# INDICE

1.1. INTRODUCCIÓN 2

1.2. PREPARANDO A ACTIVITY 2

1.3. TEXTVIEW 5

1.3.1. EVENTO INTERFACE ONCLICKLISTENER 5

1.4. EDITTEXT 6

1.4.1 EVENTO INTERFACE CLICKLISTENER 6

1.4.2. EVENTOS MÍNIMOS A CONOCER EN EL EDITTEXT 9

1.5. BUTTON - IMAGEBUTTON 9

1.5.1. EVENTO INTERFACE ONCLICKLISTENER POR EL DISEÑADOR 9

1.5.2. EVENTO INTERFACE ONCLICKLISTENER POR CÓDIGO 11

1.5.3. EVENTOS MÍNIMOS A CONOCER EN EL BUTTON 12

1.6. FLOATACTIONBUTTON 12

1.6.1. EVENTO INTERFACE ONCLICKLISTENER 12

1.6.2. EVENTOS MÍNIMOS A CONOCER EN EL FAB 14

1.7. TOGGLEBUTTON - SWITCH 14

1.7.1. EVENTO INTERFACE ONCHECKEDCHANGELISTENER 14

1.7.2. EVENTOS MÍNIMOS A CONOCER EN EL SWITCH/TOGGLEBUTTON 15

1.8. CHECKBOX 15

1.8.1. EVENTO INTERFACE ONCHECKEDCHANGELISTENER 16

1.8.2. EVENTOS MÍNIMOS A CONOCER EN EL CHECKBOX 17

1.9. RADIOBUTTON-RADIOGROUP 17

1.9.1. EVENTO INTERFACE ONCHECKEDCHANGELISTENER EN EL RADIOBUTTON 17

1.9.2. EVENTO INTERFACE ONCHECKEDCHANGELISTENER NO RADIOGROUP 19

1.9.3. EVENTOS MÍNIMOS A CONOCER EN EL RADIOBUTTON / RADIOGROUP 21

1.10. IMAGEVIEW 21

1.10.1. EVENTO INTERFACE ONCLICKLISTENER 22

1.10.2. EVENTO INTERFACE ONLONGCLICKLISTENER 22

1.10.3. EVENTOS MÍNIMOS A CONOCER EN IMAGEVIEW 25

# 1.1. Introducción

✓ En este punto vamos a ver cómo gestionar los principales eventos que se puedan producir en los Views que vimos anteriormente.

Recordar que ya vimos anteriormente cómo funcionan las interfaces y como asociarlas a los Views empleando el método setOnZZZZZZZ correspondiente.

✓ Nota: En muchos controles el evento principal que vais a querer controlar, será el 'Click' sobre el mismo. Recordar que el método que registra dicho evento está definido en la clase View y por lo tanto vamos a poder gestionar de la misma forma el evento Click en cualquiera View que tengamos.

# 1.2. Preparando a Activity

✓ Vamos a crear una única Activity donde estarán todos los Views que vamos a analizar (o ListView/Spinner/RecyclerView.

✓ Crear un paquete **Eventos** y dentro de él vamos a crear una nueva 'Empty Activity' de nombre: **UD03_01_xesteventos_base** de tipo Launcher.

Modificar el archivo **AndroidManifiest.xml** y añadir una etiqueta a la activity.

## Código del Layout:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".Eventos.UD03_01_xesteventos_base">

    <TextView
        android:id="@+id/txvTextoTextView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:text="Texto Text View"
        android:textSize="30sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <EditText
        android:id="@+id/etvTextoEditText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:ems="10"
        android:hint="Texto Edit Text"
        android:imeOptions="actionSend"
        android:inputType="textPersonName"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/txvTextoTextView" />

    <Button
        android:id="@+id/btnBoton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginBottom="8dp"
        android:onClick="presBoton"
        android:text="BOTÓN"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <ImageButton
        android:id="@+id/imgbtnImageButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginBottom="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toEndOf="@+id/btnBoton"
        app:srcCompat="@android:drawable/star_on" />

    <CheckBox
        android:id="@+id/chk_Uno"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
```

```
        android:text="Check Uno"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/etvTextoEditText" />

    <CheckBox
        android:id="@+id/chk_Dos"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Check Dos"
        app:layout_constraintStart_toEndOf="@+id/chk_Uno"
        app:layout_constraintTop_toBottomOf="@+id/etvTextoEditText" />

    <CheckBox
        android:id="@+id/chk_Tres"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Check Tres"
        app:layout_constraintStart_toEndOf="@+id/chk_Dos"
        app:layout_constraintTop_toBottomOf="@+id/etvTextoEditText" />

    <RadioGroup
        android:id="@+id/rgrpGrupoBotons"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:checkedButton="@id/rbtnOpcionA"
        android:orientation="horizontal"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/chk_Uno">

        <RadioButton
            android:id="@+id/rbtnOpcionA"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Opción A)" />

        <RadioButton
            android:id="@+id/rbtnOpcionB"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Opción B)" />
    </RadioGroup>

    <ToggleButton
        android:id="@+id/tbtnToogleBoton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="ToggleButton"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/rgrpGrupoBotons" />

    <Switch
        android:id="@+id/swbSwitchButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:text="Switch"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tbtnToogleBoton" />-->

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fabFloatActionButton"
        android:layout_width="wrap_content"
```

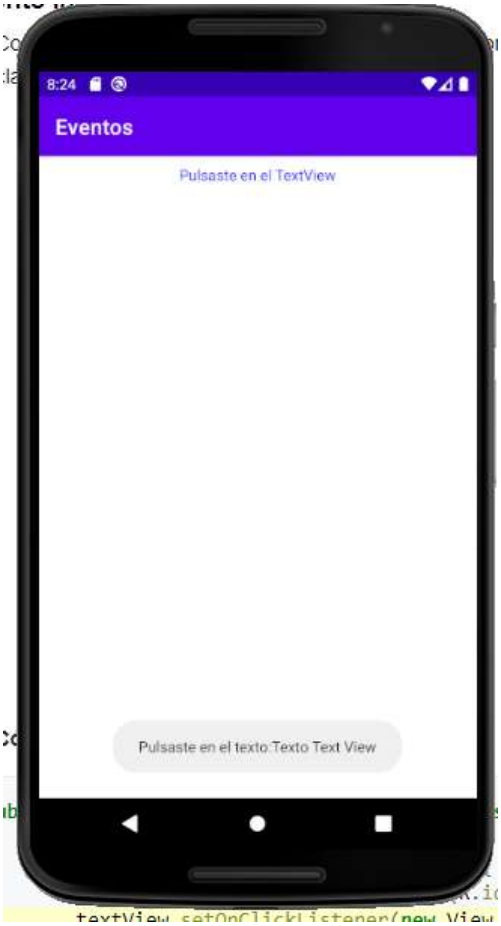
```
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        android:clickable="true"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:srcCompat="@android:drawable/ic_input_add"  />

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="114dp"
        android:layout_height="97dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:background="#AC2727"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/swbSwitchButton"
        app:srcCompat="@android:drawable/btn_star_big_on" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

# 1.3. TextView

✓ En este control podríamos querer hacer algo cuando clickeamos sobre el texto.

