TextView

Permite colocar un texto, seteando color, tamaño, posición, tipo de letra,etc.

Definición en XML:

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="TextView" />
```

Propiedades de los TextView

- **android:typeface** que sirve para indicar el tipo de letra con que se mostrará el texto de la etiqueta (ejemplo: normal, sans, serif, monospace, etc.).
- **android:textStyle** para indicar si el tipo de letra seleccionado debe mostrarse en un formato en negritas (bold), cursiva (italic) o ambas (bold|italic).
- **android:textColor** para indicar el color que tendrá el texto mostrado en la etiqueta, este debe estar expresado en formato RGB Hexa (ejemplo: #A4C639 para el verde característico de Android).
- **android:textSize** Configura el tamaño del texto. La definición del valor para este atributo se conforma de un número de punto flotante seguido de una unidad de medida, que pueden ser: sp (scaled pixels), px (pixels), dp (density-independent pixels), in (inches), mm (millimeters).
- **android:shadowColor:** el color de la sombra del texto en formato RGB.
- **android:shadowRadius:** especifica el radio de la sombra con un número de punto flotante.
- **android:shadowDx y android:shadowDy:** para indicar el desplazamiento de la sombra en los ejes X y Y respectivamente a través de un número de punto flotante.
- **android:autoLink :** Si esta setado, busca las URLs en el texto mostrado y las convierte en links. Los valores posibles son : *All,Phone,Email,Map,Web,None.*

Utilizando TextView desde código java.

En la Activity obtenemos el objeto TextView mediante el método "findViewById()" el cual recibe como parámetro el id que se genera automáticamente en la clase R y que corresponde al id del objeto TextView definido en el xml:

```
TextView tv = (TextView) findViewById(R.id.textView1);

tv.setText("Texto desde Codigo");
```

Utilizando fuentes personalizadas

Copiaremos el archivo TTF dentro del directorio assets de nuestro proyecto Android (en caso de que el directorio no exista, lo creamos).

```
TextView tv = (TextView) findViewById(R.id.text1);
Typeface font = Typeface.createFromAsset(getAssets(), "fuente.ttf");
tv.setTypeface(font);
```

La clase Typeface tiene un método estático llamado createFromAsset(), que toma un objeto de tipo AssetManager como primer parámetro y el nombre o ruta del archivo TTF como segundo parámetro. En este ejemplo estamos utilizando el asset manager por default que retorna el método getAssets() y escribimos el nombre de nuestro archivo tal cual se encuentra físicamente en la raíz del directorio assets del proyecto Android. Finalmente, con nuestro objeto de tipo Typeface creado lo pasamos como parámetro cuando llamamos al método setTypeface() del objeto TextView.

Button (Botón)

Muestra un botón al que se le pueden setear, texto, color, tamaño, imagen, etc.

Definición en XML:

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:drawableTop="@drawable/ic_launcher"
    android:text="Button" />
```

Propiedades del boton:

- **android:drawableTop** se setea la imagen que tendrá el botón (arriba del texto)
- **android:drawableLeft** se setea la imagen que tendrá el botón (a la izquierda)
- **android:drawableRight** se setea la imagen que tendrá el botón (a la derecha)
- **android:drawableBotton** se setea la imagen que tendrá el botón (abajo del texto)
- **android:drawablePadding** se setea una cantidad de relleno entre la imagen y el texto.

ToggleButton (Boton con estado ON-Off)

Este botón, tiene una indicación luminosa para cuando esta activado. y un texto que indica ON/OFF.

Definición en XML:

```
<ToggleButton
    android:id="@+id/toggleButton1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="ToggleButton" />
```

Propiedades del botón:

- **android:disabledAlpha** The alpha to apply to the indicator when disabled.
- **android:textOff** The text for the button when it is not checked.
- **android:textOn** The text for the button when it is checked.

Para setear el estado del botón desde el código Java, se utiliza el método “setChecked()” al cual se le pasa como parámetro el valor True o False.

Para leer el estado del botón desde el código Java, se utiliza el método “isChecked()” al cual devuelve el valor True o False.

CheckBox

Permite al usuario elegir una opción con true-false

Definición en XML:

```
<CheckBox
    android:id="@+id/chk1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/strCheck" />
```

En el código Java, se utiliza el método isChecked() y setChecked() para leer y cambiar el estado del objeto.

El listener que utiliza este objeto es "OnCheckedChangeListener" haciendo Override del método "onCheckedChanged()".

Radio Button y Radio group

Permite al usuario seleccionar una opción entre muchas

Definición en XML:

```
<RadioGroup
      android:id="@+id/radioGroup1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content" >

      <RadioButton
          android:id="@+id/radio0"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:checked="true"
          android:text="RadioButton" />

      <RadioButton
          android:id="@+id/radio1"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="RadioButton" />

      <RadioButton
          android:id="@+id/radio2"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:text="RadioButton" />
</RadioGroup>
```

Atributos del radiobutton:

- **android:checked** indica si esta activado, valores posibles: true o false.

