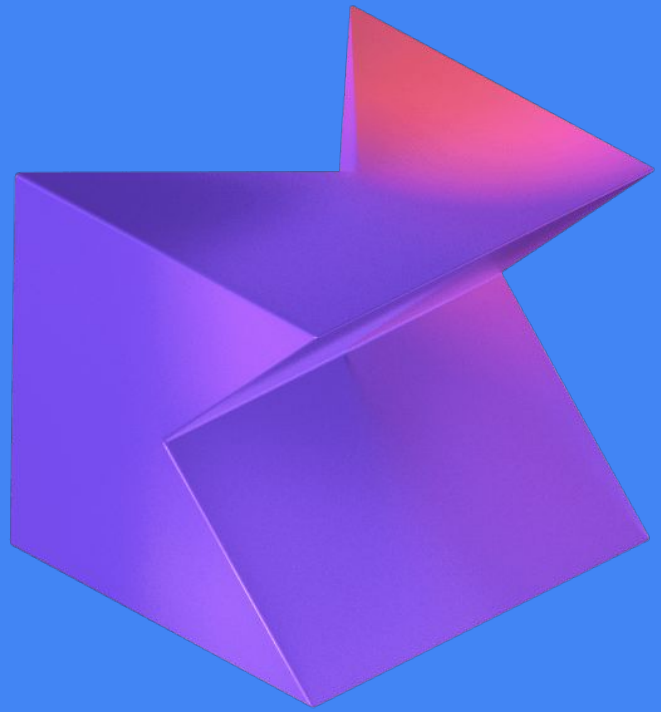
Universidad Nacional de La Matanza

Intro a KMM

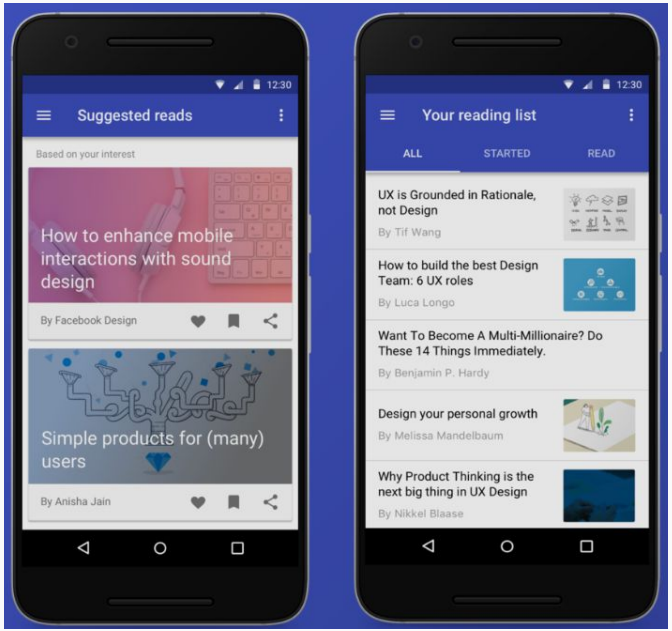


Kotlin Multiplatform Mobile

*"Comparte la **lógica** de tus aplicaciones iOS y Android mientras mantienes la UX nativa"*