## 1.3.1. Evento Interface OnClickListener

✓ El evento Click se gestiona desde la Interface OnClickListener mediante el método setOnClickListener que se encuentra definido en la clase View y por lo tanto la forma de gestionarlo es siempre la misma.



**Código da Activity**:

```
public class UD03_01_xesteventos_base extends AppCompatActivity {

    private void xestionarEventosTextView(){
        TextView textView = findViewById(R.id.txvTextoTextView);
        textView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                TextView tv = (TextView) view;
                Toast.makeText(getApplicationContext(),"Pulsaste en el texto:" +
tv.getText().toString(),Toast.LENGTH_SHORT).show();
                tv.setText("Pulsaste en el TextView");
                tv.setTextColor(Color.BLUE);
                tv.setTextSize(14);
            }
        });
    }
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d03_01_xesteventos_base);
        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_HIDDEN);  // Para que no
aparezca el teclado virtual
        xestionarEventosTextView();

    }
}
```

**Línea 5**: Llamada al método setOnClickListener para 'asociar' el evento Click al textview. Enviamos un objeto que implementa la interface OnClickListener de forma anónima (podríamos hacerlo a nivel de Activity como ya vimos anteriormente).

**Línea 8**: El view sabemos que es el TextView sobre el que pulsamos.

**Línea 21**: Para que quede con un poco de orden vamos a crear un método para gestionar cada uno de los views.

# 1.4. EditText

## 1.4.1. Evento Interface ClickListener

✓ En este view normalmente no queremos gestionar un evento de Click sobre el (se puede hacer igual que en el caso anterior).

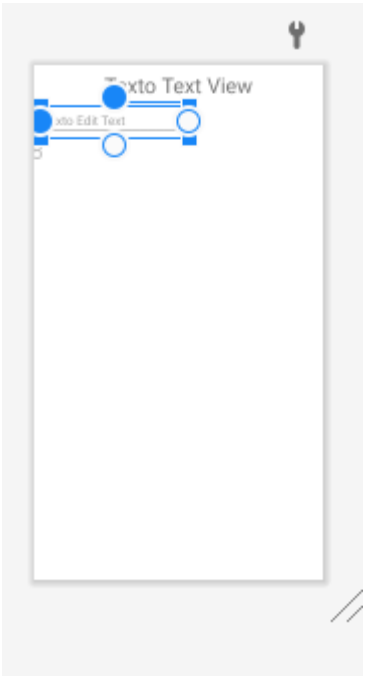
```
        findViewById(R.id.etvTextoEditText).setOnClickListener(new View.OnClickListener() {  // Xestionamos o evento de Click
cunha interface anónima
            @Override
            public void onClick(View view) {
                Toast.makeText(getApplicationContext(),"EditText Pulsado",Toast.LENGTH_LONG).show();
            }
        });
```

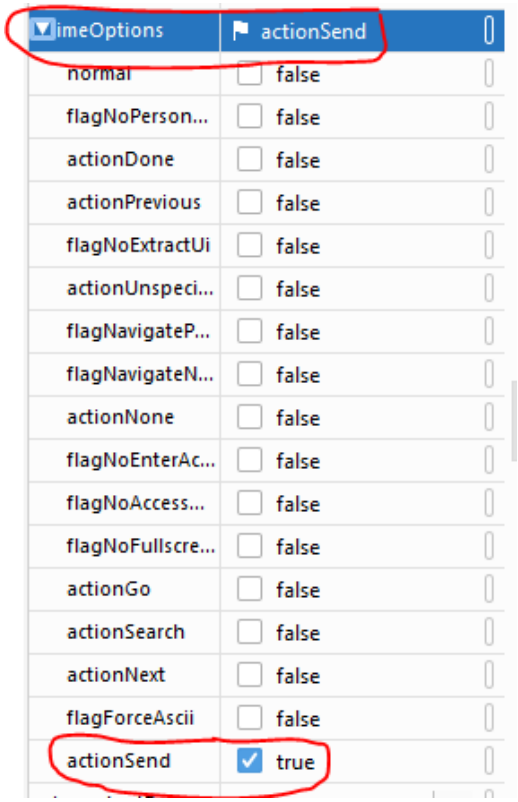
✓ Un evento que se podría querer controlar es la tecla del teclado virtual que indica una acción.

Esto se consigue llamando al método setOnEditorActionListener y pasando como parámetro un objeto de una clase que implemente a interface OnEditorActionListener.

Al implementar la interface tendremos el acceso al método onEditorAction.

Vamos a cambiar en el diseñador la acción que va a tener en el teclado virtual el EditText. Marcaremos la acción 'send':



