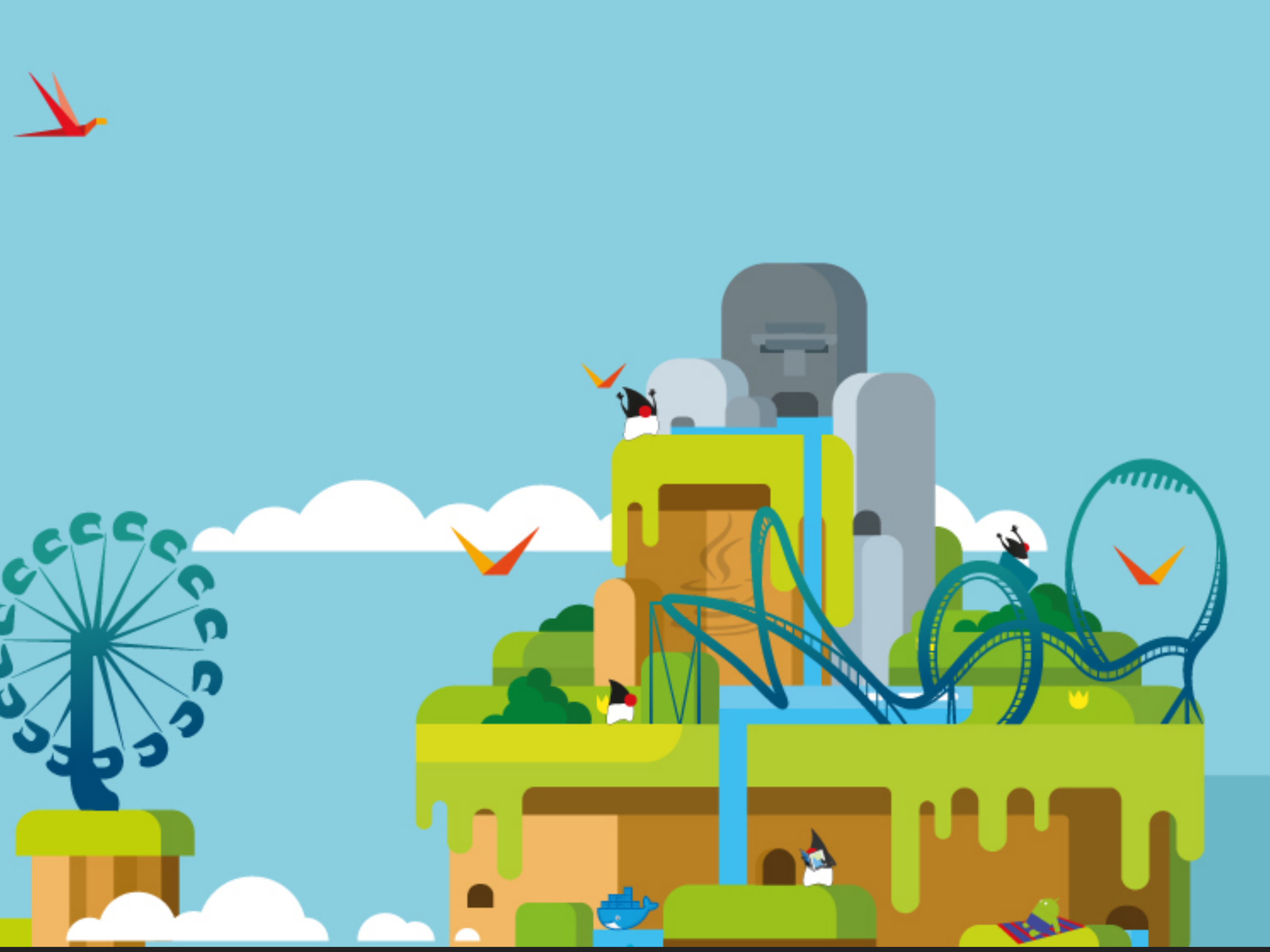
OR HOW I WAS REWRITING A CONFERENCE APP OVER WEEKENDS

