Isolated Development

Ralf Wondratschek
Principal Engineer

Isolated Development

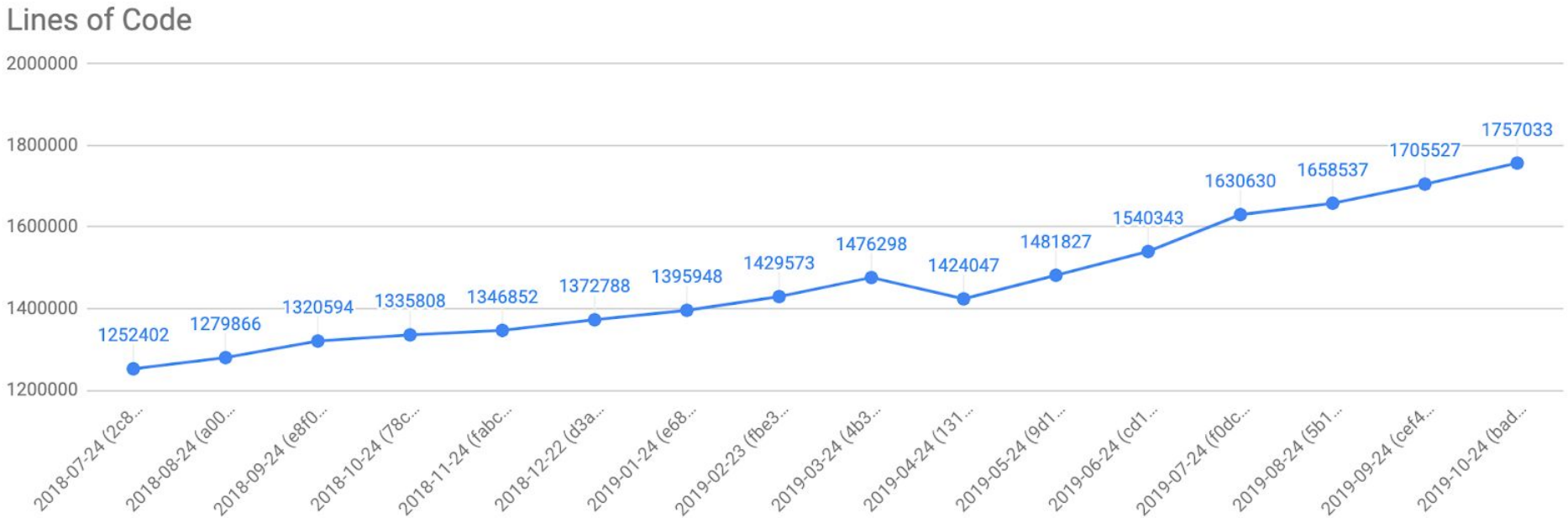
Ralf Wondratschek

@vRallev

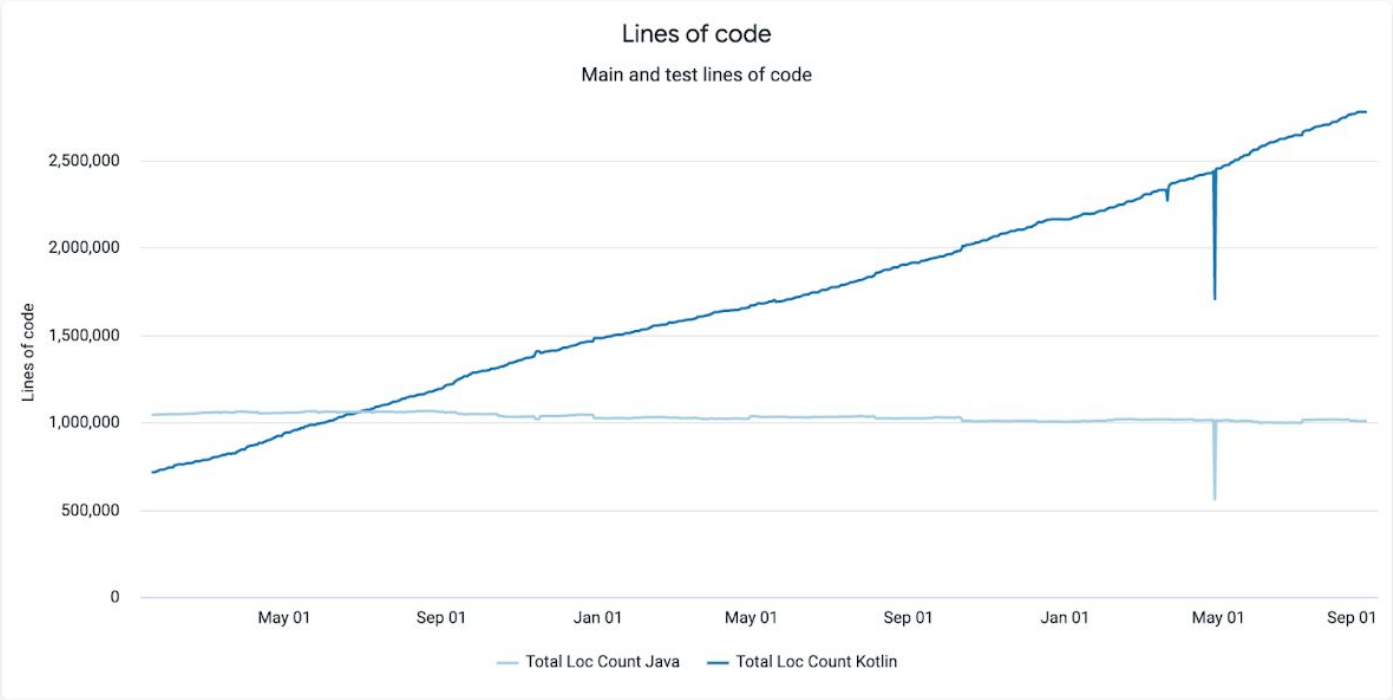
amazon

Challenges in 2018

Challenges in 2018: lines of code growth



Challenges in 2018: lines of code growth



1,007,293

LOC Java

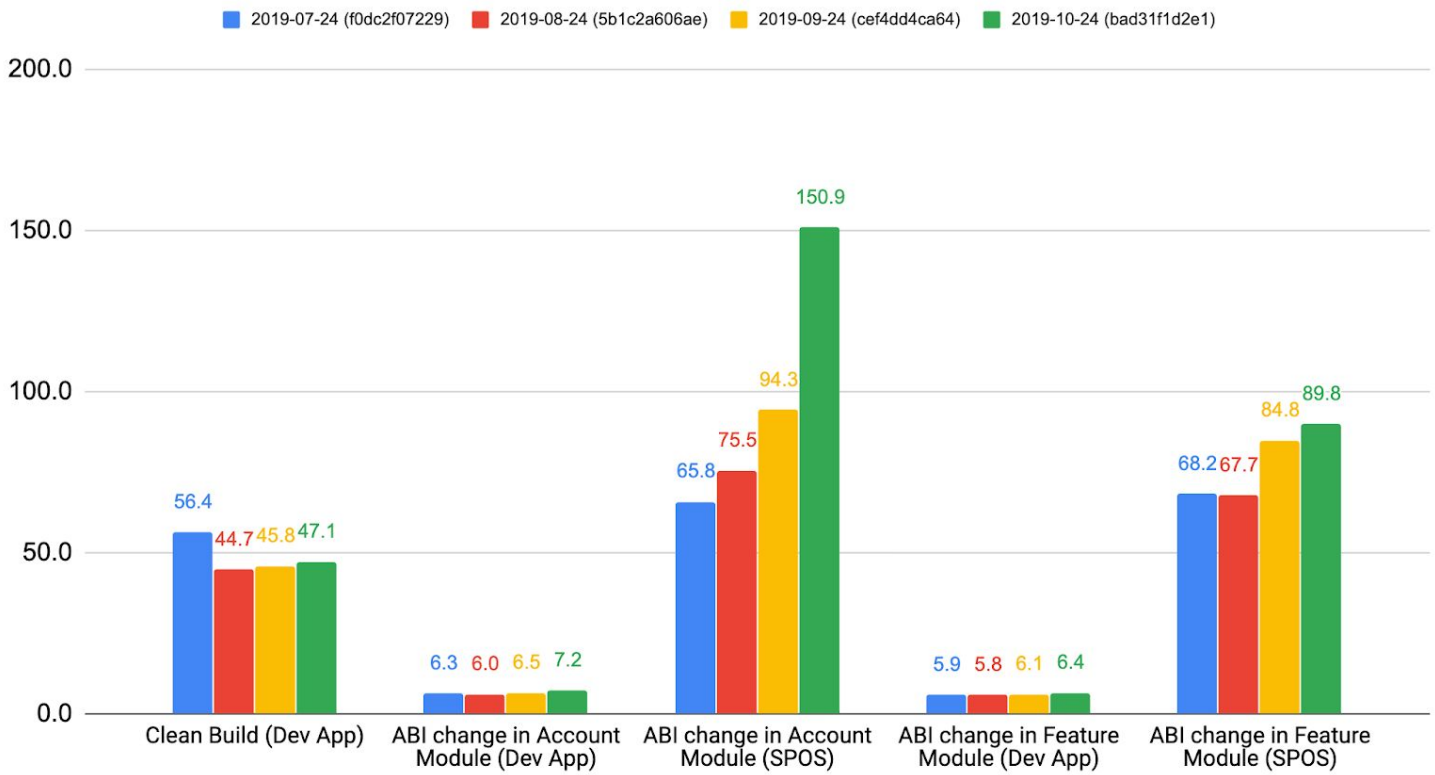
▼ -251 week over week

2,777,371

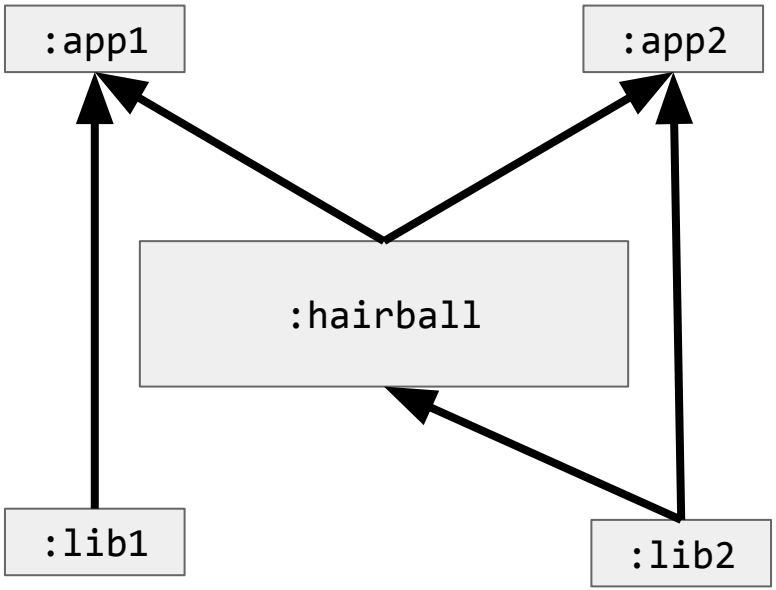
LOC Kotlin

▲ 2,604 week over week

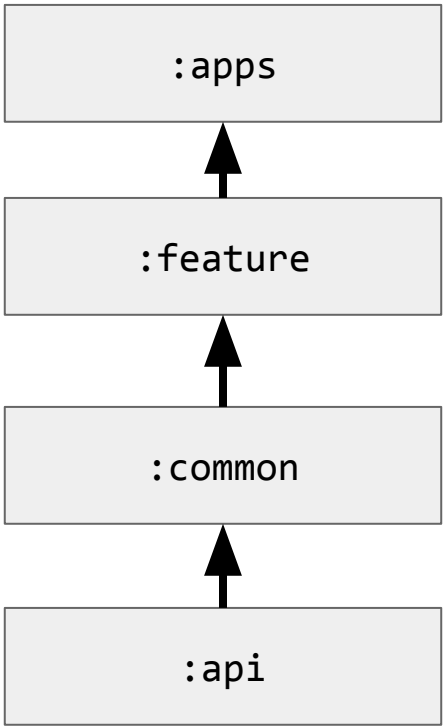
Challenges in 2018: build times are growing



