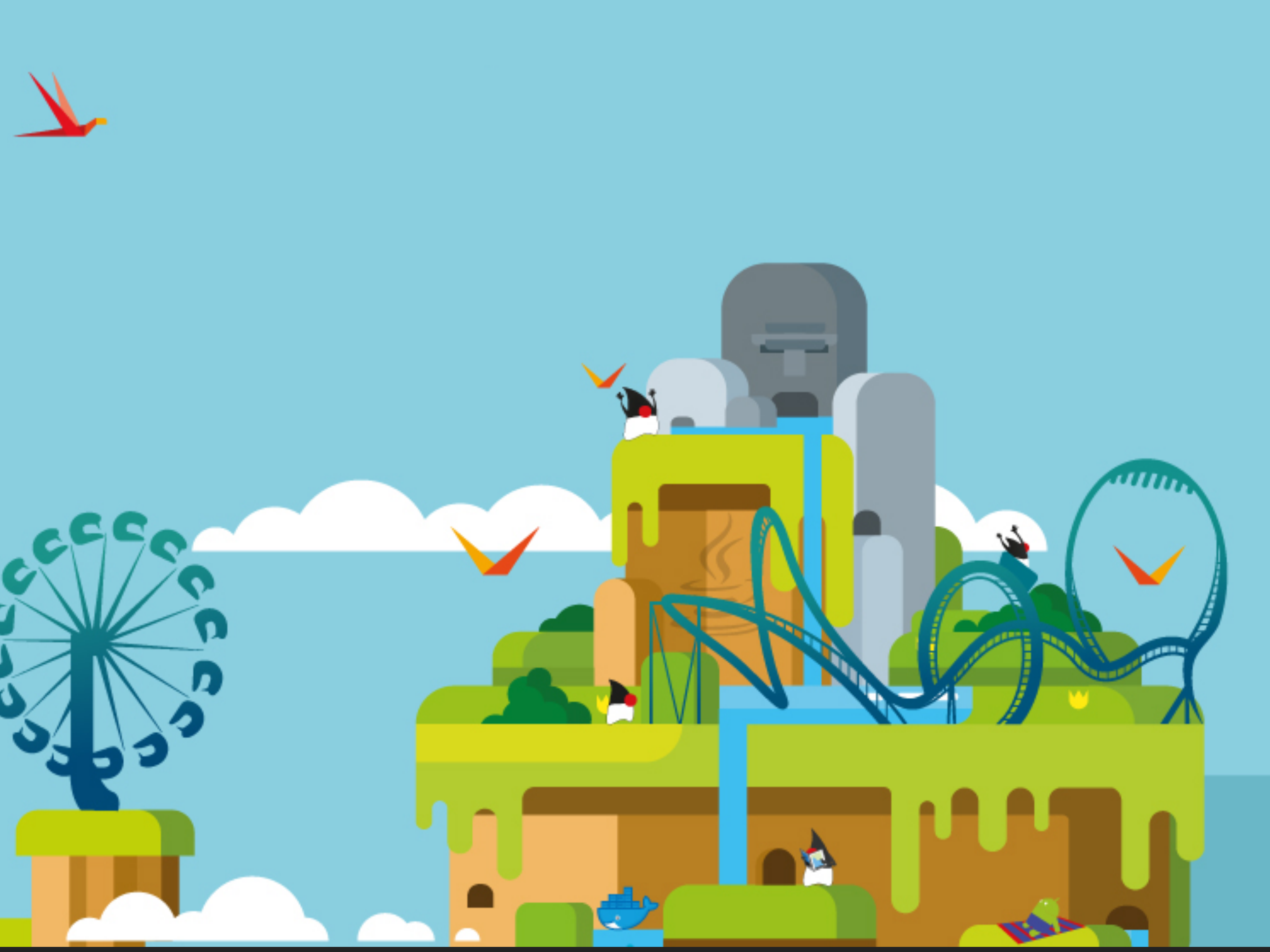
OR HOW ANDROID DEVELOPER IS WRITING IOS CONFERENCE APP OVER A WEEKENDS

KOTLIN MULTIPLATFORM ROLLER-COASTER

LIVE DEMO



ROLLER-COASTER

- an exciting entertainment in an amusement park, like a fast train that goes up and down very steep slopes and around very sudden bends
- a situation which changes from one extreme to another,
or in which a person's feelings change from one extreme to another.

