

COMPONENTES GRÁFICOS EN ANDROID.

Por lo regular los lenguajes de programación brindan alternativas de interfaces graficas de usuario para facilitar aún más el trabajo con el sistema que estamos programando, cuando se trabaja con dispositivos móviles este concepto juega un papel de suma importancia pues va a facilitar la interacción del usuario no solo con nuestra aplicación sino con el dispositivo en general.

En esta guía veremos algunos de los componentes gráficos básicos que pone a nuestra disposición el sistema operativo Android.

Pero antes de entrar en materia, es importante referenciar algunas propiedades comunes para los diferentes componentes, estos tienen relación con el tamaño, uso de texto y la forma de identificarlos.

android:id.

Propiedad muy importante porque permite definir un identificador al control.

Seguramente esta vista se usara para capturar información del usuario por esa razón debemos usar un nombre diferenciador y claro.

La forma correcta de generar un identificador es la siguiente:

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

El símbolo @+id/ quiere decir que se está generando un nuevo identificador, esto provocará que la clase R del paquete gen del proyecto cree una referencia de este objeto para que luego pueda ser manipulado por código.

Recuerde que en Android existirá una clase auto gestionada llamada R que tiene una referencia de todos los recursos del proyecto, es decir, imágenes, archivos y también de todas las vistas de todas las pantallas a las que se les haya generado un identificador.

En el ejemplo se creó el identificador myText para el control.

android:layout_width.

Define el ancho asociado a la vista, es una propiedad obligatoria y su valor se puede definir en valores absolutos o indicando alguno de los siguientes dos valores:

- *wrap_content* (Ocupa el espacio de acuerdo a su contenido)
- *match_parent* (Ocupa todo el espacio disponible)

android:layout_height.

Define el alto asociado a la vista, es una propiedad obligatoria y su valor se puede definir en valores absolutos o indicando alguno de los siguientes dos valores:

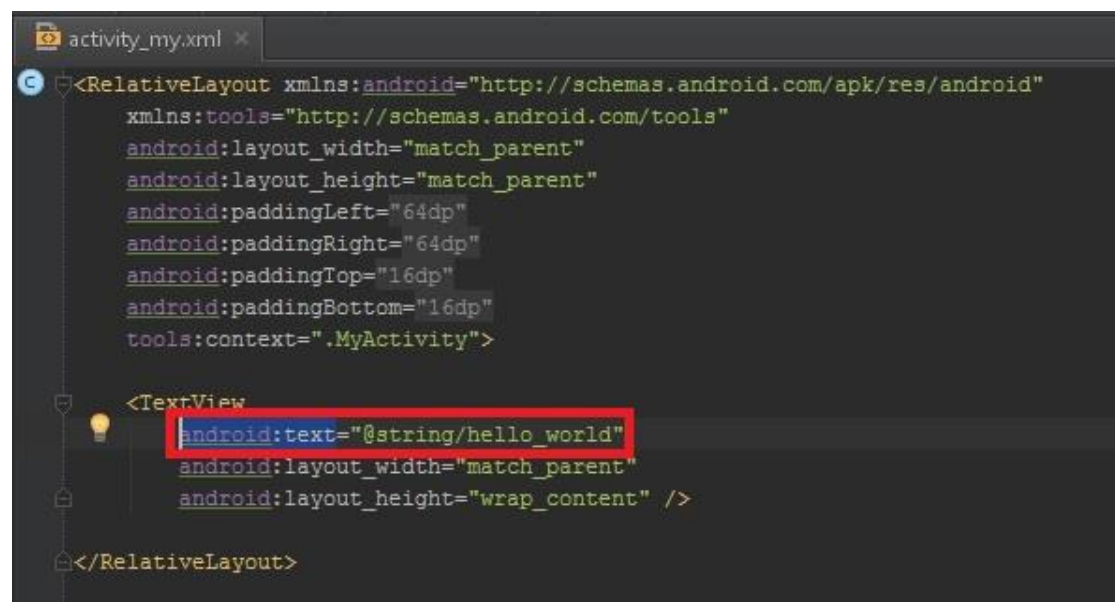
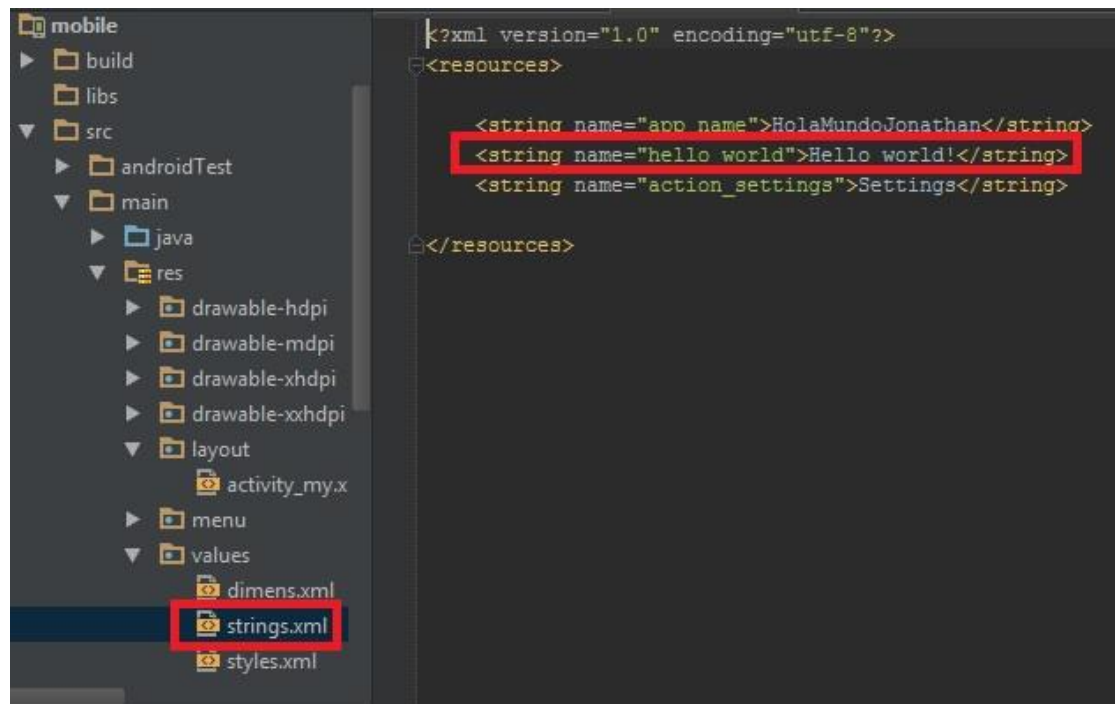
- *wrap_content* (Ocupa el espacio de acuerdo a su contenido)

