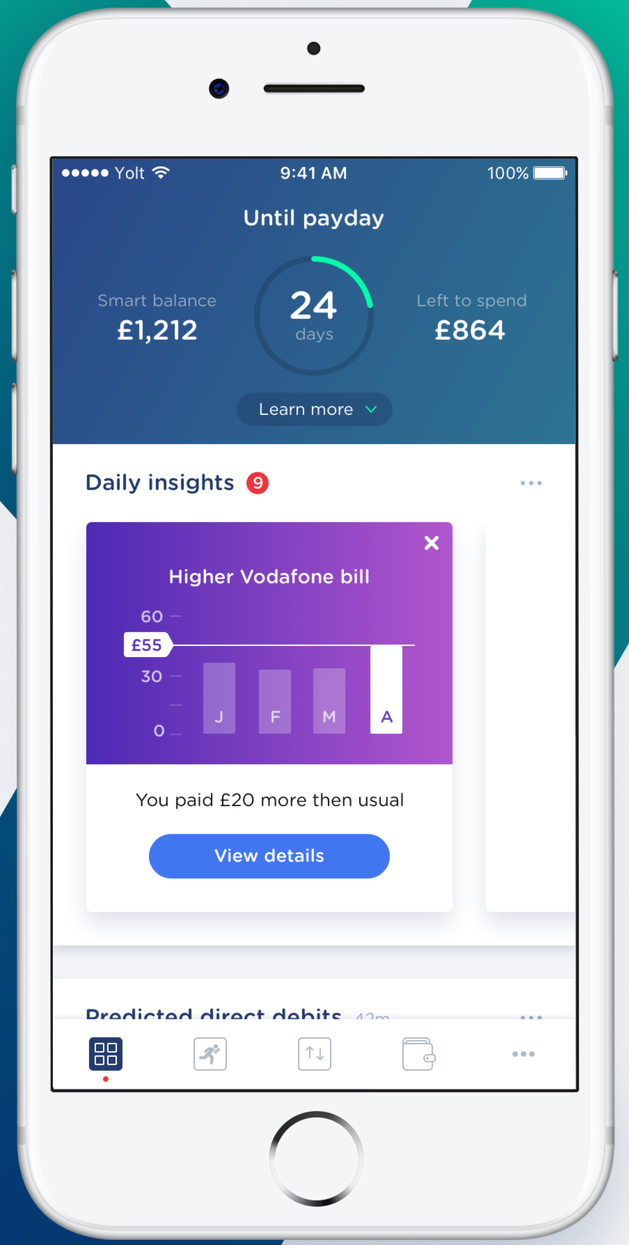
YOLT

Gradle (Incremental) Things

Again



Agenda

- What and why incremental builds
- Modularisation and “dependencies, dependencies, dependencies”
- Dagger Reflect
- ~~Moshi Reflect~~ CatchUp
- Q&A

Incremental builds

Incremental builds

- This is what happens majority of time on your dev machine
- Smaller times are better productivity
- Ideally should be matter of seconds with the small change

Modularisation and dependencies

Modules structure and dependencies

- Build speed can be very dependent on your modules graph
- Unused dependencies are making your debug app size bigger
- Unused dependencies are making your builds potentially slower
- It is nice to have everything clean and in control

Dependencies analysis plugin

https://github.com/autonomousapps/dependency-analysis-android-gradle-plugin

- Unused dependencies which can be removed
- Wrong configurations declared dependencies (api vs implementation)
- Transitive dependencies that should be declared directly
- Experimental: detect **compileOnly** dependencies

Dependencies analysis plugin

https://github.com/autonomousapps/dependency-analysis-android-gradle-plugin

The general philosophy of the plugin:

1. You should directly declare every dependency you use. Don’t rely on it being available transitively

2. You should correctly declare every dependency as implementation or api, depending on wether it is an implementation detail or part of ABI (binary API)

ABI == binary API

```
public class MyClass : MyOtherClass {}
```

MyClass is defined in module A, and **MyOtherClass** is defined in module B. Module A should declare **api(project(":B")),** because **MyOtherClass** is part of A's ABI

Dependencies analysis plugin

Limitations

- Gives false positive if module depends on the resource
- Gives false positive if module is not used in sources directly (LeakCanary)
- Doesn’t work correctly with dynamic features (from my experience)

Tivi app

Tivi 📺 (work-in-progress 👷🔧👷⛏)

This is not an official Google product

Tivi is a **work-in-progress** TV show tracking Android app, which connects to Trakt.tv. It is still in its early stages of development and currently only contains two pieces of UI. It is under heavy development.