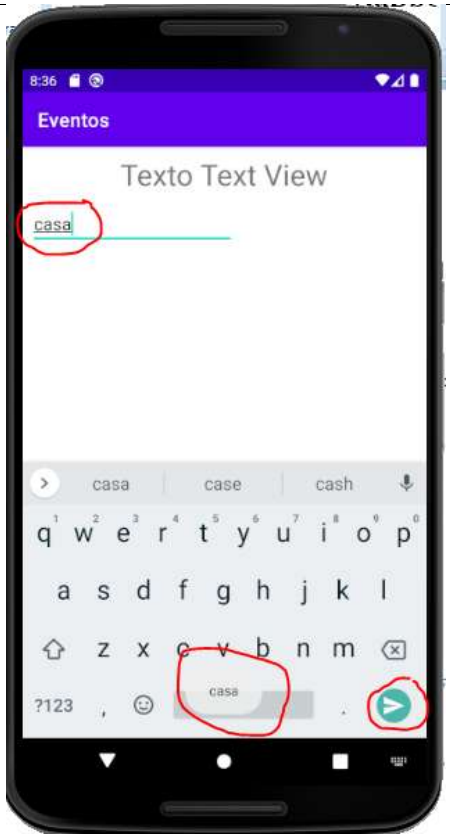




✓ Codificación en Java que captura la acción actionSend de un campo de texto (email) y es lo que hace al recoger el texto de esa caja de texto, el e-mail, y volver a imprimir un mensaje en esa misma caja de texto.

```
public class UD03_01_xesteventos_base extends AppCompatActivity {

    private void xestionarEventosTextView(){


        EditText editText = findViewById(R.id.etvTextoEditText);
        editText.setOnEditorActionListener(new TextView.OnEditorActionListener() {
            @Override
            public boolean onEditorAction(TextView textView, int i, KeyEvent keyEvent) {
                // El textView es el texto escrito en el EditText. Podemos obtener dicho texto por
cualquiera de los dos views.
                boolean handled = false;
                if (i == EditorInfo.IME_ACTION_SEND) {  // En la clase EditorInfo tenemos todas las
constantes que se identifican con cada tipo de acción
                    String textoEscrito = textView.getText().toString();

                    Toast.makeText(getApplicationContext(),textoEscrito,Toast.LENGTH_SHORT).show();
                    handled = true;
                }
                return handled;
            }
        });

    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d03_01_xesteventos_base);
        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_HIDDEN);  // Para que
no aparezca el teclado virtual
```

```
            xestionarEventosTextView();

        }
    }
```

- ✓ **Línea 4**: creamos un objeto (editText) que apunta al EditText en el que se introduce el dato. Fijarse que se está declarando como final para poder acceder a él dentro de la interface anónima (realmente, para obtener el texto escrito no será necesario).
- ✓ **Línea 6**: llamamos al método Listener (*escuchador*) del objeto editText: **setOnEditorActionListener ()**
  - o Ese método será llamado cuando se realice una acción en un EditText
  - o Recordar que un EditText es una subclase de TextView
  - o Por ejemplo, cuando se pulsa una tecla o cuando haya una acción IME seleccionada por el usuario.
  - o Como parámetro se le pasará la creación de una clase anónima que implementa una interface (OnEditorActionListener) para el cual hay que sobreescribir el único método que tiene la interface (onEditorAction()).
- ✓ **Línea 11-18**: sobreescritura del método OnEditorAction() en el que se reciben 3 parámetros:
  - o El TextView que genero el evento (v). En este caso el EditText
  - o El ID de la acción enviada
  - o Y si el evento fue generado por la tecla **Enter** o no (event)
  - o Comprobamos si el ID de la acción es el SEND, en ese caso se muestra el texto escrito.
- ✓ Este método devuelve un boolean indicando se el evento de pulsar en la tecla del teclado virtual debe ser propagado y que otros controles puedan capturar dicho evento (esto se hace haciendo un return false) o se indicamos que dicho evento ya fue gestionado y no debe propagarse más (return true).
- ✓ Para que el IDE nos cree a **clase anónima**:
  - o Escribir la llamada al método (ojo ; final incluido): **editText.setOnEditorActionListener( );**
  - o Escribir **new** entre los paréntesis: **editText.setOnEditorActionListener(new );**
  - o Pulsar CTRL+Barra espaciadora y ya el sistema completa todo lo demás.
  - o Logo sólo queda poner nuestro código en el método a sobreescribir.
- ✓ Recordar pulsar **SHIFT+CTRL+O** para importar los paquetes correspondientes o podemos hacer que lo haga automáticamente.




### 1.4.2.Eventos mínimos a conocer en el EditText

- ✓ Gestionar el evento de Click sobre cualquier EditText y saberlo hacer empleando interfaces anónimas o implementando la interface Activity.

## 1.5. Button - ImageButton

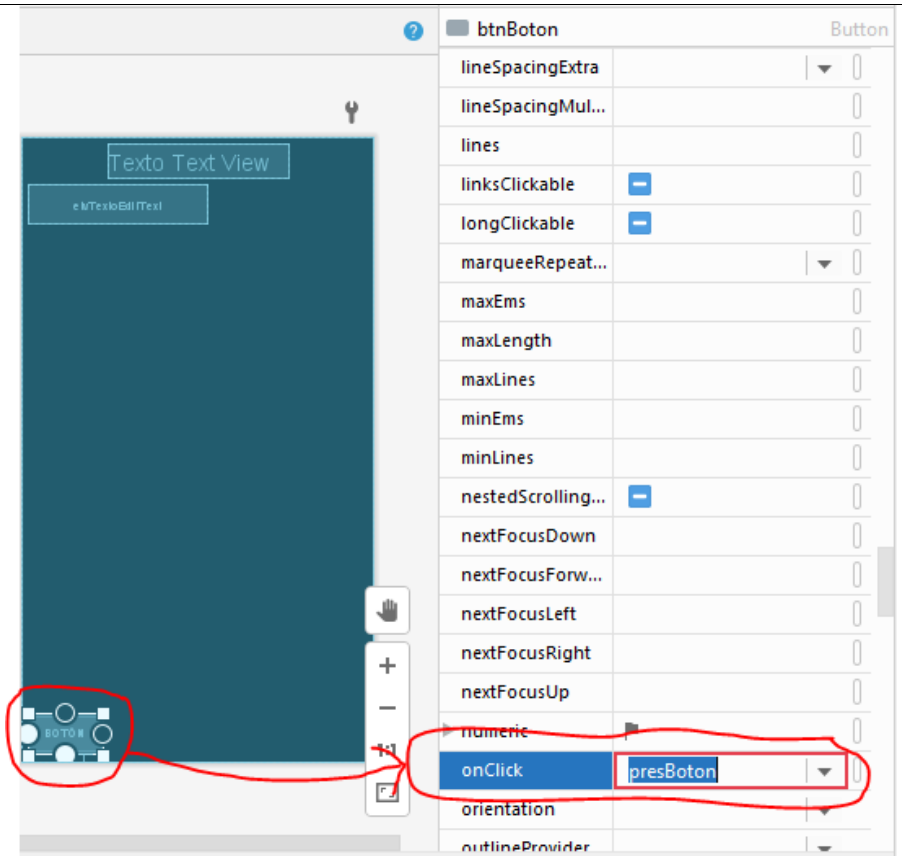
- ✓ El principal evento que queremos gestionar en un botón es el Click sobre el mismo.

