Streamlining iOS* Development with Selective Testing

Michael 'Mike' Gerasymenko

*Actually, any Swift code


600
TECHNOLOGY
1500
LIBERAL ARTS

Hey Mike...

We are interested in your talk!

Just...

Can it be a lightning talk?


Lightning
Talk?!

How many of you are writing tests? No blaming!

Happy with how long it takes to run them?

Your mobile application

There are some core jobs it is doing:

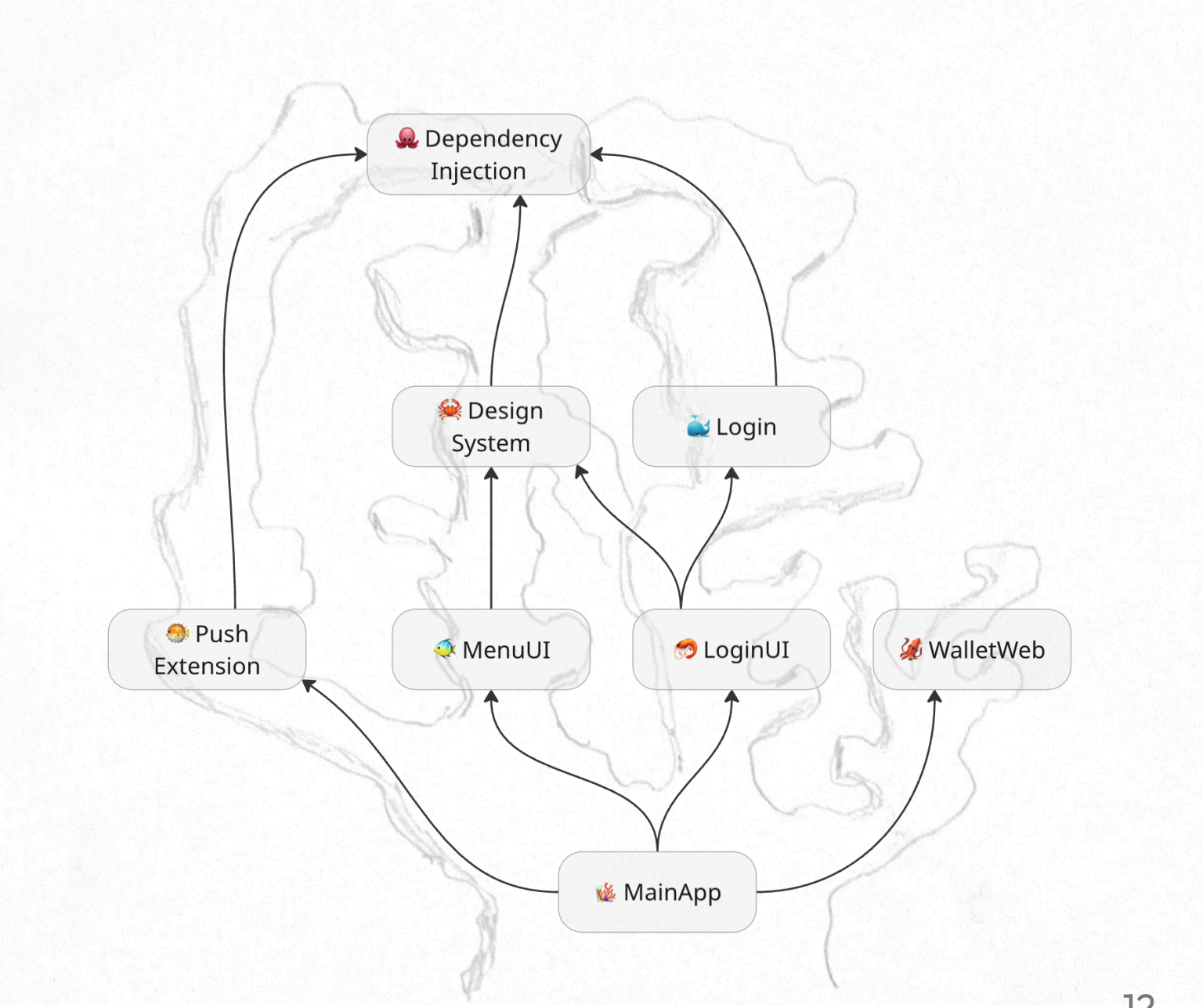
- Login and Registration,
- Preferences,
- Home UI,
- Details UI,
- ...