KOTLIN MULTIPLATFORM ROLLER-COASTER

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



DEUTSCHE TELEKOM AG

MICHAL HARAKAL

DUKECON.ORG

JavaLand

DOAG

DOAG

Apex

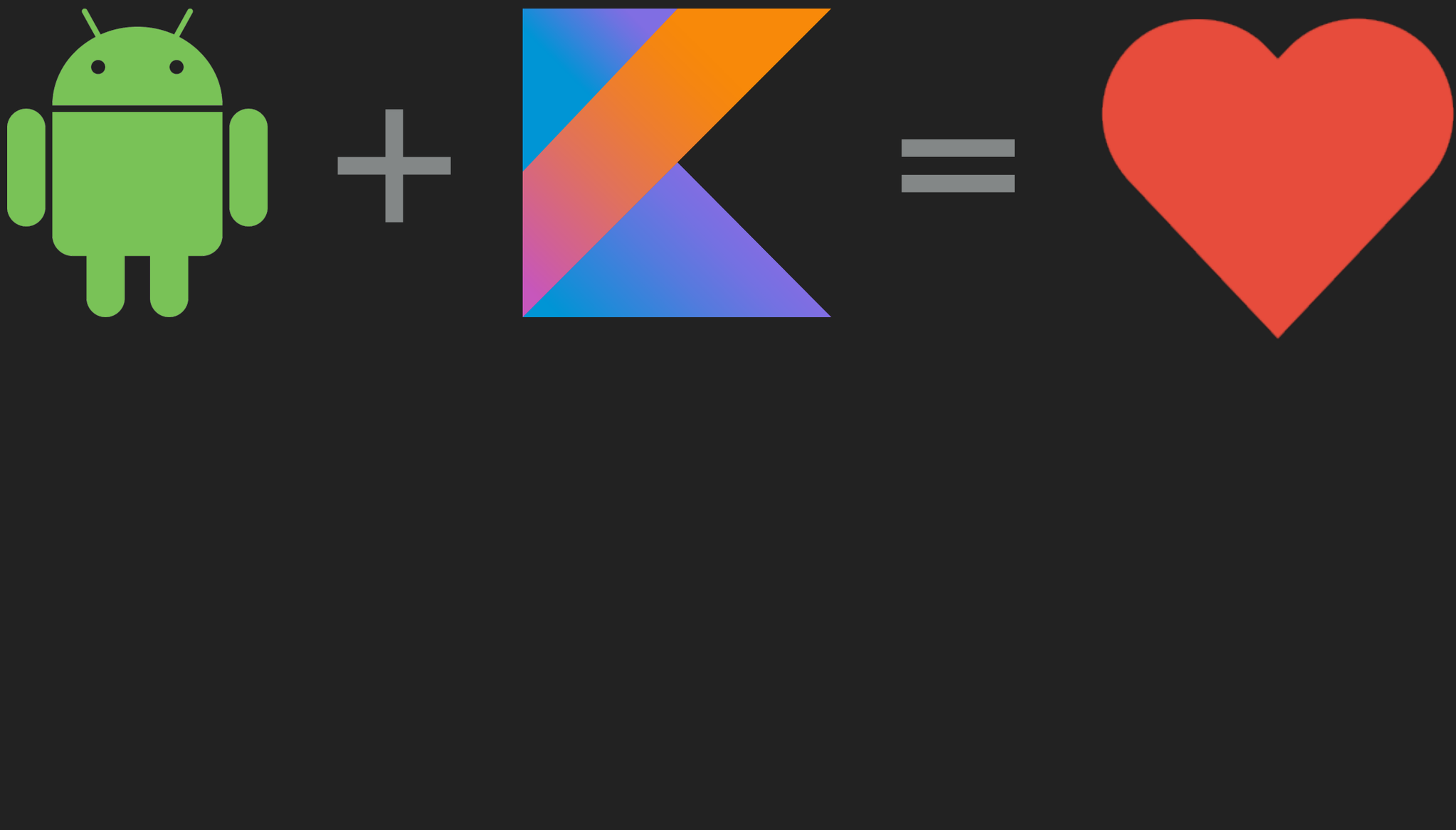
APEX connect

APACHECON

QUESTION

ANSWER

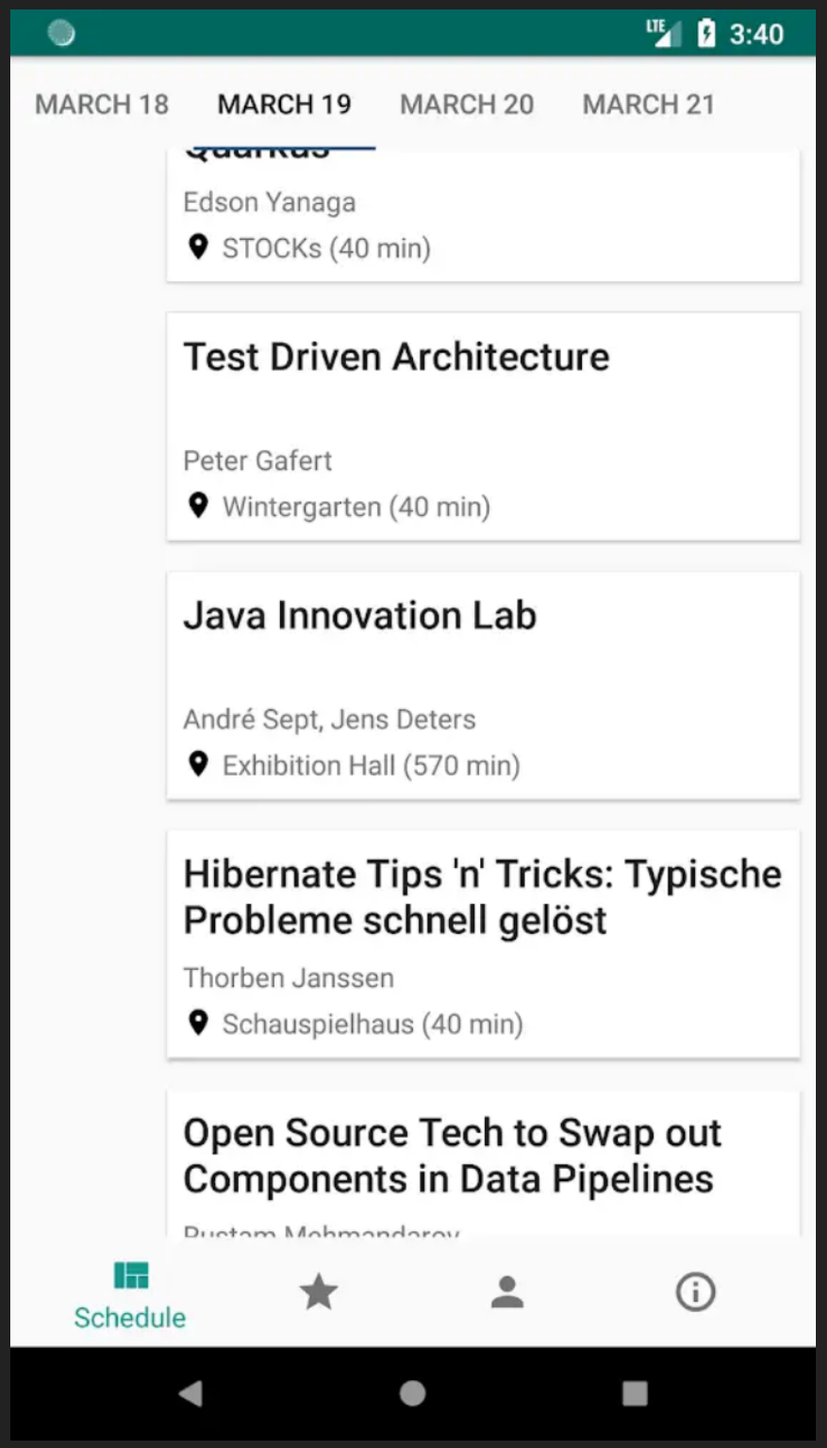
CODE SHARING IN KOTLIN



LIVE DEMO

CHALLENGE

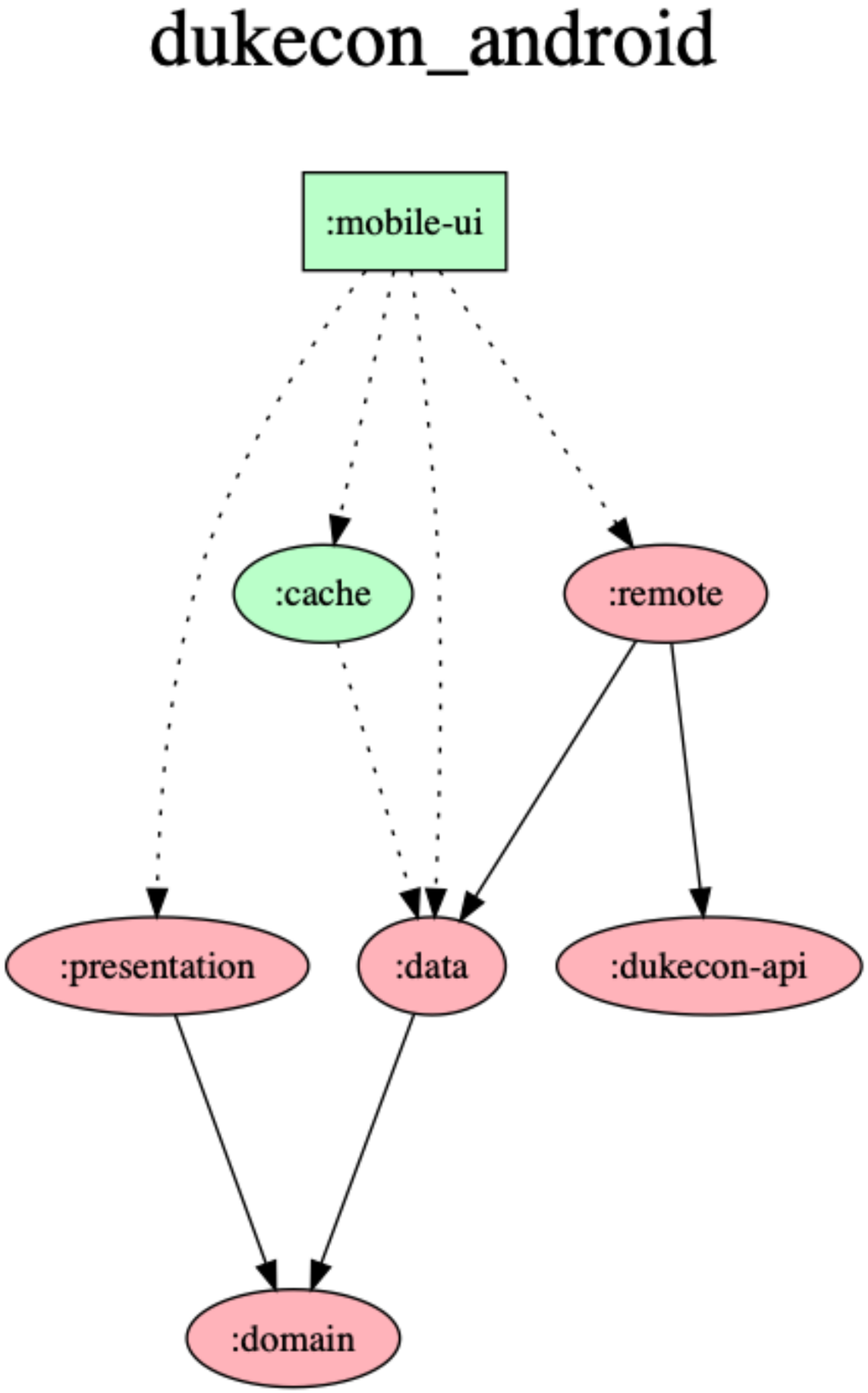
STARTING POSITION – JAVALAND NATIVE ANDROID CLIENT

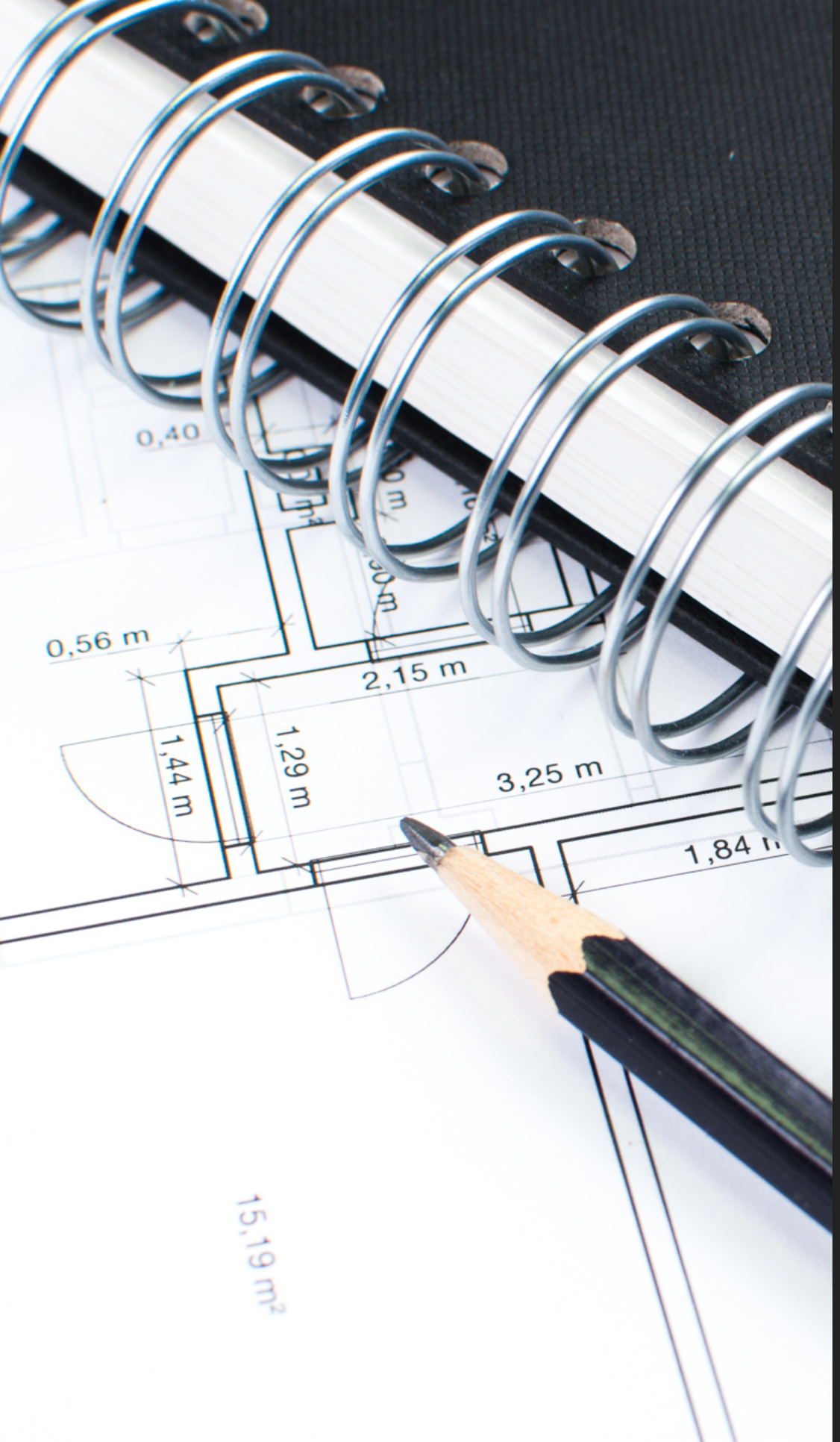


- a part of dukekon.org OSS project
- written 100% in Kotlin
- clean architecture approach with building blocks as separated gradle modules
- coroutines
- MVP for presentation layer
- as far possible, java only, to get rid of Android whenever you can
- current Android libs stack (okhttp, retrofit, picasso,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

