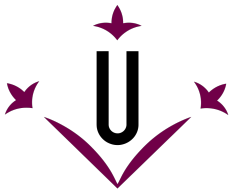

Universitat de Lleida

Primera Activitat: Resources en Android

Aplicacions per a dispositius mòbils - Grau en Enginyeria Informàtica

Pablo Fraile Alonso

6 de març de 2022

Índex

Índex de figures

1 Comprovar si s'aplica la primera bona pràctica

Veiem l'estructura del projecte i en un principi ens dona a entendre que compleix la primera bona pràctica, ja que té diferents recursos separats en diferents fitxers.

A més, l'arxiu MainActivity.kt únicament carrega el layout executant la crida *setContentView*, cosa que ens demostra que la lógica del codi no sap res de com es troba estructurat el layout. A continuació es mostra l'arbre de directoris:

```
FirstActivity
└── app
    ├── ...
    └── src
        ├── ...
        └── main
            ├── AndroidManifest.xml
            ├── java
            └── res
                ├── drawable-...
                ├── layout
                ├── mipmap-....
                └── values
```

En canvi, si ens fixem a l'arxiu de layout, veiem que han "hardcodejat" la string *Hello World* dins de la component TextView (tot i que existeixi *strings.xml*):

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

Quan realment, si es volgués separar els diferents recursos, s'hauria de

canviar el text per a que refereixi a les strings localitzades a: res/values/strings.xml.

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

Per tant, **no** es compleix la primera bona pràctica.

2 Comprovar si s'aplica la segona bona pràctica i afegir recursos alternatius

La segona pràctica consisteix en proveir recursos alternatius per suportar diverses configuracions d'un dispositiu. Veiem que, tot i que Android Studio ens hagi proporcionat diferents resources, aquests únicament venen adaptats per un dispositiu mòbil, amb orientació vertical i idioma anglès.

Per tant, no compleix la segona bona pràctica tot i que ens proporcioni les facilitats per poder afegir compatibilitat.

En el cas de la nostra aplicació, afegirem suport per a:

- Idioma català.
- Idioma castellà.
- Orientació horitzontal per mòbils.
- Dispositius tablet (orientació vertical).
- Dispositius tablet (orientació horitzontal).
- Icones personalitzades depenent de l'idioma.
- Imatges personalitzades depenent de l'idioma.

Per afegir aquestes funcionalitats, s'han hagut de modificar i crear els diferents layouts, icones i arxius de strings. A més, als layouts, s'ha fet que els components referenciïn als identificadors corresponents del recurs que volen emprar.

```
activity_main.xml:

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

<Button
    android:id="@+id/button"
```

```
        ...
        android:text="@string/buttonToast"
        ...
    />

    <ImageView
        android:id="@+id/image_1"
        android:contentDescription="@string/image_1_description"
        app:srcCompat="@drawable/image_1"
        ...
    />
```

Després de les modificacions, el directori de recursos (res) ha quedat de la següent forma:

```
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 (layout "default" -> Mobile + portrait)
│   └── activity_main.xml
├── layout-land (layout Mobile + landscape)
│   └── activity_main.xml
├── layout-large (layout Tablet + landscape)
│   └── activity_main.xml
├── layout-large-port (layout Tablet + portrait)
│   └── activity_main.xml
└── mipmap-anydpi-v26
    ├── ic_launcher_round.xml
    └── ic_launcher.xml
```

```
├── mipmap-ca-hdpi (catalan hdpi icon)
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-ca-mdpi (catalan mdpi icon)
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-ca-xhdpi (catalan xhdpi icon)
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-ca-xxhdpi (catalan xxhdpi icon)
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-ca-xxxhdpi (catalan xxxhdpi icon)
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-es-hdpi (spanish hdpi icon)
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-es-mdpi (spanish mdpi icon)
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-es-xhdpi (spanish xhdpi icon)
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-es-xxhdpi (spanish xxhdpi icon)
│   ├── ic_launcher_foreground.png
│   ├── ic_launcher.png
│   └── ic_launcher_round.png
├── mipmap-es-xxxhdpi (spanish xxxhdpi icon)
│   ├── 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 Afegir la funcionalitat de mostrar un missatge emergent (Toast) en un botó del layout

S'han provat varies opcions per afegir aquesta funcionalitat i que es comentaran en les subseccions següents.

3.1 Emprant anonymous classes/lambdas

S'obté un objecte Button a partir del resource ID definit al layout. A l'objecte button es defineix l'acció de click passant per paràmetre un objecte de tipus OnClickListener, que creem com una classe anònima.

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

Amb Kotlin, emprant el seu estil de lambda, podem simplificar a:

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

3.2 Creant una inner class

Com l'inner class es troba *"lligada"* a la classe de l'Activity, podem accedir al context de l'aplicació directament.

