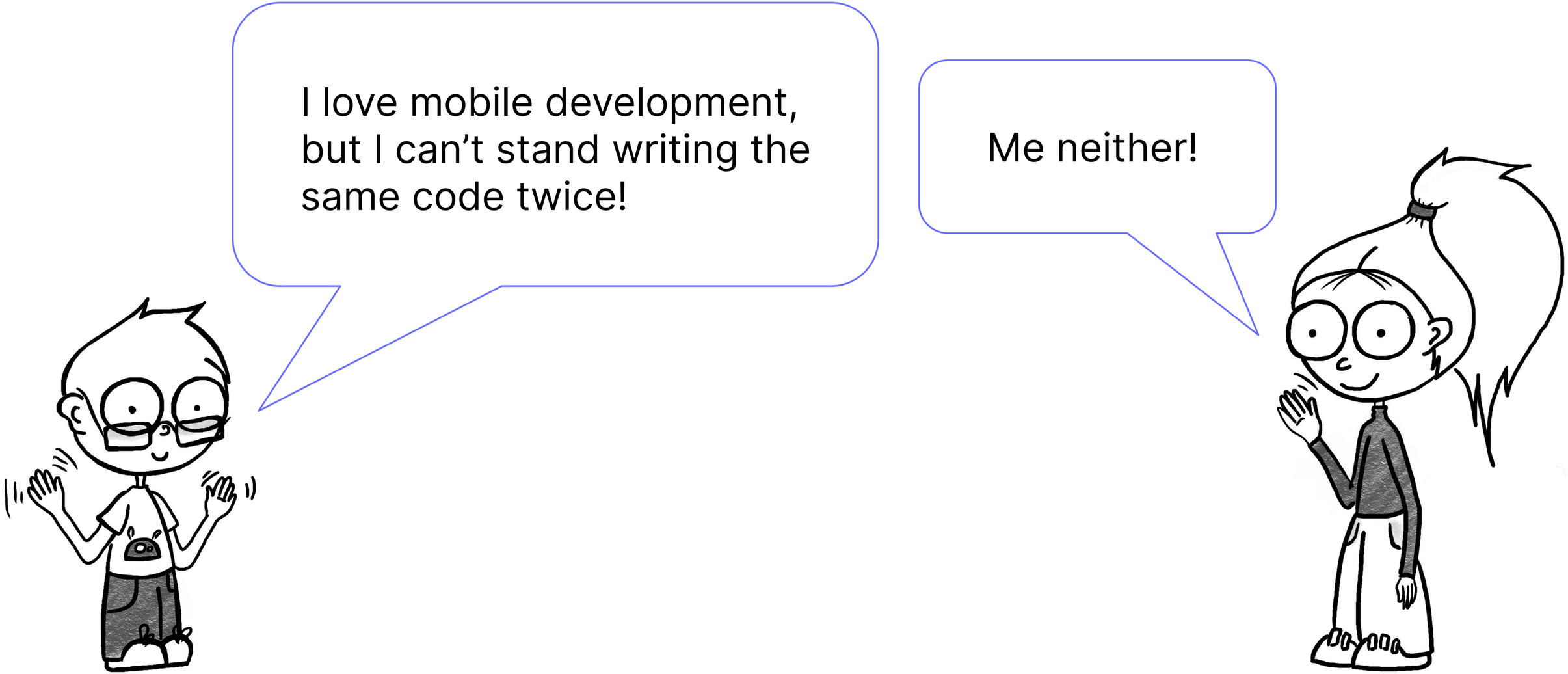
Kotlin 1.4 Online Event

It's time for Kotlin Multiplatform Mobile!

Ekaterina Petrova

@KathrinPetrova

October 14, 2020

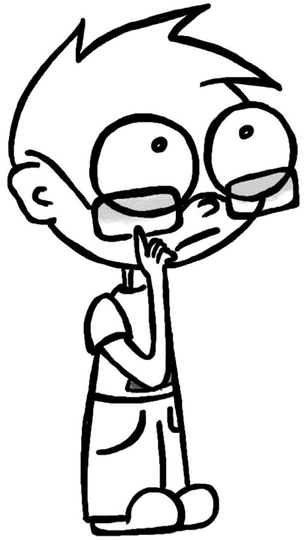

I love mobile development, but I can’t stand writing the same code twice!
Me neither!

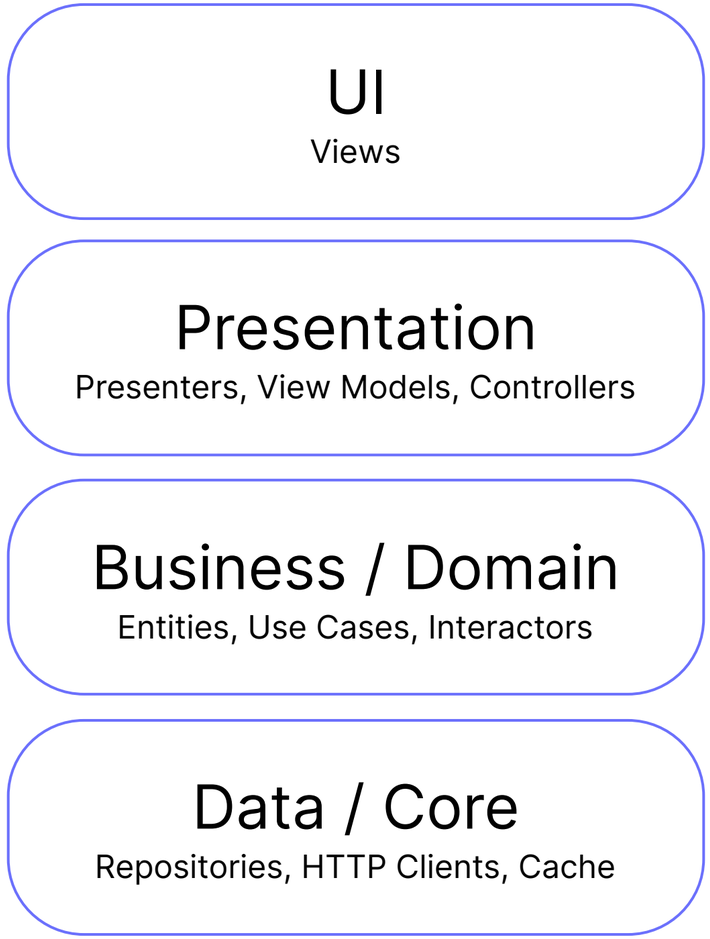
mobile cross platform frameworks

- mobile cross platform frameworks
- mobile cross platform frameworks **comparison**
- **best** mobile cross platform **framework 2019**
- cross platform mobile **development** frameworks **comparison**
- cross platform mobile **app development** frameworks
- **top** cross platform mobile **development** frameworks
- **best** cross platform mobile **development** frameworks

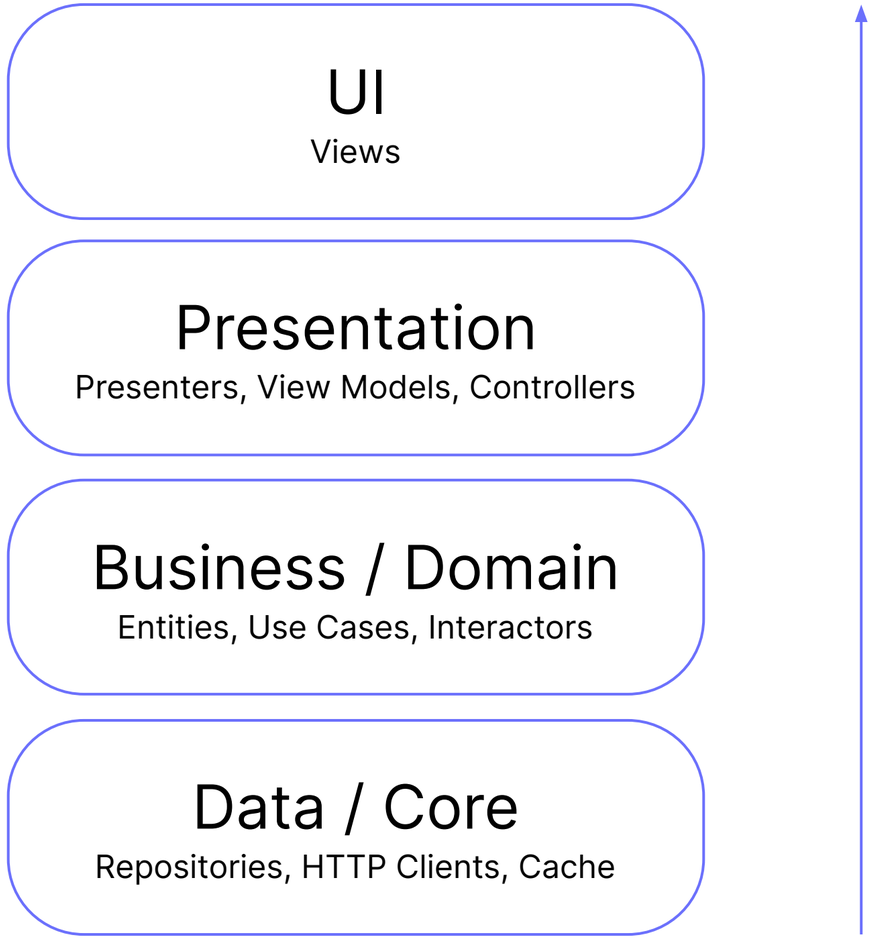
Google Search | I'm Feeling Lucky

Report inappropriate predictions




UI
Views
Presentation
Presenters, View Models, Controllers
Business / Domain
Entities, Use Cases, Interactors
Data / Core
Repositories, HTTP Clients, Cache

🖼️ UI centric



⚙️ Core centric

🖼️ UI centric

UI
Views

Presentation
Presenters, View Models, Controllers

Business / Domain
Entities, Use Cases, Interactors

Data / Core
Repositories, HTTP Clients, Cache

React Native

Flutter

Xamarin Forms

⚙️ Core centric

🖼️ UI centric

UI
Views

Presentation
Presenters, View Models, Controllers

Business / Domain
Entities, Use Cases, Interactors

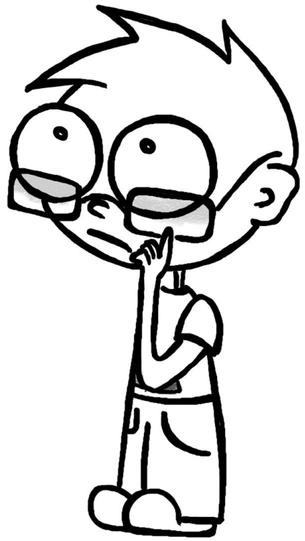
Data / Core
Repositories, HTTP Clients, Cache

React Native

Flutter

Xamarin Forms

How can I share business logic and core code?



