



UNIVERSIDAD  
DE ANTIOQUIA

1 8 0 3

## GUI - Clase View

Ing. Edwin Andrés Cubillos Vega Msc.



**Clase View**

**Creación de una vista**

**Layout**

**TextView y EditText**

**Button**

**Clase View**

Creación de una vista

Layout

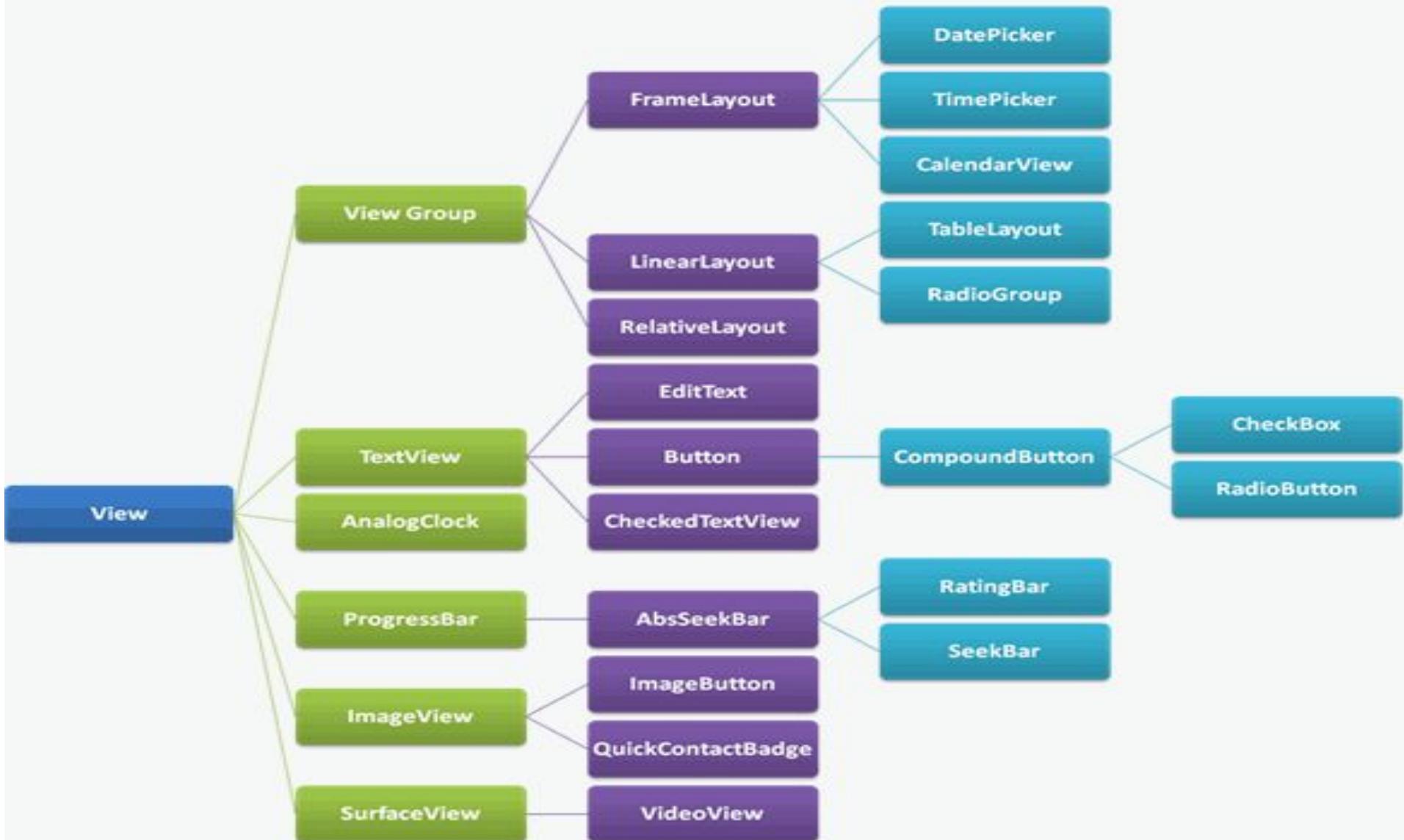
TextView y EditText

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 al 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 6. Referencia Clase View”

