OR HOW I WAS REWRITING A CONFERENCE APP OVER WEEKENDS

KOTLIN MULTIPLATFORM ROLLER-COASTER

CHALLENGE

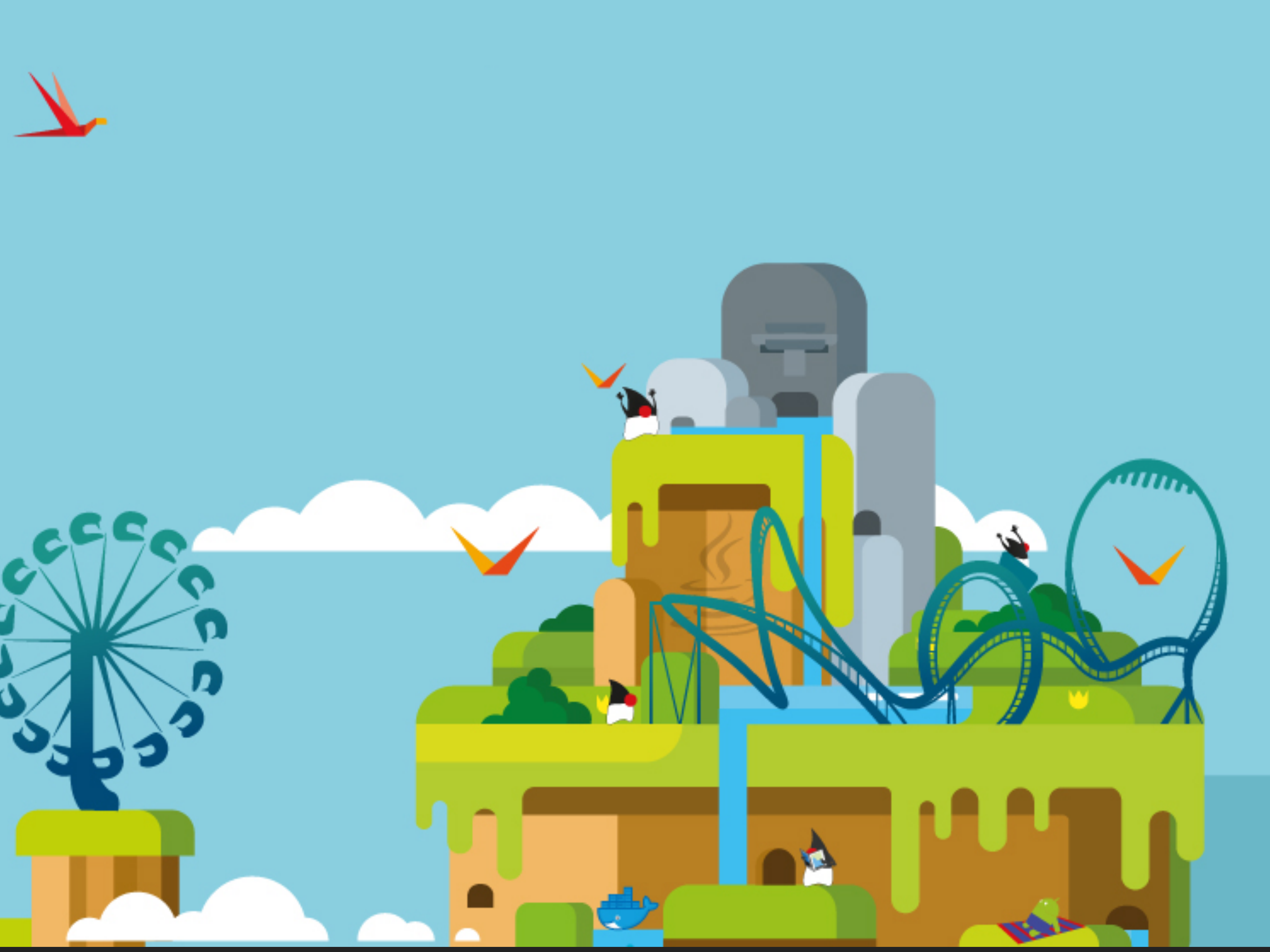
DEUTSCHE TELEKOM AG

MICHAL HARAKAL

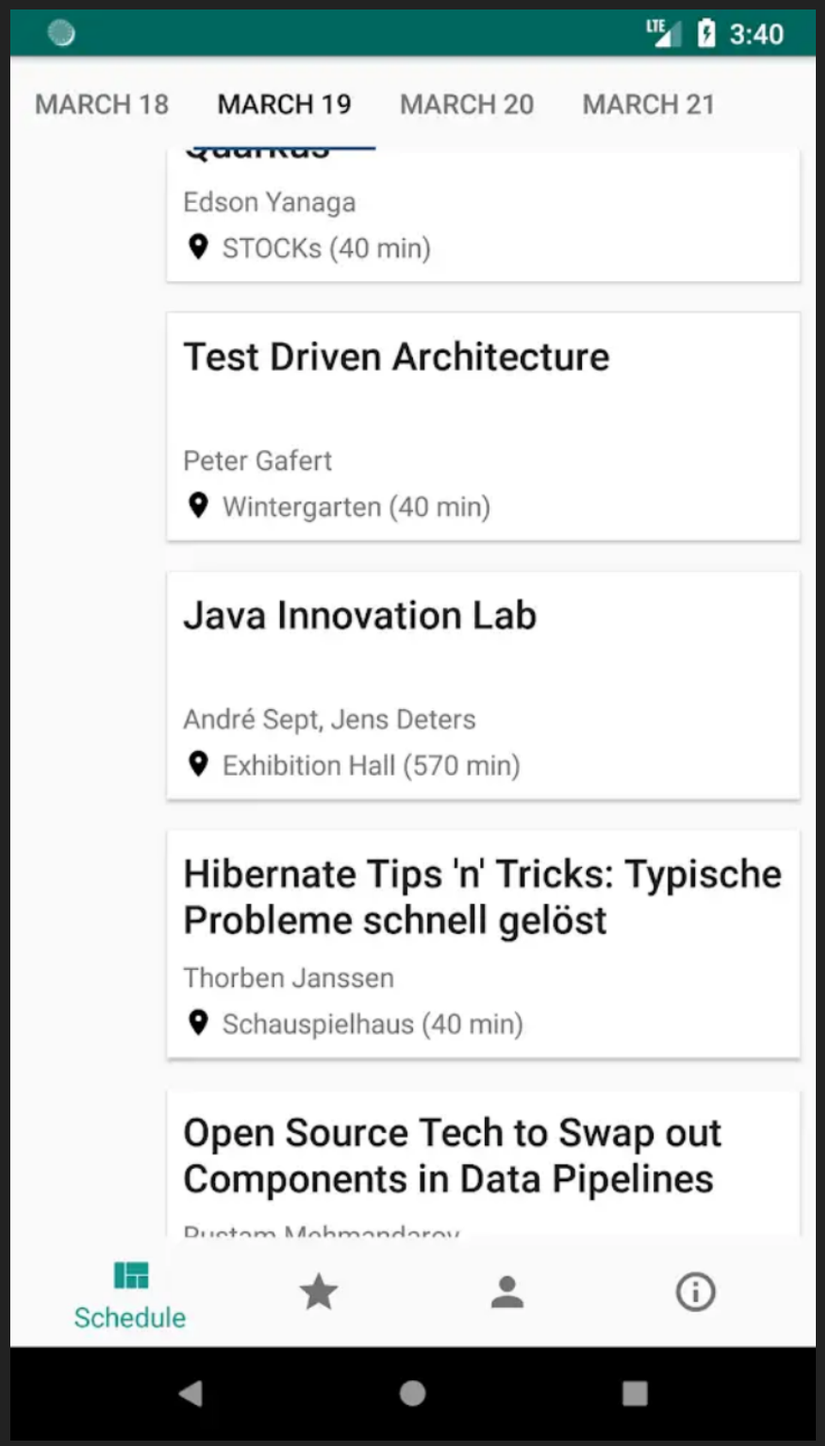
ROLLER-COASTER

- 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



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
- finest selection of Android libs(okhttp, retrofit, picasso, gson)

