Tema 4: Interfaces y menús en Android

Qué vamos a ver en este tema…

- Interfaces
- Menús

Interfaces: Vistas

Vistas: Introducción

- Terminología de interfaces gráficas en Android:
  - **Vistas**
    - Equivalentes a widgets o componentes gráficos
    - Subclases de ***View***
  - **Grupos de vistas**
    - Contienen múltiples vistas
    - Subclases de ***ViewGroup***
  - **Layouts**
    - Grupos de vistas utilizados para organizar vistas en la pantalla

Vistas

Asignar una interfaz gráfica a una actividad mediantes recursos

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    TextView miTexto = (TextView)findViewById(R.id.texto);
}
```

- **/res/layout/milayout.xml**
- **setContentView():** asignación de la interfaz a la actividad
  - Parámetro: identificador de recurso u objeto View
- **findViewById():** acceso a las vistas desde el código
  - Parámetro: el atributo android:id de la vista en el archivo de recursos
  - android:id="@+id/[IDENTIFICADOR]"

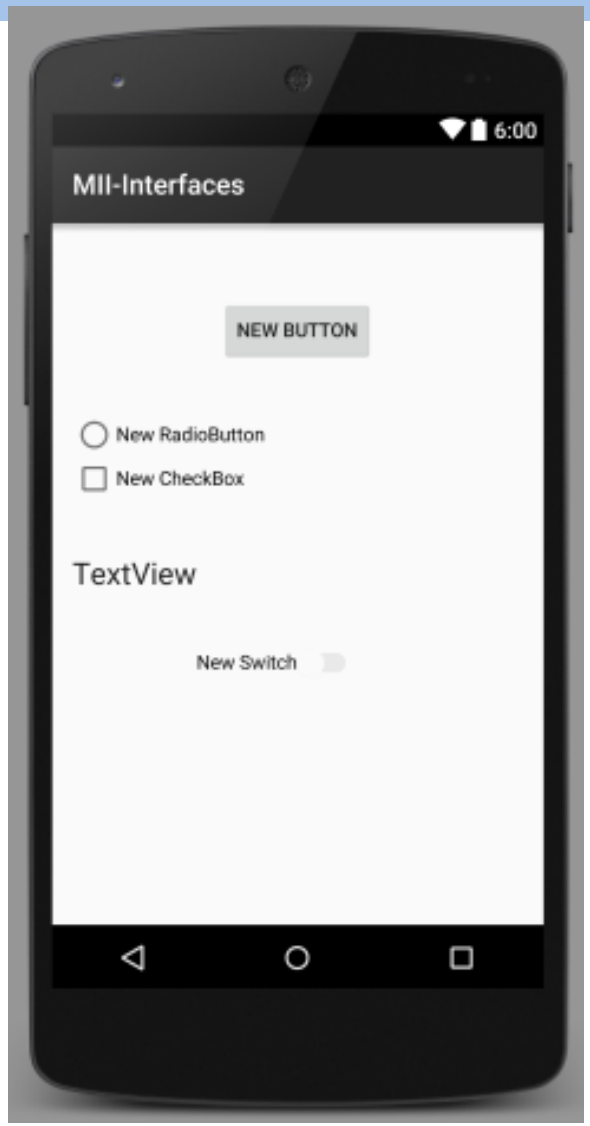
Vistas

Asignar una interfaz gráfica a una actividad mediante código

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    textView texto = new TextView(this);
    setContentView(texto);

    texto.setText("Hola Mundo!");

}
```

Vistas básicas en Android

- TextView
- EditText
- ListView
- Spinner
- Button
- CheckBox
- RadioButton
- SeekBar…



Layouts en XML

Carpeta /res/layout/

```
<?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">
    <TextView
        android:id="@+id/textView1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Introduce un texto"
        />
    <EditText
        android:id="@+id/editText1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Escribe el texto aquí"
        />
</LinearLayout>
```

Layouts desde código

Aunque es posible trabajar con los layouts directamente desde código esto no se suele recomendar

