

Enfoques de desarrollo de Aplicaciones Móviles



Aplicaciones Nativas con Android

Layouts

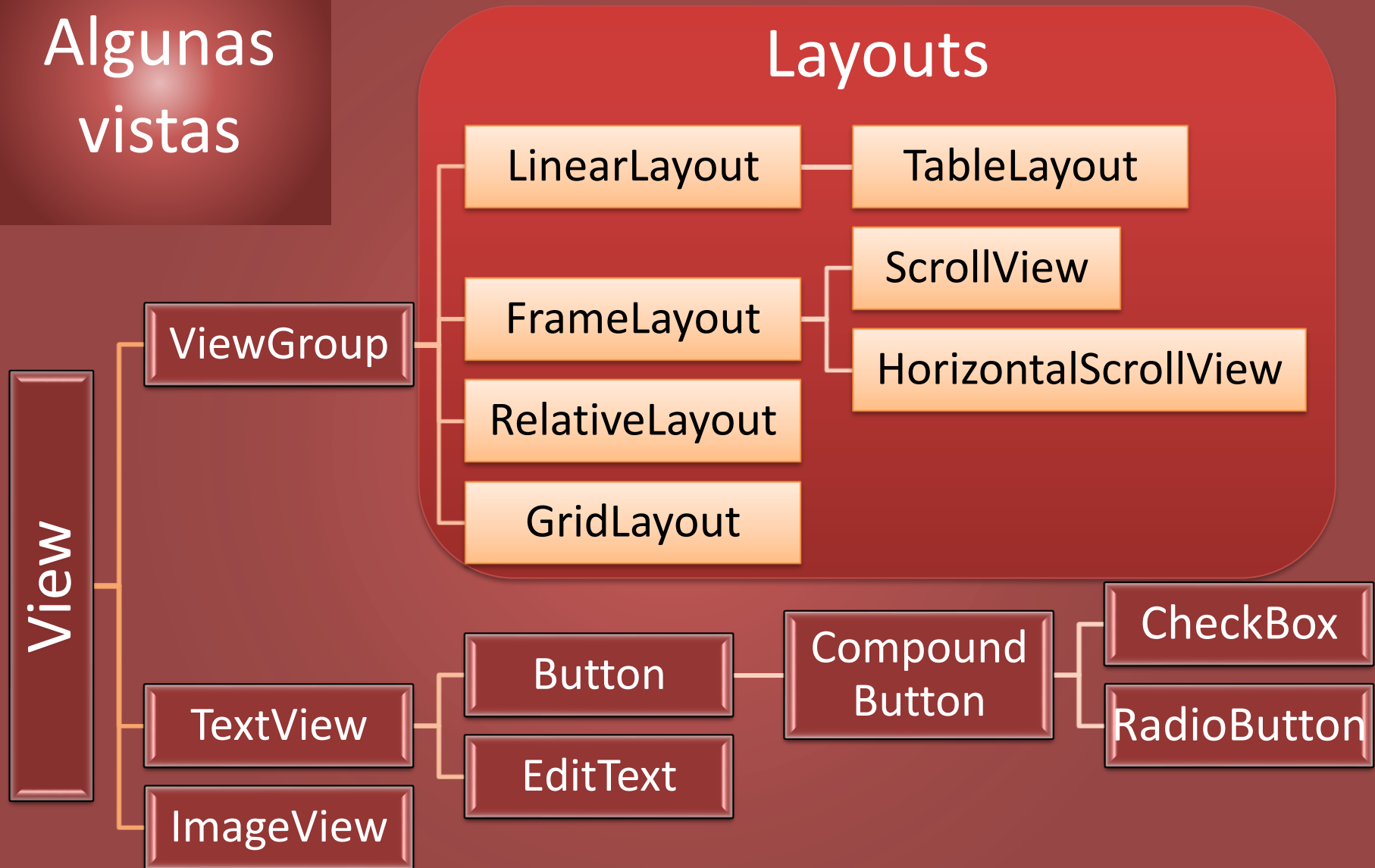
Layouts

- El **layout** de una **activity** representa el diseño de la interfaz de usuario determinando la disposición de distintos componentes visuales (vistas o **views**) en la misma.
- Los **layouts** también son vistas pero pertenecen a una categoría específica de vistas (**ViewGroup**) capaz de contener a otras vistas

Viewgroups

- Los elementos visuales simples, como el **EditText**, **TextView**, **Button**, etc. son clases particulares de vistas (**View**) que deben ser dispuestos dentro de un contenedor
- El contenedor es un **ViewGroup** que define el modo que se muestran los elementos hijos que aloja.
- Conoceremos algunos de los **viewGroups** más populares

Algunas vistas

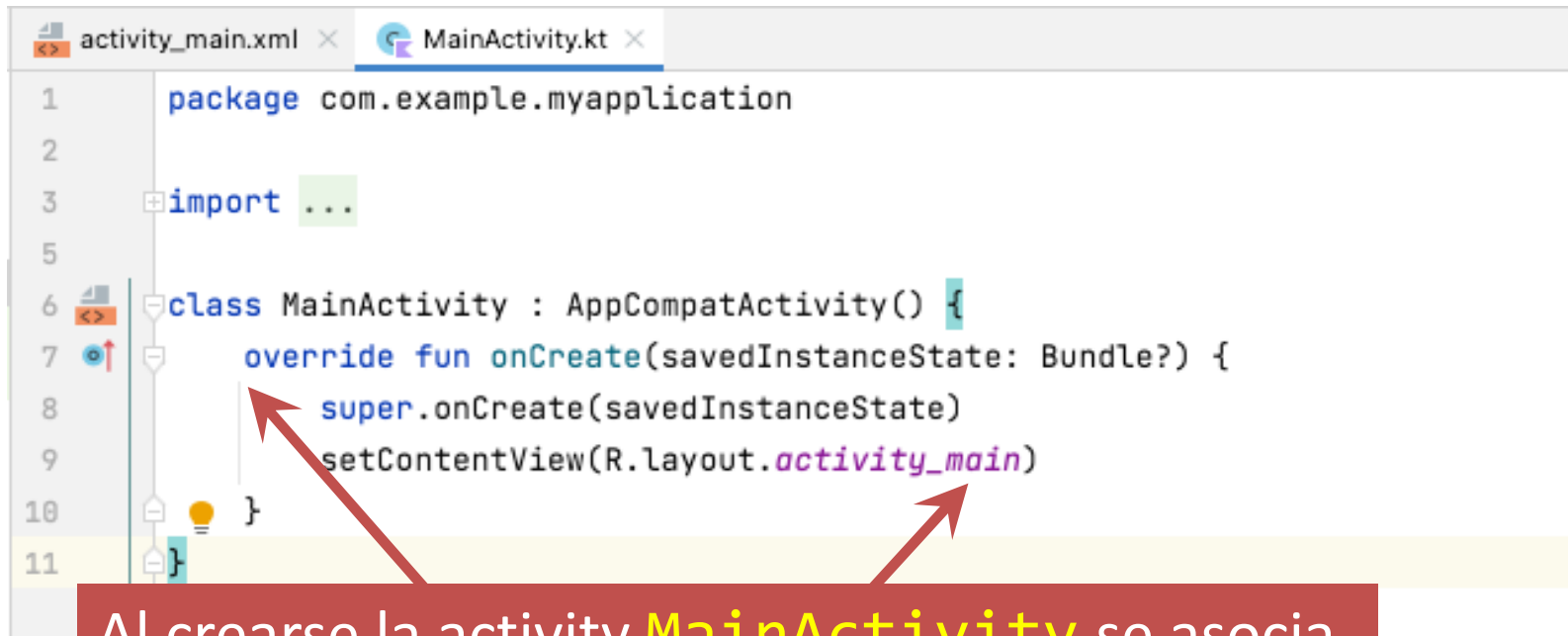


Layouts

- La interfaz de usuario puede ser definida mediante:
 - Archivos XML
 - En tiempo de ejecución, programáticamente.

Layouts desde archivos XML

- Las **Activities** que definen su interfaz por medio de **archivos XML** se asocian a estos archivos mediante **setContentView()** en el callback **onCreate()**



The screenshot shows an IDE with two tabs: 'activity_main.xml' and 'MainActivity.kt'. The 'MainActivity.kt' tab is active, displaying the following Kotlin code:

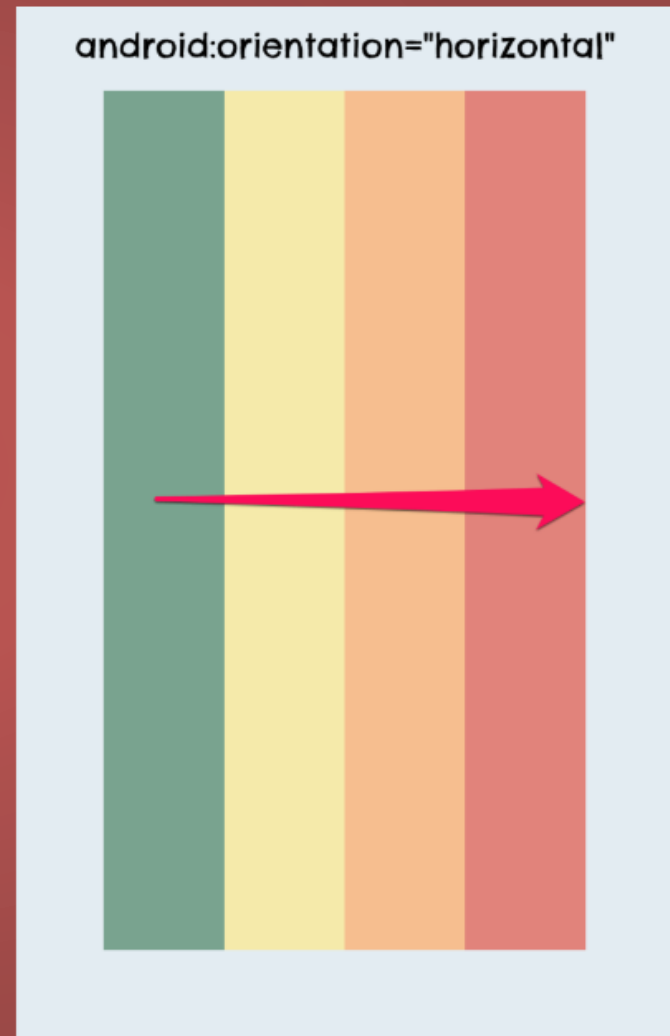
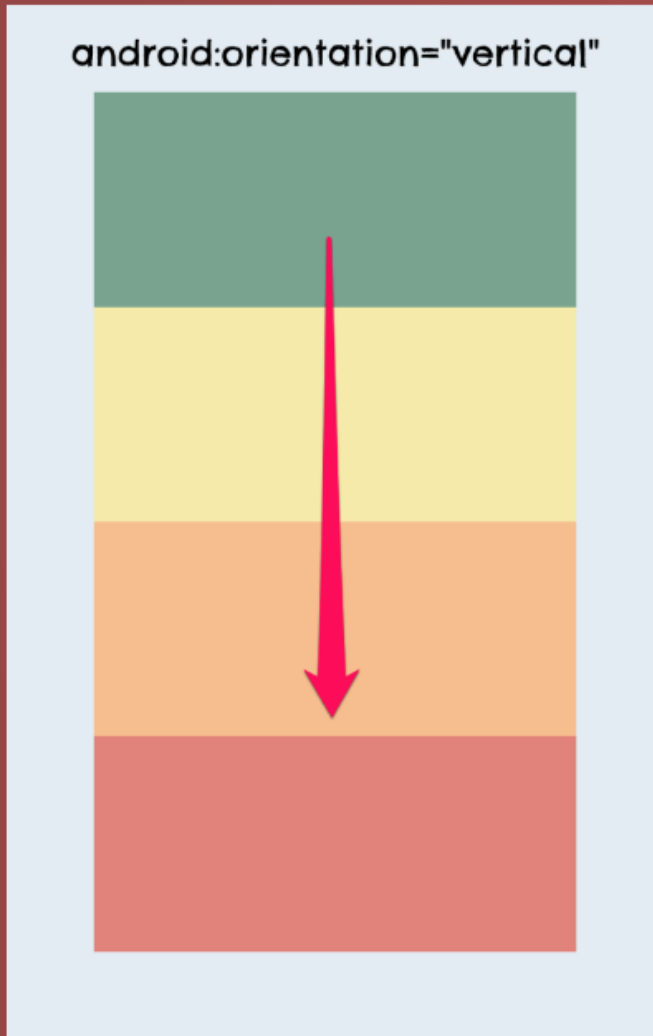
```
1 package com.example.myapplication
2
3 import ...
4
5
6 class MainActivity : AppCompatActivity() {
7     override fun onCreate(savedInstanceState: Bundle?) {
8         super.onCreate(savedInstanceState)
9         setContentView(R.layout.activity_main)
10    }
11 }
```

Two red arrows originate from a red text box at the bottom. One arrow points to the `setContentView(R.layout.activity_main)` line (line 9), and the other points to the `onCreate` method signature (line 7).

