Tema 2.1 - Usando un proyecto plantilla

Descargar estos apuntes pdf o html

Índice

Introducción

Android Studio es un IDE que se está actualizando constantemente desde Google empezó a adoptar Jetpack Compose como nueva forma de declarar Interfaces de Usuario. Es más, las propias librerías de Compose y Componentes de Material también se actualizan muy a menudo.

Cada versión de Android Studio tiene sus propias plantillas para crear proyectos nuevos. Estas plantillas se actualizan con cada nueva versión de Android Studio. Por lo tanto, esto supone que posiblemente los proyectos básico que se crean en cada versión utilicen versiones diferentes de las librerías de Compose, Material, etc.

Además, las plantillas que traen actualmente compose son muy básicas y no incluyen muchas de las librerías que se suelen utilizar en un proyecto real y el '*scaffolding*' o andamiaje básico de un proyecto a veces es un poco complejo y laborioso de crear.

Existen muchos repositorios en GitHub que contienen proyectos base para empezar a trabajar con Compose siguiendo las buenas prácticas de arquitectura de Android y diferentes interpretaciones de patrones.

De hecho, existe un repositorio oficial de Android en GitHub **https://github.com/android/architecture-templates**. Donde se está trabajando en diferentes plantillas para crear proyectos con diferentes arquitecturas y patrones de diseño que aún está en versión beta y cuya idea es integrarlo en el futuro con Android Studio. Este tipo de proyectos incluyen scripts para renombrar de forma sencilla paquetes, nombres de clases, etc.

Proyecto base

Nosotros hemos incluido un proyecto base que incluye las librerías principales a usar en el curso a modo de plantilla como si fuera una foto de las dependencias y versiones a la fecha **11/09/2024**. Pudes descargar el proyecto base desde el siguiente enlace:

- **Proyecto base 11/09/2024**

> ✋ **Importante**: Lo ideal es que partas de esta proyecto base, copiando la carpeta del '*workspace*' y la renombres con el nombre de tu aplicación. A continuación, puedes seguir los pasos que se indican en el siguiente apartado para renombrar el proyecto base.

Características del proyecto base

> 📌 **Nota**: Este apartado, se incluye a modo divulgativo para ser consultado más adelante en el curso. No es necesario que lo leas ahora pues mucho de lo que pone aún carece de sentido para nosotros

1. Usa **Kotlin** como lenguaje para el DSL de Gradle.
2. Define `libs.versions.toml` en la carpeta `gradle` que contiene las versiones de las librerías que se usan en el proyecto. Este archivo se usa para centralizar las versiones de las librerías y poder referenciarlas en los scripts de Gradle. De esta forma, si queremos cambiar la versión de una librería, solo tendremos que hacerlo en un único lugar. Podemos destacar las siguientes...

```
[versions]
agp = "8.6.0"                   # Android Gradle Plugin usado en el proyecto
kotlin = "2.0.20"               # Versión de Kotlin y el compilador de compose compatible
coreKtx = "1.13.1"              # Extensiones de Kotlin para Android
junit = "4.13.2"                # Librería de testing JUnit
junitVersion = "1.2.1"          # Versión de JUnit para Android
espressoCore = "3.6.1"          # Librería de testing Espresso
lifecycleRuntimeKtx = "2.8.6"   # Librería de ciclo de vida de las Activities
activityCompose = "1.9.2"       # Librería de Compose para las Activities
composeBom = "2024.09.02"       # Bill Of Materials de Compose

pmdmIesBalmisVersion = "24.1"   # Librería de componentes de la asignarura.
```

3. `settings.gradle.kts` donde estará la configuración del proyecto ...
   - Repositorios de donde se van a descargar los plugings de Gradle y las librerías o artefactos de Kotlin, Android. En nuestro caso, usamos el repositorio de Google, Maven Central y JitPack para las librerías que no están en los repositorios oficiales de Google o Maven Central como la librería de componentes de la asignatura.

```
dependencyResolutionManagement {
    repositoriesMode.set(RepositoriesMode.FAIL_ON_PROJECT_REPOS)
    repositories {
        google()
        mavenCentral()
        maven {
            url = uri("https://jitpack.io")
        }
    }
}
```

   - Nombre y módulos que lo forman. En nuestro caso solo tenemos un módulo llamado `app`.

```
rootProject.name = "ProyectoBase2425"
include(":app")
```

4. `build.gradle.kts` del proyecto donde única y exclusivamente se definen los pliguins a usar por Gradle en nuestro proyecto y cuyas versiones están definidas en el archivo `libs.versions.toml` en la sección [plugins]. Como mínimo serán los siguientes:

```
plugins {
    // Aplicamos el plugin de Android  que me añadirá
    // las tareas de compilación, empaquetado, etc.
    alias(libs.plugins.android.application) apply false
    // Aplicamos el plugin de Kotlin para Android
    alias(libs.plugins.kotlin.android) apply false
    // Aplicamos el plugin para el compilador de Compose
    alias(libs.plugins.compose.compiler) apply false
}
```