Android development

Tivi is an app which attempts to use the latest cutting edge libraries and tools. As a summary:

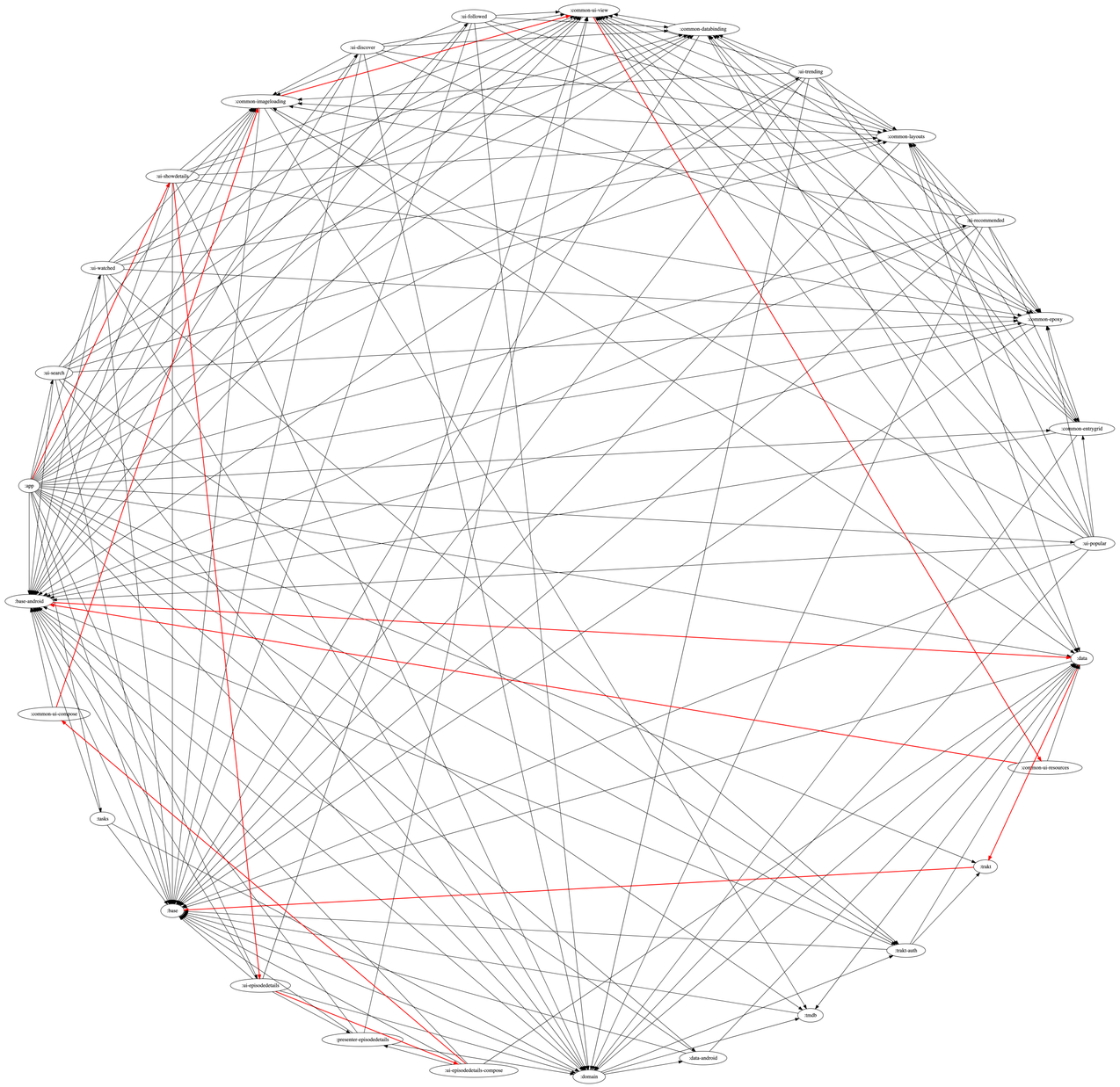
- Entirely written in Kotlin
- Uses RxJava 2
- Uses all of the Architecture Components: Room, LiveData and Lifecycle-components
- Uses dagger-android for dependency injection