Al crearse la activity **MainActivity** se asocia el layout **activity_main** definido en el archivo **activity_main.xml**

Vamos a presentar alguno de los **layouts** más populares

LinearLayout

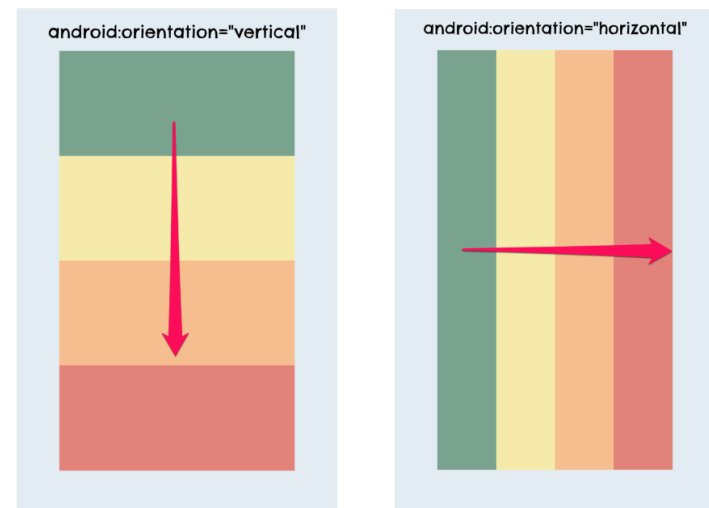


Ya lo hemos utilizado en clases anteriores

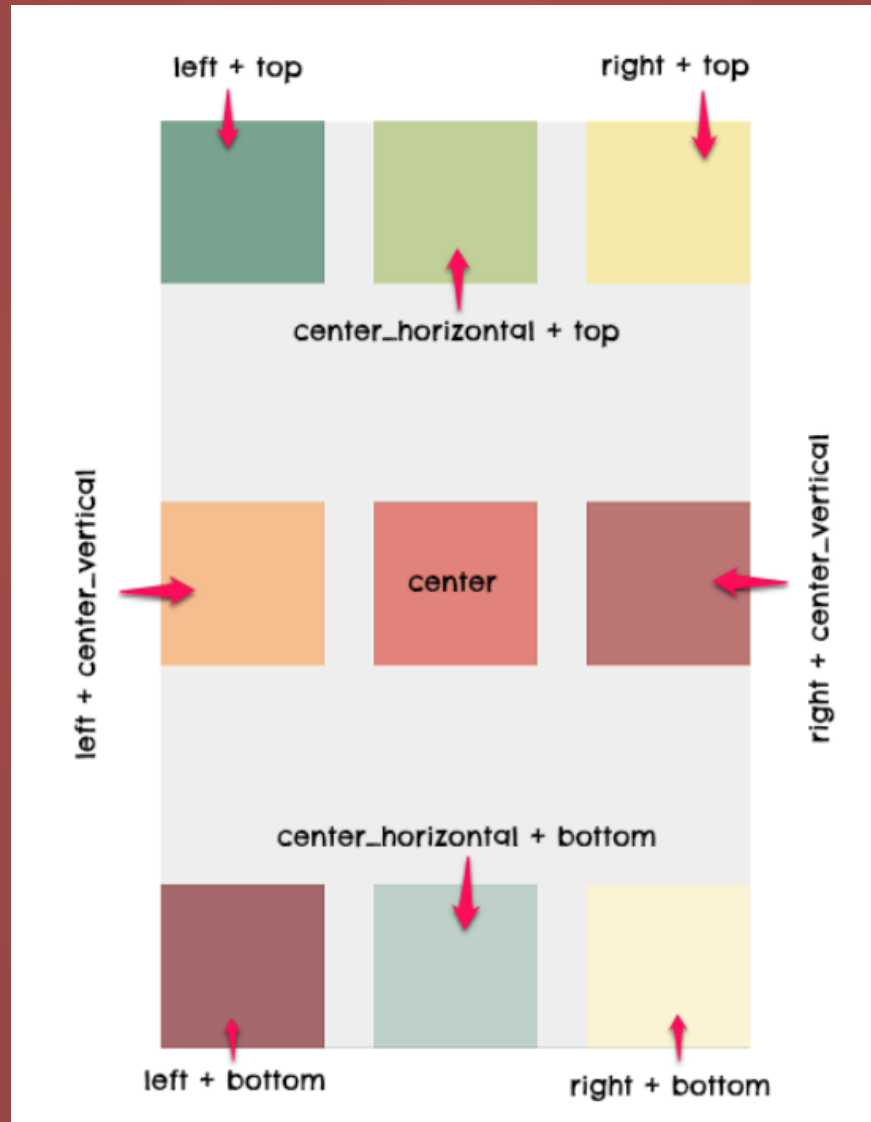
LinearLayout

- Es un **ViewGroup** que alinea a los elementos hijos en una única dirección.
- La dirección puede ser vertical u horizontal.

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



FrameLayout



FrameLayout

Modificar la vista de la **activity** principal de la siguiente manera

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

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="100dp"
        android:text="Primer texto"/>

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="50dp"
        android:text="Segundo texto"
    />
</FrameLayout>
```

FrameLayout

Modificar la vista de la **activity** principal de la siguiente manera



Un **FrameLayout** es un **ViewGroup** simple y eficiente.

Pensado para ser usado con un **view** hijo o con **Views** que admitan solapamiento.

Probar y responder



¿Cuál es el efecto en las vistas de los siguiente atributos ?

`android:layout_gravity="right"`

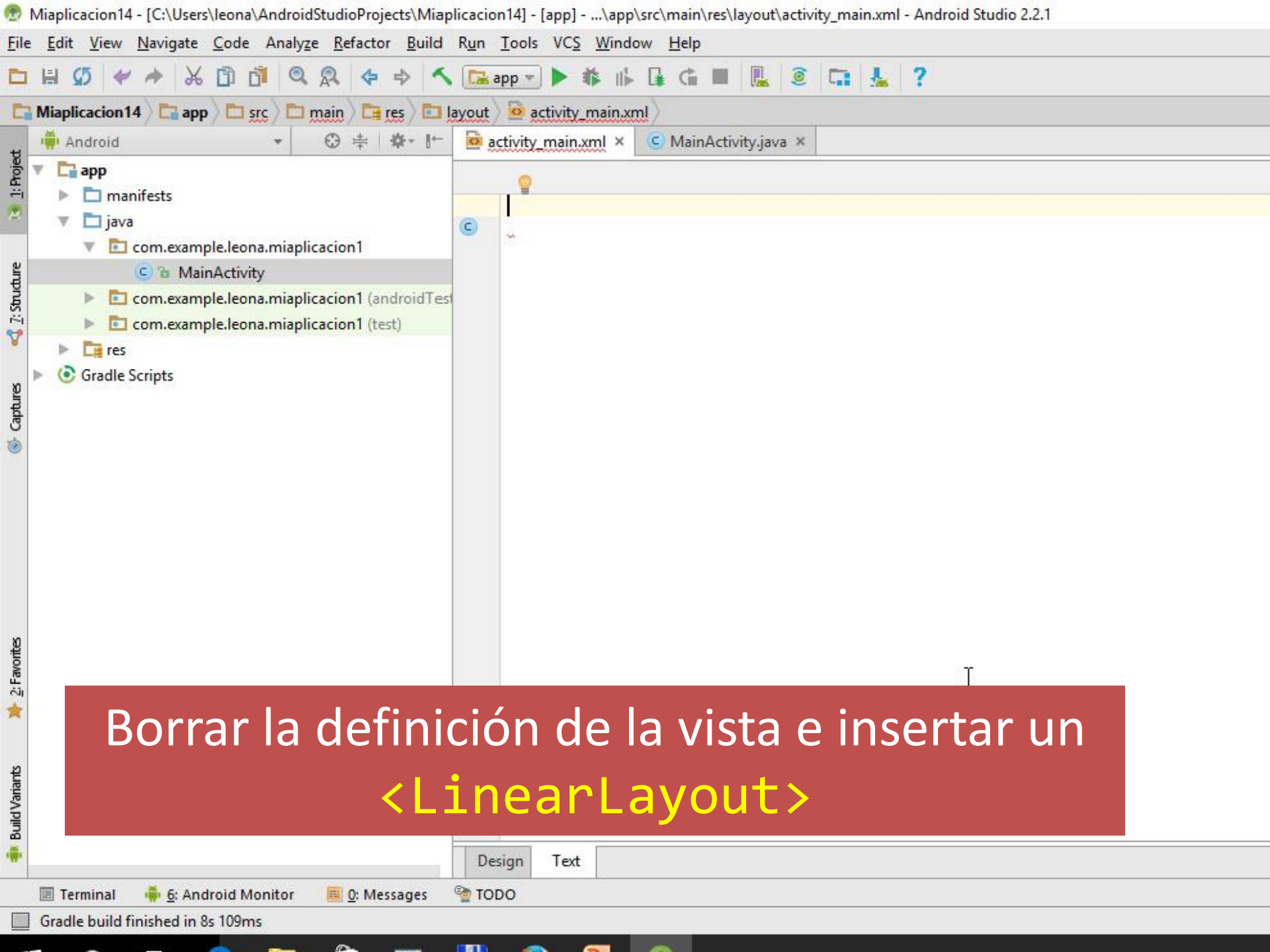
`android:layout_gravity="bottom"`

`android:layout_gravity="center"`

`android:layout_gravity="center|right"`

ScrollView y HorizontalScrollView

- **ScrollView** es un **FrameLayout** especializado que puede hacer **scroll vertical** sobre el elemento que contiene (sólo puede alojar un único hijo)
- **HorizontalScrollView** es un **FrameLayout** especializado que puede hacer **scroll horizontal** sobre el elemento que contiene (sólo puede alojar un único hijo)
- El ejemplo a continuación mostrará la necesidad de utilizar un **ScrollView**



Borrar la definición de la vista e insertar un
<LinearLayout>

The screenshot shows the XML editor for `activity_main.xml` in Android Studio. The code defines a `LinearLayout` with a vertical orientation and a `Button` with the text "Botón 1".

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

    <Button
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:text="Botón 1"
        android:layout_margin="10dp"/>

</LinearLayout>
```

Annotations:

- A red box highlights the `android:orientation="vertical"` attribute in the `LinearLayout` tag.
- A red box highlights the `android:layout_height="150dp"`, `android:text="Botón 1"`, and `android:layout_margin="10dp"` attributes in the `Button` tag.
- A red arrow points from the text "Establecer orientación vertical" to the `android:orientation="vertical"` attribute.
- A red arrow points from the text "Agregar un botón con texto 'Botón 1' y establecer sus dimensiones y margen de esta manera" to the `Button` tag.


```
activity_main.xml x MainActivity.java x
LinearLayout
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical">
    <Button
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:text="Botón 1"
        android:layout_margin="10dp"/>
    <Button
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:text="Botón 2"
        android:layout_margin="10dp"/>
    <Button
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:text="Botón 3"
        android:layout_margin="10dp"/>
    <Button
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:text="Botón 4"
        android:layout_margin="10dp"/>
</LinearLayout>
```

Copiar y pegar
tres veces para
definir así los
botones 2, 3 y 4

Ejecutar en un emulador

Ejecutar en un emulador

¿ Se visualizan
adecuadamente todos los
botones ?

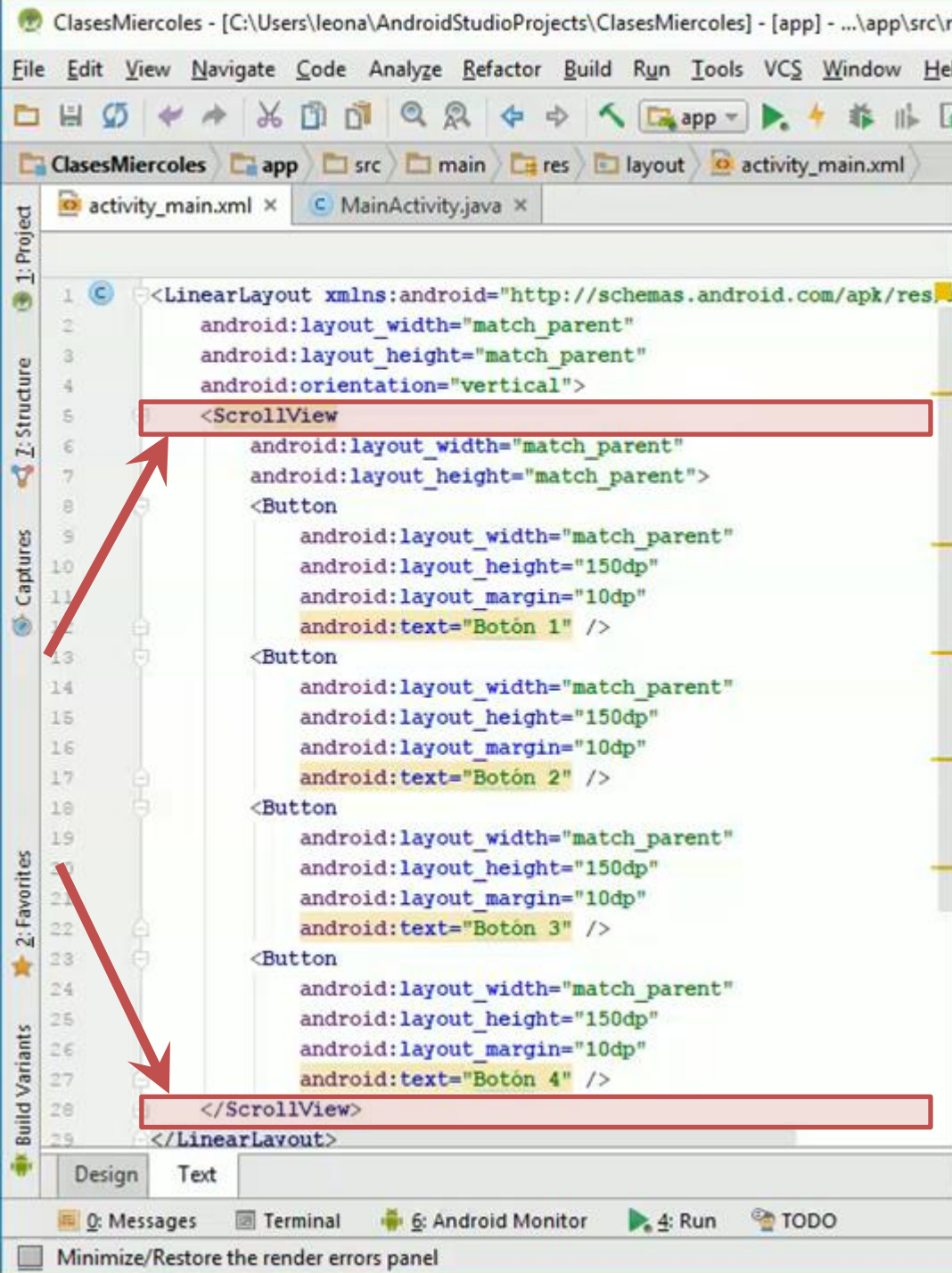
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical">
    <Button
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:text="Botón 1"
        android:layout_margin="10dp"/>
    <Button
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:text="Botón 2"
        android:layout_margin="10dp"/>
    <Button
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:text="Botón 3"
        android:layout_margin="10dp"/>
    <Button
        android:layout_width="match_parent"
        android:layout_height="150dp"
        android:text="Botón 4"
        android:layout_margin="10dp"/>
</LinearLayout>
```

Ejecutar en un emulador

<ScrollView>

A1 rescate

```
<?xml version="1.0" encoding="utf-8" android:windowBackground="@color/white">
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical">
    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent">
        <Button
            android:layout_width="match_parent"
            android:layout_height="150dp"
            android:text="Botón 3"
            android:layout_margin="10dp"/>
        <Button
            android:layout_width="match_parent"
            android:layout_height="150dp"
            android:text="Botón 4"
            android:layout_margin="10dp"/>
    </ScrollView>
</LinearLayout>
```



Pregunta:
¿Por qué esta
solución no podría
ser válida?

Respuesta:
Porque **ScrollView** es
un **FrameLayout**
especializado que sólo
puede alojar un hijo
directo

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

        <Button
            android:layout_width="match_parent"
            android:layout_height="150dp"
            android:layout_margin="10dp"
            android:text="Botón 1" />

        <Button
            android:layout_width="match_parent"
            android:layout_height="150dp"
            android:layout_margin="10dp"
            android:text="Botón 2" />

        <Button
            android:layout_width="match_parent"
            android:layout_height="150dp"
            android:layout_margin="10dp"
            android:text="Botón 3" />

        <Button
            android:layout_width="match_parent"
            android:layout_height="150dp"
            android:layout_margin="10dp"
            android:text="Botón 4" />

    </LinearLayout>
</ScrollView>
```

Solución válida:
El **ScrollView** aloja
un único hijo directo
(un **LinearLayout**
que contiene a los
botones)

Modificar la aplicación para
disponer los botones
horizontalmente.