5. `build.gradle.kts` del módulo de la aplicación. Que incluirá diferentes secciones a destacar:
   i. **plugins**: a aplicar en el módulo de la aplicación, que suelen ser los mismos que para el proyecto más alguno especifico adicional que solo se vaya a utilizar en el módulo de la aplicación. Por ejemplo, el plugin de `kotlin-parcelize` que se usa para generar código de serialización de objetos en Android.

```
plugins {
    alias(libs.plugins.android.application)
    alias(libs.plugins.kotlin.android)
    alias(libs.plugins.compose.compiler)
    id("kotlin-parcelize")
}
```

   ii. **android**: donde se definen las propiedades de la aplicación como el API del SDK de Android a usar, y la mínima versión de Android a la que se va a dirigir la aplicación, el nombre del paquete base de la aplicación, la versión del JDK de Java a usar, etc.

iii. **dependencies**: donde se definen las dependencias de la aplicación. En nuestro caso, librerías que a lo largo del curso irán **tomando sentido** para nosotros, muchas de ellas llevan el prefijo `androidx` indicándonos que son extensiones oficiales de la librería de soporte de Android y puede que en un futuro se incluyan en el propio SDK de Android y cuyas versiones están definidas en el archivo `libs.versions.toml` e incluiremos aquí a través de su alias definido en dicho fichero. Entre ellas podemos destacar:

- **Librerías específicas de Android**:

```
implementation(libs.androidx.core.ktx)
implementation(libs.androidx.lifecycle.runtime.ktx)

// Librería que permite cargar UIs de Compose en Activities
implementation(libs.androidx.activity.compose)

implementation(libs.androidx.ui)
implementation(libs.androidx.ui.graphics)

// Librería que permite previsualizar las UIs de Compose
implementation(libs.androidx.ui.tooling.preview)
```

- **Bill Of Materials (BOM)**: BOM nos permite usar las **librerías de Compose y Material** de forma coherente y que todas las librerías tengan la misma versión. Podemos decir, que es como un conjunto de versiones de librerías que se deben usar juntas.

```
implementation(platform(libs.androidx.compose.bom))
```

Algunas de estas **librerías** son:

```
androidx.compose.foundation:foundation
androidx.compose.runtime:runtime
androidx.compose.ui:ui
androidx.compose.animation:animation
```

Nosotros podremos especificar actualizaciones personalizadas de estas librerías en el archivo `libs.versions.toml`. Pero hay que tener cuidado por que el BOM ya incluye las versiones de las librerías de Compose y Material que se deben usar juntas por compatibilidad.

- **Librerías de Material Design**: Define los componentes de Material Design y aspectos de diseño que se pueden usar en Compose.

```
implementation(libs.androidx.material3)
```

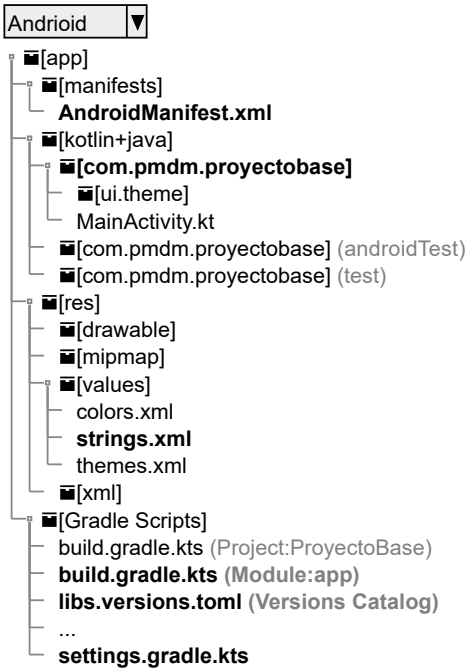
Renombrando un proyecto base

Una vez hemos descargado el proyecto base y lo hemos descomprimido.

Los pasos a realizar para renombrar correctamente el proyecto son los siguientes:

1. Copiamos el proyecto base en el lugar que deseemos.
2. Cambiamos el **nombre a la carpeta** o *'workspace'* del proyecto base, si lo consideramos necesario.Por ejemplo, `HolaMundo` 📁.
3. Abrimos **Android Studio** y seleccionamos la carpeta que contiene nuestro proyecto base.

