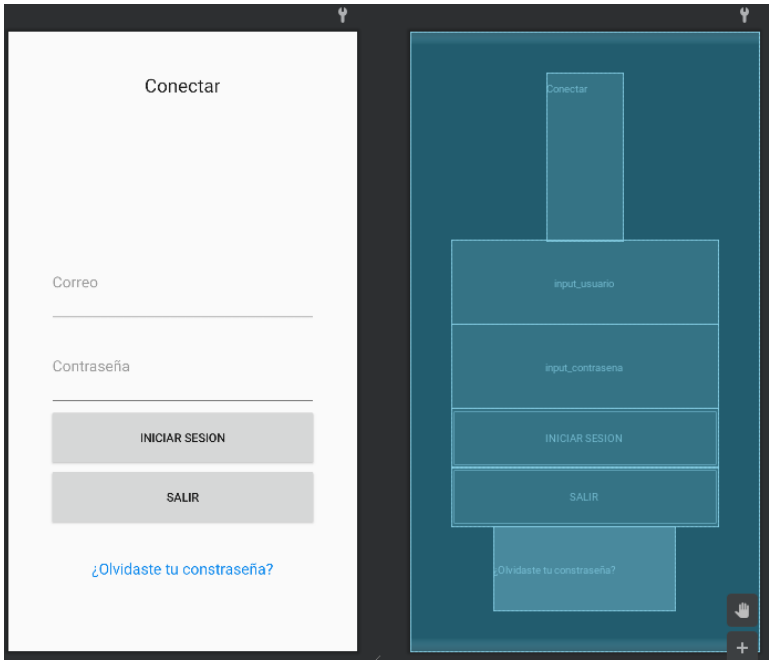
```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="48dp">

    <TextView
        android:id="@+id/texto_conectar"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_gravity="center_horizontal"
        android:layout_weight="2"
        android:text="Conectar"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <EditText
        android:id="@+id/input_usuario"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_gravity="center_horizontal"
        android:layout_weight="1"
        android:hint="Correo" />

    <EditText
        android:id="@+id/input_contrasena"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_gravity="center_horizontal"
        android:layout_weight="1"
        android:ems="10"
        android:hint="Contraseña"
        android:inputType="textPassword" />

    <Button
        android:id="@+id/boton_iniciar_sesion"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:layout_gravity="center_horizontal"
        android:layout_weight="0.1"
        android:text="Iniciar Sesion" />

    <Button
        android:layout_width="match_parent"
        android:layout_height="60dp"
        android:text="Salir"
        android:layout_gravity="bottom"
        android:onClick="finalizarActividad"
```

```
        android:layout_weight="0.1"
        android:id="@+id/btnSalir" />

    <TextView
        android:id="@+id/texto_olvidaste_contrasena"
        android:layout_width="wrap_content"
        android:layout_height="0dp"
        android:layout_gravity="center_horizontal"
        android:layout_weight="1"
        android:text="¿Olvidaste tu constraseña?"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="#0E8AEE" />
</LinearLayout>

</android.support.constraint.ConstraintLayout>
```

4. El código de **ejemplo_linear_layout.java** quedará como el siguiente:

activity_ejemplo_linear_layout.xml | ejemplo_linear_layout.java

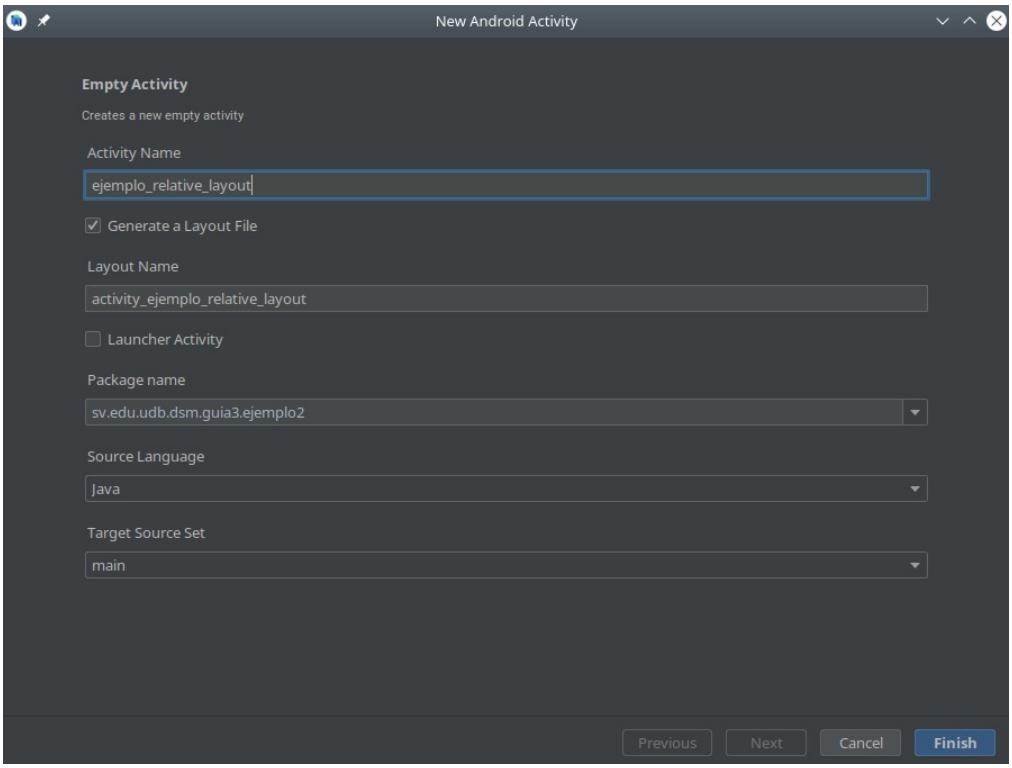
```java
package sv.edu.udb.dsm.guia3.ejemplo2;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;

public class ejemplo_linear_layout extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ejemplo_linear_layout);
    }

    public void finalizarActividad(View v){
        finish();
    }
}
```