Kotlin Multiplatform Mobile

android

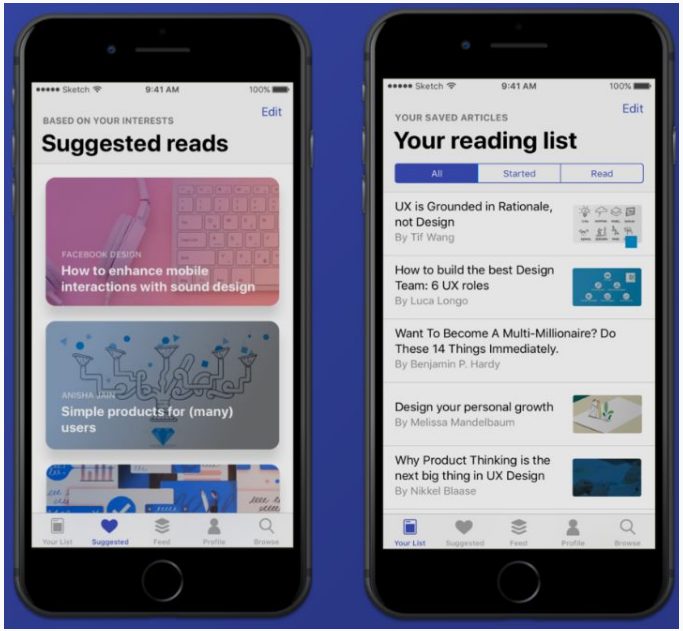


=

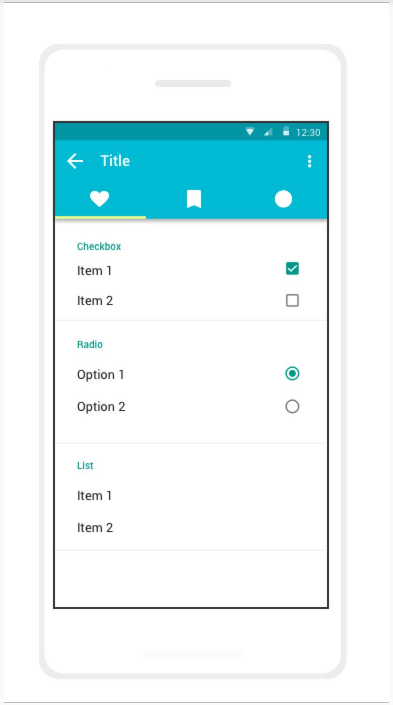
Lógica de negocio
(código compartido)

≠

Interfaz de Usuario
(código no compartido)



Kotlin Multiplatform Mobile

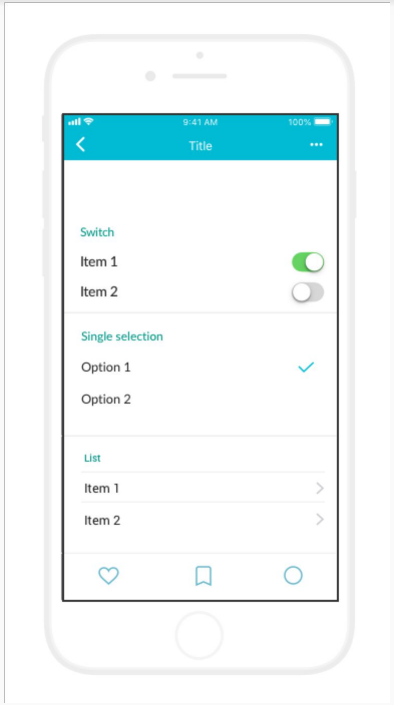


=

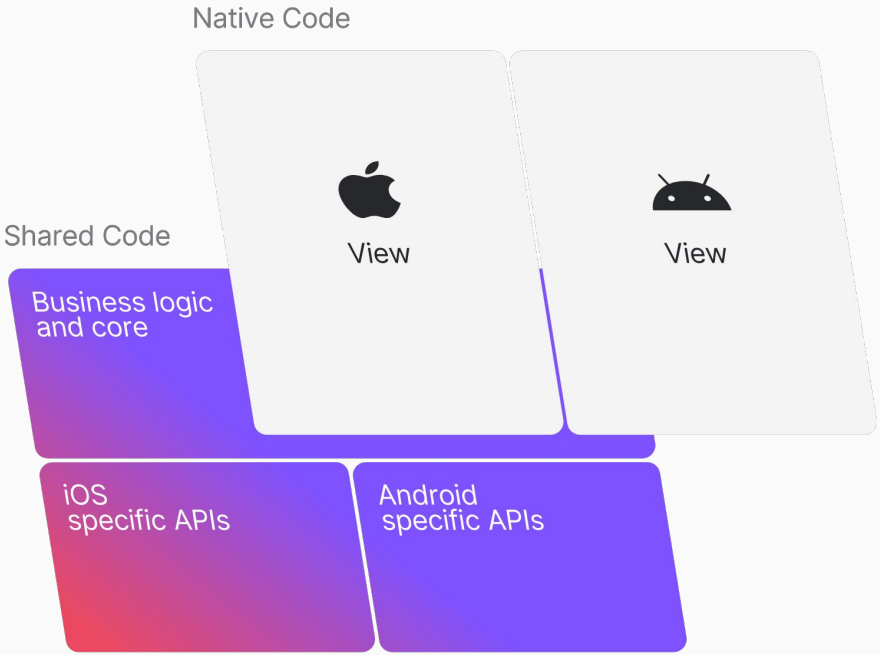
Lógica de negocio
(código compartido)