Clase View

**Creación de una vista**

Layout

TextView y EditText

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);  
    }  
}
```

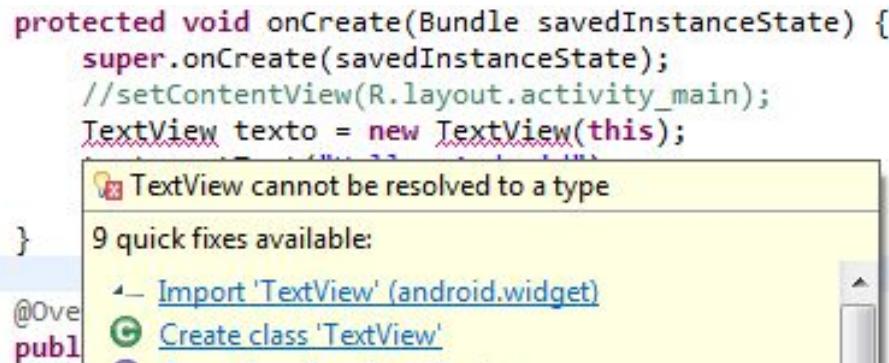
Crea la Interfaz  
Cómo??

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 + 0, 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.

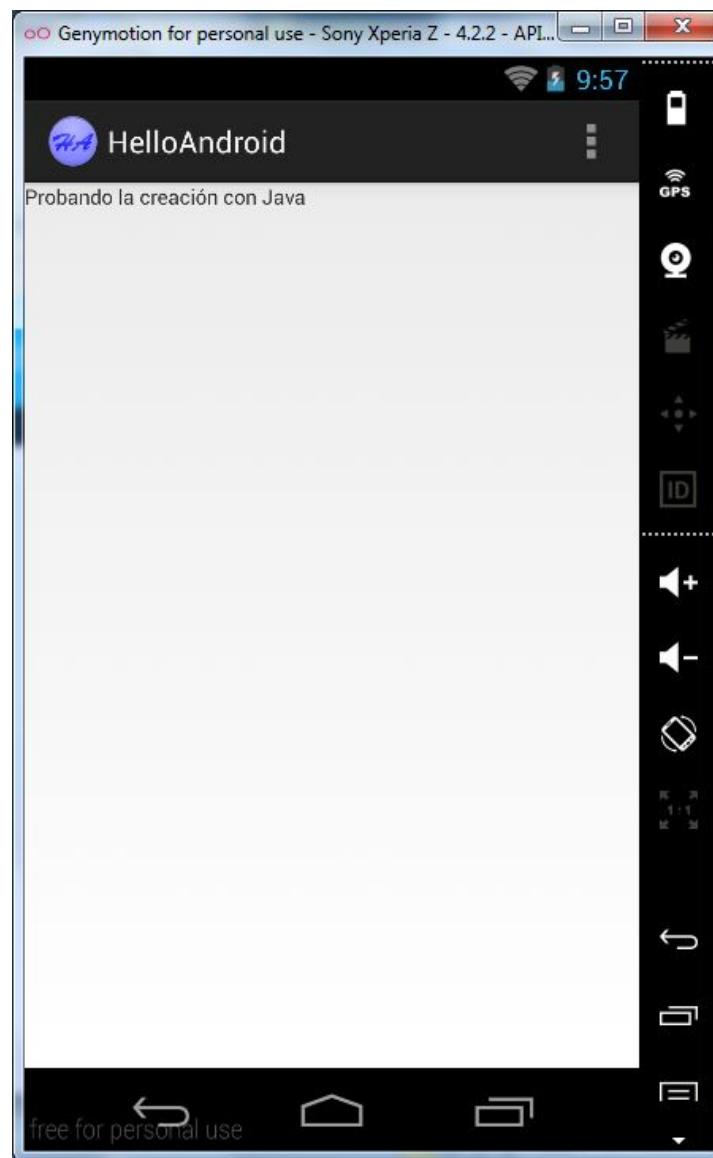
# Creación de una vista

## ❖ Ahora analicemos el código

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    //setContentView(R.layout.activity_main);  
    TextView texto = new TextView(this); 1  
    texto.setText("Probando la creación con Java"); 2  
    setContentView(texto); 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>
```

# Creación de una vista

```
<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 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.

# Creación de una vista

```
<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>
```

- ❖ @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>
```