Ejemplo de RelativeLayout

1. Agregue una nueva Activity a su aplicación, haciendo clic derecho sobre **App**, luego **New → Activity → Empty Activity**:

2. El nombre será **ejemplo_relative_layout**:



3. El contenido del archivo **activity_ejemplo_relative_layout.xml** es el siguiente:

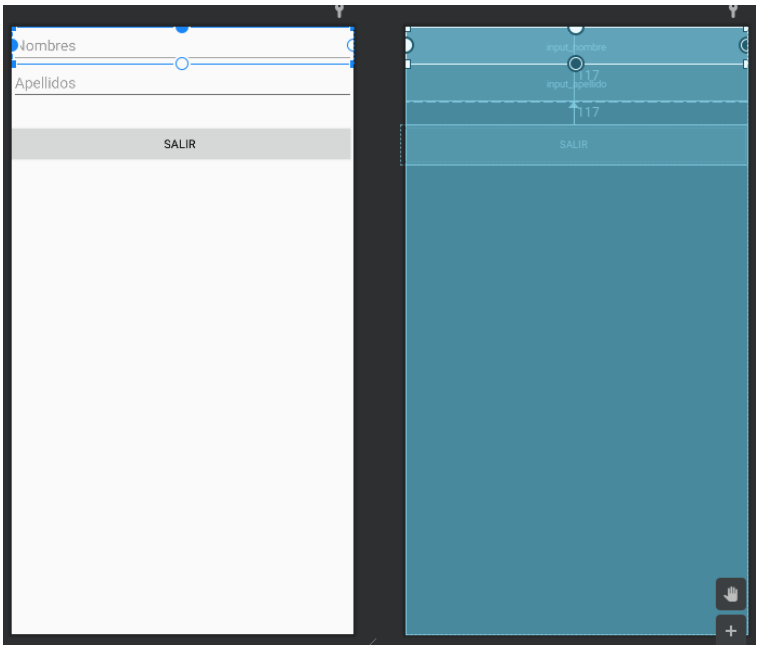
```
<?xml version="1.0" encoding="utf-8"?>
    <android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
       xmlns:app="http://schemas.android.com/apk/res-auto"
       xmlns:tools="http://schemas.android.com/tools"
       android:layout_width="match_parent"
       android:layout_height="match_parent"
       tools:context=".ejemplo_relative_layout">

       <RelativeLayout
          android:layout_width="match_parent"
          android:layout_height="match_parent">

          <EditText
             android:id="@+id/input_nombre"
             android:layout_width="match_parent"
             android:layout_height="wrap_content"
             android:layout_alignParentLeft="true"
             android:layout_alignParentStart="true"
             android:layout_alignParentTop="true"
             android:ems="10"
             android:hint="Nombres"
             android:inputType="textPersonName" />

          <EditText
             android:id="@+id/input_apellido"
             android:layout_width="match_parent"
             android:layout_height="wrap_content"
             android:layout_alignParentLeft="true"
             android:layout_alignParentStart="true"
             android:layout_below="@+id/input_nombre"
             android:ems="10"
             android:hint="Apellidos"
             android:inputType="textPersonName" />

          <Button
             android:id="@+id/btnSalir"
             android:layout_width="match_parent"
             android:layout_height="wrap_content"
             android:layout_below="@+id/input_apellido"
             android:layout_alignParentStart="true"
             android:layout_alignParentLeft="true"
             android:layout_alignParentTop="true"
             android:layout_marginStart="-7dp"
             android:layout_marginLeft="-7dp"
             android:layout_marginTop="117dp"
```

```
        android:text="Salir"
        android:onClick="finalizarActividad"/>

    </RelativeLayout>

</android.support.constraint.ConstraintLayout>
```

