CFGS DESARROLLO DE APLICACIONES MULTIPLATAFORMA

UD 2. Componentes y distribuciones

Módulo: Programación multimedia y dispositivos móviles

Víctor J. Vergel Rodríguez

Centro de Enseñanza
Gregorio Fernández

Componentes básicos de Android

- **View**

java.lang.Object
↳ android.view.View
↳ android.widget.TextView

java.lang.Object
↳ android.view.View
↳ android.widget.TextView
↳ android.widget.EditText

java.lang.Object
↳ android.view.View
↳ android.widget.TextView
↳ android.widget.Button

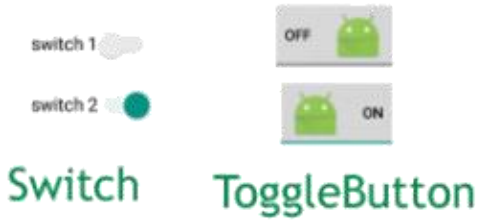
TextView, EditText, Button

```
view.visibility = View.VISIBLE // O View.INVISIBLE, View.GONE
textView.text = "Nuevo texto"
bPulsar.setOnClickListener { b ->
}
view.isEnabled = true
editText.text.clear()  → Limpiar la casilla
editText.inputType = InputType.TYPE_CLASS_TEXT // Para texto general
editText.inputType = InputType.TYPE_CLASS_NUMBER // Para números
editText.inputType = InputType.TYPE_CLASS_TEXT or InputType.TYPE_TEXT_VARIATION_PASSWORD // Contraseñas
editText.inputType = InputType.TYPE_CLASS_NUMBER or InputType.TYPE_NUMBER_VAR  //Contraseñas numéricas
```

Componentes básicos de Android

- **ToggleButton**

```
toggleButton.setOnCheckedChangeListener { _, isChecked ->
    if (isChecked) {        // Código cuando el ToggleButton está activado
    } else {        // Código cuando el ToggleButton está desactivado    }
}
toggleButton.setTextOn("Encendido")
toggleButton.setTextOff("Apagado")

val isChecked = toggleButton.isChecked
```



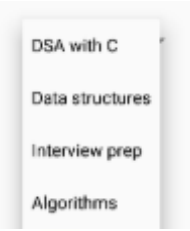
ToggleButton

```
public class ToggleButton
    extends CompoundButton
```

java.lang.Object
↳ android.view.View
↳ android.widget.TextView
↳ android.widget.Button
↳ android.widget.CompoundButton
↳ android.widget.ToggleButton

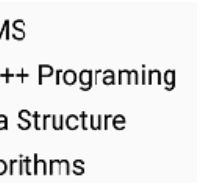
- **Spinner / ListView**

```
spinner.onItemSelectedListener = object : AdapterView.OnItemSelectedListener {
    override fun onItemSelected(parent: AdapterView<*>, view: View, position: Int, id: Long)
}
    override fun onNothingSelected(parent: AdapterView<*>) {
}

val selectedItem = spinner.selectedItem
val selectedItemPosition = spinner.selectedItemPosition
// Crear una lista de elementos
val items = listOf("Elemento 1", "Elemento 2", "Elemento 3")

val adapter = ArrayAdapter(this, android.R.layout.simple_spinner_item, items)
spinner.adapter = adapter
```



Componentes básicos de Android

- **RadioGroup, RadioButton**




```
radioGroup.setOnCheckedChangeListener { group, checkedId ->
    val radioButton: RadioButton = findViewById(checkedId)
    // Código cuando se selecciona un RadioButton
    println("Seleccionado: ${radioButton.text}")
}
```

Strings.xml

- **Caracteres especiales**

  %1$s: Reemplazar una cadena de texto (String)

```
String.format(getString(R.string.saludo), "Usuario")
```

  %2$d: Reemplazar un valor numérico entero (int)

  %3$.2f: Reemplazar un valor numérico decimal (float o double) y controlar la precisión decimal

plurals.xml

- Definimos los posibles valores

```
<resources>
    <plurals name="precios">
        <item quantity="zero">No cuesta</item>
        <item quantity="one">Cuesta un euro</item>
        <item quantity="other">Cuesta %d euros</item>
    </plurals>
</resources>
```

- Asignamos los valores pudiendo meter un valor con %d:

```
var cadenaPersonalizada:String = getResources().getQuantityString(R.plurals.alumnos, listaAlumnos.size,
listaAlumnos.size);

binding.tvContadorAlumno.setText(cadenaPersonalizada);
```

ConstraintLayout

Chains

- **Spread**: la cadena se distribuye uniformemente.
- **Spread-Inside:** La primera y la última vista de la cadena se anclan en los extremos y el resto se distribuye uniformemente.
- **Weighted:** No es un valor en el XML, mediante

```
android:layout_width="0dp"
app:layout_constraintHorizontal_weight="4"
```

- **Packed:** Se agrupan las vistas.