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

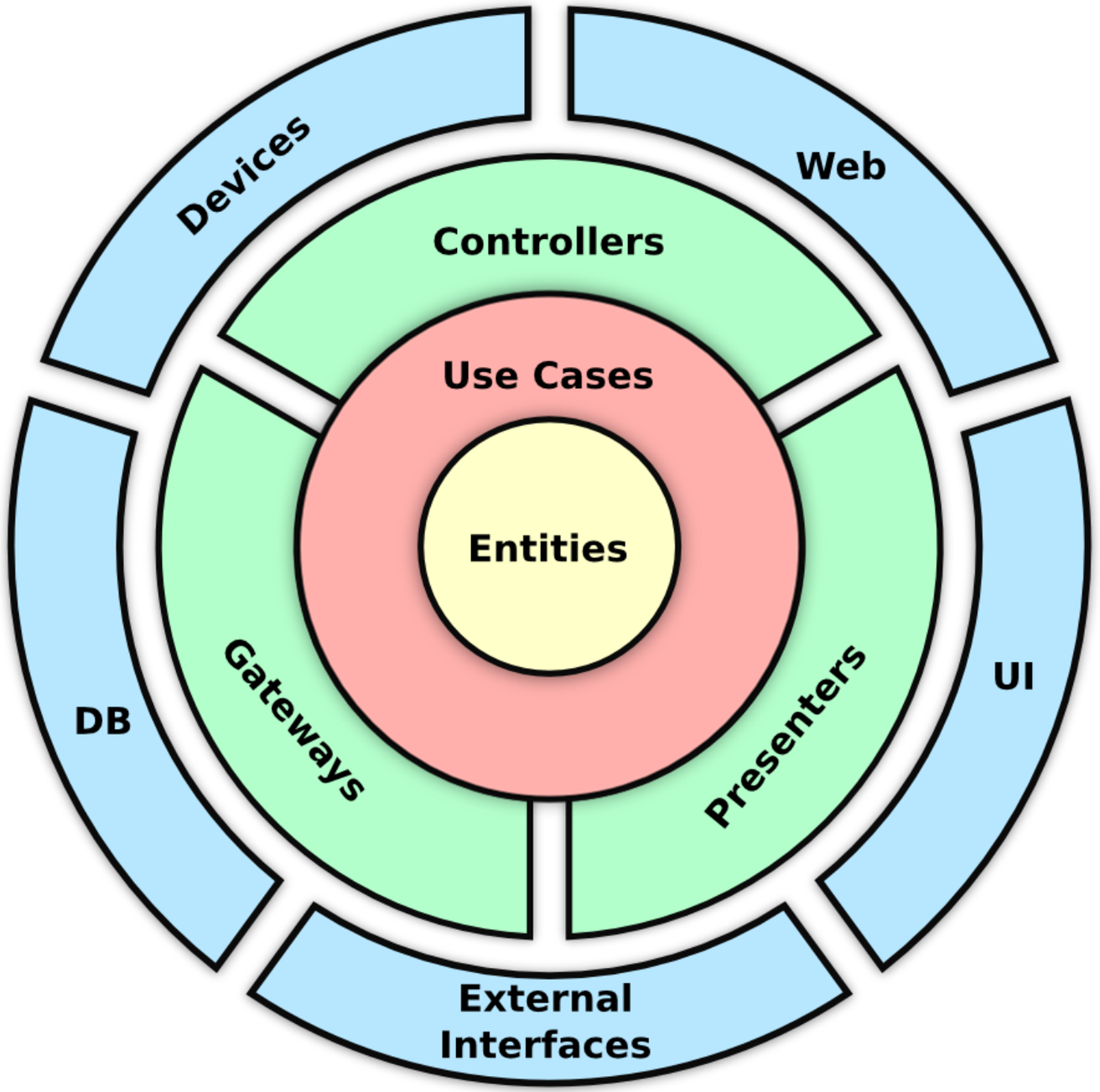
STANDING ON THE SHOULDERS OF GIANTS

STANDING ON THE SHOULDERS OF GIANTS

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

ARCHITECTURE

CLEAN ARCHITECTURE

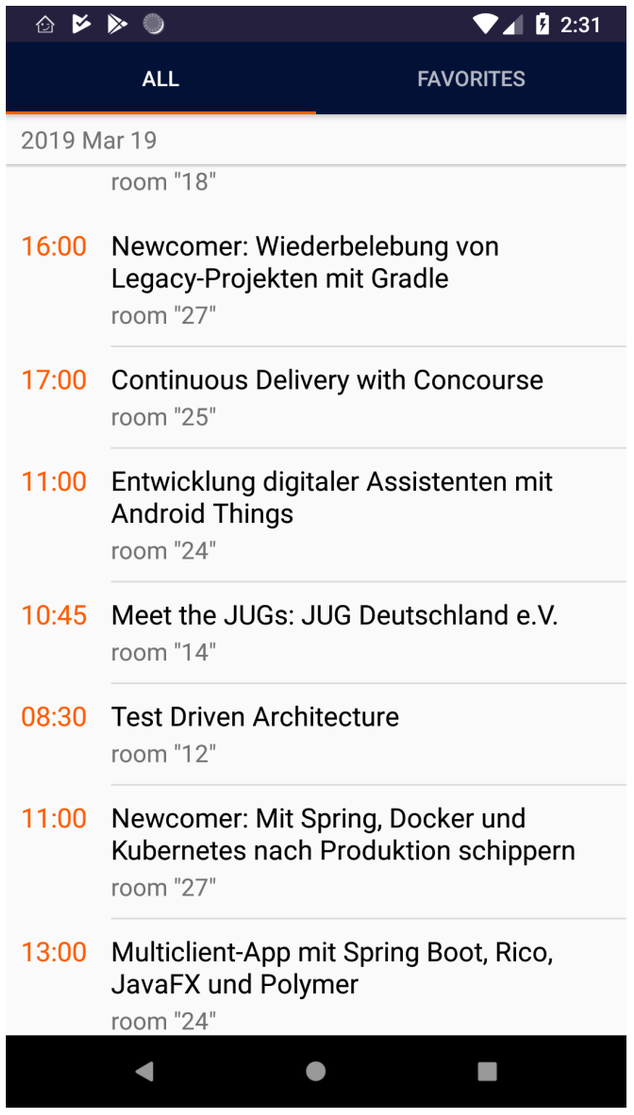


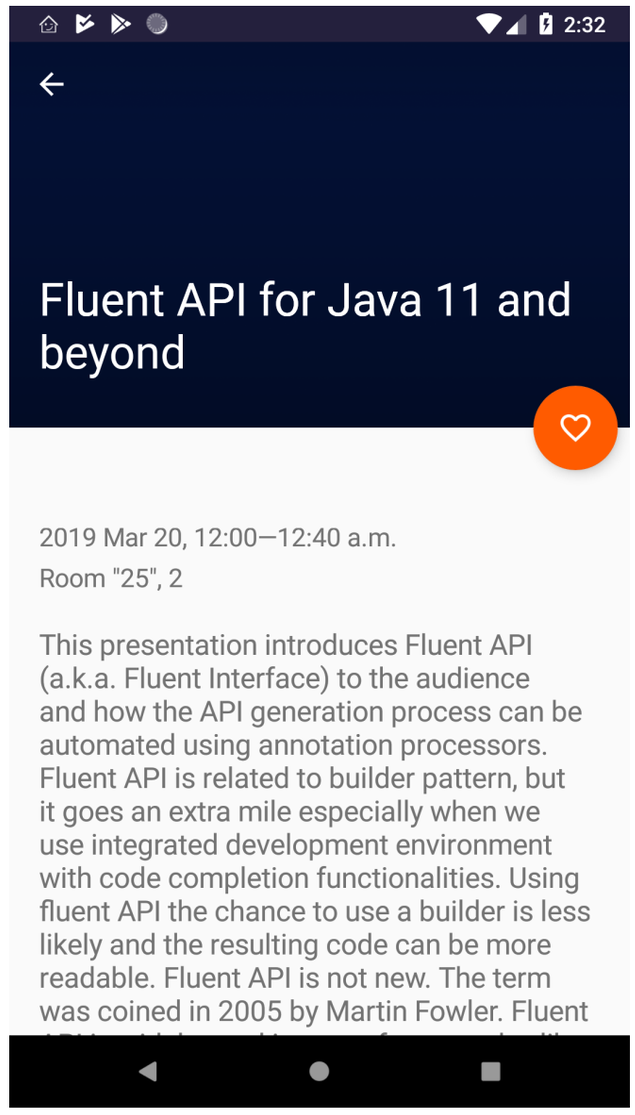
- layered
- dependencies points inwards
- keeps Android dependencies on outside