Cambridge Dictionary

HARAKAL.DE

MICHAL HARAKAL

CHALLENGE

DUKECON.ORG

JavaLand

DOAG

DOAG

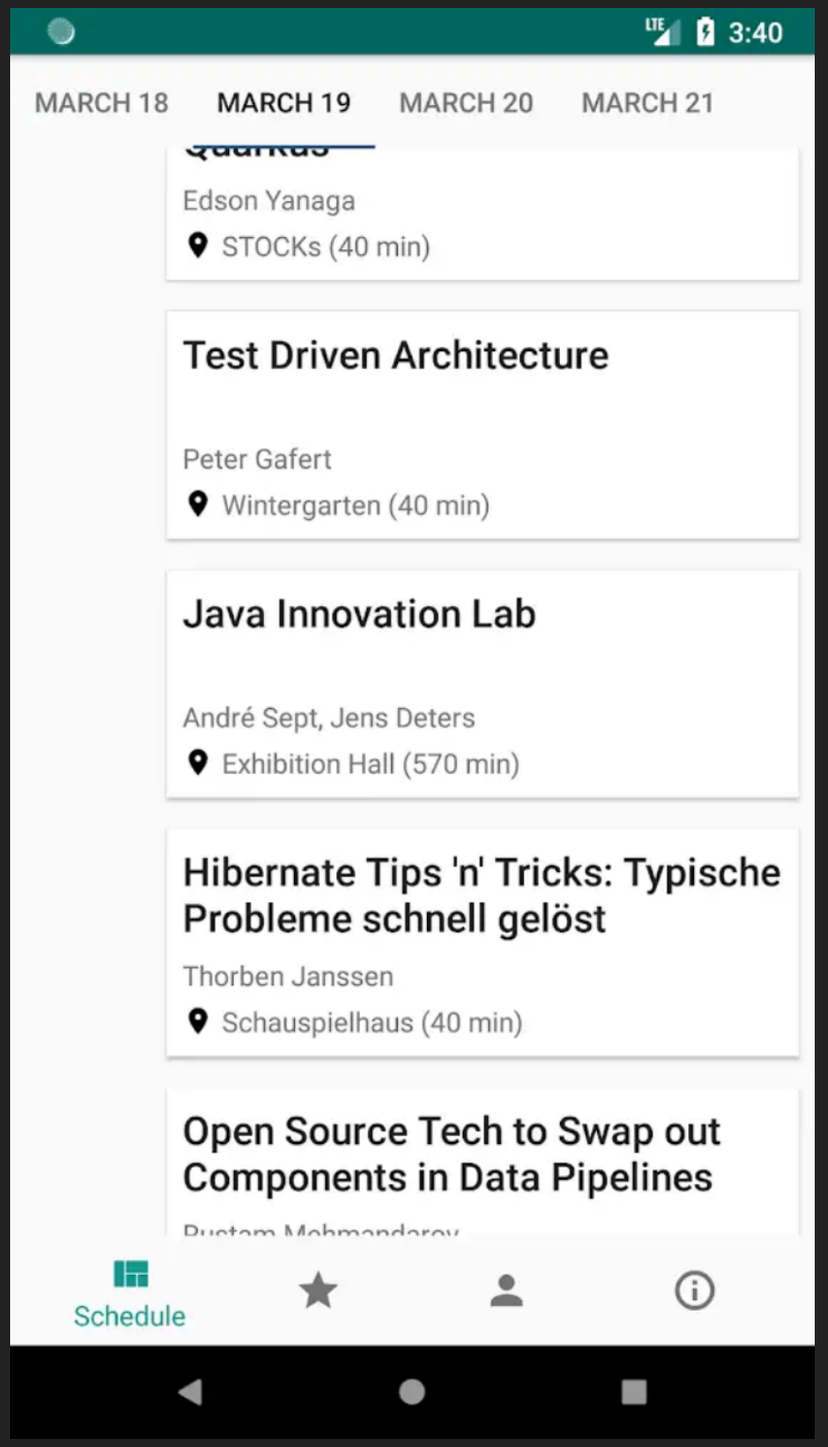
Apex

APEX connect

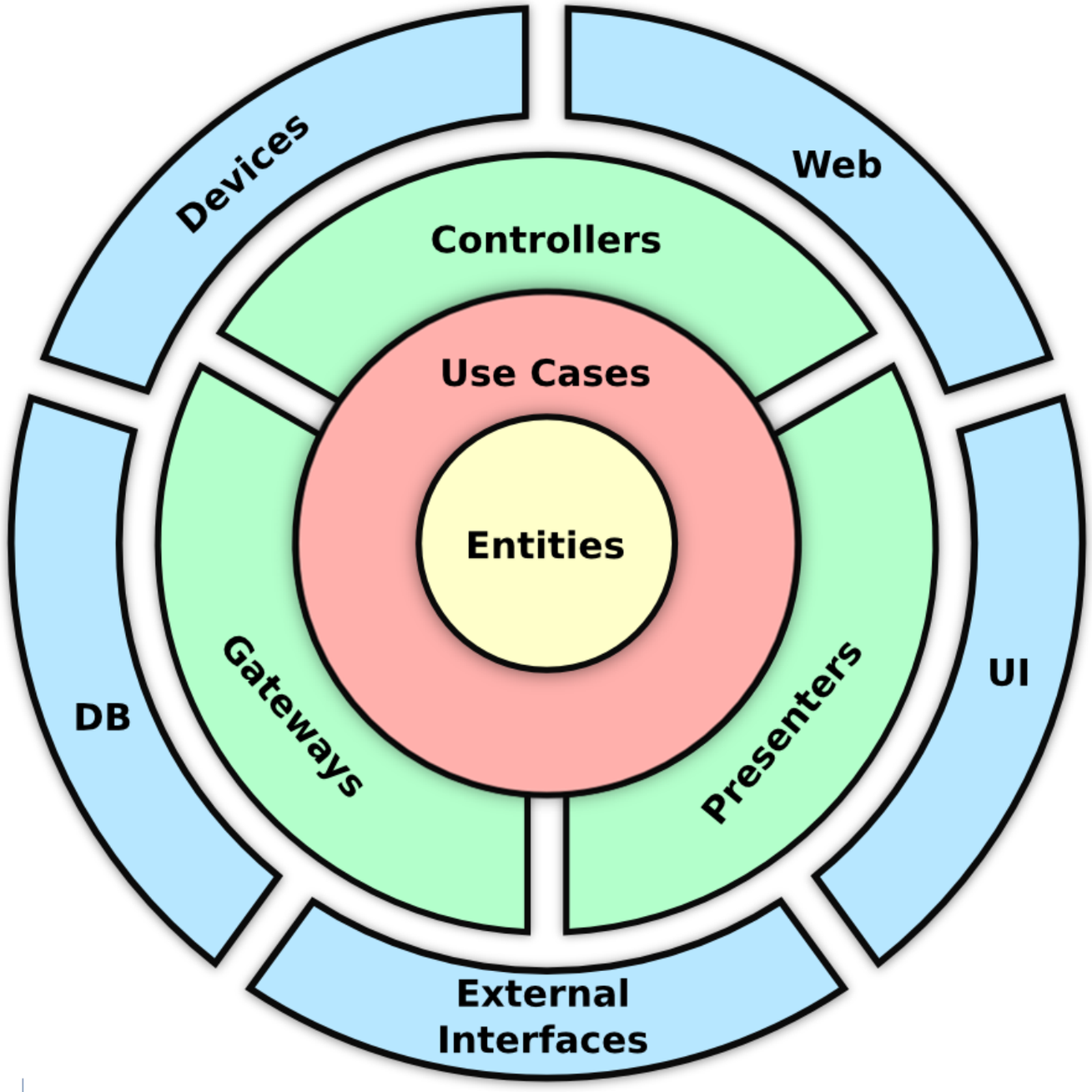
APACHECON

STARTING POSITION – JAVALAND NATIVE ANDROID CLIENT

- written 100% in Kotlin
- clean architecture approach with building blocks as separated gradle modules
- coroutines
- MVP for presentation layer
- as far possible, java only, to get rid of Android whenever you can
- current Android libs stack (okhttp, retrofit, picasso, gson)



