Asignatura: Desarrollo de Aplicaciones Móviles Nativas.
Tema: Las plantillas `Layout`.

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.

Por ejemplo, las plantillas más usuales son:
- LinearLayout
- FrameLayout
- RelativeLayout
- TableLayout
- GridLayout

El código básico de Java para todos los ejercicios de las plantillas es el siguiente:
```
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.*;
public class MainActivity extends Activity{
      public void onCreate(Bundle b) {
            super.onCreate(b);
            setContentView(R.layout.activity_main);
      }
}
```

La jerarquía de clases de la plantilla `Layout`:

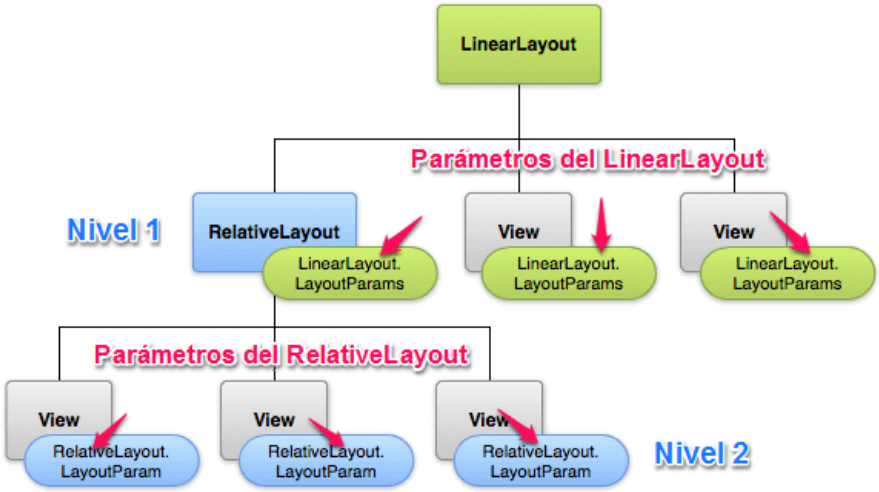

Figura 1. La jerarquía de clases de la plantilla `Layout`.

A las plantillas se les pueden asignar tamaños absolutos en notación `dps`, pero usualmente se usan los siguientes:
```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

Otros parámetros son los siguientes:

`top`	Indica la parte superior de la plantilla.
`left`	Indica la parte izquierda de la plantilla.

`right`	Se refiere a la parte derecha de la plantilla.
`bottom`	Representa el límite inferior de la plantilla.
`center_horizontal`	Centro horizontal de la plantilla.
`center_vertical`	Alineación al centro vertical de la plantilla.
`center`	`center_horizontal` y `center_vertical` de la plantilla.

EJEMPLOS.
Ejecutar cada uno de los siguientes ejemplos utilizando el mismo código Java para cada uno de los códigos de las plantillas correspondientes.

- **`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 utiliza para mostrar un único componente en su interior, como un contenedor (placeholder) sencillo para un sólo elemento sustituible, por ejemplo una imagen.

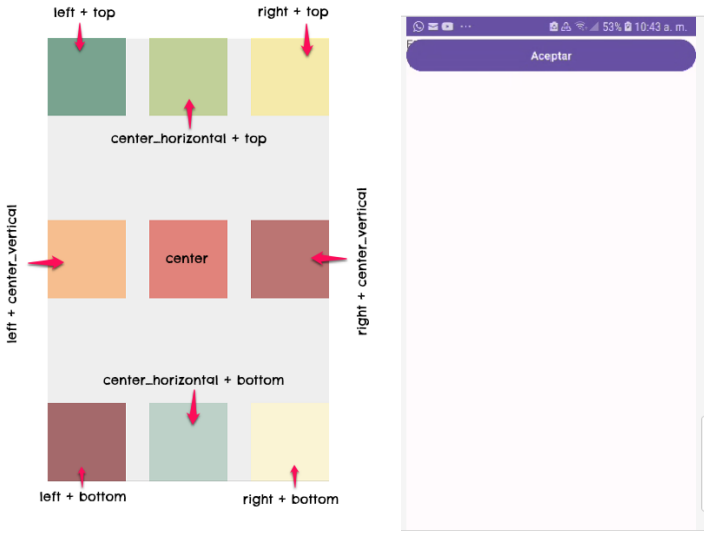

Figura 1. Plantilla `FrameLayout`.

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). Por ejemplo:

```
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Etiqueta" />
    <EditText
        android:hint="... escribir texto ..."
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <Button
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content"
        android:text="Aceptar" />
</FrameLayout>
```

Otro ejemplo es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:tools="http://schemas.android.com/tools"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      tools:context=".ActividadPrincipal">
      <Button
            android:layout_width="match_parent"
            android:layout_height="60dp"
            android:text="Saltar"
            android:id="@+id/boton_saltar"
            android:layout_gravity="center_horizontal|bottom"/>
      <ImageView
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/imagen_background"
            android:layout_gravity="top|center"
            android:src="@drawable/background_frame_layout"
            android:scaleType="centerCrop" />
      <ImageView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/imagen_estadistica"
            android:layout_gravity="center"
            android:src="@drawable/ejemplo_estadistica"
            android:padding="16dp" />
</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`.

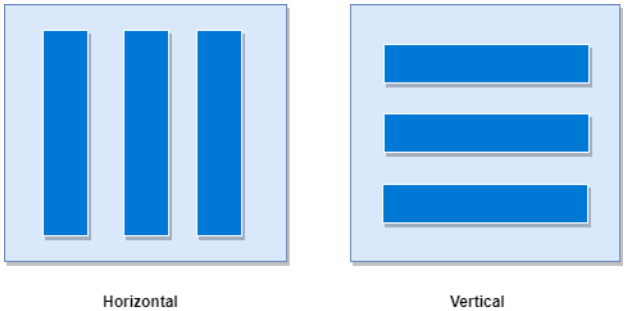

Figura 2. Plantilla `LinearLayout`.

Figura 2.1. Vertical.

Figura 2.2. Utilizando `weight`.

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 de la plantilla; 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">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Etiqueta" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="... escribir texto ..."/>
    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Aceptar" />
</LinearLayout>
```

La propiedad android:layout_weight proporciona a los componentes contenidos en la plantilla las dimensiones adecuadas entre ellas. Por ejemplo, si se incluyen en un LinearLayout vertical dos campos EditText a uno de ellos se le asigna un layout_weight="1" y al otro un layout_weight="2"; así, a toda la superficie de la plantilla queda 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). Esta plantilla es sencilla y muy versátil Ver la figura 2.2.

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <EditText
        android:id="@+id/xet1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:hint="campo de texto 1"
        android:layout_weight="1" />
    <EditText
        android:id="@+id/xet2"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:hint="campo de texto 2"
        android:layout_weight="2" />
</LinearLayout>
```

El siguiente es otro ejemplo de LinearLayout:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
      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="1"
		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="0dp"
		android:layout_gravity="center_horizontal"
		android:layout_weight="1"
		android:text="Iniciar Sesion" />
	<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:gravity="center_vertical"
		android:text="¿Olvidaste tu constraseña?"
		android:textAppearance="?android:attr/textAppearanceMedium"
		android:textColor="#0E8AEE" />
</LinearLayout>
```

- **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, como 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. Así, el 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.

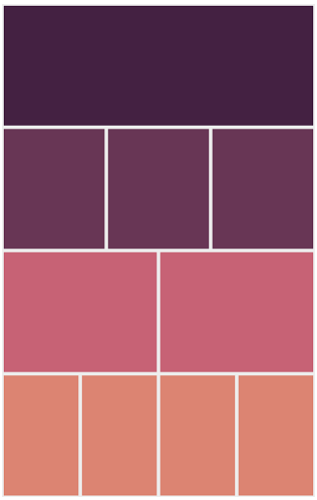

Figura 3. Plantilla `TableLayout`.