Cuando vamos a trabajar con radio buttons desde los archivos de layout en XML, generalmente tendremos primero que hacer uso del elemento *<RadioGroup>*, que indica que los *<RadioButton>* contenidos en él se encuentran vinculados, permitiendo que el usuario únicamente pueda seleccionar una opción a la vez.
Para que podamos manipular el RadioGroup desde el código Java de la aplicación deberemos asignarle un atributo *android:id* con el valor que queramos. De esta manera podemos hacer uso de los siguientes métodos:

- ***check()*** para que un radio button del grupo aparezca seleccionado. Para especificar cuál de los radio buttons se seleccionará debemos hacer uso del formato *R.id. [idDelRadioButton]*.
- ***ClearCheck()*** para limpiar el grupo de radio buttons y que ninguno aparezca seleccionado.
- ***GetCheckedRadioButtonId()*** para obtener el ID del radio button que actualmente está seleccionado por el usuario, en caso de que ninguno esté seleccionado, este método nos retornará el valor de -1.
- **isChecked()** indica si el radiobutton esta seleccionado.

Ejemplo:

```
RadioGroup rg = (RadioGroup) findViewById(R.id.radioGroup1);
int checkedId = rg.getCheckedRadioButtonId();
    switch(checkedId)
    {
          case R.id.radio0:
                Log.d("main","radio button selecionado : 0");break;
          case R.id.radio1:
                Log.d("main","radio button selecionado : 1");break;
          case R.id.radio2:
                Log.d("main","radio button selecionado : 2");break;
    }
```

El listener para escuchar los cambios en el RadioGroup es llamado "OnCheckedChangeListener" y se setea en el objeto mediante el método "setOnCheckedChangeListener()"

```
rg.setOnCheckedChangeListener(new OnCheckedChangeListener() {
          @Override
          public void onCheckedChanged(RadioGroup group, int checkedId){
                switch(checkedId)
                {
                      case R.id.radio0:
                            Log.d("main","Click radio button 0");break;
                      case R.id.radio1:
                            Log.d("main","Click radio button 1");break;
                      case R.id.radio2:
                            Log.d("main","Click radio button 2");break;
                }
          }
          });
```

EditText

Permite al usuario ingresar texto por pantalla.

Definición en XML:

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >
```

Atributos del EditText:

- **android:inputType** indica el tipo de texto que se puede ingresar en el campo, algunos de los valores posibles son Phone, Number, Text, TextPassword.
- **android:hint** muestra un texto en gris, antes de que el usuario ingrese algo en el campo, es una ayuda para saber que ingresar en dicho campo.
- **android:capitalize** indica que texto se capitaliza, Algunos valores posibles son : sentences,none,characters.
- **android:digits** limita que caracteres pueden ser ingresados, pro ejemplo al poner "01" solo pueden ingresarse los numeros "0" y "1".
- **android:cursorVisible** indica si se muestra el cursor o no, valores posibles : True-false.
- **android:autoText :** Si es True, encuentra y corrige errores en el texto.
- **android:editable :** Si es True, deja editar el texto (False para TextView y True para EditText)

Para obtener el texto escrito en este objeto desde el código Java, se utiliza el método "getText()" el cual devuelve un objeto "Editable", mediante el método "toString()" obtenemos el Texto:

```
public class Pantalla1Activity extends Activity {

    private EditText et;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        et = (EditText) findViewById(R.id.editText1);
        Button bt = (Button) findViewById(R.id.button1);

        bt.setOnClickListener(new OnClickListener() {

                    @Override
                    public void onClick(View v) {

                    Log.d("main","Texto Ingresado:"+et.getText().toString());
                    }
            });
    }
}
```

Los eventos mas comunes para un EditText son:

- *OnCLickListener*
- *OnKeyListener*
- *onFocusChangedListener*

los cuales se setean al objeto mediante los métodos:

- setOnClickListener()
- setOnKeyListener()
- setOnFocusChangedListener()

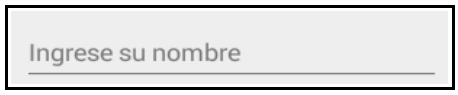
Uso de Hint y mejoras con material design

Utilizando la biblioteca de soporte de Material Design, podremos mejorar entre otras miles de cosas, la apariencia del EditText. Para comenzar deberemos asegurarnos que la biblioteca "design" esté incluida en el archivo buidl.gradle del módulo app:

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:22.2.0'
    compile 'com.android.support:design:22.2.0'
}
```

Nota: Las versiones pueden variar ya que el SDK se va actualizando constantemente.

Si colocamos simplemente un EditText con un atributo de hint, veremos lo siguiente:



Al hacer click y comenzar a escribir, el hint se borrará. Modificaremos el Layout para obtener un hint el cual mediante una animación se colocará sobre el campo para escribir en el momento de hacerle click:

Para lograr este comportamiento, envolveremos al EditText en un TextInputLayout

```
<android.support.design.widget.TextInputLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    app:hintTextAppearance="@style/TextAppearance.AppCompat">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Ingrese su nombre"
        android:id="@+id/editText3" />

</android.support.design.widget.TextInputLayout>
```

ImageView

El control ImageView permite mostrar imágenes en la aplicación. La propiedad más interesante es **android:src**, que permite indicar la imagen a mostrar.

```
<ImageView
android:id="@+id/imgFoto"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:src="@drawable/ic_launcher" />
```

Otra propiedad muy utilizada tiene que ver con la escala que tomará la imagen, dicha propiedad se llama android:scaleType, y puede tomar los siguientes valores:

- centerInside
- center
- fitCenter
- centerCrop
- fitEnd
- fitStart
- fitXY
- matrix

A continuación se muestran dos ejemplos:

centerCrop

fitCenter