CHALLENGE DIARY

WEEKEND 1 – KOTLINKONF SPEAKS DUKECON

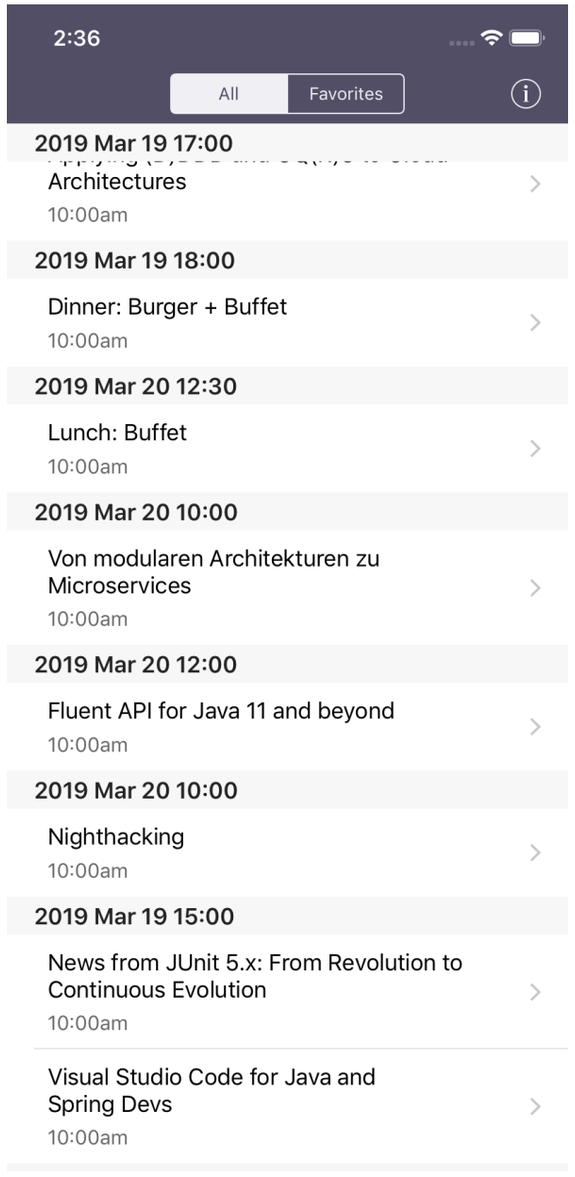
- swagger generated remote client and data object
- Android running

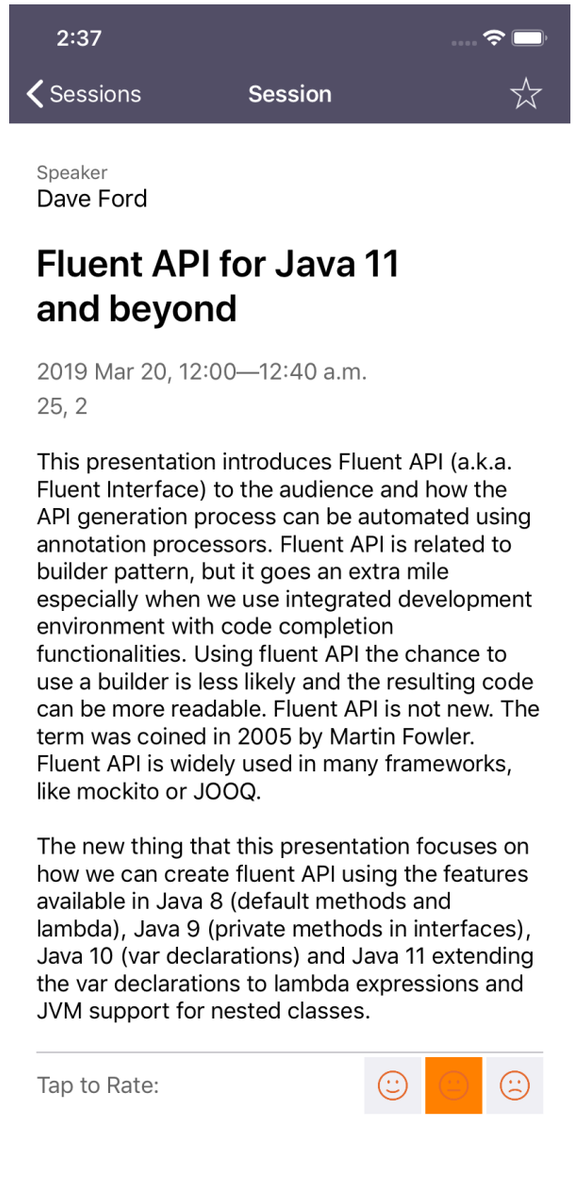




WEEKEND 1 – IOS SPEAKS DUKECON

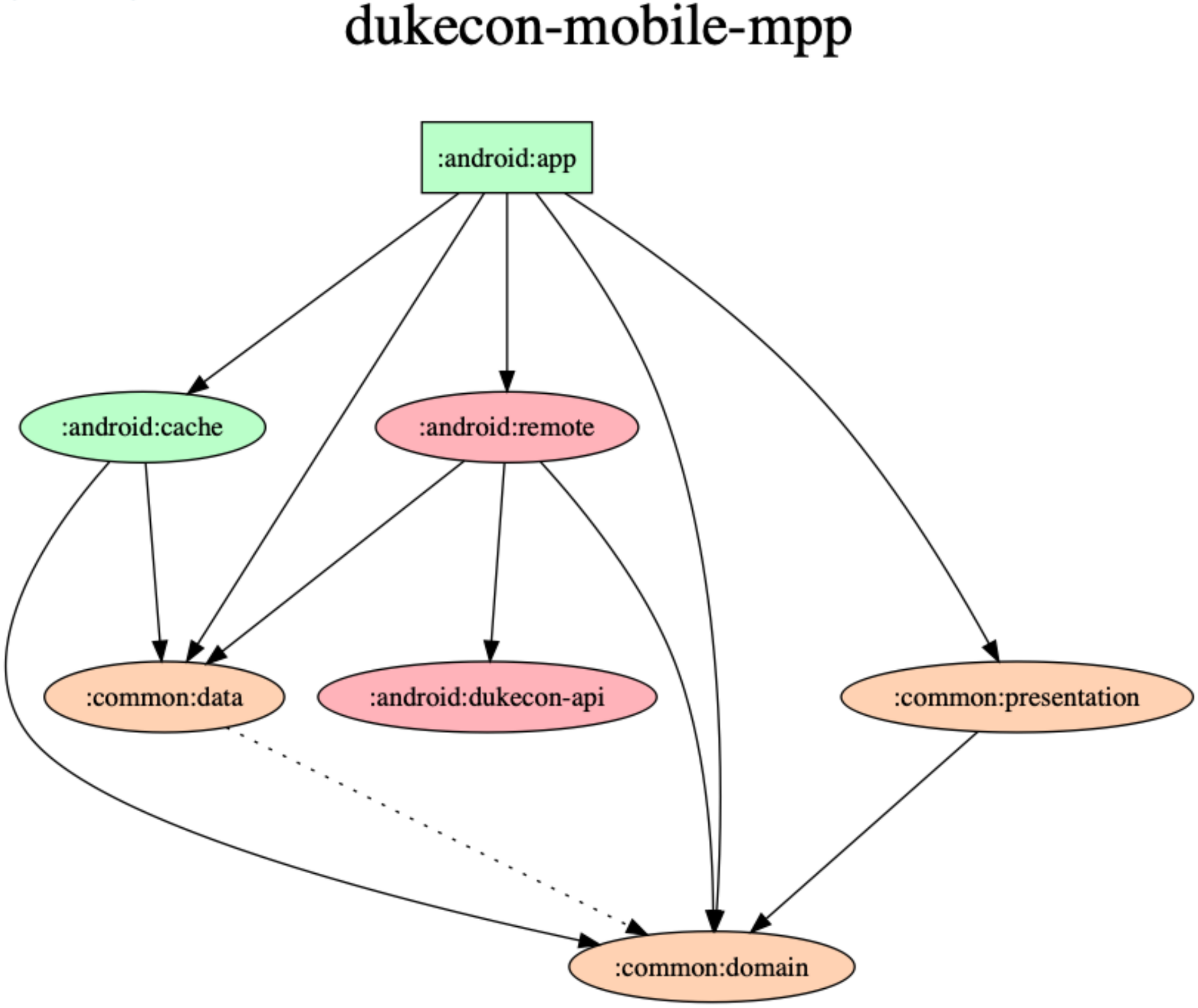
- how to debug?
- iOS specifics wisdom required
- *AppCode*