Figura 3.1 Plantilla `GridLayout`.

En forma predeterminada, el ancho de cada columna se corresponde con el ancho del mayor componente de esa columna, pero existe una serie de propiedades que ayudan 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 para dejar 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 de **posicionamiento tabular** con `TableLayout` que posee un y algunos componentes:

```
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <TableRow>
        <TextView android:text="Celda1.1" />
        <TextView android:text="Celda1.2" />
        <TextView android:text="Celda1.3" />
    </TableRow>
    <TableRow>
        <TextView android:text="Celda2.1" />
        <TextView android:text="Celda2.2" />
        <TextView android:text="Celda2.3" />
    </TableRow>
    <TableRow>
        <TextView android:text="Celda3.1"
        android:layout_span="2" />
```

```
        <TextView android:text="Celda3.2" />
    </TableRow>
</TableLayout>
```

La plantilla `GridLayout` posee una **tabulación irregular** pero configurable como `TableLayout`. Un ejemplo con `GridLayout` y algunos componentes:

```
<GridLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:rowCount="2"
    android:columnCount="3"
    android:orientation="horizontal" >
    <TextView android:text="Celda1.1" />
    <TextView android:text="Celda1.2" />
    <TextView android:text="Celda1.3" />
    <TextView android:text="Celda2.1" />
    <TextView android:text="Celda2.2" />
    <TextView android:text="Celda2.3" />
    <TextView android:text="Celda3.1" android:layout_columnSpan="2" />
    <TextView android:text="Celda3.2" />
</GridLayout>
```

- **`RelativeLayout`**

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

```
<RelativeLayout
     xmlns:android="http://schemas.android.com/apk/res/android"
     android:layout_width="match_parent"
     android:layout_height="match_parent" >
     <EditText
          android:id="@+id/xet1"
          android:layout_width="match_parent"
          android:layout_height="wrap_content" />
     <Button
          android: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 de la plantilla padre (`android:layout_alignParentRight="true"`), además de dejar un margen de 10 pixeles a su izquierda (`android:layout_marginLeft="10px"`).

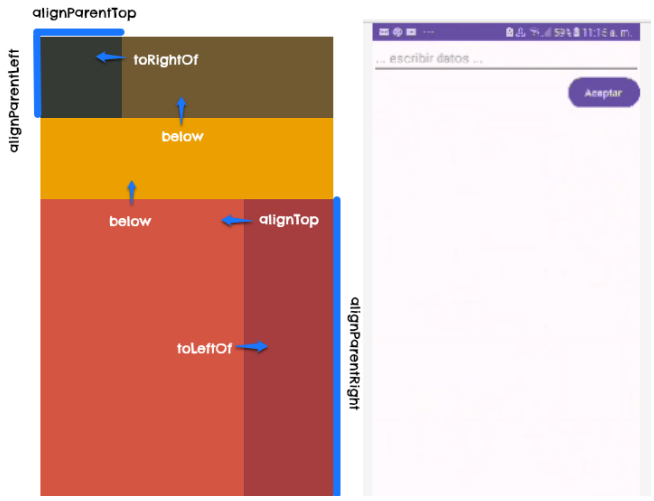

Figura 4. Plantilla `RelativeLayout`.

En una plantilla `RelativeLayout` se tienen varias propiedades para colocar los componentes en 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 a la 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 las 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 las plantillas)	`android:padding`
	`android:paddingBottom`
	`android:paddingTop`
	`android:paddingLeft`
	`android:paddingRight`

- **`ScrollView`**

El `ScrollView` permite incluir una jerarquía de views con el fin de desplazar su contenido a lo largo de la pantalla, cuando sus dimensiones exceden el tamaño de esta.

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <ImageView
            android:id="@+id/imageView"
            android:layout_width="wrap_content"
            android:layout_height="200dp"
            android:src="@mipmap/ic_launcher" />
        <Button
            android:id="@+id/button"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Aceptar" />
        <TextView
            android:id="@+id/textView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Números:\n0\n1\n2\n3\n4\n5\n6\n7\n8\n9
            \n10\n11\n12\n13\n14\n15\n16\n17\n18\n19\n20
            \n21\n22\n23\n24\n25\n26\n27\n28\n29\n30" />
    </LinearLayout>
</ScrollView>
```