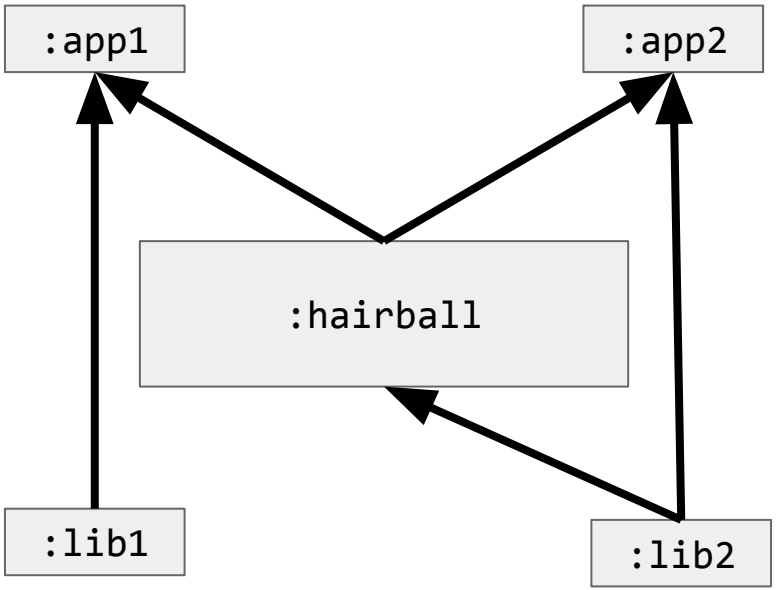
Challenges in 2018: module structure



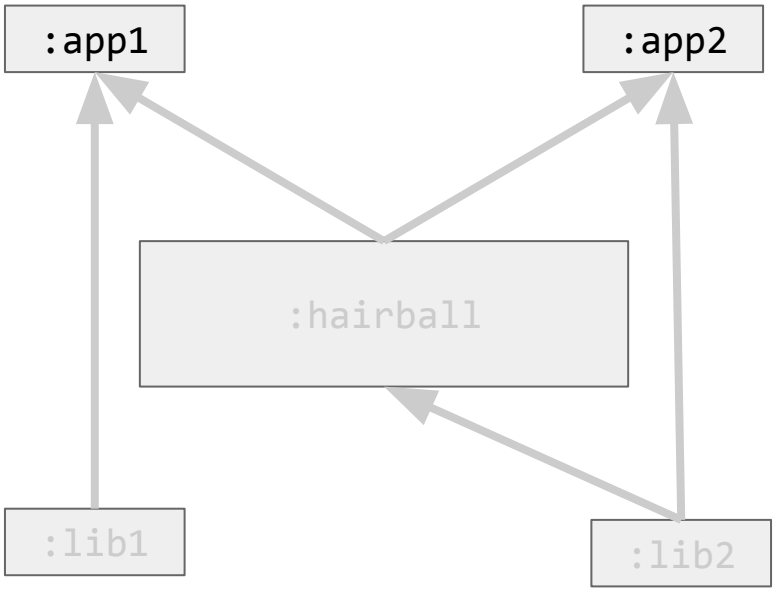
Challenges in 2018: module structure

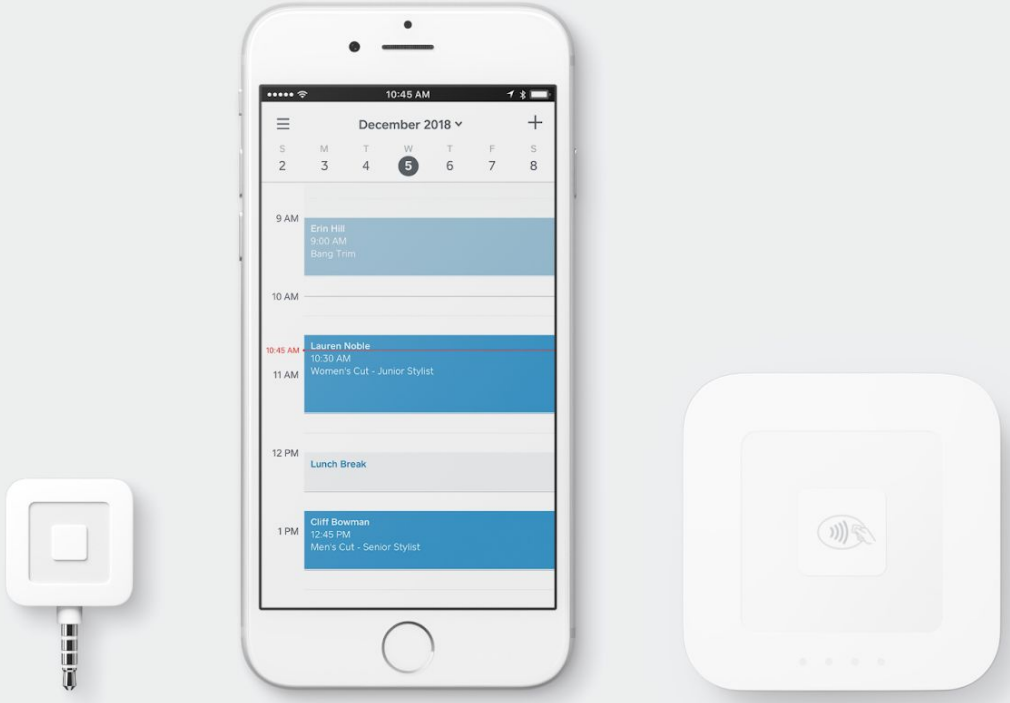


Challenges in 2018: multiple apps with different flavors

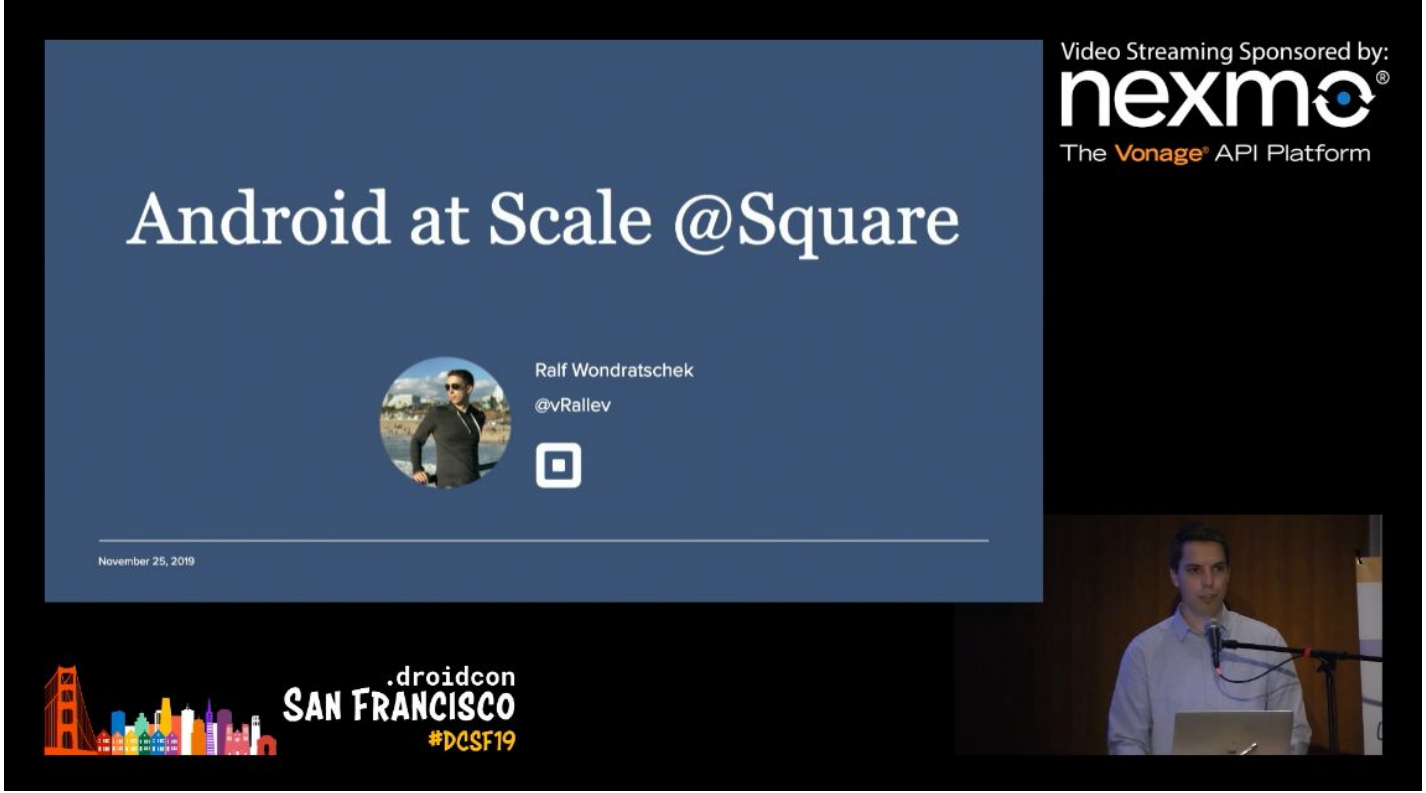


Challenges in 2018: multiple apps with different flavors




10:45 AM
December 2018
S M T W T F S
2 3 4 5 6 7 8
9 AM
Erin Hill
9:00 AM
Bang Trim
10 AM
10:45 AM
Lauren Noble
10:30 AM
Women's Cut - Junior Stylist
11 AM
12 PM
Lunch Break
1 PM
Cliff Bowman
12:45 PM
Men's Cut - Senior Stylist

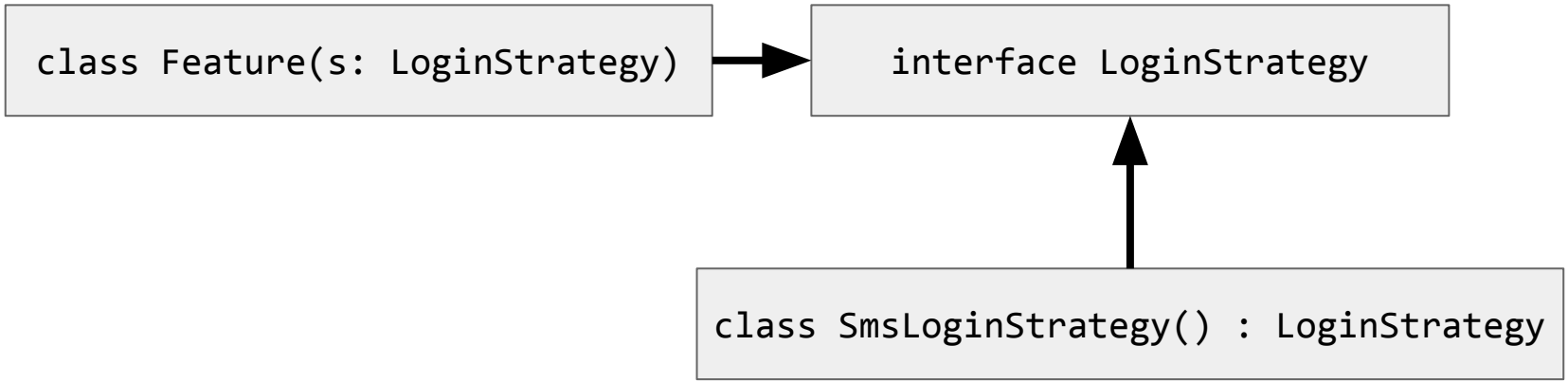

12:37 PM
Current Sale
Charge $65.00
1
Note
$0.00
1
2
3
4
5
6
7
8
9
C
0
Keypad
Library



https://bit.ly/3D8SsUl

Dependency inversion

Dependency inversion



Dependency inversion

```
class SmsLoginStrategy @Inject constructor(
 private val helper: Helper,
 private val helper: Helper1,
 private val helper: Helper2,
 private val helper: Helper3,
 private val helper: Helper4,
 private val helper: Helper5,
 private val helper: Helper6,
 private val helper: Helper7,
 private val helper: Helper8
) : LoginStrategy
```

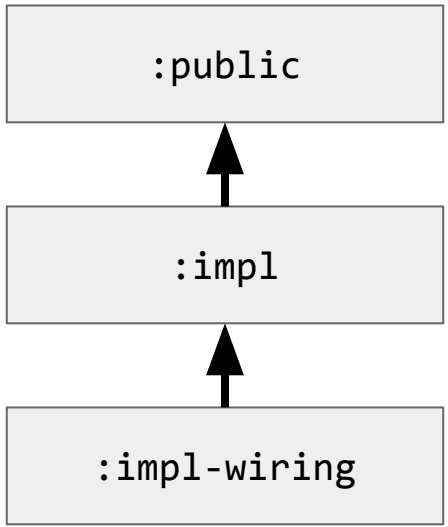
Dependency inversion

```
dependencies {
 api project(':helper1')
 api project(':helper2')
 api project(':helper3')
 api project(':helper4')
 api project(':helper5')
 api project(':helper6')
 api project(':helper7')
 api project(':helper8')
}
```

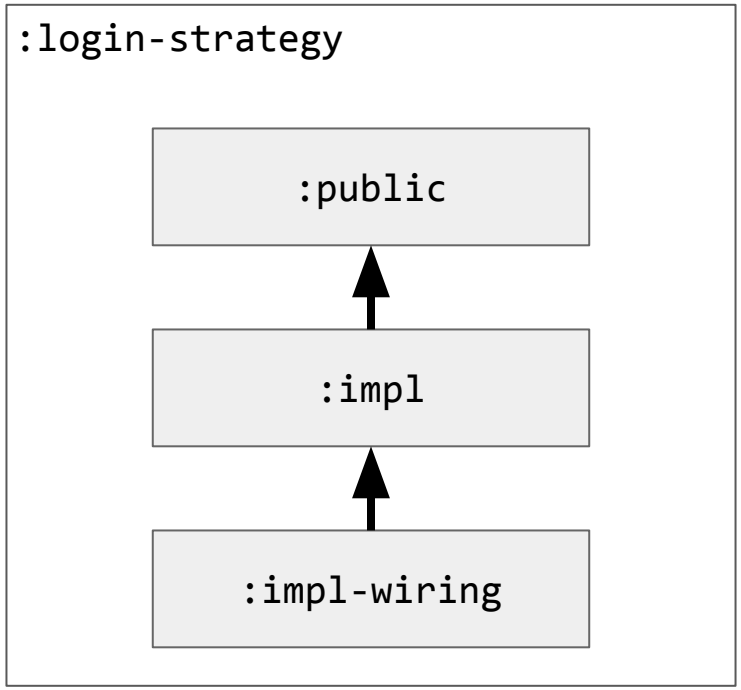
Dependency inversion

```
dependencies {
 api project(':login-strategy')
}
```

