Controles básicos (1)

■ Etiqueta de texto: *TextView*

Permite mostrar un texto que se establecerá mediante el atributo *android:text*.

Dispone de multitud de atributos para modificar su apariencia: fuente (fontFamily), tamaño (textSize), color (textColor), etc.

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

Controles básicos (2)

■ Edición de texto: EditText

Permite al usuario introducir un texto por medio del teclado.

Podemos indicar al usuario el tipo de texto esperado mediante el atributo *android:hint*.

```
<EditText
    android:id="@+id/nombre"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:hint="Introduzca su nombre" />
```

Controles básicos (3)

■ Edición de texto: EditText

Podemos facilitar el trabajo al usuario mostrando un teclado específico en función de la información a introducir mediante el atributo *android:inputType* (se pueden utilizar varios separando con "|"):

- text (valor por defecto): texto normal
- textCapCharacters: mayúsculas
- textCapSentences: primera letra mayúscula
- textCapWords: primera letra de cada palabra mayúscula

Controles básicos (4)

■ Edición de texto: EditText

- textMultiLine: texto en varias líneas
- textNoSuggestions: desactiva sugerencias
- textUri: web URL
- textEmailAddress: correo electrónico
- textShortMessage: activa el acceso a smiley
- textPassword: contraseña (oculta)
- textVisiblePassword: contraseña (visible)
- number | date | phone: teclado numérico

Controles básicos (5)

■ Botón: Button

Permite realizar una acción asociada al botón que se puede establecer directamente mediante la propiedad *onClick*.

```
<Button
    android:id="@+id/botAceptar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/botOk"
    android:onClick="metodo" />
```

Controles básicos (6)

■ Casilla de verificación: CheckBox

Representa una casilla cuyo valor se activa al pulsar sobre ella.

La propiedad *checked* permite establecer el valor inicial: *true* (activo) o *false* (inactivo).

```
<CheckBox
    android:id="@+id/checkbox1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:checked="true"
    android:text="@string/opcion1" />
```

Controles básicos (7)

■ Casilla de verificación: *CheckBox*

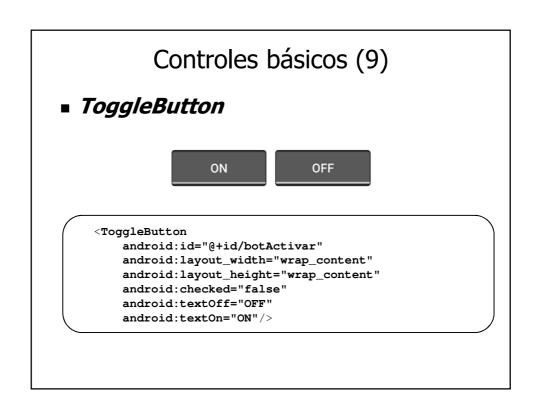
Cada vez que cambia el estado de un checkBox se invoca al método onCheckedChanged que recibirá dos parámetros: el objeto cuyo estado ha cambiado y el nuevo estado.

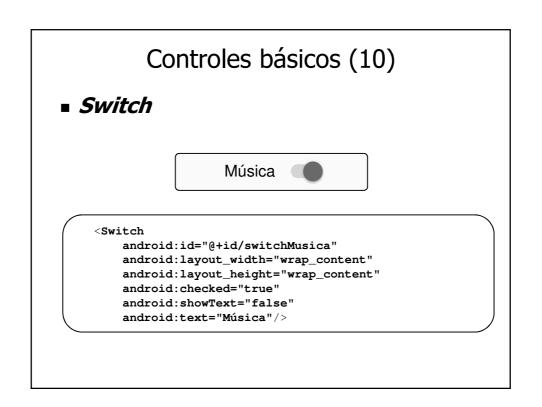
Controles básicos (8)

■ ToggleButton y Switch

Son botones con dos estados (*on/off*) con un funcionamiento similar al *checkBox*, pero diferente apariencia.

Ambos tienen las propiedades "textOn" y "textOff" para establecer el texto que se muestra en cada estado, pero Switch requiere activar la propiedad "showText" para que se muestre el estado y suele utilizarse cuando no es necesario.





Controles básicos (11)

■ RadioGroup y RadioButton

Permiten crear un grupo de opciones excluyentes entre sí, de manera que sólo se puede seleccionar una de las opciones.

RadioGroup es un contenedor en el que se enmarcan todas las opciones posibles, mientras que RadioButton será cada una de esas opciones.

RadioButton, al igual que CheckBox, dispone de la propiedad checked y del método onCheckedChanged.

Controles básicos (12)

■ RadioGroup y RadioButton

Controles básicos (13)

■ RadioGroup y RadioButton

Existen métodos para obtener y/o modificar el estado desde el elemento *RadioGroup*:

- *void check* (*int id*): activa el *RadioButton* cuyo identificador sea "id".
- *void clearCheck* (): desactiva todos los *RadioButton*.
- *int getCheckedRadioButtonId* (): devuelve el "id" del *RadioButton* seleccionado.

Controles básicos (14)

■ RadioGroup y RadioButton

Y también desde el propio RadioButton:

- *void toggle* (): modifica el estado del *RadioButton*.

Controles básicos (15)

■ Imagen: *ImageView*

ImageView nos permite añadir imágenes a la aplicación de forma muy sencilla

- src. ruta de la imagen a insertar.
- contentDescription: descripción de la imagen (accesibilidad).

```
<ImageView
   android:id="@+id/logo"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:src="@drawable/img_logo"
   android:contentDescription="@string/logo" />
```

Controles básicos (16)

■ Gestionar el click

A la hora de asignar un comportamiento a un botón de nuestra aplicación tenemos tres alternativas:

- Gestionar el *click* en los botones de forma separada desde XML.
- Gestionar el *click* en los botones de forma separada desde Java.
- Gestionar el *click* en la actividad de forma conjunta.

Controles básicos (17)

• Gestionar el *click* en los botones de forma separada desde XML.

Es el mecanismo más sencillo ya que basta con establecer el método a ejecutar en la propiedad "onClick" de cada uno de los botones y añadir el código del método a la clase asociada a la Actividad.

Este método es válido a partir de la API nivel 4 pero presenta problemas en algunos dispositivos y además rompe con la separación entre interfaz y aplicación.

Controles básicos (18)

• Gestionar el *click* en los botones de forma separada desde XML.

```
public class MainActivity extends Activity {
    // Otros métodos
   public void metodoBotOK (View vista) {
        // Código del método asociado al botón
    }
}
```

Controles básicos (19)

• Gestionar el *click* en los botones de forma separada desde Java.

Consiste en añadir un *Listener* para cada botón mediante *setOnClickListener* y sobrecargar su método *onClick* con el código a ejecutar.

```
Button btn1 = (Button) findViewById(R.id.botOk);
btn1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View vista) {
        // Código del método asociado al botón
    }
});
```

Controles básicos (20)

■ Gestionar los *click* en la actividad de forma conjunta.

Hacemos que la actividad implemente la interfaz *onClickListener* y en el método *onClick* identificamos el botón pulsado y le asignamos el código que le corresponde.

```
public class MainActivity extends Activity implements
  View.OnClickListener { ... }
```

```
Button btn1 = (Button) findViewById(R.id.botOk);
btn1.setOnClickListener(this);
Button btn2 = (Button) findViewById(R.id.botCancel);
btn1.setOnClickListener(this);
```

Controles básicos (21)

■ Gestionar los *click* en la actividad de forma conjunta.