Asignatura: Application Development for Mobile Devices.
Tarea 2: Plantillas `Layout`.

Ejecutar cada uno de los códigos de las plantillas siguientes, relacionadas con su plantilla `Layout` correspondiente y el código Java

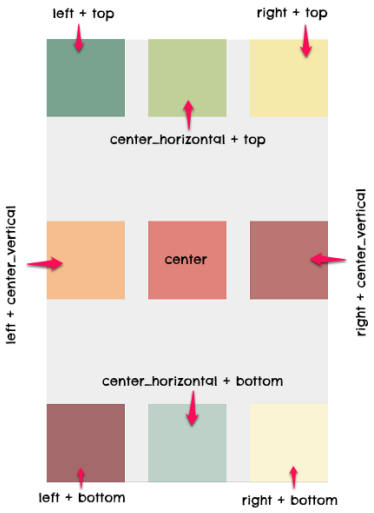
Las plantillas `Layout` son elementos no visuales que controlan la distribución, la posición y las dimensiones de los componentes gráficos que se insertan en su interior. Estas plantillas heredan de la clase `ViewGroup`, al igual que otros contenedores.

El código básico de Java para todas las plantillas es el siguiente:

```
importandroid.app.Activity;
importandroid.os.Bundle;
importandroid.view.View;
importandroid.widget.*;
publicclassMainActivityextendsActivity{

      publicvoidonCreate(Bundle b) {
            super.onCreate(b);
            setContentView(R.layout.activity_main);
      }
}
```

- **`FrameLayout`**

Ésta es la más sencilla de las plantillas `Layout` de Android. Un `FrameLayout` coloca todos sus componentes hijos alineados con su esquina superior izquierda y cada componente quedará oculto por el componente siguiente, a menos que este último tenga transparencia. Por ello, esta plantilla se utilizapara mostrar un único componente en su interior, como un contenedor (placeholder) sencillo para un sólo elemento sustituible, por ejemplo una imagen.



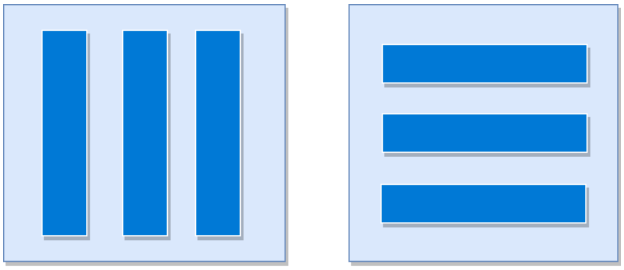
Los componentes incluidos en un `FrameLayout` deben incluir sus propiedades `android:layout_width` y `android:layout_height`, que pueden tomar los valores `match_parent` (para que el componente tome la dimensión de su layout contenedor) o `wrap_content` (para que el componente tome la dimensión de su contenido).

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
android:layout_width="match_parent"
android:layout_height="match_parent">

<EditTextandroid:id="@+id/xet1"
android:layout_width="match_parent"
android:layout_height="match_parent" />

</FrameLayout>
```

- **LinearLayout**

Esta plantilla acomoda en una columna, uno tras otro, todos sus elementos componentes en forma horizontal o vertical, según se establezca su propiedad `android:orientation`.



Al igual que en un `FrameLayout`, los elementos contenidos en un `LinearLayout` establecen sus propiedades `android:layout_width` y `android:layout_height` para determinar sus dimensiones dentro delaplantilla; pero en el caso de un `LinearLayout`, se tiene la propiedad `android:layout_weight`.

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

<EditTextandroid:id="@+id/xet1"
android:layout_width="match_parent"
android:layout_height="match_parent" />

<Buttonandroid:id="@+id/xbn1"
android:layout_width="wrap_content"
android:layout_height="match_parent" />

</LinearLayout>
```

La propiedad `android:layout_weight` permite dar a los componentes contenidos en la plantillalas dimensiones proporcionales entre ellas.Por ejemplo, si se incluyen en un `LinearLayout` vertical a dos campos `EditText` ya uno de ellos se le establece un `layout_weight=”1”` y al otro un `layout_weight=”2”`, el efecto es que toda la superficie dela plantilla quede ocupada por los dos campos de texto y que además el segundo sea el doble de alto que el primero (la relación entre sus propiedades `weight`).

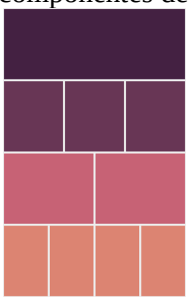
```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
```

```
android:orientation="vertical">

<EditTextandroid:id="@+id/xet1"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_weight="1" />

<EditTextandroid:id="@+id/xet2"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:layout_weight="2" />

</LinearLayout>
```

A pesar de parecer muy sencilla esta plantilla, resulta ser muy versátil.