4. Cuando Gradle finalice todo seleccionaremos la vista `Android`. En esta más que carpeta lo que contendrá una serie de unidades organizativas jerárquicas de los elementos del proyecto similar al esquema de la derecha.
   renombraremos el paquete de la aplicación, para ello haremos **click derecho** sobre el paquete `com.pmdm.proyectobase` y seleccionaremos `Refactor -> Rename` o `Ctrl + R, R` si usamos el Keymap de Visual Studio.
   - Seleccionaremos `All Directories`
   - Cambiaremos `proyectobase` por el nombre de nuestra app en minúsculas, sin espacios y sin caracteres especiales. Por ejemplo, `holamundo` y pulsaremos `Refactor`. Si nos ofrece un preview de los cambios pulsaremos `Do Refactor`.



5. Abriremos el archivo gradle del módulo de la aplicación que se encuentra en `app → Gradle Scripts → build.gradle.kts (Module:app)`
   Cambiaremos el nombre del **paquete** de nuestro programa principal `com.pmdm.proyectobase` por el que hayamos puesto en el paso 4. En nuetro ejemplo `com.pmdm.holamundo`. Fíjate en el siguiente script de ejemplo.

```
android {
    namespace = "com.pmdm.holamundo"
    compileSdk = 34

    defaultConfig {
        applicationId = "com.pmdm.holamundo"
        minSdk = 28
        targetSdk = 34
        versionCode = 1
        versionName = "1.0"
...
```

> ✋ **Importante**: Puesto que hemos cambiado un archivo de la configuración de Gradle deberemos pulsar el botón de sincronización de Gradle que aparece en la barra de herramientas arriba a la derecha con el **icono del elefante de Gradle** para que los cambios tengas efecto.

6. Aunque los paquetes se hayan renombrado, todavía quedaría por realizar algún paso más. Si abrimos el fichero `app → manifests → AndroidManifest.xml` todavía aparece el nombre `ProyectoBase` en la etiqueta y en el `Theme`. Tendremos pues que hacer lo siguiente:

```
<application ...>
    ...
    <activity
        // Si se produce un error aquí deberemos sincornizar el
        // proyecto con Gradle o poner la ruta completa a la
        // definición, esto es, android:name="com.pmdm.holamundo.MainActivity"
        android:name=".MainActivity"

        android:exported="true"
        android:label="@string/app_name"

        // Renombraremos el ProyectoBase a HolaMundo con
        // la refactorización del menú contextual o Ctrl + R, R
        android:theme="@style/Theme.HolaMundo">
        ...
    </activity>
```

7. Cambiaremos el nombre de la APP en `app → res → values → strings.xml`. Por ejemplo, por `HolaMundo`.

```
<resources>
    <string name="app_name">HolaMundo</string>
</resources>
```

8. Abriremos al archivo de configuración de Gradle `app → Gradle Scripts → settings.gradle.kts` y cambiaremos el nombre del proyecto `ProyectoBase` por el nuevo nombre que le queramos dar. Por ejemplo, `HolaMundo`.

```
...
rootProject.name = "HolaMundo"
include(":app")
```

> 📌 **Nota**: Este será el nombre del proyecto que aparecerá al abrir el proyecto con Android Studio independientemente del nombre que le hayamos puesto a la carpeta del *'Workspace'*.

9. Además del nombre del **tema de la MainActivity** que hemos cambiado en el *'manifest'*, deberemos cambiar el nombre del **tema de compose**. Para ello iremos `app → java → com.pmdm.holamundo.ui.theme → Theme.kt` y cambiaremos el nombre del tema `ProyectoBaseTheme` por `HolaMundoTheme` para seguir la nomenclatura de nuestro ejemplo.

> 📌 ***Nota***: Recuerda que es importante hacer un refactorización del nombre del tema con `Ctrl + R, R` para que se cambie en todos los lugares donde se usa.

```
@Composable
fun HolaMundoTheme(
    darkTheme: Boolean = isSystemInDarkTheme(),
    dynamicColor: Boolean = true,
    content: @Composable() () -> Unit
) {
    ...
}
```

10. Por último, sincronizaremos una vez más el proyecto con la opción de menú `File → Sync Project with Gradle Files` o `Ctrl + Mayús + O`. También podemos pulsar el botón de sincronización de Gradle que aparece en la barra de herramientas arriba a la derecha con el **icono del elefante de Gradle**.

Ampliando la plantilla del proyecto base

Durante el curso iremos aplicando la arquitectura de una aplicación de Android propuesta por Google y que veremos en el siguiente tema. Para implementarla iremos añadiendo una organización de paquetes, clases, nombres y diferentes librerías según vayamos avanzando en el curso.

Por tanto, a medida que vayamos avanzando en el curso podemos ir actualizando la plantilla o esqueleto de proyecto base, con las nuevas librerías y organización de paquetes y clases que vayamos viendo y así tener un '*Template*' personalizado para iniciar un proyecto de forma rápida.