

**UNIVERSIDAD
DE ANTIOQUIA**

1 8 0 3

Sesión 2. Interfaz de Usuario en Android

Ing. Edwin Andrés Cubillos Vega Msc.





Clase View

Creación de una vista

Layout

TextView

Button



Clase View



Creación de una vista



Layout



TextView



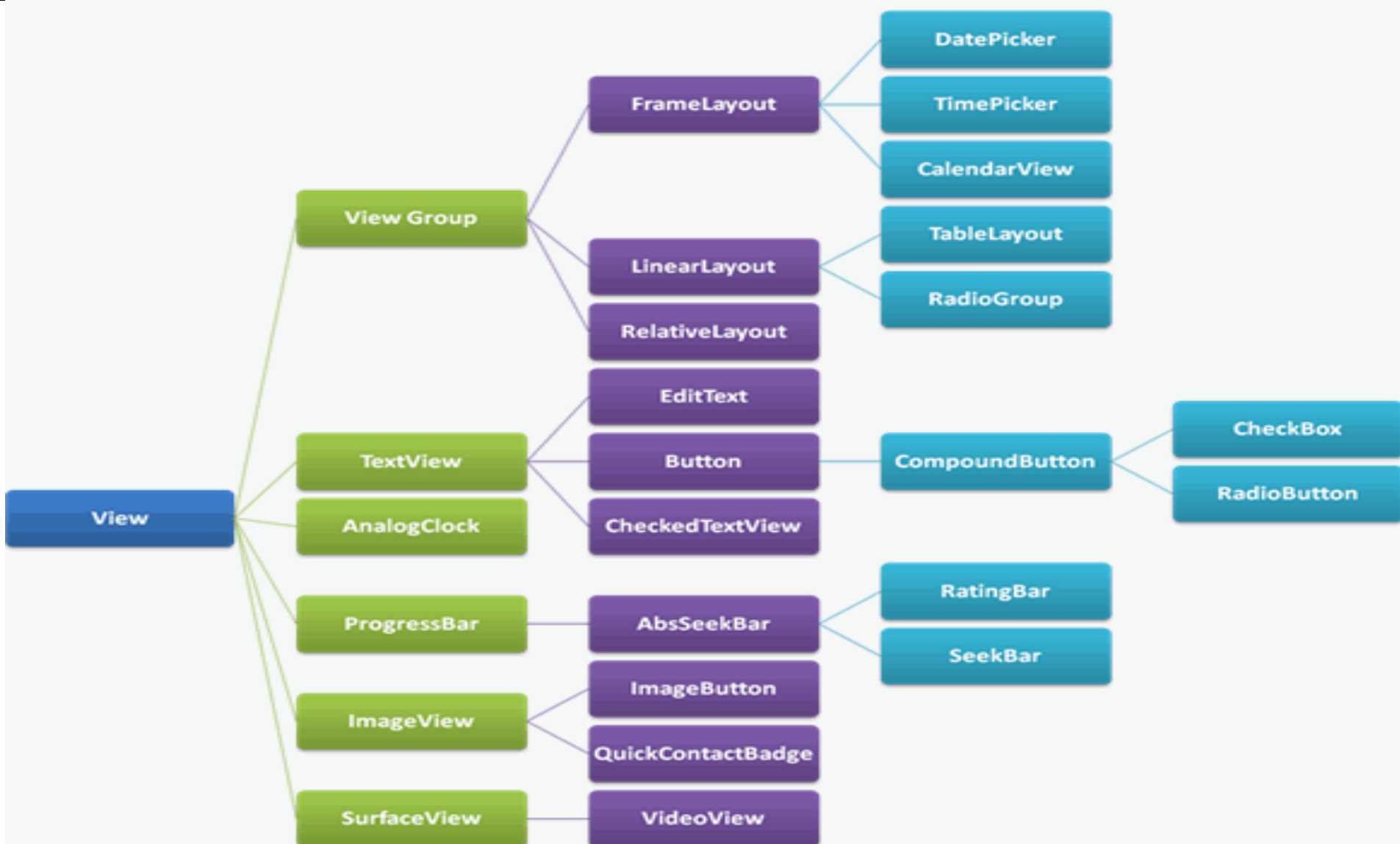
Button

- ❖ Las vistas (view) es la clase básica en Android a partir del cual se crean los elementos de una interfaz de usuario.
- ❖ Contiene numerosas subclases, cada una con funciones específicas
- ❖ Hay una jerarquía que representa a la clase View y todos sus elementos
- ❖ Mas información en este link: [Click](#)
- ❖ [Interfaz de Usuario](#)





Clase View



❖ Atributos de la clase View

Revisar el documento “Sesión 2. Referencia Clase View”

A diagram showing a table of contents. On the left, a large light blue circle is partially visible. A grey arc curves from the top left towards the bottom right, passing through five colored circles (yellow, green, blue, purple, orange). Each circle is followed by a rounded rectangular box containing text.