El scroll ahora debe ser
horizontal.

El ancho de los botones debe
ser de 150dp



```
<HorizontalScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">
        <Button
            android:layout_width="150dp"
            android:layout_height="match_parent"
            android:layout_margin="10dp"
            android:text="Botón 1" />
        <Button
            android:layout_width="150dp"
            android:layout_height="match_parent"
            android:layout_margin="10dp"
            android:text="Botón 2" />
        <Button
            android:layout_width="150dp"
            android:layout_height="match_parent"
            android:layout_margin="10dp"
            android:text="Botón 3" />
    </LinearLayout>
</HorizontalScrollView>
```

Solución:
Se debe utilizar un
<HorizontalScrollView>

Vamos a acceder
programáticamente a los
elementos visuales (vistas o
views) de la activity

Para identificar los **views** del
layout vamos a usar el atributo **id**

Atributo **id**

<Button

android:id="@+id/boton1"

android:layout_width="150dp"

android:layout_height="match_parent"

android:layout_margin="10dp"

android:text="Botón 1" />

.xml

El **id** es un atributo especial que permite identificar al elemento desde el código kotlin

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

```
<LinearLayout
```

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

```
<Button
```

```
    android:id="@+id/boton1"  
    android:layout_width="150dp"  
    android:layout_height="match_parent"  
    android:layout_margin="10dp"  
    android:text="Botón 1" />
```

```
<Button
```

```
    android:id="@+id/boton2"  
    android:layout_width="150dp"  
    android:layout_height="match_parent"  
    android:layout_margin="10dp"  
    android:text="Botón 2" />
```

```
<Button
```

```
    android:id="@+id/boton3"
```

```
    android:layout_width="150dp"
```

Agregar el atributo
android:id a
cada uno de los
botones

¿ Qué significa "@+id/boton1" ?

- Al tipear **@+id/** el entorno convierte una etiqueta en un recurso con un nombre determinado.
- Así al tipear **"@+id/boton1"** se crea un recurso llamado **boton1** que luego puede referenciarse desde el código kotlin por medio de la clase estática **R.id**
- La clase estática **R** y sus clases miembros anidadas se generan automáticamente
- En versiones más recientes, el plugin de Gradle genera el **archivo de bytecode R.class directamente en lugar del archivo R.java.**

```
1 package com.example.myapplication
2
3 import ...
4
5
6
7 class MainActivity : AppCompatActivity() {
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10        setContentView(R.layout.activity_main)
11        val b: Button;
12        b = findViewById<Button>(R.id.boton1)
13        b.setText("Este es el botón 1");
14    }
```

Agregar estas tres instrucciones
al método **onCreate()**

Ejecutar
sobre el
emulador y
comprobar el
resultado



```
package com.cacic2017.escuela.clasesmiercoles;

import androidx.appcompat.app.AppCompatActivity;

public class MainActivity extends AppCompatActivity {

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

        Button b;
        b = (Button) this.findViewById(R.id.button1);
        b.setText("Este es el botón 1");

        Button b2;
        b2 = (Button) this.findViewById(R.id.button2);
        b2.setText("Este es el botón 2");
    }
}
```

Messages Terminal Android Monitor Run TODO

Gradle build finished in 3s 823ms (13 minutes ago)

Explicación del código Java

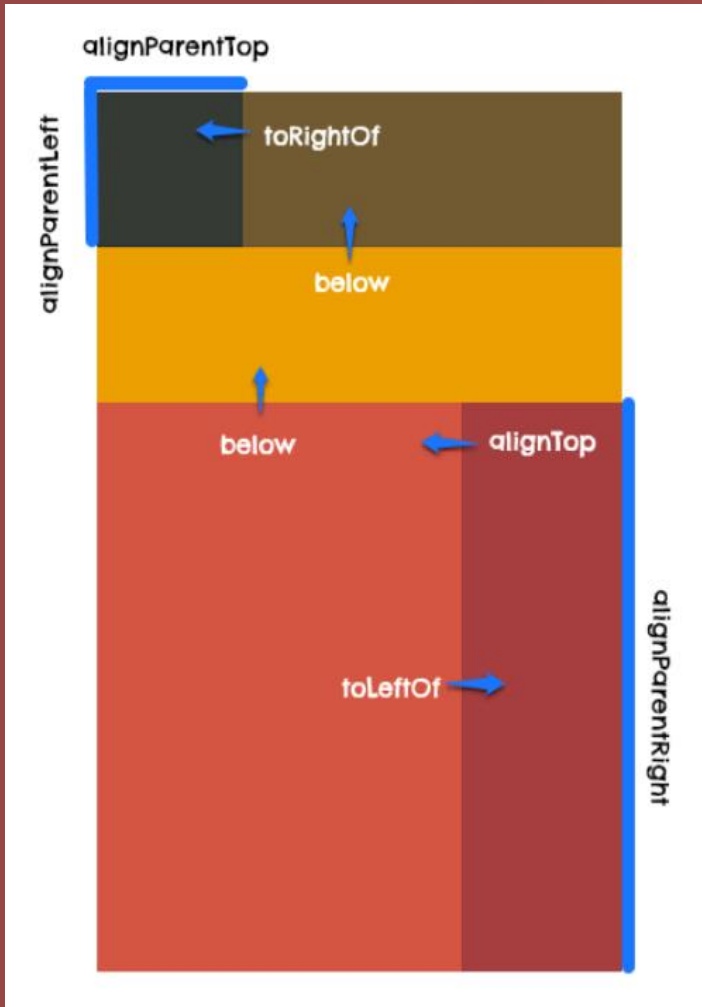
Se define la constante **b** de tipo **Button**

Se asigna a la constante **b** el objeto **View** de la **activity** cuyo **id** es **boton1** (es necesario *castear* a **Button** porque **findViewById** devuelve un objeto **View**)

```
6
7 class MainActivity
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10        setContentView(R.layout.activity_main)
11        val b: Button;
12        b = findViewById<Button>(R.id.boton1)
13        b.setText("Este es el botón 1");
14    }
15 }
```

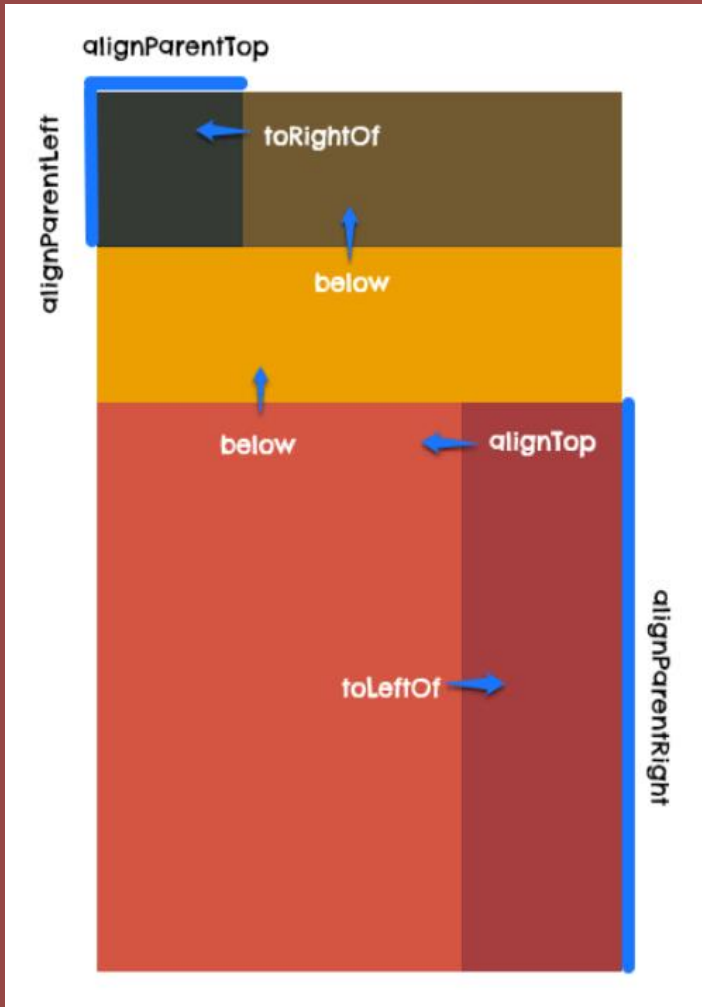
Se establece el texto del **botón 1**

RelativeLayout



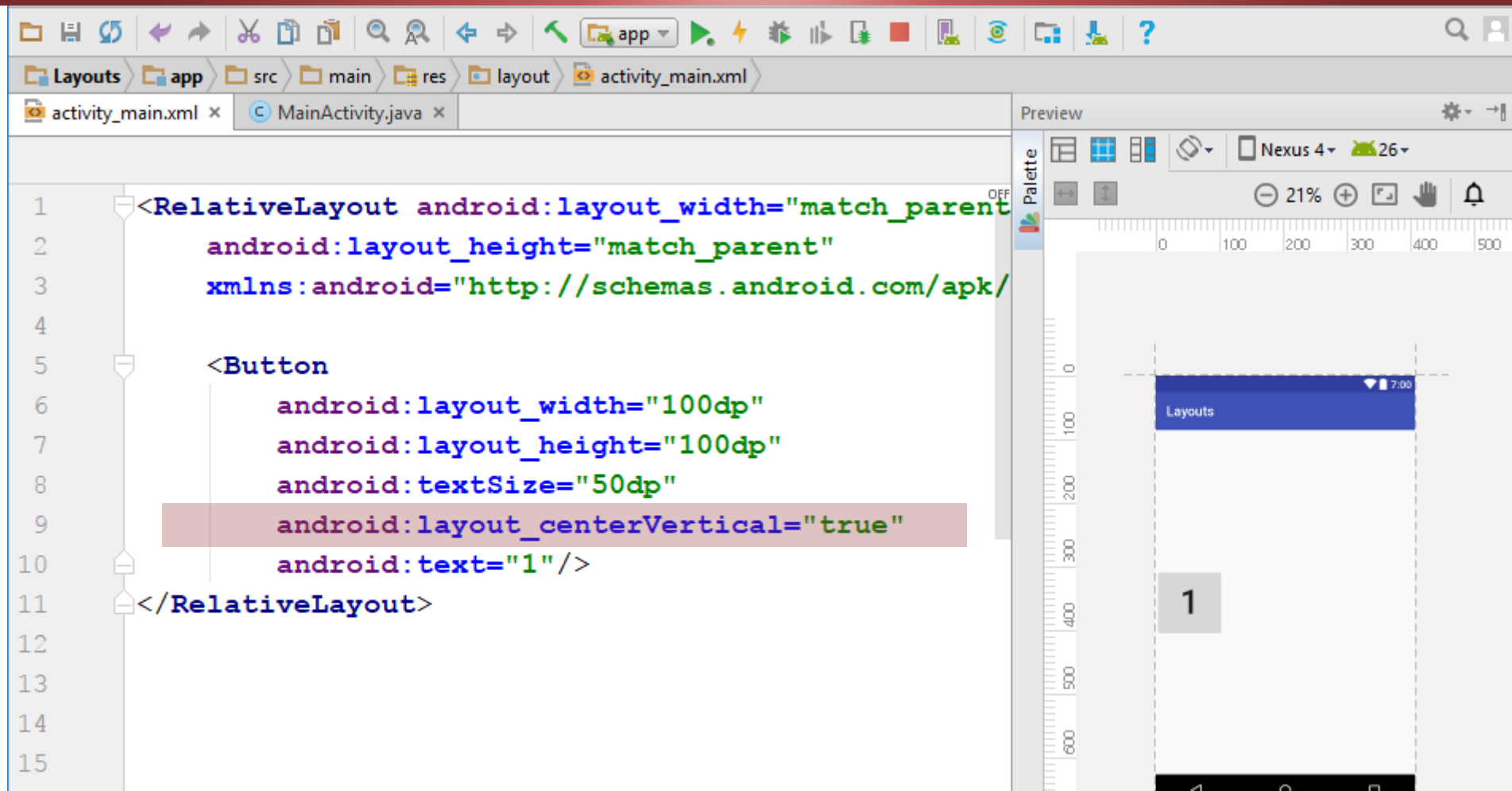
Es un **Layout** donde las posiciones de los elementos contenidos (hijos) pueden ser descritas en relación con otros elementos hijos o bien en relación con el padre, es decir, el propio **Layout**

RelativeLayout



Elimina la necesidad de tener **ViewGroups** anidados, mejorando la performance

RelativeLayout



layout_centerVertical: Centra al **View** verticalmente con respecto a los límites de su contenedor

RelativeLayout

layout_centerHorizontal: Centra al **View** horizontalmente con respecto a los límites de su contenedor

The screenshot displays the Android Studio IDE. On the left, the XML editor shows the following code:

```
1 <RelativeLayout android:layout_width="match_parent"
2   android:layout_height="match_parent"
3   xmlns:android="http://schemas.android.com/apk/
4
5   <Button
6       android:layout_width="100dp"
7       android:layout_height="100dp"
8       android:textSize="50dp"
9       android:layout_centerHorizontal="true"
10      android:text="1"/>
11 </RelativeLayout>
```

The line `android:layout_centerHorizontal="true"` is highlighted with a red background. On the right, the Design tab shows a visual representation of the layout on a mobile device screen. A blue header bar at the top contains the text "Layouts" and a status bar with the time "7:00". Below the header, a large gray square with the number "1" is centered horizontally. At the bottom of the screen, there is a black navigation bar with three icons: a back arrow, a circle, and a square. The interface includes a ruler at the top and a vertical ruler on the right side of the design preview.