≠

Interfaz de Usuario
(código no compartido)



Kotlin Multiplatform Mobile



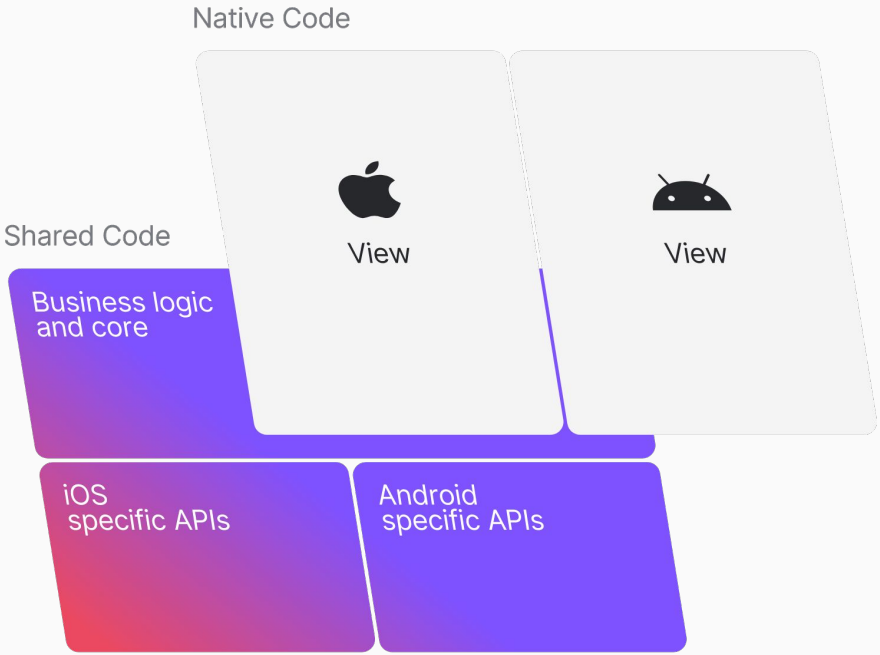
✅ ¿Qué comprende el código compartido?

- Lógica de negocio: Funcionalidades que no dependen del SO. Ej: calcular total de una compra con descuentos, agregar items a favoritos, enviar un mensaje de chat, etc.
- Networking, almacenamiento de datos, analytics

Opcionalmente, tenemos la posibilidad de acceder a APIs específicas de la plataforma

Código 100% Kotlin!

Kotlin Multiplatform Mobile



❌ ¿Qué no se comparte?

- Código "nativo": La interfaz de usuario se implementa en cada plataforma: las pantallas, botones, navegación entre pantallas, animaciones, etc.

  Se usa Kotlin en Android y Swift en iOS

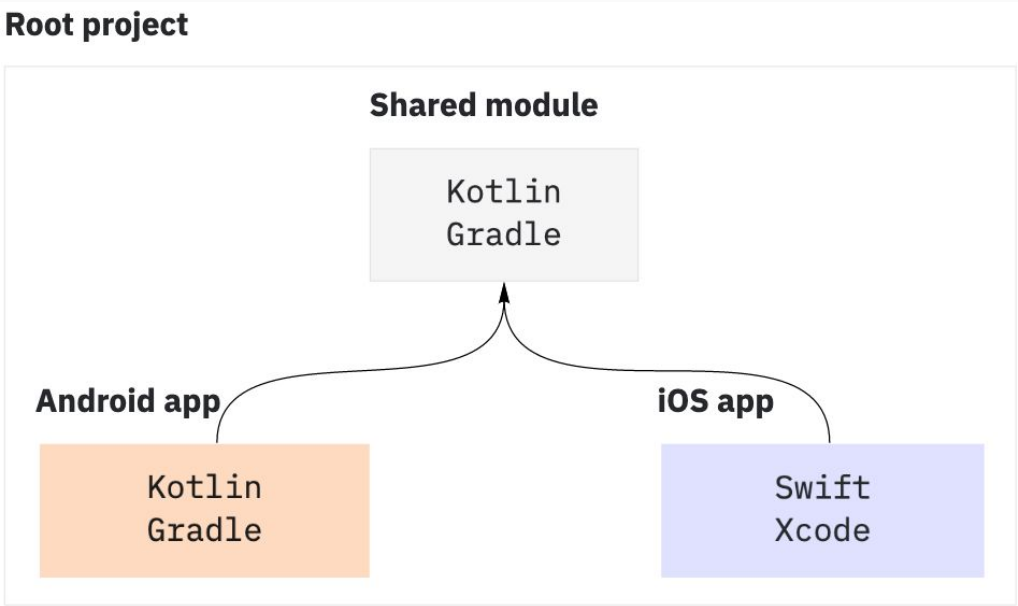
¿Qué herramientas vamos a usar?