DUKECON.ORG

JavaLand

DOAG

DOAG

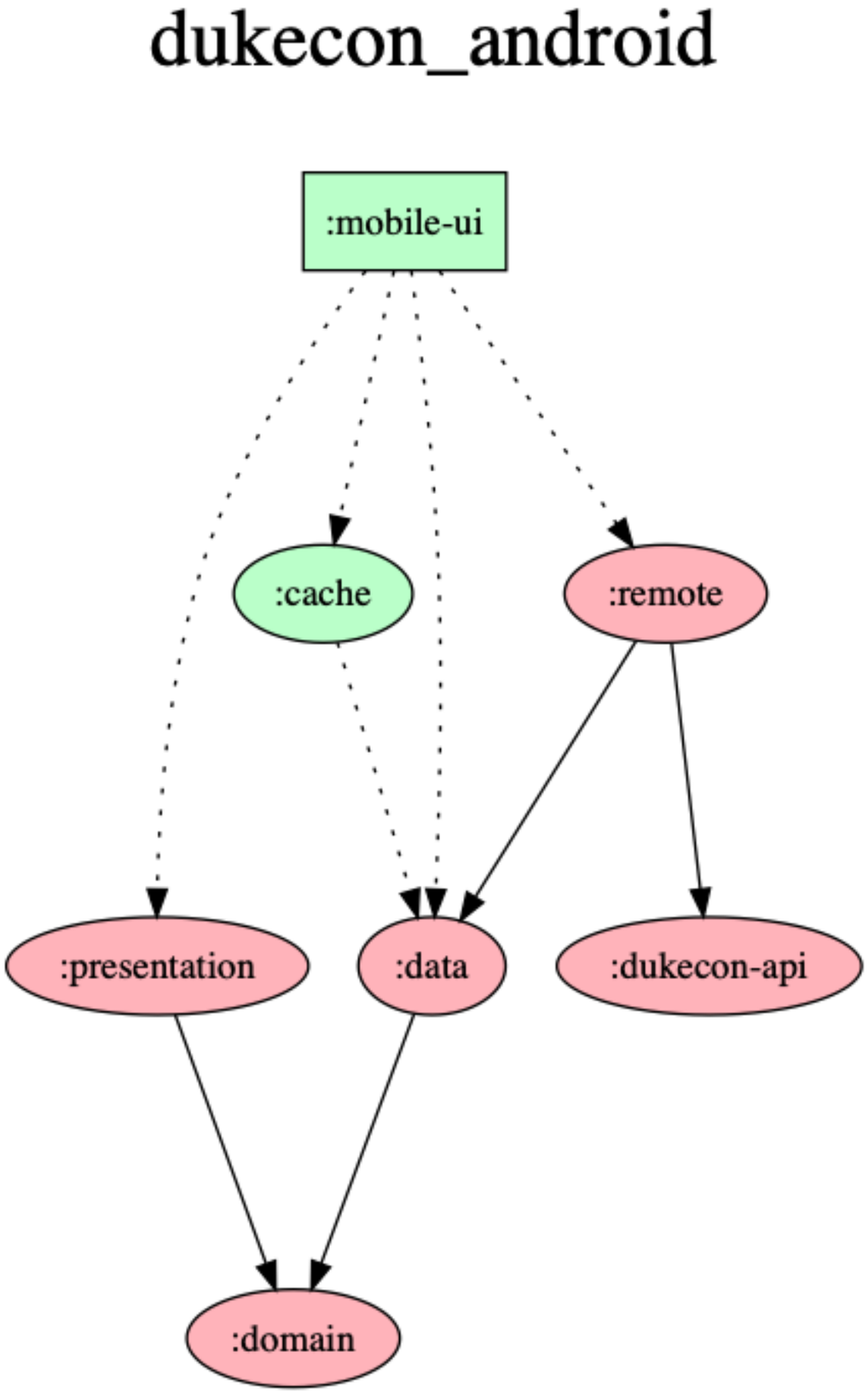
Apex

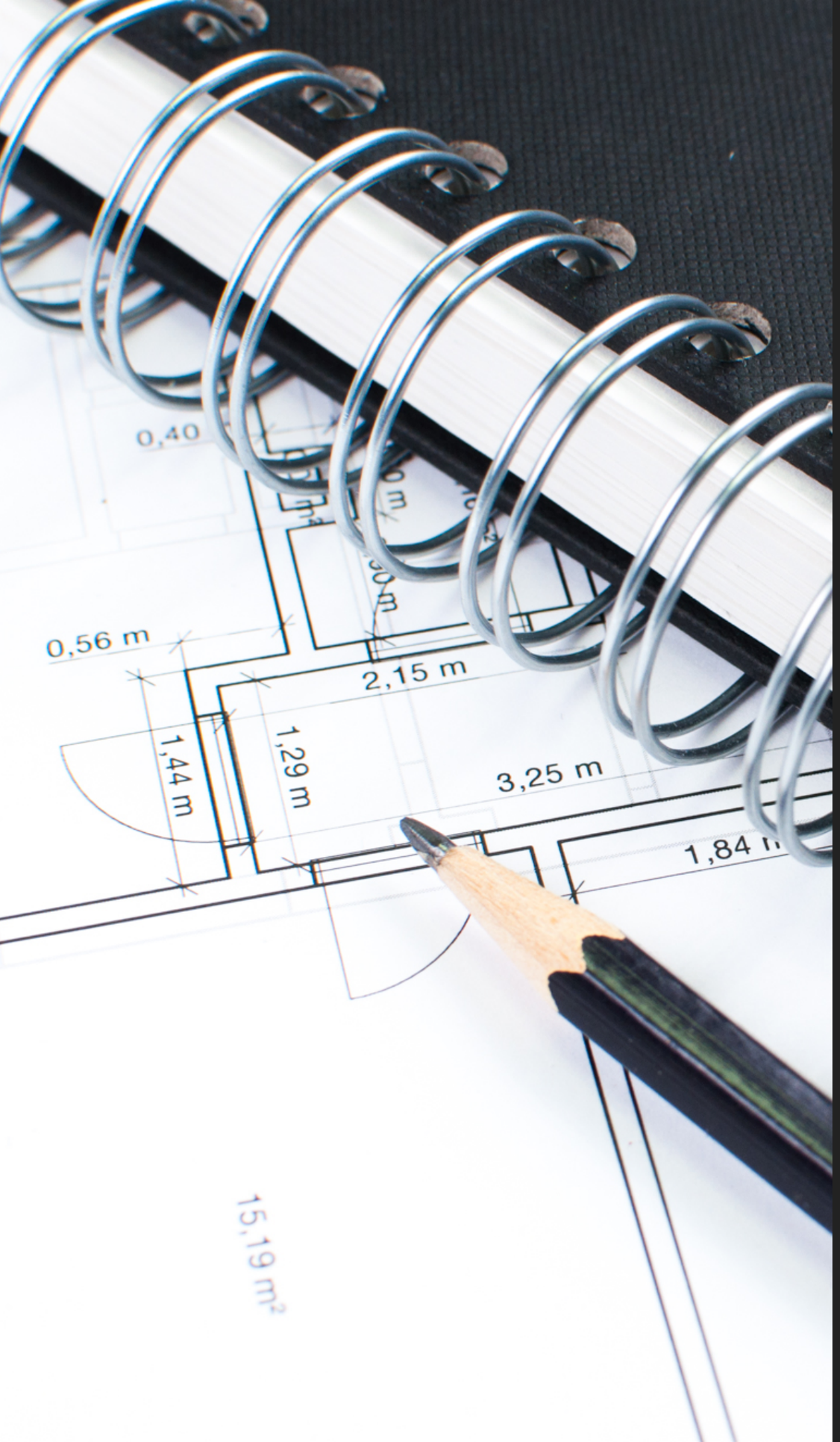
APEX connect

APACHECON

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