  Este tipo de evento se puede 'implementar' por código o a través del diseñador.

  A través del diseñador es más fácil pero no se aconseja ya que estamos mezclando aspectos visuales con eventos que se producen en la view.

### 1.5.1.Evento Interface OnClickListener por el diseñador

- ✓ Esta opción no está recomendada.
- ✓ Es la forma más sencilla de desencadenar una acción pero no es la más aconsejable ya que estamos a mezclar el diseño con la programación de eventos.
- ✓ Vamos a hacerlo de dos formas:
  - Creando un método para cada Botón.
  - Creando un único método para todos los botones. Hay que controlar que botón fue el que se pulso. La gestión de los Click´s la hacemos como indicamos anteriormente.
- ✓ Lo que tenemos que hacer es a nivel gráfico, en el diseñador, ir a la propiedad onClick del botón y escribir el nombre del método al que queremos que llame cuando se pulsa el botón.




✓ Ahora a nivel de nivel de código tenemos que definir un método con ese nombre o un parámetro de tipo View de la forma:

```
public class UD03_01_xesteventos_base extends AppCompatActivity {

    public void presBoton(View v){
        Button boton = (Button)v;
        Toast.makeText(getApplicationContext(),"Pulsaste el botón con texto " + bo-
ton.getText().toString(),Toast.LENGTH_SHORT).show();
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d03_01_xesteventos_base);
        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_HIDDEN);  // Para que no
aparezca el teclado virtual
            }
}
```

✓ El parámetro v es el view que provoca el evento click.

A través de él es como sabemos si tenemos varios botones en el diseñador con el mismo nombre de método en la propiedad onClick.

Nota: Recordar eliminar la propiedad onClick del diseñador para continuar con el manual.

## 1.5.2.Evento Interface OnClickListener por código

✓ Dicho evento está asociado a la interface OnClickListener da clase View.

Al implementar tenemos el acceso al método onClick siendo el parámetro View el view que provoco el evento Click.



### Código de la Activity:

```
public class UD03_01_xesteventos_base extends AppCompatActivity {

    public void presBoton(View v){
        Button boton = (Button)v;
        Toast.makeText(getApplicationContext(),"Pulsaste el botón con texto " + bo-
ton.getText().toString(),Toast.LENGTH_SHORT).show();
    }



    private void xestionarEventosButton(){
        findViewById(R.id.btnBoton).setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View view) {
                        Button boton = (Button)view;        // Sabemos que es un Button al hacerlo me-
diante interface anónimas
                        EditText editText = findViewById(R.id.etvTextoEditText);
                        TextView textView = findViewById(R.id.txvTextoTextView);

                        Toast.makeText(getApplicationContext(),"Pulsaste el botón con texto " + bo-
ton.getText().toString(),Toast.LENGTH_SHORT).show();
                        textView.setText(editText.getText().toString());    // Pasamos el contenido del
EditText al TextView
                    }
                }
        );
```

```
        findViewById(R.id.imgbtnImageButton).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                ImageButton boton = (ImageButton) view;        // Sabemos que es un ImageButton al hacer-
lo mediante interface anónima
                EditText editText = findViewById(R.id.etvTextoEditText);
                TextView textView = findViewById(R.id.txvTextoTextView);

                Toast.makeText(getApplicationContext(),"Pulsaste la ImageBut-
ton",Toast.LENGTH_SHORT).show();
            }
        });
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d03_01_xesteventos_base);

        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_HIDDEN);  // Para que no
aparezca el teclado virtual
        xestionarEventosButton();
    }
}
```

### 1.5.3.Eventos mínimos a conocer en el Button

✓ Gestionar el evento de Click sobre cualquier Button y saberlo hacer empleando interfaces anónimas o implementando la interface en la Activity.

```
        Button btn = findViewById(R.id.btnBoton);           // Referenciamos a un botón que estea no Layout da Activity


        btn.setOnClickListener(new View.OnClickListener() {  // Xestionamos o evento de Click cunha interface anónima
            @Override
            public void onClick(View view) {
                Button boton = (Button) view;  // Podemos obter a referencia ao view que provocou o evento do Click
                Toast.makeText(getApplicationContext(),"Botón Pulsado",Toast.LENGTH_LONG).show();
            }
        });
```

## 1.6. FloatActionButton

### 1.6.1.Evento Interface OnClickListener

✓ El evento Click se gestiona desde la Interface OnClickListener mediante el método setOnClickListener que se encuentra definido en la clase View y por lo tanto la forma de gestionarlo siempre es la misma.


## Código de la Activity:

```
public class UD03_01_xesteventos_base extends AppCompatActivity {

            private void xestionarEventosFab(){
            findViewById(R.id.fabFloatActionButton).setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    FloatingActionButton fab = (FloatingActionButton) view;  //En este caso el View es un
FAB

                    TextView textView = findViewById(R.id.txvTextoTextView);
                    textView.setText("Pulsado el FAB");
                }
            });
        }


        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_u_d03_01_xesteventos_base);

            getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_HIDDEN);  // Para
que no aparezca o teclado virtual
            xestionarEventosFab();
        }
    }
```

### 1.6.2.Eventos mínimos a conocer en el FAB

- ✓ Gestionar el evento del Click sobre cualquier Float Action Button y saberlo hacer empleando interfaces anónimas o implementando la interface en la Activity.

## 1.7. ToggleButton - Switch

- ✓ En este tipo de botones el evento que nos va a interesar gestionar va a ser cuando se pulsa el botón y **se produce un cambio de estado**.
- ✓ Recordar que estos botones tienen dos estados (activado / desactivado).
- ✓ Como en los casos anteriores, también podemos gestionar el Click, pero es más correcto emplear la opción anterior.
- ✓ Recordar también que ya vimos como obtener el estado (on/off) de este tipo de botones empleando el método isChecked().

### 1.7.1.Evento Interface OnCheckedChangeListener

- ✓ Evento que se produce cuando hay un cambio de estado en el botón.
- ✓ Para hacer que el Switch/Toggle Button 'escuche' este tipo de evento tenemos que llamar al método setOnCheckedChangeListener() y pasarle como parámetro un objeto que implemente la interface OnCheckedChangeListener.
- ✓ Fijarse que dicha interface pertenece a una clase CompoundButton igual que el método setOnCheckedChangeListener. Por lo tanto no está definido a nivel de View y no todos los views implementan dicho evento (solamente los CompoundButton).
- ✓ Por lo tanto no podemos emplear como en los casos anteriores dos botones: findViewById(R.id.id_switchbutton).setOnCheckedChangeListener() ya que dicho método no se encuentra.
- ✓ Podemos hacer el findViewById y guardar la referencia en un objeto del tipo correspondiente y después llamar al chamar método, o si no queremos guardar la referencia en una variable, tenemos que hacer un Cast de lo que devuelva el findViewById.