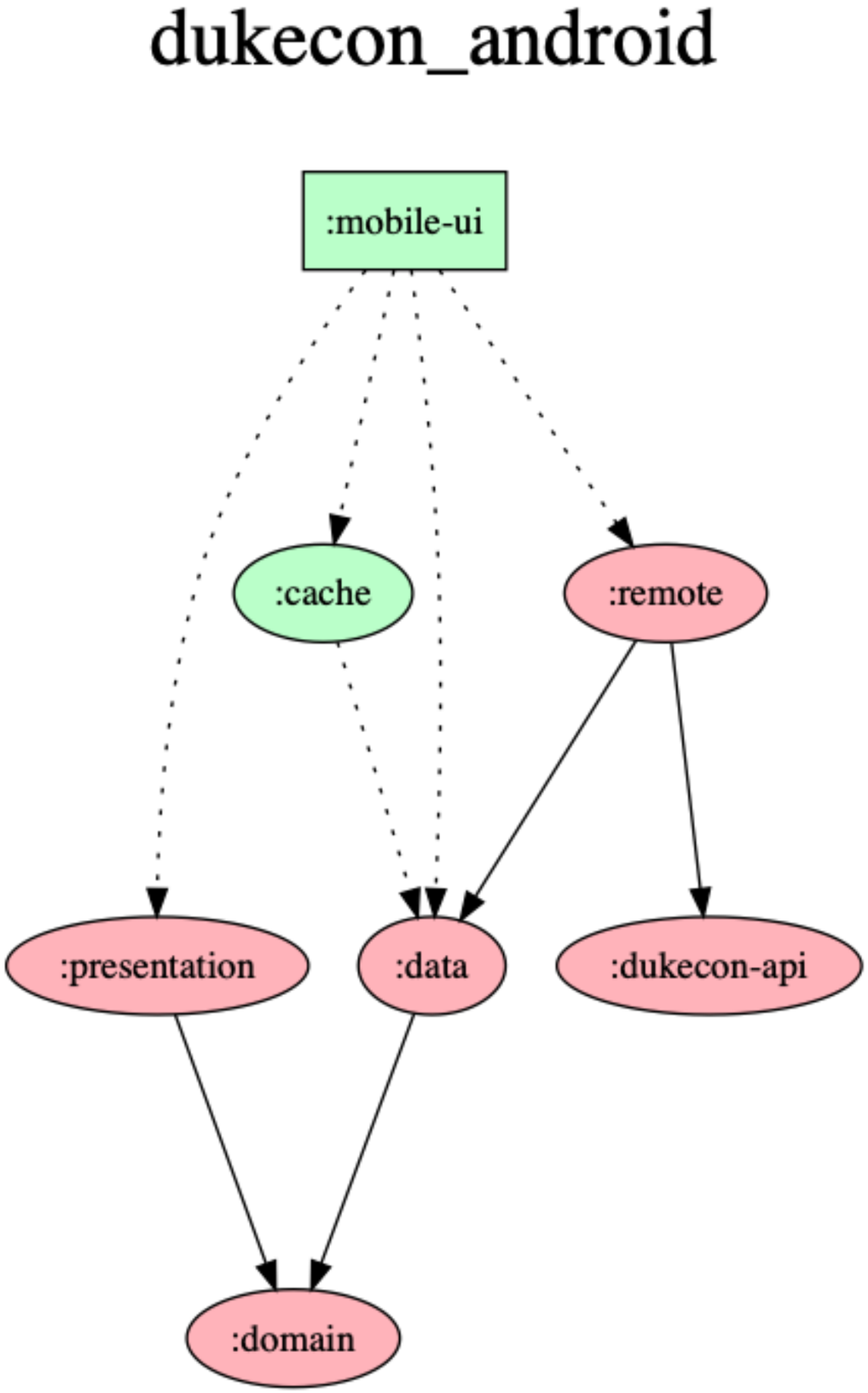
CLEAN ARCHITECTURE

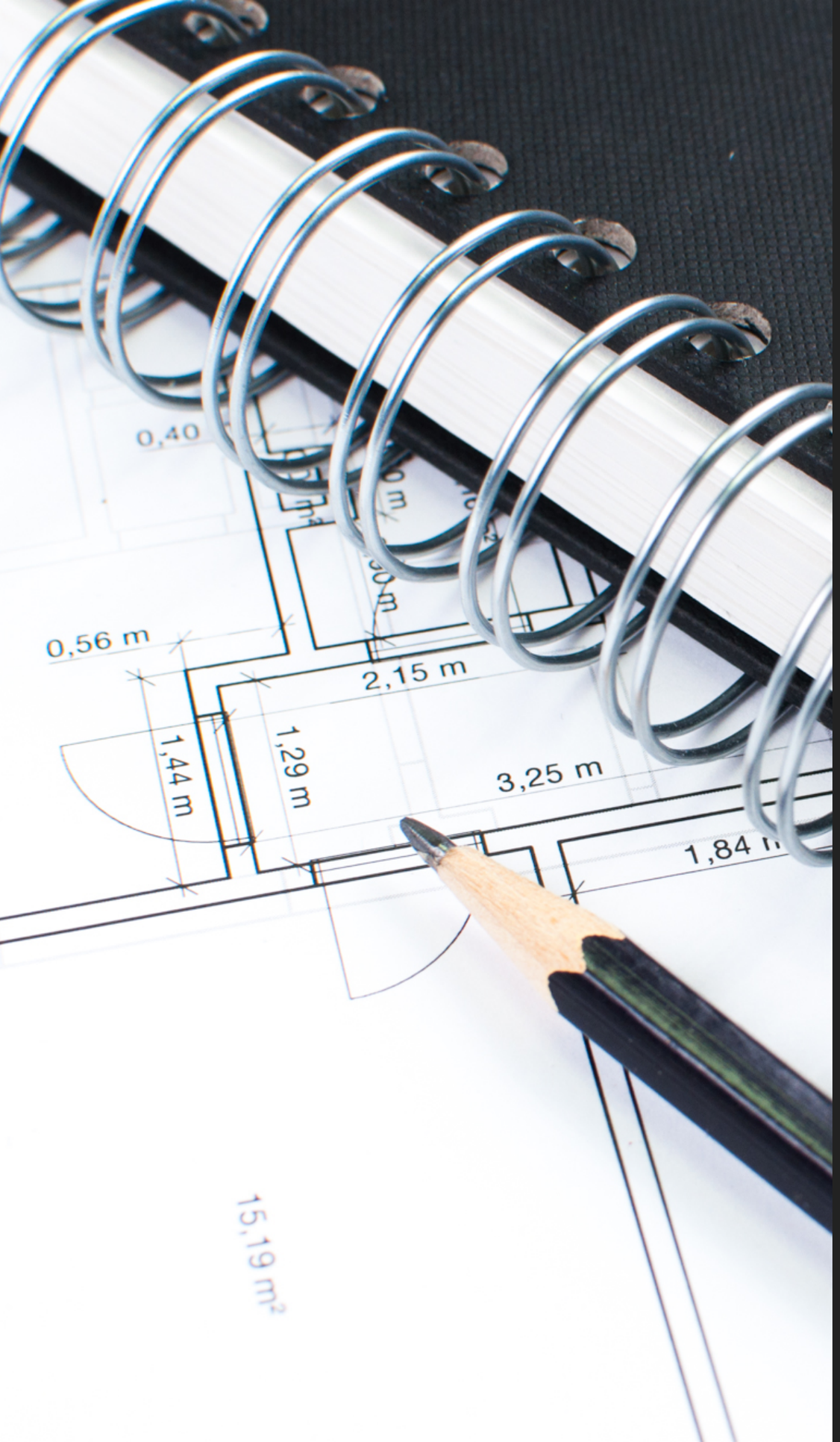


- layered
- dependecies points inwards
- keeps Android dependencies on outside

INITIAL MODULES STRUCTURE

- only **cache** and **app** are Android specific
- java only modules
- **dukecon-api** - Retrofit client and DTOs generated completely with swagger





PLAN

PLAN

- Fokus on Android first
- convert java modules to Kotlin Multiplatform
- migrate platform dependent to Kotlin Multiplatform (**remote**, **cache**)
- adapt existing iOS App (**Kotlinkonf, DroidconKotlin**)