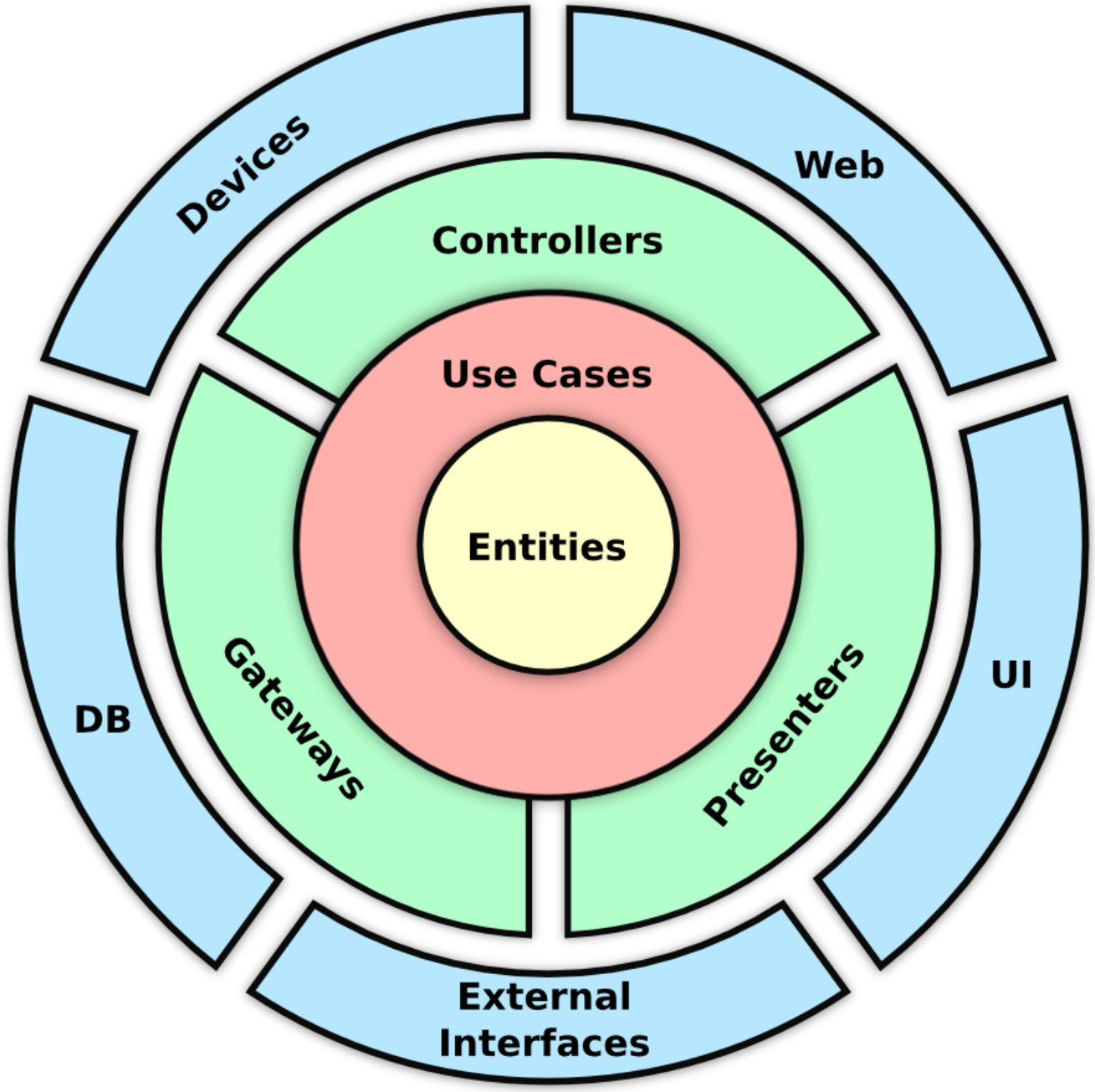
- Fokus on Android first
- convert java modules to Kotlin Multiplatform
- migrate platform dependent to Kotlin Multiplatform (**remote**, **cache**)
- use **Kotlinkonf** app for iOS