WEEKEND 2 – COMMON

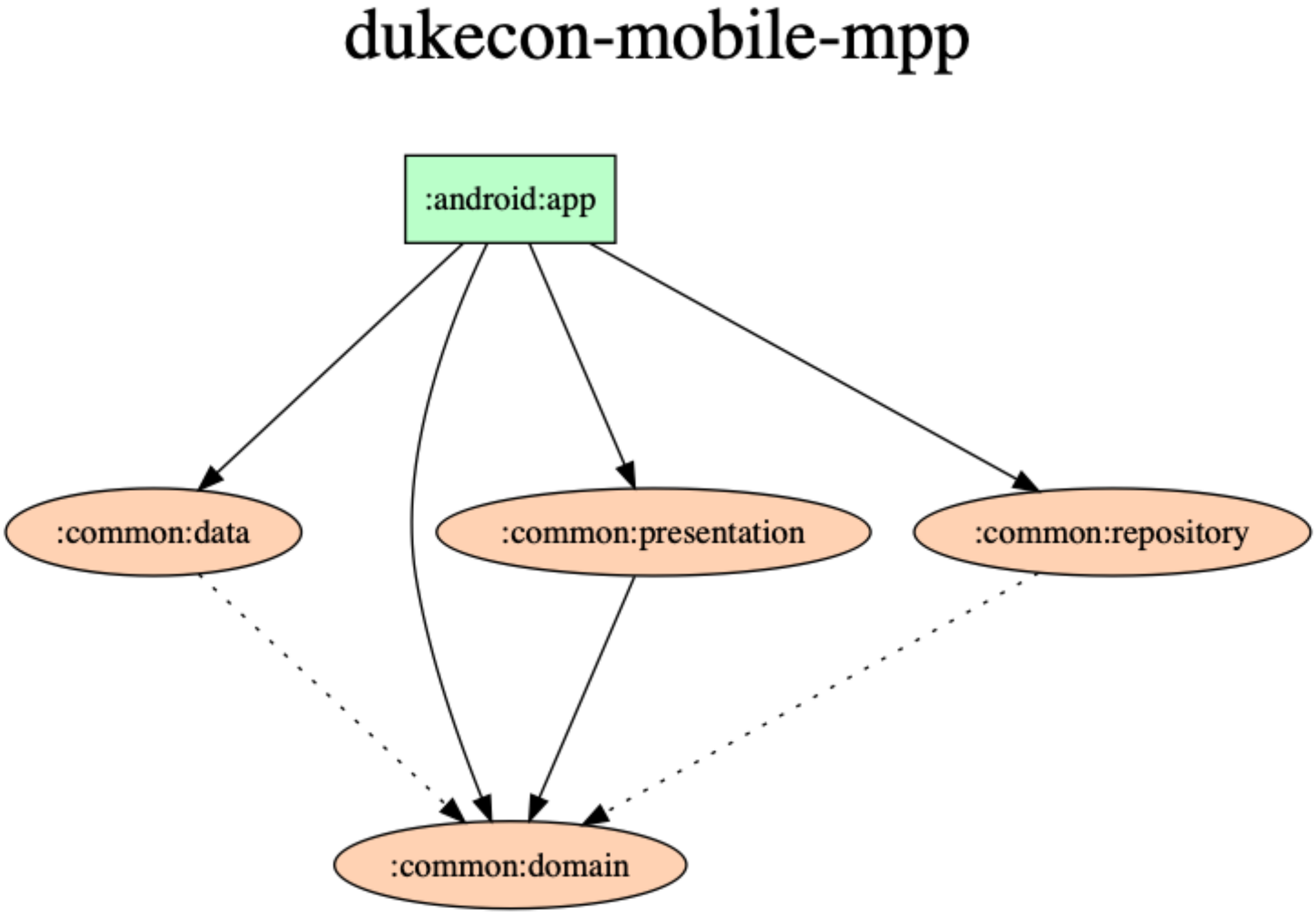
- moving java only, platform independent classes into Kotlin MPP
- removing code related to a *network awareness*
- removing **@Inject** annotations



WEEKEND 3

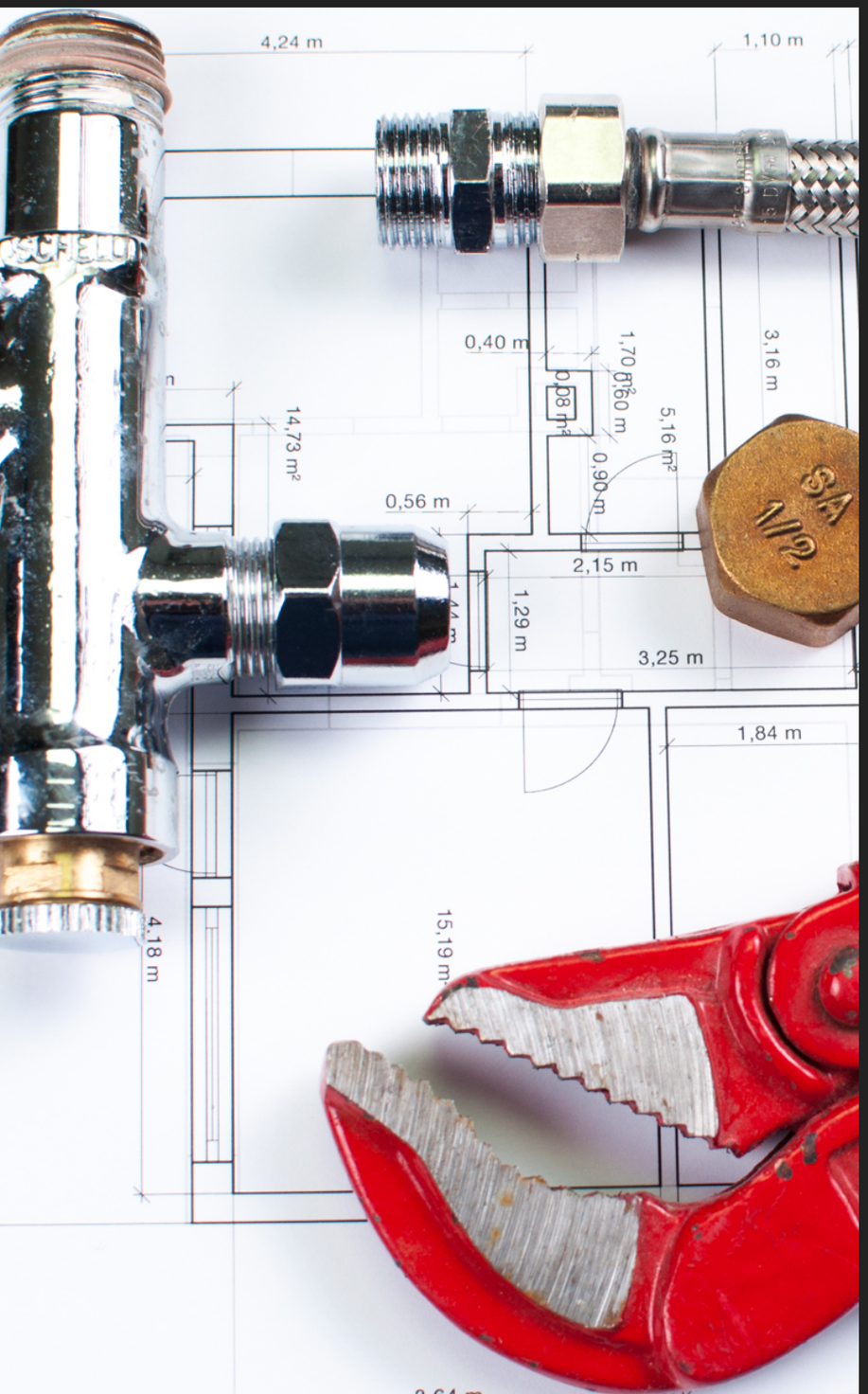
- Android App runs with Kotlin MPP Module
- still issues with Date Time
- original heavy usage of constructor injected parameters has to be replaced by manual calls in Dagger module classes

WEEKEND 3


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

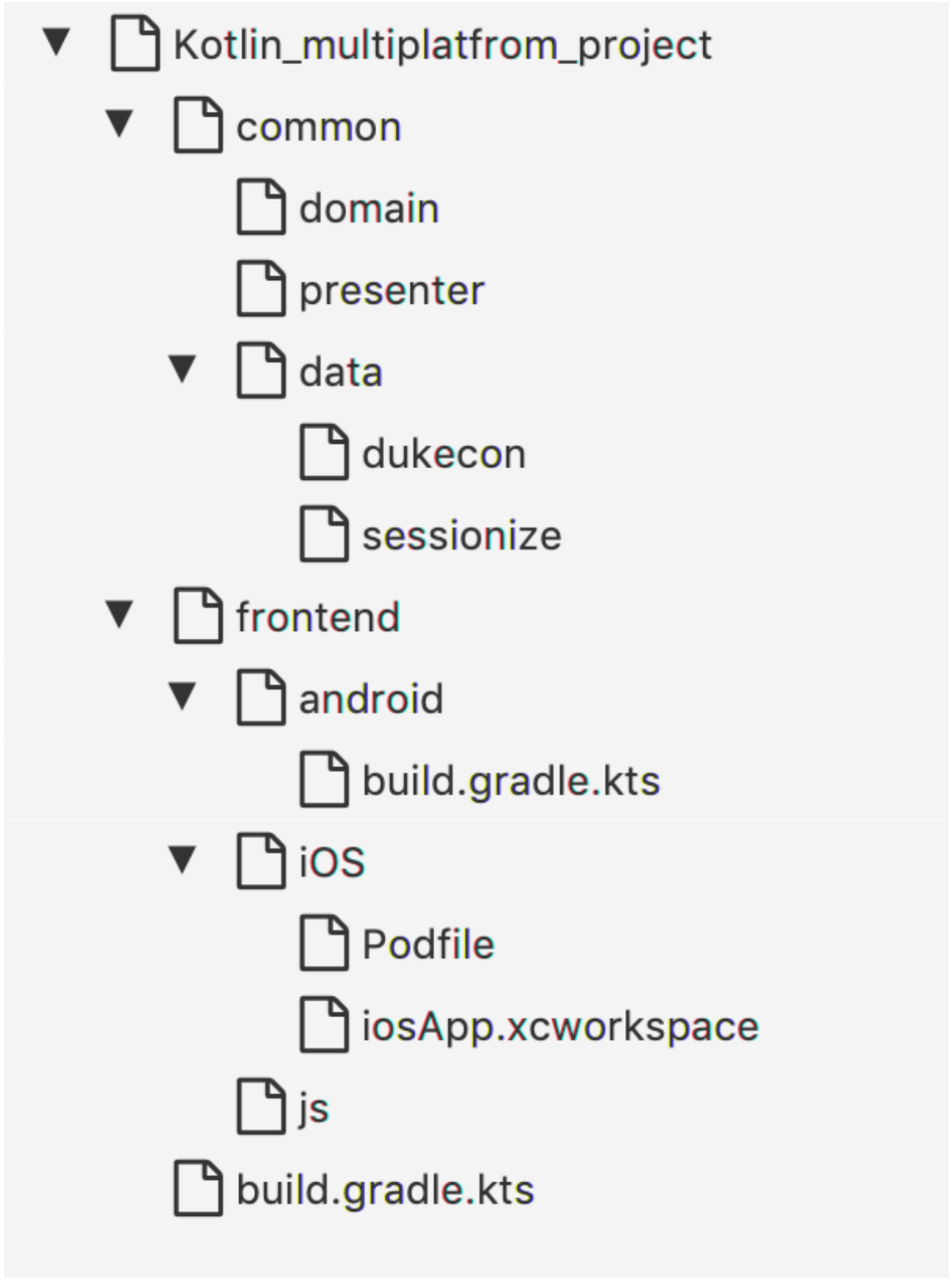
TASK TO SOLVE

PLUMBING



- Gradle modules
- Dependency management

GRADLE MODULES



GRADLE WITH KOTLIN SCRIPT

- build.gradle.kts
- iOS Framework stitching
- Spring Dependecy Management
- support for **BOM**
  - **kotlin.coroutines**
  - **ktor**