Clase View

Creación de una vista

Layout

TextView

Button



Creación de una vista

- ❖ Una interfaz en Android se puede desarrollar de 3 métodos diferentes:
 1. Utilizando código java
 2. Utilizando código xml
 3. Utilizando el wizard para Interfaces de usuario
- ❖ Un desarrollador normalmente utiliza el primer y al menos uno de los dos últimos métodos.



Creación de una vista

1. Utilizando código java

```
public class MainActivity extends Activity {
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    setContentView(R.layout.activity_main);
```

```
}
```

→ Crea la Interfaz
Cómo??

Comentar la última línea de código y agregar:

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    //setContentView(R.layout.activity_main);
```

```
    TextView texto = new TextView(this);
```

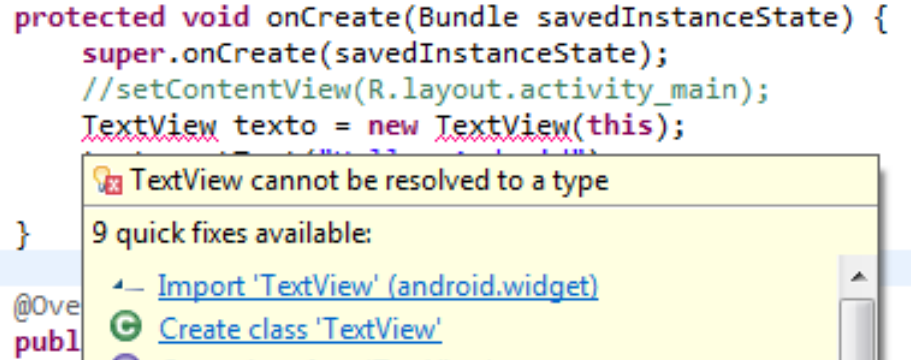
```
    texto.setText("Probando la creación con Java");
```

```
    setContentView(texto);
```

```
}
```

Creación de una vista

- ❖ Al agregar este código se subraya el TextView en rojo, porque Java no lo reconoce.
- ❖ Es necesario agregar el paquete que lo contiene.
- ❖ Coloque el mouse encima del objeto TextView y selección import ...



- ❖ O también se digita Ctrl + Shift + O, y se añaden automáticamente todos los imports faltantes.



Creación de una vista

- ❖ La interfaz de usuario de Android se basa en una clase llamada View (Vista).
- ❖ Una vista es un objeto que se puede dibujar y se utiliza como un elemento en el diseño de la interfaz de usuario:
- ❖ Botón, una imagen, una etiqueta de texto, etc.
- ❖ Cada uno de estos elementos se define como una subclase de la clase View.
- ❖ La subclase para representar un texto es TextView.

❖ Ahora analicemos el código

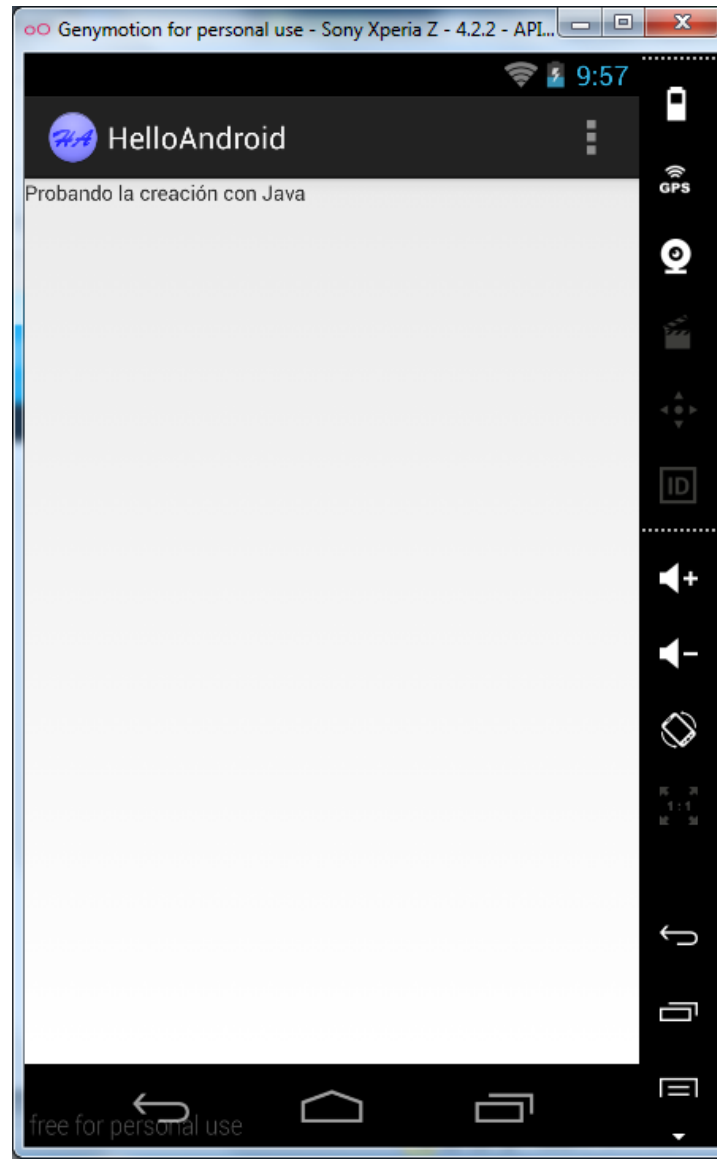
```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    //setContentView(R.layout.activity_main);  
    TextView texto = new TextView(this);  
    texto.setText("Probando la creación con Java");  
    setContentView(texto);  
}
```

Diagrama de anotaciones:

- Una flecha verde apunta desde la línea `TextView texto = new TextView(this);` al número 1.
- Una flecha verde apunta desde la línea `texto.setText("Probando la creación con Java");` al número 2.
- Una flecha verde apunta desde la línea `setContentView(texto);` al número 3.

1. Se crea un objeto de la clase TextView
2. Se define que se visualizará en el TextView mediante setText()
3. Por último con setContentView se indica la vista utilizada por la actividad

Creación de una vista



2. Utilizando lenguaje XML

- ❖ Android proporciona una alternativa para el diseño de interfaces de usuario, los ficheros de diseño basados en XML.
- ❖ Entrar a la carpeta *res/layout/activity_main.xml*