https://github.com/chrisbanes/tivi

YOLT

Demo #1

Tivi app modules structure



Modules graph assert plugin

https://github.com/jraska/modules-graph-assert

- Assert depth of the modules graph
- Assert the correct dependencies order
- Generate your graph picture

Modules graph assert plugin

https://github.com/jraska/modules-graph-assert

```
moduleGraphAssert {
  maxHeight = 4
  // modules prefixed with ":feature:" -> prefix ":lib:" -> prefix :core:
  moduleLayers = [":feature:\\S*", ":lib\\S*", ":core\\S*"]
  moduleLayersExclude = [":feature-about -> :feature-legacy-about”]
  //regex to match module names
  restricted = [':feature-[a-z]* -X> :forbidden-to-depend-on']
}
```

YOLT

Demo #2

Can I drop jetifier?

https://github.com/plnice/can-i-drop-jetifier

- Checks whether there are any dependencies using support library instead of AndroidX artifacts

- Has own configuration

```
canIDropJetifier {
  verbose = true // Default: false, set to true to print the dependencies tree down to the old artifact
  includeModules = false // Default: true, print out not only external (library) dependencies, but also module dependencies that use old artifacts
  analyzeOnlyAndroidModules = false // Default: true, analyze only modules that use com.android.application or com.android.library plugins
  configurationRegex = ".*RuntimeClasspath" // Performance optimization: checks only configurations that match provided regex
  parallelMode = true // Default: false, experimental: run analysis of modules in parallel
  parallelModePoolSize = 4 // Default: max available processors - 1, experimental: pool size for analysis in parallel
}
```

YOLT

Demo #3

Reflection instead of code generation

Dagger Reflection

Dagger Reflect

https://github.com/JakeWharton/dagger-reflect

- Reflection based implementation of Dagger dependency library
- Two ways of the integration
- Doesn’t work with abstract classes – only interfaces are working
- Component visibility should be always public
- Doesn’t work with Producers

Dagger Reflect

Two ways of the integration

- Partial
  - Doesn’t require code changes to switch between reflection and code generation
  - Still have small code generation part
- Full
  - Requires code changes to have bridge
  - No code generation

Delect

https://github.com/soundcloud/delect

- Uses partial Dagger reflect integration
- Automatically changes dependencies for code generation to reflection version based on flag
- Make sure you use @Component.Builder and @Component.Factory for components creation
- Set retention police to RUNTIME for all @Qualifier and @MapKey annotations

YOLT

Demo #4

Numbers

How does it improve the build?

Path	Started after	Duration	Class
:core:kaptDebugKotlin	23.766s	11.193s	org.jetbrains.kotlin.gradle.internal.KaptWithoutKotlincTask
:dribbble:kaptDebugKotlin	1m 13.744s	9.108s	org.jetbrains.kotlin.gradle.internal.KaptWithoutKotlincTask
:core:kaptGenerateStubsDebugKotlin	14.742s	9.022s	org.jetbrains.kotlin.gradle.internal.KaptGenerateStubsTask
:designernews:kaptDebugKotlin	1m 15.643s	8.945s	org.jetbrains.kotlin.gradle.internal.KaptWithoutKotlincTask
:designernews:kaptGenerateStubsDebugKotlin	1m 8.720s	6.920s	org.jetbrains.kotlin.gradle.internal.KaptGenerateStubsTask
:about:kaptDebugKotlin	1m 13.810s	6.905s	org.jetbrains.kotlin.gradle.internal.KaptWithoutKotlincTask
:search:kaptDebugKotlin	1m 12.775s	5.309s	org.jetbrains.kotlin.gradle.internal.KaptWithoutKotlincTask
:about:kaptGenerateStubsDebugKotlin	1m 8.718s	5.089s	org.jetbrains.kotlin.gradle.internal.KaptGenerateStubsTask
:dribbble:kaptGenerateStubsDebugKotlin	1m 8.723s	5.019s	org.jetbrains.kotlin.gradle.internal.KaptGenerateStubsTask
:search:kaptGenerateStubsDebugKotlin	1m 8.726s	4.046s	org.jetbrains.kotlin.gradle.internal.KaptGenerateStubsTask