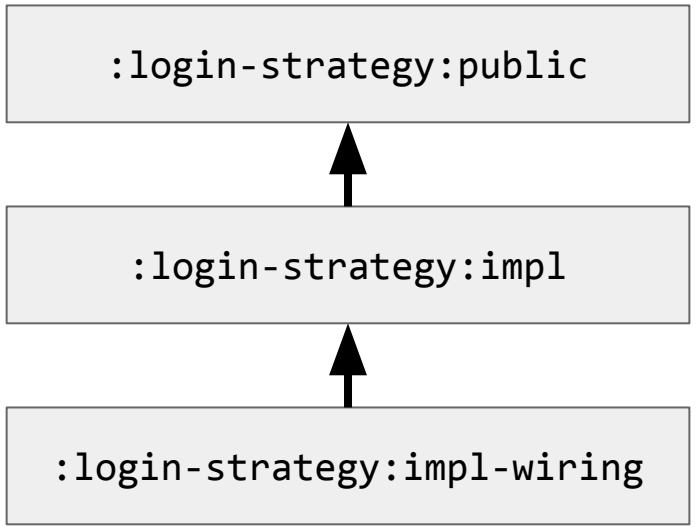
Module Structure



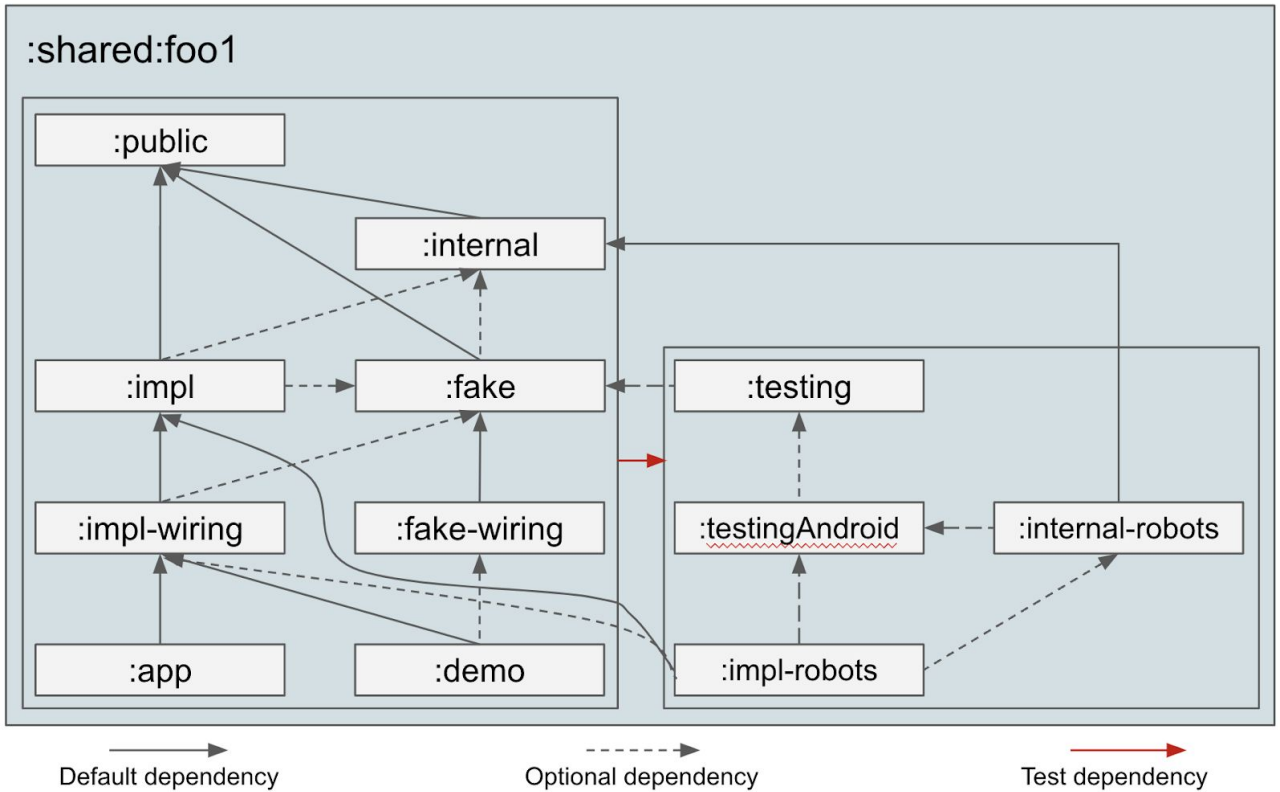
Module Structure



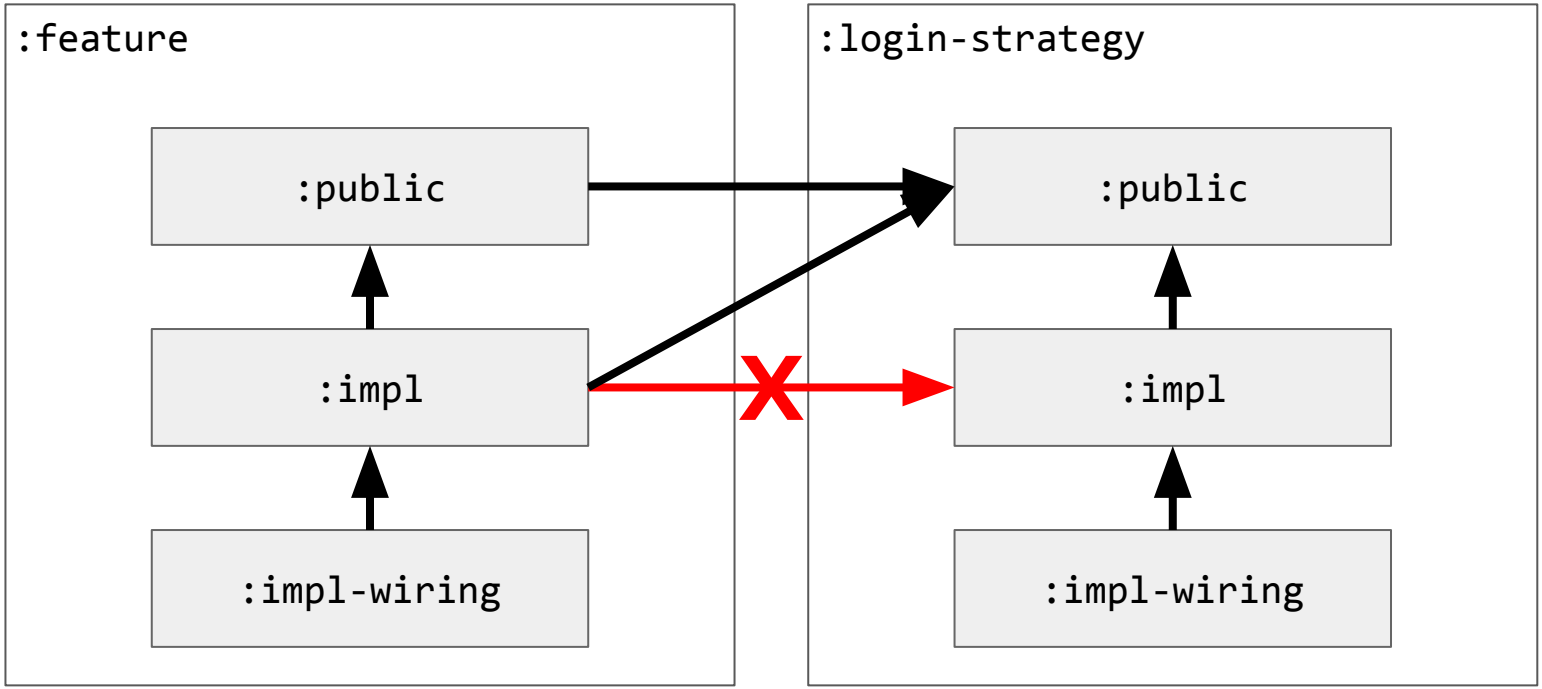
Module Structure



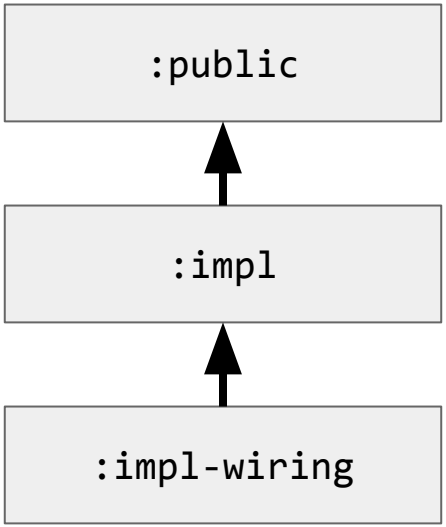
Module Structure



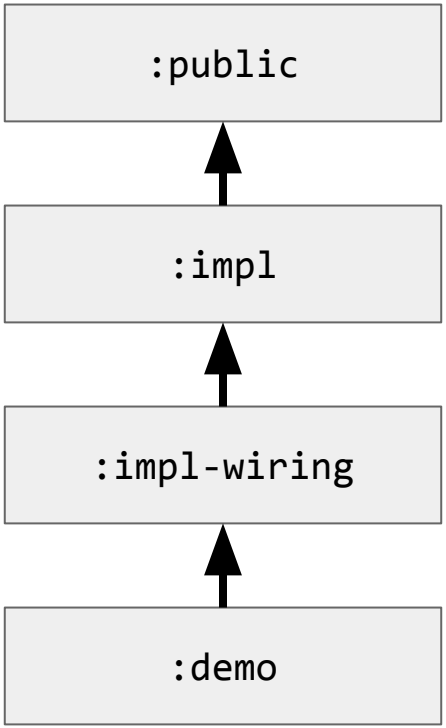
Module Structure



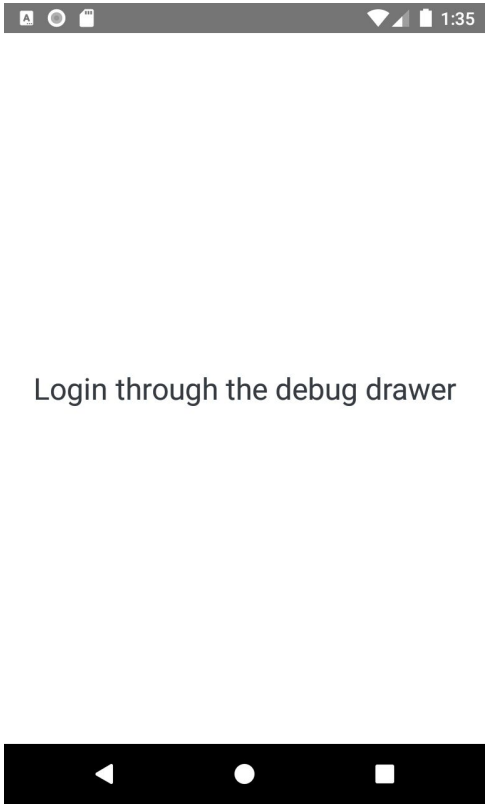
Development Apps

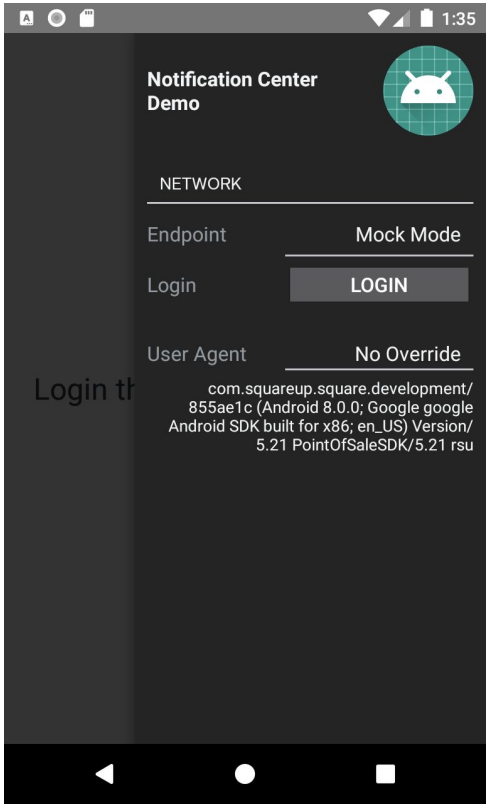


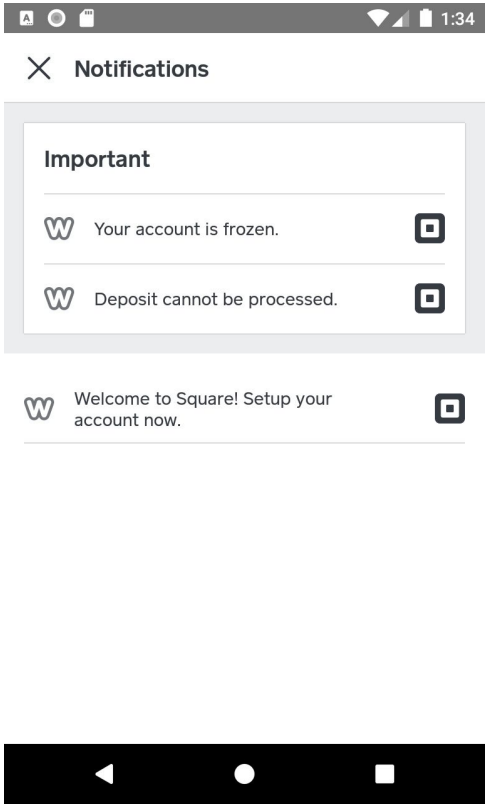
Development Apps



Development Apps





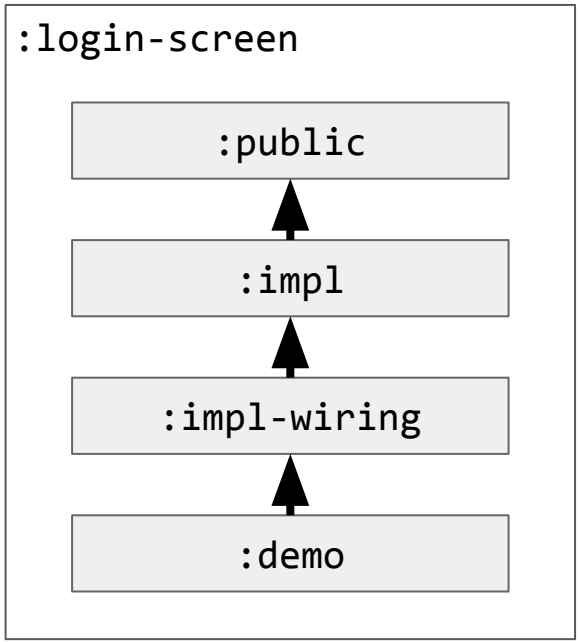


Development Apps

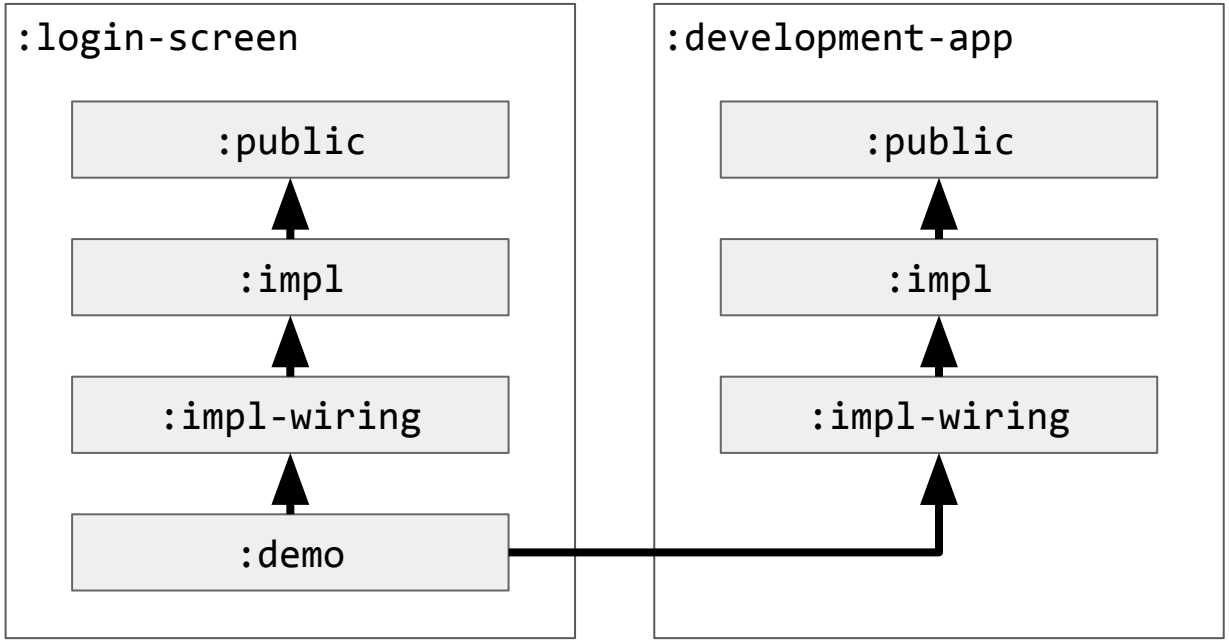
- Install a feature with an isolated application on the device
  - Launch the feature directly
  - Shorter build times

- Easy way to experiment and merge code

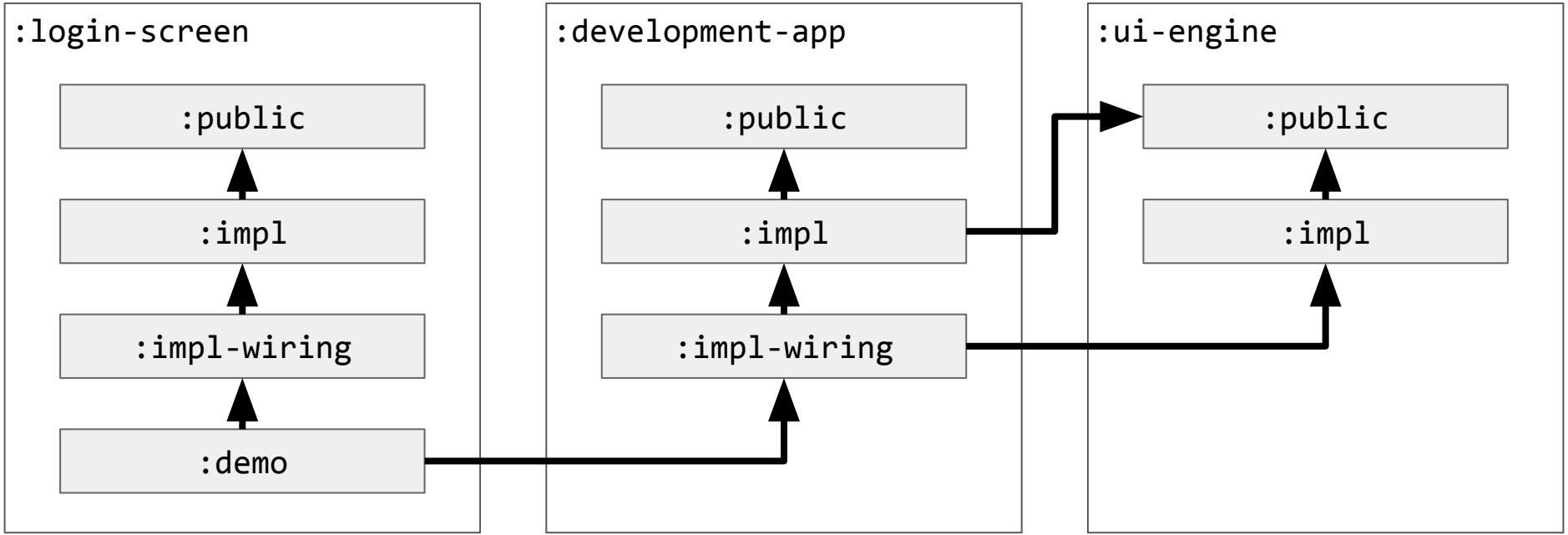
Development Apps



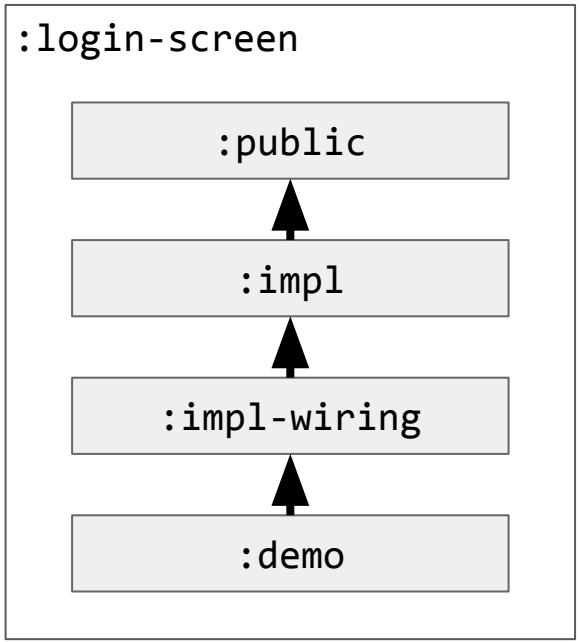
Development Apps



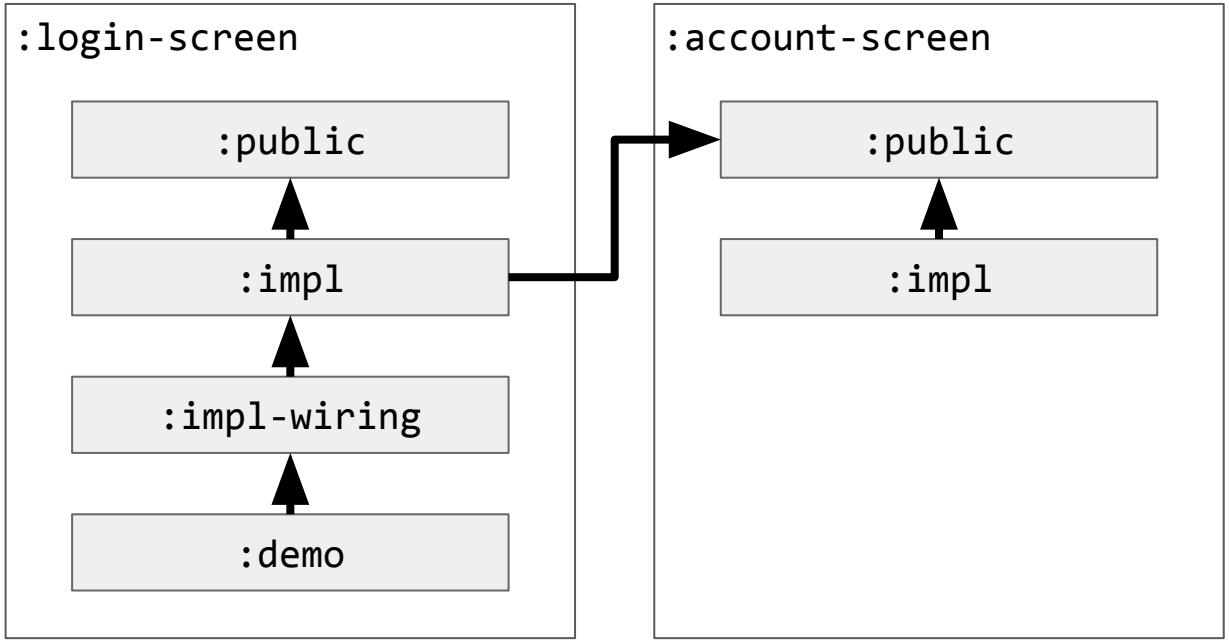
Development Apps



Development Apps



Development Apps



Overview

Workflow is an application framework that provides architectural primitives.