```
<RelativeLayout 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="com.edwinacubillos.helloandroid.MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

```
<RelativeLayout 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="com.edwinacubillos.helloandroid.MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

Creación de una vista

- ❖ RelativeLayout es un contenedor de elementos tipo View.
- ❖ xmlns:android, y xmlns:tools son declaraciones de espacios de nombres de XML que utilizaremos en este fichero (este tipo de parámetro solo es necesario especificarlo en el primer elemento)
- ❖ layout_width y layout_height permiten definir el ancho y el alto de la vista.
- ❖ La tabulación al interior de RelativeLayout indica jerarquía, esto quiere decir que el TextView está al interior de RelativeLayout.

```
<RelativeLayout 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="com.edwinacubillos.helloandroid.MainActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

</RelativeLayout>
```

Creación de una vista

- ❖ @string/hello_world: es una referencia de tipo String
- ❖ Esta referencia se define en el fichero res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">HelloAndroid</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>
</resources>
```

- ❖ Se recomienda utilizar este archivo para el manejo de varios idiomas dentro de la misma aplicación (se amplía luego)



Creación de una vista

- ❖ Modificar el string "hello world"

```
<string name="hello_world">Prueba utilizando XML!</string>
```

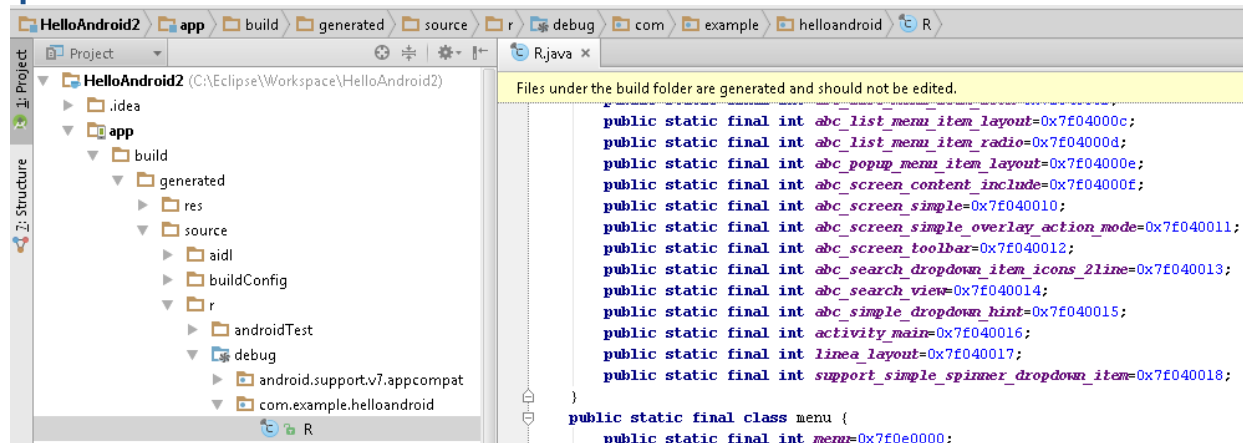
- ❖ En el archivo MainActivity.java eliminar las líneas del item anterior y descomentar

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

- ❖ R.layout.activity_main corresponde a un objeto View que será creado en tiempo de ejecución a partir del recurso *activity_main.xml*

Creación de una vista

- ❖ Android Studio crea automáticamente este identificador en la clase R del proyecto a partir de los elementos de la carpeta *res*