STANDING ON THE SHOULDERS OF GIANTS

STANDING ON THE SHOULDERS OF GIANTS

- Chicago Roboto
- Kotlin clean architecture starter by **Joe Birch**
- Dukecon.org
- **Kotlinkonf**
- SDKSearch by **Jake Wharton**

CLEAN ARCHITECTURE

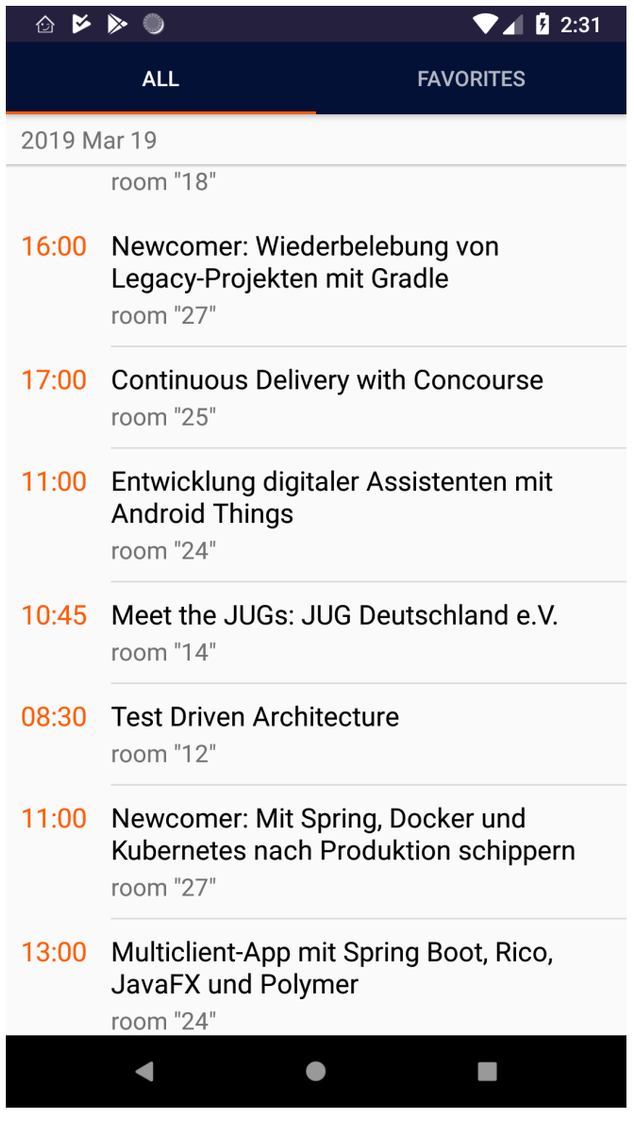


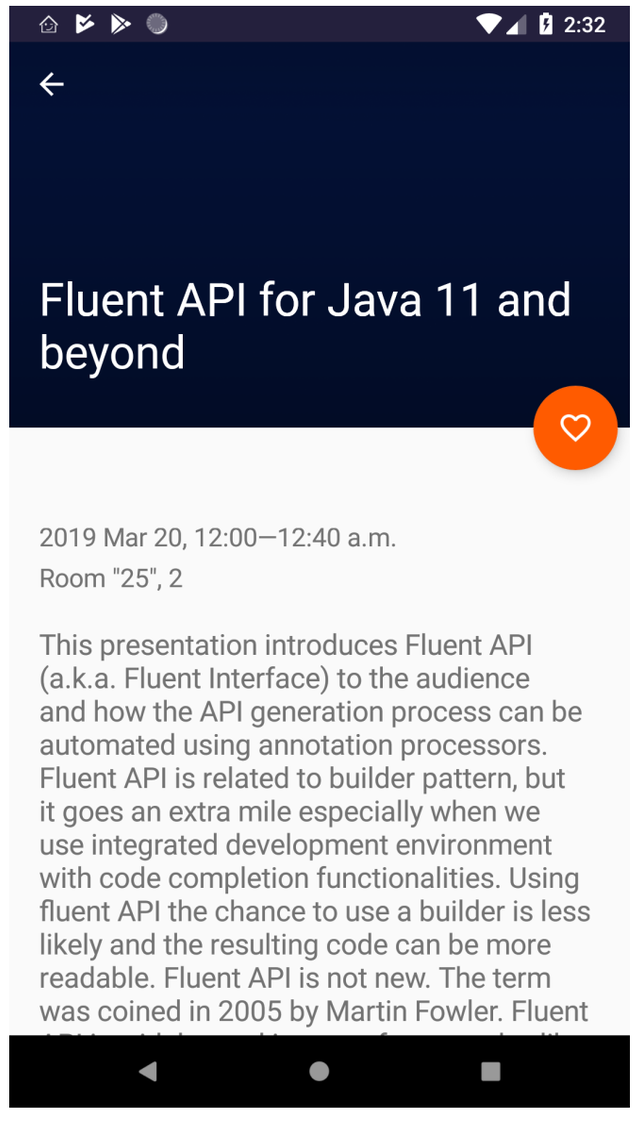
- layered
- dependecies points inwards
- keeps Android dependencies on outside