4. El código de **ejemplo_relative_layout.java** quedará como el siguiente:

activity_ejemplo_relative_layout.xml | ejemplo_relative_layout.java

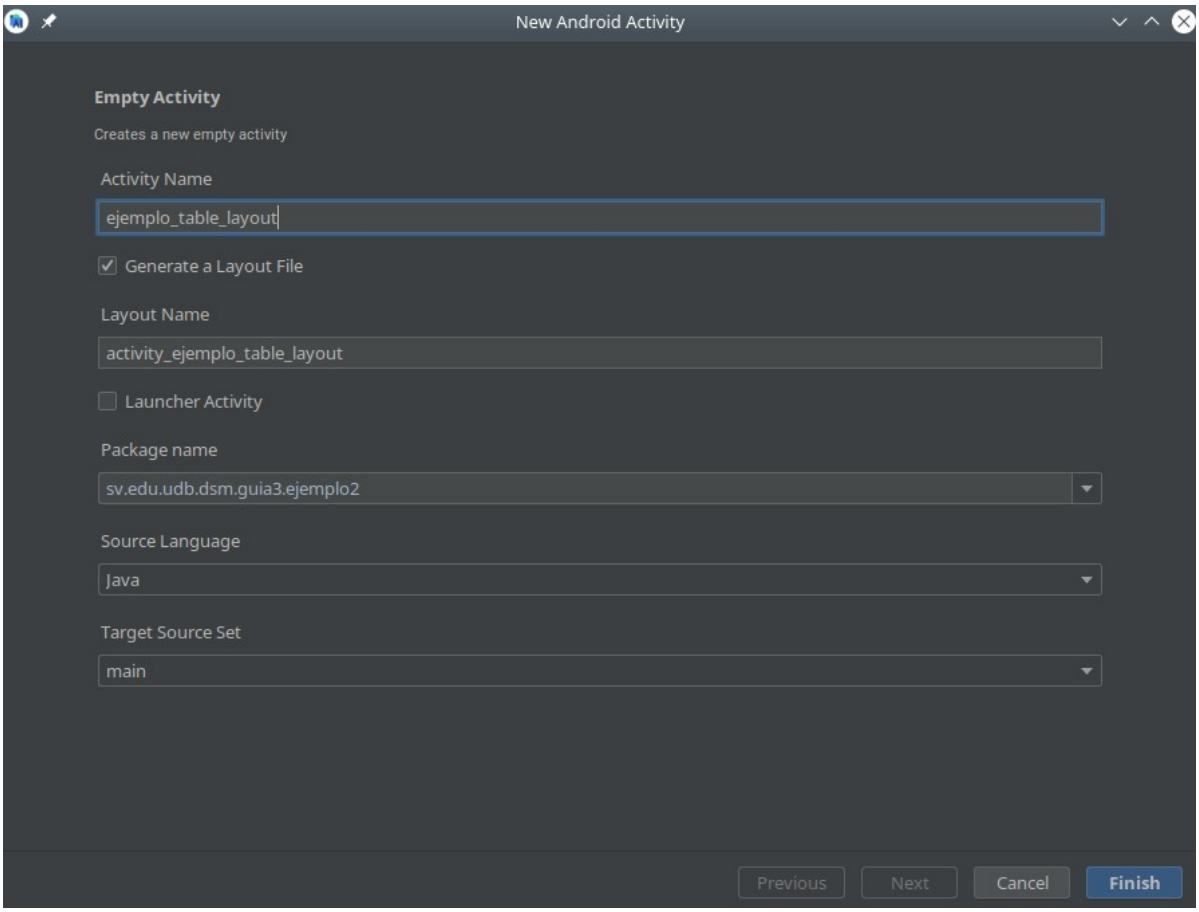
```
1  package sv.edu.udb.dsm.guia3.ejemplo2;
2
3  import ...
6
7  public class ejemplo_relative_layout extends AppCompatActivity {
8
9      @Override
10     protected void onCreate(Bundle savedInstanceState) {
11         super.onCreate(savedInstanceState);
12         setContentView(R.layout.activity_ejemplo_relative_layout);
13     }
14
15     public void finalizarActividad(View v){
16         finish();
17     }
18 }
```



Ejemplo de TableLayout

1. Agregue una nueva Activity a su aplicación, haciendo clic derecho sobre **App**, luego **New → Activity → Empty Activity**:

2. El nombre será **ejemplo_table_layout**:



3. El contenido del archivo **activity_ejemplo_table_layout.xml** es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
    <android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto"
      xmlns:tools="http://schemas.android.com/tools"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      tools:context=".ejemplo_table_layout">

      <TableLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:padding="16dp"
        android:stretchColumns="1">

        <TableRow android:background="#128675">

            <TextView
                android:layout_column="0"
                android:padding="3dip"
                android:text="Producto"
                android:textColor="@android:color/white" />

            <TextView
                android:layout_column="2"
                android:padding="3dip"
                android:text="Subtotal"
                android:textColor="@android:color/white" />
        </TableRow>

        <TableRow>

            <TextView
                android:layout_column="0"
                android:padding="3dip"
                android:text="Jabón de manos x 1" />

            <TextView
                android:layout_column="2"
                android:gravity="left"
                android:padding="3dip"
                android:text="$2" />
        </TableRow>

        <TableRow>

            <TextView
                android:layout_column="0"
                android:padding="3dip"
                android:text="Shampoo Monster x 1" />

            <TextView
                android:layout_column="2"
                android:gravity="left"
                android:padding="3dip"
                android:text="$10" />
        </TableRow>

        <TableRow
            android:layout_width="match_parent"
```

```
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_column="0"
        android:padding="3dip"
        android:text="Pastas Duria x 2" />