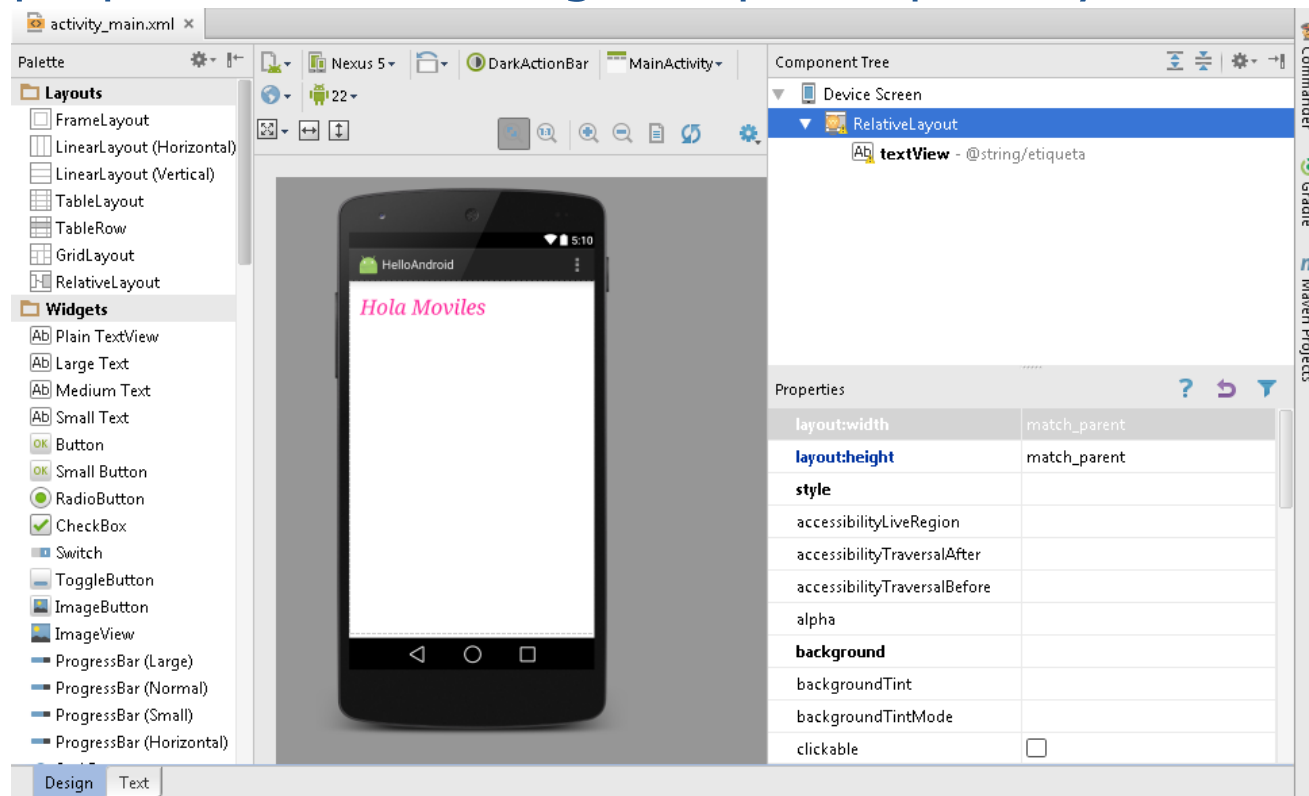


- ❖ Los identificadores de la clase R son números que informan al gestor de recursos, que datos ha de cargar.
- ❖ Por lo tanto no se trata de verdaderos objetos, estos serán creados en tiempo de ejecución solo cuando sea necesario usarlos.

Creación de una vista

3. Utilizando el Wizard

- ❖ Es solo arrastrar y Pegar.
- ❖ El código XML se genera Automáticamente
- ❖ Las propiedades se configuran por el panel y no con código