```
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()
    }
}
```

3.3 Creant una classe estàtica

Com en aquest cas és una classe estàtica (per defecte en Kotlin), no podem accedir al context directament ja que no estem lligats a la classe, per tant, accedim al context a partir de la View passada per paràmetre.

```
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()
        }
    }
}
```

3.4 Afegir quina funció s'ha d'executar al xml del layout

En aquest cas, es va emprar l'atribut *onClick* del xml. Aquest no es tant recomanable com els anteriors[1] ja que s'ha d'afegir l'opció a tots els diferents layouts, i no ho tenim localitzat en l'arxiu kotlin de l'activitat.

En els layouts:

```
<Button
    android:id="@+id/button"
    android:text="@string/buttonToast"
    android:onClick="showToast"
    ....
/>
```

En el codi:

```
fun showToast(view: View) =
    Toast.makeText(
        applicationContext,
        R.string.toastText,
        Toast.LENGTH_SHORT
    ).show()
```

[1] De fet, aquesta opció es marca com a deprecated (obsoleta).

4 Imatges del projecte i repositori remot

Com no es poden afegir tots els arxius xml i recursos del projecte en un document pdf, s'ha decidit afegir un link al repositori remot de git, per si el lector/a vol consultar qualsevol arxiu. El repositori es pot trobar aquí.

Figura 1: Aplicació en un mòbil, idioma anglès i orientació vertical

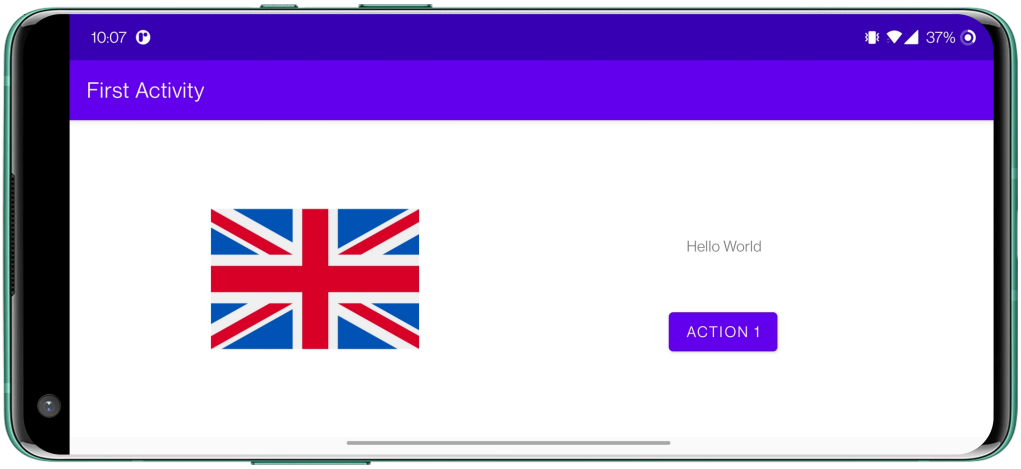

Figura 2: Aplicació en un mòbil, idioma anglès i orientació horitzontal