⚙️ Core centric

🖼️ UI centric

UI
Views

Presentation
Presenters, View Models, Controllers

Business / Domain
Entities, Use Cases, Interactors

Data / Core
Repositories, HTTP Clients, Cache

React Native

Flutter

Xamarin Forms

🔥 for thin clients, simple apps, MVP

⚙️ Core centric

🏞️ UI centric

UI
Views

Presentation
Presenters, View Models, Controllers

Business / Domain
Entities, Use Cases, Interactors

Data / Core
Repositories, HTTP Clients, Cache

React Native

Flutter

Xamarin Forms

🔥 for thin clients,
simple apps, MVP

🔥 for complex apps,
Fat clients,
strict requirements
for the UI quality

⚙️ Core centric

🏞️ UI centric

UI
Views

Presentation
Presenters, View Models, Controllers

Business / Domain
Entities, Use Cases, Interactors

Data / Core
Repositories, HTTP Clients, Cache

React Native

Flutter

Xamarin Forms

Xamarin Native

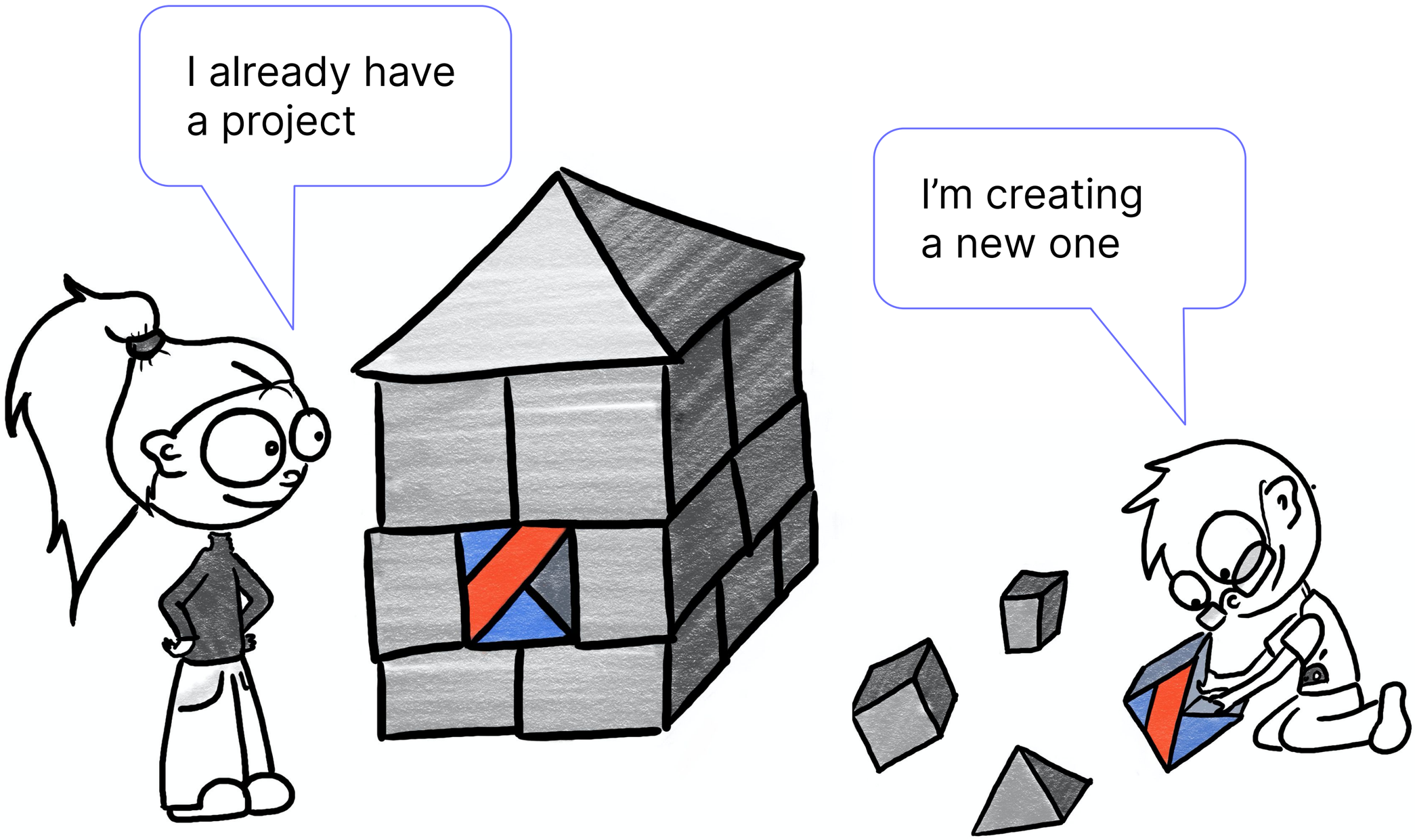
JavaScript
+ JavaScriptCore

C++
+ wrappers

🔥 for thin clients, simple apps, MVP

🔥 for complex apps, Fat clients, strict requirements for the UI quality

⚙️ Core centric


I already have a project
I'm creating a new one

🖼️ UI centric

UI
Views

Presentation
Presenters, View Models, Controllers

Business / Domain
Entities, Use Cases, Interactors

Data / Core
Repositories, HTTP Clients, Cache

Kotlin
Multiplatform
Mobile

🔥 for thin clients,
simple apps, MVP

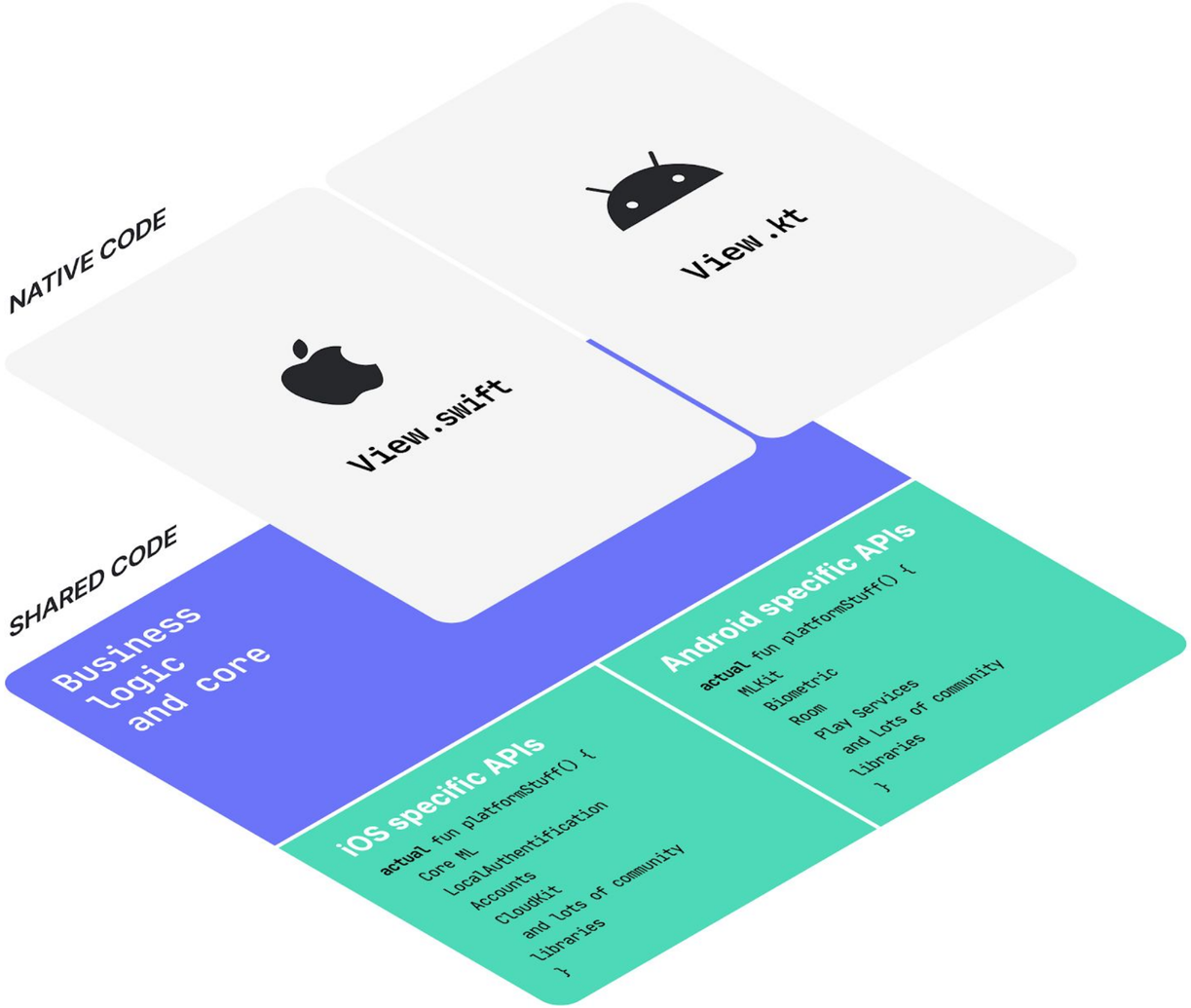
🔥 for complex apps,
Fat clients,
strict requirements
for the UI quality