Clase View



Creación de una vista



Layout

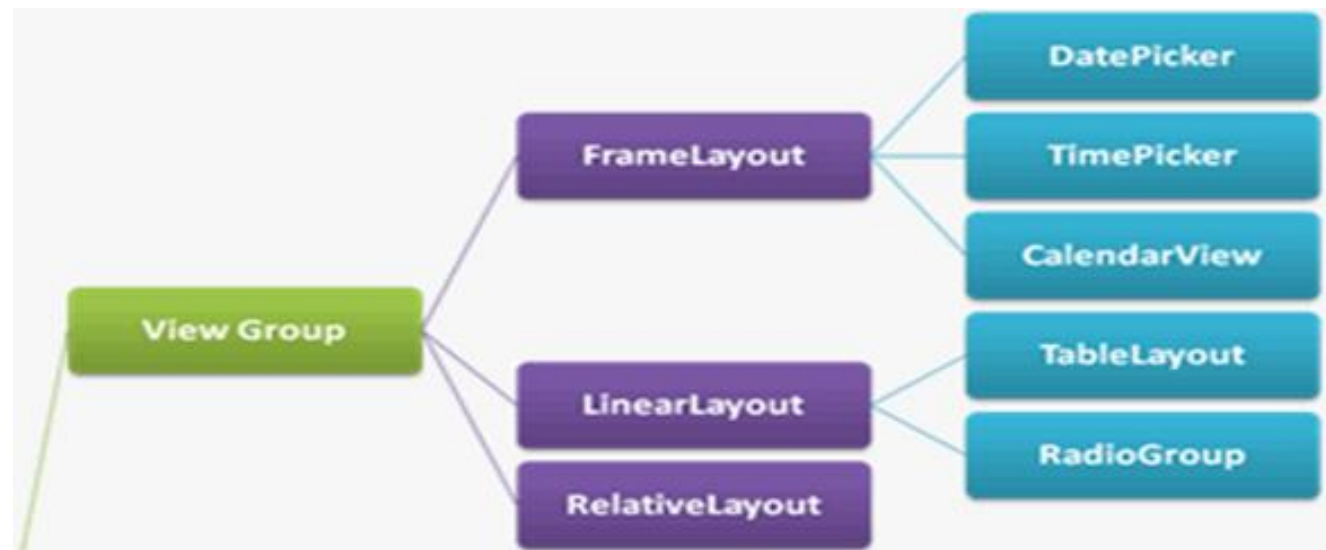


TextView



Button

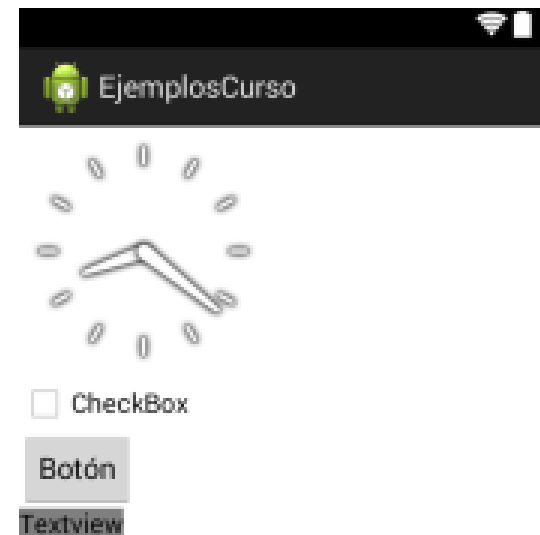
- ❖ Los layouts son elementos no visuales, destinados a controlar la distribución, posición y dimensiones de los controles que se insertan en su interior
- ❖ Estos componentes extienden a la clase base ViewGroup
- ❖ Capaces de contener a otros controles



Hay varios tipos de layouts:

❖ **LinearLayout:** Dispone los elementos en una fila o columna

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">
    <AnalogClock
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="CheckBox" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Botón" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Texto cualquiera" />
</LinearLayout>
```



❖ **TableLayout:** Distribuye los elementos de forma tabular

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <TableRow>
        <AnalogClock
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <CheckBox
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="CheckBox" />
    </TableRow>
    <TableRow>
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Botón" />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Texto cualquiera" />
    </TableRow>
</TableLayout>
```