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    LinearLayout ll = new LinearLayout(this);
    ll.setOrientation(LinearLayout.VERTICAL);
    TextView texto = new TextView(this);
    EditText edicion = new EditText(this);
    texto.setText("Introduce un texto");
    edicion.setText("Escribe el texto aquí");
    int lHeight = LinearLayout.LayoutParams.WRAP_CONTENT;
    int lWidth = LinearLayout.LayoutParams.WRAP_CONTENT;
    ll.addView(texto, new LinearLayout.LayoutParams(lHeight, lWidth));
    ll.addView(edicion, new LinearLayout.layoutParams(lHeight, lWidth));
    setContentView(ll);

}
```

Consejos para mejorar el rendimiento de layouts

- La creación de la interfaz es un proceso costoso

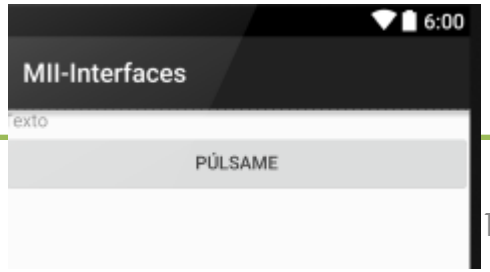
- Consejos para interfaces eficientes:
  - No anidar elementos de forma innecesaria
  - Usar el menor número de elementos posible
  - Huir de los anidamientos profundos

Vistas de Texto

- **TextView**
  - Etiqueta de texto
  - Atributos relevantes
    - android:text
    - android:textColor
    - android:textSize
  - Métodos
    - setText()
    - appendText()
    - getText()
- **EditText**
  - Hereda de TextView, editable

Button

```
<?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">
        <TextView
             android:id="@+id/texto"
             android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:text="Texto"
             />
        <Button
             android:id="@+id/boton"
             android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:text="Púlsame"
             />
   </LinearLayout>
```



Button

- Para definir la acción que ejecutará el botón tenemos dos opciones:
  - Desde código
  - Definiendo el manejador del evento OnClick en el archivo XML de layout

```
<Button
    android:id="@+id/boton"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Púlsame"
    android:onClick="accion"
    />
```

```
public void accion(View vista){
    //Aqui la acción que haga el botón

}
```

Checkbox

- Botón con dos posibles estados
- Independientes unos de otros
- **Atributos**
  - android:checked
  - android:text
- **Métodos**
  - isChecked
  - setChecked

CheckBox desde layouts

```
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android
   "
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <TextView
             android:id="@+id/texto"
             android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:text="Texto"
             />
        <CheckBox
             android:id="@+id/micheckbox"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:text="Púlsame"
             android:checked="false"
             />
   </LinearLayout>
```

CheckBox desde código

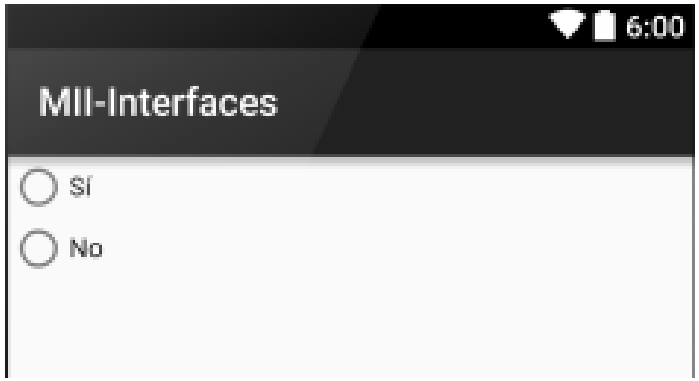
```
texto = (TextView)findViewById(R.id.texto);
miCheckbox = (CheckBox)findViewById(R.id.micheckbox);
miCheckbox.setChecked( true );
miCheckbox.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        // Acción a realizar si está pulsado o no
        if (miCheckbox.isChecked()) {
            texto.setText("activado");
        } else {
            texto.setText("desactivado");
        }
    }
});
```

RadioButton

- Botón con dos estados
- Una vez seleccionado, no puede dejar de estarlo por acción directa del usuario
- Agrupados en elementos de tipo RadioGroup

RadioButton