CHALLENGE DIARY

WEEKEND 1 – KOTLINKONF SPEAKS DUKECON

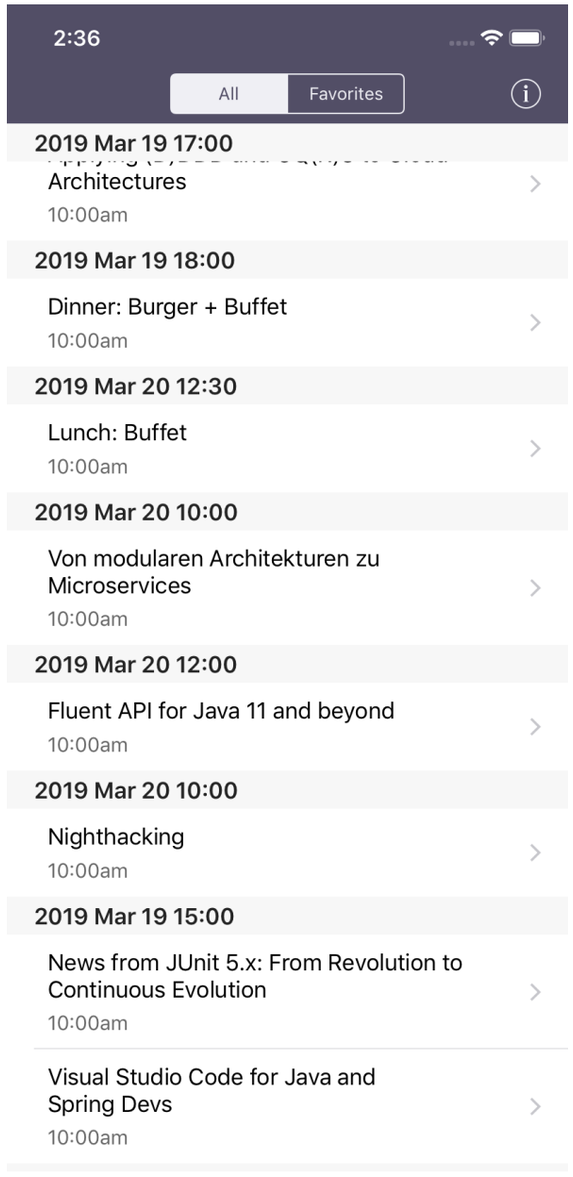
- swagger generated remote client and data object
- Android running