Android Studio 4.2 o superior (preferentemente Chipmunk 2021.2.1 Patch 2)

Plugin Kotlin Multiplatform Mobile (0.3.3, usa plugin Kotlin 212 - 1.7.10)

[Opcional] Mac con macOS y XCode 11.3 (o superior) para iOS

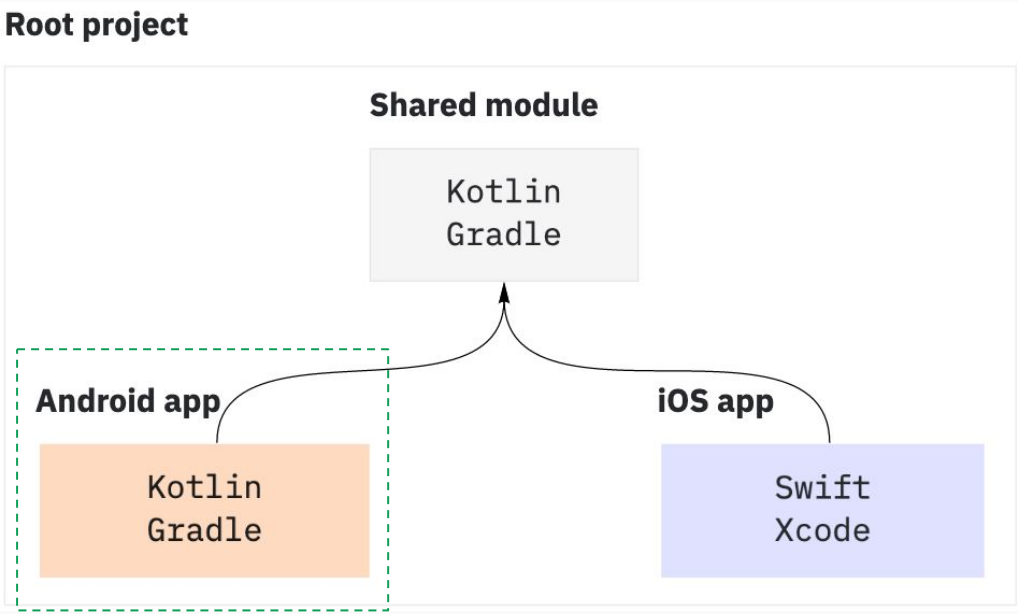
Estructura de un proyecto KMM



Al crear un nuevo proyecto KMM, nos vamos a encontrar con 3 módulos:

- Compartido (Shared module)
- Android App
- iOS App

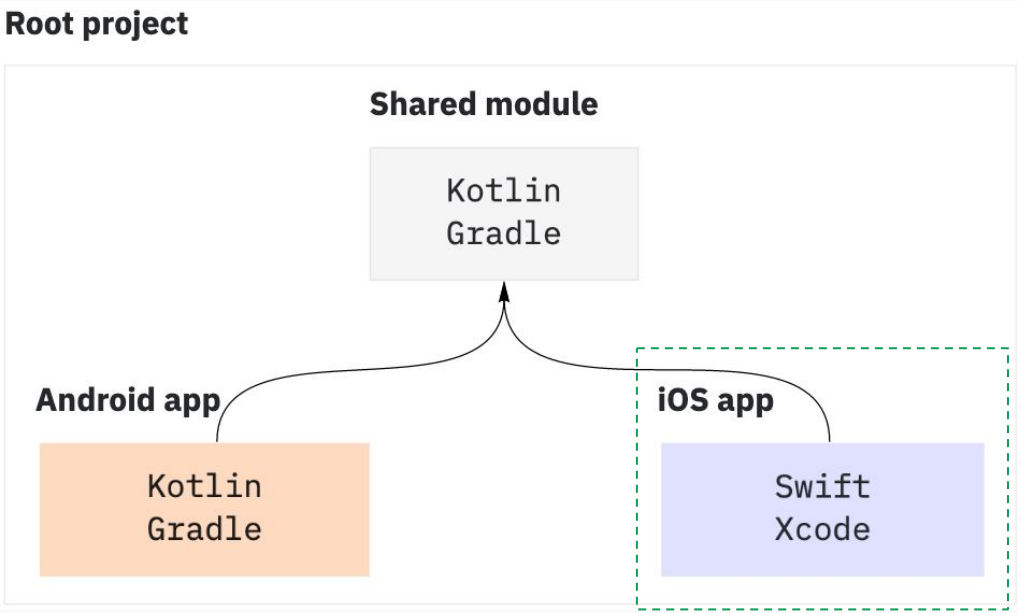
Estructura de un proyecto KMM



Android app:

- Contiene la interfaz gráfica y al compilarlo genera la **aplicación Android**
- Incluye al módulo compartido como dependencia, para poder utilizar su código
- Se programa utilizando Kotlin y se compila mediante Gradle

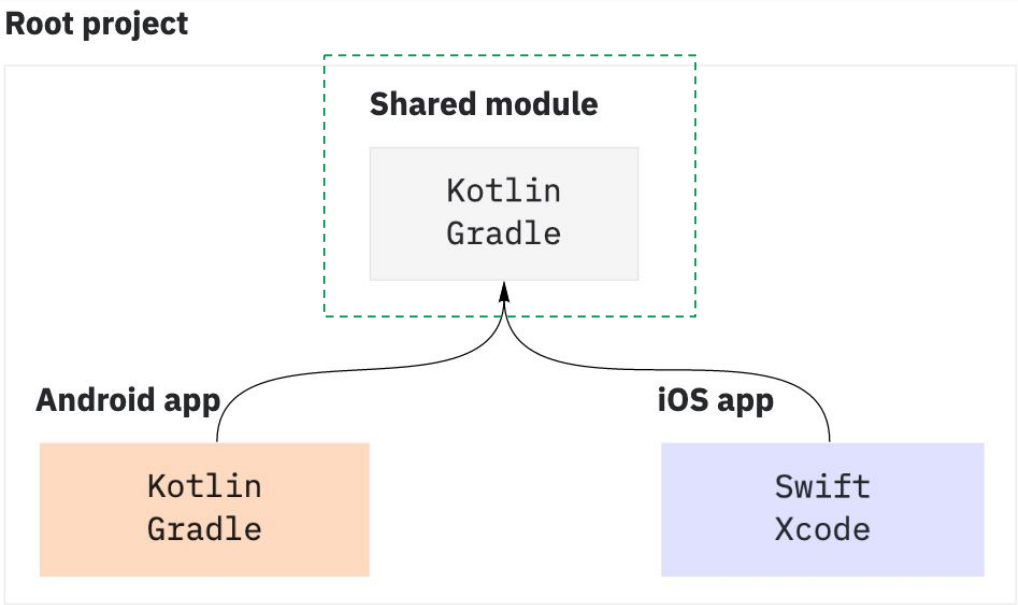
Estructura de un proyecto KMM



iOS app:

- Contiene la interfaz gráfica y al compilarlo genera la **aplicación iOS**
- Incluye al módulo compartido como dependencia, para poder utilizar su código
- Se programa utilizando Swift y se compila mediante XCode

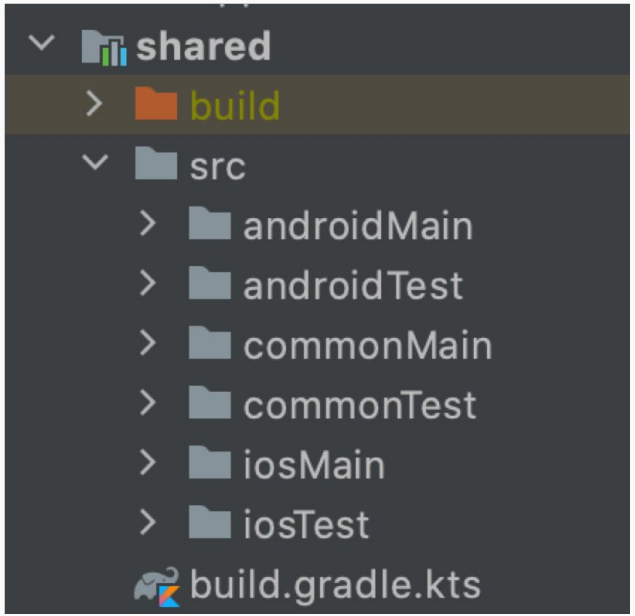
Estructura de un proyecto KMM



Shared module:

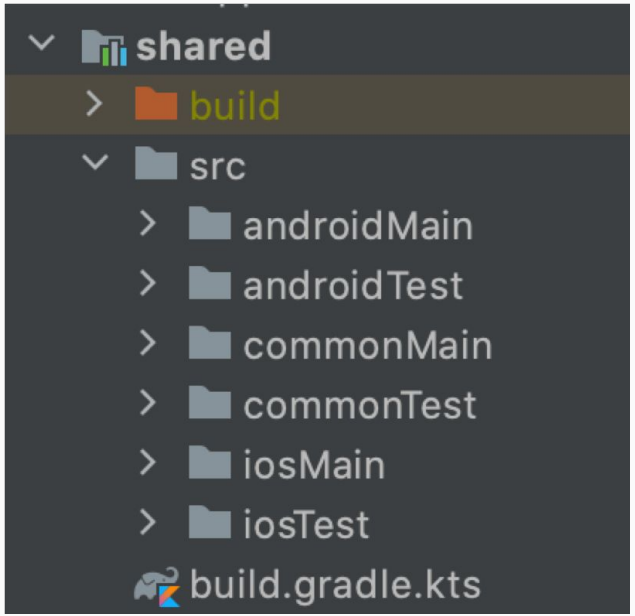
- Es el módulo multiplataforma, que contiene la lógica de negocio en común.
- Se programa en Kotlin y se compila utilizando Gradle
- Genera un **.jar** para Android y un **.framework** para iOS

Módulo compartido



- Escrito en Kotlin
- Genera un artefacto para cada plataforma (*.jar* para Android y *.framework* para iOS)
- Puede tener acceso a APIs de cada plataforma mediante el mecanismo **expect - actual**
- Permite usar librerías multiplataforma, como Serialization, Ktor, SQLDelight, Multiplatform Settings...

Módulo compartido



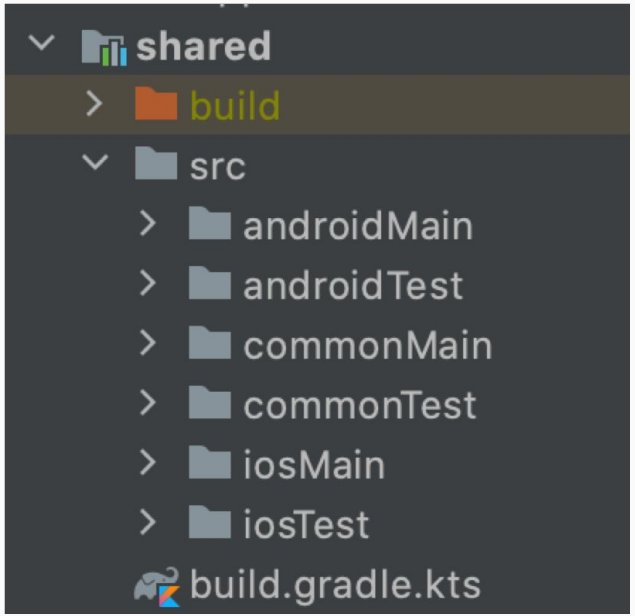
SourceSets
Son los distintos directorios donde podemos escribir el código multiplataforma.

Su nomenclatura es la siguiente: [*plataforma*][*tipo*]

Los valores posibles son la combinación de:
Plataforma: common, android, ios
Tipo: main, test

Ej: androidTest, iosMain, commonMain

Módulo compartido



build.gradle.kts

- Es el archivo de configuración del módulo compartido
- Entre otras cosas, en él declaramos las **dependencias** (librerías externas) de las plataformas

Práctica

- Instalación de Android Studio y plugin de KMM
- Creación de un proyecto KMM
- Compilar aplicación para Android y iOS
- Recorrido por estructura de archivos
- Módulo compartido: sourceSets y build.gradle

Bibliografía

- Página oficial de KMM
- Documentación oficial KMM
- Librerías KMM