Figura 5. Plantilla `ScrollView`

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

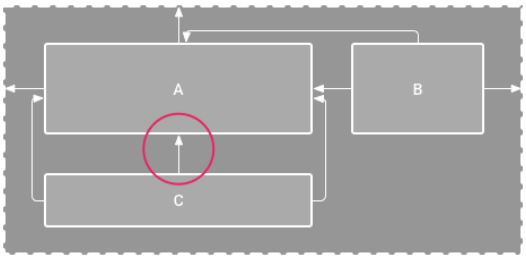

Figura 6. Plantilla `ConstraintLayout`.

Esta plantilla incluye un modo llamado Autoconnect, el cual una vez activado, cada vez que se agrega un componente a la plantilla 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 posee una etiqueta `androidx`. También, ingresar una etiqueta `<color>`, con el color de su preferencia, como se indica con letra negrita. Por ejemplo:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.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">
    <View
        android:id="@+id/logo"
        android:layout_width="300dp"
        android:layout_height="80dp"
        android:background="@color/colorAccent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.15" />
    <EditText
        android:id="@+id/etUsername"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:hint="Añade tu email"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/logo"
        app:layout_constraintVertical_bias="0.20" />
    <EditText
        android:id="@+id/etPassword"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
```

```
        android:hint="Añade la contraseña"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etUsername"
        app:layout_constraintVertical_bias="0.05" />
    <Button
        android:id="@+id/btnLogin"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginEnd="16dp"
        android:text="Login"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etPassword"
        app:layout_constraintVertical_bias="0.95" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Figura 7. Plantilla `ConstraintLayout` con componentes y sus nodos `Autoconnect`.

- **`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.

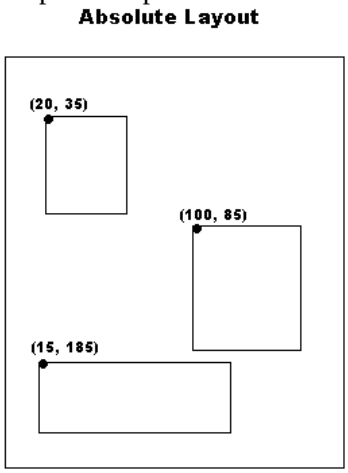

Figura 8. Distribución de componentes con `AbsoluteLayout`.

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

```
<AbsoluteLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="match_parent"
    android:layout_width="match_parent">
    <AnalogClock
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="21dp"
        android:layout_y="247dp" />
    <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:layout_x="29dp"
        android:layout_y="151dp"
        android:text="Un botón" />
    <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>
```

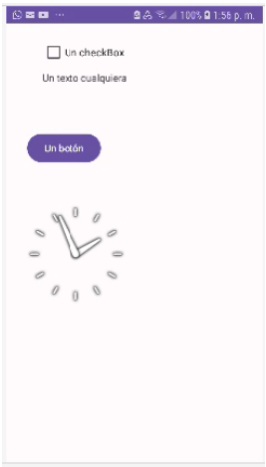

Figura 9. Plantilla `AbsoluteLayout`.

Ejercicio.
Diseñar una aplicación móvil que muestre la interface de una calculadora con un campo de texto, diez dígitos (0-9) y las operaciones de suma (+) e igualdad (=).

NOTA:

- En cada una de las plantillas modificar o agregar una etiqueta `TextView` que muestre su nombre.
- Generar un reporte con las imágenes de cada una de las plantillas obtenidas y guardarlo con la sintaxis NombrePlantillasGrupo.pdf y enviarlo al sitio indicado por el profesor.