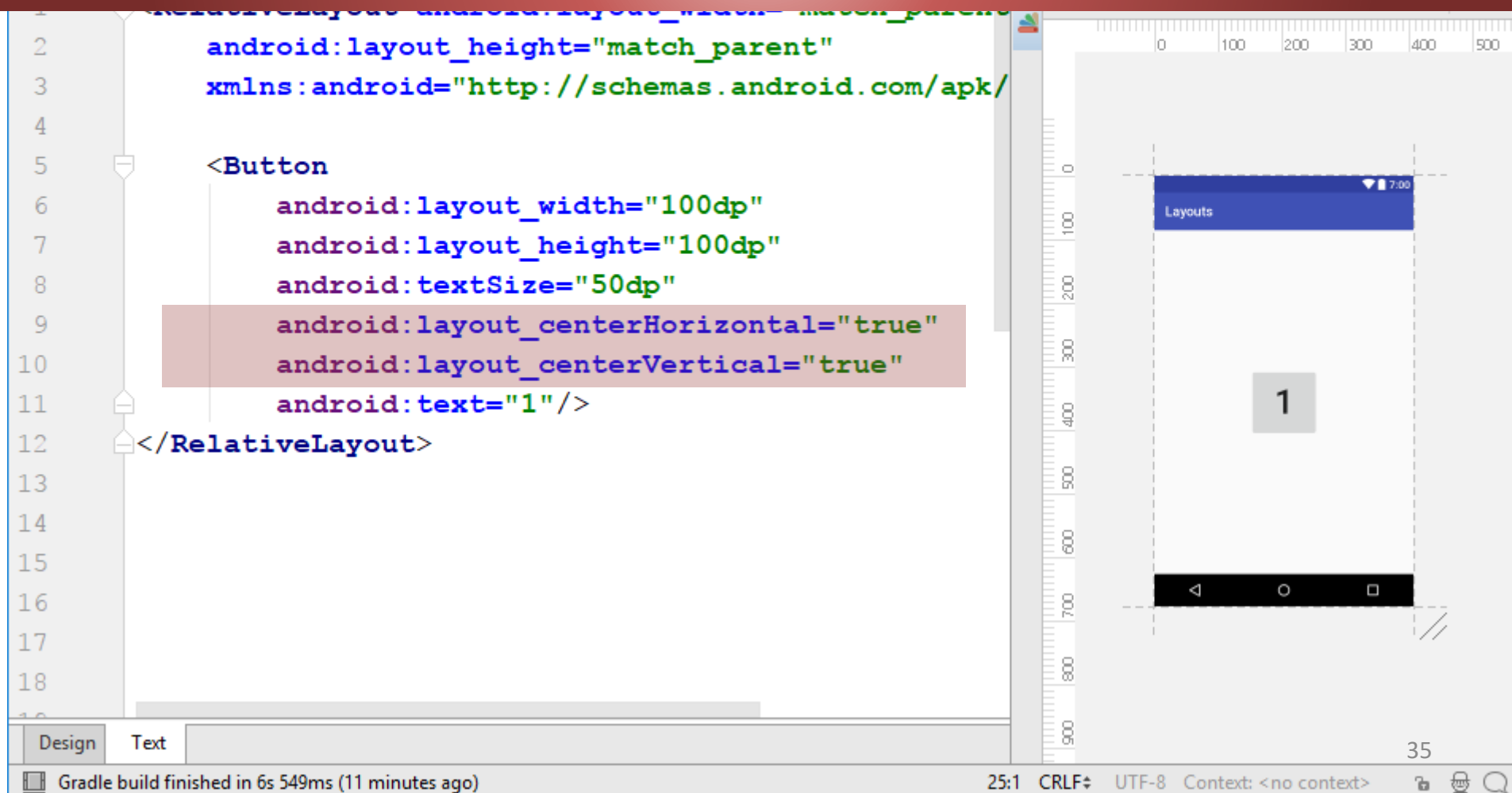
Design Text

Gradle build finished in 6s 549ms (7 minutes ago)

25:1 CRLF UTF-8 Context: <no context>

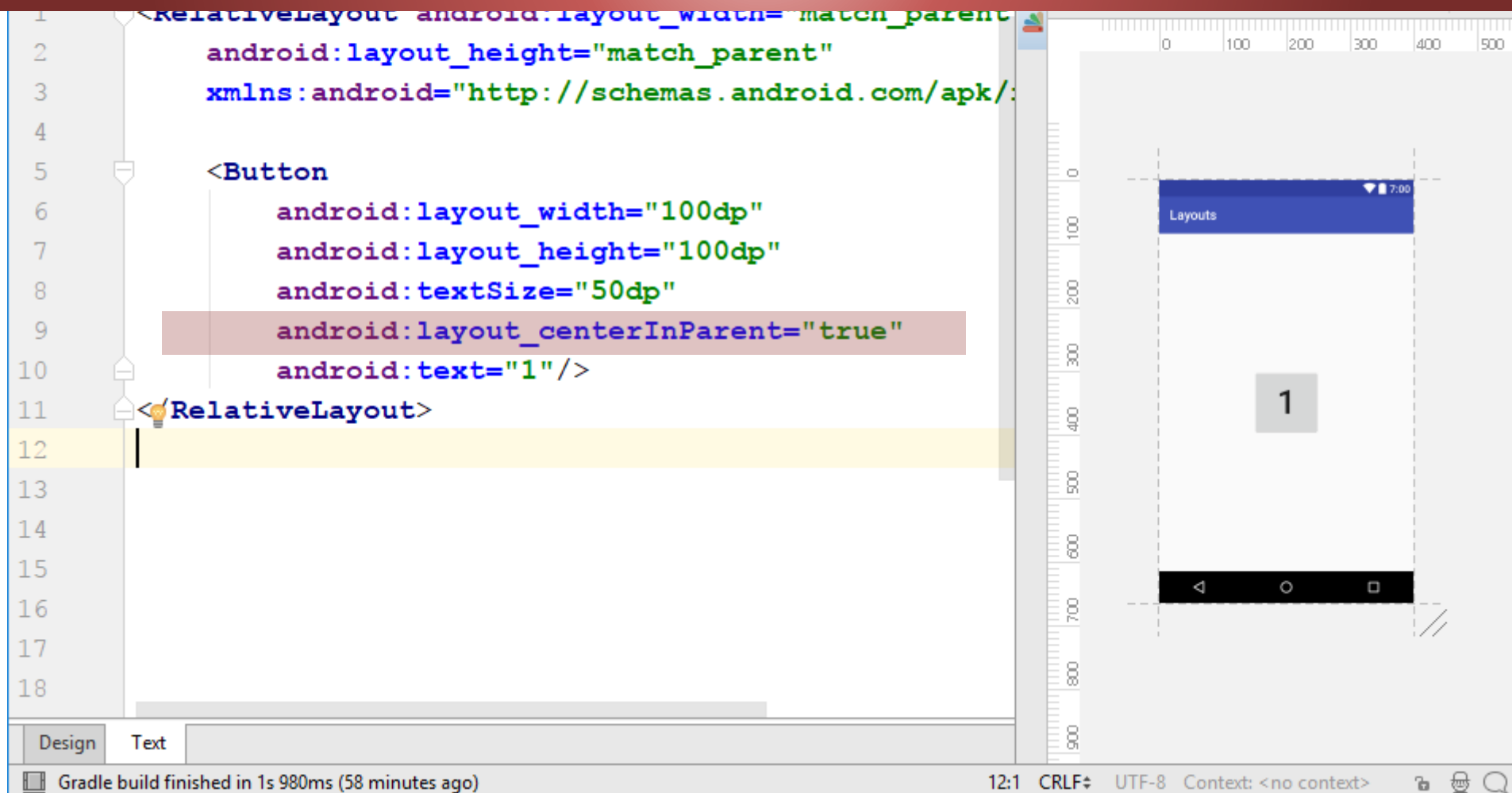
RelativeLayout

layout_centerVertical combinado con **layout_centerHorizontal** central al **View** con respecto a los límites de su contenedor



RelativeLayout

layout_centerInParent: Centra al **View** con respecto a los límites de su contenedor (mismo efecto que el anterior)



RelativeLayout

layout_alignParentBottom: Alinea el borde inferior del **View** con el borde inferior del contenedor

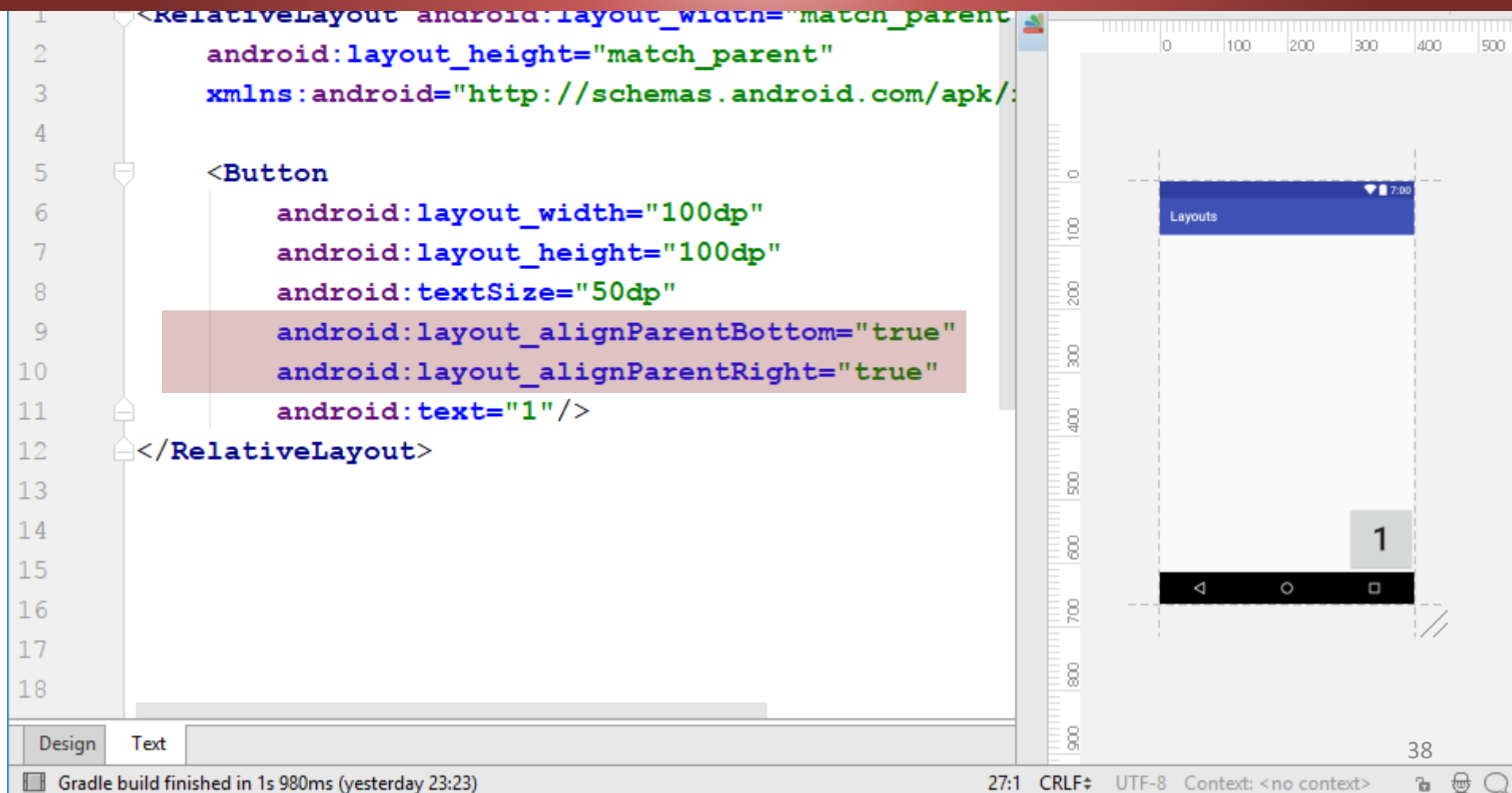
The screenshot displays the Android Studio IDE. On the left, the XML editor shows the following code:

```
1 <RelativeLayout android:layout_width="match_parent"
2   android:layout_height="match_parent"
3   xmlns:android="http://schemas.android.com/apk/
4
5   <Button
6     android:layout_width="100dp"
7     android:layout_height="100dp"
8     android:textSize="50dp"
9     android:layout_centerInParent="true"
10    android:layout_alignParentBottom="true"
11    android:text="1"/>
12 </RelativeLayout>
```

The line `android:layout_alignParentBottom="true"` is highlighted with a red background. On the right, the Design tab shows a visual representation of the layout on a mobile device. A blue header bar at the top contains the text "Layouts". At the bottom of the screen, a large grey square with the number "1" is positioned, aligned with the bottom edge of the device's screen area. The status bar at the very bottom shows the time as 7:00. The bottom of the screen features a toolbar with "Design" and "Text" tabs, and a status bar with system information including "Gradle build finished in 1s 980ms (yesterday 23:23)", "27:1", "CRLF", "UTF-8", "Context: <no context>", and icons for search, help, and other functions.