STANDING ON THE SHOULDERS OF GIANTS

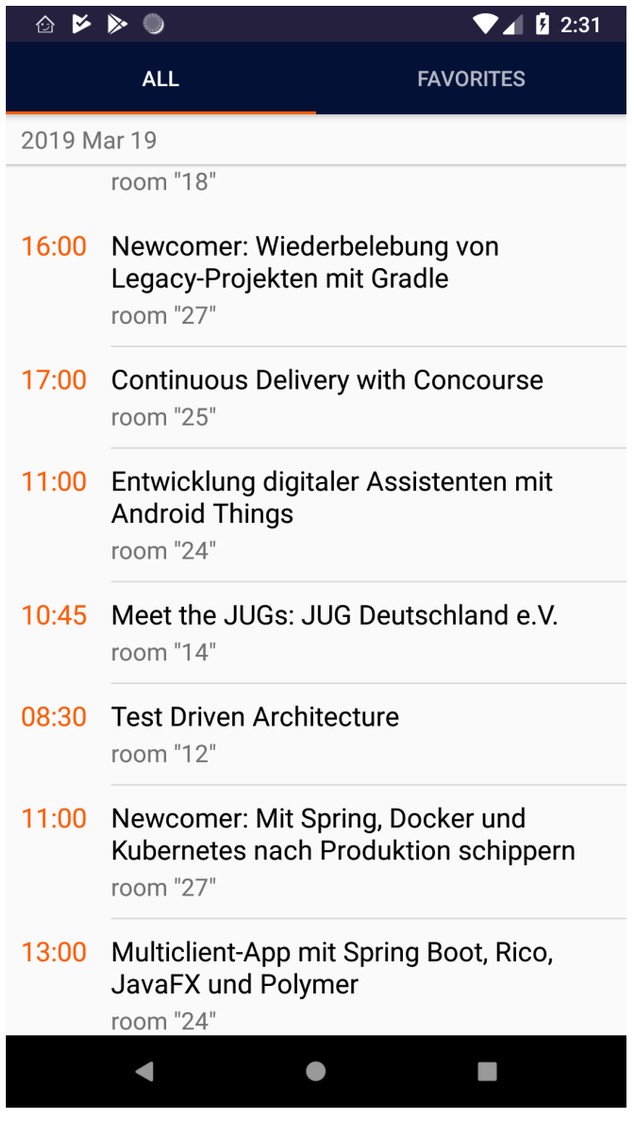
STANDING ON THE SHOULDERS OF GIANTS

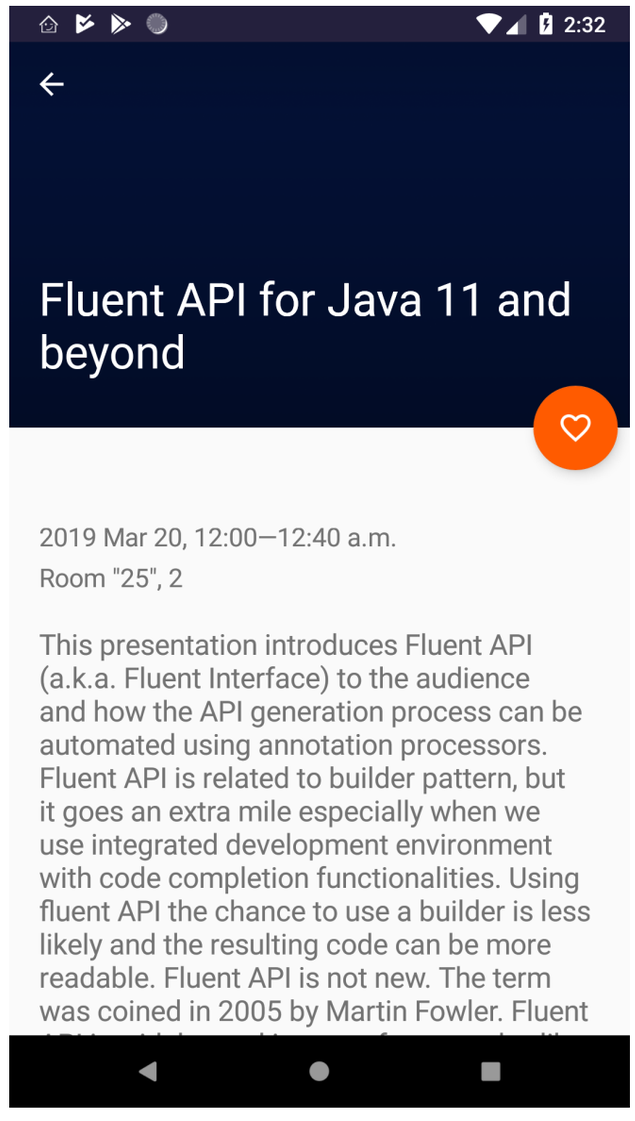
- Chicago Roboto
- Kotlin clean architecture starter by **Joe Birch**
- Dukecon.org
- Kotlinkonf
- **DroidconKotlin** by Touchlab

CHALLENGE DIARY

WEEKEND 1 – ANDROID SPEAKS DUKECON

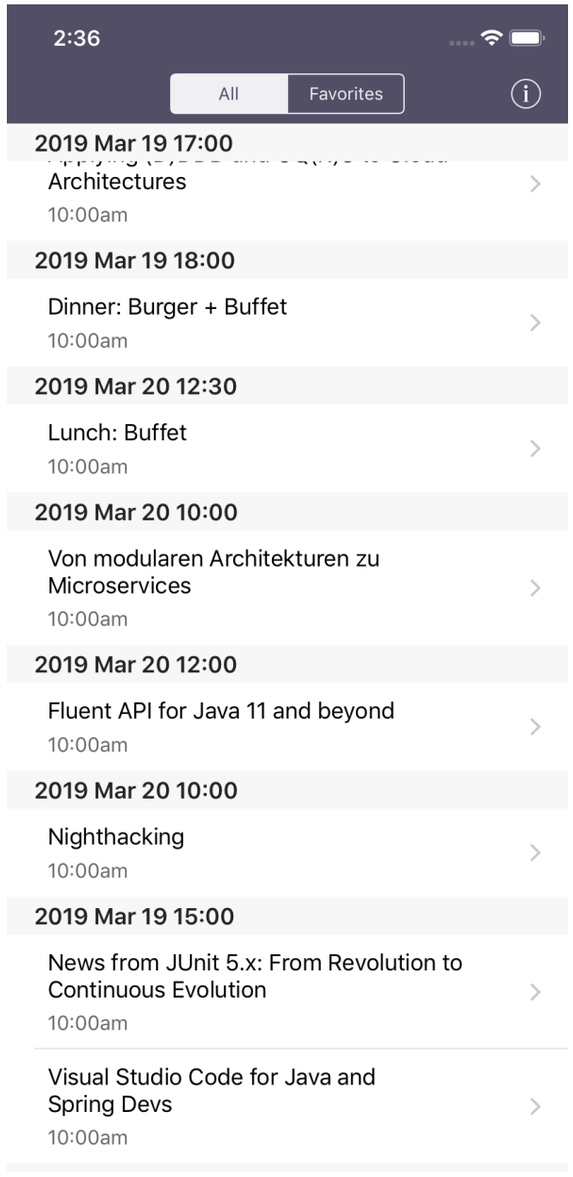
- swagger generated remote client and data object
- Android running