Growth

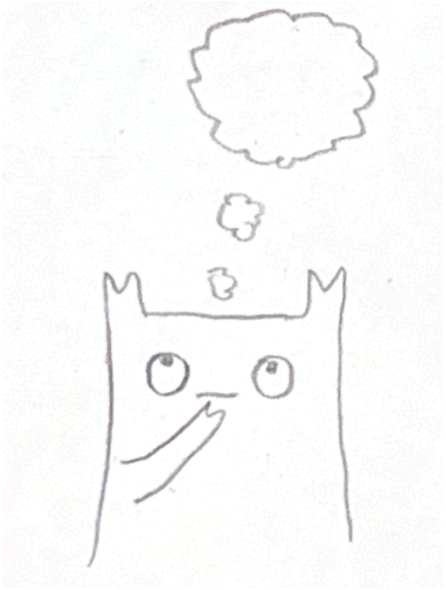
As your company grows 🌱, so does your app. Over time, multiple teams would work on a single application. Letting them **separate responsibilities** and allowing them to **own** their part of the application is crucial.

Each team can own a set of modules and some parts of the main application's code.

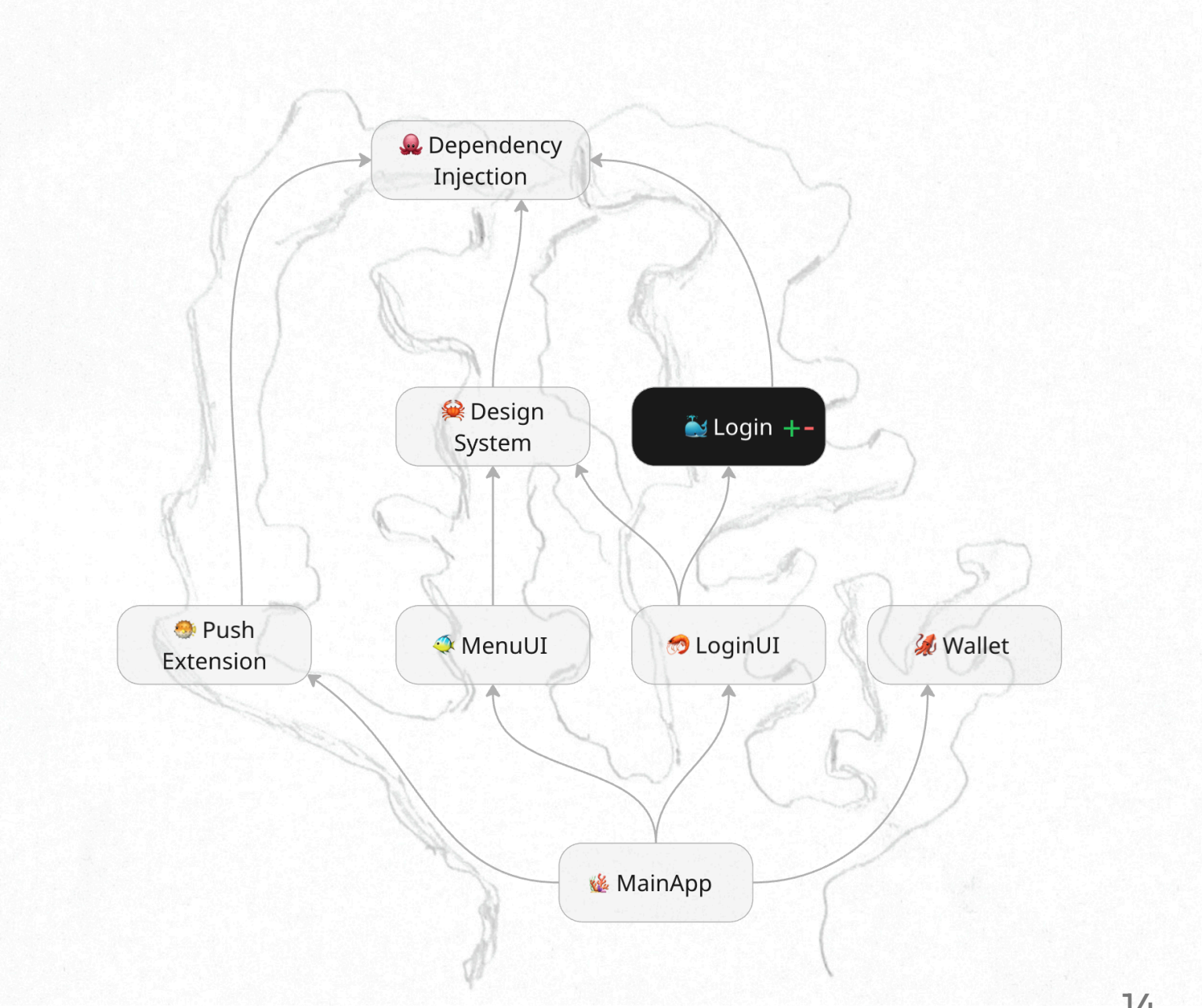
Modules

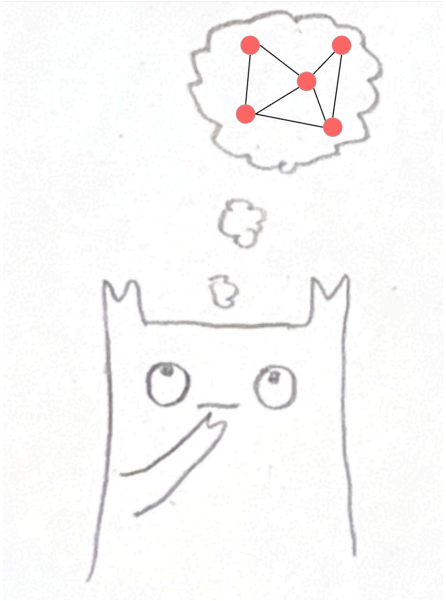

Dependency Injection
Design System
Login
Push Extension
MenuUI
LoginUI
WalletWeb
MainApp

Running all those tests is taking so much time!