- `match_parent` (Ocupa todo el espacio disponible)

android:text.

Define el contenido textual asociado a la vista. Aunque su valor puede ser asignado de forma directa, se recomienda fuertemente usar el archivo de strings disponible en la carpeta de recursos.

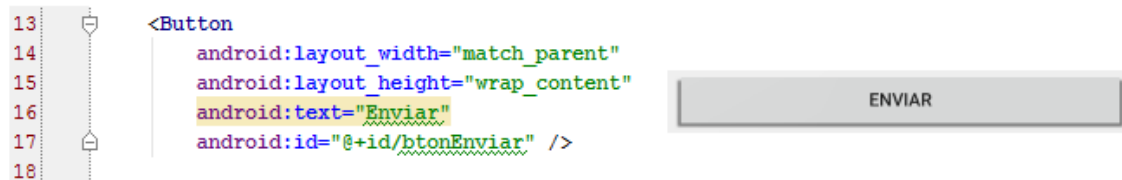
Esta propiedad es comun en vistas que en la interface del dispositivo despliegan texto



Componentes básicos.

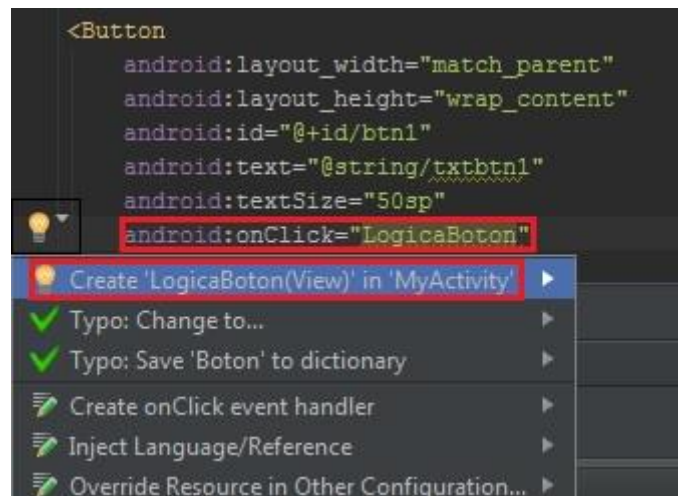
Button.

Estos corresponden a los botones básicos que conocemos, tan solo presenta un botón con su respectivo texto internamente.



El botón cumpliría la función básica de esperar un evento cuando sea presionado.

El evento más importante a implementar es el `onClick()`. Es importante resaltar que el IDE de Android Studio ofrece ventajas muy importantes a la hora de autogenerar código para el control de eventos en las vistas, ya que basta con solo asociarlo en el xml y el genera el fragmento de la función Java en la actividad donde relacionaremos la lógica a ejecutar en el evento.