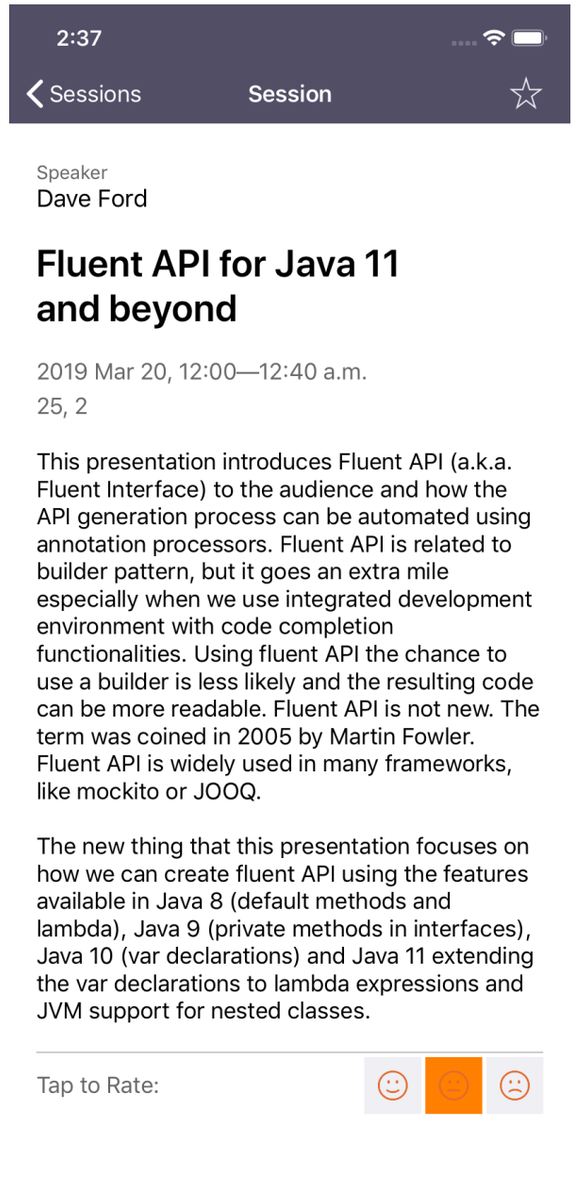




WEEKEND 1 – IOS SPEAKS DUKECON

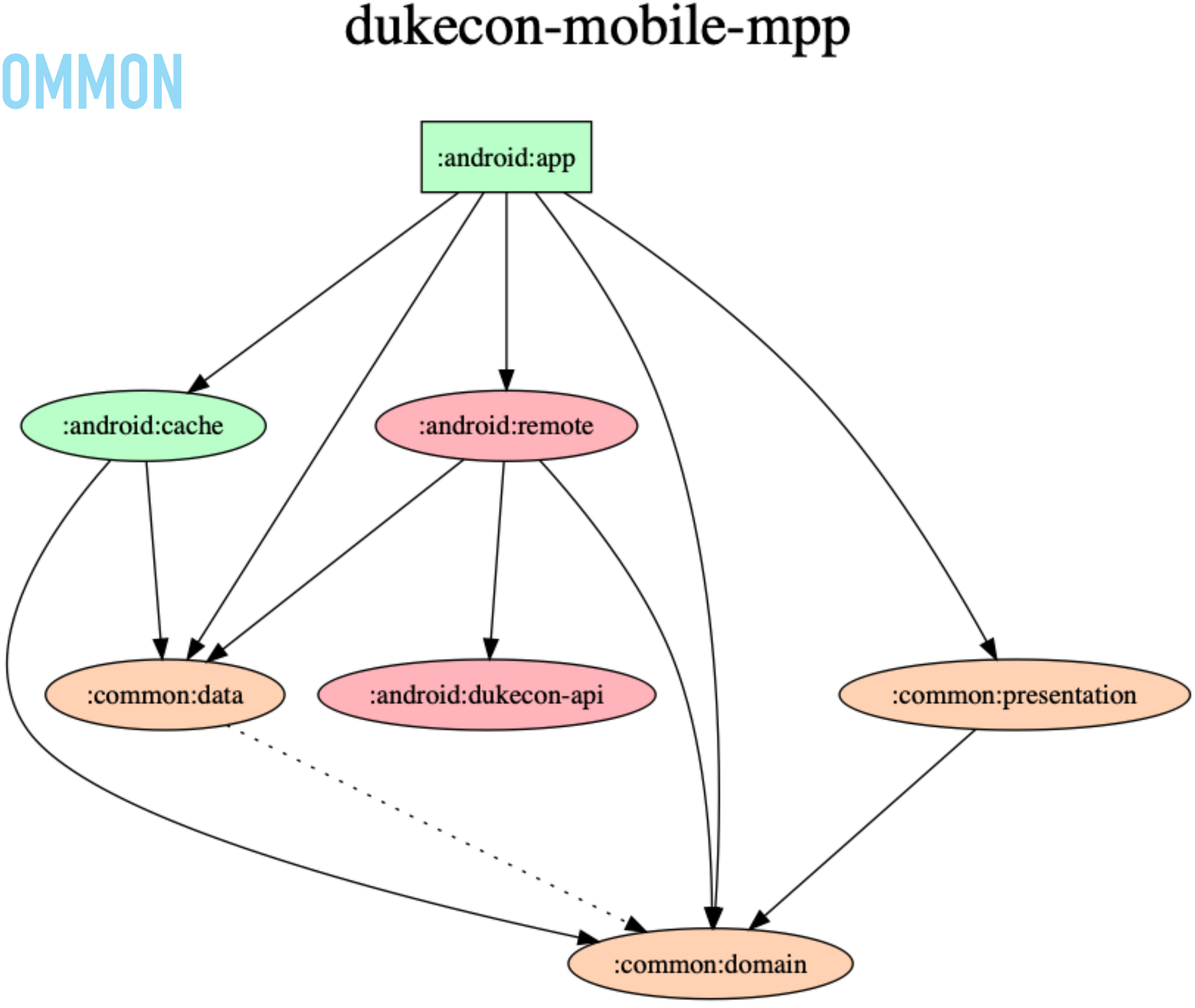
- how to debug?
- iOS specifics wisdom required
- gradle vs. **build.sh**
- *AppCode*





WEEKEND 2 – COMMON

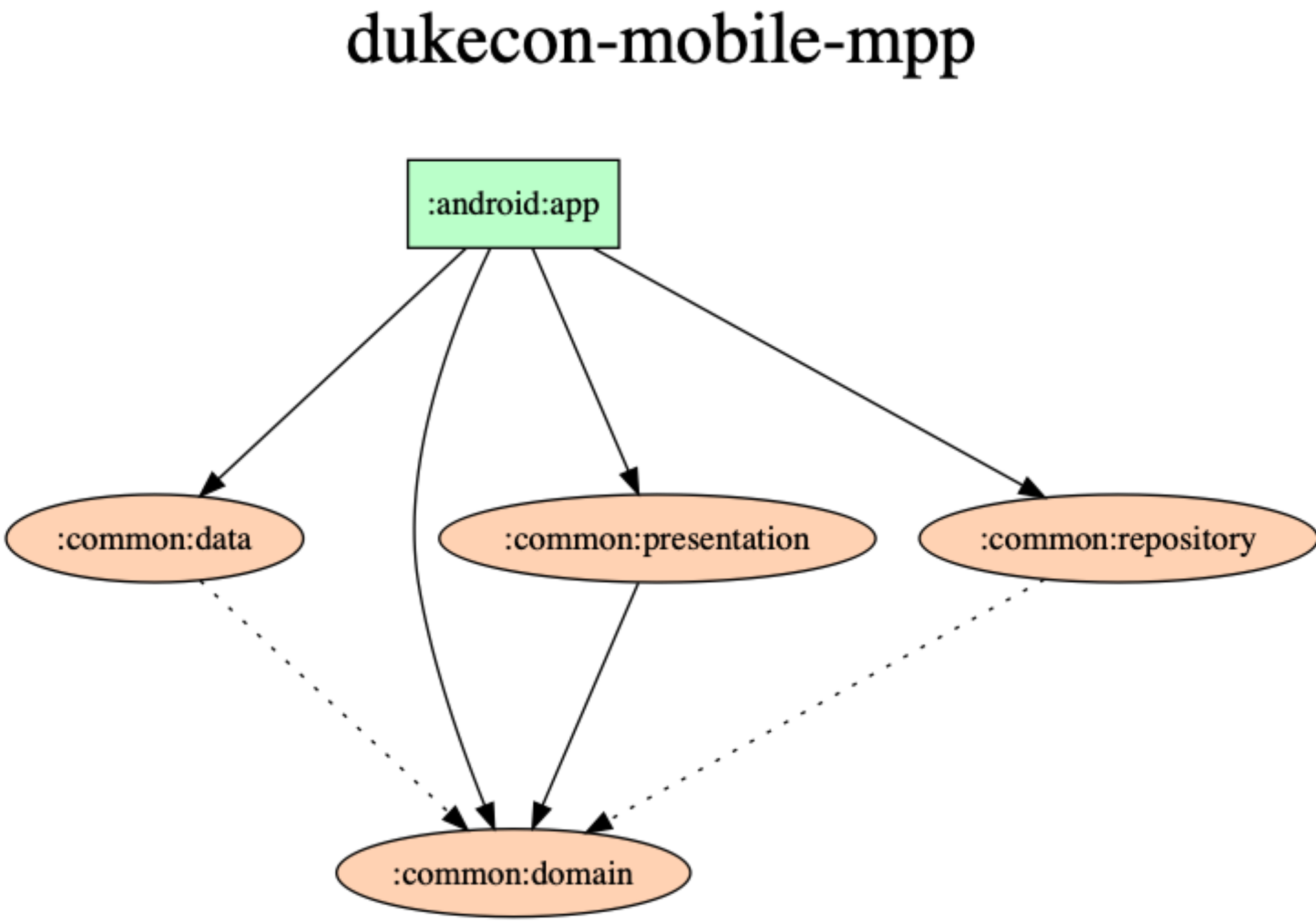
- moving java only, platform independent classes into Kotlin MPP
- removing code related to a *network awareness*
- removing **@Inject** annotations