Workflow is:

- Written in and used for Kotlin and Swift
- A unidirectional data flow library that uses immutable data within each Workflow. Data flows in a single direction from source to UI, and events in a single direction from the UI to the business logic.
- A library that supports writing business logic and complex UI navigation logic as state machines, thereby enabling confident reasoning about state and validation of correctness.
- Optimized for composability and scalability of features and screens.
- Corresponding UI frameworks that bind Rendering data classes for “views” (including event callbacks) to Mobile UI frameworks for Android and iOS.
- A corresponding testing framework that facilitates simple-to-write unit tests for all application business logic and helps ensure correctness.

https://square.github.io/workflow/

:account-screen:public

```
interface AccountScreenWorkflow : Workflow<Unit, Unit, LayeredScreen<PosLayering>>
```

:account-screen:impl

```
class RealAccountScreenWorkflow @Inject constructor(
 …
) : AccountScreenWorkflow
```

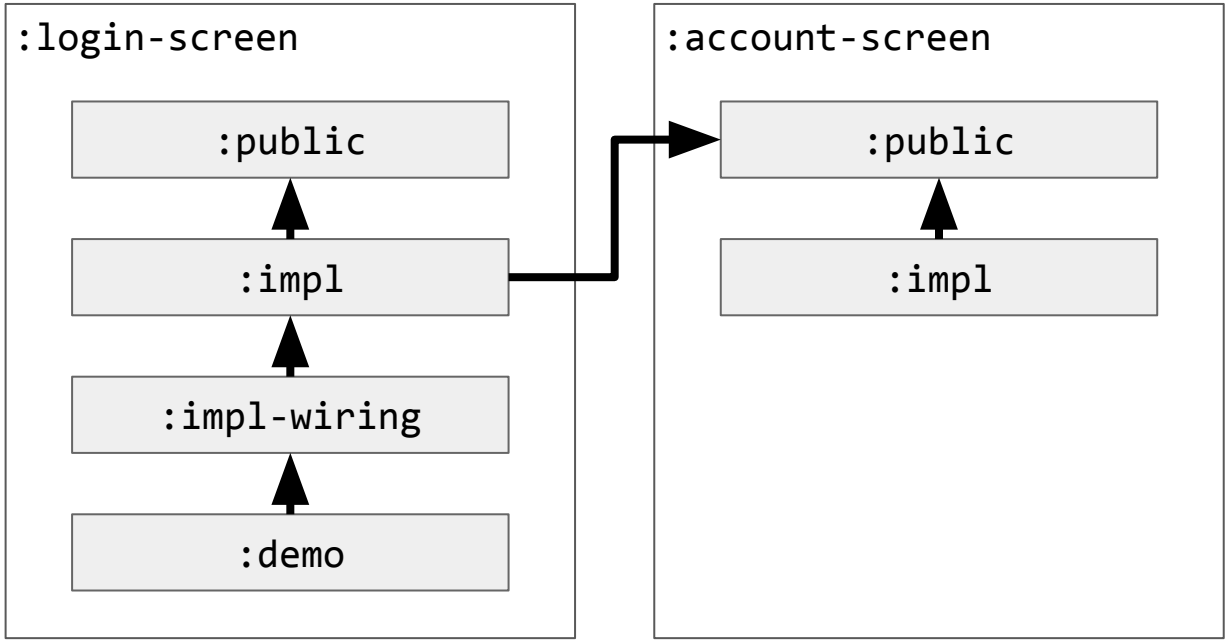
:login-screen:public

```
interface LoginScreenWorkflow : Workflow<Unit, Unit, LayeredScreen<PosLayering>>
```

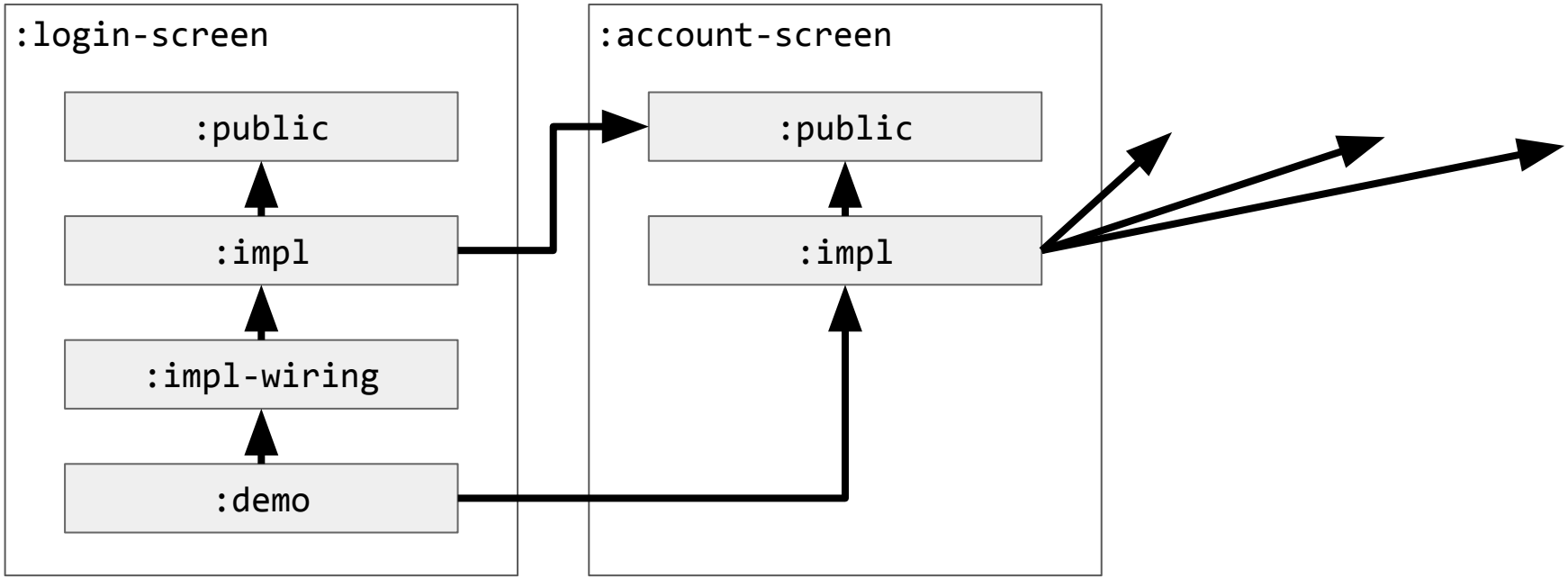
:login-screen:impl

```
class RealLoginScreenWorkflow @Inject constructor(
 private val accountScreenWorkflow: Provider<AccountScreenWorkflow>
) : LoginScreenWorkflow
```

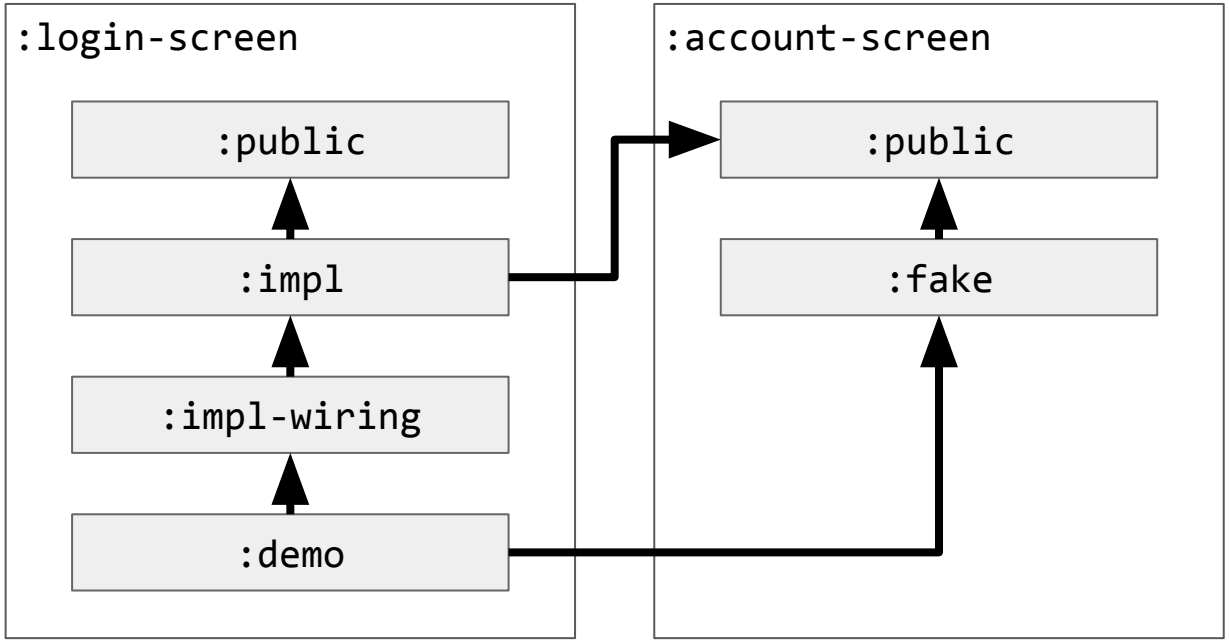
Development Apps



Development Apps



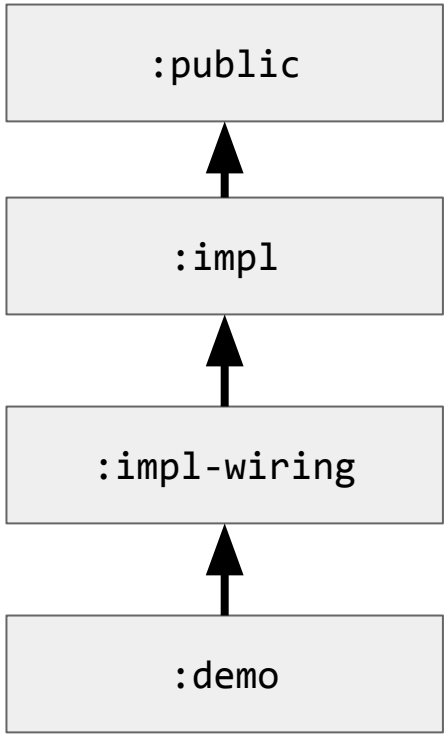
Development Apps



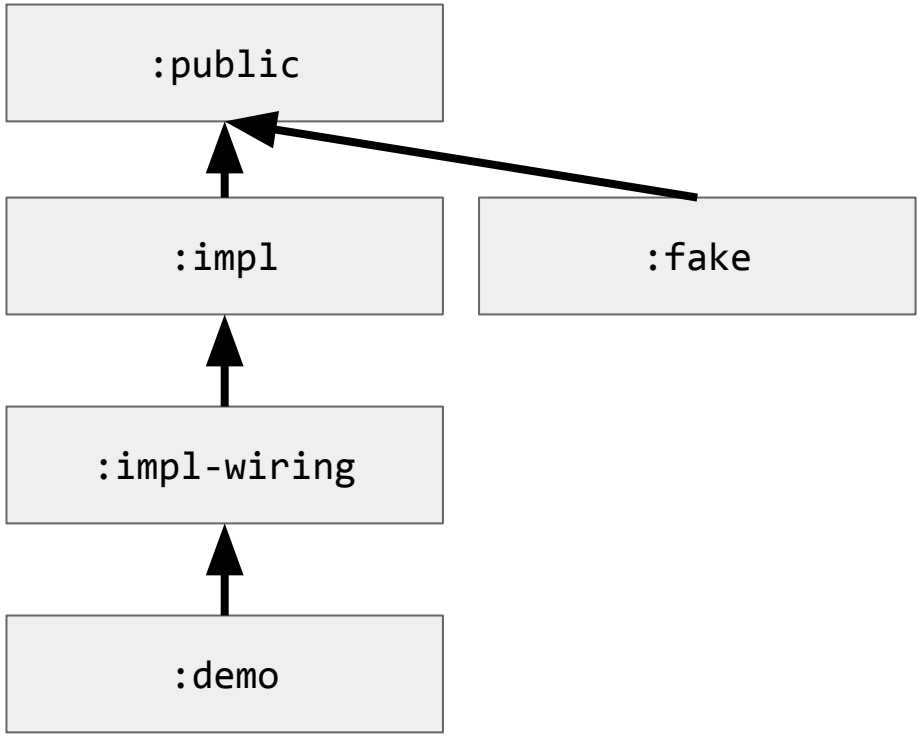
:account-screen:fake

```
class FakeAccountScreenWorkflow @Inject constructor() : AccountScreenWorkflow {
 // ...
}
```

Module Structure



Module Structure



:login-screen:demo

```
android {
  defaultConfig {
    applicationId "com.squareup.sample.login.screen.demo"
  }
}

dependencies {
  // …
  implementation project(':account-screen:fake')
  implementation project(':development-app:impl-wiring)
}
```

:login-screen:demo

```
@DevelopmentAppComponent
class DemoLoginApp : DevelopmentApplication()

@Module
@ContributesTo(ActivityScope::class)
object DemoLoginMainActivityModule {
  @Provides
  fun provideFeatureWorkflowProvider(
    workflow: LoginScreenWorkflow
  ): FeatureProvider =
       FeatureWorkflowProvider(
         workflow = workflow,
         props = Unit,
       )
}
```

:login-screen:demo

```
@DevelopmentAppComponent
class DemoLoginApp : DevelopmentApplication()

@Module
@ContributesTo(ActivityScope::class)
object DemoLoginMainActivityModule {
  @Provides
  fun provideFeatureWorkflowProvider(
    workflow: LoginScreenWorkflow
  ): FeatureProvider =
       FeatureWorkflowProvider(
         workflow = workflow,
         props = Unit,
       )
}
```

:login-screen:demo

```
@DevelopmentAppComponent
class DemoLoginApp : DevelopmentApplication()

@Module
@ContributesTo(ActivityScope::class)
object DemoLoginMainActivityModule {
  @Provides
  fun provideFeatureWorkflowProvider(
    workflow: LoginScreenWorkflow
  ): FeatureProvider =
       FeatureWorkflowProvider(
         workflow = workflow,
         props = Unit,
       )
}
```

UI Tests

- Part of the development apps
  - ➜ Faster builds
  - ➜ Tend to be very stable
  - ➜ More build cache hits
  - ➜ Automatic test shardening

- Integration tests live in the final applications
  - ➜ Test robots are shared across all apps

Mechanisms

Mechanisms: module structure through convention plugins

```
plugins {
   id 'com.android.library
   id 'org.jetbrains.kotlin.android'
   id 'org.jetbrains.kotlin.kapt'
   id 'com.squareup.anvil'
}

dependencies {
   api project(':account-screen:public')
   ...
}


tasks.withType(KotlinCompile).configureEach {
   kotlinOptions {
       jvmTarget = JavaVersion.VERSION_11
   }
}
```

Mechanisms: module structure through convention plugins

```
// :login-screen:impl
plugins {
   id 'com.squareup.android.lib
}


// :login-screen:demo
plugins {
   id 'com.squareup.demo-app
}
```

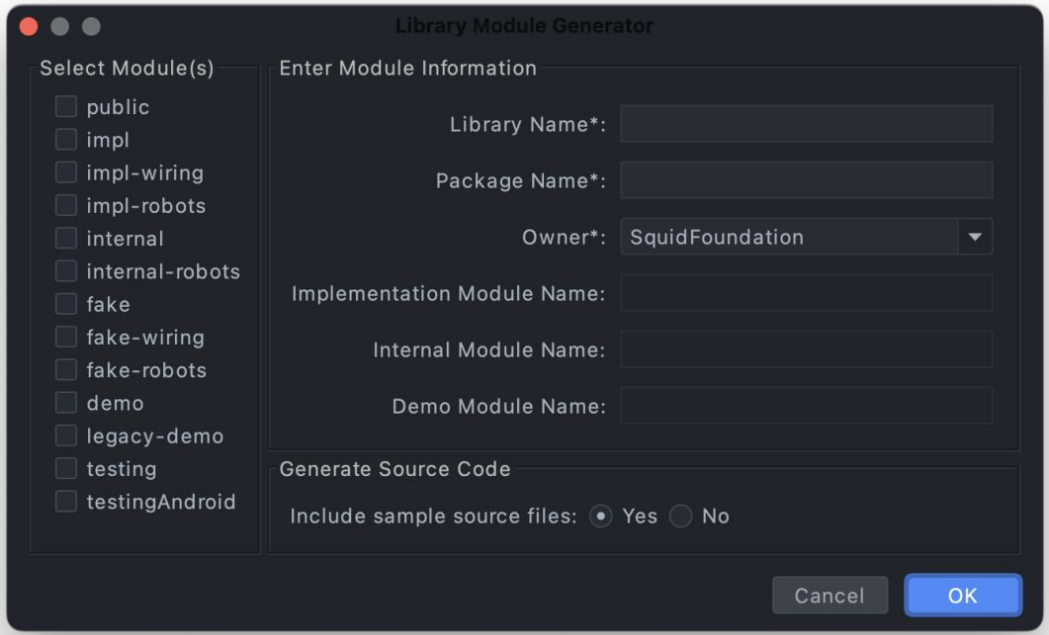
https://developer.squareup.com/blog/herding-elephants/