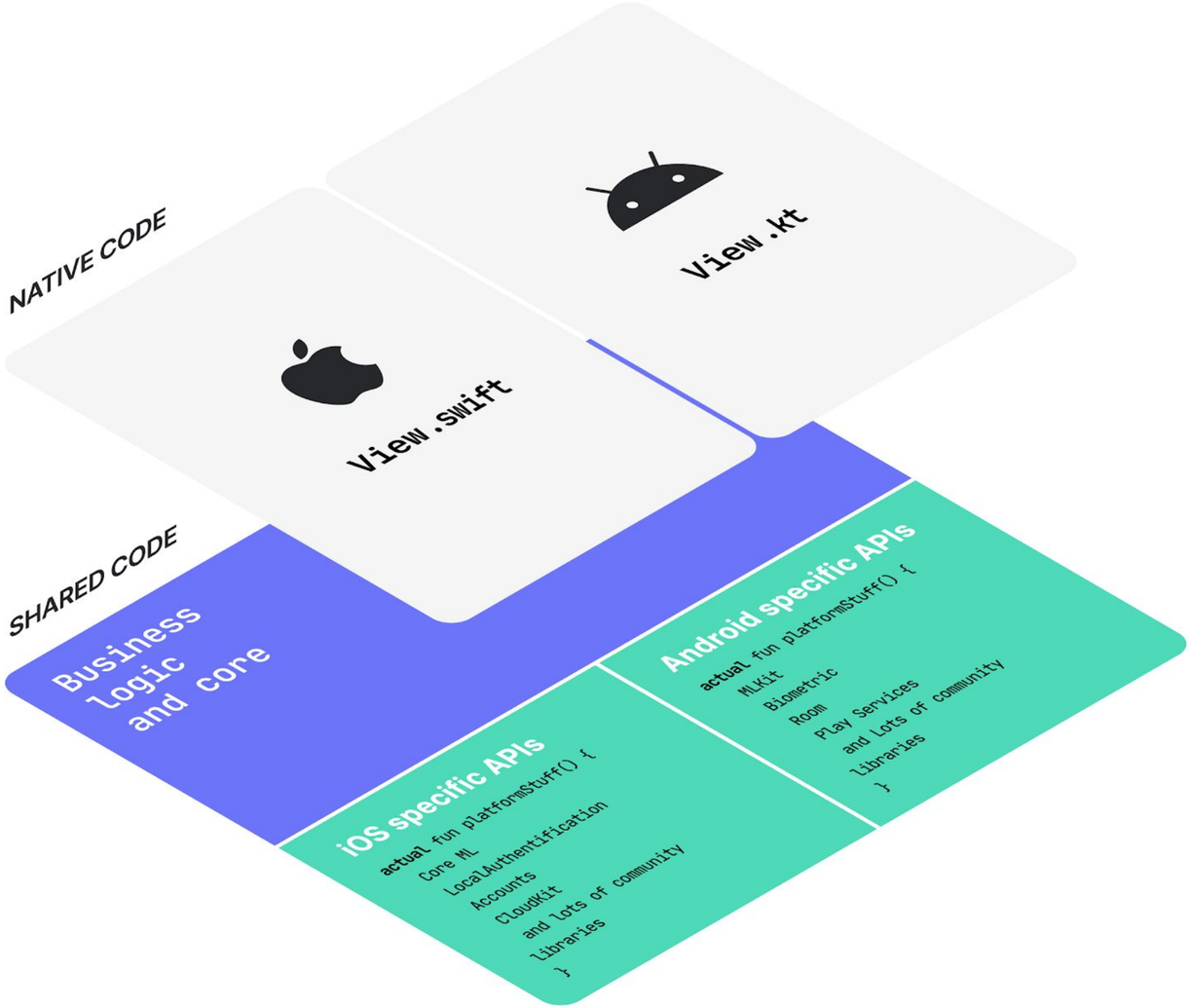
⚙️ Core centric

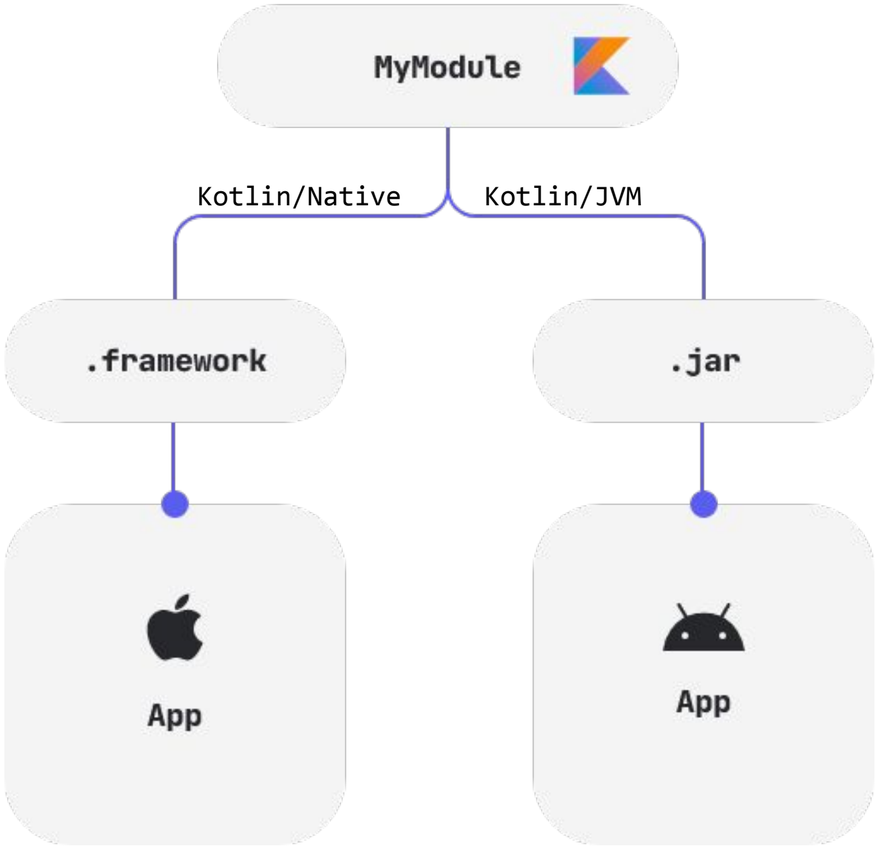
What is Kotlin Multiplatform Mobile?

KMM =

⚙️ Kotlin Multiplatform + 📱 Mobile Features


NATIVE CODE
View.swift
View.kt
SHARED CODE
Business logic and core
iOS specific APIs
actual fun platformStuff() {
Core ML
LocalAuthentification
Accounts
CloudKit
and lots of community libraries
}
Android specific APIs
actual fun platformStuff() {
MLKit
Biometric
Room
Play Services
and Lots of community libraries
}


NATIVE CODE
View.swift
View.kt
SHARED CODE
Business logic and core
iOS specific APIs
actual fun platformStuff() {
Core ML
LocalAuthentification
Accounts
CloudKit
and lots of community libraries
}
Android specific APIs
actual fun platformStuff() {
MLKit
Biometric
Room
Play Services
and Lots of community libraries
}


MyModule
Kotlin/Native
Kotlin/JVM
.framework
.jar
App
App

KMM =

⚙️ Kotlin Multiplatform + 📱 Mobile Features

Multiplatform Gradle DSL

Kotlin/Native +

Kotlin/JVM +

Kotlin +

CocoaPods integration +

Android Studio Plugin +

...