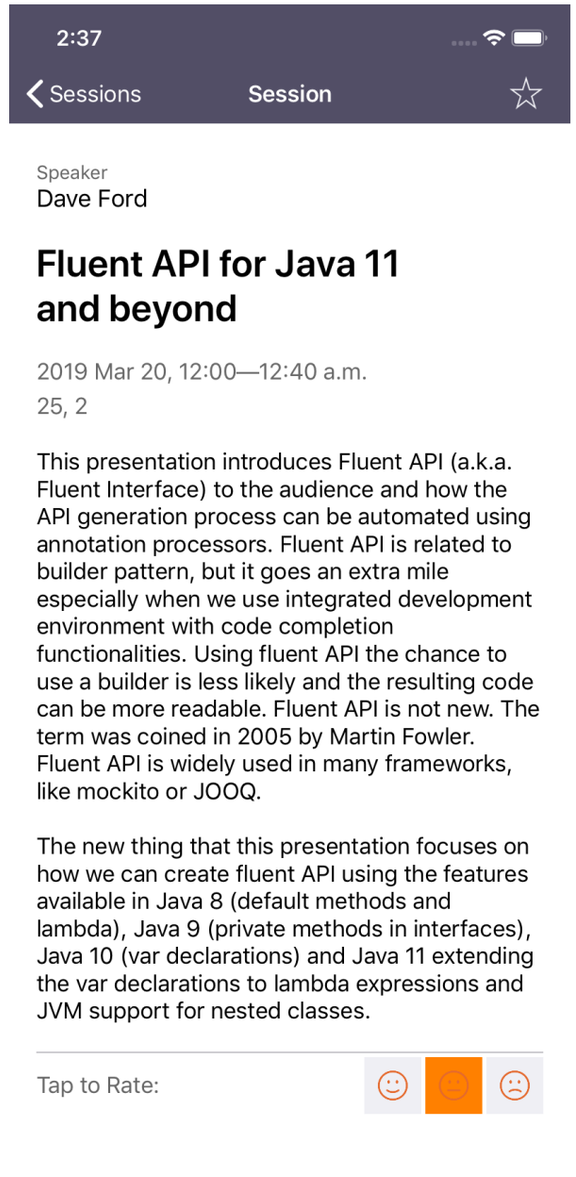




WEEKEND 1 – IOS SPEAKS DUKECON

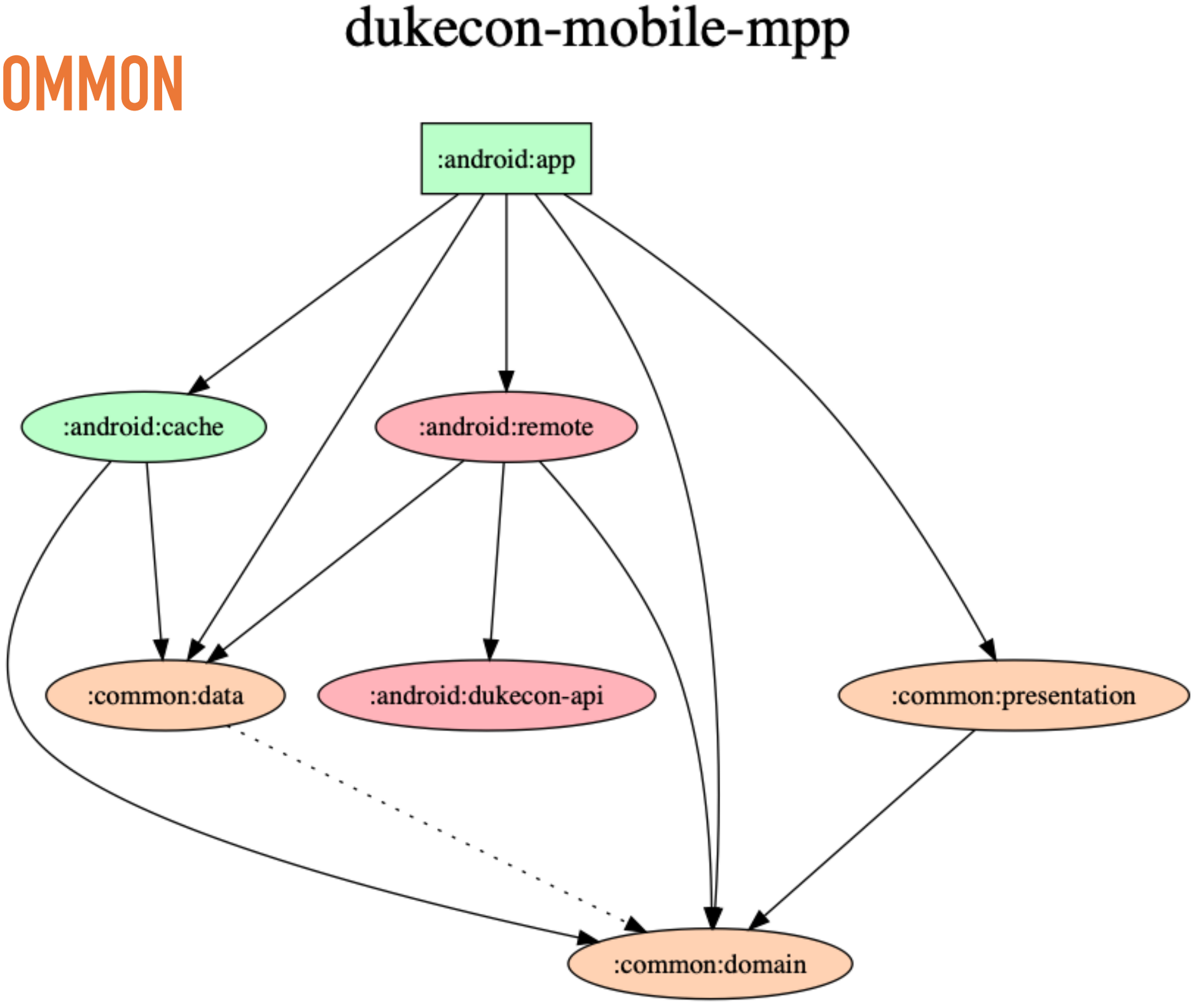
- how to debug?
- iOS specifics wisdom required
- *AppCode*
- packaging as xcode-framework





WEEKEND 2 – COMMON

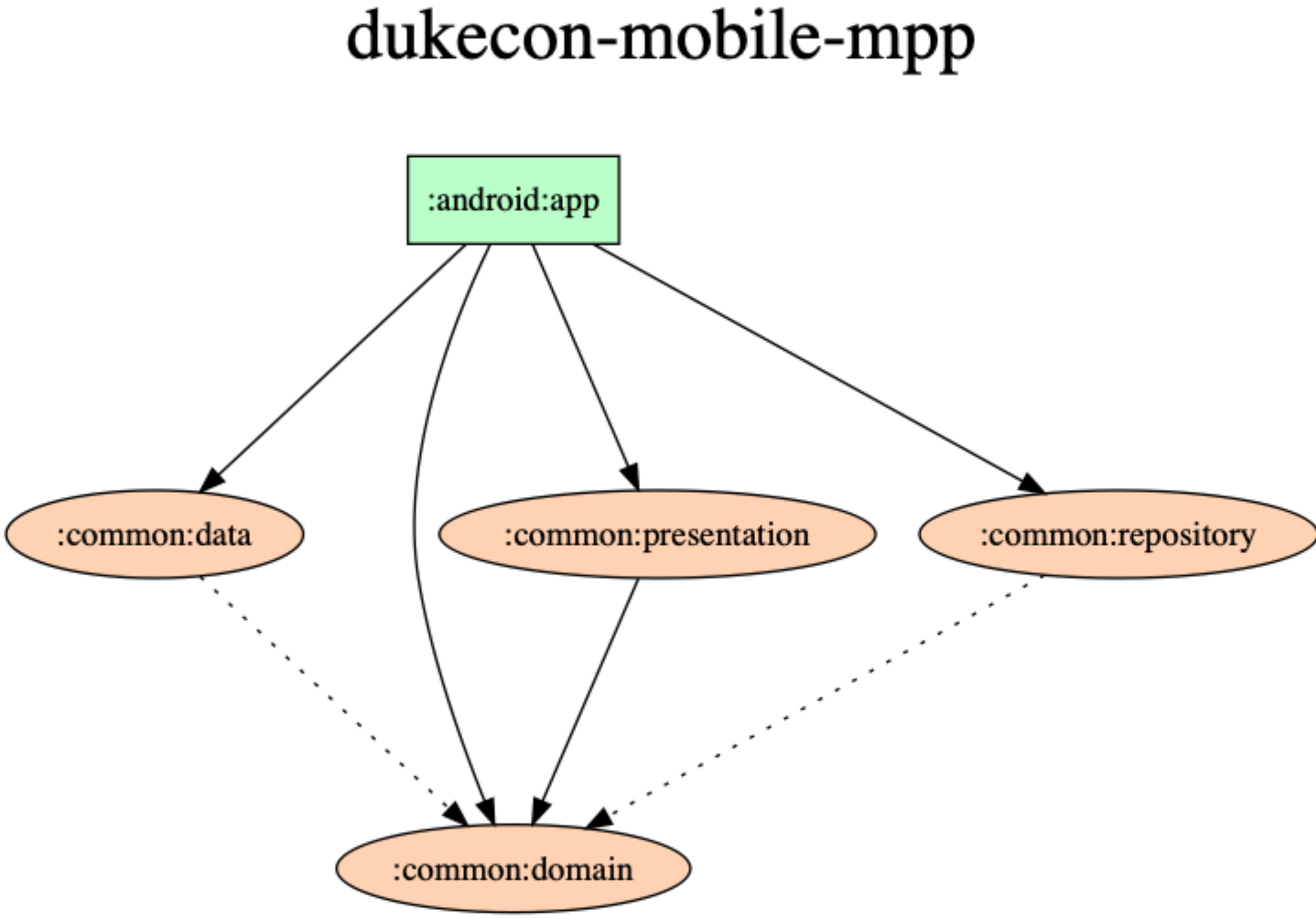
- moving java only, platform independent classes into Kotlin MPP
- removing code related to a *network awareness*
- removing **@Inject** annotations