Mechanisms: module structure enforced through Lint rules

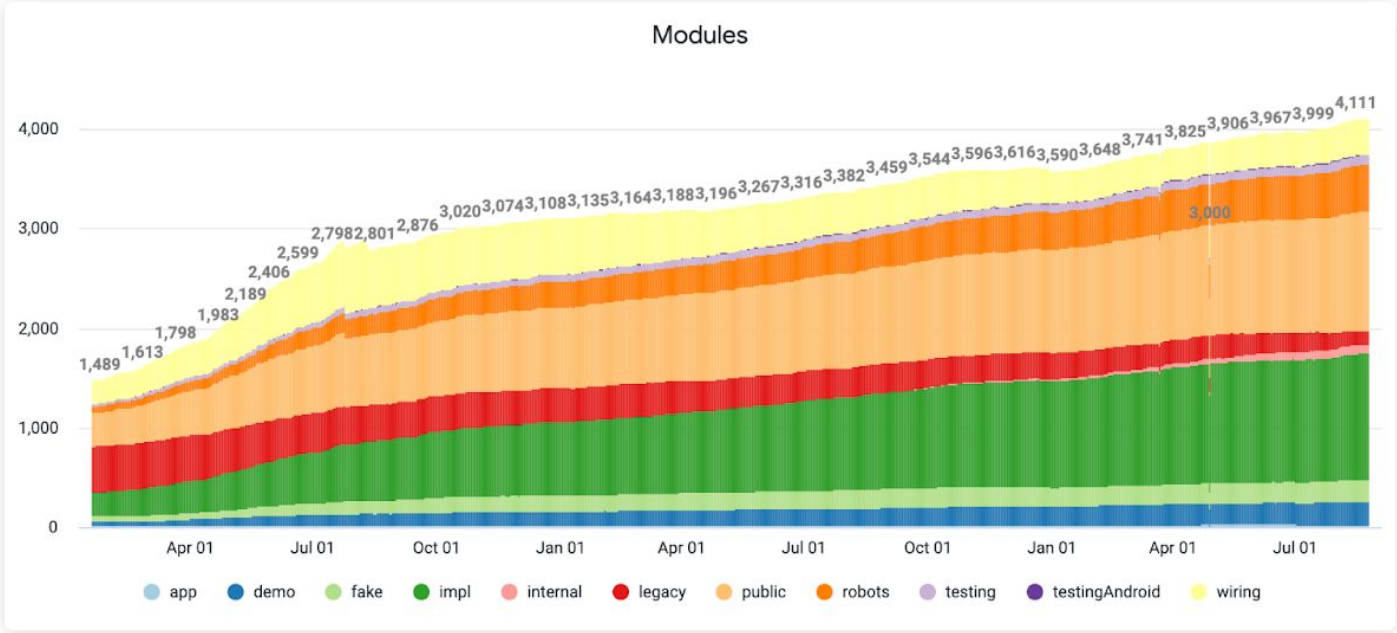
```
class NoImplDependencyModuleCheck : ModuleCheck {
  override fun runCheck(project: Project) {
    …

    fail(
      "No :impl dependency is allowed. Requested: " +
         "${implDependencies.joinToString { it.dependencyName }} in project: $project."
    )
  }
}
```

Mechanisms: library generator



Mechanisms: library generator



4,119

total modules

▲ 6 week over week

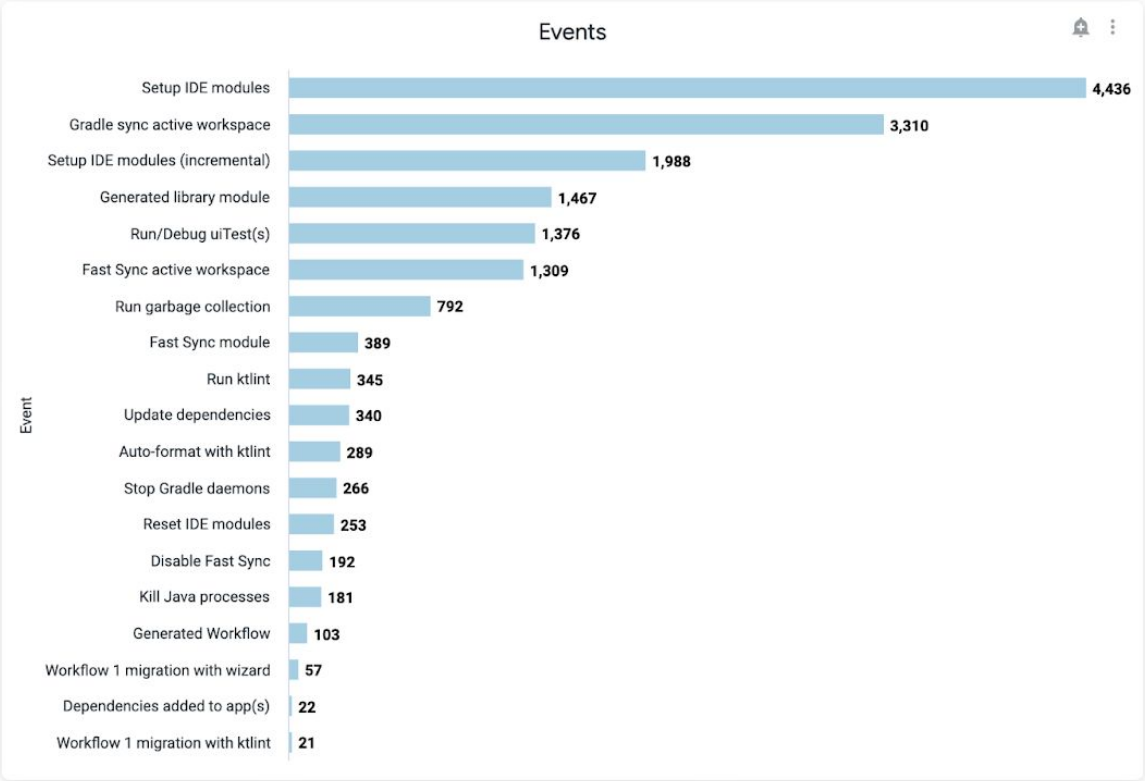
Mechanisms: library generator

111
Monthly Active Users

1092.3
Total Eng Hours Saved

1,467
Modules Generated

103
Workflows Generated



Mechanisms: design patterns & frameworks

- Workflow
  - ➜ Composition was key
  - ➜ https://square.github.io/workflow/
- Dependency inversion
  - ➜ Enforced through the module structure
- Dependency Injection
  - ➜ Anvil: https://github.com/square/anvil/
  - ➜ Dagger 2: https://dagger.dev/

Mechanisms: design patterns & frameworks: Anvil

```
class RealLoginScreenWorkflow @Inject constructor(
 private val accountScreenWorkflow: Provider<AccountScreenWorkflow>
) : LoginScreenWorkflow
```

https://github.com/square/anvil/

Mechanisms: design patterns & frameworks: Anvil

```
class RealLoginScreenWorkflow @Inject constructor(
 private val accountScreenWorkflow: Provider<AccountScreenWorkflow>
) : LoginScreenWorkflow

@Module
abstract class LoginScreenWorkflowModule {
   @Binds abstract fun bindLoginScreenWorkflow(
       workflow: RealLoginScreenWorkflow
   ) : LoginScreenWorkflow
}
```

https://github.com/square/anvil/

Mechanisms: design patterns & frameworks: Anvil

```
class RealLoginScreenWorkflow @Inject constructor(
 private val accountScreenWorkflow: Provider<AccountScreenWorkflow>
) : LoginScreenWorkflow

@Module
abstract class LoginScreenWorkflowModule {
   @Binds abstract fun bindLoginScreenWorkflow(
       workflow: RealLoginScreenWorkflow
   ) : LoginScreenWorkflow
}

@Component(
   modules = {
       LoginScreenWorkflowModule::class,
   }
)
interface AppComponent
```

https://github.com/square/anvil/

Mechanisms: design patterns & frameworks: Anvil

```
@ContributesBinding(ActivityScope::class)
class RealLoginScreenWorkflow @Inject constructor(
 private val accountScreenWorkflow: Provider<AccountScreenWorkflow>
) : LoginScreenWorkflow
```

https://github.com/square/anvil/

Mechanisms: design patterns & frameworks: Anvil

```
@ContributesBinding(ActivityScope::class)
class RealLoginScreenWorkflow @Inject constructor(
 private val accountScreenWorkflow: Provider<AccountScreenWorkflow>
) : LoginScreenWorkflow
```

https://github.com/square/anvil/

Mechanisms: design patterns & frameworks: Anvil

```
@DevelopmentAppComponent
class DemoLoginApp : DevelopmentApplication()

@Module
@ContributesTo(ActivityScope::class)
object DemoLoginMainActivityModule {
  @Provides
  fun provideFeatureWorkflowProvider(
    workflow: LoginScreenWorkflow
  ): FeatureProvider =
       FeatureWorkflowProvider(
         workflow = workflow,
         props = Unit,
       )
}
```

https://github.com/square/anvil/

Mechanisms: design patterns & frameworks: Anvil

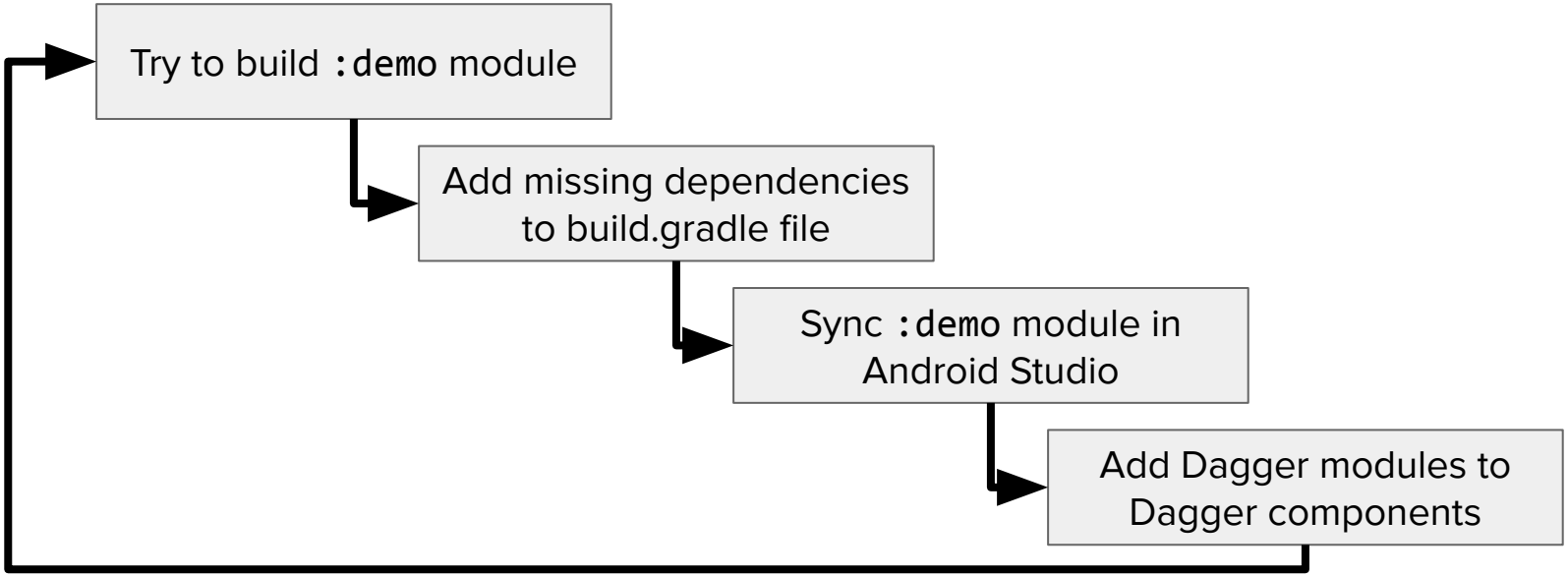
```
@DevelopmentAppComponent
class DemoLoginApp : DevelopmentApplication()

@Module
@ContributesTo(ActivityScope::class)
object DemoLoginMainActivityModule {
  @Provides
  fun provideFeatureWorkflowProvider(
    workflow: LoginScreenWorkflow
  ): FeatureProvider =
      FeatureWorkflowProvider(
        workflow = workflow,
        props = Unit,
      )
}
```