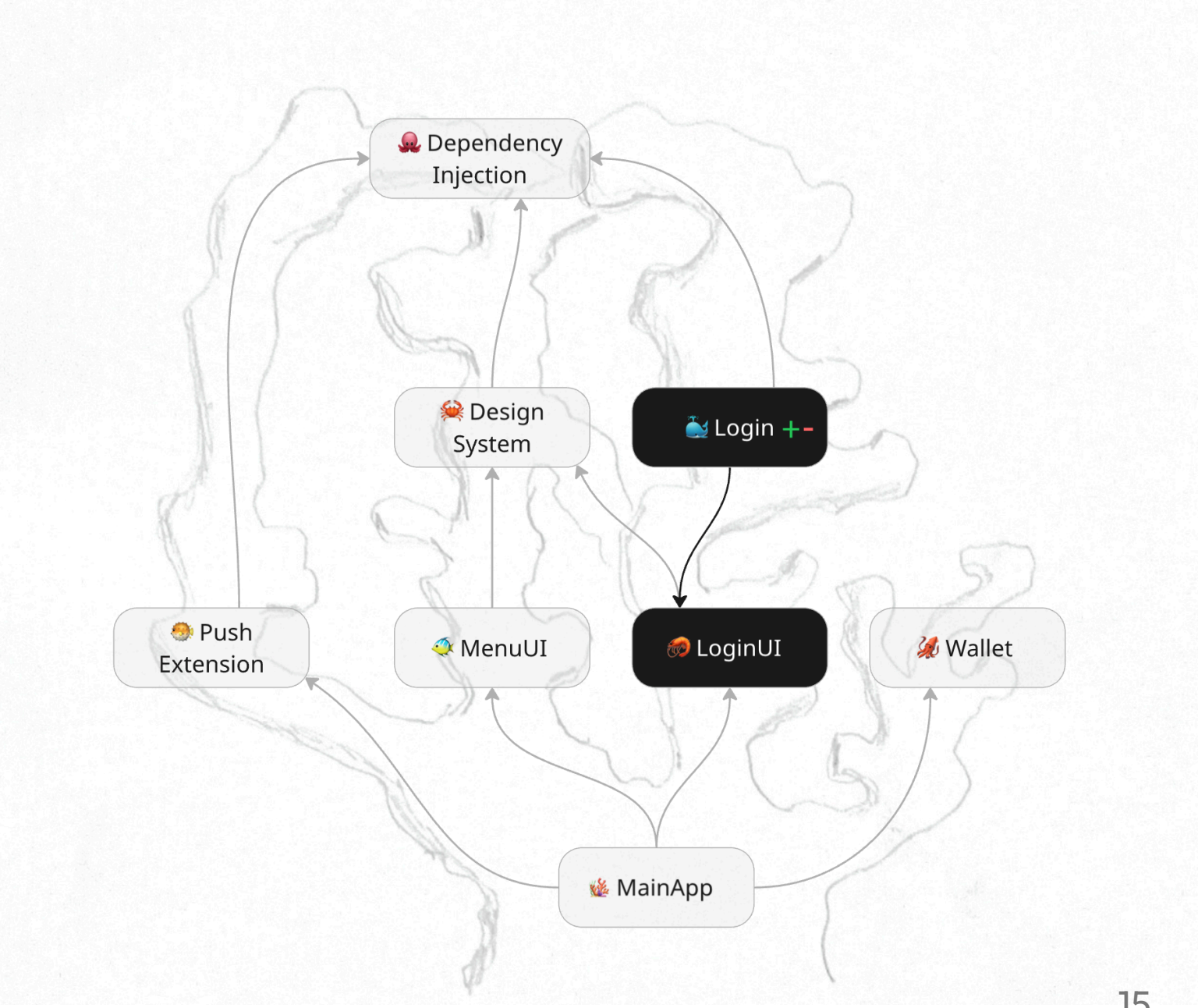


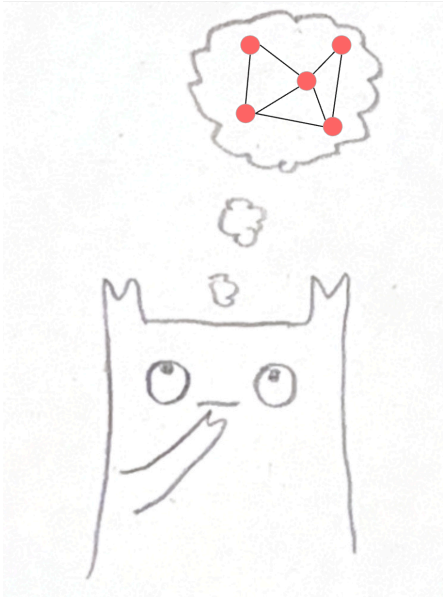
If the 📦 *Login* module is changed...