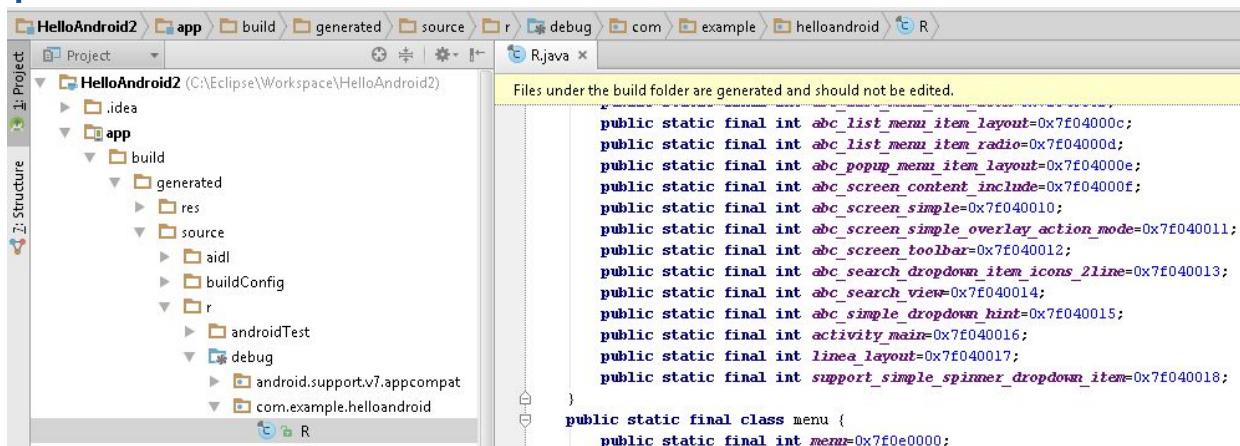
- ❖ En el archivo MainActivity.java eliminar las líneas del item anterior y descomentar

```
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*

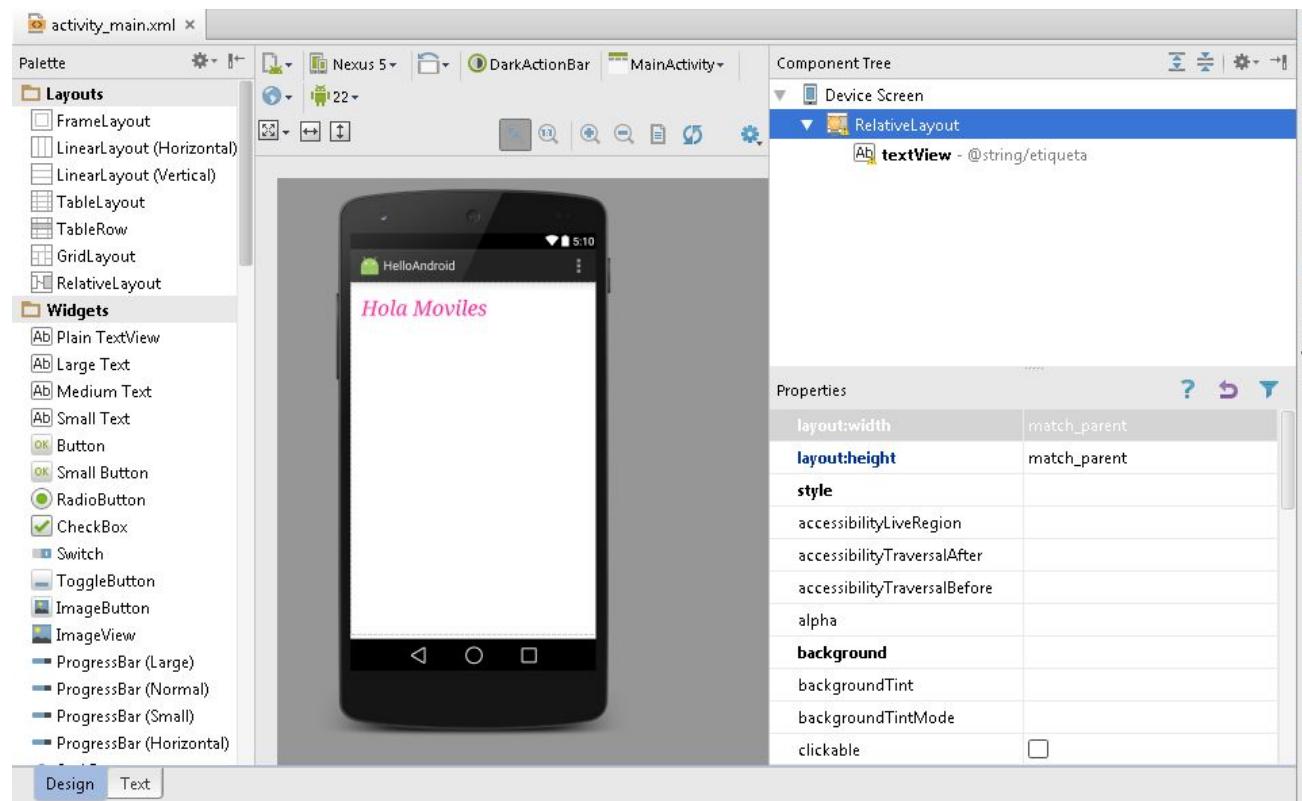


- ❖ Los identificadores de la clase R son números que informan al gestor de recursos, que datos ha de cargar.
- ❖ Por lo tanto no se trata de verdaderos objetos, estos serán creados en tiempo de ejecución solo cuando sea necesario usarlos.

# 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**

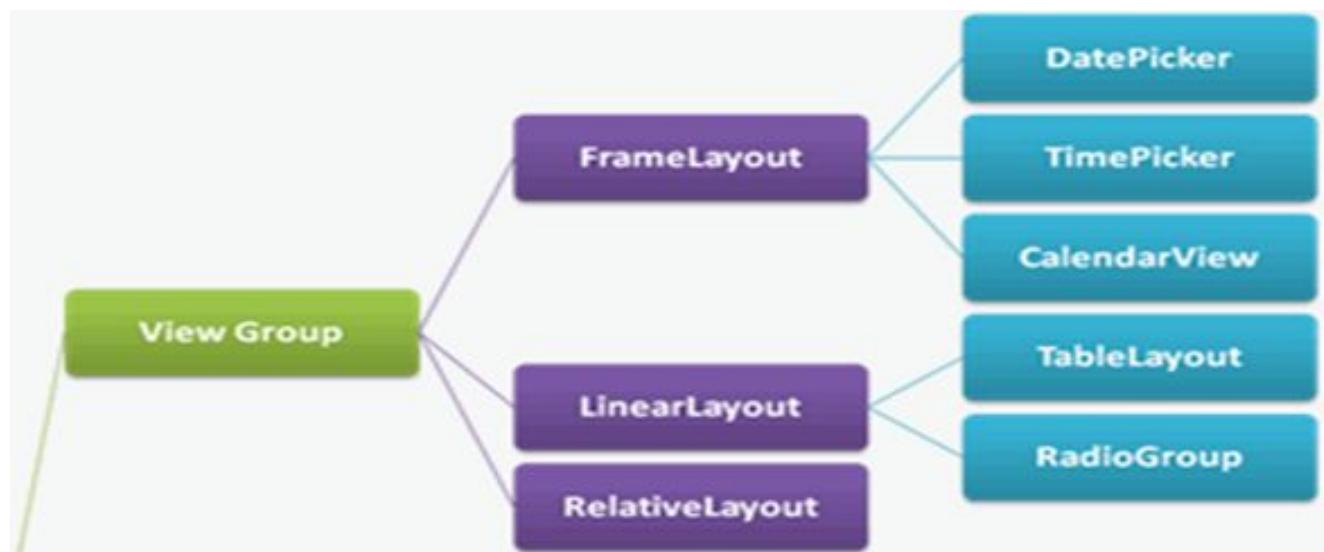
**Layout**

**TextView y EditText**

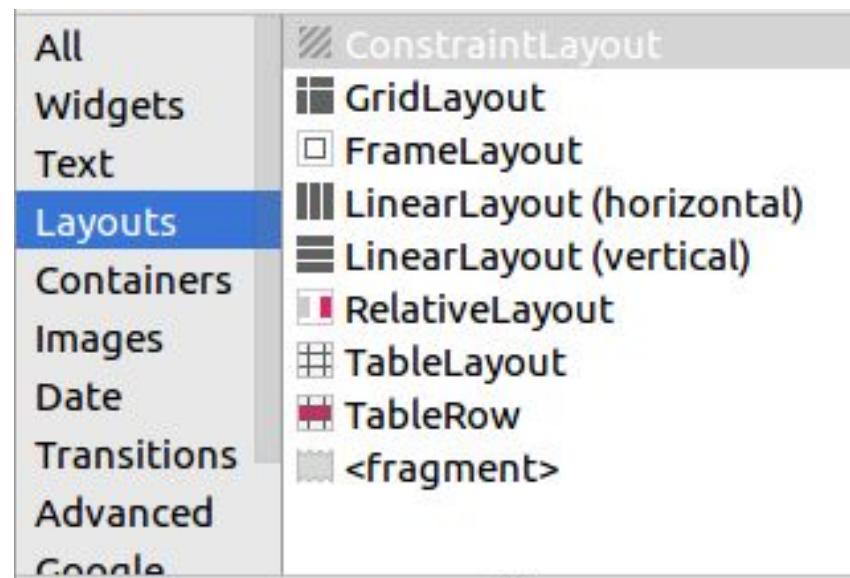
**Button**

# Layout

- ❖ 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



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



Hay varios tipos de layouts:

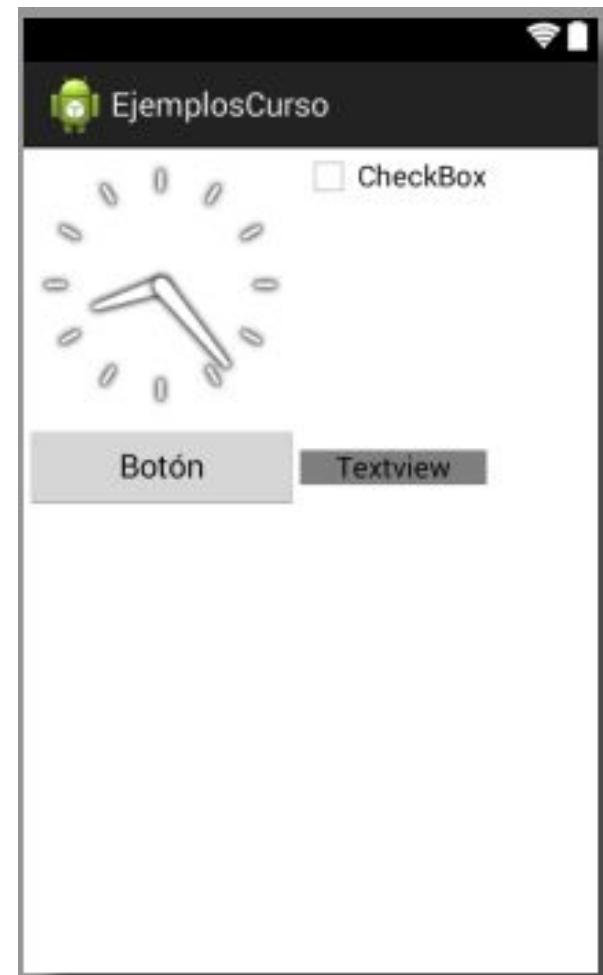
- ❖ **LinearLayout:** Dispone los elementos en una fila o columna