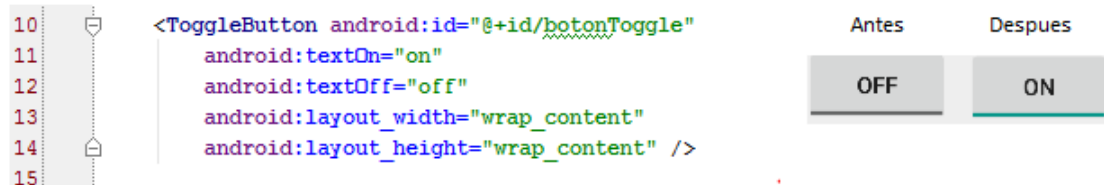


Al indicar que se va a crear el método, en el archivo de lógica se tendrá el siguiente código que presenta un mensaje emergente:

```
public void LogicaBoton(View view) {
    Toast.makeText(this, getResources().getString(R.string.msgbtn1), Toast.LENGTH_SHORT).show();
}
```

ToggleButton.

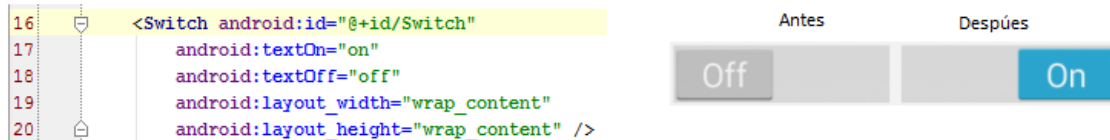
El tipo ToggleButton corresponde a un tipo de botón que puede mantenerse en 2 posibles estados, presionado o no presionado, para este no solo se define una única propiedad text sino 2 dependiendo del estado que posea en el momento, usando para eso las propiedades android:textOn y android:textOff.



Al igual que los botones tradicionales el ToggleButton se puede programar para que responda al evento click pero muy seguramente solo necesitara saber en qué estado se encuentra, esto lo puede lograr consultando el método isChecked() que devuelve el valor de true en el caso de que este pulsado y false en caso contrario.

Switch.

Este control puede simular un interruptor muy parecido al funcionamiento del ToggleButton pero con un aspecto visual distinto.



Hay que tener en cuenta que dependiendo de la versión de Android que estemos usando, el aspecto visual de nuestros componentes puede cambiar, en este caso el control Switch para la versión 5.0 de Android se vería así:



ImageButton.

Este tipo de botón permite vincular una imagen al mismo para ser mostrada al usuario en vez de un texto como es común, para hacerlo podemos tomar la imagen de las carpetas /res/drawable y la cargarla usando la propiedad android:src, en este caso vamos a usar una imagen directamente de los recursos de Android.

```
22 <ImageButton android:id="@+id/botonImagen"  
23     android:layout_width="wrap_content"  
24     android:layout_height="wrap_content"  
25     android:src="@drawable/abc_ic_menu_cut_mtrl_alpha" /
```

TextView.

Estos corresponden a las etiquetas de texto básicas que se muestran a los usuarios, estos en otros lenguajes son conocidos como labels y permiten mostrar un texto asociado a la propiedad android:text, estas etiquetas tienen otras propiedades que permiten manipular su tamaño, color de fondo, color de la fuente y tipo de letra, estilo entre otras

```
16 <TextView android:id="@+id/etiqueta"  
17     android:layout_width="match_parent"  
18     android:layout_height="wrap_content"  
19     android:text="Texto a Mostrar"/>  
20
```

EditText

La vista EditText de Android es equivalente al Textbox de ASP y C# o al TextField de JAVA, es un control que permite la captura de texto ingresado por el usuario en tiempo de ejecución.

Propiedades importantes:

android:hint : Esta propiedad permite desplegar un texto sobre el control a manera de ayuda mientras el usuario aun no ingrese información. (Preferible sobre la propiedad text)

android:inputType: Determina el tipo de entrada que se admitirá para la caja de texto, esto permite cierto tipo de validación de los datos y además le indica al sistema operativo que tipo de teclado (virtual) debe utilizarse.

Entre las opciones que podemos asociar para esta propiedad son las siguientes:

- Text
- Number
- Texturi
- Textpassword
- Phone
- Date

```
<EditText
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/edit1"
    android:hint="@string/hint_edit1"
    android:inputType="text"
    android:textSize="50sp"
/>
```



CheckBox

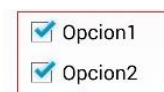
El control checkbox se utiliza para marcar o desmarcar opciones en una aplicación. La forma de definirlo en la interface y las propiedades disponibles para su manipulación por código son análogas a las comentadas en el control ToogleButton.

Este control hereda del control TextView por lo que todas las opciones de formato ya comentadas para este control también son válidas para el checkBox.