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_column="2"
        android:gravity="left"
        android:padding="3dip"
        android:text="$18" />
</TableRow>

<TableRow
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_column="0"
        android:padding="3dip"
        android:text="Detergente Limpiadin x 1" />

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_column="2"
        android:gravity="left"
        android:padding="3dip"
        android:text="$13,4" />
</TableRow>

<View
    android:layout_height="2dip"
    android:background="#FF909090" />

<TableRow>
```

```
        <TextView
            android:layout_column="1"
            android:padding="3dip"
            android:text="Subtotal"
            android:textColor="#128675" />

        <TextView
            android:id="@+id/textView7"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="2"
            android:layout_gravity="left"
            android:gravity="right"
            android:padding="3dip"
            android:text="$43,4" />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:id="@+id/textView6"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="1"
            android:padding="3dip"
            android:text="Costo envío"
            android:textColor="#128675" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="2"
            android:layout_gravity="left"
            android:gravity="right"
            android:padding="3dip"
            android:text="$10" />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:id="@+id/textView8"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

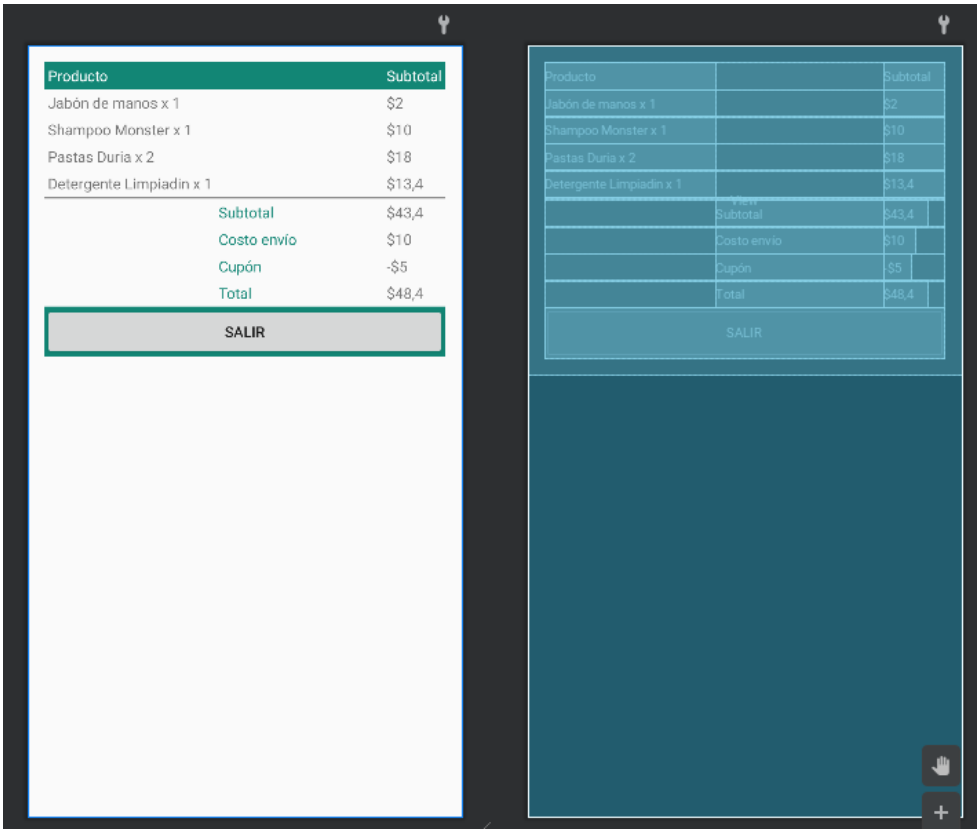
```
            android:layout_column="1"
            android:padding="3dip"
            android:text="Cupón"
            android:textColor="#128675" />

        <TextView
            android:id="@+id/textView9"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="2"
            android:layout_gravity="left"
            android:gravity="right"
            android:padding="3dip"
            android:text="-$5" />
    </TableRow>

    <TableRow
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:id="@+id/textView5"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="1"
            android:padding="3dip"
            android:text="Total"
            android:textColor="#128675" />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="2"
            android:layout_gravity="left"
            android:gravity="right"
            android:padding="3dip"
            android:text="$48,4" />
    </TableRow>

    <TableRow android:background="#128675">
        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight=".50"
            android:text="Salir"
            android:onClick="finalizarActividad"/>
    </TableRow>
</TableLayout>
```

```
</android.support.constraint.ConstraintLayout>
```