```
<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>
```



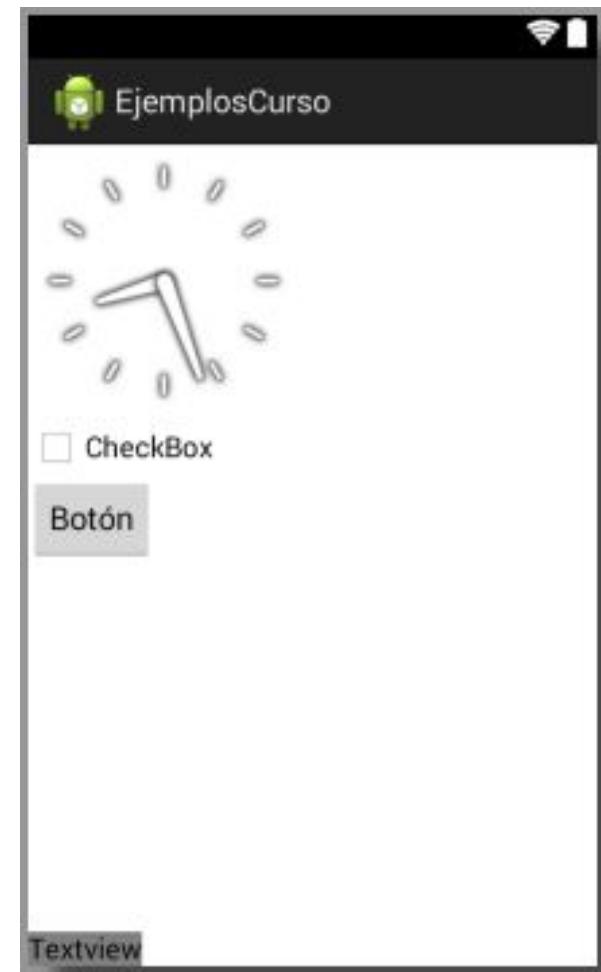
## ❖ **TableLayout:** Distribuye los elementos de forma tabular