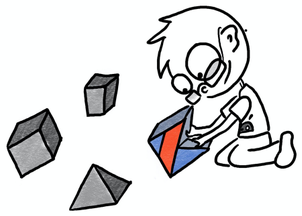
Creating a mobile application with KMM

Step by step

Going cross-platform with KMM

✅ New project

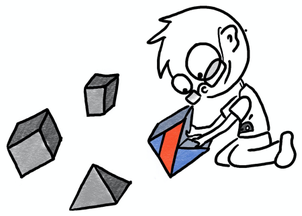
🤔 Existing project

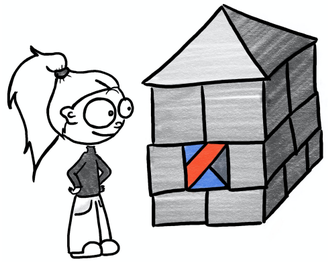


Going cross-platform with KMM

✅ New project

🤔 Existing project

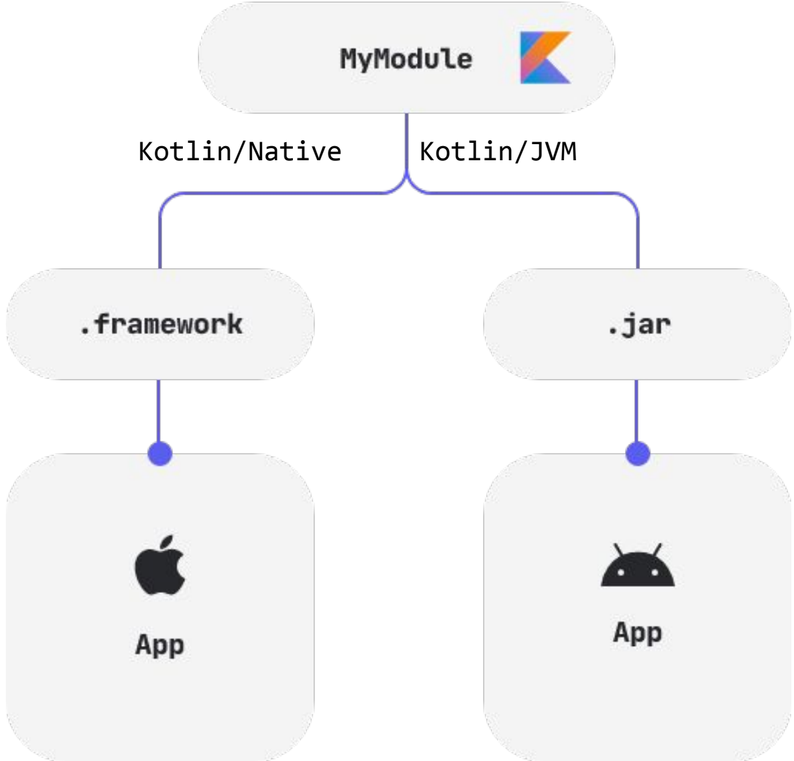


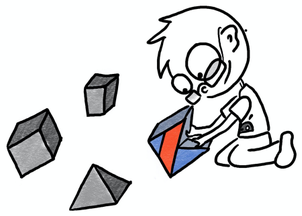


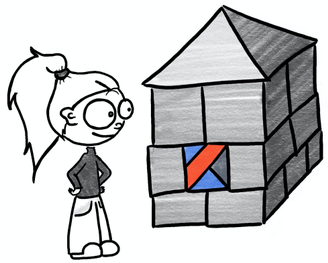
Going cross-platform with KMM

✔️ New project

✔️ Existing project



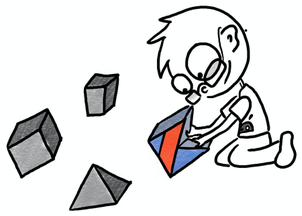




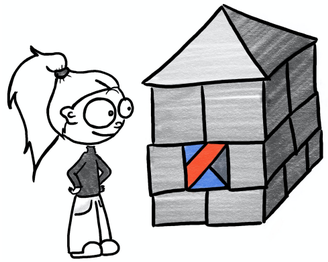
Going cross-platform with KMM

✅ New project

✅ Existing project



Connect `.framework` to iOS project and `.jar` to Android project



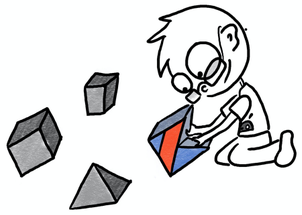
Going cross-platform with KMM

✔️ New project

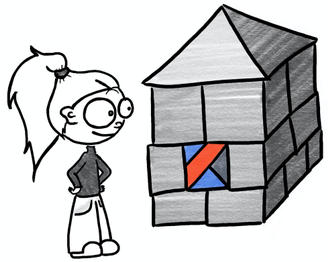
✔️ Existing project

Write shared code in KMM Module

Implement native UI in Android/iOS projects



Connect `.framework` to iOS project and `.jar` to Android project



Going cross-platform with KMM

✅ New project

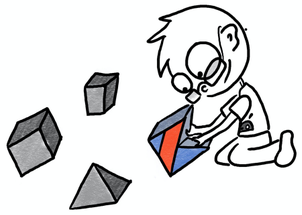
Write shared code in KMM Module

Implement native UI in Android/iOS projects

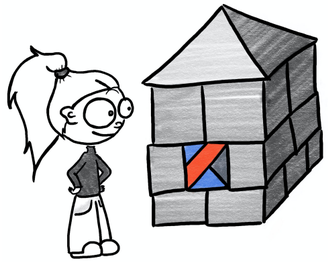
✅ Existing project

Take code written in Kotlin for Android

Refactor it to be compatible with iOS



Connect `.framework` to iOS project and `.jar` to Android project



Going cross-platform with KMM

✅ New project

✅ Existing project

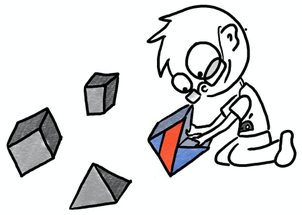
Take a new feature

Write shared code in KMM Module

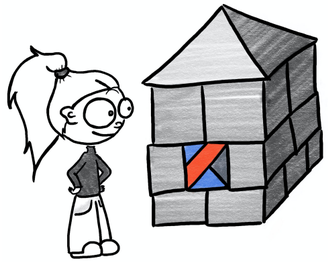
Take code written in Kotlin for Android

Implement native UI in Android/iOS projects

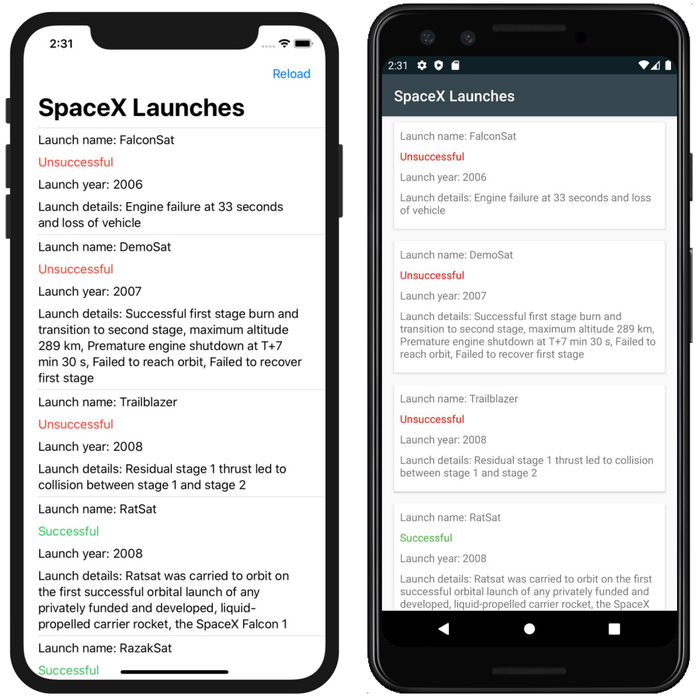
Refactor it to be compatible with iOS

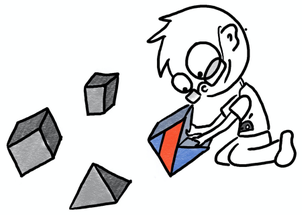


Connect `.framework` to iOS project and `.jar` to Android project



✅ Networking
✅ Cache
✅ Native UI





What to share?

KMM App Architecture

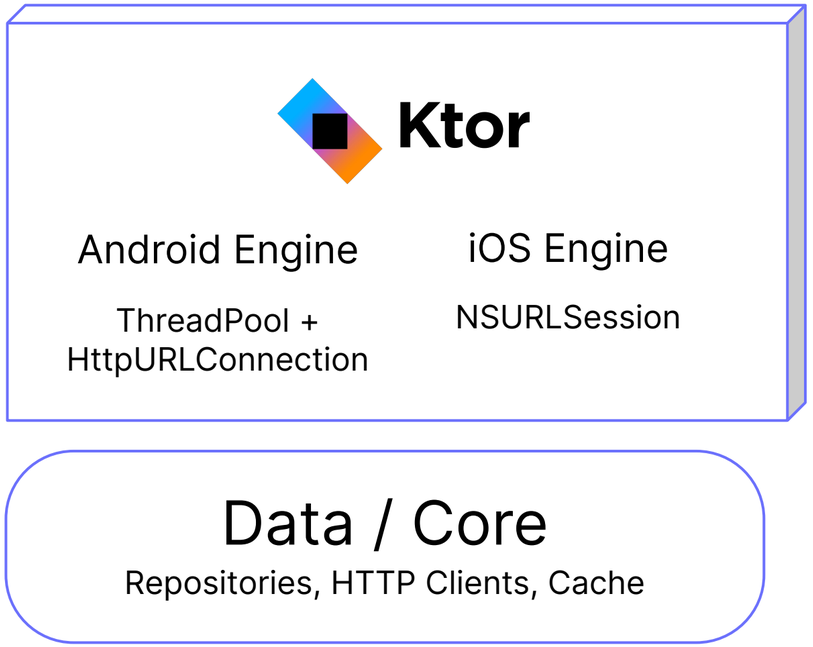
What to share?

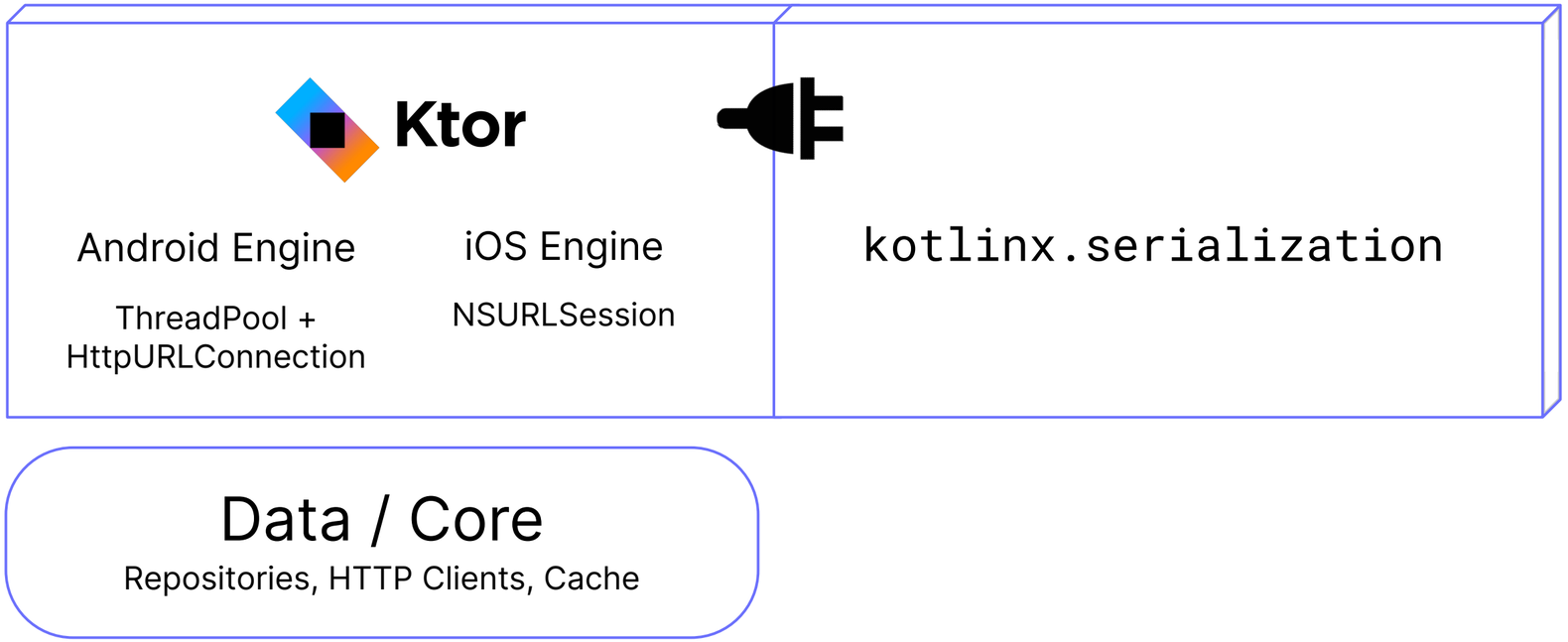
UI
Views

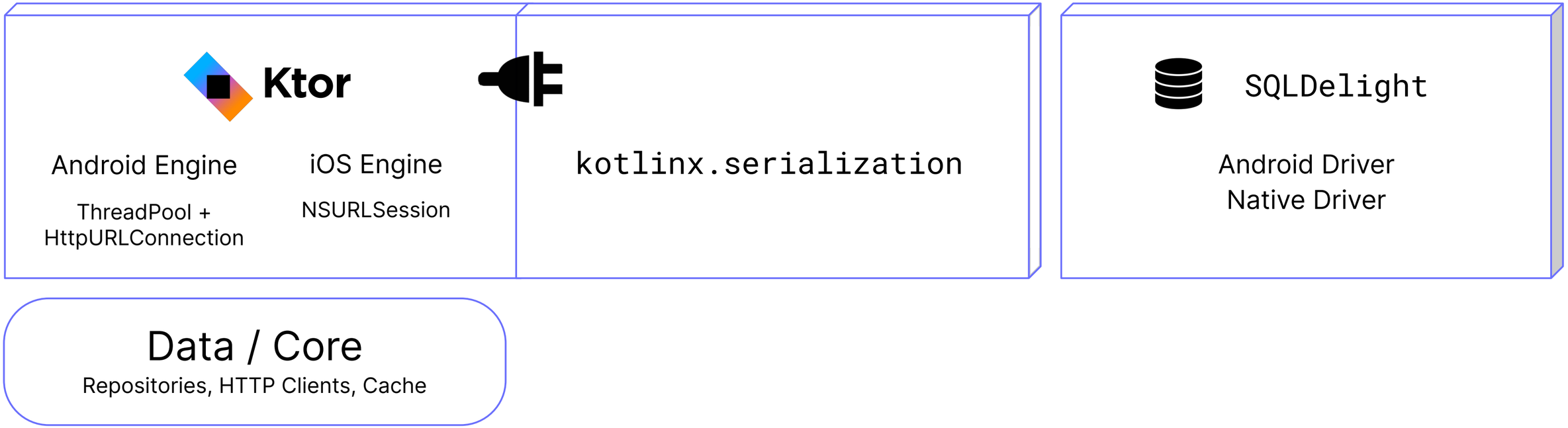
Presentation
Presenters, View Models, Controllers