4. El código de **ejemplo_table_layout.java** quedará como el siguiente:

activity_ejemplo_table_layout.xml | ejemplo_table_layout.java

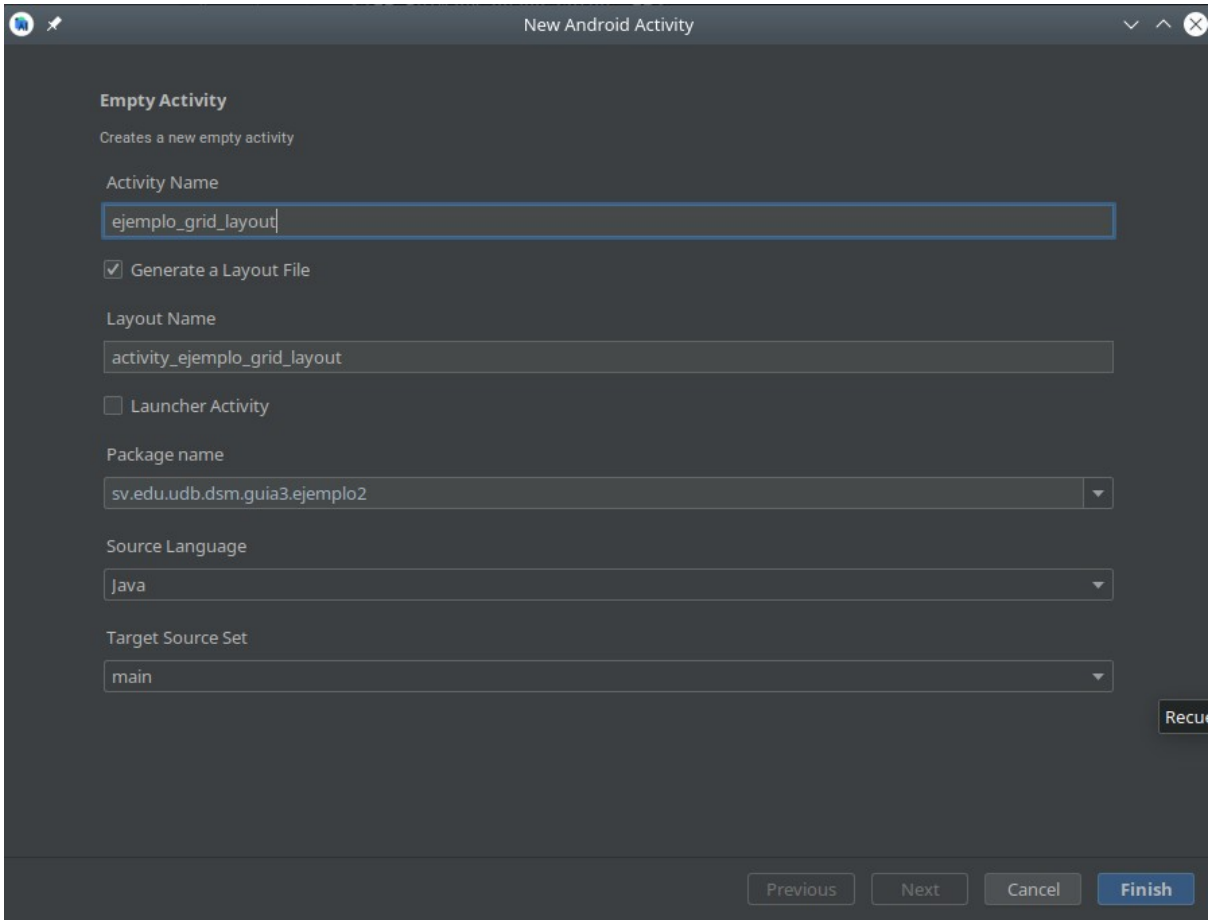
```java
package sv.edu.udb.dsm.guia3.ejemplo2;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;

public class ejemplo_table_layout extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ejemplo_table_layout);
    }

    public void finalizarActividad(View v){
        finish();
    }
}
```



Ejemplo de GridLayout

1 Agregue una nueva Activity a su aplicación, haciendo clic derecho sobre **App**, luego **New → Activity → Empty Activity**:

2 El nombre será **ejemplo_grid_layout**:



3 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>
```

4 El contenido del archivo **activity_ejemplo_grid_layout.xml** es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ejemplo_grid_layout">

    <GridLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:alignmentMode="alignBounds"
        android:columnCount="4"
        android:rowCount="4">

        <TextView
            android:id="@+id/numero_7"
            style="@style/AppTheme.BotonCalculadora.Azul"
            android:layout_column="0"
            android:layout_row="0"
            android:text="7" />

        <TextView
            android:id="@+id/numero_8"
            style="@style/AppTheme.BotonCalculadora.Azul"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="1"
            android:layout_row="0"
            android:text="8" />

        <TextView
            android:id="@+id/numero_9"
            style="@style/AppTheme.BotonCalculadora.Azul"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_column="2"
            android:layout_row="0"
            android:text="9" />

        <TextView
            android:id="@+id/numero_4"
            style="@style/AppTheme.BotonCalculadora.Azul"
            android:layout_height="wrap_content"
            android:layout_column="0"
            android:layout_row="1"
            android:text="4" />

        <TextView
            android:id="@+id/numero_5"
            style="@style/AppTheme.BotonCalculadora.Azul"
            android:layout_width="wrap_content"
```