NETWORKING OR KTOR FUN

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

KTOR

ktorio / ktor

Watch 156 | Star 5,601 | Fork 415

Code | Issues 344 | Pull requests 35 | Security | Insights

Releases | Tags

Latest release

1.2.4

08fcd12

1.2.4

cy6erGn0m released this 2 days ago · 56 commits to master since this release

> Published 2 Sep 2019

- Fixed multipart form header entity separator
- Fixed crypto in IE11 (#1283)
- Marked response transient in the client exception (#1256)
- Fixed network on main thread in okhttp engine close
- Fixed follow redirect iOS (#1000)
- Kotlin 1.3.50
- kotlinx.coroutines 1.3.0

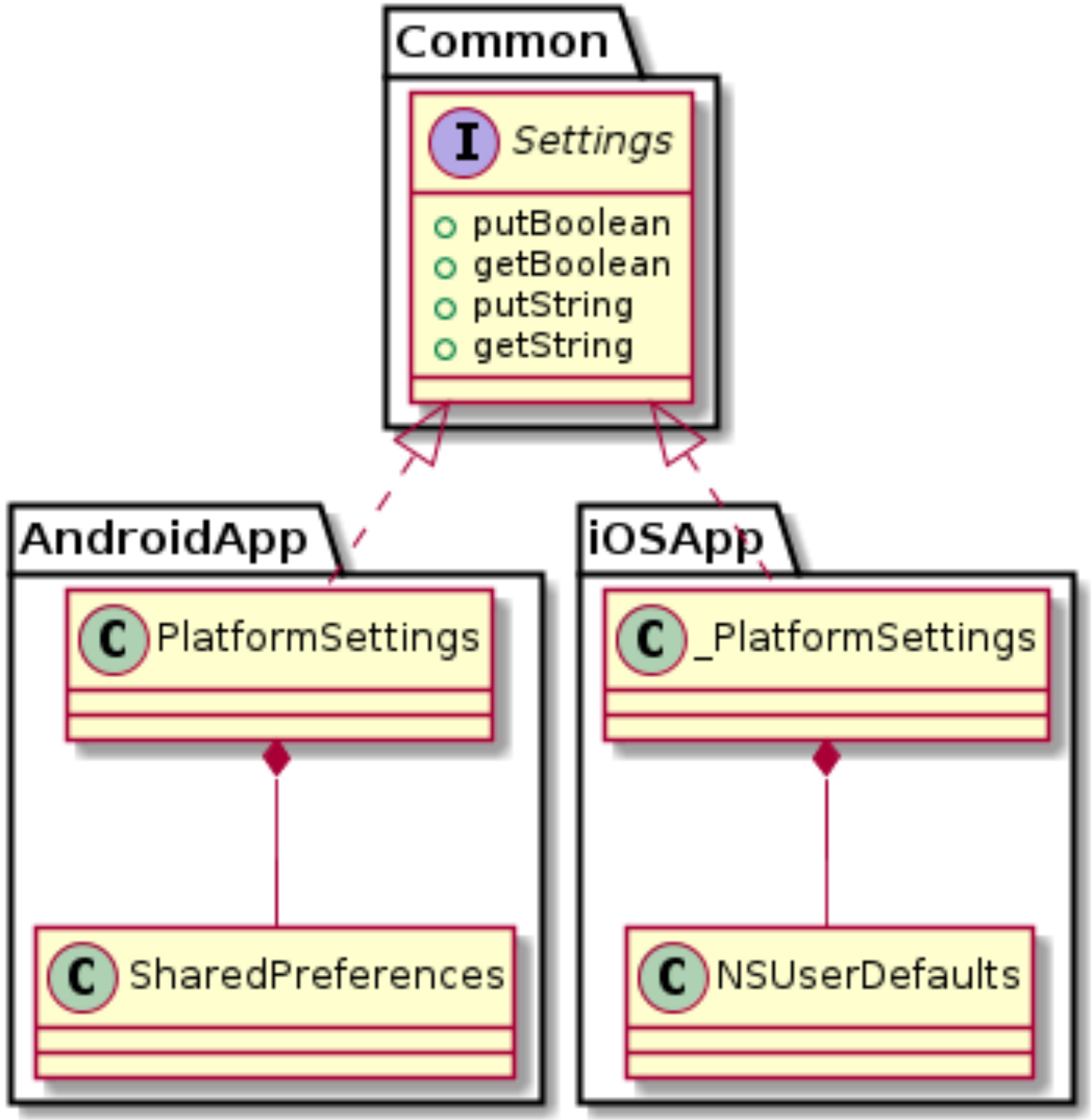
DATE AND TIME

- *MockedTimeProvider* for testing
- **JodaTime** originally
- Migrated to **ThreeTen** later
- using GMTDate from **ktor-utils**