❖ **RelativeLayout:** Dispone los elementos en relación a otro o al padre.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >
    <AnalogClock
        android:id="@+id/AnalogClock01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true" />
    <CheckBox
        android:id="@+id/CheckBox01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="CheckBox"
        android:layout_below="@+id/AnalogClock01"/>
    <Button
        android:id="@+id/Button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Botón"
        android:layout_below="@+id/CheckBox01" />
    <TextView
        android:id="@+id/TextView01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:text="Texto cualquiera"/>
</RelativeLayout>
```



Otros tipos de layout

- ❖ **ScrollView**. Visualiza una columna de elementos; cuando estos no caben en pantalla se permite un deslizamiento vertical.
- ❖ **ListView**: Visualiza una lista deslizable verticalmente de varios elementos. Su utilización es algo compleja pero muy potente.
- ❖ **GridView**: Visualiza una cuadrícula deslizable de varias filas y varias columnas
- ❖ **TabHost**: Proporciona una lista de ventanas seleccionables por medio de etiquetas que pueden ser pulsadas por el usuario para seleccionar la ventana que desea visualizar.
- ❖ **ViewFlipper**: Permite visualizar una lista de elementos de forma que se visualice uno a la vez, se puede usar para cada cierto intervalo de tiempo.

- ❖ Todos estos Layout se pueden implementar utilizando el wizard gráfico de Android

Layouts

 ConstraintLayout

 GridLayout

 FrameLayout

 LinearLayout (horizontal)

 LinearLayout (vertical)

 RelativeLayout

 TableLayout

 TableRow

 <fragment>

- ❖ Implementar un conversor de dólares a pesos colombianos



Clase View

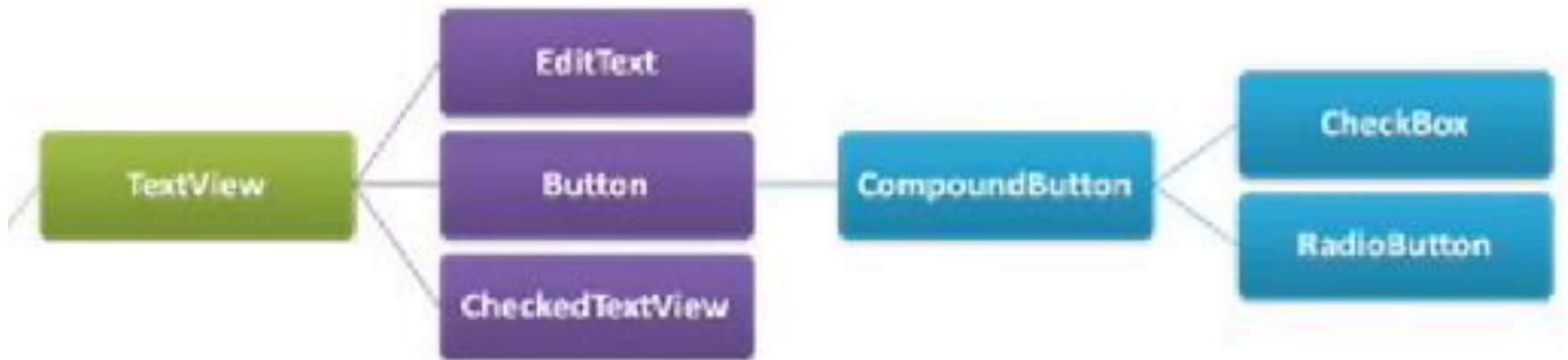
Creación de una vista

Layout

TextView

Button

- ❖ Descendiente de la clase View
- ❖ A su vez hay varias subclases que descienden de TextView



Atributos XML

- ❖ Existen gran variedad de atributos para los TextView
- ❖ Los atributos de los TextView se pueden clasificar en tres categorías:
 - Básicos: text, text_style, typeface, gravity, text_appearances, hint
 - Modifican tamaño del Texto: text_size, width, height, text_scale_x, max_length, lines, max_lines, min_lines
 - modifican Color: text_color, text_color_link, text_color_highlight, text_color_hint
- ❖ El color se puede indicar de tres formas: ([mas información](#))
 - Código Alfa RGB, “#7F00FF00”
 - Colores por defecto de android: @android:color/blue
 - Colores definidos por el usuario: @color/mi_color
- ❖ Es necesario crear un archivo .xml llamado [colors.xml](#) y agregarlo a la carpeta values ubicado en los recursos (res)

The screenshot shows the Android Studio IDE with the following components:

- Project View (Left):** Shows the project structure for 'HelloAndroid2'. The 'res' directory is expanded, showing 'values' which contains 'colors.xml'.
- Editor (Right):** Displays the content of 'colors.xml'. The XML is as follows:

```
<?xml version="1.0" encoding="utf-8" ?>
<resources>
    <item name="black" type="color">#000000</item>
    <item name="navy" type="color"> #000080</item>
    <item name="darkblue" type="color">#00008b</item>
    <item name="mediumblue" type="color">#0000cd</item>
    <item name="blue" type="color">#0000ff</item>
    <item name="midnightblue" type="color">#191970</item>
    <item name="indigo" type="color">#4b0082</item>
    <item name="maroon" type="color">#800000</item>
    <item name="darkred" type="color">#8b0000</item>
    <item name="purple" type="color">#800080</item>
    <item name="darkgreen" type="color">#008000</item>
    <item name="darkmagenta" type="color">#800080</item>
    <item name="darkviolet" type="color">#800080</item>
    <item name="darkslategray" type="color">#2f4f4f</item>
    <item name="darkslateblue" type="color">#483d8b</item>
    <item name="green" type="color">#008000</item>
    <item name="red" type="color">#ff0000</item>
    <item name="firebrick" type="color">#b22222</item>
    <item name="brown" type="color">#a52a2a</item>
    <item name="saddlebrown" type="color">#8b4513</item>
    <item name="crimson" type="color">#dc143c</item>
    <item name="mediumvioletred" type="color">#c71585</item>
    <item name="teal" type="color">#008080</item>
    <item name="blueviolet" type="color">#8a2be2</item>
    <item name="darkolivegreen" type="color">#556b2f</item>
    <item name="forestgreen" type="color">#228b22</item>
    <item name="darkcyan" type="color">#008b8b</item>
    <item name="darkorchid" type="color">#9932cc</item>
    <item name="sienna" type="color">#a0522d</item>
    <item name="deeppink" type="color">#ff1493</item>
    <item name="dimgrey" type="color">#696969</item>