WEEKEND 3 – FINAL WEEKEND

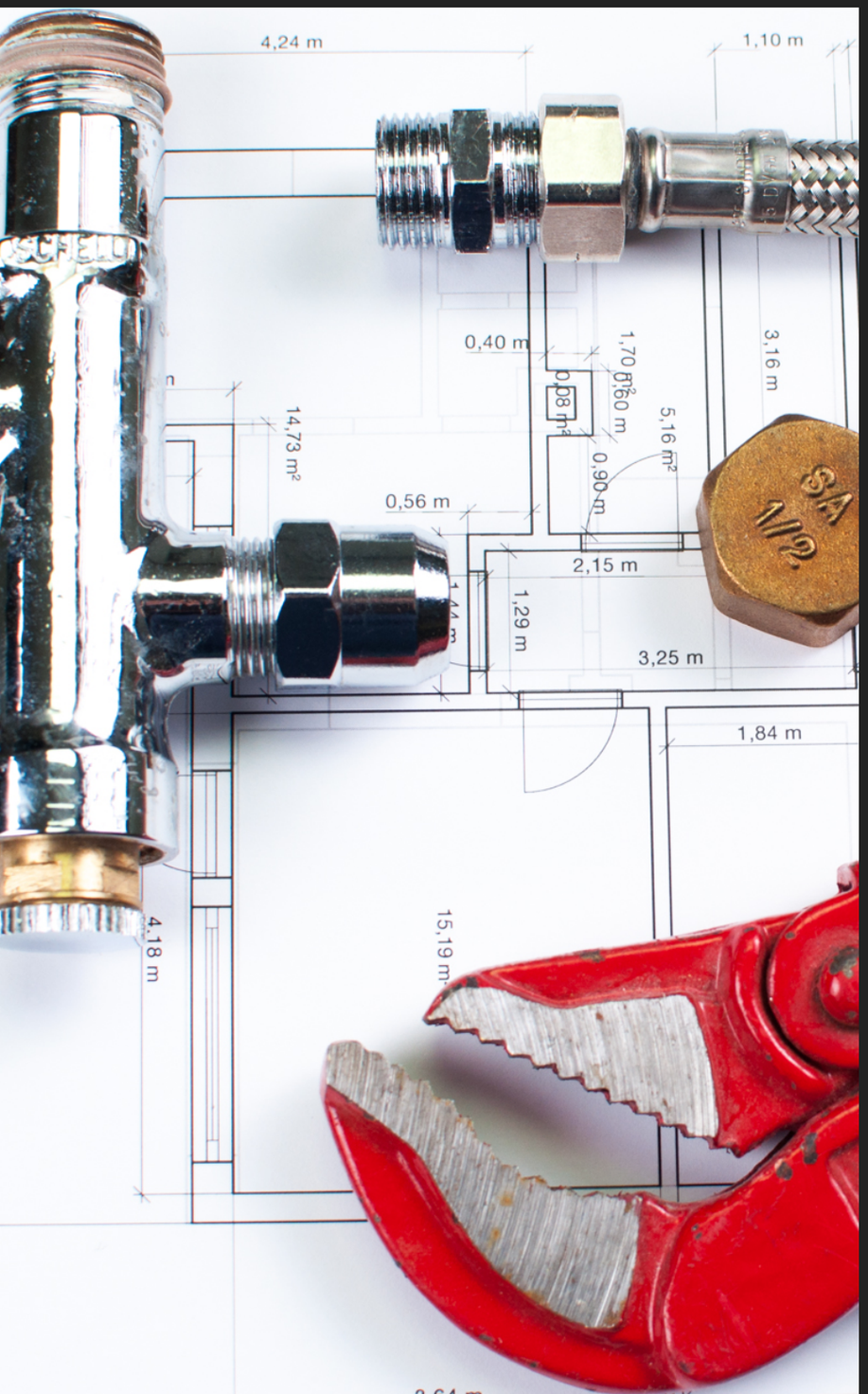
- Android App runs with Kotlin MPP Module
- still issues with Date Time
- original heavy usage of constructor injected parameters has to be replaced by manual calls in Dagger module classes

WEEKEND 3


dukecon-mobile-mpp
:android:app
:common:data
:common:presentation
:common:repository
:common:domain

TASK TO SOLVE

PLUMBING



- Gradle multiplatform modules
  - common
  - android
  - iOS
- Dependency management
  - Spring Dependecy Management
  - support for **BOM**

NETWORKING OR KTOR FUN

- IntelliJ **Ktor** plugin generates basic project structure, API service and DTO classes for serialisation
- manual changes required for DTOs (@Optional)
- hard to debug
- logging not working
- Okhttp by default, SSL not working on 4.4