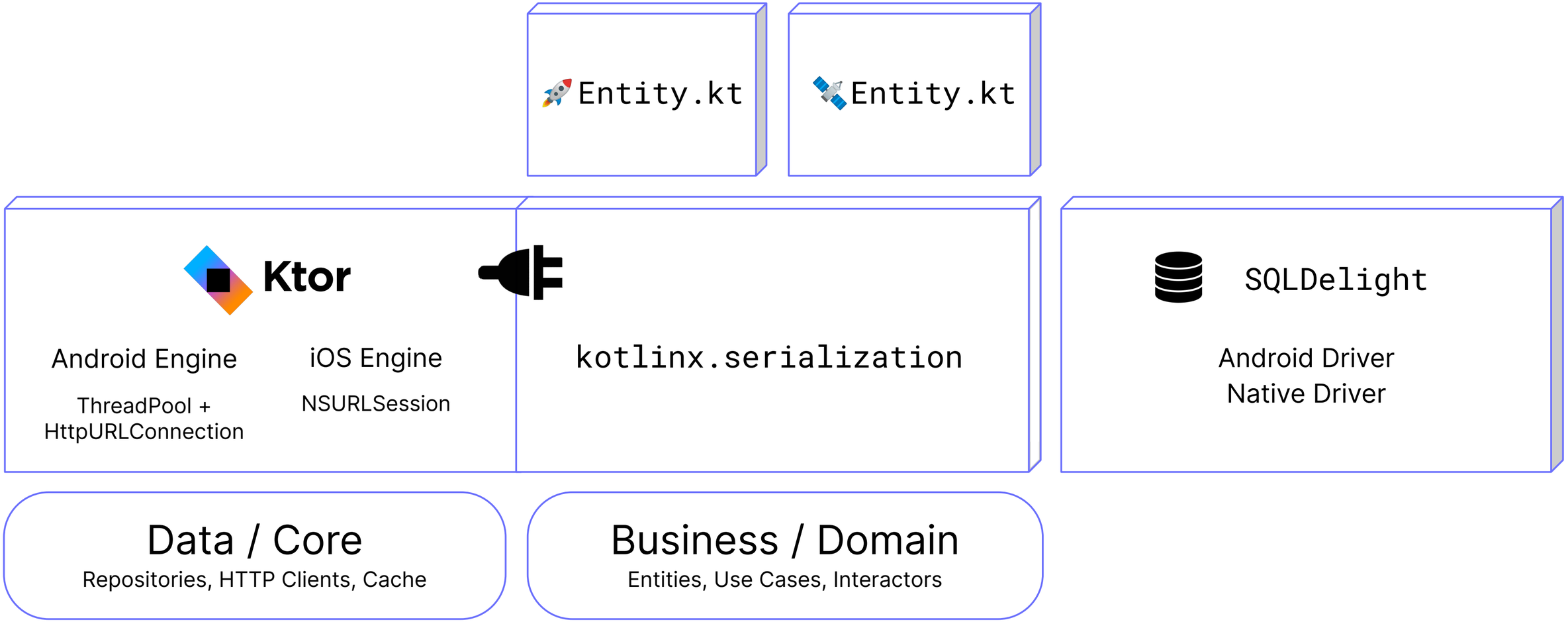
Business / Domain
Entities, Use Cases, Interactors

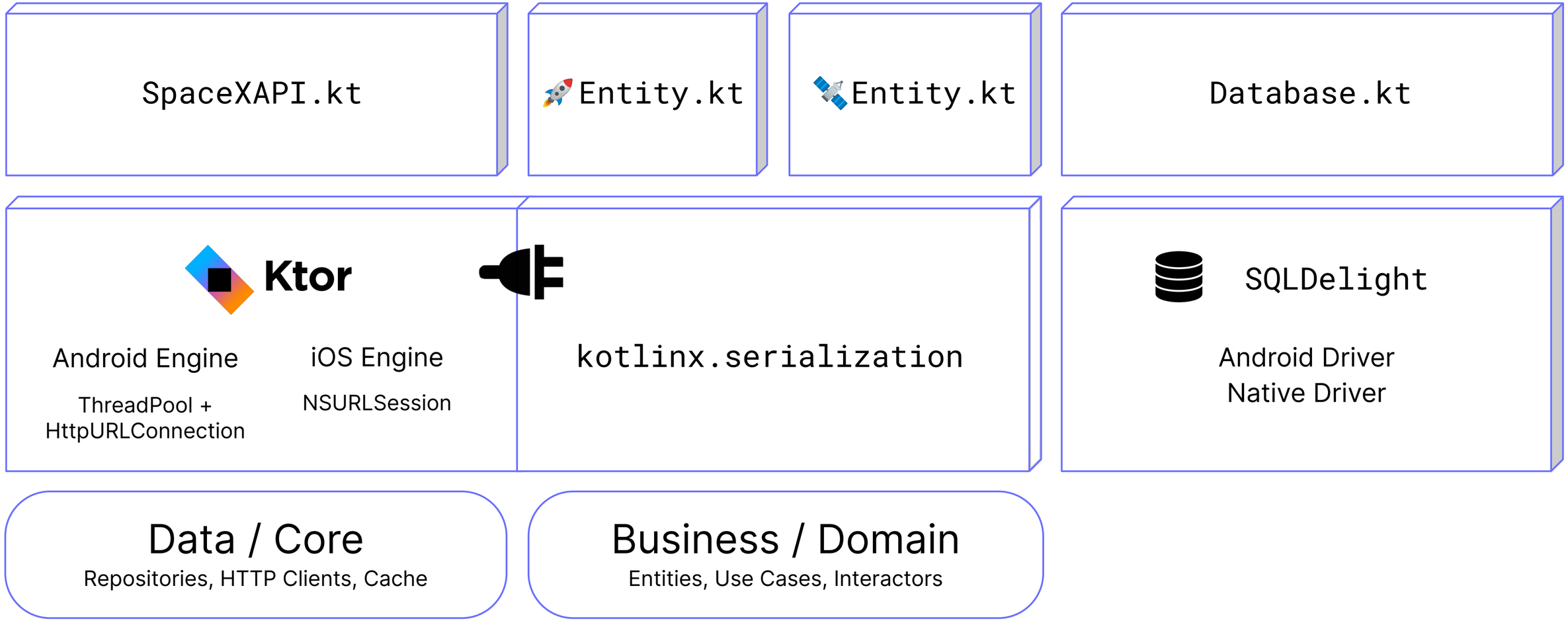
Data / Core
Repositories, HTTP Clients, Cache

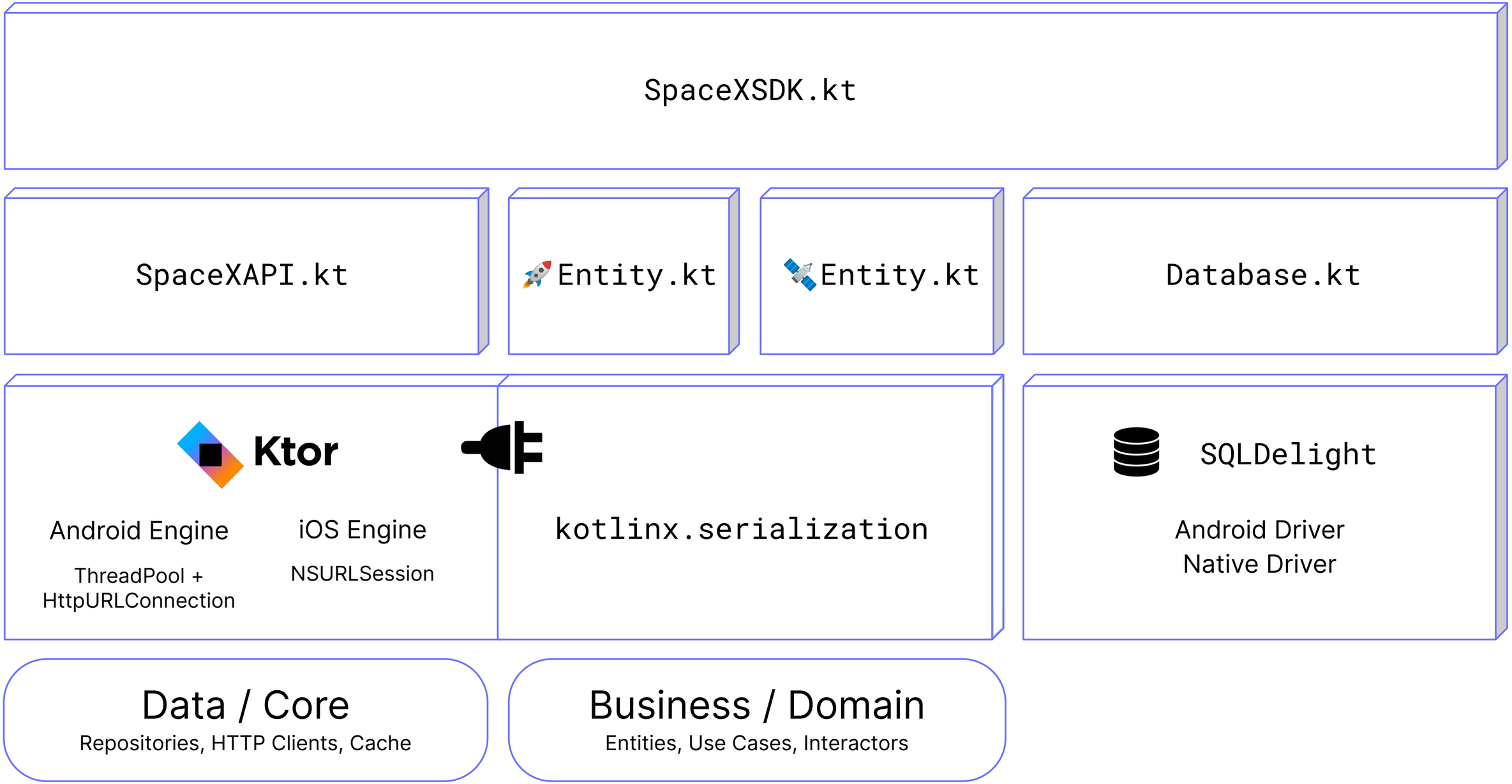

Ktor
Android Engine
ThreadPool +
HttpURLConnection
iOS Engine
NSURLSession
Data / Core
Repositories, HTTP Clients, Cache

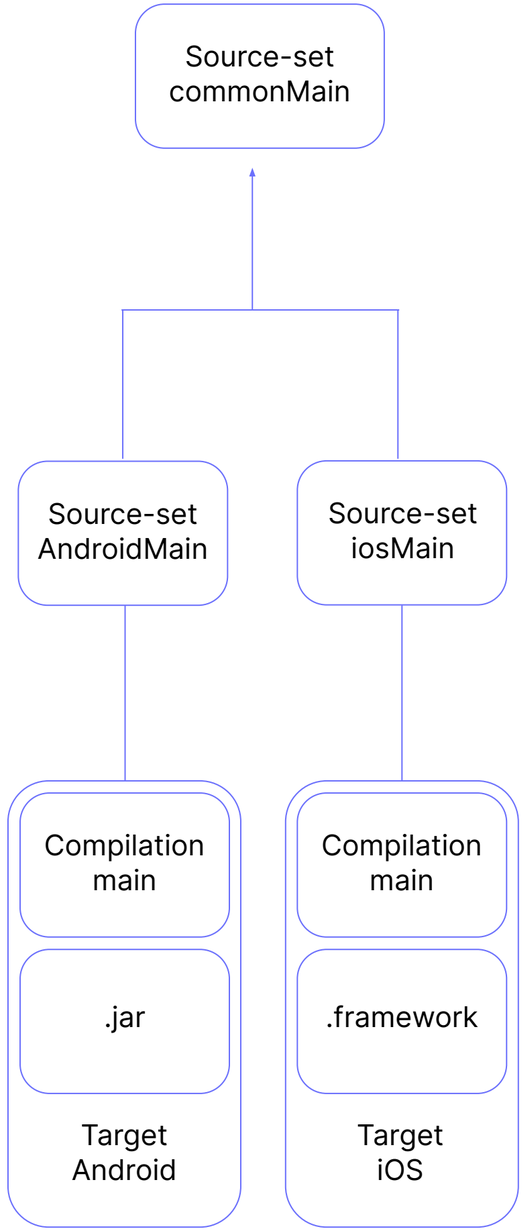

Ktor
Android Engine
ThreadPool +
HttpURLConnection
iOS Engine
NSURLSession
kotlinx.serialization
Data / Core
Repositories, HTTP Clients, Cache


