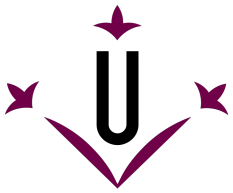


Universitat de Lleida

Primera Activitat: Resources en Android

Aplicacions per a dispositius mòbils - Grau en Enginyería Informàtica

Pablo Fraile Alonso

March 5, 2022

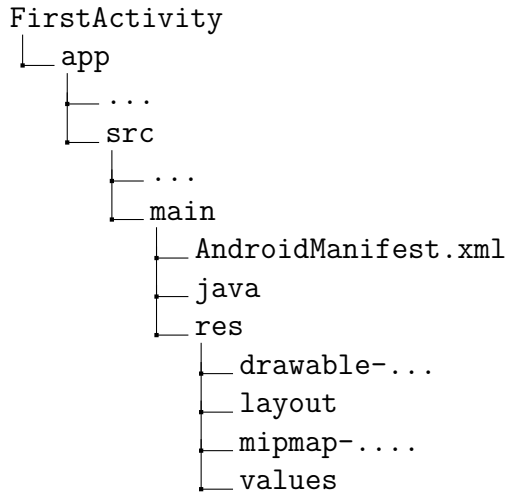
Contents

List of Figures

List of Tables

1 Comprovar si s'aplica la primera bona pràctica

Veiem l'estructura del projecte i en un principi si que compleix la primera bona pràctica, ja que té diferents recursos separats en diferents arxius, tal i com es veu en l'arbre d'arxius:



En cambi, si ens fixem al arxiu de layout, veiem que tot i que tingui creat l'arxiu *strings.xml*, han "hardcodejat" la string "Hello World" dins de la component Textview:

```
<TextView
    android:id="@+id/textView"
    ....
    android:text="Hello World"
    ...
/>
```

Quant realment, si es volgués separar els diferents recursos, s'hauria de cambiar el text a que referencii a les strings localitzades a: res/values/strings.xml.

```
<TextView
    android:id="@+id/textView"
    ....
    android:text="@string/hello_world"
```

```
    ...
/>
```

2 Comprobar si se aplica la segunda buena práctica y añadir recursos alternativos para la app.

La segona pràctica consisteix en "Provide alternative resources to support specific device configurations". Veiem que tot i que Android Studio ens hagi proporcionat diferents resources, aquests únicament venen adaptat per un dispositiu móvil (no tablet), amb layout portrait (vertical) i idioma anglés, per tant es podría dir que no compleix la segona bona pràctica.

```
res
├── drawable
│   ├── image_1.png
│   ├── image_2.png
│   └── image_3.png
├── drawable-ca
│   ├── image_1.png
│   ├── image_2.png
│   └── image_3.png
├── drawable-es
│   ├── image_1.png
│   ├── image_2.png
│   └── image_3.png
├── layout
│   └── activity_main.xml
├── layout-land
│   └── activity_main.xml
├── layout-large
│   └── activity_main.xml
├── layout-large-port
│   └── activity_main.xml
└── mipmap-anydpi-v26
    ├── ic_launcher_round.xml
    └── ic_launcher.xml
```

```
├── mipmap-ca-hdpi
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-ca-mdpi
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-ca-xhdpi
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-ca-xxhdpi
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-ca-xxxhdpi
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-es-hdpi
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-es-mdpi
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-es-xhdpi
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-es-xxhdpi
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-es-xxxhdpi
│   └── ic_launcher_foreground.png
```

```
│
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-hdpi
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-mdpi
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-xhdpi
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-xxhdpi
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-xxxhdpi
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── values
│   ├── colors.xml
│   ├── ic_launcher_background.xml
│   ├── strings.xml
│   └── themes.xml
├── values-ca
│   ├── ic_launcher_background.xml
│   └── strings.xml
├── values-es-rES
│   ├── ic_launcher_background.xml
│   └── strings.xml
└── values-night
    └── themes.xml
```

3 Añadir la funcionalidad de mostrar un mensaje emergente (Toast) en un botón del layout

Se han provado varias opciones:

3.1 Pasar como parametro a la función setOnClickListener una View con un método implementado onClickListener

```
val button = findViewById<Button>(R.id.button)
button.setOnClickListener (
    View.OnClickListener() {
        Toast.makeText(
            applicationContext,
            R.string.toastText,
            Toast.LENGTH_SHORT
        ).show()
    })
}
```

Que en kotlin, con su estilo de lambda, podemos simplificar a:

```
val button = findViewById<Button>(R.id.button)
button.setOnClickListener {
    Toast.makeText(
        applicationContext,
        R.string.toastText,
        Toast.LENGTH_SHORT
    ).show()
}
```

También, podriamos haver creado una inner class Toaster para poder ejecutar el boton:

```
private fun setUpButton() {
    val button = findViewById<Button>(R.id.button)
    button.setOnClickListener (Toaster())
}

private inner class Toaster : View.OnClickListener {
    override fun onClick(p0: View?) {
        Toast.makeText(
            applicationContext,
            R.string.toastText,
            Toast.LENGTH_LONG
        ).show()
    }
}
```

En caso de que la hiciesemos estática, tendriamos un poco más de problemas, ya que deveríamos passar la View por parámetro para poder obtener el contexto:

```
fun setUpButton() {
    val button = findViewById<Button>(R.id.button)
    button.setOnClickListener (Toaster())
}

private class Toaster : View.OnClickListener {
    override fun onClick(p0: View?) {
        if (p0 != null) {
            Toast.makeText(
                p0.context,
                R.string.toastText,
                Toast.LENGTH_SHORT
            ).show()
        }
    }
}
```