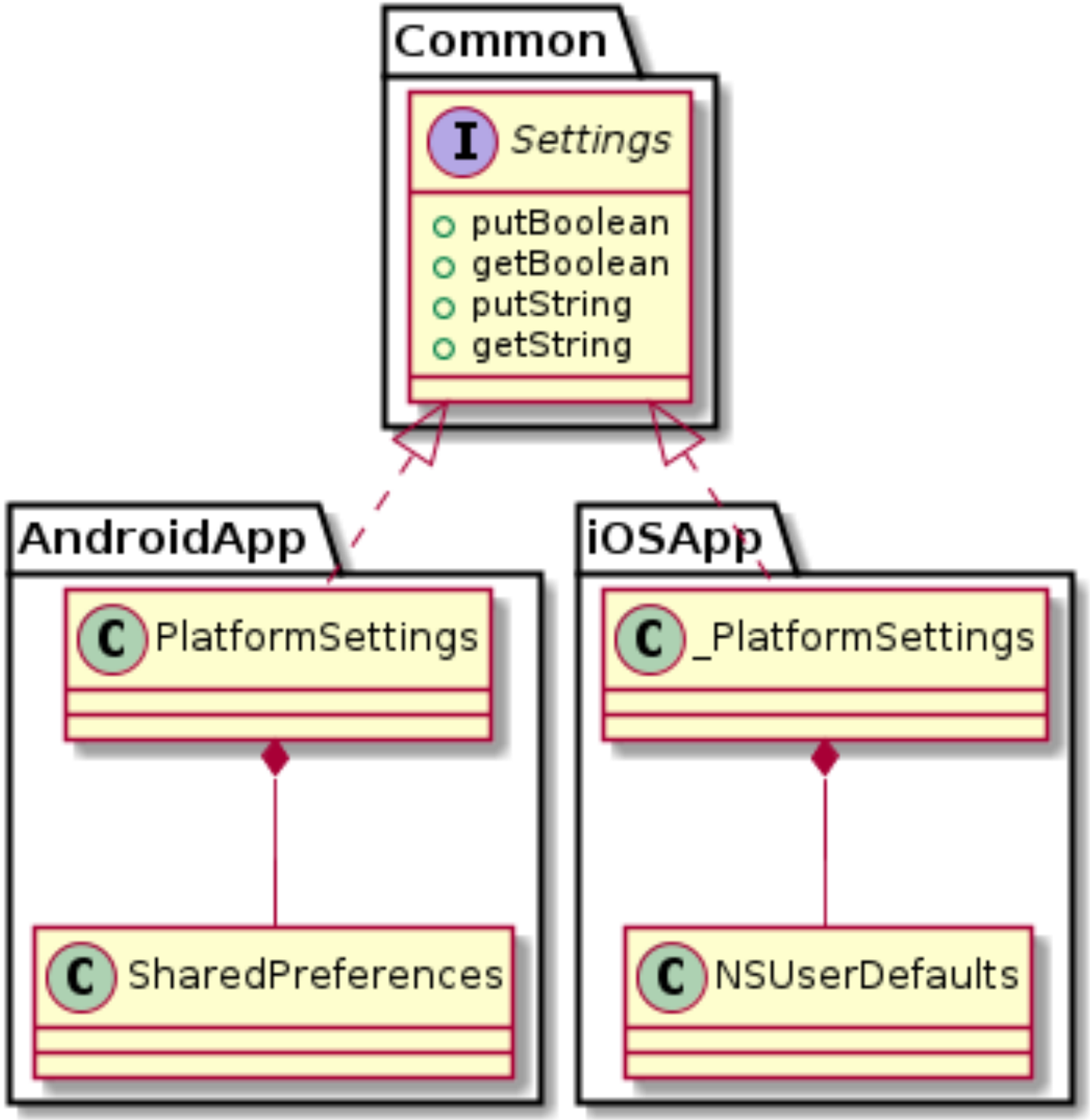
DATE AND TIME

- mockable *TimeProvider* for testing
- **JodaTime** originally
- Migrated to **ThreeTen** later
- **GMTDate**
- POSIX