Ktor
Android Engine
ThreadPool +
HttpURLConnection
iOS Engine
NSURLSession
kotlinx.serialization
SQLDelight
Android Driver
Native Driver
Data / Core
Repositories, HTTP Clients, Cache

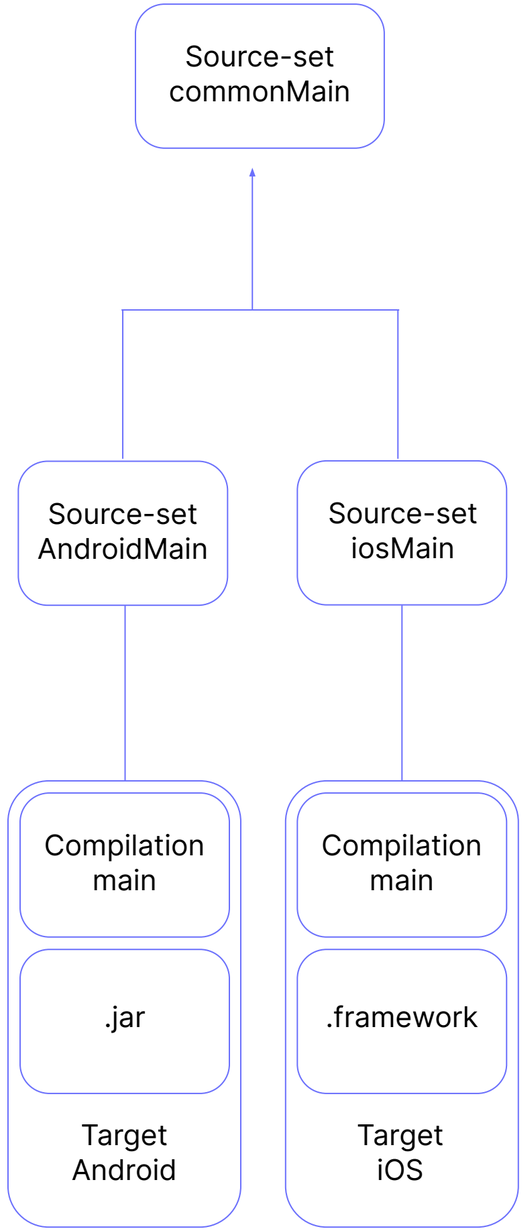

🚀 Entity.kt
🛰️ Entity.kt
Ktor
Android Engine
ThreadPool + HttpURLConnection
iOS Engine
NSURLSession
kotlinx.serialization
SQLDelight
Android Driver
Native Driver
Data / Core
Repositories, HTTP Clients, Cache
Business / Domain
Entities, Use Cases, Interactors


SpaceXAPI.kt
🚀 Entity.kt
🛰️ Entity.kt
Database.kt
Ktor
Android Engine
ThreadPool + HttpURLConnection
iOS Engine
NSURLSession
kotlinx.serialization
SQLDelight
Android Driver
Native Driver
Data / Core
Repositories, HTTP Clients, Cache
Business / Domain
Entities, Use Cases, Interactors


SpaceXSDK.kt
SpaceXAPI.kt
🚀 Entity.kt
🛰️ Entity.kt
Database.kt
Ktor
Android Engine
ThreadPool + HttpURLConnection
iOS Engine
NSURLSession
kotlinx.serialization
SQLDelight
Android Driver
Native Driver
Data / Core
Repositories, HTTP Clients, Cache
Business / Domain
Entities, Use Cases, Interactors

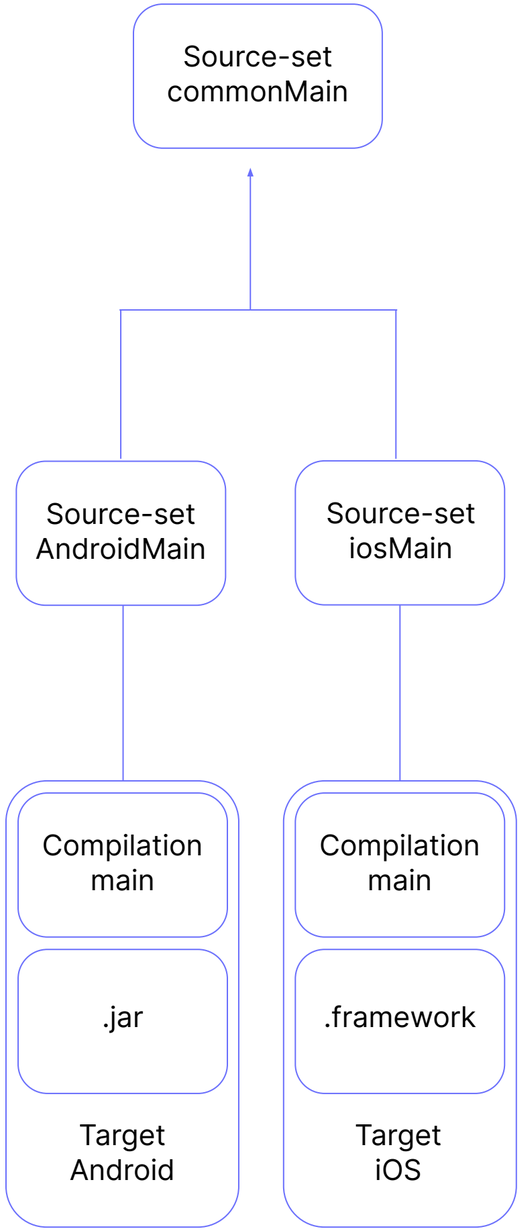

Source-set
commonMain
Source-set
AndroidMain
Source-set
iosMain
Compilation
main
.jar
Target
Android
Compilation
main
.framework
Target
iOS

```kotlin
kotlin {
 android()
 ios()
}
```



```kotlin
kotlin {
 android()
 ios()

  sourceSets {
    val commonMain by getting {
      dependencies {
        implementation("io.ktor:ktor-client-core:$kVrs")
        implementation("org.jetbrains.kotlinx:kotlinx-serialization-core:$sVrs")
        implementation("io.ktor:ktor-client-serialization:$kVrs")
        implementation("com.squareup.sqldelight:runtime:$sqlVrs")
      }
    }
  }
}
```



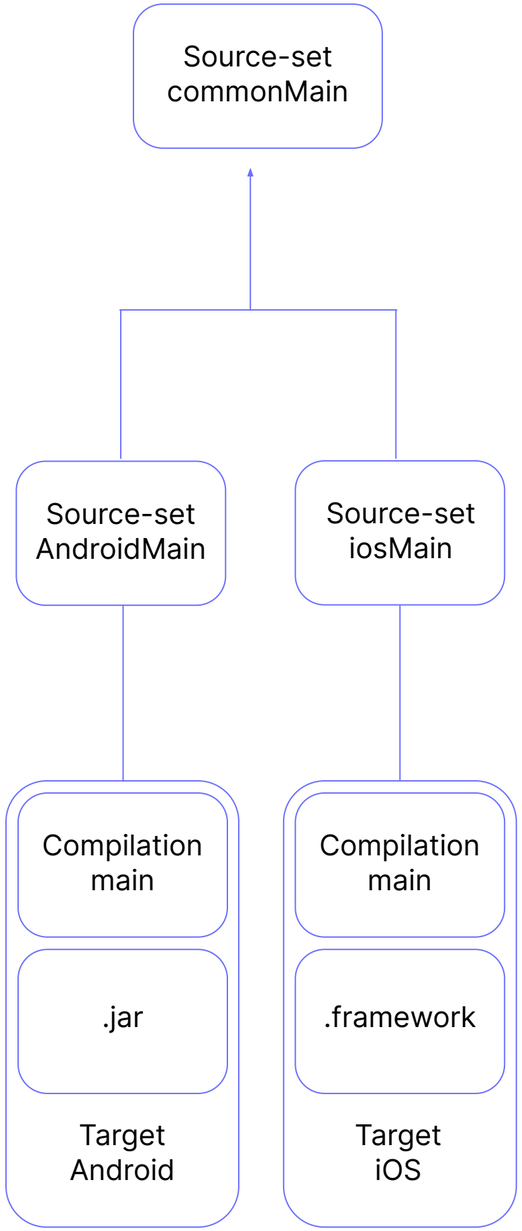
✔️ stdlib by default

```kotlin
kotlin {
 android()
 ios()

  sourceSets {
    val commonMain by getting {
      dependencies {
        implementation("io.ktor:ktor-client-core:$kVrs")
        implementation("org.jetbrains.kotlinx:kotlinx-serialization-core:$sVrs")
        implementation("io.ktor:ktor-client-serialization:$kVrs")
        implementation("com.squareup.sqldelight:runtime:$sqlVrs")
      }
    }
    val androidMain by getting {
      dependencies {
        implementation("io.ktor:ktor-client-android:$kVrs")
        implementation("com.squareup.sqldelight:android-driver:$sVrs")
      }
    }
    val iosMain by getting {
      dependencies {
        implementation("io.ktor:ktor-client-ios:$kVrs")
        implementation("com.squareup.sqldelight:native-driver:$sVrs")
      }
    }
  }
}
```

✅ `stdlib` by default

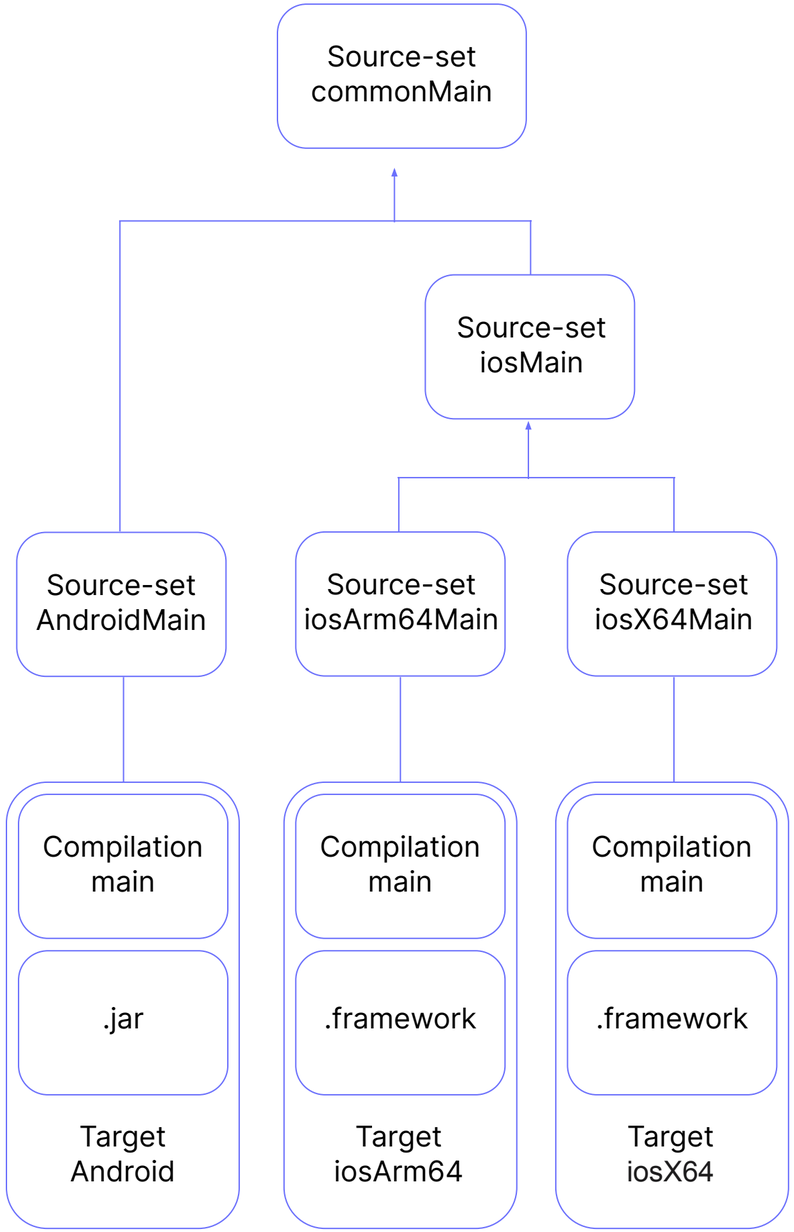
✅ specifying dependencies only once



```kotlin
kotlin {
 android()
 ios()

  sourceSets {
    val commonMain by getting {
      dependencies {
        implementation("io.ktor:ktor-client-core:$kVrs")
        implementation("org.jetbrains.kotlinx:kotlinx-serialization-core:$sVrs")
        implementation("io.ktor:ktor-client-serialization:$kVrs")
        implementation("com.squareup.sqldelight:runtime:$sqlVrs")
      }
    }
    val androidMain by getting {
      dependencies {
        implementation("io.ktor:ktor-client-android:$kVrs")
        implementation("com.squareup.sqldelight:android-driver:$sVrs")
      }
    }
    val iosMain by getting {
      dependencies {
        implementation("io.ktor:ktor-client-ios:$kVrs")
        implementation("com.squareup.sqldelight:native-driver:$sVrs")
      }
    }
  }
}
```