RelativeLayout

layout_alignParentRight: Alinea el borde derecho del **View** con el borde derecho de su contenedor



The screenshot displays the Android Studio interface. On the left, the XML code for a RelativeLayout is shown, with a red box highlighting the alignment attributes for a Button. On the right, the design preview shows a blue header bar and a button labeled '1' positioned at the bottom right corner of the screen. The status bar at the bottom indicates the build status and system information.

```
1 <RelativeLayout android:layout_width="match_parent"
2   android:layout_height="match_parent"
3   xmlns:android="http://schemas.android.com/apk/
4
5   <Button
6     android:layout_width="100dp"
7     android:layout_height="100dp"
8     android:textSize="50dp"
9     android:layout_alignParentBottom="true"
10    android:layout_alignParentRight="true"
11    android:text="1"/>
12 </RelativeLayout>
```

Design Preview: A blue header bar with the text "Layouts" is at the top. A button labeled "1" is positioned at the bottom right corner of the screen. The status bar at the bottom shows the time 7:00 and battery level.

Bottom Bar: Design Text

Bottom Status Bar: Gradle build finished in 1s 980ms (yesterday 23:23) 27:1 CRLF UTF-8 Context: <no context>

RelativeLayout

layout_alignParentLeft: Alinea el borde izquierdo del **View** con el borde izquierdo de su contenedor

The screenshot displays the Android Studio IDE. On the left, the XML editor shows the following code:

```
1 <RelativeLayout android:layout_width="match_parent"
2   android:layout_height="match_parent"
3   xmlns:android="http://schemas.android.com/apk/
4
5   <Button
6     android:layout_width="100dp"
7     android:layout_height="100dp"
8     android:textSize="50dp"
9     android:layout_alignParentBottom="true"
10    android:layout_alignParentRight="true"
11    android:layout_alignParentLeft="true"
12    android:text="1"/>
13 </RelativeLayout>
14
15
16
17
18
```

The XML code is partially highlighted with a red box. On the right, the Design tab shows a visual representation of the layout on a mobile device. It features a blue header bar with the text "Layouts", a large gray rectangular area containing the number "1", and a black navigation bar at the bottom with standard Android icons. The design view includes a ruler at the top and a vertical ruler on the right side.

At the bottom of the screen, the status bar shows "Gradle build finished in 1s 980ms (yesterday 23:23)", the time "24:1", and the encoding "UTF-8". The page number "39" is visible in the bottom right corner.

La mayoría de los atributos que posicionan los **Views** dentro de un **RelativeLayout** afectan a uno de los bordes del **View**, es por eso que al combinarlos puede cambiar la dimensión de este elemento.

RelativeLayout

layout_alignParentTop: Alinea el borde superior del **View** con el borde superior de su contenedor

The screenshot displays the Android Studio IDE. On the left, the XML code for a `RelativeLayout` is shown. A `Button` is defined with the following attributes:

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

    <Button
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:textSize="50dp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true"
        android:text="1"/>

</RelativeLayout>
```

The attributes `android:layout_alignParentBottom="true"`, `android:layout_alignParentRight="true"`, `android:layout_alignParentLeft="true"`, and `android:layout_alignParentTop="true"` are highlighted with a red box, indicating they all apply simultaneously to the button.

On the right, the Design view shows a visual representation of the layout. It features a blue header bar labeled "Layouts" and a large gray rectangular area containing the number "1". The number is positioned at the top-left corner of the gray area, demonstrating the effect of the `layout_alignParentTop` and `layout_alignParentLeft` attributes. The interface includes a ruler at the top and bottom, and a status bar at the bottom showing the time as 7:00.

At the bottom of the screen, the status bar indicates the Gradle build finished in 1s 980ms (yesterday 23:23), the current time is 22:1, and the encoding is UTF-8.

RelativeLayout

layout_above: Alinea el borde inferior de un **View** con el borde superior de otro **View**

The screenshot displays the Android Studio IDE. On the left, the XML code for a `RelativeLayout` is shown in the 'Text' tab. The code defines two buttons: 'boton1' and '2'. Button '2' is positioned above button '1' using the `layout_above` attribute. The 'Design' tab on the right shows a visual representation of this layout on a mobile device screen. Two gray rectangular boxes, labeled '1' and '2', are shown. Box '2' is positioned directly above box '1', with a dashed line and an arrow indicating the alignment of their top and bottom edges. The status bar at the bottom of the screen shows 'Gradle build finished in 3s 641ms (2 minutes ago)', '26:1 CRLF UTF-8 Context: <no context>', and a page number '42'.

```
1 <RelativeLayout android:layout_width="match_parent"
2   android:layout_height="match_parent"
3   xmlns:android="http://schemas.android.com/apk/
4   <Button
5       android:id="@+id/boton1"
6       android:layout_width="150dp"
7       android:layout_height="150dp"
8       android:textSize="50dp"
9       android:layout_centerInParent="true"
10      android:text="1"/>
11   <Button
12       android:layout_width="100dp"
13       android:layout_height="100dp"
14       android:textSize="50dp"
15       android:text="2"
16       android:layout_above="@id/boton1"/>
17 </RelativeLayout>
18
```

RelativeLayout

layout_alignTop: Alinea el borde superior de un **View** con el borde superior de otro **View**

The screenshot displays the Android Studio IDE. On the left, the XML code for a `RelativeLayout` is shown in the 'Text' tab. The code defines two buttons: '1' and '2'. Button '1' is centered and has a width and height of 150dp. Button '2' is positioned below button '1' and has a width and height of 100dp. The `layout_alignTop` attribute is used on button '2' to align its top edge with the bottom edge of button '1'.

```
1 <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     android:layout_width="match_parent"
3     android:layout_height="match_parent"
4     xmlns:android="http://schemas.android.com/apk/res/android">
5     <Button
6         android:id="@+id/boton1"
7         android:layout_width="150dp"
8         android:layout_height="150dp"
9         android:textSize="50dp"
10        android:layout_centerInParent="true"
11        android:text="1"/>
12    <Button
13        android:layout_width="100dp"
14        android:layout_height="100dp"
15        android:textSize="50dp"
16        android:text="2"
17        android:layout_alignTop="@id/boton1"/>
18 </RelativeLayout>
```

On the right, the 'Design' tab shows a visual representation of the layout. It features a blue header bar labeled 'Layouts' at the top. Below it, two gray squares are shown: square '1' is centered, and square '2' is positioned directly below it, demonstrating the `layout_alignTop` alignment. A black navigation bar is at the bottom. The interface includes a ruler at the top and a vertical ruler on the left of the design area.

Design | Text

Gradle build finished in 6s 549ms (today 19:09) 17:44 CRLF UTF-8 Context: <no context>

RelativeLayout

layout_alignBaseline: Alinea la línea base de un **View** con la línea base de otro **View**.

The screenshot displays the Android Studio IDE. On the left, the XML editor shows the following code:

```
1 <RelativeLayout android:layout_width="match_parent"
2   android:layout_height="match_parent"
3   xmlns:android="http://schemas.android.com/apk/
4   <Button
5       android:id="@+id/boton1"
6       android:layout_width="150dp"
7       android:layout_height="150dp"
8       android:textSize="50dp"
9       android:layout_centerInParent="true"
10      android:text="1"/>
11   <Button
12       android:layout_width="100dp"
13       android:layout_height="100dp"
14       android:textSize="50dp"
15       android:text="2"
16       android:layout_alignBaseline="@id/boton1"/
17 </RelativeLayout>
18
```

On the right, the Design view shows a visual representation of the layout. It features a blue header bar labeled 'Layouts' at the top. Below it, two gray rectangular buttons are shown. Button '1' is centered in the upper half of the screen. Button '2' is positioned to the left of button '1', and its baseline is aligned with the baseline of button '1', as indicated by a horizontal dashed line. A black navigation bar is visible at the bottom of the design area. The interface includes a ruler at the top and left, and a status bar at the bottom showing '44'.

Design Text

Gradle build finished in 3s 641ms (4 minutes ago)

1:1 CRLF UTF-8 Context: <no context>

RelativeLayout

layout_alignBottom: Alinea el borde inferior de un **View** con el borde inferior de otro **View**

The screenshot displays the Android Studio IDE. On the left, the XML code for a `RelativeLayout` is shown. It contains two `Button` elements. The first button, with ID `@+id/boton1`, is centered and has a width and height of `150dp`. The second button, with text "2", has a width and height of `100dp` and is aligned to the bottom of the first button using the `layout_alignBottom="@id/boton1"` attribute. On the right, the design preview shows a mobile screen with a blue header labeled "Layouts". Two gray rectangular buttons are visible: one labeled "1" (corresponding to `boton1`) and one labeled "2" positioned directly below it. A vertical dashed line indicates the alignment of the bottom edges of both buttons. The bottom of the screen shows the Android navigation bar. The status bar at the top right indicates a time of 7:00. The bottom status bar shows the system time as 16:37, the encoding as UTF-8, and the context as "<no context>".

```
1 <RelativeLayout android:layout_width="match_parent"
2   android:layout_height="match_parent"
3   xmlns:android="http://schemas.android.com/apk/
4   <Button
5       android:id="@+id/boton1"
6       android:layout_width="150dp"
7       android:layout_height="150dp"
8       android:textSize="50dp"
9       android:layout_centerInParent="true"
10      android:text="1"/>
11   <Button
12       android:layout_width="100dp"
13       android:layout_height="100dp"
14       android:textSize="50dp"
15       android:text="2"
16       android:layout_alignBottom="@id/boton1"/>
17 </RelativeLayout>
18
```

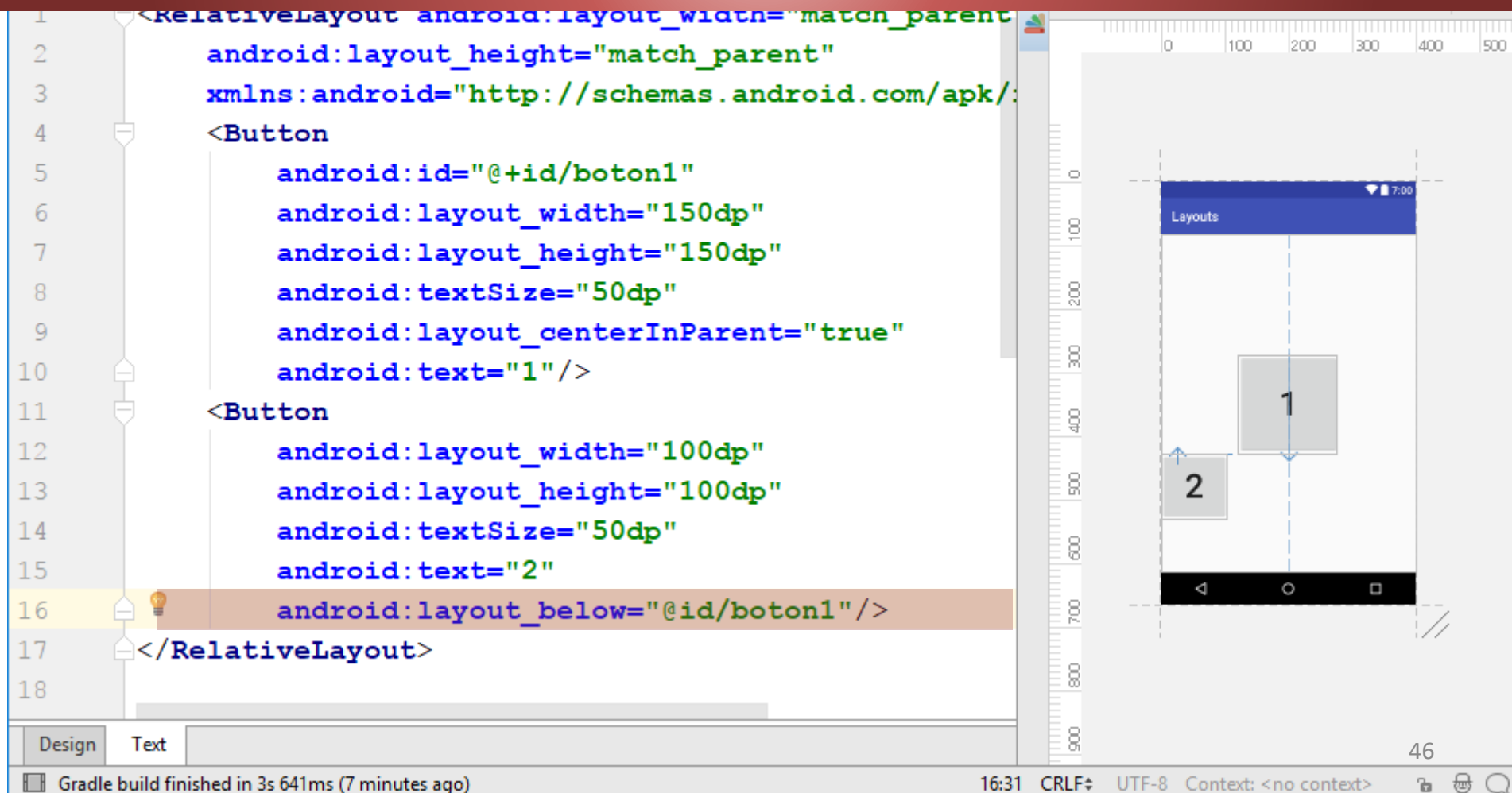
Design Preview: A mobile screen showing two buttons. Button 1 is centered. Button 2 is positioned below Button 1, aligned to its bottom edge. A vertical dashed line indicates the alignment of the bottom edges of both buttons.

Bottom status bar: 16:37 CRLF UTF-8 Context: <no context>

RelativeLayout

layout_below: Alinea el borde superior de un **View** con el borde inferior de otro **View**