WEEKEND 3 – FINAL WEEKEND

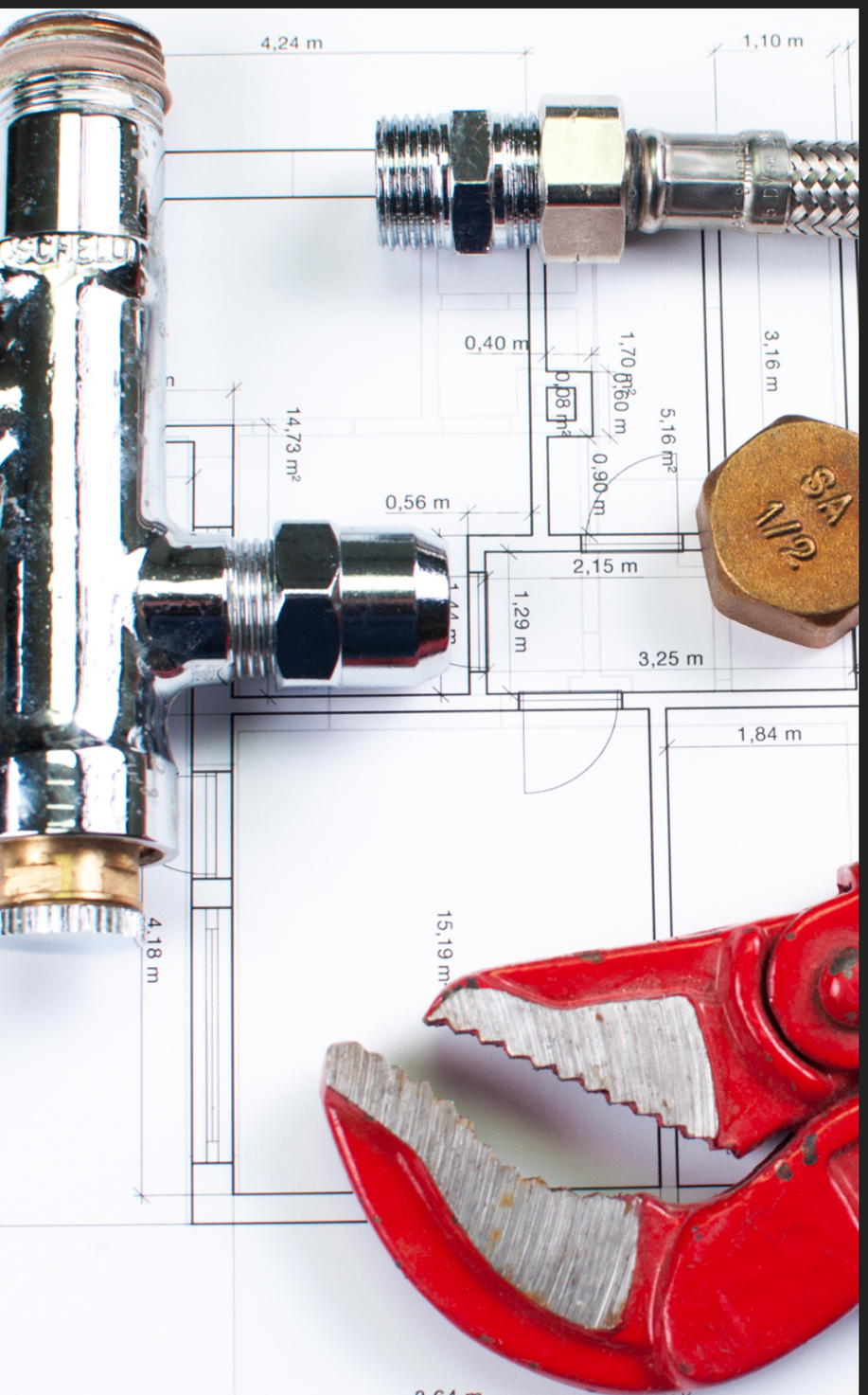
- Android App runs with Kotlin MPP Module
- still issues with Date Time
- original heavy usage of constructor injected parameters has to be replaced by manual calls in Dagger module classes

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

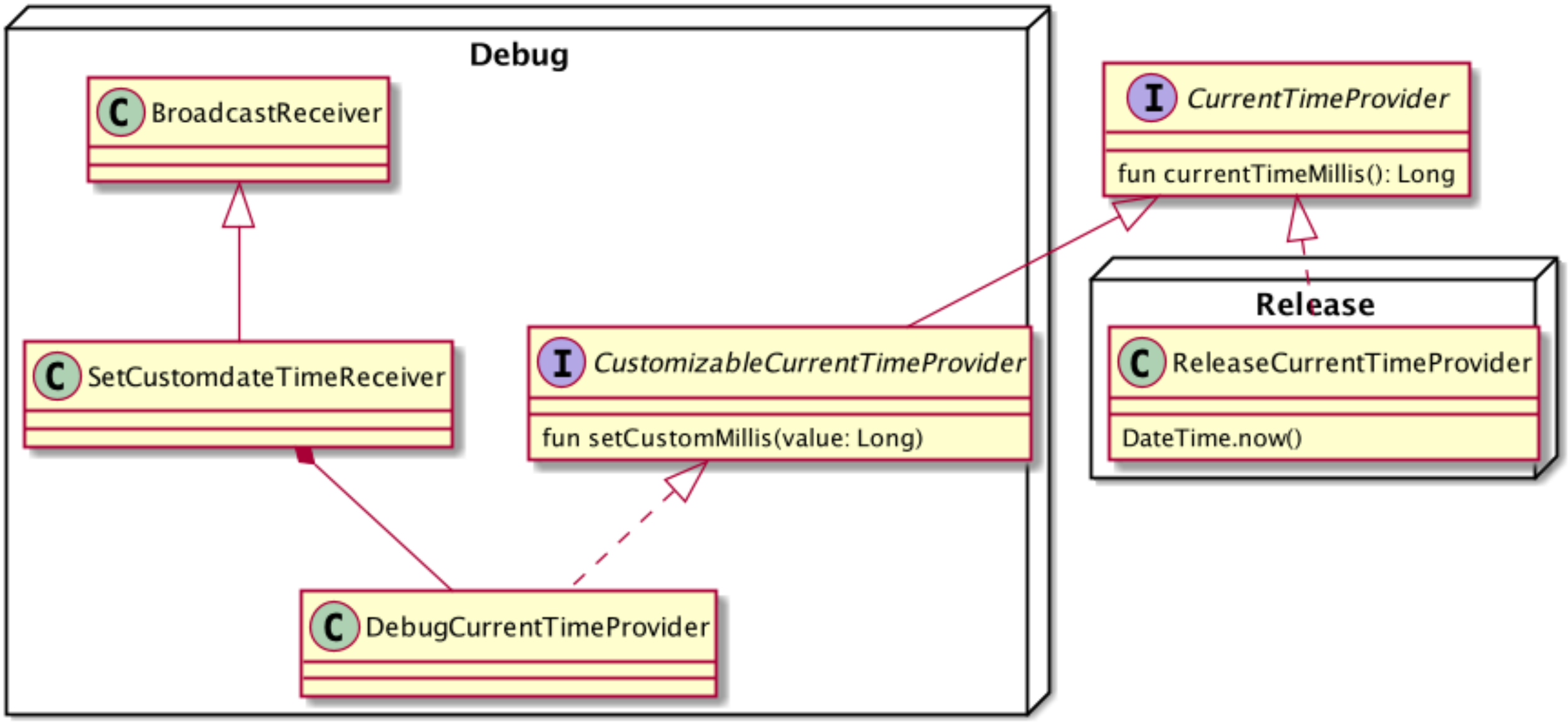
Okhttp by default, SSL not working on 4.4