- ✅ `stdlib` by default
- ✅ specifying dependencies only once
- ✅ hierarchical project structure support



Diving into Kotlin Multiplatform

Dmitry Savvinov
Team Lead
in Kotlin Multiplatform Mobile

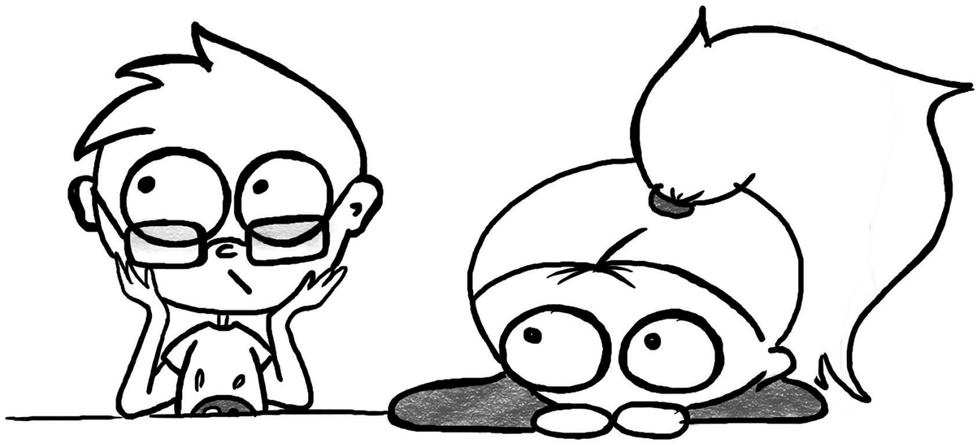
What to share?

UI
Views

Presentation
Presenters, View Models, Controllers

Business / Domain
Entities, Use Cases, Interactors

Data / Core
Repositories, HTTP Clients, Cache



What to share?

UI
Views

Presentation
Presenters, View Models, Controllers

Business / Domain
Entities, Use Cases, Interactors

Data / Core
Repositories, HTTP Clients, Cache

MVIKotlin
★ 320

MVI framework for Kotlin Multiplatform

Category: Architecture
Gradle: com.arkivanov.mvikotlin:mvikotlin:2.0.0-preview4
Kotlin: 1.3.70
Targets: androidJvm, ios_arm64, ios_x64, js, jvm, linux_x64, common

GITHUB

trikot.viewmodels

Meta abstraction of visual comp
Multiplatform

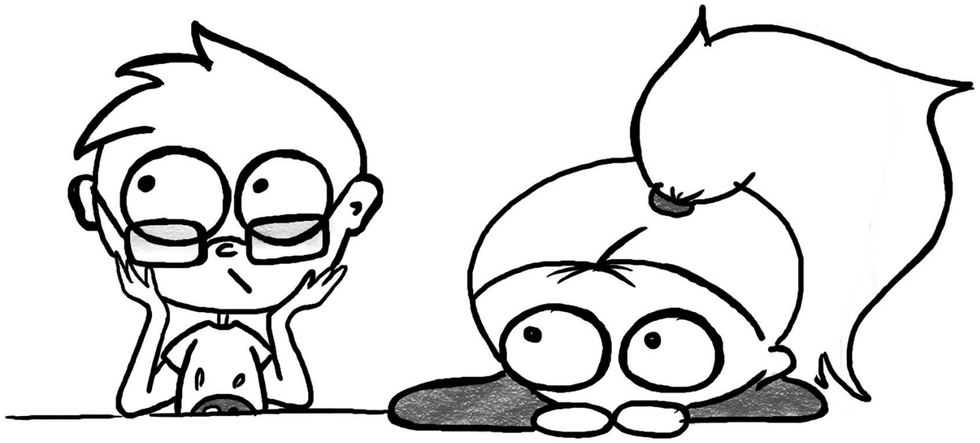
Category: UI

moko-mvvm

Model-View-ViewModel architecture components fo
mobile (android & ios) Kotlin Multiplatform development

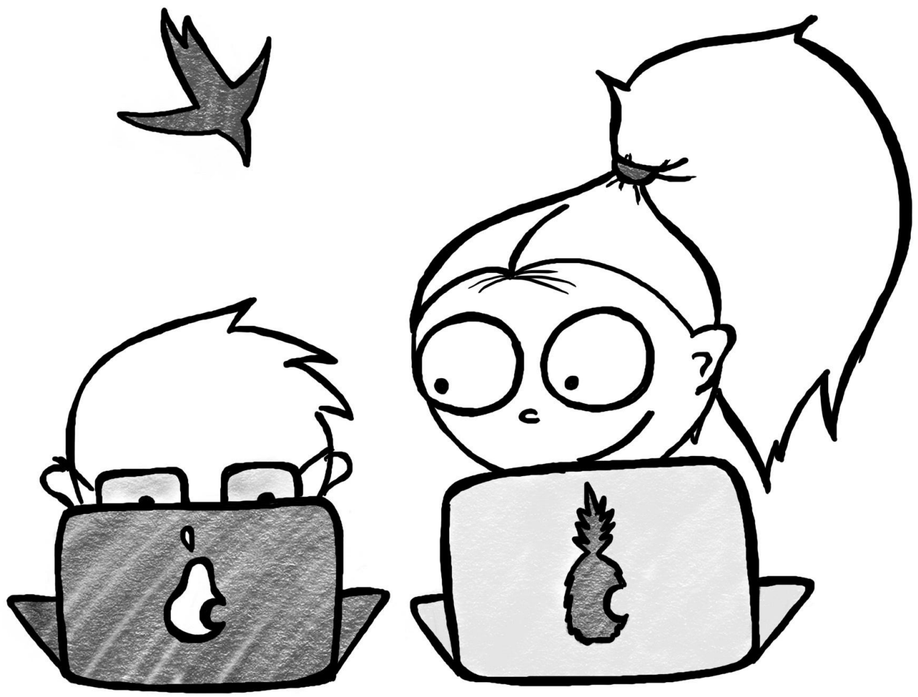
Category: Architecture
Gradle: dev.icerock.moko:mvvm:0.8.0
Kotlin: 1.4.0
Targets: androidJvm, ios_arm64, ios_x64, common

GITHUB



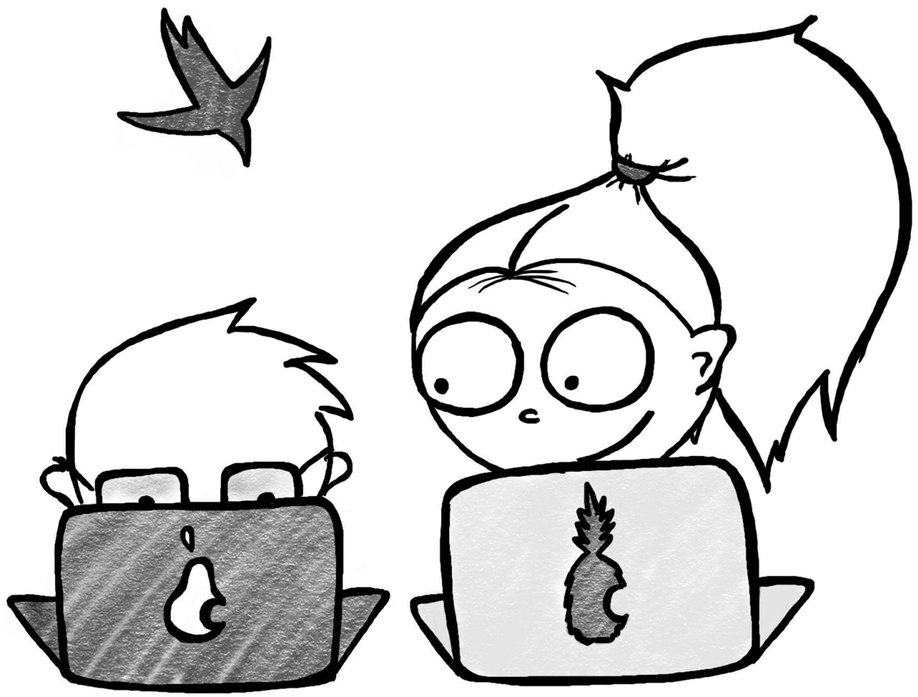
Working on shared code with KMM

Tight integration with the iOS development process



Tight integration with the iOS development process

- Call Kotlin code from Objective-C/Swift and use iOS libraries from Kotlin
- Integrate KMM module in iOS project through CocoaPods
- Write, run, test, debug shared code all in Android Studio



Bidirectional interoperability with Objective-C/Swift

```
fun getLaunches(): List<RocketLaunch> {
   return database.getAllLaunches()
}
```

.kt

```
let launches = sdk.getLaunches()
let launch: RocketLaunch? = launches.first
```

.swift

Bidirectional interoperability with Objective-C/Swift

```kt
suspend fun getLaunches(): List<RocketLaunch> {
    val cachedLaunches = database.getAllLaunches()
    return if (cachedLaunches.isNotEmpty()) {
        cachedLaunches
    } else {
        api.getAllLaunches()
    }
}
```

.kt

```swift
sdk.getLaunches() { launches, _ in
    let launch: RocketLaunch? = launches?.first
}
```