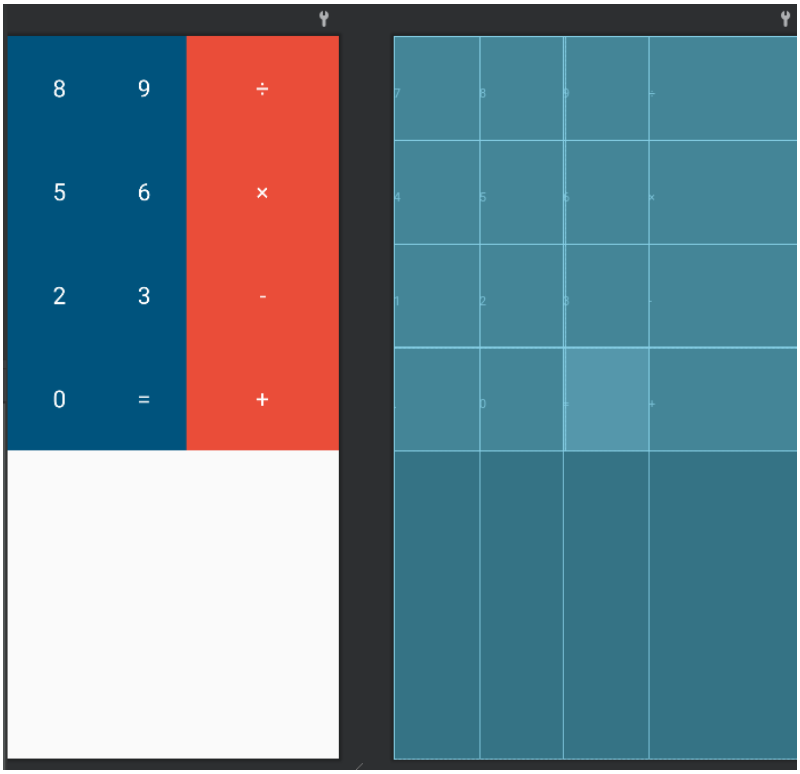
```
    android:layout_height="wrap_content"
    android:layout_column="1"
    android:layout_row="1"
    android:text="5" />

<TextView
    android:id="@+id/numero_6"
    style="@style/AppTheme.BotonCalculadora.Azul"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="2"
    android:layout_row="1"
    android:text="6" />

<TextView
    android:id="@+id/signo_por"
    style="@style/AppTheme.BotonCalculadora.Rojo"
    android:layout_column="3"
    android:layout_gravity="fill"
    android:layout_row="1"
    android:gravity="center"
    android:text="×" />

<TextView
    android:id="@+id/textView10"
    style="@style/AppTheme.BotonCalculadora.Rojo"
    android:layout_column="3"
    android:layout_gravity="fill_horizontal"
    android:layout_row="0"
    android:text="÷" />

<TextView
    android:id="@+id/numero_1"
    style="@style/AppTheme.BotonCalculadora.Azul"
    android:layout_column="0"
    android:layout_row="2"
    android:text="1" />

<TextView
    android:id="@+id/numero_2"
    style="@style/AppTheme.BotonCalculadora.Azul"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="1"
    android:layout_row="2"
    android:text="2" />

<TextView
    android:id="@+id/numero_3"
    style="@style/AppTheme.BotonCalculadora.Azul"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_column="2"
    android:layout_row="2"
```

```
        android:text="3" />

    <TextView
        android:id="@+id/signo_menos"
        style="@style/AppTheme.BotonCalculadora.Rojo"
        android:layout_column="3"
        android:layout_gravity="fill_horizontal"
        android:layout_row="2"
        android:gravity="center"
        android:text="-" />

    <TextView
        android:id="@+id/punto"
        style="@style/AppTheme.BotonCalculadora.Azul"
        android:layout_column="0"
        android:layout_gravity="fill_horizontal"
        android:layout_row="3"
        android:gravity="center_horizontal"
        android:text="." />

    <TextView
        android:id="@+id/cero"
        style="@style/AppTheme.BotonCalculadora.Azul"
        android:layout_column="1"
        android:layout_row="3"
        android:text="0" />

    <TextView
        android:id="@+id/signo_igual"
        style="@style/AppTheme.BotonCalculadora.Azul"
        android:layout_column="2"
        android:layout_gravity="fill_horizontal"
        android:layout_row="3"
        android:text="=" />

    <TextView
        android:id="@+id/signo_mas"
        style="@style/AppTheme.BotonCalculadora.Rojo"
        android:layout_column="3"
        android:layout_gravity="fill_horizontal"
        android:layout_row="3"
        android:text="+" />

  </GridLayout>

</android.support.constraint.ConstraintLayout>
```