**Código de la Activity:**

```
public class UD03_01_xesteventos_base extends AppCompatActivity {
    private void xestionarSwitchToogle(){
        ((Switch)findViewById(R.id.swbSwitchButton)).setOnCheckedChangeListener(new CompoundBut-
ton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
                Switch swButton = (Switch)compoundButton;       // Cuidado de no emplear como nombre de
la variable switch
                TextView textView = findViewById(R.id.txvTextoTextView);
                textView.setText("Pulsaste en el switch con texto " + swButton.getText() + " y su nuevo
estado es " + String.valueOf(b));
            }
        });

        ((ToggleButton)findViewById(R.id.tbtnToogleBoton)).setOnCheckedChangeListener(new CompoundBut-
ton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
                ToggleButton toggleButton = (ToggleButton) compoundButton;       // Sabemos que es un
ToggleButton. Recordar que tenemos
                // el método compoundButton.getId() para obtener el id se hacemos la gestión de eventos a
nivel de Activity
                TextView textView = findViewById(R.id.txvTextoTextView);
                textView.setText("Pulsaste en el switch con texto " + toggleButton.getText().toString()+
" y su nuevo estado es " + String.valueOf(b));
            }
        });
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d03_01_xesteventos_base);

        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_HIDDEN);  // Para que no
aparezca el teclado virtual
        xestionarSwitchToogle();
    }
}
```

### 1.7.2.Eventos mínimos a conocer en el Switch/ToggleButton

- ✓ Gestionar el evento de Click y CheckedChange sobre cualquier Switch / ToggleButton y saberlo hacer empleando interfaces anónimas o implementando la interface en la Activity.

## 1.8. CheckBox

- ✓ El CheckBox tiene el mismo comportamiento que el caso del Switch / ToggleButton.
- ✓ En este tipo de Views el evento que nos va a interesar gestionar va a ser cuando se pulse en el Check y **se produce un cambio de estado.**
- ✓ Recordar que estos botones tienen dos estados (activado / desactivado).
- ✓ Como en los casos anteriores, también podemos gestionar el Click, pero es más correcto emplear la opción anterior.
- ✓ Al igual que en el caso de los Buttons, la gestión del evento Click se puede hacer desde el Layout.
- ✓ Recordar que ya vimos como obtener el estado (on/off) de este tipo de botones empleando el método isChecked().

## 1.8.1.Evento Interface OnCheckedChangeListener

- ✓ Evento que se produce cuando hay un cambio de estado en el Check.
- ✓ Para hacer que un CheckBox 'escuche' este tipo de evento tenemos que llamar al método setOnCheckedChangeListener() y pasarle como parámetro un objeto que implemente la interface OnCheckedChangeListener.
- ✓ Fijarse que dicha interface pertenece a una clase CompoundButton igual que el método setOnCheckedChangeListener. Por lo tanto no está definido a nivel de View y no todos los views implementan dicho evento (solamente los CompoundButton).
- ✓ Por lo tanto no podemos emplear como en los casos anteriores de los botones: findViewById(R.id.id_switchbutton).setOnCheckedChangeListener() ya que dicho método no se encuentra.
- ✓ Podemos hacer el findViewById y guardar la referencia en un objeto de tipo correspondiente y después llamar al método, o se no queremos guardar la referencia en una variable, tendremos que hacer un Cast de lo devuelva el findViewById.


### Código da Activity:

```
public class UD03_01_xesteventos_base extends AppCompatActivity {
    private void xestionarEventosCheckBox(){
        ((CheckBox)findViewById(R.id.chk_Uno)).setOnCheckedChangeListener(new CompoundBut-
ton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
                CheckBox checkBox = (CheckBox)compoundButton;
                Toast.makeText(getApplicationContext(),"Pulsaste en el check con texto " + check-
Box.getText().toString() + " y estado a " + String.valueOf(b),Toast.LENGTH_SHORT).show();
            }
        });
        ((CheckBox)findViewById(R.id.chk_Dos)).setOnCheckedChangeListener(new CompoundBut-
ton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
```

```
                CheckBox checkBox = (CheckBox)compoundButton;
                Toast.makeText(getApplicationContext(),"Pulsaste en el check con texto " + check-
Box.getText().toString() + " y estado a " + String.valueOf(b),Toast.LENGTH_SHORT).show();
            }
        });
        ((CheckBox)findViewById(R.id.chk_Tres)).setOnCheckedChangeListener(new CompoundBut-
ton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
                CheckBox checkBox = (CheckBox)compoundButton;
                Toast.makeText(getApplicationContext(),"Pulsaste en el check con texto " + check-
Box.getText().toString() + " y estado a " + String.valueOf(b),Toast.LENGTH_SHORT).show();
            }
        });
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d03_01_xesteventos_base);

        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_HIDDEN);  // Para que no
aparezca el teclado virtual
        xestionarEventosCheckBox();
    }
}
```

### 1.8.2.Eventos mínimos a conocer en el CheckBox

- ✓ Gestionar el evento de Click sobre cualquier CheckBox y saberlo hacer empleando interfaces anónimas o implementando la interface en la Activity.
- ✓ Gestionar el evento CheckedChange sobre cualquier CheckBox y saberlo hacer empleando interfaces anónimas o implementando la interface en la Activity.