.swift

Bidirectional interoperability with Objective-C/Swift

```kt
@Throws(Exception::class)
suspend fun getLaunches(): List<RocketLaunch> {
    val cachedLaunches = database.getAllLaunches()
    return if (cachedLaunches.isNotEmpty()) {
        cachedLaunches
    } else {
        api.getAllLaunches()
    }
}
```

.kt

```swift
sdk.getLaunches() { launches, error in
    if let launches = launches {
        print(launches)
    } else {
        print(error?.localizedDescription)
    }
}
```

.swift

Kotlin/Native interoperability with Swift/Objective-C

Supports all basic concepts, including

- Companion objects
- Data classes
- Extensions
- Objective-C generics
- And so on kotl.in/objc_interop

Integration with CocoaPods

```
kotlin {
   android()
   ios()

   cocoapods {
       summary = "CocoaPods test library"
       homepage = "https://github.com/JetBrains/kotlin"
       pod("AFNetworking", "~> 4.0.0")
       podfile = project.file("../ios-app/Podfile")
   }
}
```

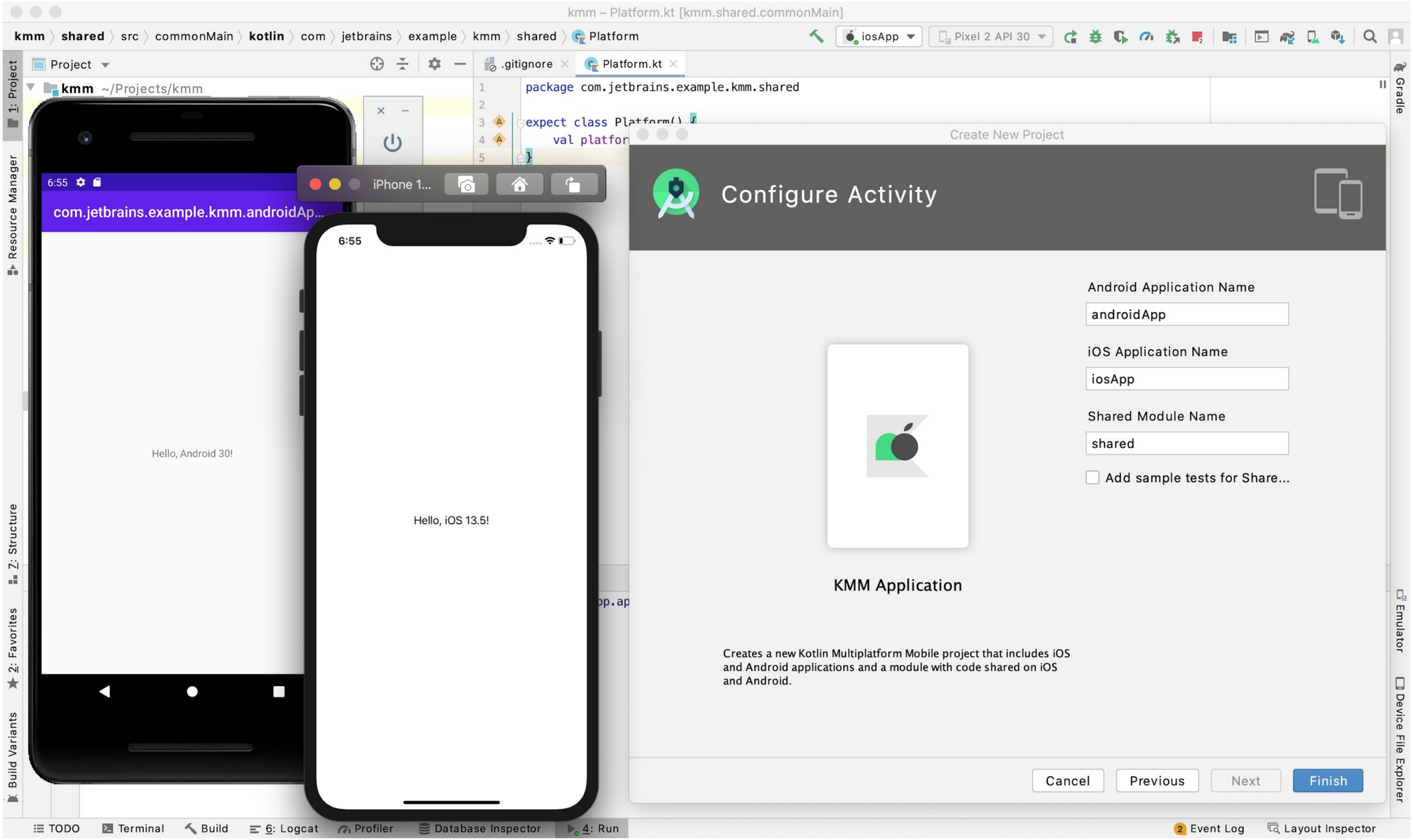
build.gradle.kts

```
use_frameworks!

target 'ios-app' do
    pod 'kotlin_library', :path => '../kotlin-library'
end
```

Podfile

Working on shared code without switching IDEs


kmm – Platform.kt [kmm.shared.commonMain]
kmm shared src commonMain kotlin com jetbrains example kmm shared Platform
iosApp
Pixel 2 API 30
Project
kmm ~/Projects/kmm
.gitignore
Platform.kt
package com.jetbrains.example.kmm.shared
expect class Platform() {
val platfor
}
6:55
com.jetbrains.example.kmm.androidAp...
Hello, Android 30!
iPhone 1...
6:55
Hello, iOS 13.5!
Create New Project
Configure Activity
Android Application Name
androidApp
iOS Application Name
iosApp
Shared Module Name
shared
Add sample tests for Share...
KMM Application
Creates a new Kotlin Multiplatform Mobile project that includes iOS and Android applications and a module with code shared on iOS and Android.
Cancel
Previous
Next
Finish
TODO
Terminal
Build
6: Logcat
Profiler
Database Inspector
4: Run
Event Log
Layout Inspector
1: Project
Resource Manager
7: Structure
2: Favorites
Build Variants
Gradle
Emulator
Device File Explorer

KMM – Platform.kt [KMM.shared.commonMain]

mmonMain 〉 kotlin 〉 com 〉 jetbrains 〉 example 〉 kmm 〉 shared

androidApp | Pixel 2 API 30

Project

- KMM ~/Projects/KMM
  - .gradle
  - .idea
  - androidApp
  - build
  - gradle
  - iosApp
  - shared
    - build
    - src
      - androidMain
      - androidTest
      - commonMain
        - kotlin [commonMain] sources root
          - com.jetbrains.example.kmm.shared
            - Greeting
            - Platform
      - commonTest
      - iosMain
      - iosTest
        - kotlin [iosTest] test sources root
          - com.jetbrains.example.kmm.shared
            - iosTest.kt
    - build.gradle.kts
  - .gitignore

Greeting.kt | iosTest.kt | Platform.kt

```kotlin
package com.jetbrains.example.kmm.shared

expect class Platform() {
    val platform: String
}
```

Run: iosApp | GreetingTest.testExample

```
/Users/Ekaterina.Petrova/Projects/KMM/build/ios/Debug-iphonesimulator/iosApp.app

Process finished with exit code 0
```

TODO | Terminal | Build | 6: Logcat | Profiler | Database Inspector | 4: Run | 5: Debug | Event Log | Layout Inspector

Build iOS application finished in 5 s 593 ms (moments ago)

5:2 LF UTF-8 4 spaces

iPhone 1...

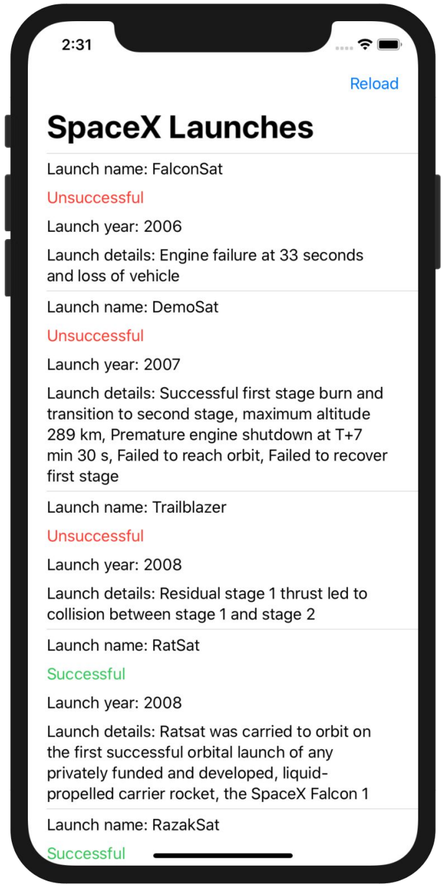
7:52

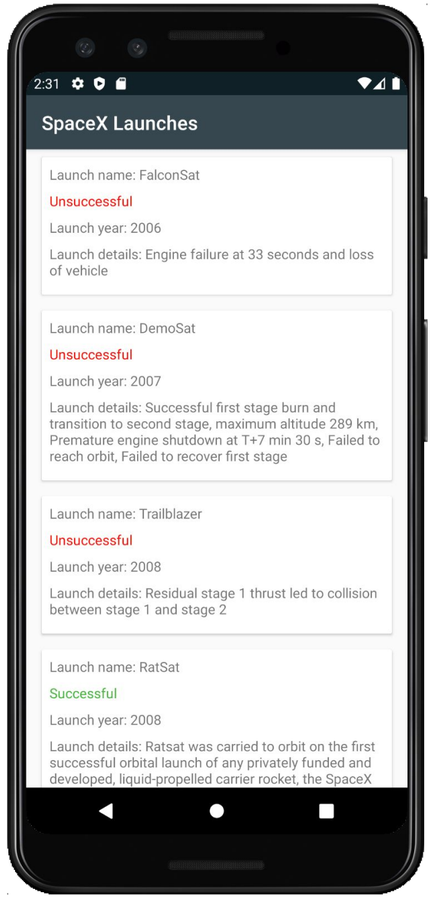
Files | Shortcuts | Contacts | Watch | iosApp

Test & Deploy your mobile apps with KMM

As usual 🤗

1. Introduction
2. Creating the KMM project
3. Adding dependencies to the multiplatform library
4. Creating an application data model
5. Configuring SQLDelight and implementing cache logic
6. Implementing an API service
7. Building SDK
8. Creating the Android application
9. Creating the IOS application
10. Summary





https://kotl.in/try-kmm

It’s time for Kotlin Multiplatform Mobile!

- KMM Goes Alpha! 🎉

It’s time for Kotlin Multiplatform Mobile!

- KMM Goes Alpha! 🎉
- Easy to start
- Integrate in existing projects with minimal cost

It’s time for Kotlin Multiplatform Mobile!

- KMM Goes Alpha! 🎉
- Easy to start
- Integrate in existing projects with minimal cost
- Be the part of a growing community and influence the development of the whole ecosystem

How to start?

Learn how to KMM on the new developer portal

kotl.in/kmm-doc

Find inspiration in stories from various teams who are already using KMM in production

kotl.in/kmm-cases

Try the new KMM Plugin for Android Studio

kotl.in/kmm-plugin

Thanks!
Have a nice Kotlin

@KathrinPetrova