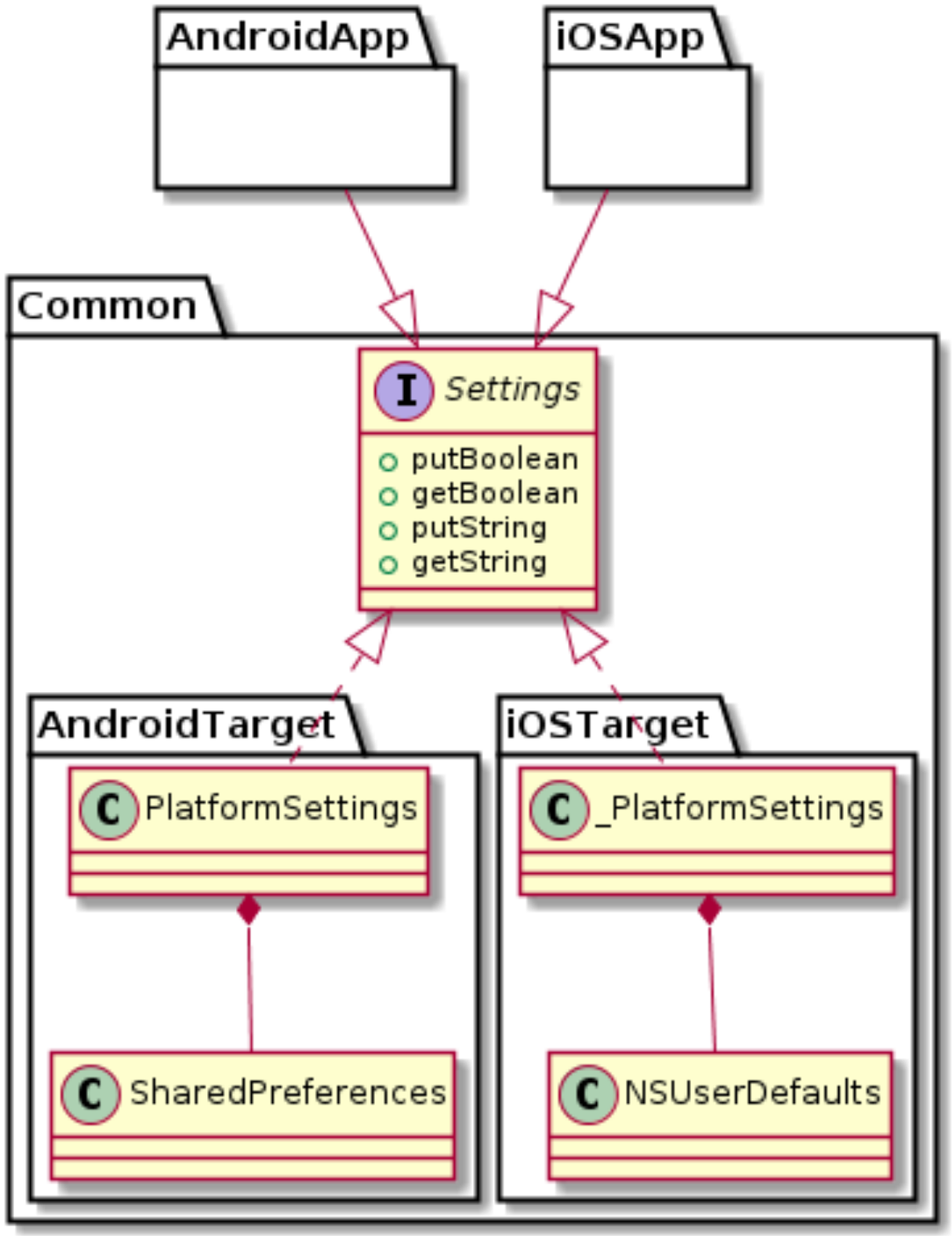
LOCAL CACHE

- Ktor serialised gson objects
- shared preferences on Android
- NSUserDefaults

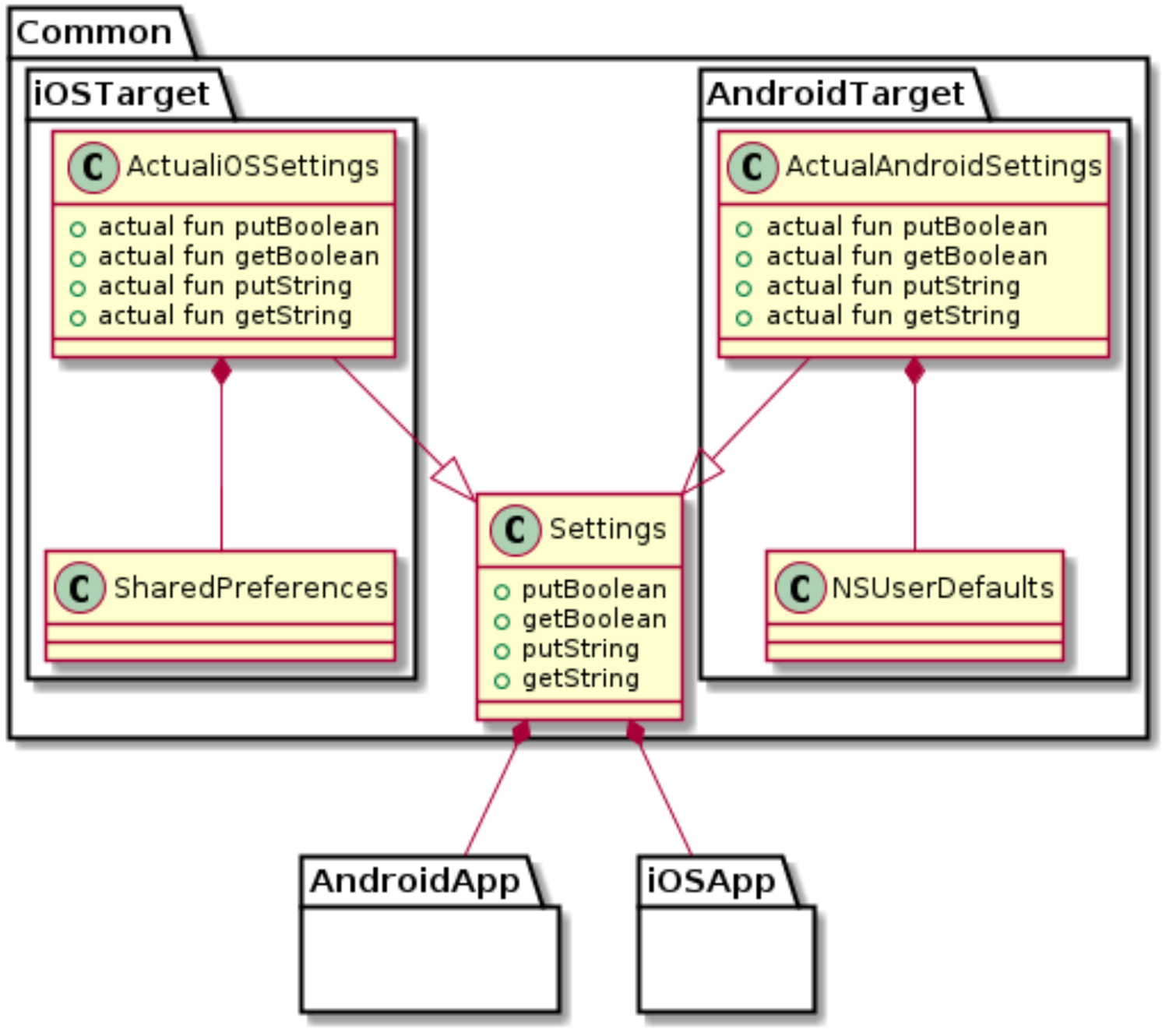
INTERFACES


Common
Settings
putBoolean
getBoolean
putString
getString
AndroidApp
PlatformSettings
SharedPreferences
iOSApp
_PlatformSettings
NSUserDefaults

INTERFACE MULTIPLATFORM TARGET



EXPECTED, ACTUAL



IOS

IOS

- added support into AppCode 2019.1
- Xcode Kotlin Multiplatform by **Touchlab**
- support for cocoapods
- no syntax for **suspended** in swift

BEFORE COCOAPODS

```
task packForXCode(type: Sync) {
    final File frameworkDir = new File(buildDir, "xcode-frameworks")
    final String mode = project.findProperty("XCODE_CONFIGURATION")?.toUpperCase() ?: 'DEBUG'
    final def framework = kotlin.targets.ios.binaries.getFramework("shared", mode)

    inputs.property "mode", mode
    dependsOn framework.linkTask

    from { framework.outputFile.parentFile }
    into frameworkDir

    doLast {
        new File(frameworkDir, 'gradlew').with {
            text = "#!/bin/bash\nexport 'JAVA_HOME=${System.getProperty("java.home")}'\ncd
'${rootProject.rootDir}'\n./gradlew \$@\n"
            setExecutable(true)
        }
    }
}

tasks.build.dependsOn packForXCode
```

WITH COCOAPODS

```
cocoapods {
        summary = "Shared common library"
        homepage = "https://github.com/dukecon"
}
```

COCOAPOD FILE

```
spec.script_phases = [
  {
      :name => 'Build dukecon',
      :execution_position => :before_compile,
      :shell_path => '/bin/sh',
      :script => <<-SCRIPT
          set -ev
          REPO_ROOT="$PODS_TARGET_SRCROOT"
          "$REPO_ROOT/../../gradlew" -p "$REPO_ROOT" :common:dukecon:syncFramework \
              -Pkotlin.native.cocoapods.target=$KOTLIN_TARGET \
              -Pkotlin.native.cocoapods.configuration=$CONFIGURATION \
              -Pkotlin.native.cocoapods.cflags="$OTHER_CFLAGS" \
              -Pkotlin.native.cocoapods.paths.headers="$HEADER_SEARCH_PATHS" \
              -Pkotlin.native.cocoapods.paths.frameworks="$FRAMEWORK_SEARCH_PATHS"
      SCRIPT
  }
]
```

NEXT STEPS

NEXT STEPS

- Use BOMs (ktor, coroutines)
- separate features to modules
- migrate *gradle.build* to **kts**
- Clean architecture on iOS
  - VIPER
  - CleanSwift

SMALL ADVICES

- Start small
- Look for pragmatic solution (DateTime)
- make it a home game on target host
  - POSIX on iOS and MacOSX
- create and use BOM for **Klibs**
- **use** official documentation

TAKEAWAYS

DEVELOPING WITH KOTLIN MULTIPLATFROM

- You can use Android Studio or InteliJ to create a reusable KMP component which can be imported into an Xcode project.
- Because similarities between Swift's and Kotlin's syntax code written in Kotlin can be easy read by iOS developers.
- KMP is **not** the final step to accomplishing 100% shared code. UI logic must still be programmed natively for every platform.

DEVELOPING WITH KOTLIN MULTIPLATFROM

- Using Kotlin Multiplatform, you can avoid repeating lots of logic to develop an app running on multiple platforms.
- Proper architecture can increase an ratio of shared code

THANK YOU

LINKS

- https://kotlinlang.org/docs/tutorials/native/mpp-ios-android.html
- https://github.com/touchlab/DroidconKotlin
- https://github.com/JetBrains/kotlinconf-app
- https://dukecon.org

CREDITS

- all photos 2019 (c) Adriana Harakalova