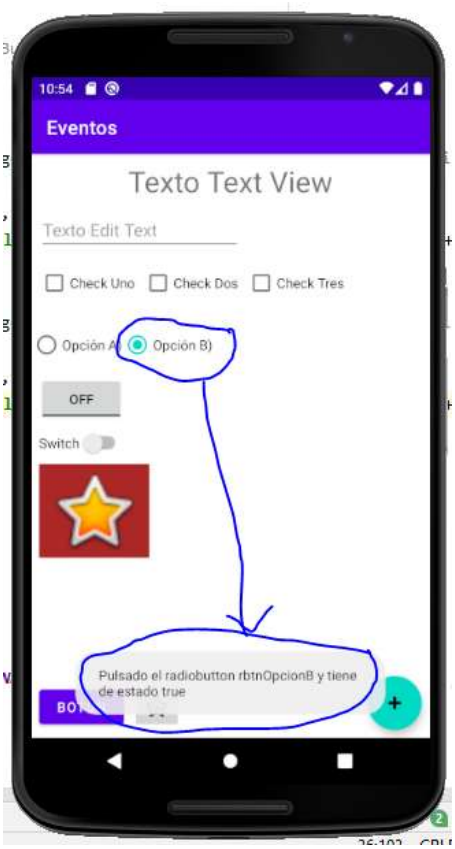
## 1.9. RadioButton-RadioGroup

- ✓ El RadioButton tiene el mismo comportamiento que en el caso del Switch / ToggleButton / CheckBox explicado anteriormente.
- ✓ En este tipo de Views el evento que nos va a interesar gestionar va a ser cuando se pulsa en el RadioButton y se produce un cambio de estado.
- ✓ Recordar que estos botones tienen dos estados (activado / desactivado).
- ✓ Como en los casos anteriores, también podemos gestionar el Click, pero es más correcto emplear la opción anterior.
- ✓ Al igual que en el caso de los Buttons, la gestión del evento Click se puede hacer desde el Layout.
- ✓ Recordar que ya vimos como obtener el estado (activado / desactivado) de este tipo de botones empleando el método isChecked().

### 1.9.1.Evento Interface OnCheckedChangeListener en el RadioButton

- ✓ Evento que se produce cuando hay un cambio de estado en un RadioButton concreto.
- ✓ Para hacer que un RadioButton 'escuche' este tipo de evento tenemos que llamar al método setOnCheckedChangeListener() y pasarle como parámetro un objeto que implemente la interface OnCheckedChangeListener.

- ✓ Fijarse que dicha interface pertenece a una clase CompoundButton igual que el método setOnCheckedChangeListener. Por lo tanto no está definido a nivel de View y no todos los views implementan dicho evento (solamente los CompoundButton).
- ✓ Por lo tanto no podemos emplear como en los casos anteriores de los botones: findViewById(R.id.id_switchbutton).setOnCheckedChangeListener() ya que dicho método no se encuentra.
- ✓ Podemos hacer el findViewById y guardar la referencia a un objeto del tipo correspondiente y después llamar al método, o si no queremos guardar la referencia de una variable, tendremos que hacer un Cast de lo que devuelve el findViewById.




## Código da Activity:

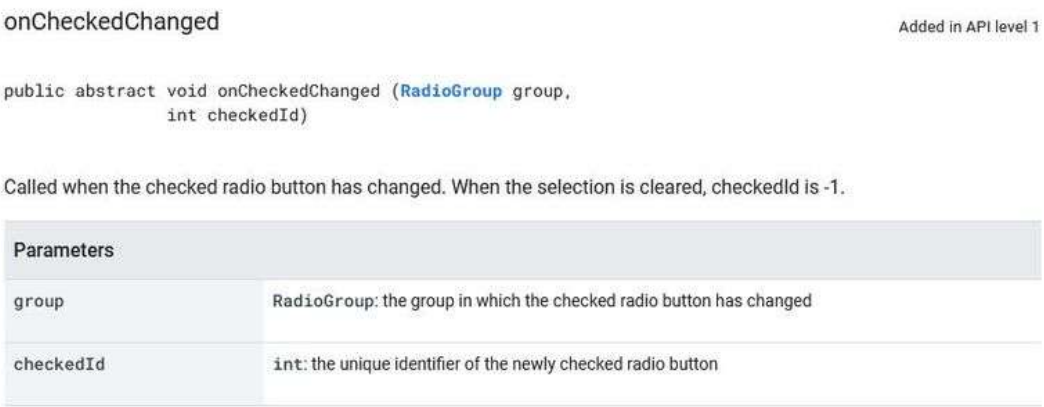
```
public class UD03_01_xesteventos_base extends AppCompatActivity {
    private void xestionarEventosRadioButton(){
        ((RadioButton)findViewById(R.id.rbtnOpcionA)).setOnCheckedChangeListener(new CompoundBut-
ton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
                Toast.makeText(getApplicationContext(),"Pulsado el radiobutton rbtnOpcionA y tiene de
estado " + String.valueOf(b),Toast.LENGTH_SHORT).show();
            }
        });
        ((RadioButton)findViewById(R.id.rbtnOpcionB)).setOnCheckedChangeListener(new CompoundBut-
ton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
                Toast.makeText(getApplicationContext(),"Pulsado el radiobutton rbtnOpcionB y tiene de
estado " + String.valueOf(b),Toast.LENGTH_SHORT).show();
            }
        });
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d03_01_xesteventos_base);
```

```
        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_HIDDEN);  // Para que no aparezca el teclado virtual
        xestionarEventosRadioButton();
    }
}
```

- ✓ Fijarse que al ejecutar la aplicación nos van a salir dos mensajes. Cuando el radiobutton Opción a) pasa de true a false y cuando el radiobutton Opción b) pasa de false a true.
- ✓ Esta sería una forma de hacerlo, pero veremos a continuación que en vez de crear un Listener por cada radiobutton, vamos a hacer que sea el RadioGroup el que escuche el cambio de estado e informe de qué radiobutton cambio de estado (**esta será la forma correcta de hacerlo**).

## 1.9.2.Evento Interface OnCheckedChangeListener no RadioGroup

- ✓ Evento que se produce cando hay un cambio de estado en un RadioGroup y se llamará a dicho evento cuando cualquier RadioButton que este dentro del RadioGroup cambie de estado.
- ✓ Para hacer que un RadioGroup 'escuche' este tipo de evento tenemos que llamar al método setOnCheckedChangeListener() del RadioGroup e pasarle como parámetro un objeto que implemente la interface OnCheckedChangeListener del ViewGroup.
- ✓ Fijarse que dicha interface pertenece a una clase ViewGroup igual que el método setOnCheckedChangeListener.
- ✓ Por lo tanto no podemos emplear como en los casos anteriores de los botones: findViewById(R.id.id_viewgroup).setOnCheckedChangeListener() ya que dicho método no se encuentra.
- ✓ Podemos hacer el findViewById y guardar la referencia a un objeto de tipo correspondiente y después llamar al chamar al método, o se no queremos guardar la referencia en una variable, tendremos que hacer un Cast de lo que devuelva el findViewById.
- ✓ Se consultamos la guía de referencia de Android podemos observar cómo el método de la interface OnCheckedChangeListener se llama onCheckedChanged y tiene como segundo parámetro el id del radiobutton que cambio de estado.