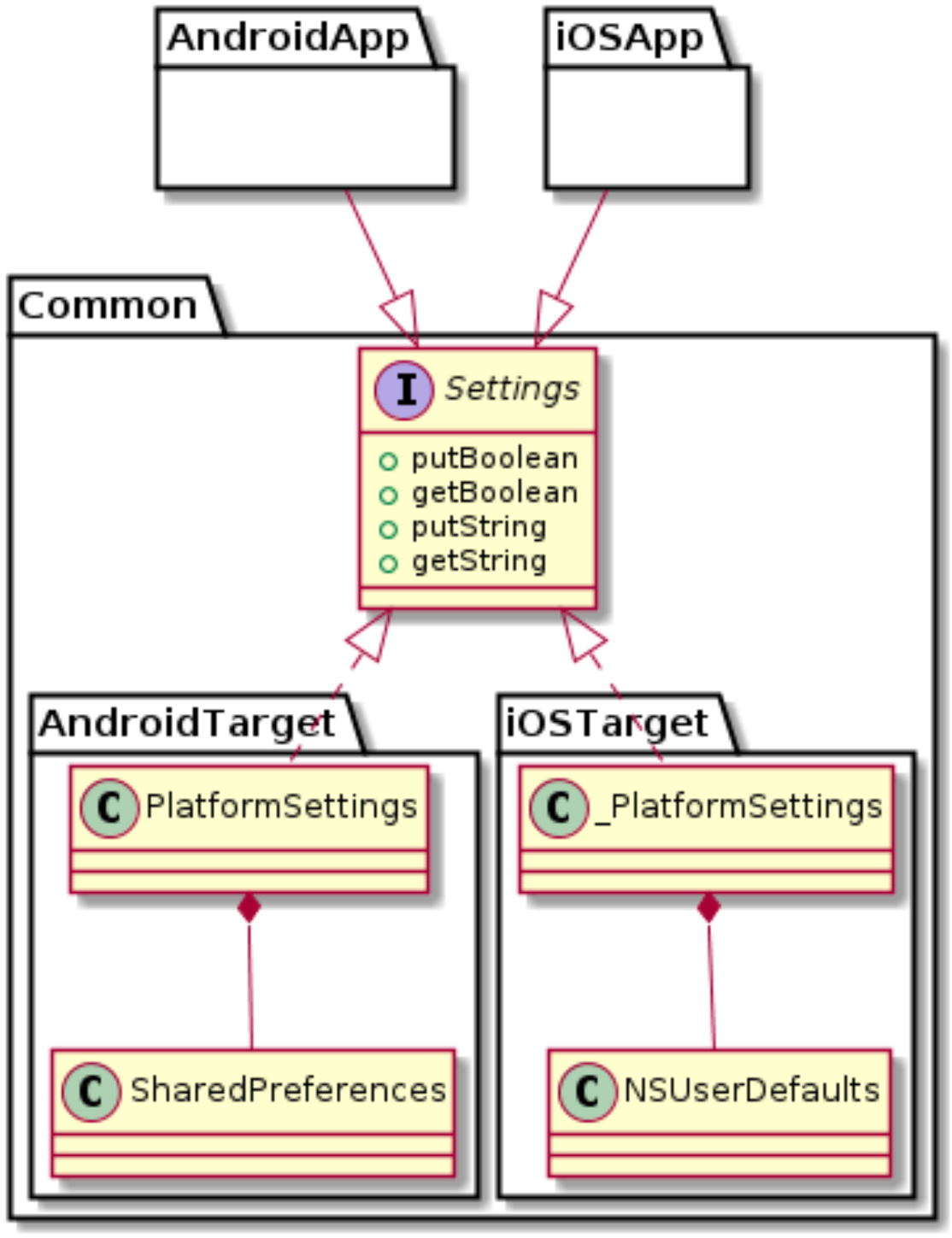
LOCAL CACHE

- **kotlinx.serialization**
- shared preferences on Android
- NSUserDefaults

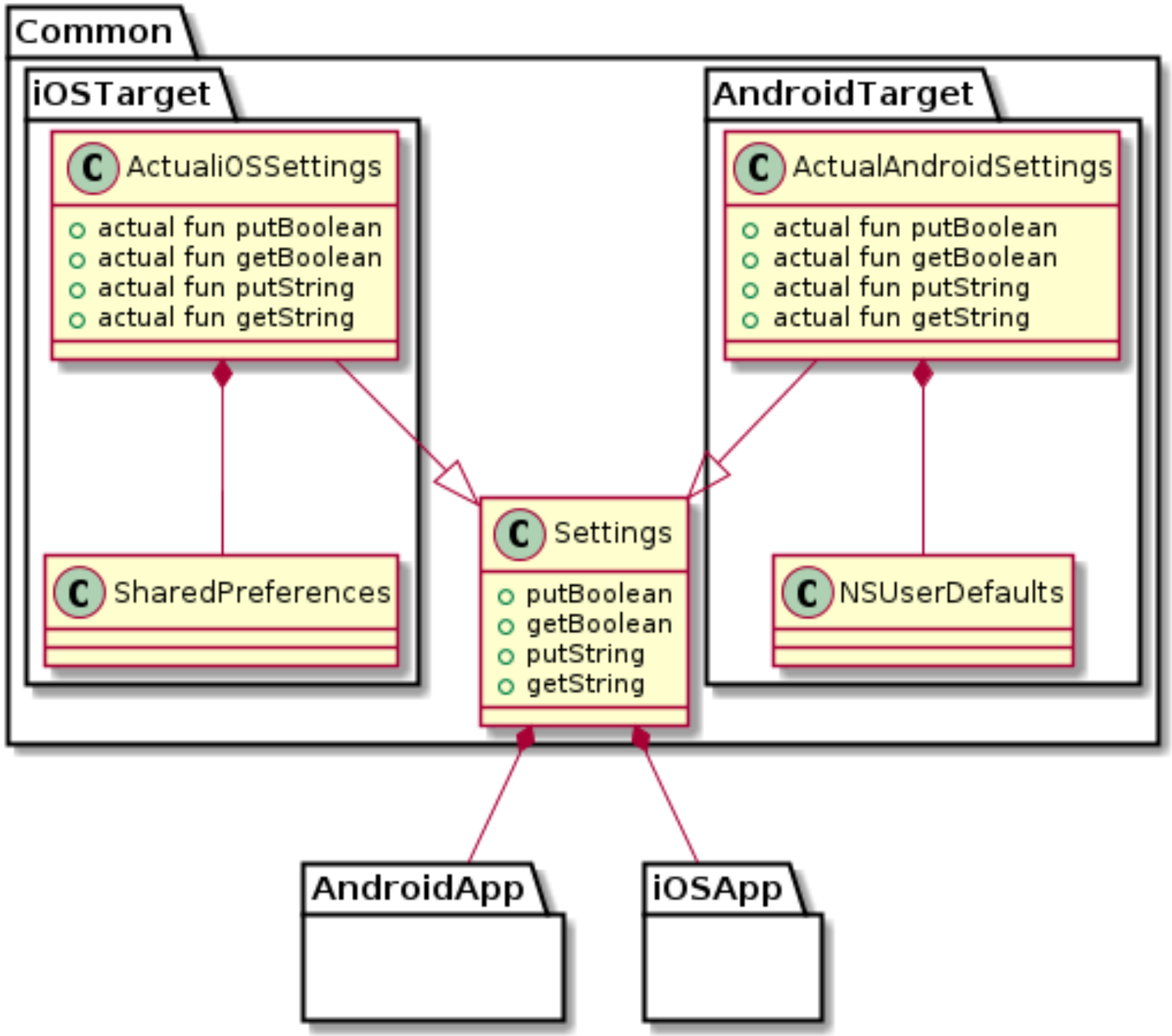
INTERFACES


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

INTERFACES MULTIPLATFORM



EXPECTED – ACTUAL



KOTLIN NATIVE -ACTUAL

```kotlin
package org.dukecon.domain.aspects.log

import org.slf4j.LoggerFactory

actual fun log(level: LogLevel, tag: String, message: String, error: Throwable) {
    val logger = LoggerFactory.getLogger(tag)
    when (level) {
        is LogLevel.DEBUG -> logger.debug(message, error)
        is LogLevel.INFO -> logger.info(message, error)
        is LogLevel.WARN -> logger.warn(message, error)
        is LogLevel.ERROR -> logger.error(message, error)
    }
}

actual fun log(level: LogLevel, tag: String, message: String) {
    val logger = LoggerFactory.getLogger(tag)
    when (level) {
        is LogLevel.DEBUG -> logger.debug(message)
        is LogLevel.INFO -> logger.info(message)
        is LogLevel.WARN -> logger.warn(message)
        is LogLevel.ERROR -> logger.error(message)
    }
}
```

KOTLIN NATIVE-EXPECTED

```kotlin
package org.dukecon.domain.aspects.log

sealed class LogLevel {
    object DEBUG : LogLevel()
    object INFO : LogLevel()
    object WARN : LogLevel()
    object ERROR : LogLevel()
}

expect fun log(level: LogLevel, tag: String, message: String)

expect fun log(level: LogLevel, tag: String, message: String, error: Throwable)
```

IOS

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

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"
}
```

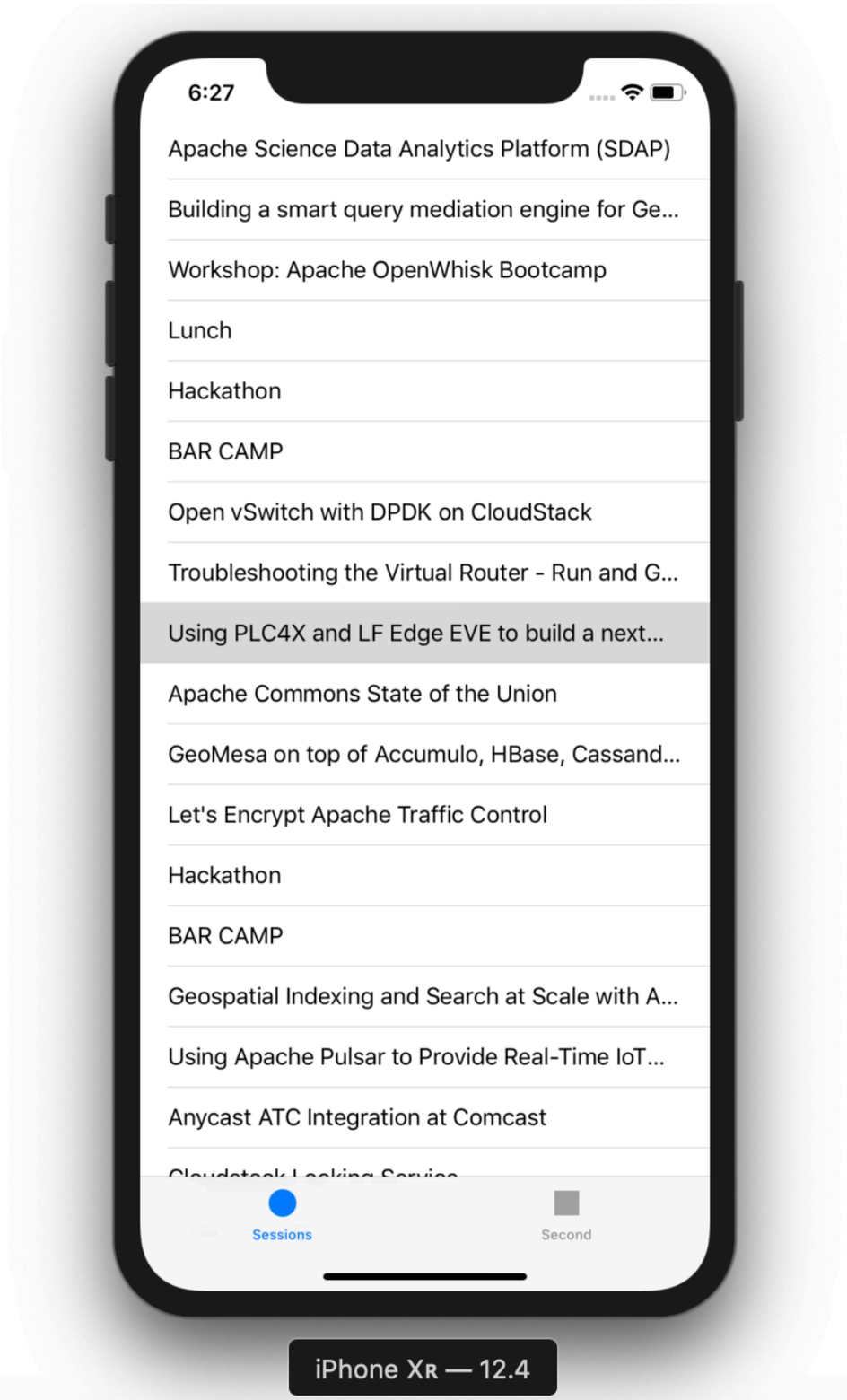
COCOAPOD FILE

```
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
  }
]
```

MODEL VIEW PRESENTER IN SWIFT

```
import UIKit
import dukeconcombined

class FirstViewController: UIViewController, UITableViewDelegate, UITableViewDataSource,
                           EventListContractView {

    lazy var presenter = {
        EventListPresenterProvider().get()
    }()

    private var events: [EventView] = []
    private var sessionsTable: UITableView!