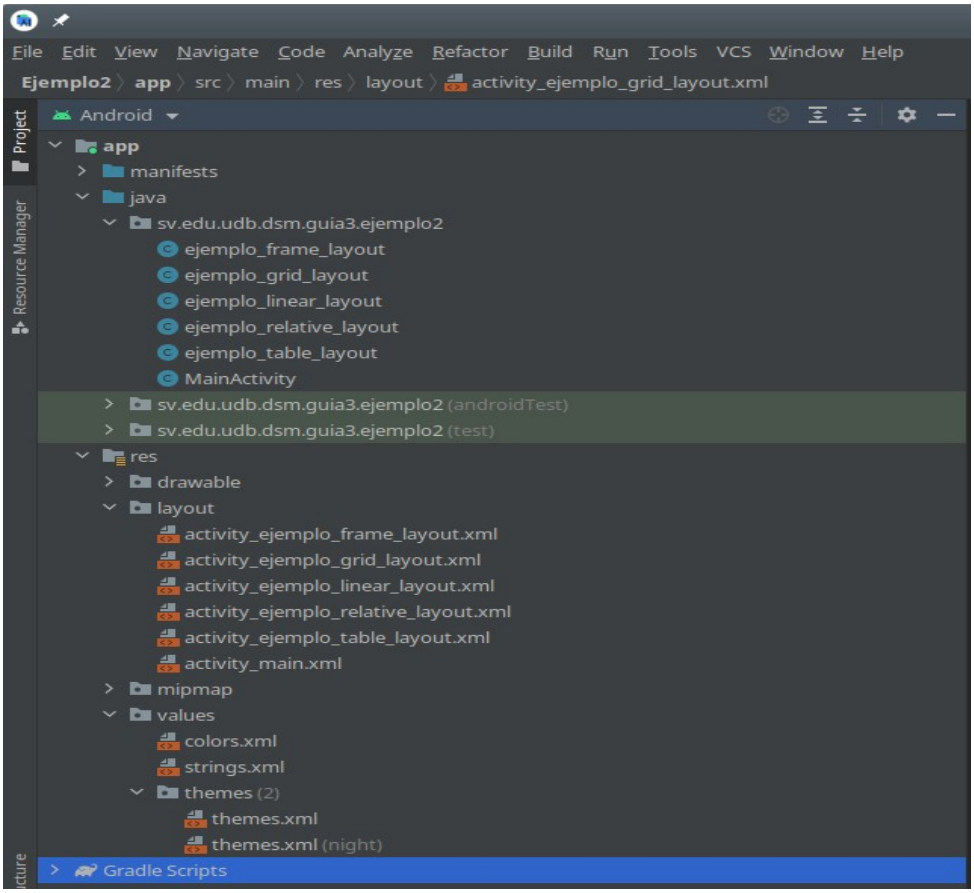
5 El código de **ejemplo_table_layout.java** quedará como el siguiente:

activity_ejemplo_grid_layout.xml | ejemplo_grid_layout.java

```java
package sv.edu.udb.dsm.guia3.ejemplo2;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class ejemplo_grid_layout extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_ejemplo_grid_layout);
    }
}
```



Hasta el momento se tiene una estructura de nuestro proyecto como la siguiente:



Vamos a trabajar con la actividad que se crea por defecto en el proyecto para llamar a todas Actividades que se han creado hasta el momento, para ello, vamos a realizar lo siguiente:

El contenido del archivo **activity_main.xml** es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
    <android.support.constraint.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto"
      xmlns:tools="http://schemas.android.com/tools"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      tools:context=".MainActivity">

      <Button
        android:id="@+id/btnFrame"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="7dp"
        android:layout_marginTop="184dp"
```

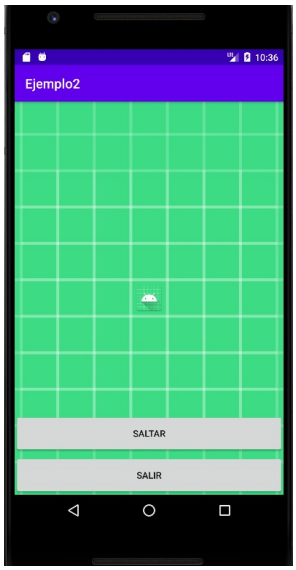
```
        android:onClick="onClickFrame"
        android:text="Frame Layout"
        app:layout_constraintStart_toStartOf="@+id/btnRelative"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/btnLinear"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="22dp"
        android:onClick="onClickLinear"
        android:text="Linear Layout"
        app:layout_constraintStart_toStartOf="@+id/btnFrame"
        app:layout_constraintTop_toBottomOf="@+id/btnFrame" />

    <Button
        android:id="@+id/btnRelative"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="118dp"
        android:layout_marginTop="23dp"
        android:onClick="onClickRelative"
        android:text="Relative Layout"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnLinear" />

    <Button
        android:id="@+id/btnTable"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="23dp"
        android:layout_marginEnd="8dp"
        android:onClick="onClickTable"
        android:text="Table Layout"
        app:layout_constraintEnd_toEndOf="@+id/btnRelative"
        app:layout_constraintTop_toBottomOf="@+id/btnRelative" />