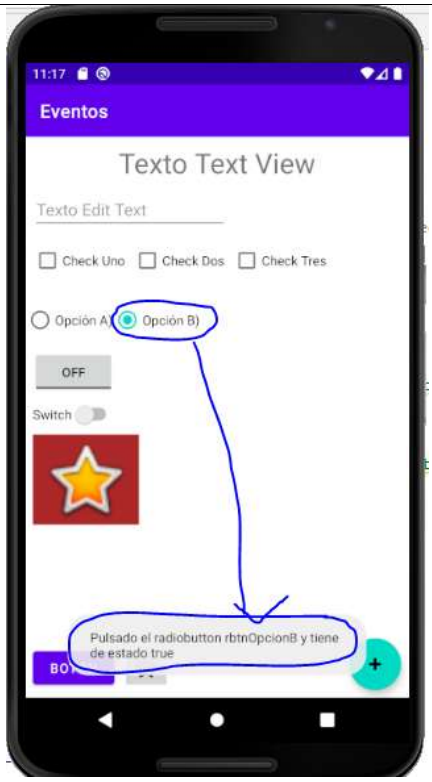
onCheckedChanged

Added in API level 1

```
public abstract void onCheckedChanged (RadioGroup group,
                int checkedId)
```

Called when the checked radio button has changed. When the selection is cleared, checkedId is -1.

| Parameters | |
|---|---|
| group | RadioGroup: the group in which the checked radio button has changed |
| checkedId | int: the unique identifier of the newly checked radio button |




```
public class UD03_01_xesteventos_base extends AppCompatActivity {
    private void xestionarEventosRadioGroup(){
        ((RadioGroup)findViewById(R.id.rgrpGrupoBotons)).setOnCheckedChangeListener(new Radio-
Group.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(RadioGroup radioGroup, int i) {
                RadioButton radioButton = findViewById(i);
                switch(i){
                    case R.id.rbtnOpcionA:
                        Toast.makeText(getApplicationContext(),"Pulsado el radiobutton rbtnOpcionA y tie-
ne de estado " + String.valueOf(radioButton.isChecked()),Toast.LENGTH_SHORT).show();
                        break;
                    case R.id.rbtnOpcionB:
                        Toast.makeText(getApplicationContext(),"Pulsado el radiobutton rbtnOpcionB y tie-
ne de estado " + String.valueOf(radioButton.isChecked()),Toast.LENGTH_SHORT).show();
                        break;
                }
            }
        });

    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d03_01_xesteventos_base);

        getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_HIDDEN);  // Para que no
aparezca el teclado virtual
        xestionarEventosRadioGroup();

    }
}
```

- ✓ **Nota**:
  - La diferencia de gestionar los eventos por cada RadioButton, en el que pulsar en otro provocaba dos eventos, aquí solamente se captura el evento de seleccionar un radiobutton.
  - Indicar que si por defecto está seleccionado el radiobutton a nivel gráfico (del layout) esto no provoca la ejecución del evento de click o del cambio de estado.
- ✓ Se queremos que se provoque dicho evento, no debemos de seleccionar ningún radiobutton en el diseño y en el código hacer una llamada al método check(int id) del RadioGroup después de registrar el listener.

### 1.9.3. Eventos mínimos a conocer en el RadioButton / RadioGroup

- ✓ Gestionar el evento de Click sobre cualquier RadioButton y saberlo hacer empleando interfaces anónimas o implementando la interface en la Activity.
- ✓ Gestionar el evento de CheckedChangeListener sobre cualquier RadioButton y saberlo hacer empleando las interfaces anónimas o implementando la interface en la Activity.
- ✓ Gestionar el evento de CheckedChangeListener sobre el RadioGroup y saberlo hacer empleando interfaces anónimas o implementando la interface en la Activity.

```
// Ejemplo de gestión de evento sobre un RadioButton concreto
((RadioButton)findViewById(R.id.rbtnOpcionB)).setOnCheckedChangeListener(new CompoundBut-
ton.OnCheckedChangeListener() { // Referenciamos a un RadioButton do Layout y  asociamos un listener
    @Override
    public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
        Toast.makeText(getApplicationContext(),"Cambiado estado de rbtnOpcionB:" +
b,Toast.LENGTH_SHORT).show();
    }
});


// Ejemplo de gestión de evento sobre un RadioGroup
((RadioGroup)findViewById(R.id.rgrpGrupoBotons)).setOnCheckedChangeListener(new Radio-
Group.OnCheckedChangeListener() {  // Referenciamos a un RadioGroup do Layout e asociamos un listener
    @Override
    public void onCheckedChanged(RadioGroup radioGroup, int i) {
        RadioButton rb = findViewById(i);   // Podemos obtener una referencia al RadioButton dentro
del RadioGroup que cambia de estado
        switch (i){
            case R.id.rbtnOpcionA:
                Toast.makeText(getApplicationContext(),"Cambiado estado Opción A: " +
String.valueOf(rb.isChecked()),Toast.LENGTH_SHORT).show();
                break;
            case R.id.rbtnOpcionB:
                Toast.makeText(getApplicationContext(),"Cambiado estado Opción B: " +
String.valueOf(rb.isChecked()),Toast.LENGTH_SHORT).show();
                break;
        }
    }
});
```

## 1.10. ImageView

- ✓ En el caso de las Image vamos a poder querer gestionar el evento Click sobre la imagen (por ejemplo para lanzar la cámara del dispositivo) o el evento LongClick que se produce cuando mantenemos durante un o dos segundos pulsado el View.
- ✓ Al igual que el OnClickListener, OnLongClickListener es una interface que va a poder estar asociada a cualquier View y por lo tanto se va a poder emplear con cualquiera.

### 1.10.1. Evento Interface OnClickListener

✓ El evento Click se gestiona desde la Interface OnClickListener mediante el método setOnClickListener que se encuentra definido en la clase View y por lo tanto la forma de gestionarlo siempre es la misma.

```
public class UD03_01_xesteventos_base extends AppCompatActivity {

        private void xestionarEventosImageView(){

            findViewById(R.id.imageView).setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    Toast.makeText(getApplicationContext(),"Pulsaste en la ima-
gen",Toast.LENGTH_SHORT).show();
                }
            });

        }
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_u_d03_01_xesteventos_base);

            getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_HIDDEN);  // Para
que no aparezca el teclado virtual
            xestionarEventosImageView();
        }
    }
```

### 1.10.2. Evento Interface OnLongClickListener

✓ Cando sobre una vista se pulsa por duración de 1 segundo o más se lanza el evento **LongClick** que es capturado por el método onLongClick() asociado a la interface OnLongClickListener.