```
<?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>
```



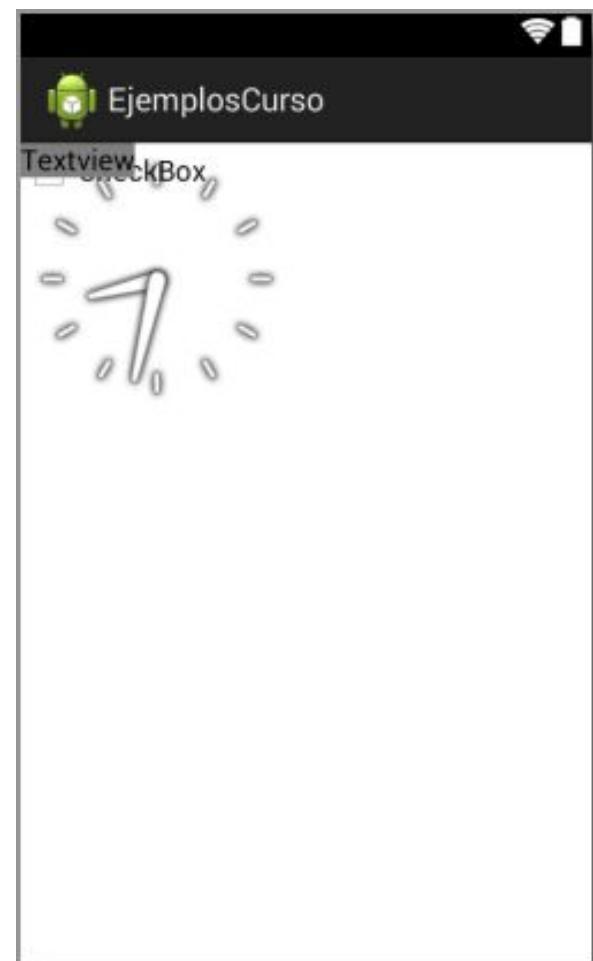
## ❖ **RelativeLayout:** Dispone los elementos en relación a otro o al padre.