    <Button
        android:id="@+id/btnGrid"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="24dp"
        android:onClick="onClickGrid"
        android:text="Grid Layout"
        app:layout_constraintStart_toStartOf="@+id/btnTable"
        app:layout_constraintTop_toBottomOf="@+id/btnTable" />
```

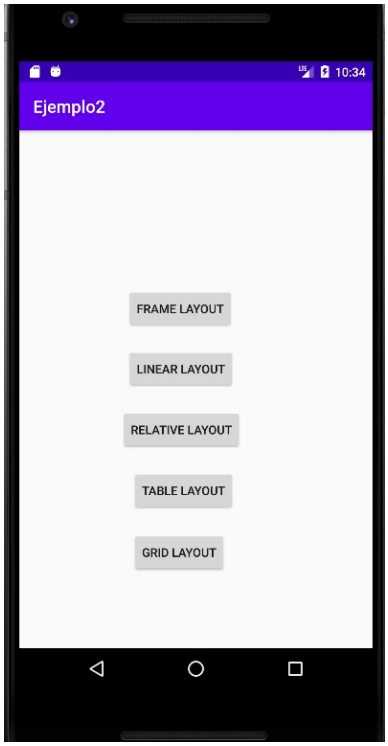
```
</android.support.constraint.ConstraintLayout>
```

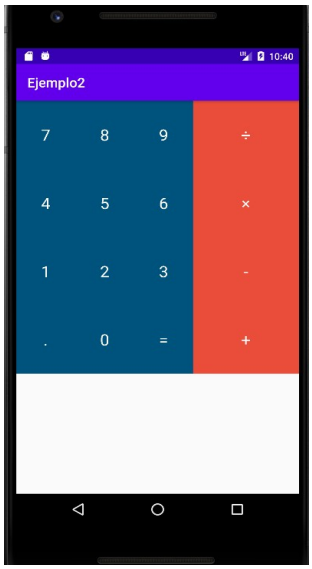
El código de **MainActivity.java** quedará como el siguiente:

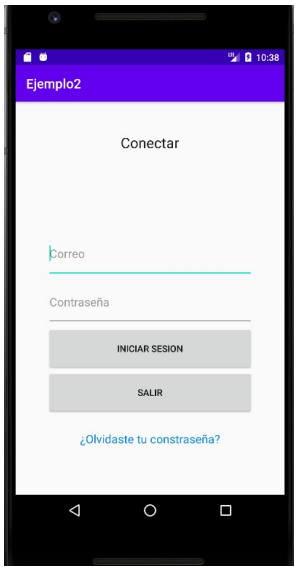
```java
package sv.edu.udb.dsm.guia3.ejemplo2;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class MainActivity extends AppCompatActivity {

    private Button btFrame, btLinear,btRelative,btGrid,btTable;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void onClickFrame(View v){
        Intent llamar = new Intent( packageContext: this,ejemplo_frame_layout.class);
        startActivity(llamar);
    }

    public void onClickLinear(View v){
        Intent llamar = new Intent( packageContext: this,ejemplo_linear_layout.class);
        startActivity(llamar);
    }

    public void onClickRelative(View v){
        Intent llamar = new Intent( packageContext: this,ejemplo_relative_layout.class);
        startActivity(llamar);
    }

    public void onClickTable(View v){
        Intent llamar = new Intent( packageContext: this,ejemplo_table_layout.class);
        startActivity(llamar);
    }

    public void onClickGrid(View v){
        Intent llamar = llamar = new Intent( packageContext: this,ejemplo_grid_layout.class);
        startActivity(llamar);
    }
}
```

Por último, ejecutar la aplicación y observe su funcionamiento:

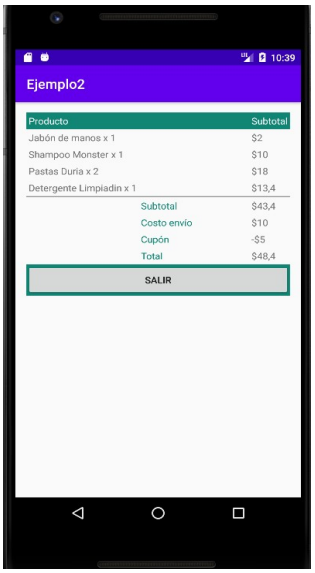

10:36
Ejemplo2
SALTAR
SALIR


10:34
Ejemplo2
FRAME LAYOUT
LINEAR LAYOUT
RELATIVE LAYOUT
TABLE LAYOUT
GRID LAYOUT


10:40
Ejemplo2
7 8 9 ÷
4 5 6 ×
1 2 3 -
. 0 = +


10:38
Ejemplo2
Conectar
Correo
Contraseña
INICIAR SESION
SALIR
¿Olvidaste tu constraseña?


10:38
Ejemplo2
Nombres
Apellidos
SALIR


10:39
Ejemplo2
Producto Subtotal
Jabón de manos x 1 $2
Shampoo Monster x 1 $10
Pastas Duria x 2 $18
Detergente Limpiadin x 1 $13,4
Subtotal $43,4
Costo envío $10
Cupón -$5
Total $48,4
SALIR

V. DISCUSION DE RESULTADOS

1. Agrega un archivo strings al Ejemplo 1 para brindar soporte en el idioma francés traduciendo los valores saludo1 y saludo2. Prueba la aplicación haciendo la configuración adecuada para el idioma francés.
2. 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.
3. 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.
4. Agregue el botón Salir al GridLayout y funcionamiento similar al resto.
5. Mejore la distribución de los controles de la Actvidad principal de manera de que tenga una mejor presentación a la vista del usuario, puede utilizar cualquier tipo de Layout que considere.

VII. BIBLIOGRAFIA

- developer.android. (s. f.). Información general sobre los recursos de las aplicaciones, de https://developer.android.com/guide/topics/resources/providing-resources