Found 12 tasks executed in 6 projects totaling 1m 17.685s

348 tasks executed in 8 projects in 1m 31s.

Path	Started after	Duration	Class
:core:kaptGenerateStubsDebugKotlin	9.038s	7.109s	org.jetbrains.kotlin.gradle.internal.KaptGenerateStubsTask
:core:kaptDebugKotlin	16.149s	9.067s	org.jetbrains.kotlin.gradle.internal.KaptWithoutKotlincTask
:app:kaptGenerateStubsDebugKotlin	45.708s	2.192s	org.jetbrains.kotlin.gradle.internal.KaptGenerateStubsTask
:app:kaptDebugKotlin	47.902s	1.915s	org.jetbrains.kotlin.gradle.internal.KaptWithoutKotlincTask
:about:kaptGenerateStubsDebugKotlin	52.372s	1.853s	org.jetbrains.kotlin.gradle.internal.KaptGenerateStubsTask
:designernews:kaptGenerateStubsDebugKotlin	52.373s	2.529s	org.jetbrains.kotlin.gradle.internal.KaptGenerateStubsTask
:dribbble:kaptGenerateStubsDebugKotlin	52.373s	1.876s	org.jetbrains.kotlin.gradle.internal.KaptGenerateStubsTask
:search:kaptGenerateStubsDebugKotlin	52.374s	1.596s	org.jetbrains.kotlin.gradle.internal.KaptGenerateStubsTask
:search:kaptDebugKotlin	53.971s	0.861s	org.jetbrains.kotlin.gradle.internal.KaptWithoutKotlincTask
:about:kaptDebugKotlin	54.226s	3.410s	org.jetbrains.kotlin.gradle.internal.KaptWithoutKotlincTask
:dribbble:kaptDebugKotlin	54.250s	3.570s	org.jetbrains.kotlin.gradle.internal.KaptWithoutKotlincTask

Found 12 tasks executed in 6 projects totaling 39.853s

348 tasks executed in 8 projects in 1m 3s.