✓ Igual que hicimos con la forma de gestionar un Click sobre un View, ahora deberemos hacer uso del método setOnLongClickListener() o cual permite 'registrar' la interface que gestiona el evento.

Dicho método espera recibir como parámetro un objeto de una clase que implemente la interface OnLongClickListener.

Podemos hacerlo de forma anónima o implementando dicha interface de una clase cualquiera o en la propia Activity.

Al hacerlo deberemos de implementar el método de la interface que en este caso es el método onLongClick().

Como dije antes, el método setOnLongClickListener está definido en la clase View y por lo tanto cualquier View va a poder registrar este evento.

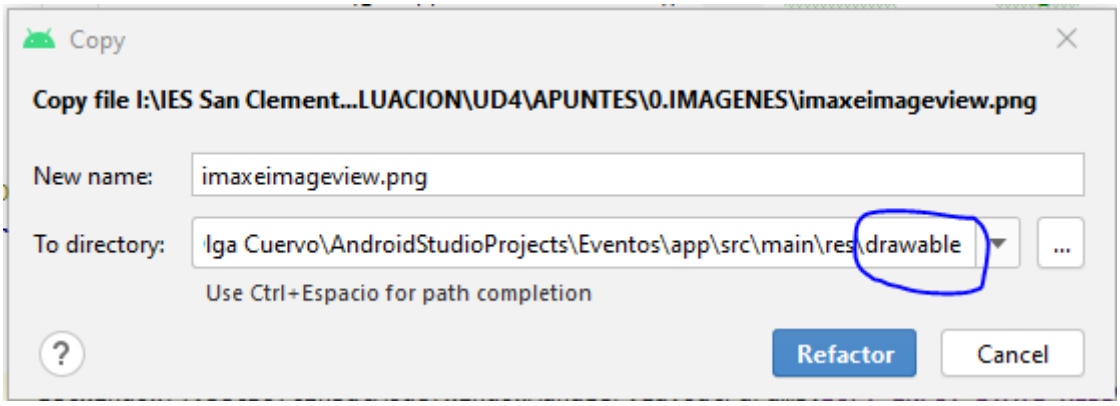
✓ El método onLongClick() devuelve un booleano para poder comprobar si el evento se consumió o no. Se llegamos a pulsar un segundo (o más) o no.

- Devuelve **true** se pudimos capturar el evento y ya no hace nada más.
- Devuelve **false** se no se pudo capturar el evento y continua llamando a otros escuchadores tipo on-Click.

✓ NOTA: Como paso previo a realizar el ejemplo, copia una imagen a /res/drawable/ de nombre **imaxeimageview.png**.

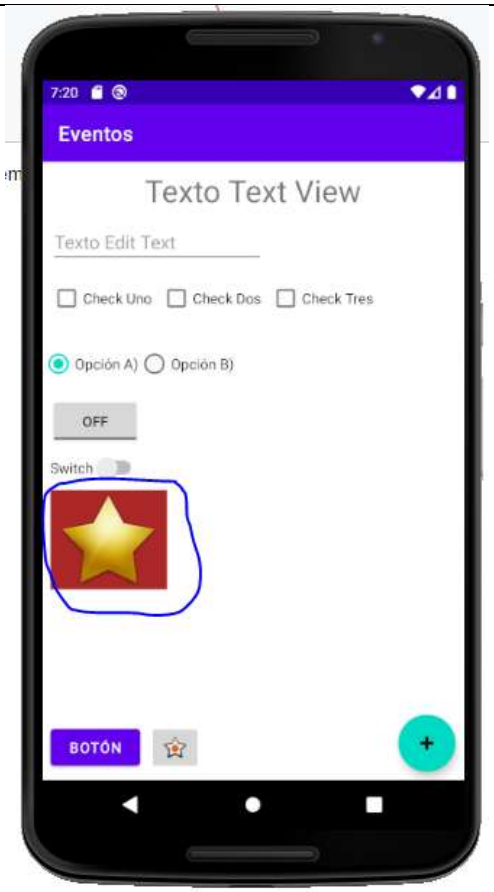
Ya vimos anteriormente como cargar una imagen.

## Evento LongClick




Desde un explorador de archivos arrastramos la imagen a la carpeta **/res/drawable/**. Importante que la mueva a la carpeta drawable, no drawable-v24.


Al pulsar (hacer Click) aparece un mensaje.




Al hacer un LongClick (mantener pulsado en la imagen sin soltar) cambiamos la imagen del ImageView.

## Código da Activity:

```
public class UD03_01_xesteventos_base extends AppCompatActivity {
        private void xestionarEventosImageView(){

            findViewById(R.id.imageView).setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    Toast.makeText(getApplicationContext(),"Pulsaste en la ima-
gen",Toast.LENGTH_SHORT).show();
                }
            });

            findViewById(R.id.imageView).setOnLongClickListener(new View.OnLongClickListener() {
                @Override
                public boolean onLongClick(View view) {
                    ImageView imageButton = (ImageView) view;   // El view que provoco el evento fue el
ImageView
                    imageButton.setImageResource(R.drawable.imaxeimageview);

                    return true;    // No queremos que haga el Click a continuación
                }
            });

        }
        @Override
        protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_u_d03_01_xesteventos_base);

            getWindow().setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_STATE_HIDDEN);  // Para
```

```
que no aparezca el teclado virtual
            xestionarEventosImageView();
        }
    }
```

**Líneas 4-9**: Método onClick. No tiene nada diferente a lo visto.

**Líneas 14-18**: Método onLongClick.

- ✓ Fijarse que dicho método devuelve un boolean. Lo que indica este boolean es si queremos que el evento siga 'propagándose'. En el ejemplo, después de hacer el LongClick, produciría un evento Click.

  Si devolvemos true estamos indicando que ya gestionamos el evento y no queremos que siga propagándose (por eso no hace la llamada al método onClick).

### 1.10.3. Eventos mínimos a conocer en ImageView

- ✓ Gestionar el evento de Click y LongClick sobre cualquier ImageView y saberlo hacer empleando interfaces anónimas o implementando la interface en la Activity.