```
<?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>
```



- ❖ **FrameLayout:** Permite el cambio dinámico de los elementos que contiene.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <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"
        android:visibility="invisible" />
    <Textview
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Texto cualquiera" />
</FrameLayout>
```



## 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.

**Clase View**

**Creación de una vista**

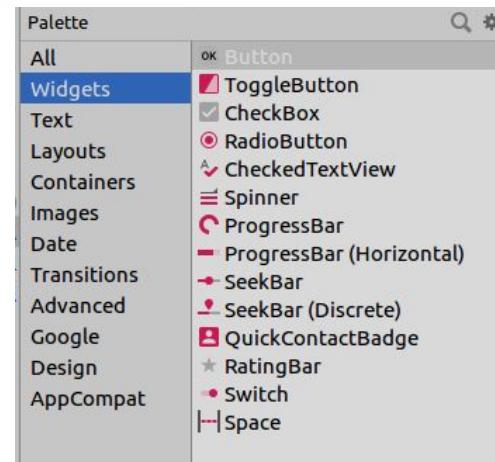
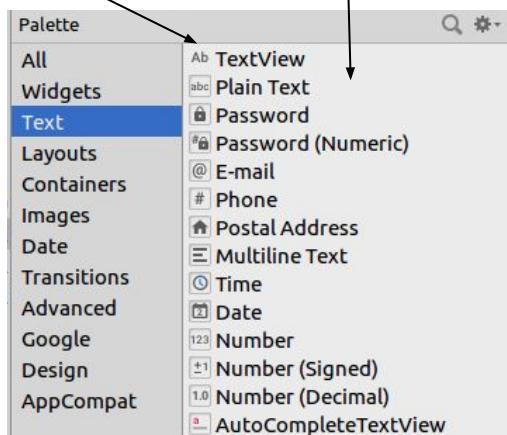
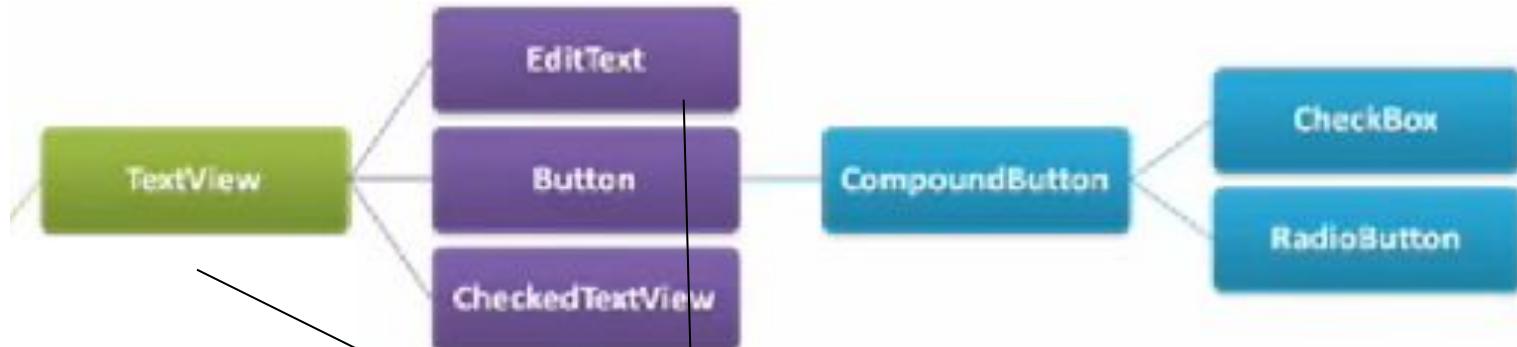
**Layout**

**TextView y EditText**

**Button**

# TextView

- ❖ 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 (Revisar DocumentoReferencia Clase View)
- ❖ 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
  - modifigan 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)