DATE AND TIME

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

```
$ adb shell am broadcast -a
org.dukecon.android.ui.intent.TIME --es set_time "12:33:00"
```

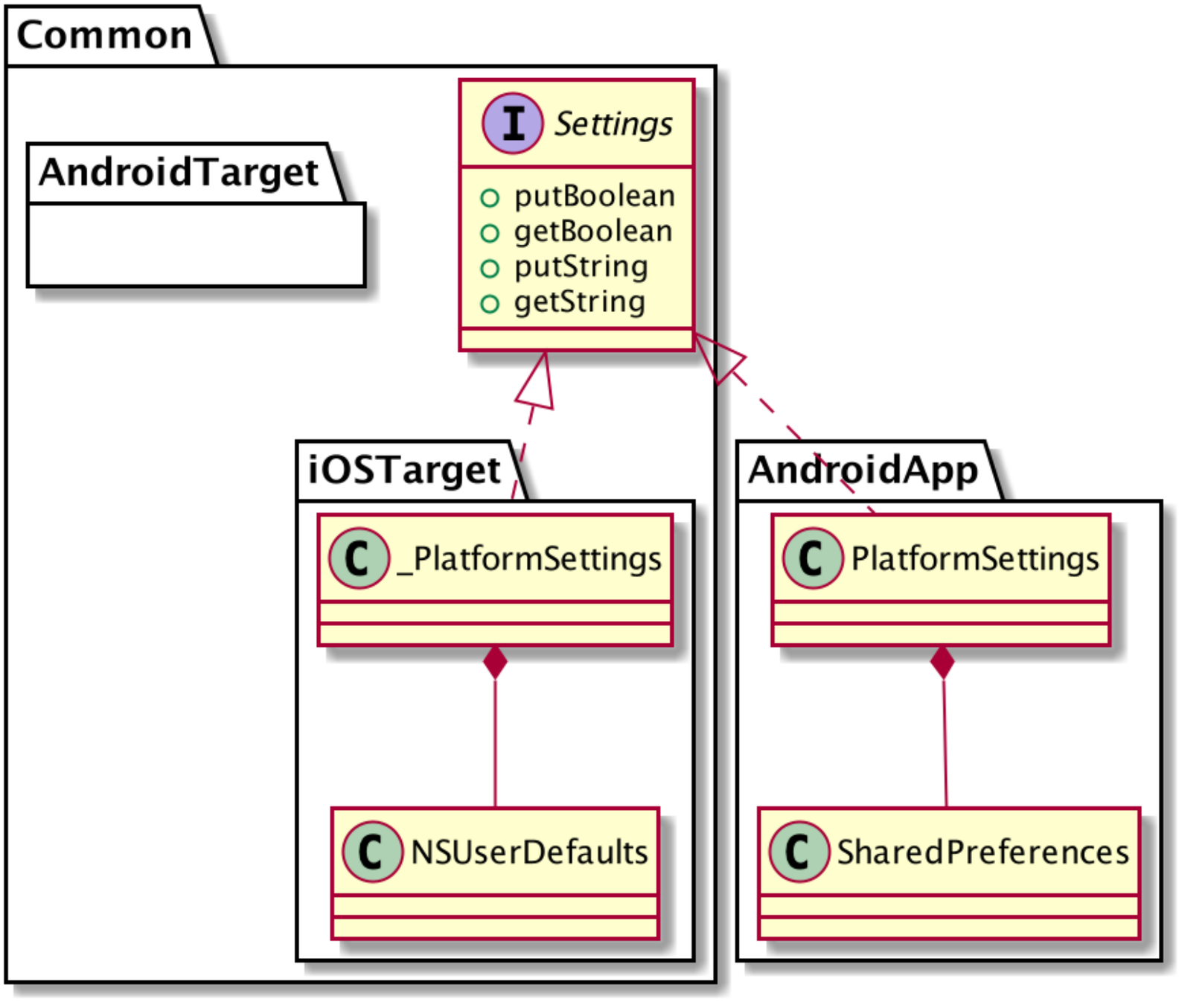
CURRENT TIME PROVIDER



LOCAL CACHE

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

SETTINGS


Common
AndroidTarget
Settings
putBoolean
getBoolean
putString
getString
iOSTarget
_PlatformSettings
NSUserDefaults
AndroidApp
PlatformSettings
SharedPreferences

IOS

- added support into AppCode 2019.1 via Plugin
- Xcode Kotlin Multiplatform by **Touchlab**

RESULT

LESSONS LEARNED

- Dagger
- iOS support
- Software architecture
- bringing Kotlin Multiplatform into real life projects, code sharing with iOS is the key

DOWNS

- MPP project is hard to setup
  - RTFM
  - clear cache
    - restart IDE
- iOS support
  - no debuging
- DI

UPS

- Android Prototype is running
- Promising current events on iOS
- Chance to use it in real projects already now
- Kotlin is just cool