https://github.com/square/anvil/

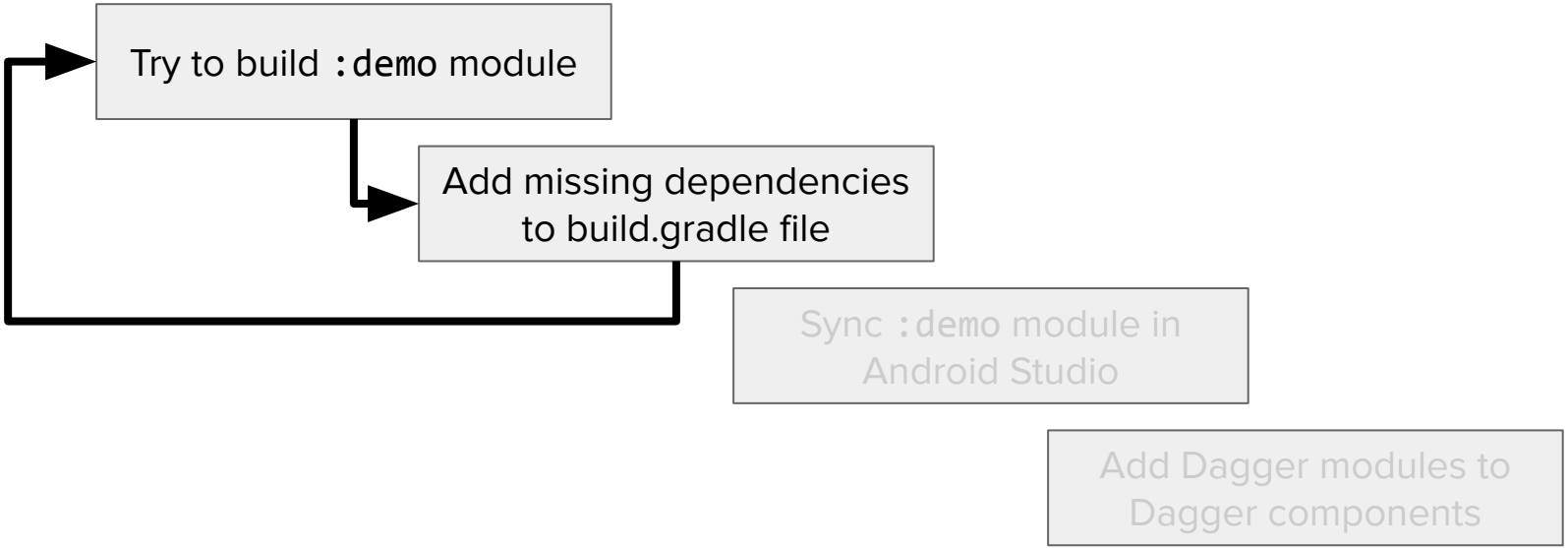
https://bit.ly/3WbPHdm

Mechanisms: design patterns & frameworks: Anvil



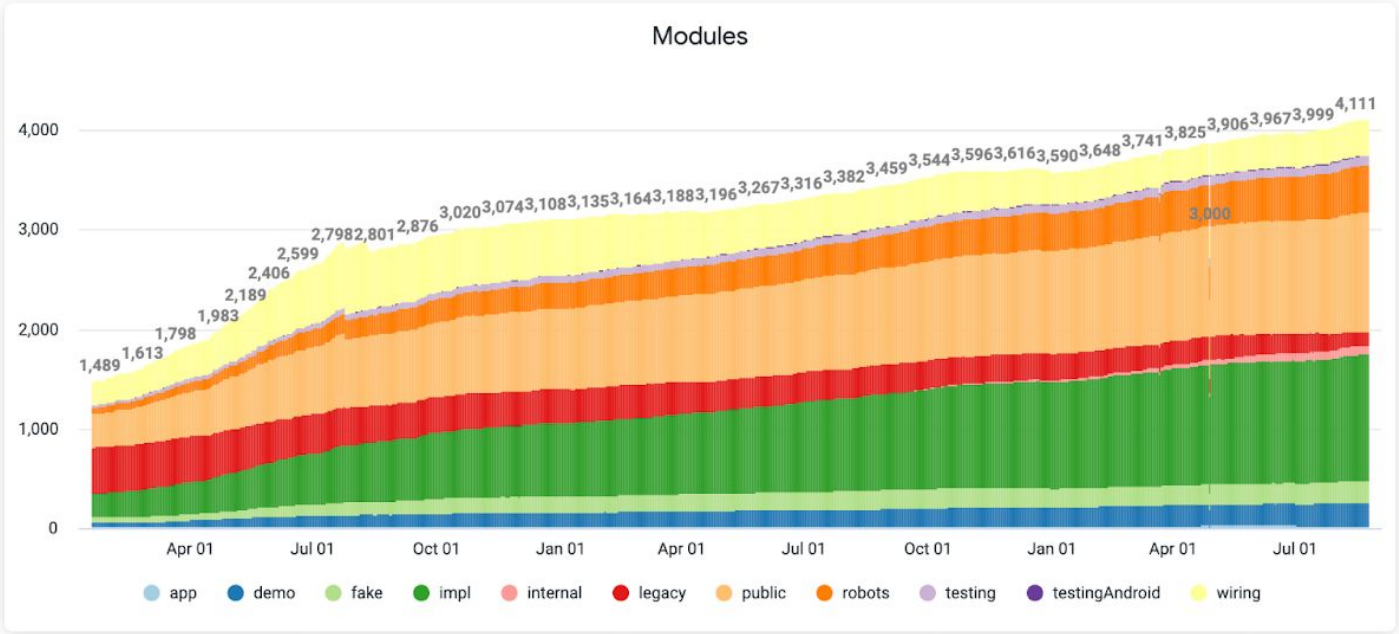
https://bit.ly/3WbPHdm

Mechanisms: design patterns & frameworks: Anvil



https://bit.ly/3WbPHdm

Mechanisms: dashboards

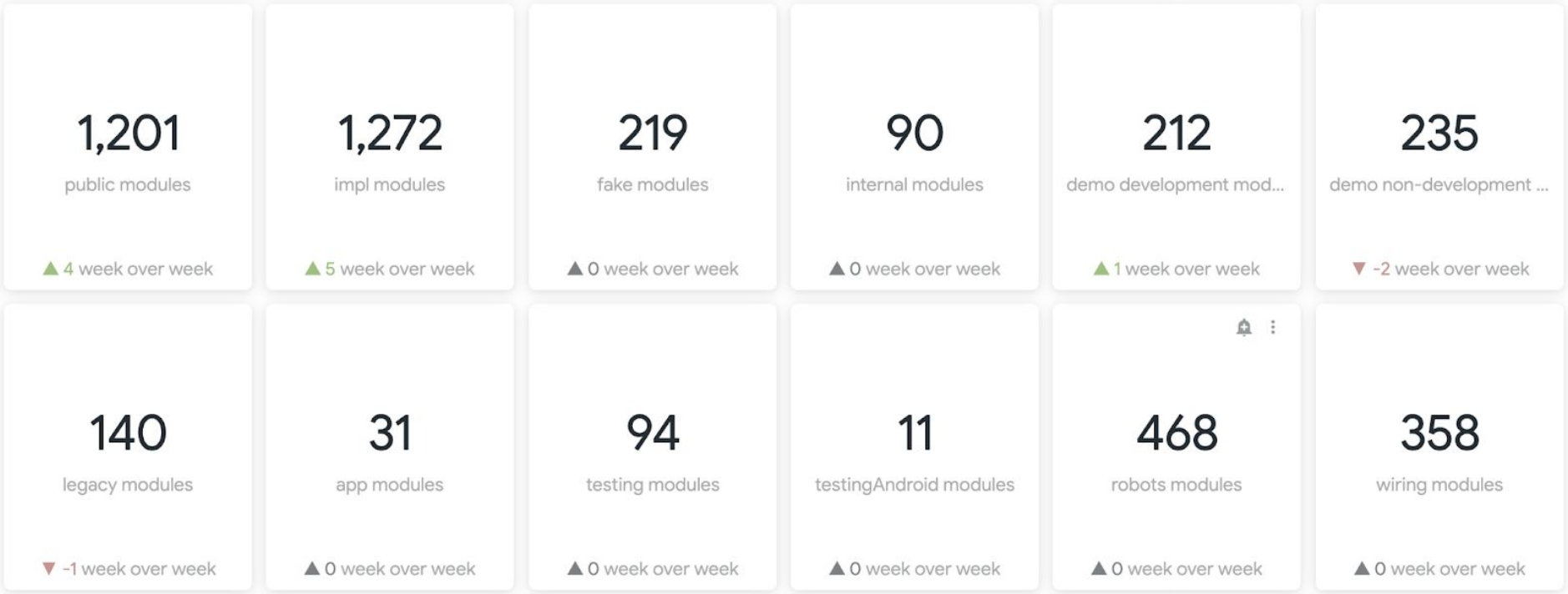


4,119

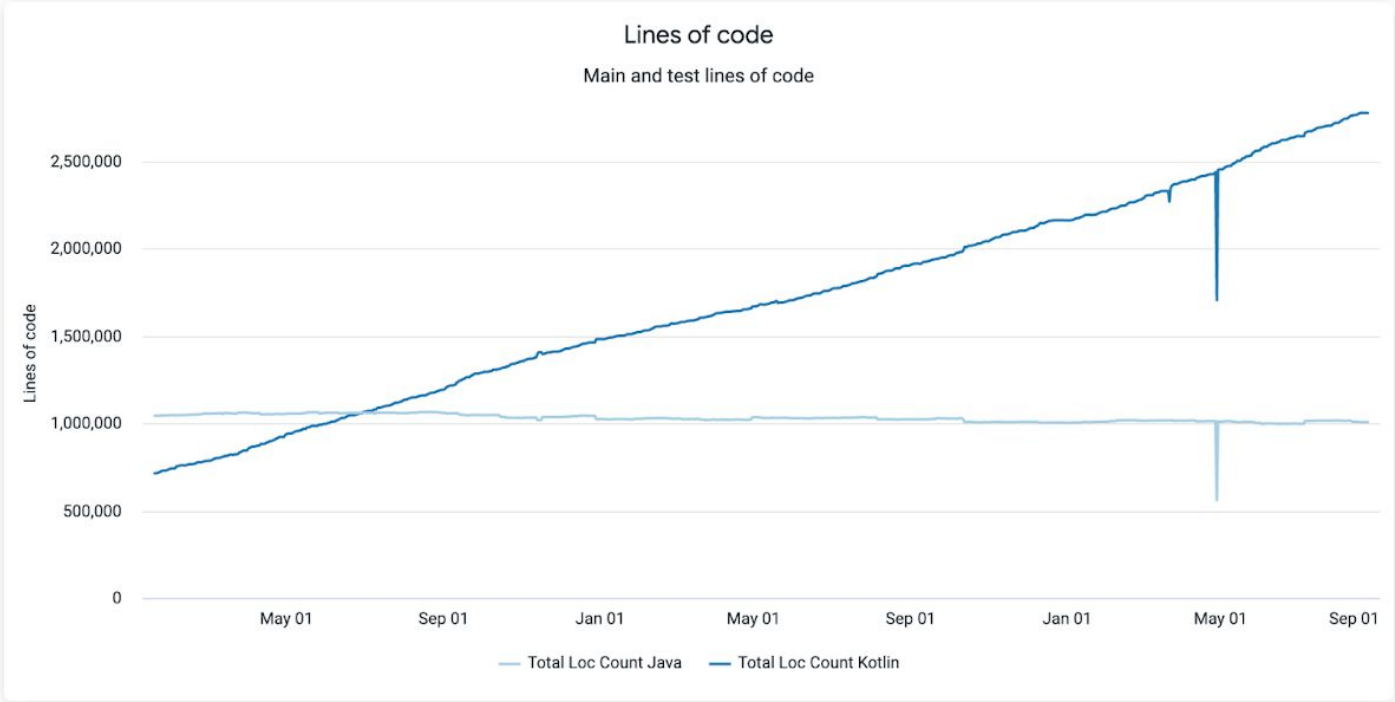
total modules

▲ 6 week over week

Mechanisms: dashboards



Mechanisms: dashboards



1,007,293

LOC Java

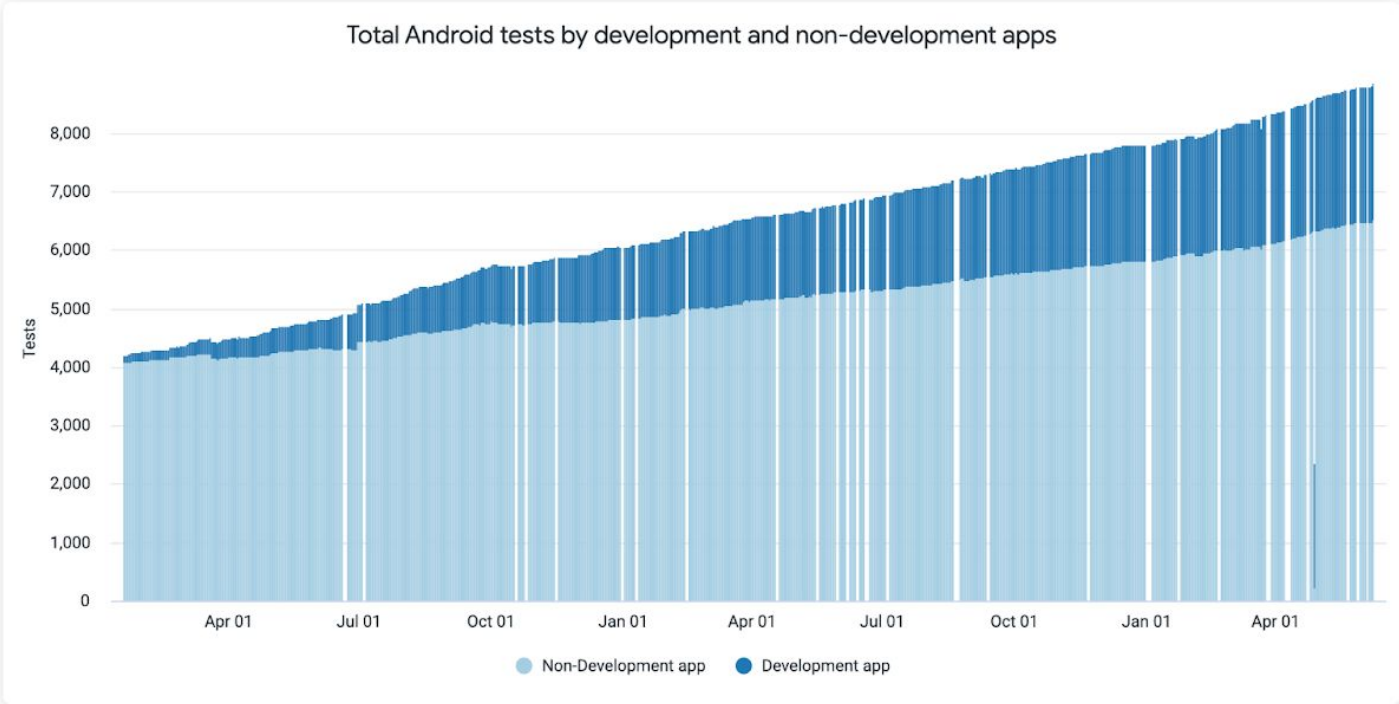
▼ -251 week over week

2,777,371

LOC Kotlin

▲ 2,604 week over week

Mechanisms: dashboards



2,338

Android development tests