- **TableLayout**

Un `TableLayout` distribuye los componentes en forma tabular, definiendo las filas y columnas necesarias, y la posición de cada componente dentro de la tabla. La estructura de la tabla se define de forma similarque en HTML, es decir, indicando las filas que conforman la tabla (objetos `TableRow`), y dentro de cada fila las columnas necesarias, con la excepción de que no existe ningún objeto especial para definir una columna (un `TableColumn`) sino que directamente se insertan los componentes necesarios dentro del `TableRow` y cada componente insertado (por ejemplo un componente sencillo u otro `ViewGroup`) corresponderá a una columna de la tabla. De esta formael número final de filas de la tabla se corresponderá con el número de elementos `TableRow` insertados, y el número total de columnas quedará determinado por el número de componentes de la fila que más componentes contenga.



En forma predeterminada, el ancho de cada columna se corresponde con el ancho del mayorcomponente de esa columna, pero existe una serie de propiedades queayudan a modificar este comportamiento:

- `android:stretchColumns`. Indica las columnas que se pueden expandir para absorber el espacio libre dejado por las demás columnas a la derecha de la pantalla.
- `android:shrinkColumns`. Indica las columnas que se pueden contraer paradejar espacio al resto de columnas que se puedan salir por la derecha de la pantalla.
- `android:collapseColumns`. Indica las columnas de la tabla que se desean ocultar completamente.

Estas propiedades de`TableLayout` pueden recibir una lista de índices de columnas separados por comas, por ejemplo `android:stretchColumns="1,2,3"`), o un asterisco para indicar que debe aplicar a todas las columnas, por ejemplo, `android:stretchColumns="*"`.

Otra característica importante es que una celda determinada pueda ocupar el espacio de varias columnas de la tabla (similar al atributo `colspan` de HTML). Esto se indica con la propiedad `android:layout_span` del componente en particular que deberá tomar ese espacio.

Un ejemplo con algunos componentes:

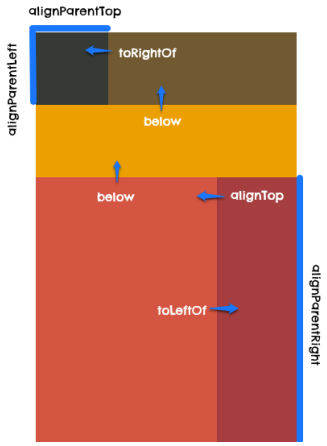
```
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:stretchColumns="1" >

<TableRow>
<TextViewandroid:text="Celda 1.1" />
<TextViewandroid:text="Celda 1.2" />
</TableRow>

<TableRow>
<TextViewandroid:text="Celda 2.1" />
<TextViewandroid:text="Celda 2.2" />
</TableRow>

<TableRow>
<TextViewandroid:text="Celda 3"
android:layout_span="2" />
</TableRow>

</TableLayout>
```

- **RelativeLayout**

Esta plantilla especificala posición de cada elemento de forma relativa a su elemento padre o a cualquier otro elemento incluido en lamisma plantilla. Por ejemplo, si se incluye un nuevo elemento A se puede indicarque debe colocarse debajo del elemento B y alineado a la derecha dela plantilla padre. Por ejemplo:

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

<EditTextandroid:id="@+id/xet1"
android:layout_width="match_parent"
android:layout_height="wrap_content" />

<Buttonandroid:id="@+id/xbn1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/xet1"
android:layout_alignParentRight="true" />

</RelativeLayout>
```

En el ejemplo, el botón `xbn1` se colocará debajo del campo de texto `xet1` (`android:layout_below="@id/xet1"`) y alineado a la derecha dela plantilla padre (`android:layout_alignParentRight="true"`), además de dejar un margen de 10 pixeles a su izquierda (`android:layout_marginLeft="10px"`).



En un `RelativeLayout` se tienen varias propiedades para colocar los componentesen donde se desee. Las propiedades principales son las siguientes:

Tipo	Propiedades

Posición relativa a otro componente	`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`

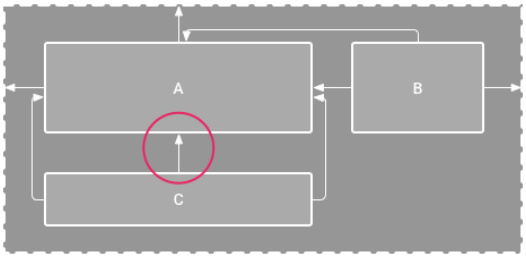
Posición relativa ala plantilla padre	`android:layout_alignParentLeft`
	`android:layout_alignParentRight`
	`android:layout_alignParentTop`
	`android:layout_alignParentBottom`
	`android:layout_centerHorizontal`
	`android:layout_centerVertical`
	`android:layout_centerInParent`

Opciones de margen (también disponibles para el resto de plantillas)	`android:layout_margin`
	`android:layout_marginBottom`
	`android:layout_marginTop`
	`android:layout_marginLeft`
	`android:layout_marginRight`

Opciones de espaciado o padding (también disponibles para el resto de plantillas)	`android:padding`
	`android:paddingBottom`
	`android:paddingTop`
	`android:paddingLeft`
	`android:paddingRight`

OTRAS PLANTILLAS:

- **`ConstraintLayout`**. Es una versión más flexible y eficiente que `RelativeLayout`.

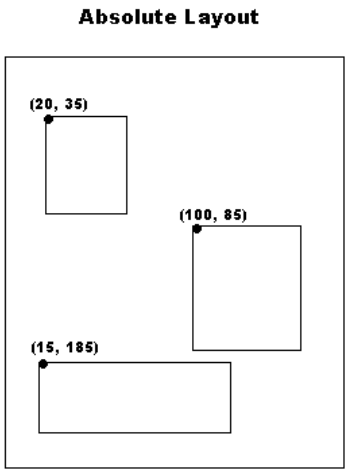


Estaplantilla incluye un modo llamado Autoconnect, el cual una vez activado, cada vez que se agrega un componente alaplantilla se permite crear dos o más restricciones (constraint) para cada componente, para asignarlo al espacio donde se le ha soltado. Su configuración ofrece un modo automático para convertir un `LinearLayout` en un `ConstraintLayout`

Observar que la plantilla ofrece una etiqueta `androidx`. Por ejemplo:

```
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas...
android:layout_height="match_parent"
android:layout_width="match_parent">
<AnalogClock
android:id="@+id/AnalogClock01"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
app:layout_constraintLeft_toLeftOf="parent"
app:layout_constraintTop_toTopOf="parent"/>
<CheckBox
android:id="@+id/CheckBox01"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Un checkBox"
app:layout_constraintTop_toBottomOf="@+id/AnalogClock01"
app:layout_constraintTop_toTopOf="parent"/>
<Button
android:id="@+id/Button01"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Un botón"
app:layout_constraintTop_toBottomOf="@+id/CheckBox01"
app:layout_constraintLeft_toLeftOf="@+id/CheckBox01"/>
<TextView
android:id="@+id/TextView01"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Un texto cualquiera"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

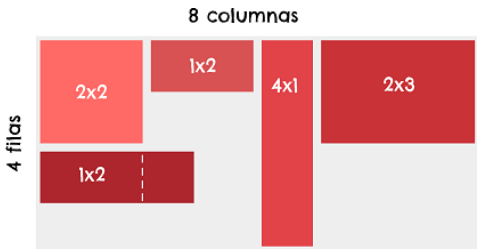
- **`AbsoluteLayout`**. Esta plantilla utilizar las coordenadas (x,y) en donde se desea visualizar cada elemento. No es muy recomendable utilizar este tipo de plantilla pues se considera obsoleta.



La aplicación debe visualizarse correctamente en dispositivos con cualquier tamaño de pantalla ya que no es una buena práctica trabajar con coordenadas absolutas.

```
<AbsoluteLayoutxmlns:android="http://schemas.
android:layout_height="match_parent"
android:layout_width="match_parent">
<AnalogClock
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_x="50px"
android:layout_y="50px"/>
<CheckBox
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Un checkBox"
android:layout_x="150px"
android:layout_y="50px"/>
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Un botón"
android:layout_x="50px"
android:layout_y="250px"/>
<TextView
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Un texto cualquiera"
android:layout_x="150px"
android:layout_y="200px"/>
</AbsoluteLayout>
```

- **`GridLayout`**. Un `GridLayout` es un `ViewGroup` que alinea sus elementos hijos en una malla (grid). Se utiliza para evitar el anidamiento de plantillas `LinearLayout` para crear diseños complejos.



Algunos atributos importantes del `GridLayout` son:

- `columnCount`: Número de columnas que tendrá la malla.
- `rowCount`: Número de filas de la cuadrícula.
- `useDefaultMargins`: Se asigna el valor `true` para los márgenes predeterminados entre los componentes.

Se especifica la cantidad de filas y columnas que ocupará una celda con los atributos `android:layout_rowSpan` y `android:layout_columnSpan`. Por ejemplo:

```
<?xmlversion="1.0" encoding="utf-8"?>
<GridLayoutxmlns:android="http://schemas.android.com/apk/res/android"
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"
android:layout_column="0"
android:layout_row="0"
android:text="7" />

<TextView
android:id="@+id/numero_8"
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"
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"
android:layout_height="wrap_content"
android:layout_column="0"
android:layout_row="1"
android:text="4" />

<TextView
```

```
android:id="@+id/numero_5"
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"
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"
android:layout_column="3"
android:layout_gravity="fill"
android:layout_row="1"
android:gravity="center"
android:text="×" />

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

<TextView
android:id="@+id/numero_1"
android:layout_column="0"
android:layout_row="2"
android:text="1" />

<TextView
android:id="@+id/numero_2"
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"
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="match_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="match_horizontal"
android:layout_row="3"
android:text="+" />
</GridLayout>
```

NOTA: Generar un reporte con las imágenes de cada una de las plantillas obtenidas y guardarlo con la sintaxis NombreTarea1Grupo.pdf y enviarlo al sitio indicado por el profesor.