```
<RadioGroup
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">

<RadioButton android:id="@+id/radio_si"

   android:layout_width="wrap_content"

   android:layout_height="wrap_content"
           android:text="Sí" />

 <RadioButton android:id="@+id/radio_no"

   android:layout_width="wrap_content"

   android:layout_height="wrap_content"
           android:text="No" />
</RadioGroup>
```



RadioButton desde código

```
botonSi = (RadioButton)findViewById(R.id.radio_si);
botonNo = (RadioButton)findViewById(R.id.radio_no);

botonSi.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        valor = true;
    }
});
botonNo.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        valor = false;
    }
});
```

Layouts en Android

Los Layouts básicos de Android se pueden clasificar en:

- **FrameLayout**: coloca todos los elementos en la esquina superior izquierda
- **LinearLayout**: dispone todos los elementos en una única fila o columna
- **TableLayout**: organiza los elementos en forma de tabla con varias filas y columnas
- **RelativeLayout**: layout flexible para la colocación de elementos en posiciones relativas a otros (es el más usado por defecto)
- **Gallery**: lista horizontal de vistas con scroll

http://developer.android.com/guide/topics/ui/layout-objects.html

Linear Layout

Con este layout los elementos se disponen en una fila o columna. Sus dos atributos principales son:

- android:gravity
- android:weight


```
<LinearLayout
xmlns:android="http://schemas.androi
d.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">

    [CONTENIDO DEL LAYOUT]

</LinearLayout>
```

TableLayout

Esta layout organiza los elementos en una rejilla dividida en filas y columnas

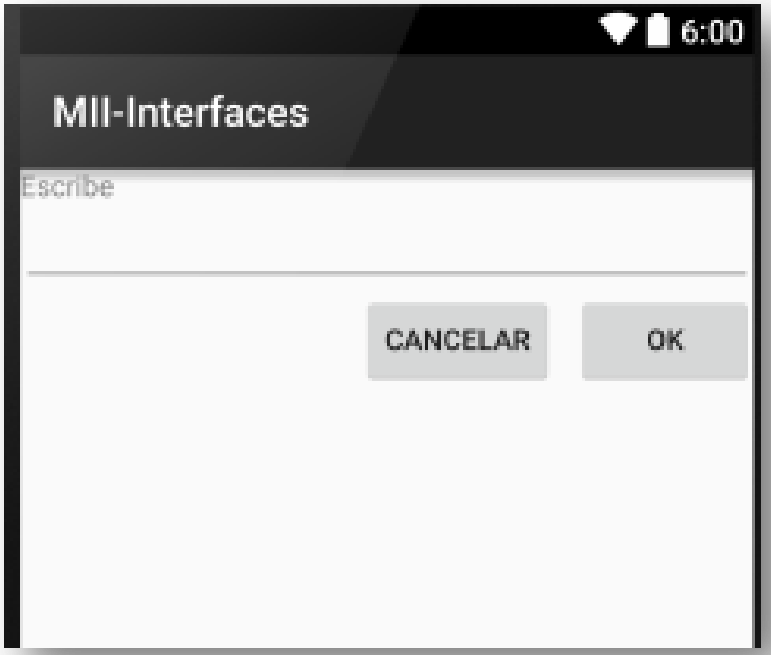
Cada fila es definida por un elemento *TableRow*

- *Cada fila tiene un conjunto de vistas*
- *Cada columna tiene sus propios atributos*

RelativeLayout

- Es el más usado y el mas sencillo ya que permite una gran libertad de diseño

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
    android:layout_width="match_parent"
android:layout_height="match_parent">
    <TextView
        android:id="@+id/etiqueta"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Escribe"/>
    <EditText
        android:id="@+id/entrada"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/etiqueta"/>
    <Button
        android:id="@+id/ok"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/entrada"
        android:layout_alignParentRight="true"
        android:layout_marginLeft="10dip"
        android:text="OK" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/ok"
        android:layout_alignTop="@id/ok"
        android:text="Cancelar" />
</RelativeLayout>
```



Interfaces: Menus

Menús

- Los menús son un componente básico de la mayoría de interfaces de las aplicaciónes.
- En Android podemos encontrar 3 tipos de menús:
  - **Menús de opciones** y de AppBar (Principales)
  - **Menús contextuales**
  - **Popup menús**

Menús

- Al igual que con las vistas, podemos trabajar con los menús desde el código o como recursos.