```
1 <RelativeLayout android:layout_width="match_parent"
2   android:layout_height="match_parent"
3   xmlns:android="http://schemas.android.com/apk/
4   <Button
5       android:id="@+id/boton1"
6       android:layout_width="150dp"
7       android:layout_height="150dp"
8       android:textSize="50dp"
9       android:layout_centerInParent="true"
10      android:text="1"/>
11   <Button
12       android:layout_width="100dp"
13       android:layout_height="100dp"
14       android:textSize="50dp"
15       android:text="2"
16       android:layout_below="@id/boton1"/>
17 </RelativeLayout>
18
```



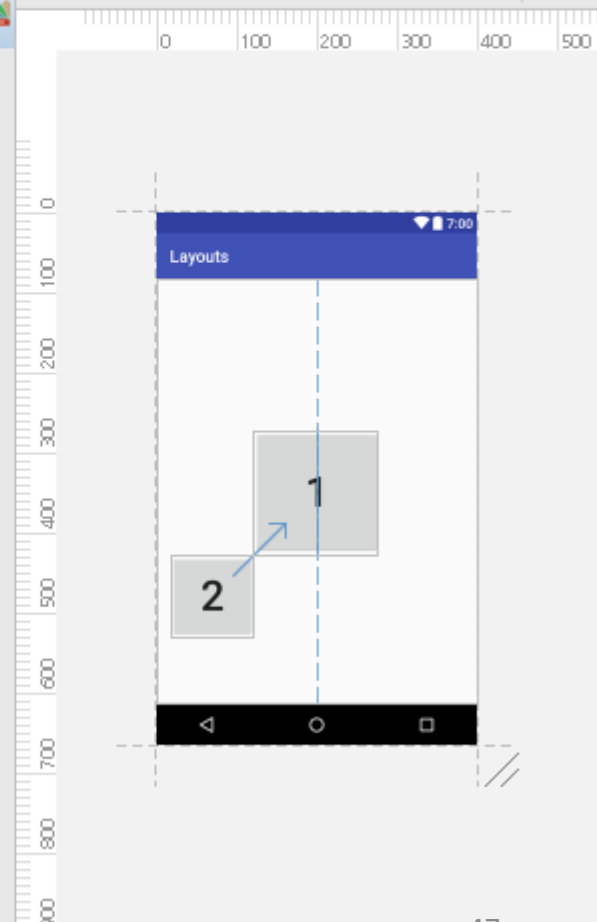
Design Text

Gradle build finished in 3s 641ms (7 minutes ago) 16:31 CRLF UTF-8 Context: <no context>

RelativeLayout

layout_toLeftOf: Alinea el borde derecho de un **View** con el borde izquierdo de otro **View**

```
1 <RelativeLayout android:layout_width="match_parent"
2   android:layout_height="match_parent"
3   xmlns:android="http://schemas.android.com/apk/
4   <Button
5       android:id="@+id/boton1"
6       android:layout_width="150dp"
7       android:layout_height="150dp"
8       android:textSize="50dp"
9       android:layout_centerInParent="true"
10      android:text="1"/>
11   <Button
12       android:layout_width="100dp"
13       android:layout_height="100dp"
14       android:textSize="50dp"
15       android:text="2"
16       android:layout_below="@id/boton1"
17       android:layout_toLeftOf="@id/boton1"/>
18 </RelativeLayout>
```



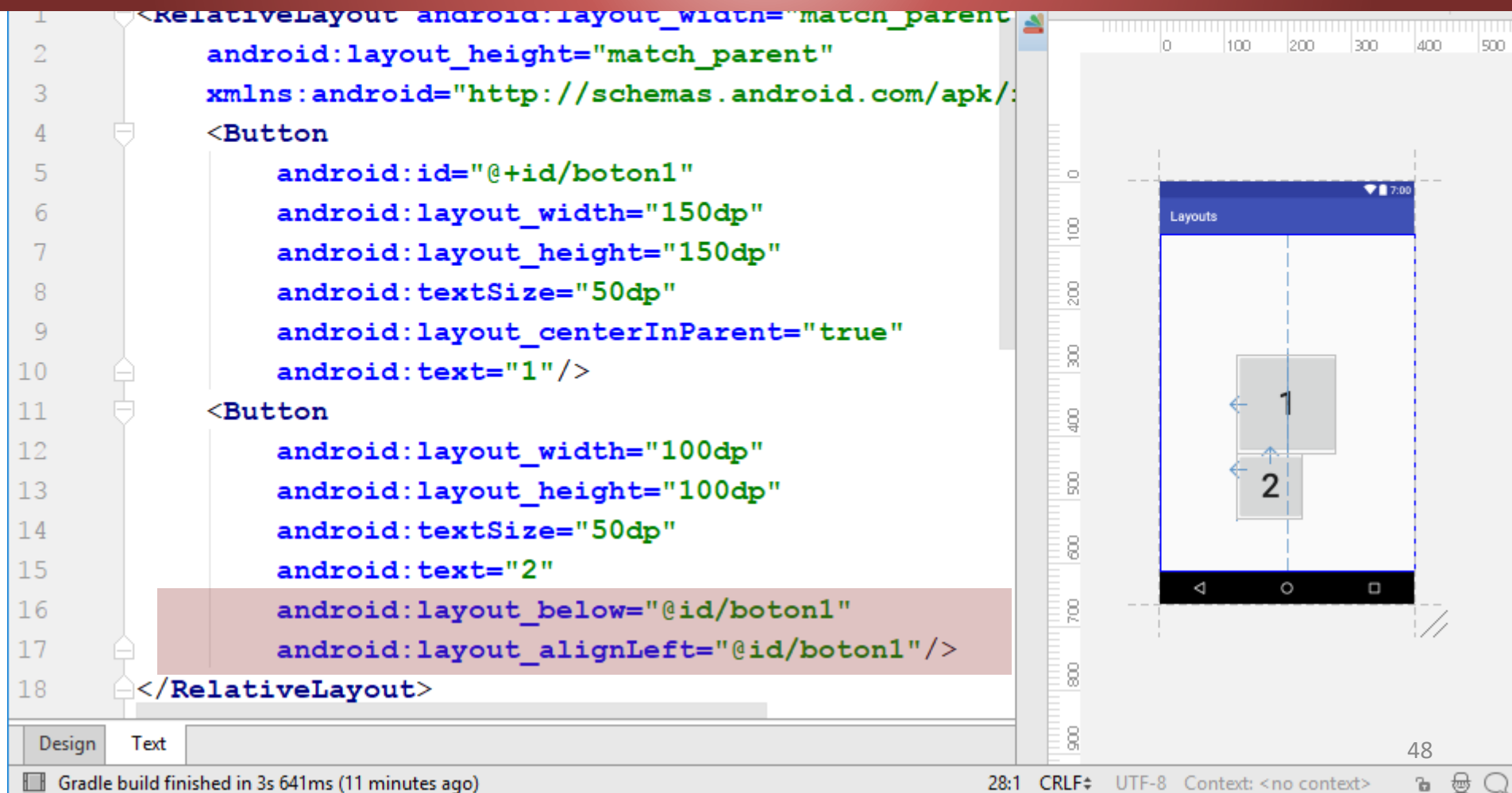
The diagram shows a mobile screen with a blue header bar labeled 'Layouts' and a black navigation bar at the bottom. Two gray squares represent views. View 1 is a larger square centered horizontally and vertically. View 2 is a smaller square positioned to the left of View 1. A dashed vertical line and an arrow indicate that the right edge of View 2 is aligned with the left edge of View 1, demonstrating the `layout_toLeftOf` property.

Design Text

Gradle build finished in 3s 641ms (9 minutes ago) 17:45 CRLF UTF-8 Context: <no context>

RelativeLayout

layout_alignLeft: Alinea el borde izquierdo de un **View** con el borde izquierdo de otro **View**.



The screenshot displays the Android Studio IDE. On the left, the XML code for a `RelativeLayout` is shown in the 'Text' tab. The code defines two buttons: 'boton1' and '2'. Button '2' is positioned below and to the left of 'boton1' using the `layout_below` and `layout_alignLeft` attributes. The 'Design' tab on the right shows a visual representation of this layout on a mobile device screen. Two gray rectangular buttons are visible: a larger one at the top and a smaller one below it, aligned to the left. Blue dashed lines and arrows indicate the alignment constraints. The status bar at the bottom shows 'Gradle build finished in 3s 641ms (11 minutes ago)', the time '28:1', and the encoding 'UTF-8'.

```
1 <RelativeLayout android:layout_width="match_parent"
2   android:layout_height="match_parent"
3   xmlns:android="http://schemas.android.com/apk/:"
4   <Button
5       android:id="@+id/boton1"
6       android:layout_width="150dp"
7       android:layout_height="150dp"
8       android:textSize="50dp"
9       android:layout_centerInParent="true"
10      android:text="1"/>
11   <Button
12       android:layout_width="100dp"
13       android:layout_height="100dp"
14       android:textSize="50dp"
15       android:text="2"
16       android:layout_below="@id/boton1"
17       android:layout_alignLeft="@id/boton1"/>
18 </RelativeLayout>
```

48

Gradle build finished in 3s 641ms (11 minutes ago) 28:1 CRLF UTF-8 Context: <no context>

RelativeLayout

layout_alignRight: Alinea el borde derecho de un **View** con el borde derecho de otro **View**

The screenshot displays the Android Studio IDE. On the left, the XML editor shows the following code:

```
1 <RelativeLayout android:layout_width="match_parent"
2   android:layout_height="match_parent"
3   xmlns:android="http://schemas.android.com/apk/
4   <Button
5       android:id="@+id/boton1"
6       android:layout_width="150dp"
7       android:layout_height="150dp"
8       android:textSize="50dp"
9       android:layout_centerInParent="true"
10      android:text="1"/>
11   <Button
12       android:layout_width="100dp"
13       android:layout_height="100dp"
14       android:textSize="50dp"
15       android:text="2"
16       android:layout_below="@id/boton1"
17       android:layout_alignRight="@id/boton1"/>
18 </RelativeLayout>
```

On the right, the Design tab shows a visual representation of the layout on a mobile device. It features two overlapping gray squares labeled '1' and '2'. Square '1' is centered in the parent container. Square '2' is positioned below square '1' and is aligned to the right edge of square '1', as indicated by a dashed vertical line and an arrow. The interface includes a ruler at the top and bottom, and a status bar at the bottom right showing the time 17:36 and various system icons.

RelativeLayout

layout_toRightOf: Alinea el borde izquierdo de un **View** con el borde derecho de otro **View**

The screenshot displays the Android Studio IDE. On the left, the XML code for a `RelativeLayout` is shown. The code defines two buttons: `boton1` and `boton2`. `boton1` is centered in the parent layout. `boton2` is positioned below and to the right of `boton1` using the `layout_toRightOf` attribute. The code is as follows:

```
1 <RelativeLayout android:layout_width="match_parent"
2   android:layout_height="match_parent"
3   xmlns:android="http://schemas.android.com/apk/:"
4   <Button
5       android:id="@+id/boton1"
6       android:layout_width="150dp"
7       android:layout_height="150dp"
8       android:textSize="50dp"
9       android:layout_centerInParent="true"
10      android:text="1"/>
11   <Button
12       android:layout_width="100dp"
13       android:layout_height="100dp"
14       android:textSize="50dp"
15       android:text="2"
16       android:layout_below="@id/boton1"
17       android:layout_toRightOf="@id/boton1"/>
18 </RelativeLayout>
```

On the right, the design preview shows a mobile screen with two buttons. Button 1 is a large square in the center. Button 2 is a smaller square located below and to the right of Button 1. A blue arrow points from Button 2 to Button 1, indicating the alignment. The interface includes a ruler at the top and bottom, and a status bar at the bottom showing the time as 7:00.

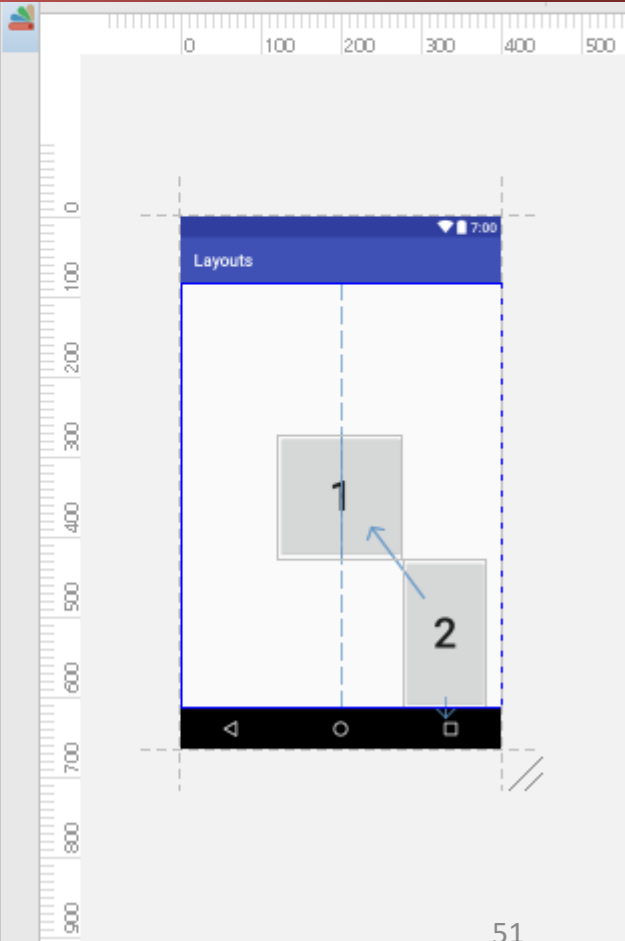
At the bottom of the screen, the status bar shows the following information:

- Gradle build finished in 3s 641ms (13 minutes ago)
- 21:1 CRLF UTF-8 Context: <no context>
- 50

RelativeLayout

Se pueden combinar distintas disposiciones para obtener resultados muy variados