# TextView

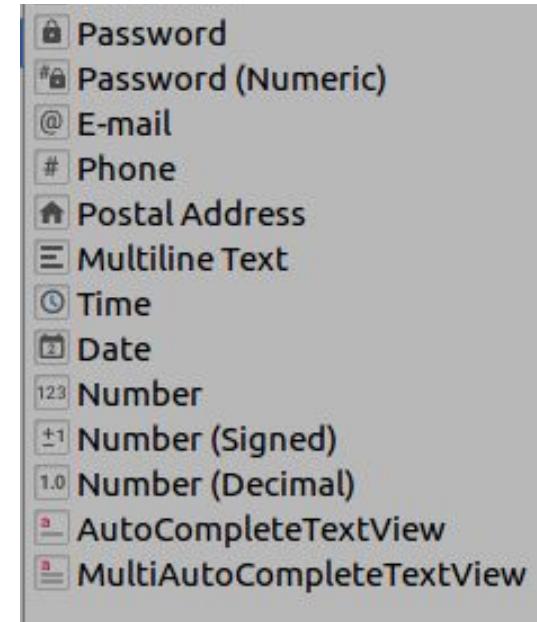


The screenshot shows an Android studio interface with the code editor open to the file `colors.xml`. The left sidebar lists various resource files: `anifests`, `va`, `s`, `drawable`, `layout`, `mipmap`, `values`, `colors.xml` (selected), `strings.xml (2)`, and `styles.xml`. The right pane displays the XML code for `colors.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
    <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">#8b008b</item>
    <item name="darkmagenta" type="color">#9400d3</item>
    <item name="darkviolet" type="color">#006400</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>
```

## 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



# EditText

All  
Widgets  
**Text**  
Layouts  
Containers  
Images  
Date  
Transitions  
Advanced  
Google  
Design  
AppCompat

Ab TextView  
abc Plain Text  
🔒 Password  
🔓 Password (Numeric)  
@ E-mail  
# Phone  
🏡 Postal Address  
☰ Multiline Text  
⌚ Time  
📅 Date  
123 Number  
± Number (Signed)  
1.0 Number (Decimal)  
🔤 AutoCompleteTextView  
🔤 MultiAutoCompleteTextView

```
<EditText  
    android:id="@+id/editText2"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:ems="10"  
    android:inputType="textPersonName"  
    android:text="Name"  
    tools:layout_editor_absoluteX="37dp"  
    tools:layout_editor_absoluteY="40dp" />  
  
<EditText  
    android:id="@+id/editText3"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:ems="10"  
    android:inputType="textPassword"  
    tools:layout_editor_absoluteX="37dp"  
    tools:layout_editor_absoluteY="104dp" />  
  
<EditText  
    android:id="@+id/editText4"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:ems="10"  
    android:inputType="textEmailAddress"  
    tools:layout_editor_absoluteX="37dp"  
    tools:layout_editor_absoluteY="168dp" />  
  
<EditText  
    android:id="@+id/editText5"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:ems="10"  
    android:inputType="phone"  
    tools:layout_editor_absoluteX="37dp"  
    tools:layout_editor_absoluteY="234dp" />  
  
<EditText  
    android:id="@+id/editText6"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:ems="10"  
    android:inputType="textMultiLine"  
    tools:layout_editor_absoluteX="37dp"  
    tools:layout_editor_absoluteY="298dp" />
```

# EditText

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



I Plain Text

I Number



I E-mail



**Clase View**

**Creación de una vista**

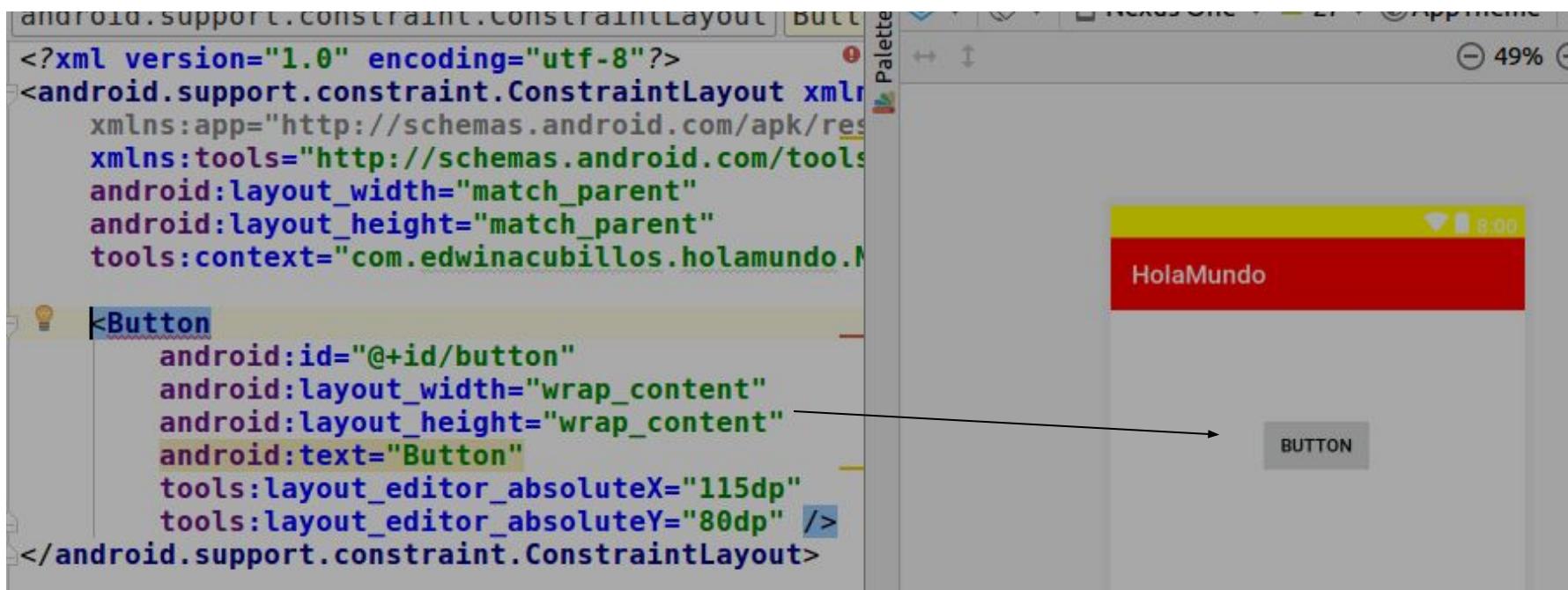
**Layout**

**TextView y EditText**

**Button**

# Button

## ❖ Button



The screenshot shows the Android Studio interface. On the left, the XML layout file is displayed:

```
    android.support.constraint.ConstraintLayout Button
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.edwinacubillos.holamundo.MainActivity">
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button"
        tools:layout_editor_absoluteX="115dp"
        tools:layout_editor_absoluteY="80dp" />
</android.support.constraint.ConstraintLayout>
```

The preview window on the right shows a red button with the text "HolaMundo". A callout arrow points from the word "BUTTON" in the XML code to the red button in the preview.

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
}
```

# RadioButton

ATTENDING?

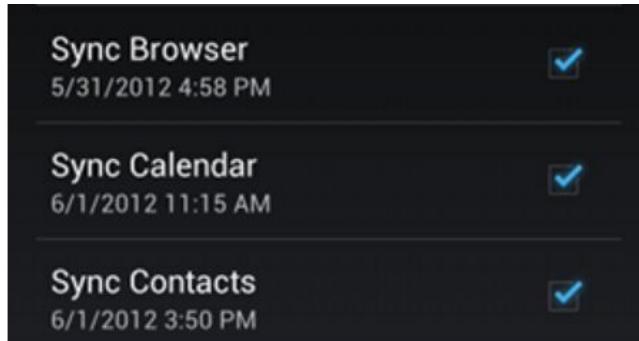
Yes     Maybe     No

```
<?xml version="1.0" encoding="utf-8"?>
<RadioGroup xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <RadioButton android:id="@+id/radio_pirates"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/pirates"
        android:onClick="onRadioButtonClicked"/>
    <RadioButton android:id="@+id/radio_ninjas"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/ninjas"
        android:onClick="onRadioButtonClicked"/>
</RadioGroup>
```

```
public void onRadioButtonClicked(View view) {
    // Is the button now checked?
    boolean checked = ((RadioButton) view).isChecked();

    // Check which radio button was clicked
    switch(view.getId()) {
        case R.id.radio_pirates:
            if (checked)
                // Pirates are the best
                break;
        case R.id.radio_ninjas:
            if (checked)
                // Ninjas rule
            break;
    }
}
```

# CheckBox



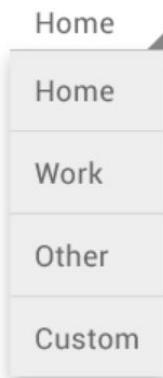
```
public void onCheckboxClicked(View view) {
    // Is the view now checked?
    boolean checked = ((CheckBox) view).isChecked();

    // Check which checkbox was clicked
    switch(view.getId()) {
        case R.id.checkbox_meat:
            if (checked)
                // Put some meat on the sandwich
            else
                // Remove the meat
            break;
        case R.id.checkbox_cheese:
            if (checked)
                // Cheese me
            else
                // I'm lactose intolerant
            break;
        // TODO: Veggie sandwich
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <checkBox android:id="@+id/checkbox_meat"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/meat"
        android:onClick="onCheckboxClicked"/>
    <checkBox android:id="@+id/checkbox_cheese"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/cheese"
        android:onClick="onCheckboxClicked"/>
</LinearLayout>
```

# Spinner

jay@gmail.com



```
<Spinner  
    android:id="@+id/planets_spinner"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content" />
```

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <string-array name="planets_array">  
        <item>Mercury</item>  
        <item>Venus</item>  
        <item>Earth</item>  
        <item>Mars</item>  
        <item>Jupiter</item>  
        <item>Saturn</item>  
        <item>Uranus</item>  
        <item>Neptune</item>  
    </string-array>  
</resources>
```

```
Spinner spinner = (Spinner) findViewById(R.id.spinner);  
// Create an ArrayAdapter using the string array and a default spinner layout  
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,  
    R.array.planets_array, android.R.layout.simple_spinner_item);  
// Specify the layout to use when the list of choices appears  
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);  
// Apply the adapter to the spinner  
spinner.setAdapter(adapter);
```