- Para ello se pueden emplear los métodos:
  - onCreateOptionsMenu
  - onOptionsItemSelected

Menús como recurso

- Archivo XML en /res/menu/
- Cada menú en un fichero separado
- Permite definir diferentes menús según configuración hardware e idioma
- Nombre del fichero: identificador del recurso
  - Elemento raíz *<menu>*
  - Elementos *<item>*
- Submenús por medio de elementos

```
<?xml version="1.0" encoding="utf-8"?>
<menu
xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/new_game"
        android:icon="@drawable/ic_new_game"
        android:title="@string/new_game"
        android:showAsAction="ifRoom"/>
    <item android:id="@+id/help"
        android:icon="@drawable/ic_help"
        android:title="@string/help" />
</menu>
```

Menús como recurso

- Para asociar un menú a una actividad

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
```

Manejando la opción seleccionada

- Para llevar a cabo las acciones pertinentes cuando el usuario seleccione una se puede emplear el método **onOptionsItemSelected.**
- Elemento seleccionado pasado como parámetro (objeto MenuItem)
- Identificarlo mediante getItemId()

```
public boolean onOptionsItemSelected(MenuItem elemento) {
    super.onOptionsItemSelected(elemento);
    // Comprobamos qué elemento del menú fue seleccionado
    switch (elemento.getItemId()) {
        // Comparamos con los identificadores
        case (ELEMENTO_MENU):
            [ ... hacer algo ... ]
            return true;
    }
    // Devolvemos false si no hemos hecho nada con el
    // elemento seleccionado
    return false;
}
```

Menús de opciones

- Este es el tipo de menú donde se deben definir las acciones mas importantes de la app, tales como “buscar” o “Settings”

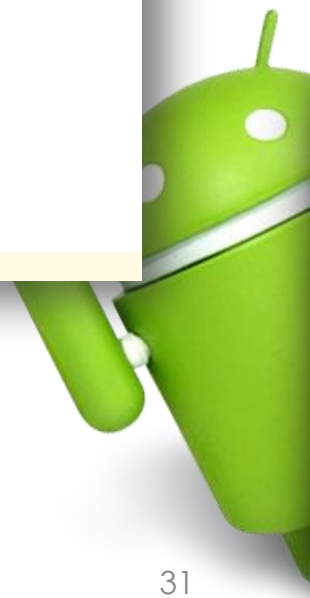
Menús contextuales

- Una Activity puede tener asociada un menú contextual
- El menú se muestra si
  - La activity tiene el foco
  - Se pulsa la pantalla durante tres segundos
- Para definir los menús contextuales:
  - Registrar la activity a la que se le asignará el menú contextual
  - Sobrecargar *onCreateContextMenu*
- Manejador de evento para la selección de un elemento: similar a un menú normal, con onContextItemSelected

Creación de menús contextuales

- Registrar el menú en el elemento View

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    super.onCreate(savedInstanceState);
    EditText texto = new EditText(this);
    setContentView(texto);
    registerForContextMenu(texto);
}
```



Creación de menús contextuales

- Sobrecargar el método onCreateContextMenu

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenu.ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    menu.setHeaderTitle("Menú contextual");
    menu.add(0, menu.FIRST, Menu.NONE, "El1").setIcon(R.drawable.menu_item);
    menu.add(0, menu.FIRST+1, Menu.NONE, "El2").setCheckable(true);
    menu.add(0, menu.FIRST+2, Menu.NONE, "El3").setShortcut('3','3');
    SubMenu sub = menu.addSubMenu("Submenú");
    sub.add("Elemento de submenú");
}
```

Selección de opción en menú contextual

- Tenemos un manejador de eventos propio ***onContextItemSelected***

```
@Override
public boolean onContextItemSelected(MenuItem item)
{
    super.onContextItemSelected(item);
    [ Selección de la opción ]
    return false;
}
```