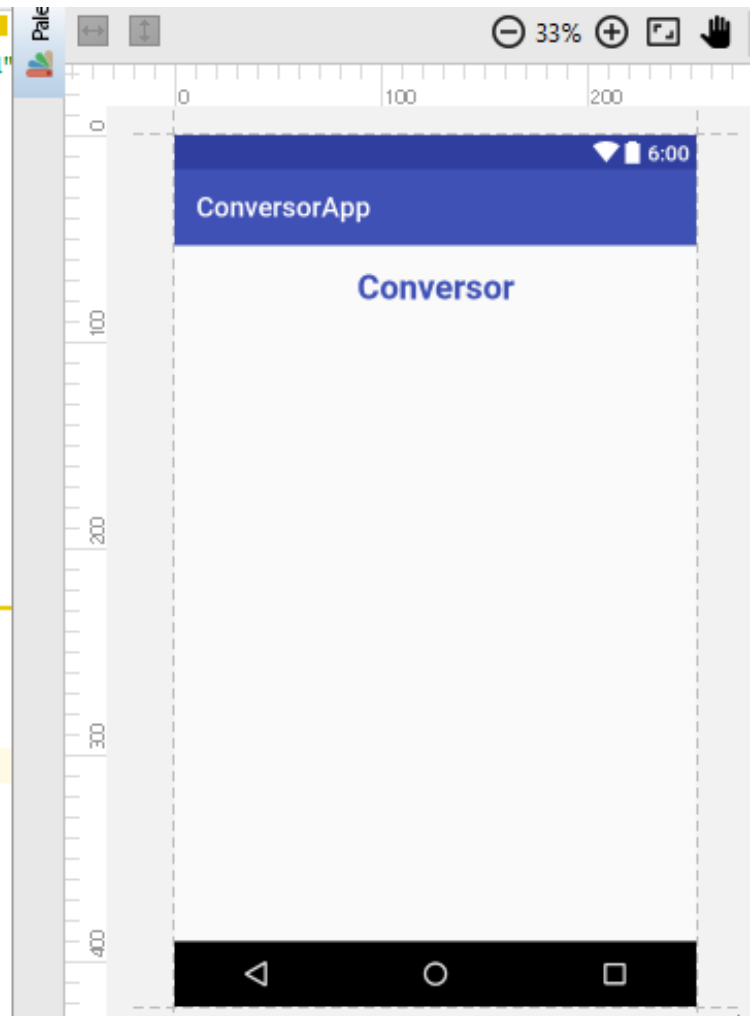
```

TextView

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.edwinacubillos.conversorapp.MainActivity">

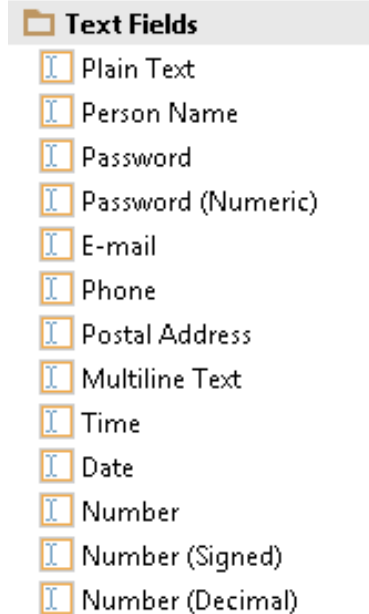
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textStyle="bold"
        android:text="Conversor"
        android:textSize="25sp"
        android:textColor="@color/colorPrimary"
        android:gravity="center"/>

</RelativeLayout>
```



EditText

- ❖ Permite al usuario entrar texto a la aplicación.
- ❖ Puede ser de una línea o multilínea.
- ❖ Permite otras acciones como copiar, pegar, cortar y autocompletar
- ❖ Se pueden usar diferentes tipos de EditText para validar entrada de texto



- ❖ Al seleccionar un tipo de EditText automáticamente Android cargará un teclado para tal fin



 Plain Text

 Number



 E-mail



- ❖ La diferencia en el script xml es una sola línea, que le da la propiedad requerida

Text Fields

- Plain Text
- Person Name
- Password
- Password (Numeric)
- E-mail
- Phone
- Postal Address
- Multiline Text
- Time
- Date
- Number
- Number (Signed)
- Number (Decimal)

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:ems="10" >

<EditText
    android:id="@+id/editText2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/editText1"
    android:ems="10"
    android:inputType="textPersonName" />

<EditText
    android:id="@+id/editText3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/editText2"
    android:ems="10"
    android:inputType="textPassword" />