    override func viewDidLoad() {
        super.viewDidLoad()

        sessionsTable = UITableView(frame: CGRect(x: 0, y: barHeight, width: displayWidth, height:
displayHeight - barHeight))
        sessionsTable.register(UITableViewCell.self, forCellReuseIdentifier: "SessionCell")
        sessionsTable.dataSource = self
        sessionsTable.delegate = self
        self.view.addSubview(sessionsTable)

        presenter.onAttach(view: self)
        presenter.setDate(conferenceDay: 9, showFavoritesOnly: false)

    }

    func showSessions(sessions: [EventView]) {
        events = sessions
        sessionsTable.reloadData()
    }
```

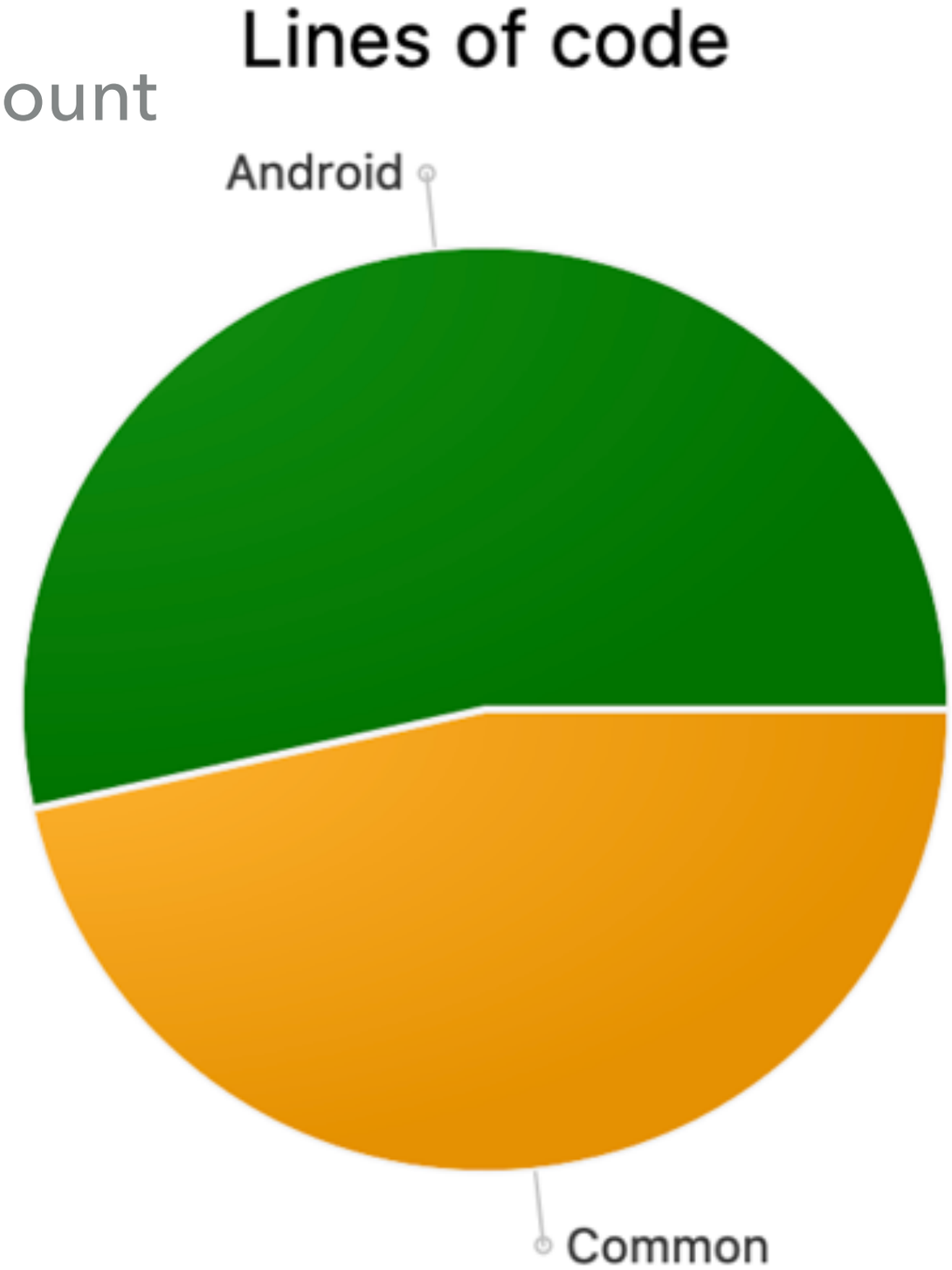
PROTOTYP, JUST STARTED



RESULT

NUMBERS

- no multidex (<65535) methods count
- 4.9MB debug
  - no shrinking
    - no optimisation
- lines of code
  - Android 2362
  - Common 2057 (**48%**)



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

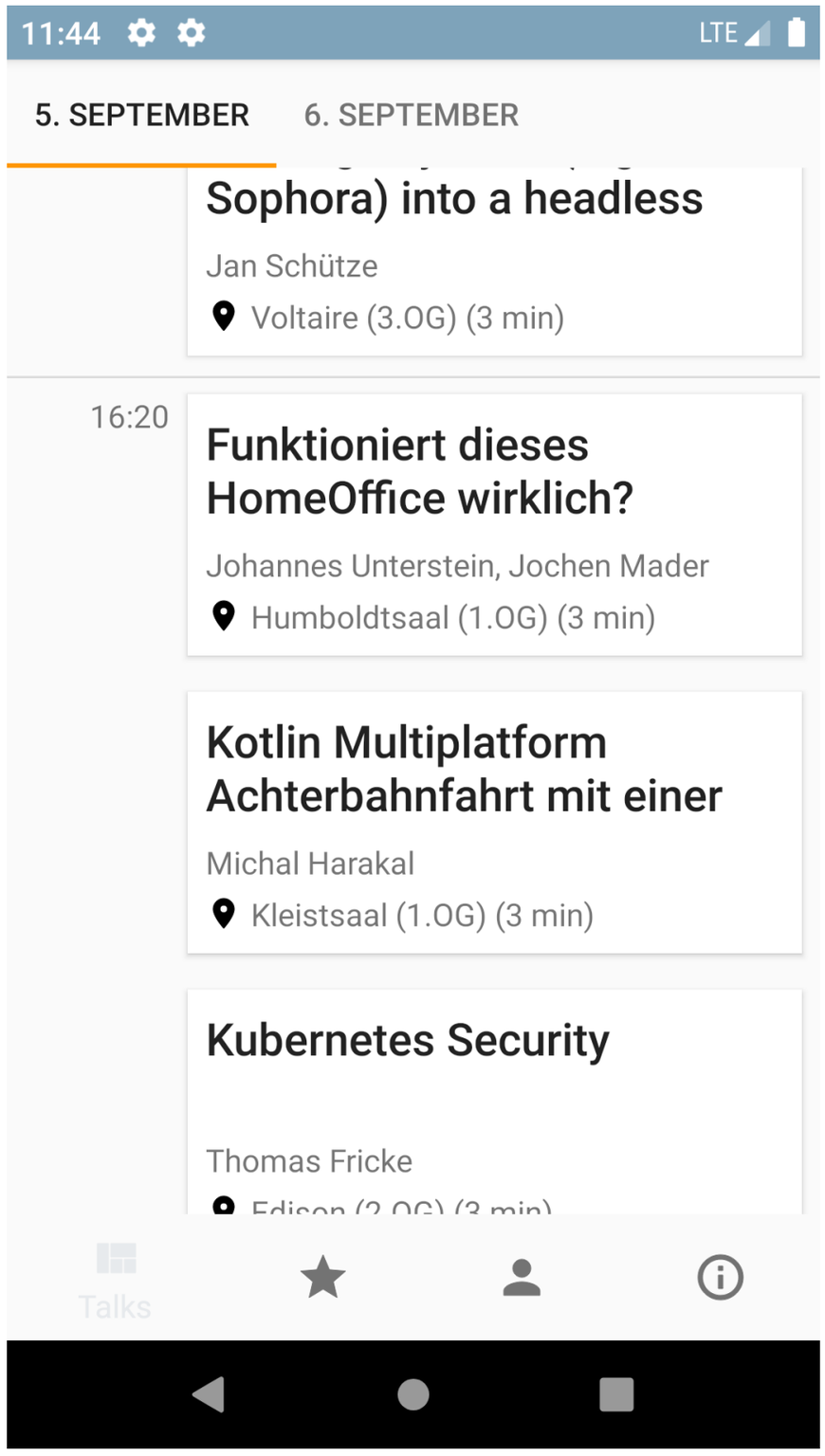
- Using Kotlin Multiplatform and proper architecture, you can write the important with once and share it in apps running on multiple platforms.
- Proper architecture can increase a ratio of shared code

SMALL HINTS

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

SURPRISE

BED-CON 2019


11:44
LTE
5. SEPTEMBER
6. SEPTEMBER
Sophora) into a headless
Jan Schütze
Voltaire (3.OG) (3 min)
16:20
Funktioniert dieses HomeOffice wirklich?
Johannes Unterstein, Jochen Mader
Humboldtsaal (1.OG) (3 min)
Kotlin Multiplatform Achterbahnfahrt mit einer
Michal Harakal
Kleistsaal (1.OG) (3 min)
Kubernetes Security
Thomas Fricke
Talks

THANK YOU

LINKS

- Android Robot image https://developer.android.com/distribute/marketing-tools/brand-guidelines
- Kotlin logo https://resources.jetbrains.com/storage/products/kotlin/docs/kotlin_logos.zip
- https://kotlinlang.org/docs/tutorials/native/mpp-ios-android.html
- https://github.com/touchlab/DroidconKotlin
- https://github.com/JetBrains/kotlinconf-app
- https://dukecon.org