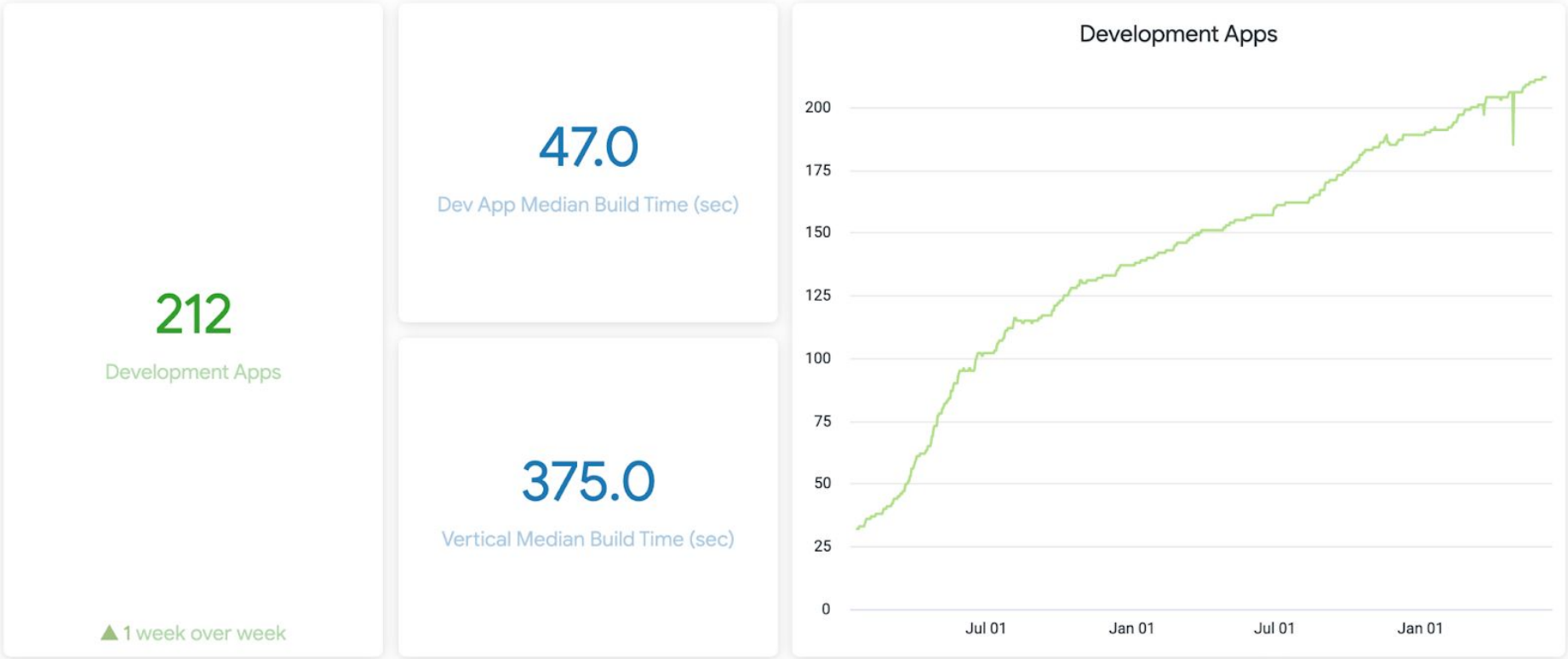
▲ 10 week over week

9,386

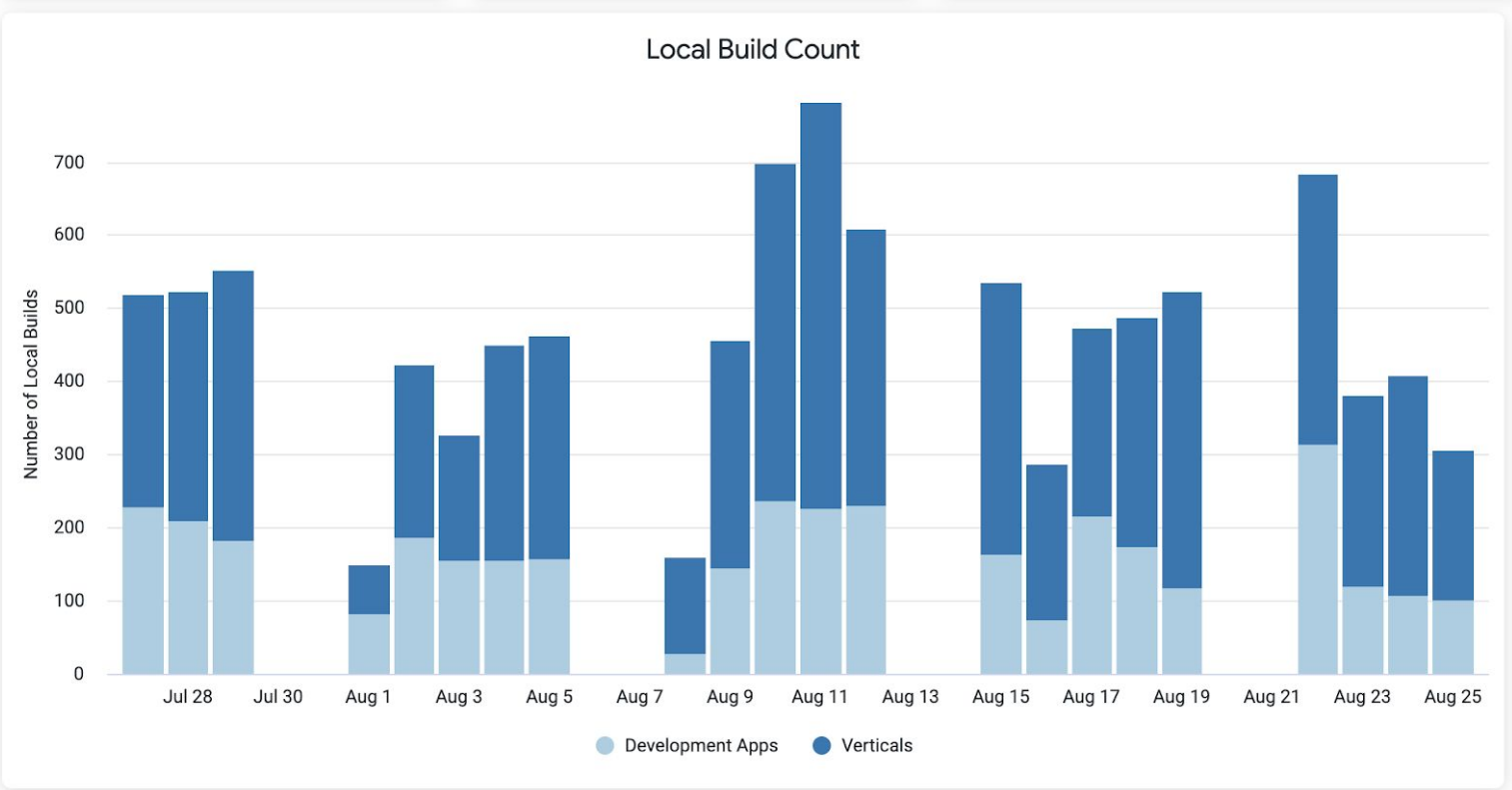
Android non-development tests

▲ 6 week over week

Mechanisms: dashboards



Mechanisms: dashboards



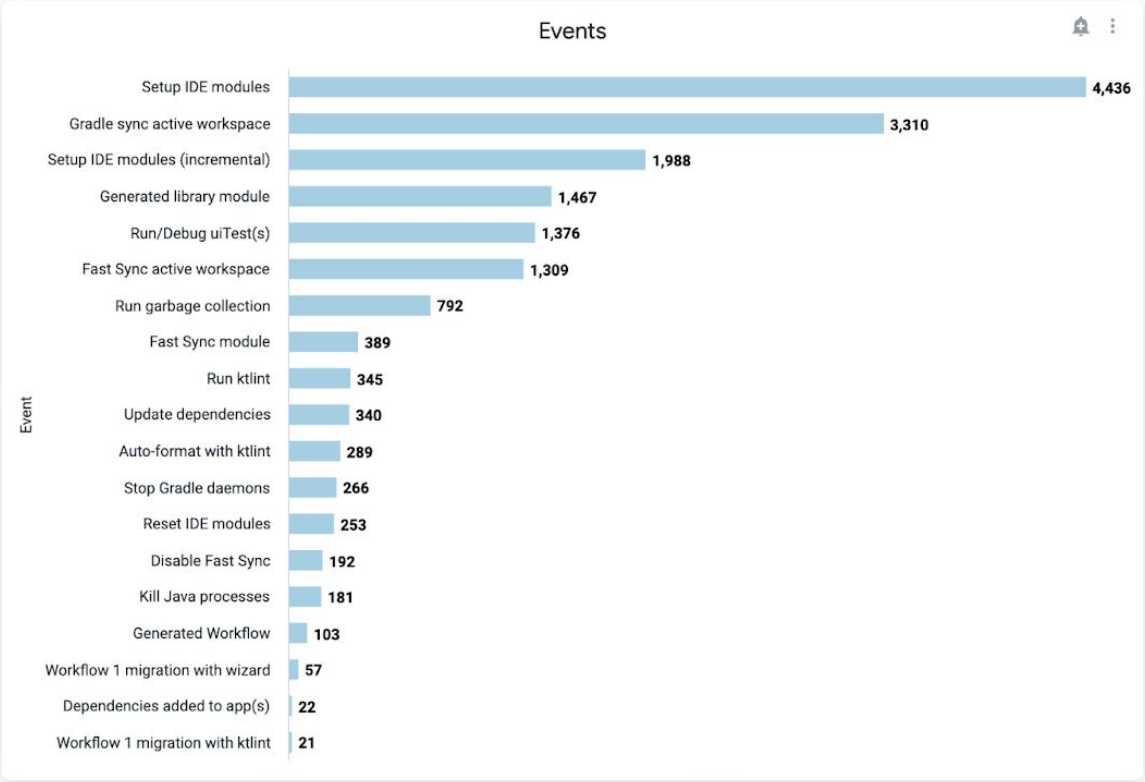
Mechanisms: dashboards

111
Monthly Active Users

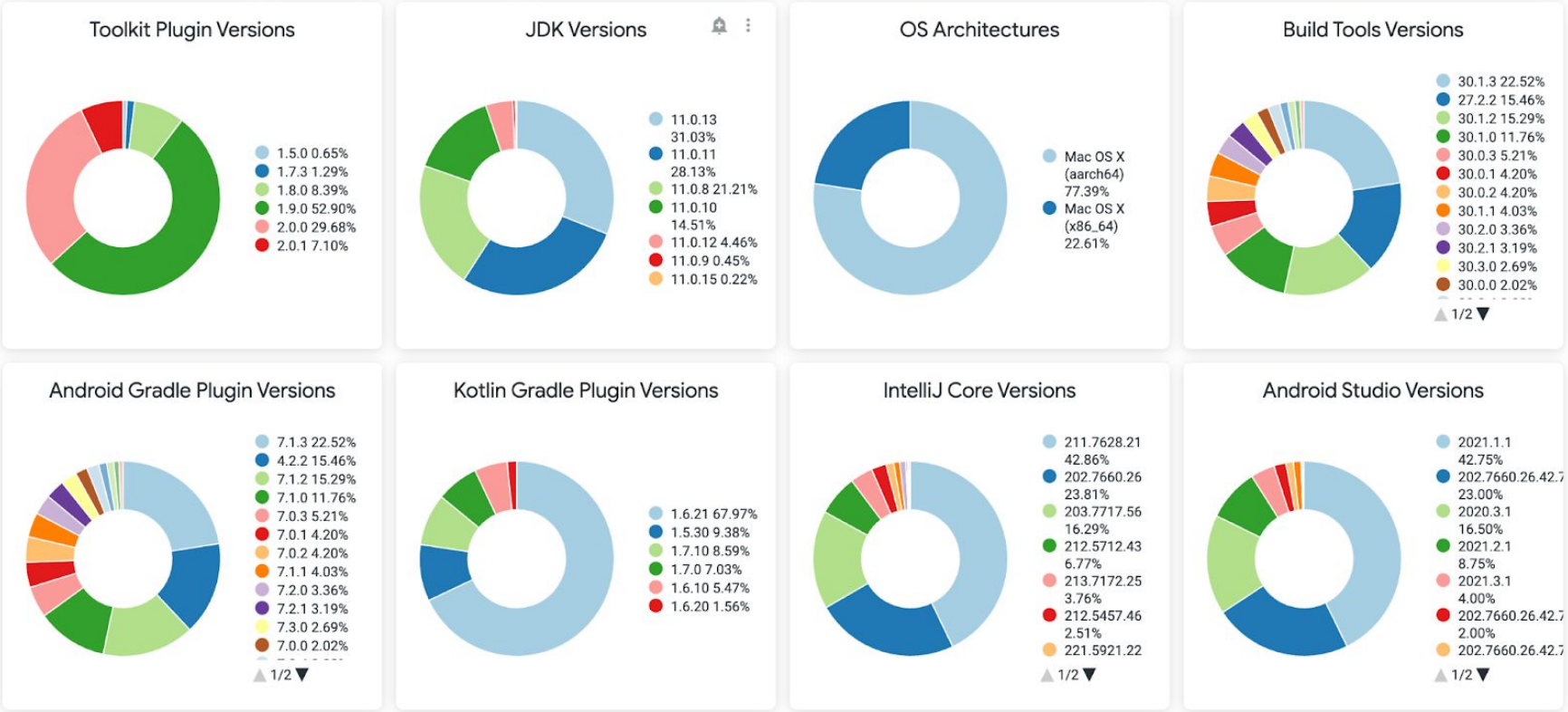
1092.3
Total Eng Hours Saved

1,467
Modules Generated

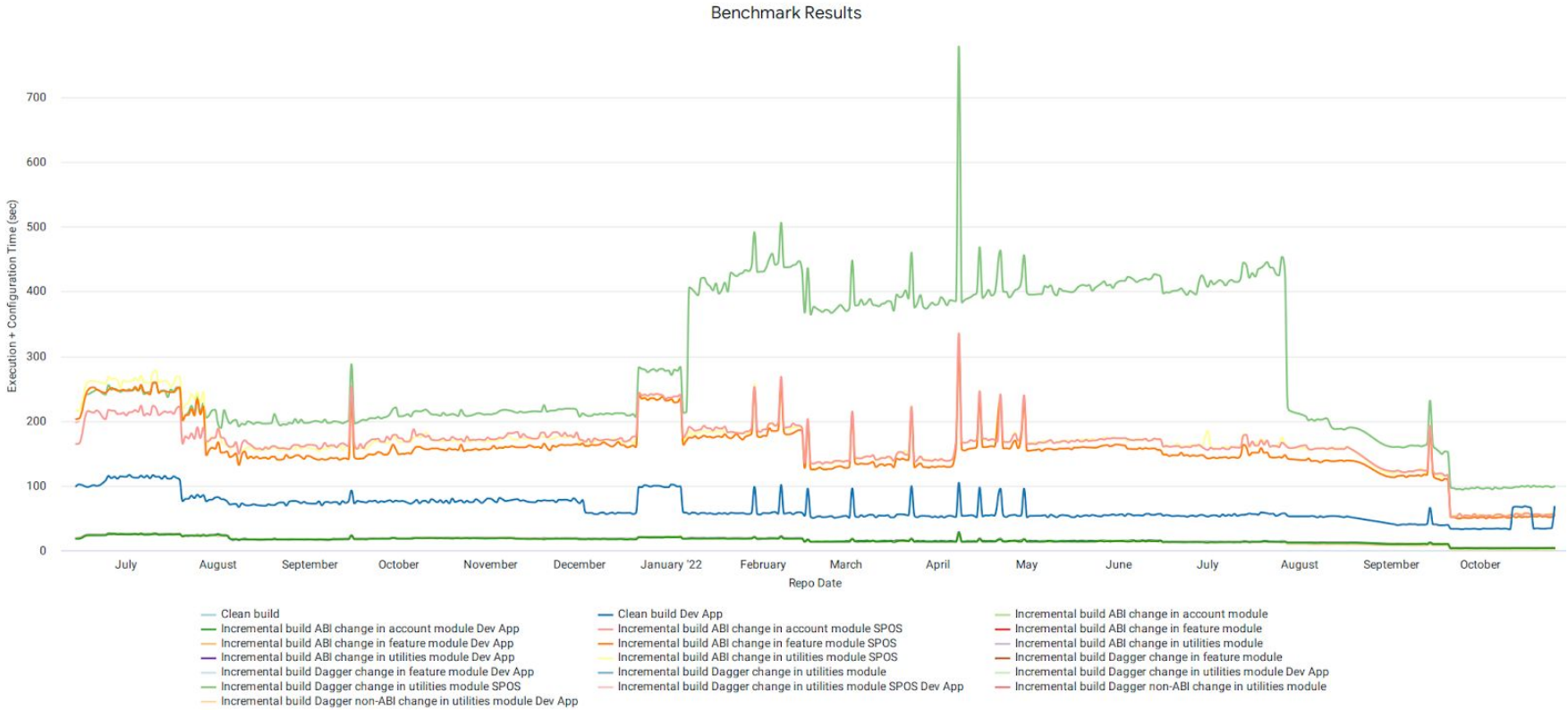
103
Workflows Generated



Mechanisms: dashboards

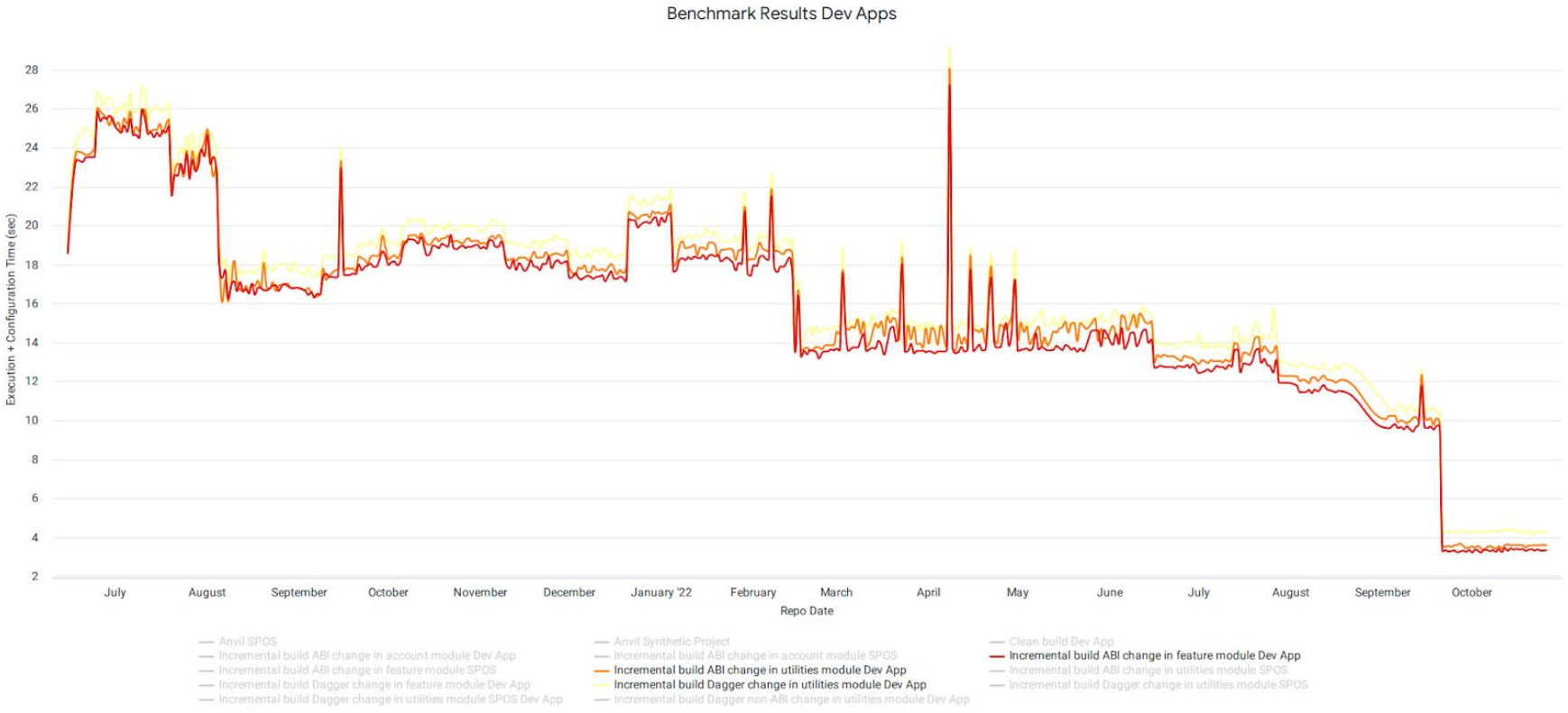


Mechanisms: benchmarks



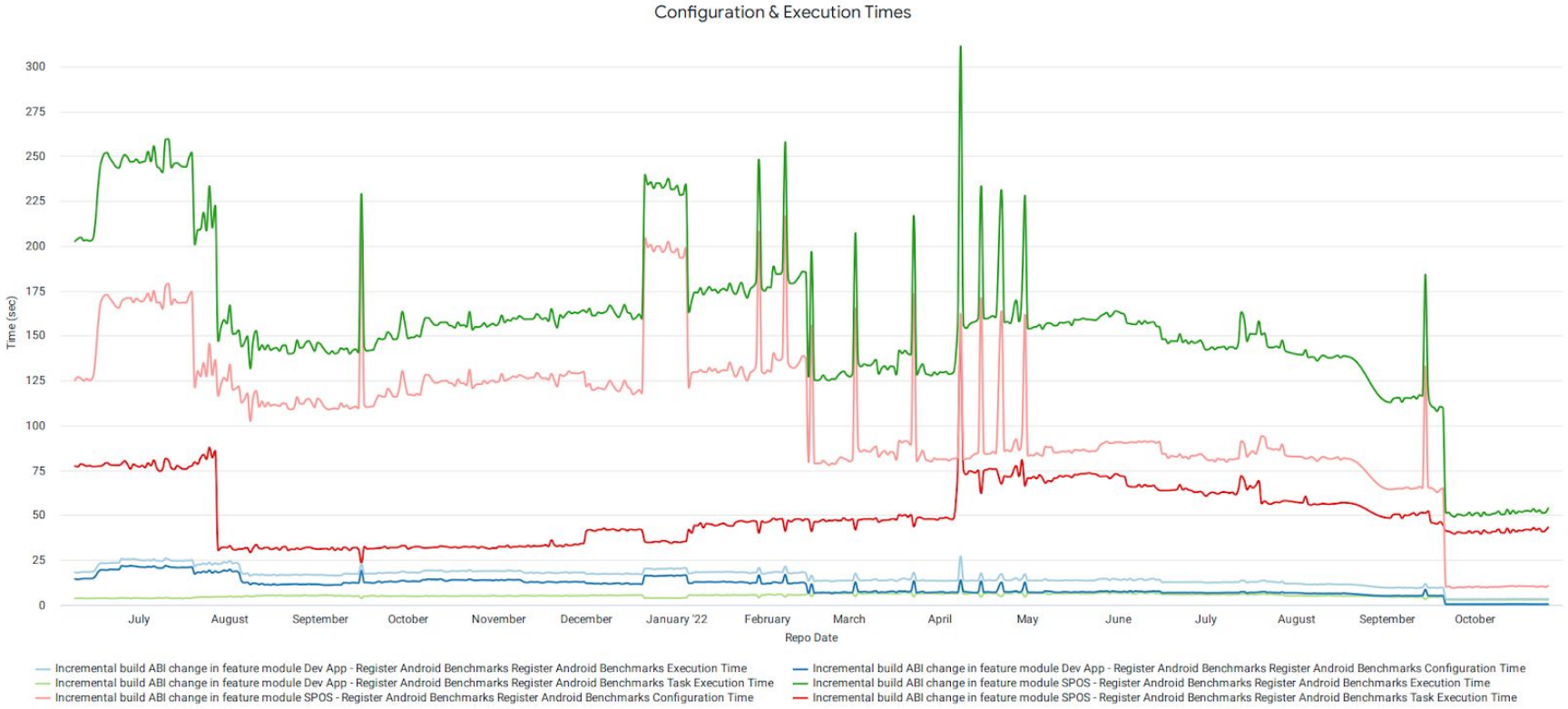
https://developer.squareup.com/blog/measure-measure-measure/