```
android:layout_height="match_parent"
xmlns:android="http://schemas.android.com/apk/
<Button
    android:id="@+id/boton1"
    android:layout_width="150dp"
    android:layout_height="150dp"
    android:textSize="50dp"
    android:layout_centerInParent="true"
    android:text="1"/>
<Button
    android:layout_width="100dp"
    android:layout_height="100dp"
    android:textSize="50dp"
    android:text="2"
    android:layout_below="@id/boton1"
    android:layout_toRightOf="@id/boton1"
    android:layout_alignParentBottom="true"/>
</RelativeLayout>
```

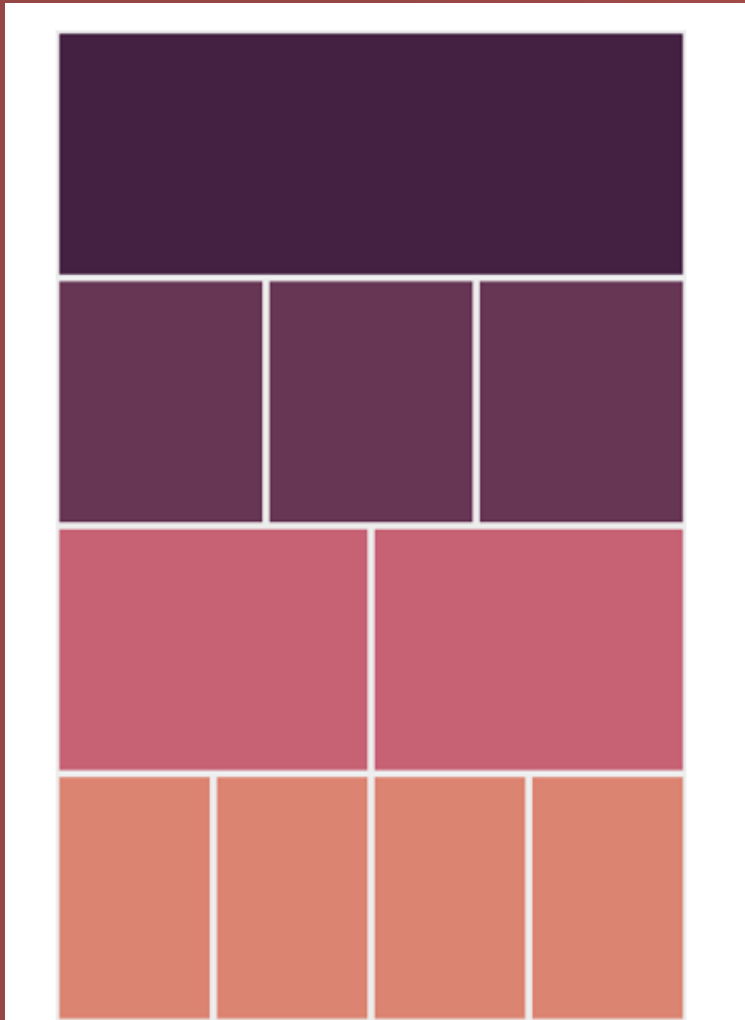


Relacionado con las alineaciones horizontales y teniendo en cuenta los idiomas **RTL** también existen:

- **layout_alignParentStart**
- **layout_alignParentEnd**
- **layout_alignStart**
- **layout_alignEnd**
- **layout_toStartOf**
- **layout_toEndOf**

Para el idioma español **Start** funciona como **Left** y **End** funciona como **Right**. Para los idiomas **RTL** justo lo contrario

TableLayout

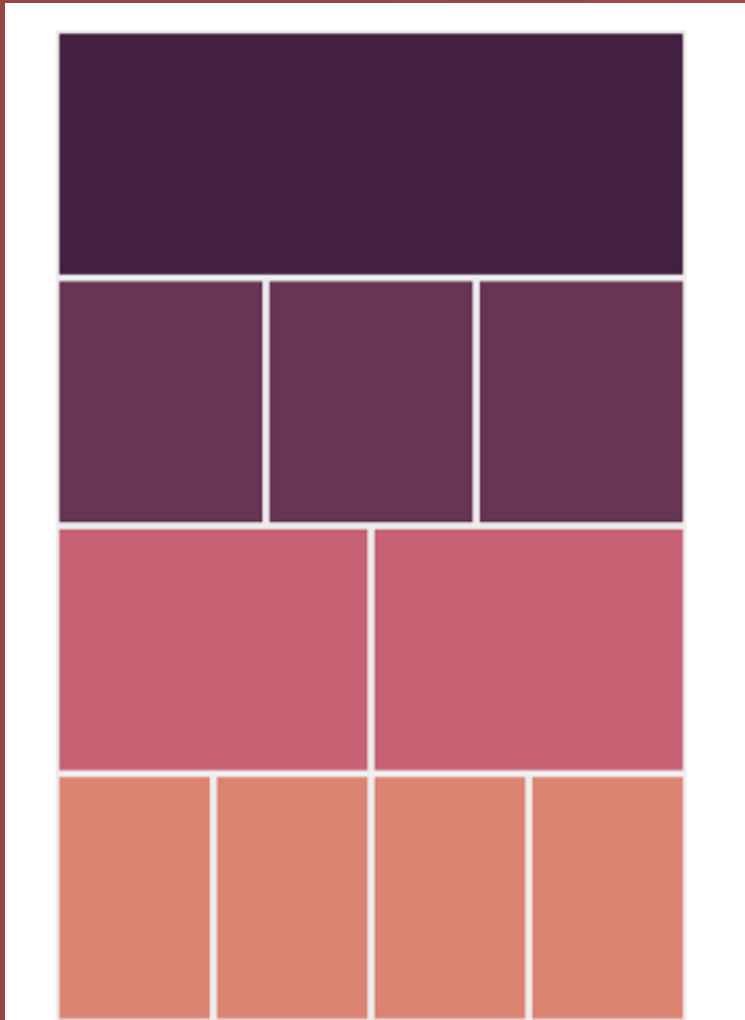


Es un **ViewGroup** que ubica a sus hijos en filas y columnas

Generalmente está compuesto de objetos **TableRow**.

Cada **TableRow** puede tener 0, 1 o más celdas.

TableLayout

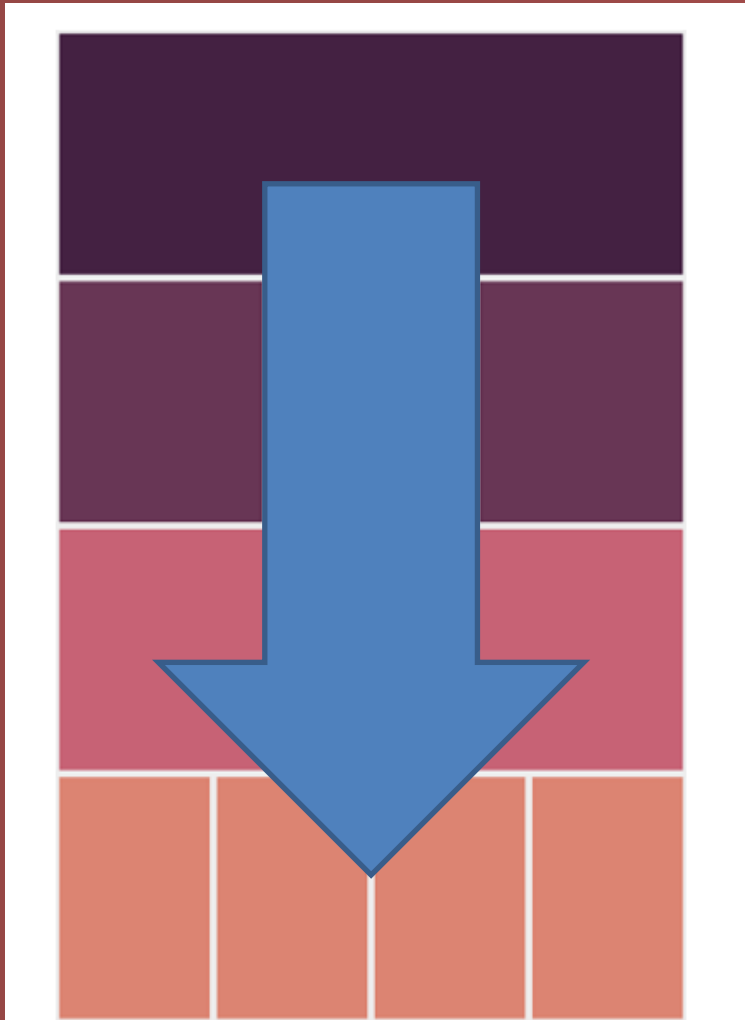


La tabla tendrá tantas columnas como aquel **TableRow** con más celdas.

Una celda podría expandirse ocupando más de una columna.

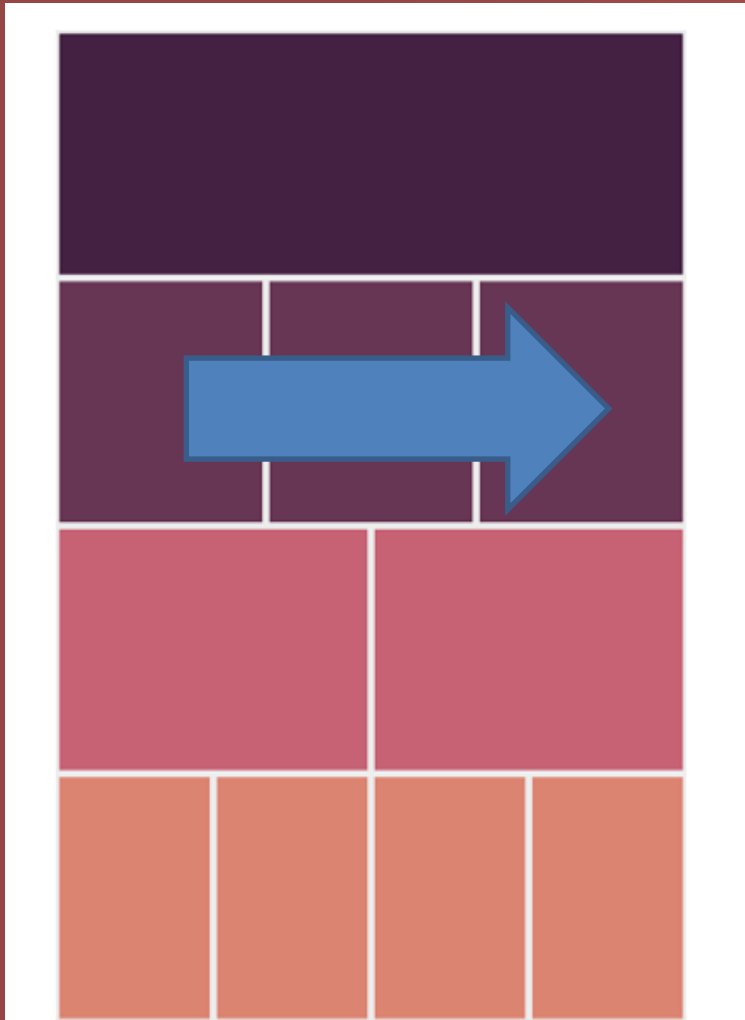
Un **TableLayout** no contempla bordes. Simplemente organiza elementos visuales

TableLayout



Un **TableLayout** es una especialización de un **LinearLayout** con orientación vertical.

TableLayout



Asimismo, cada **TableRow** es una especialización de un **LinearLayout** con orientación horizontal.

Ejemplo de TableLayout

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="*">
    <TableRow>
        <Button
            android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="button1" />
        <Button
            android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="button2" />
        <Button
            android:id="@+id/button3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="button3" />
    </TableRow>
    <TableRow>
        <Button
            android:id="@+id/button4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_span="3"
            android:text="button4" />
    </TableRow>
</TableLayout>
```



Ejemplo de TableLayout

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="*">
    <TableRow>
        <Button
            android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="button1" />
        <Button
            android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="button2" />
        <Button
            android:id="@+id/button3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="button3" />
    </TableRow>
    <TableRow>
        <Button
            android:id="@+id/button4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_span="3"
            android:text="button4" />
    </TableRow>
</TableLayout>
```

Estira las columnas para ocupar todo el ancho del TableLayout



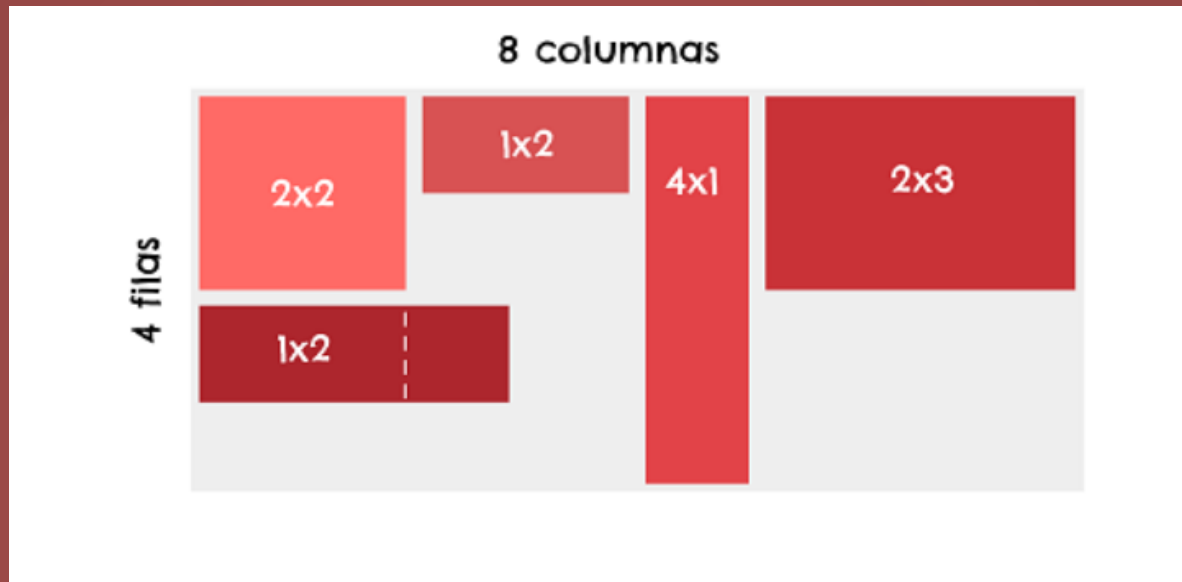
Ejemplo de TableLayout

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="*">
    <TableRow>
        <Button
            android:id="@+id/button1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="button1" />
        <Button
            android:id="@+id/button2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="button2" />
        <Button
            android:id="@+id/button3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="button3" />
    </TableRow>
    <TableRow>
        <Button
            android:id="@+id/button4"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_span="3"
            android:text="button4" />
    </TableRow>
</TableLayout>
```