En el código de la aplicación se puede consultar si este control se encuentra o no seleccionado por medio del método `isChecked()` que devuelve el valor de *true* si ha sido seleccionado o *false* en caso contrario, también se puede usar el método `setChecked(valor)` para establecer un estado concreto para el control, donde valor *true* seria para seleccionado y *false* para no seleccionado.

El evento que normalmente se programa de este control y que se ejecuta cada vez que cambia de estado (seleccionado / No seleccionado) es el `onCheckedChangeListener`.

```
<CheckBox
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/chexkbox1"
    android:text="@string/chbox1"
/>
<CheckBox
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/chexkbox2"
    android:text="@string/chbox2"
/>
```



RadioButton

Al igual que el control CheckBox, el RadioButton se usa para marcar o desmarcar una determinada opción, la diferencia radica en el tipo de selección que se desea realizar. Normalmente los RadioButton se usan en grupo para definir un conjunto de opciones de las cuales solo podrá seleccionar una, es decir, cada opción es mutuamente excluyente por lo que

al seleccionar una opción se desmarcara automáticamente la que previamente estaba seleccionada. Estos grupos se definen mediante el elemento `RadioGroup` que a su vez contendrá todos los elementos `RadioButton` que representan las opciones.

Los `RadioGroups` pueden definir la propiedad `android:orientation` con los valores “vertical” u “Horizontal” para determinar la forma en que se ordenaran los `RadioButton` que contiene.

Lo `RadioButton` por su parte además del ancho y alto deberían definir la propiedad `android:text` para asociar el texto de la opción que se representa y `android:id` para asociar un código al control para que pueda ser manipulado desde la lógica de la aplicación.

Para manipular el control desde la lógica de la aplicación se puede hacer uso de diferentes métodos entre los cuales encontramos el `check(id)` que sirve para fijar al `radiobutton` identificado con el `id` que se pasa como parámetro como seleccionado, el método `clearCheck()` sirve para desmarcar todos los `radiobutton` del grupo y el método `getCheckedRadioButtonId()` sirve para obtener el `id` del `radiobutton` que dentro del grupo está seleccionado o el valor de -1 si no hay ningún elemento seleccionado.

El evento más importante de este control también es el `onCheckedChanged`, que se lanza cuando un elemento del grupo es seleccionado (recuerde que la selección de un elemento del grupo indica la des selección de otro).

```
<RadioGroup
    android:layout_marginTop="20sp"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/radio1"
        android:text="@string/rad1"/>
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/radio2"
        android:text="@string/rad2"/>
</RadioGroup>
```



Ejercicio

Crear una aplicación móvil que permita simular la creación de contactos personales incluyendo en su elaboración los siguientes views básicos:

- TextView
- EditText
- Button
- ToogleButton
- CheckBox
- RadioButton

La interface debe ser similar a la siguiente imagen.

PracticaControlesBasicos

Nuevo Contacto

Nombre y Apellido

Telefono Celular

Correo Electronico

Genero

☐ Masculino

☐ Femenino

Seleccione sus temas de interes

☐ Redes Sociales

☐ Video Juegos

☐ Programación

☐ Bases e Datos

Ver resumen antes de guardar?

No

Guardar

Implementar la lógica necesaria para que al presionar el botón guardar imprima un mensaje por medio de un Toast donde se proyecte un resumen de todo lo descrito en la creación del nuevo contacto si es que el usuario habilita la opción de ver resumen (ToogleButton) o el mensaje de usuario creado.

PracticaControlesBasicos

Nuevo Contacto

Jonathan Guerrero

3123334444

jonathan@algo.com

Genero

☒ Masculino

☐ Femenino

Seleccione sus temas de interes

☒ Redes Sociales

☐ Video Juegos

☐ Programación

☒ Bases e Datos

Ver resumen antes de guardar?

Si

Guardar

PracticaControlesBasicos

Nuevo Contacto

Jonathan Guerrero

3123334444

jonathan@algo.com

Genero

☒ Masculino

☐ Femenino

Seleccione sus temas de interes

☒ Redes Sociales

☐ Video Juegos

☐ Programación


☒ Bases e Datos

Ver resumen antes de guardar?

Si

Guardar

Nombre: Jonathan Guerrero
Telefono Celular: 3123334444
Correo Electronico: jonathan@algo.com
Genero: Masculino
Temas de Interes:
Redes Sociales
Bases de Datos

 PracticaControlesBasicos

Nuevo Contacto

Genero

☒ Masculino

☐ Femenino

Seleccione sus temas de interes


☒ Redes Sociales

☐ Video Juegos

☐ Programación

☒ Bases e Datos

Ver resumen antes de guardar?

 PracticaControlesBasicos

Nuevo Contacto

Genero

☒ Masculino

☐ Femenino

Seleccione sus temas de interes

☒ Redes Sociales

☐ Video Juegos

☐ Programación

☒ Bases e Datos

Ver resumen antes de guardar?