Mechanisms: benchmarks



https://developer.squareup.com/blog/measure-measure-measure/

Mechanisms: benchmarks



https://developer.squareup.com/blog/measure-measure-measure/

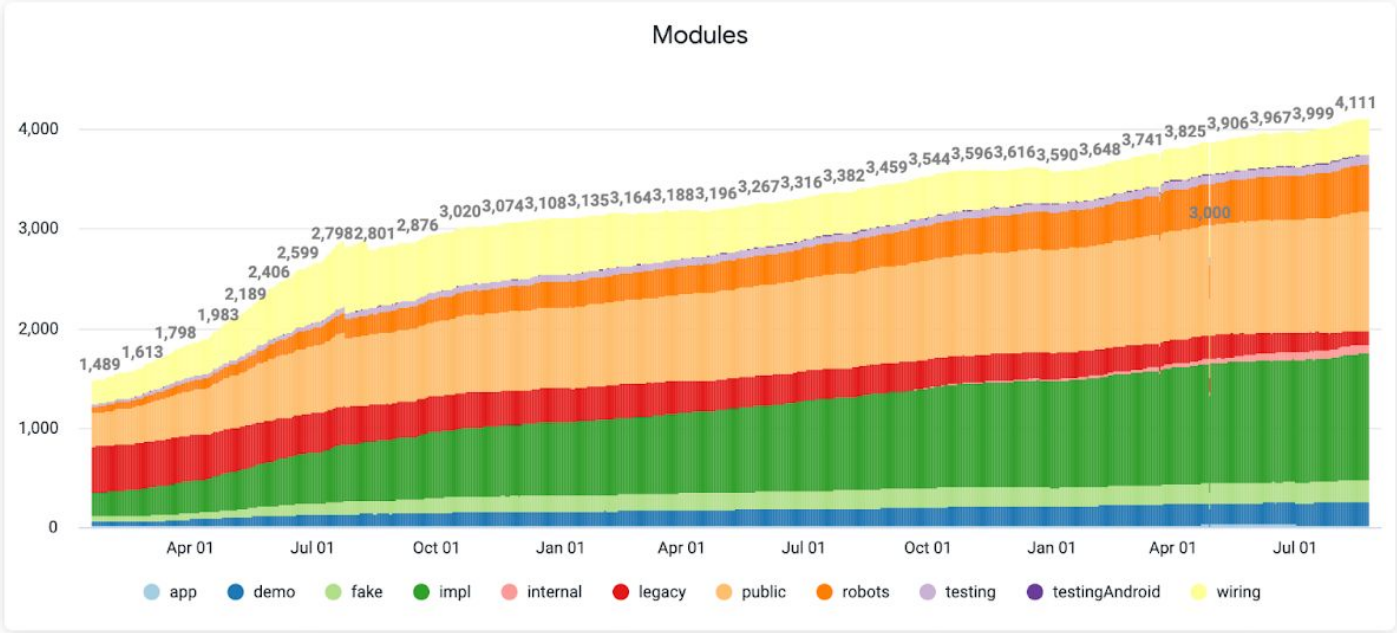
Mechanisms: migrations

- We were a small team and relied on the help of feature teams to contribute to common goals
  - ➜ Legacy module migration
  - ➜ Adopt Anvil
  - ➜ Extract UI tests into `:demo` apps

https://www.droidcon.com/2022/09/29/migration-without-migraines-automatic-migration-at-scale/

Challenges

Challenges: module count



4,119

total modules

▲ 6 week over week

Challenges: module count

Solution:

- Convention plugins
- Benchmarks
- Anvil
- Configuration caching
- Partial IDE sync
  ➜ https://github.com/dropbox/focus

Challenges: legacy modules

Legacy modules break our dependency rules

Solution:

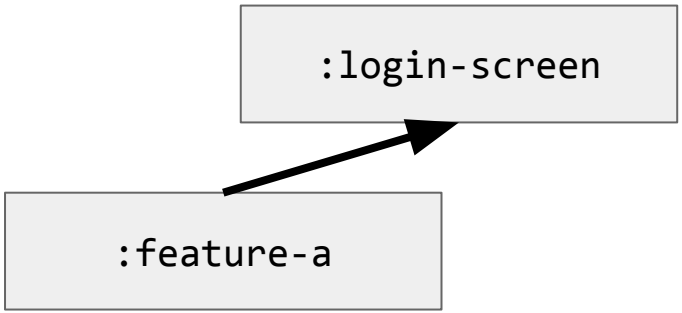
- Finish migration

Challenges: granular modules

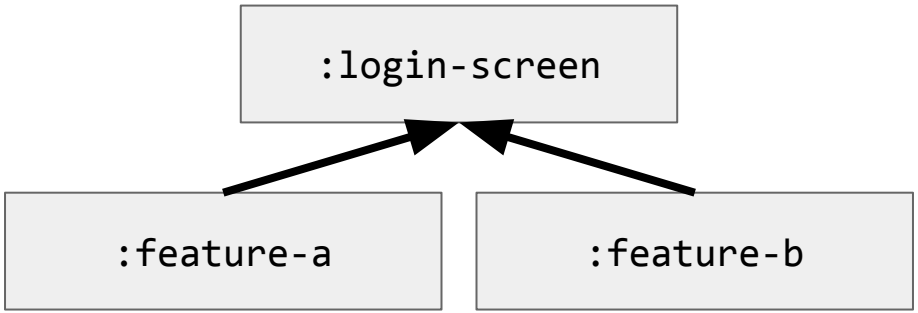
Anti-pattern

Challenges: granular modules

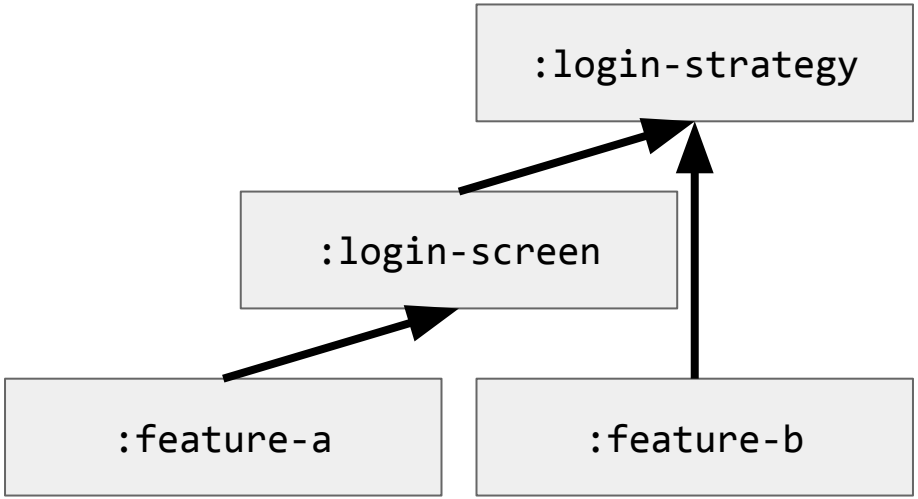
Challenges: granular modules



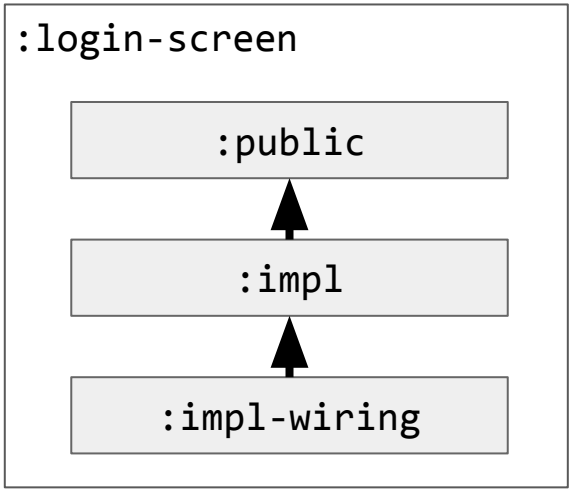
Challenges: granular modules



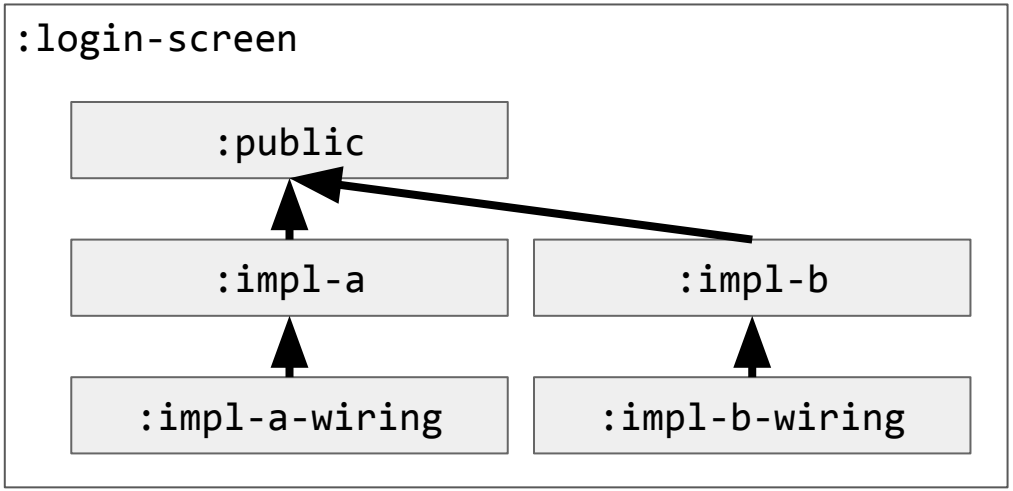
Challenges: granular modules



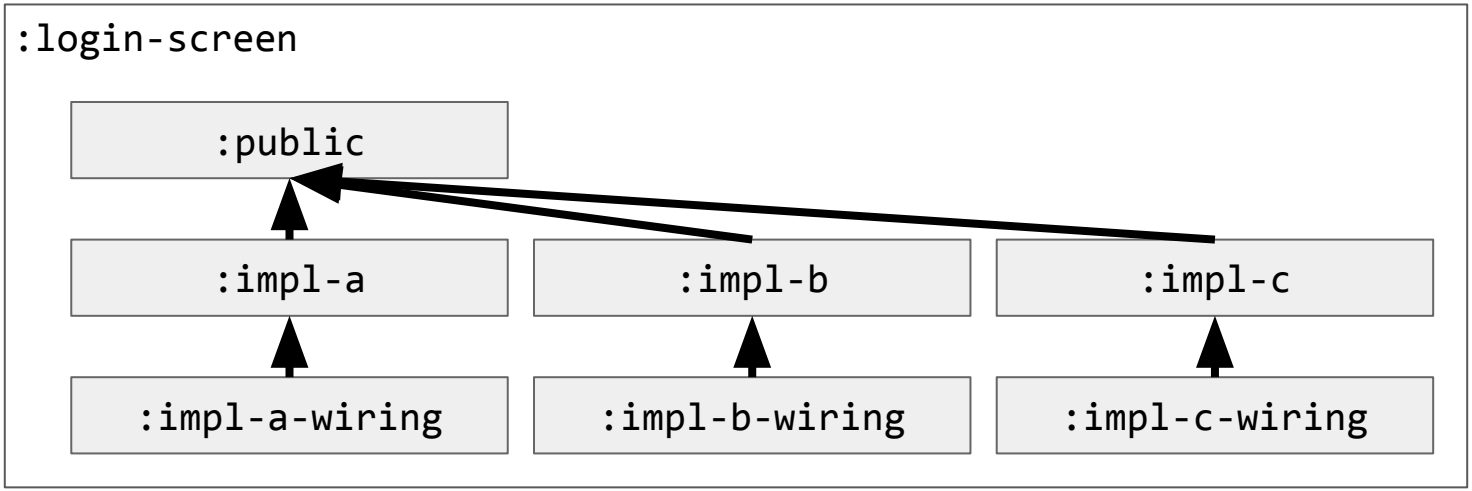
Challenges: granular modules



Challenges: granular modules

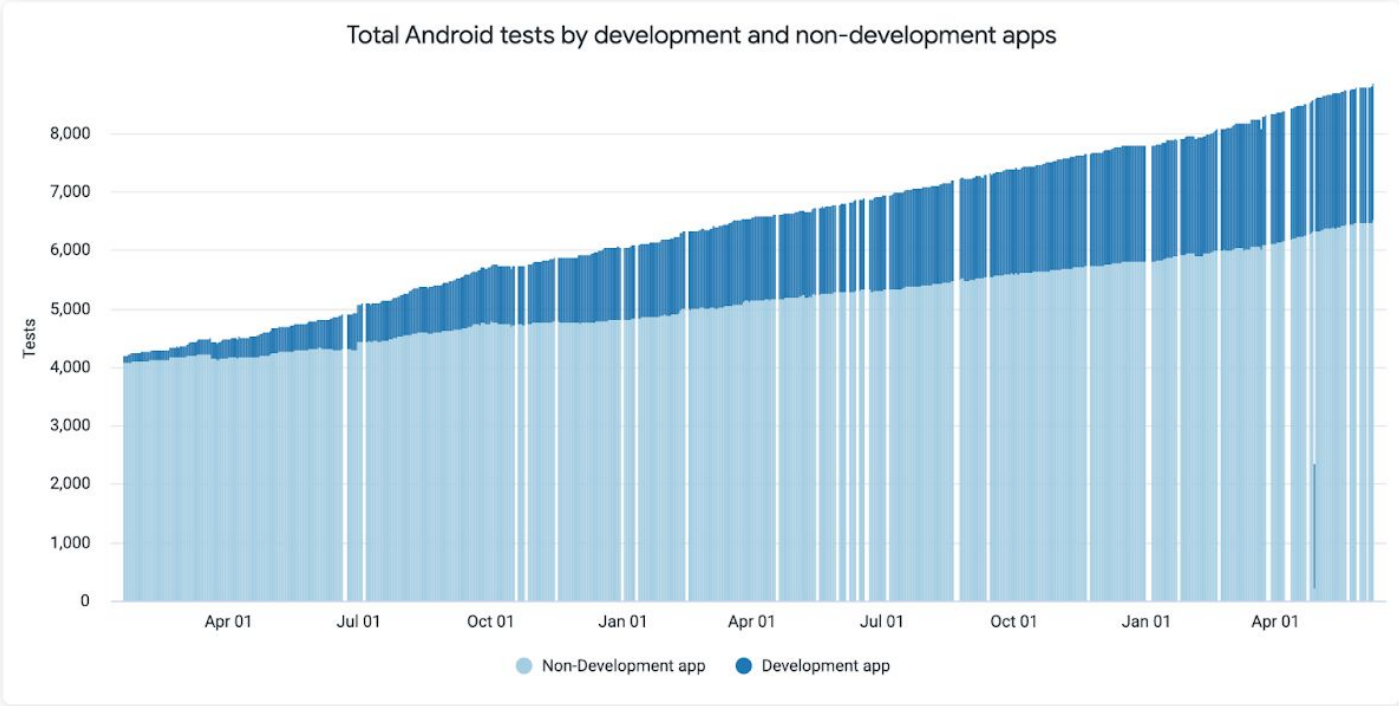


Challenges: granular modules



Challenges: development app shell

Challenges: extract UI tests



2,338

Android development tests

▲ 10 week over week

9,386

Android non-development tests

▲ 6 week over week

Isolated Development

Ralf Wondratschek

@vRallev

amazon