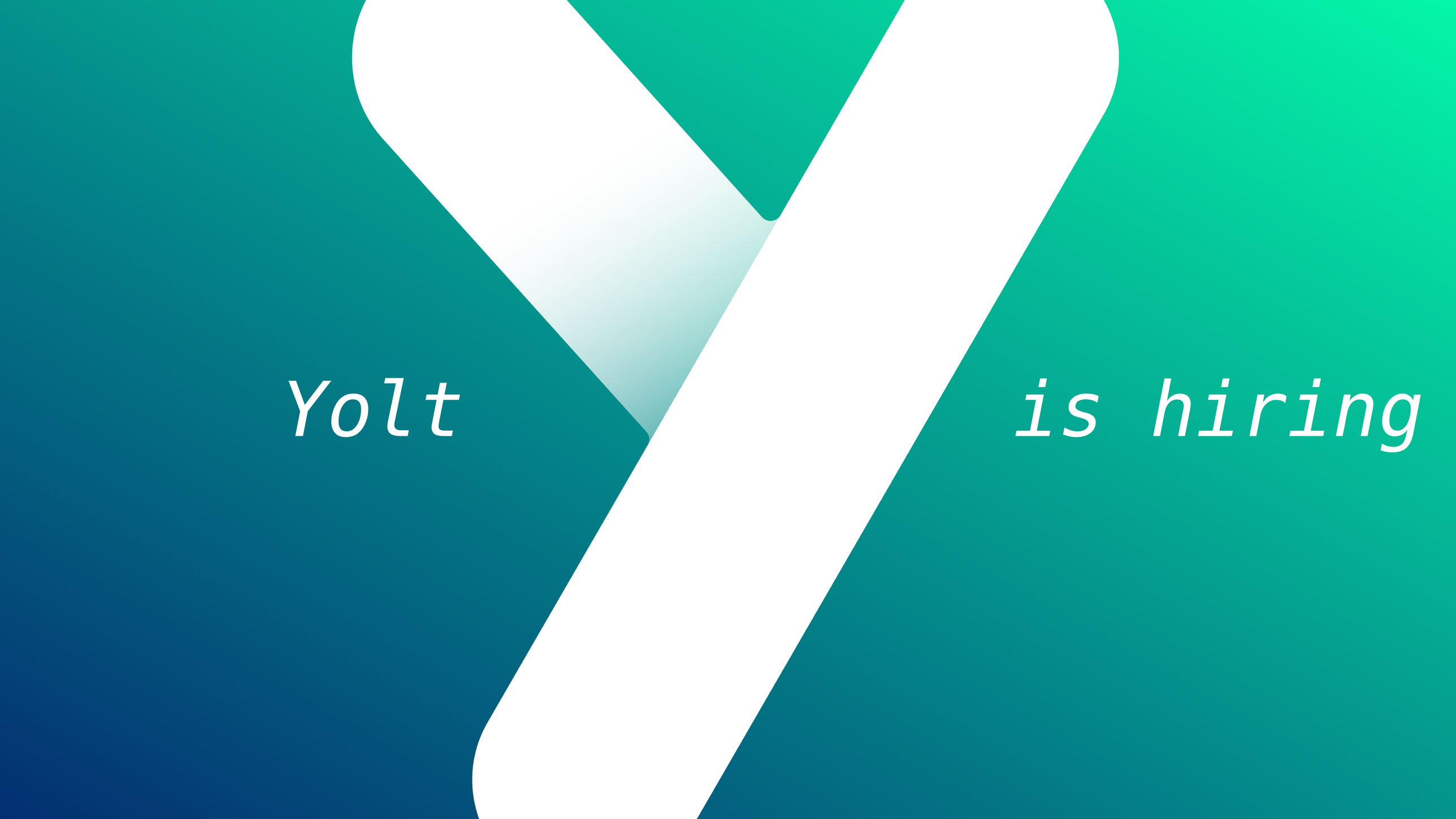
Moshi reflection

CatchUp

https://github.com/ZacSweers/CatchUp

Technologies

- Kotlin
- RxJava 2/AutoDispose
- Debugging tooling as a first class citizen in the debug build
- Leak Canary, Chuck, Scalpel, debug drawer, Flipper, bug reporting, the works
- AndroidX/Jetpack
- Dagger 2
- One of the more interesting parts of CatchUp is that its service architecture is a Dagger-powered plugin system
- Room (Arch components)
- AutoValue + extensions
- Firebase
- Coil
- Apollo GraphQL
- Standard Square buffet of Okio/OkHttp 3/Retrofit 2/Moshi
- Inspector


Yolt
is hiring

Links

- Dependencies analysis plugin (https://github.com/autonomousapps/dependency-analysis-android-gradle-plugin)
- Graph assert plugin (https://github.com/jraska/modules-graph-assert)
- “Can I drop jetifier” plugin (https://github.com/plnice/can-i-drop-jetifier)
- Dagger reflect (https://github.com/JakeWharton/dagger-reflect)
- Delect plugin (https://github.com/soundcloud/delect)
- Nelson’s article about Deject (https://developers.soundcloud.com/blog/dagger-reflect)
- CatchUp project (https://github.com/ZacSweers/CatchUp)
- Tivi project (https://github.com/chrisbanes/tivi)
- Plaid project (https://github.com/android/plaid)

Q&A
(Please)