La vista ocupa 3
columnas

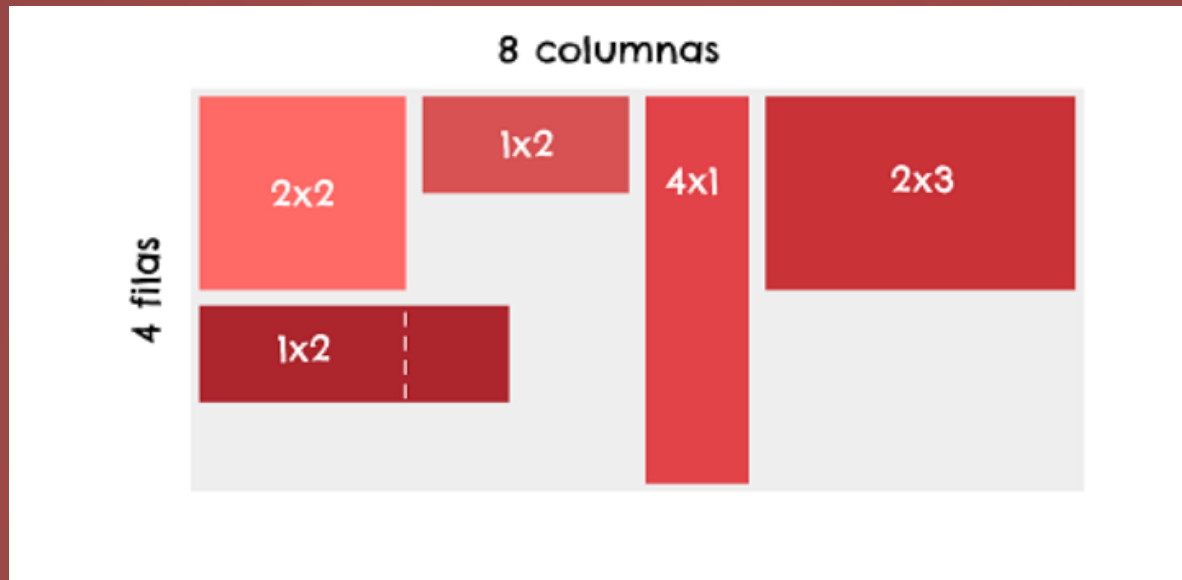


GridLayout



- Incluido a partir de Android 4.0 (API 14)
- Es un **ViewGroup** que ubica a sus hijos en una grilla rectangular.
- Es posible especificar cantidad de filas y columnas.
- A diferencia del **TableLayout**, es posible expandir una celda de forma horizontal o vertical.

GridLayout



- No existe un concepto análogo a **TableRow**. Los elementos hijos se irán colocando ordenadamente por filas o columnas (dependiendo de la propiedad **android:orientation**) hasta completar el número de filas o columnas.

Ejemplo de GridLayout

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="4"
    android:orientation="horizontal" >

    <Button android:text="Botón 1.1" />
    <Button android:text="Botón 1.2" />
    <Button android:text="Botón 1.3"
        android:layout_rowSpan="2"/>
    <Button android:text="Botón 1.4" />

    <Button android:text="Botón 2.1"
        android:layout_columnSpan="2" />
    <Button android:text="Botón 2.4" />

</GridLayout>
```



Ejemplo de GridLayout

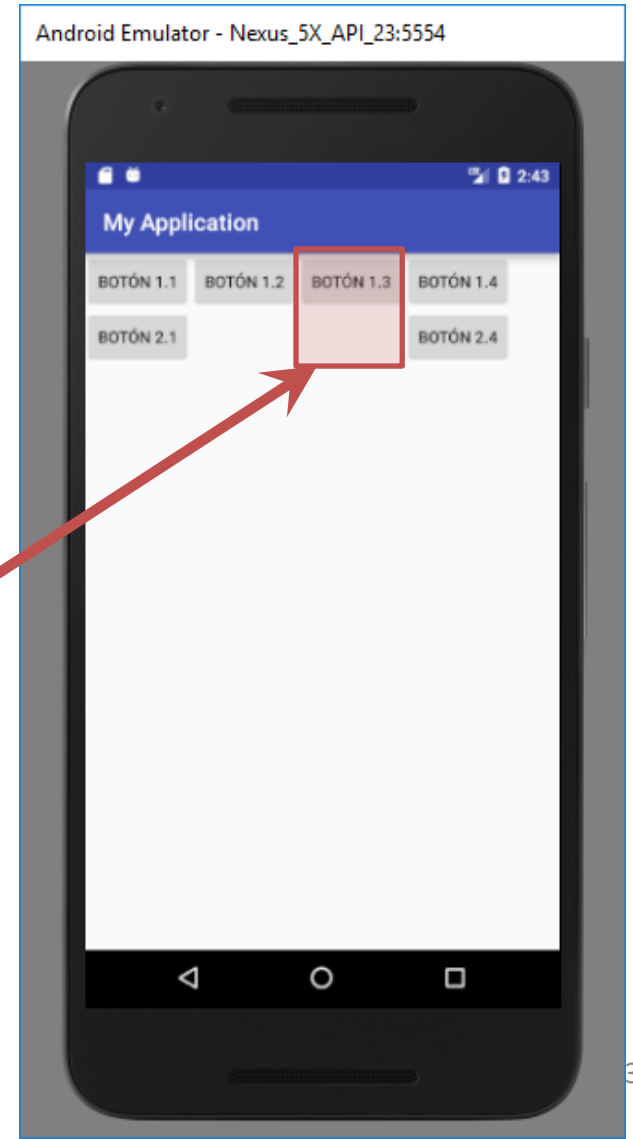
```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="4"
    android:orientation="horizontal" >

    <Button android:text="Botón 1.1" />
    <Button android:text="Botón 1.2" />
    <Button android:text="Botón 1.3"
        android:layout_rowSpan="2"/>
    <Button android:text="Botón 1.4" />

    <Button android:text="Botón 2.1"
        android:layout_columnSpan="2" />
    <Button android:text="Botón 2.4" />

</GridLayout>
```

La vista ocupa 2 filas



Ejemplo de GridLayout

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="4"
    android:orientation="horizontal" >

    <Button android:text="Botón 1.1" />
    <Button android:text="Botón 1.2" />
    <Button android:text="Botón 1.3"
        android:layout_rowSpan="2"/>
    <Button android:text="Botón 1.4" />

    <Button android:text="Botón 2.1"
        android:layout_columnSpan="2" />
    <Button android:text="Botón 2.4" />

</GridLayout>
```

La vista ocupa 2
columnas



Ejemplo de GridLayout

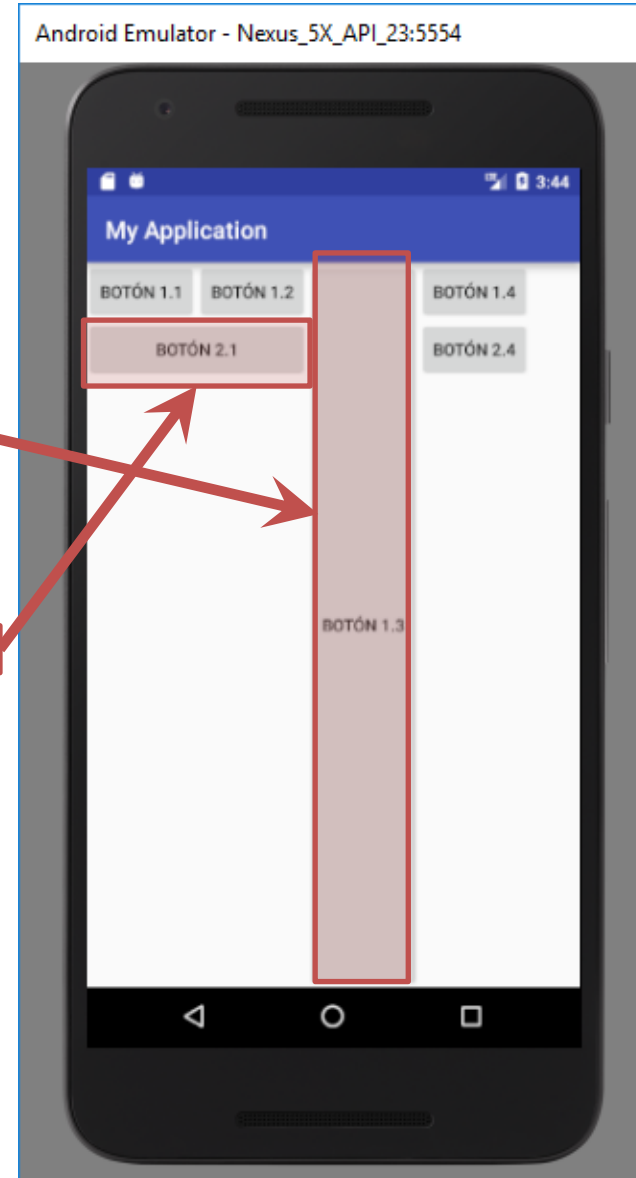
```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="4"
    android:orientation="horizontal" >

    <Button android:text="Botón 1.1" />
    <Button android:text="Botón 1.2" />
    <Button android:text="Botón 1.3"
        android:layout_rowSpan="2"
        android:layout_gravity="fill_vertical"/>
    <Button android:text="Botón 1.4" />

    <Button android:text="Botón 2.1"
        android:layout_columnSpan="2"
        android:layout_gravity="fill_horizontal"/>
    <Button android:text="Botón 2.4" />

</GridLayout>
```

Es posible expandir las vistas para que ocupen toda la celda



Ejemplo de GridLayout

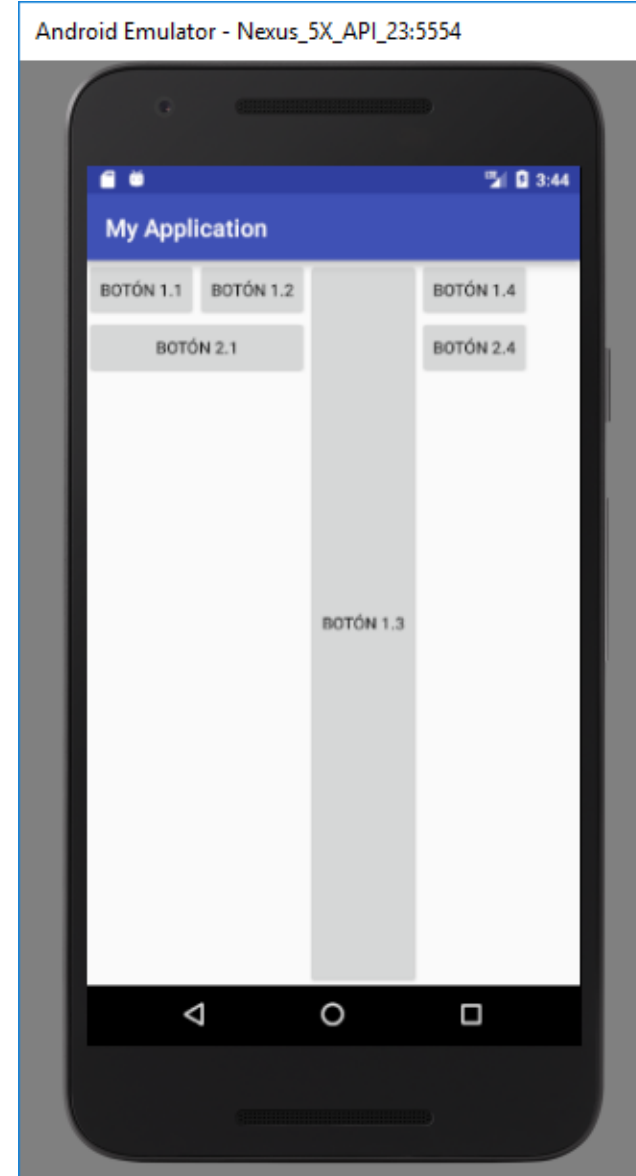
```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="4"
    android:orientation="horizontal" >

    <Button android:text="Botón 1.1" />
    <Button android:text="Botón 1.2" />
    <Button android:text="Botón 1.3"
        android:layout_rowSpan="2"
        android:layout_gravity="fill_vertical"/>
    <Button android:text="Botón 1.4" />

    <Button android:text="Botón 2.1"
        android:layout_columnSpan="2"
        android:layout_gravity="fill_horizontal"/>
    <Button android:text="Botón 2.4" />

</GridLayout>
```

¿Cuál es la razón para que el botón 1.3 se expanda al total del dispositivo?



Ejemplo de GridLayout

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:columnCount="4"
    android:orientation="horizontal" >

    <Button android:text="Botón 1.1" />
    <Button android:text="Botón 1.2" />
    <Button android:text="Botón 1.3"
        android:layout_rowSpan="2"
        android:layout_gravity="fill_vertical"/>
    <Button android:text="Botón 1.4" />

    <Button android:text="Botón 2.1"
        android:layout_columnSpan="2"
        android:layout_gravity="fill_horizontal"/>
    <Button android:text="Botón 2.4" />

</GridLayout>
```

Cambiar el valor a wrap_content
y probar en emulador