<EditText
    android:id="@+id/editText4"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentLeft="true"
    android:layout_below="@+id/editText3"
    android:ems="10"
    android:inputType="numberPassword" />
```



Clase View

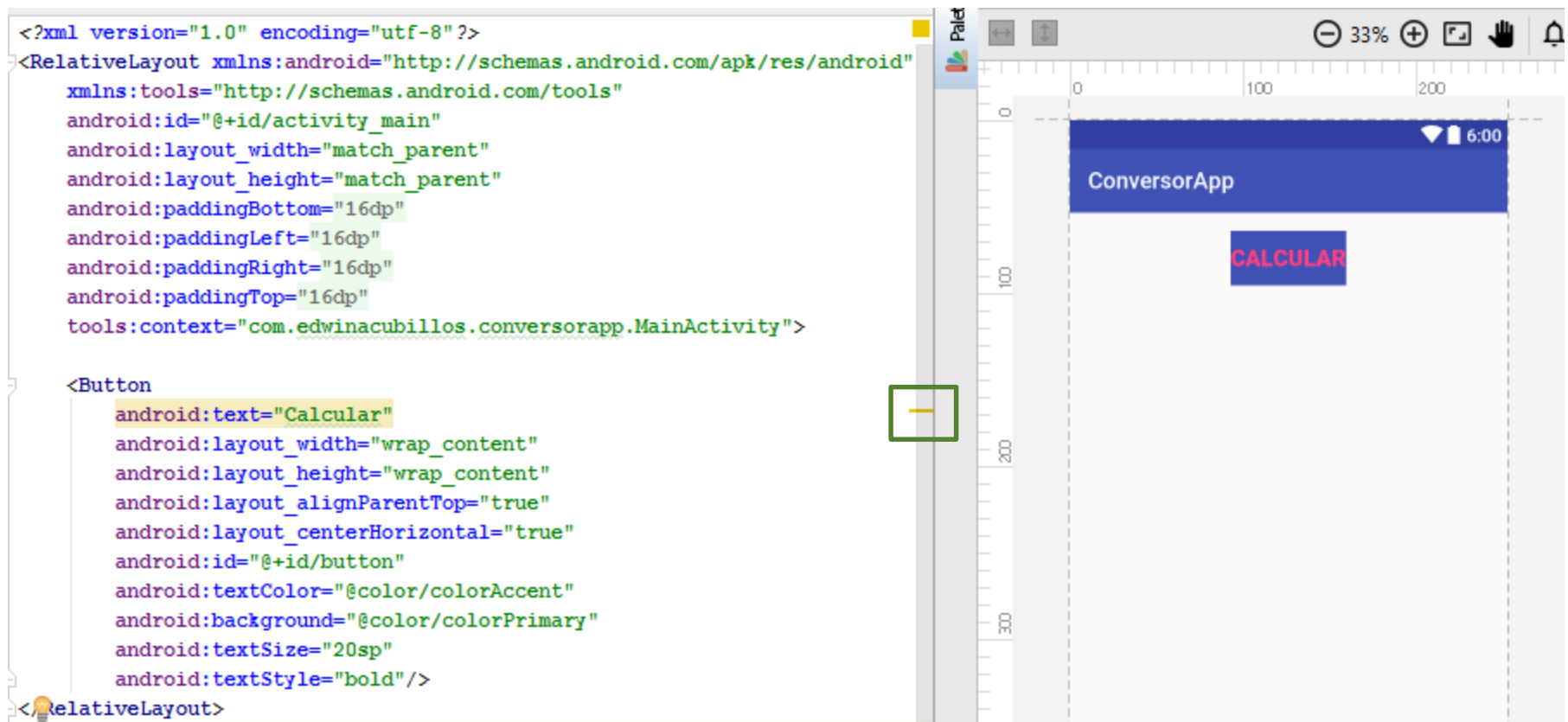
Creación de una vista

Layout

TextView

Button

❖ Button



The image shows the Android Studio interface. On the left, the XML code for the layout is displayed. The code defines a `RelativeLayout` containing a `Button` with the text "Calcular". The button's text color is set to `@color/colorAccent`, the background to `@color/colorPrimary`, and the text size to 20sp in bold. On the right, the visual preview shows a blue header bar with the text "ConversorApp", a status bar at the top with a Wi-Fi icon and the time 6:00, and a blue button with the text "CALCULAR" in red. A green box highlights the button in the preview, and a corresponding green box highlights the `<Button>` tag in the XML code.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="16dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    tools:context="com.edwinacubillos.conversorapp.MainActivity">

    <Button
        android:text="Calcular"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_centerHorizontal="true"
        android:id="@+id/button"
        android:textColor="@color/colorAccent"
        android:background="@color/colorPrimary"
        android:textSize="20sp"
        android:textStyle="bold"/>

</RelativeLayout>
```

Dos formas de implementar su funcionalidad:

1. Se debe definir el objeto `View.OnClickListener()` y se asocia al botón mediante `setOnClickListener()`

```
Button button = (Button) findViewById(R.id.button_send);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // Do something in response to button click
    }
});
```

2. Se asigna un método al botón en el XML del Layout usando android:onClick

```
<?xml version="1.0" encoding="utf-8"?>
<Button xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```

Cuando se hace click se llama al método sendMessage, el cual debe ser público y recibir un view como único parámetro

```
/** Called when the user touches the button */
public void sendMessage(View view) {
    // Do something in response to button click
}
```