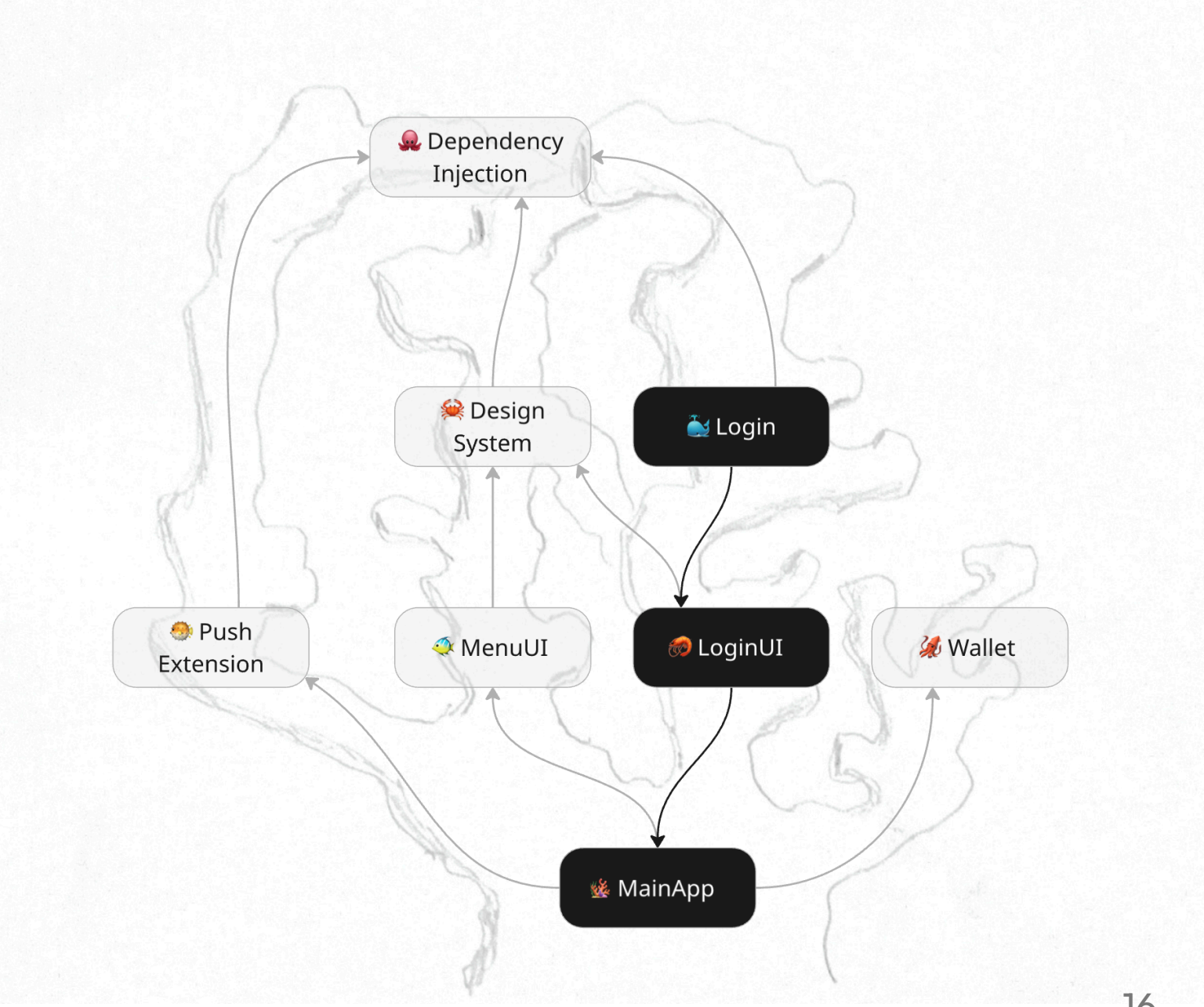


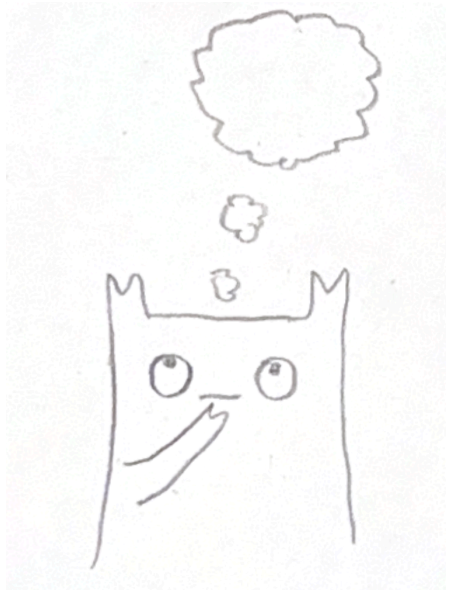
It would only affect the 📦 *LoginUI*...





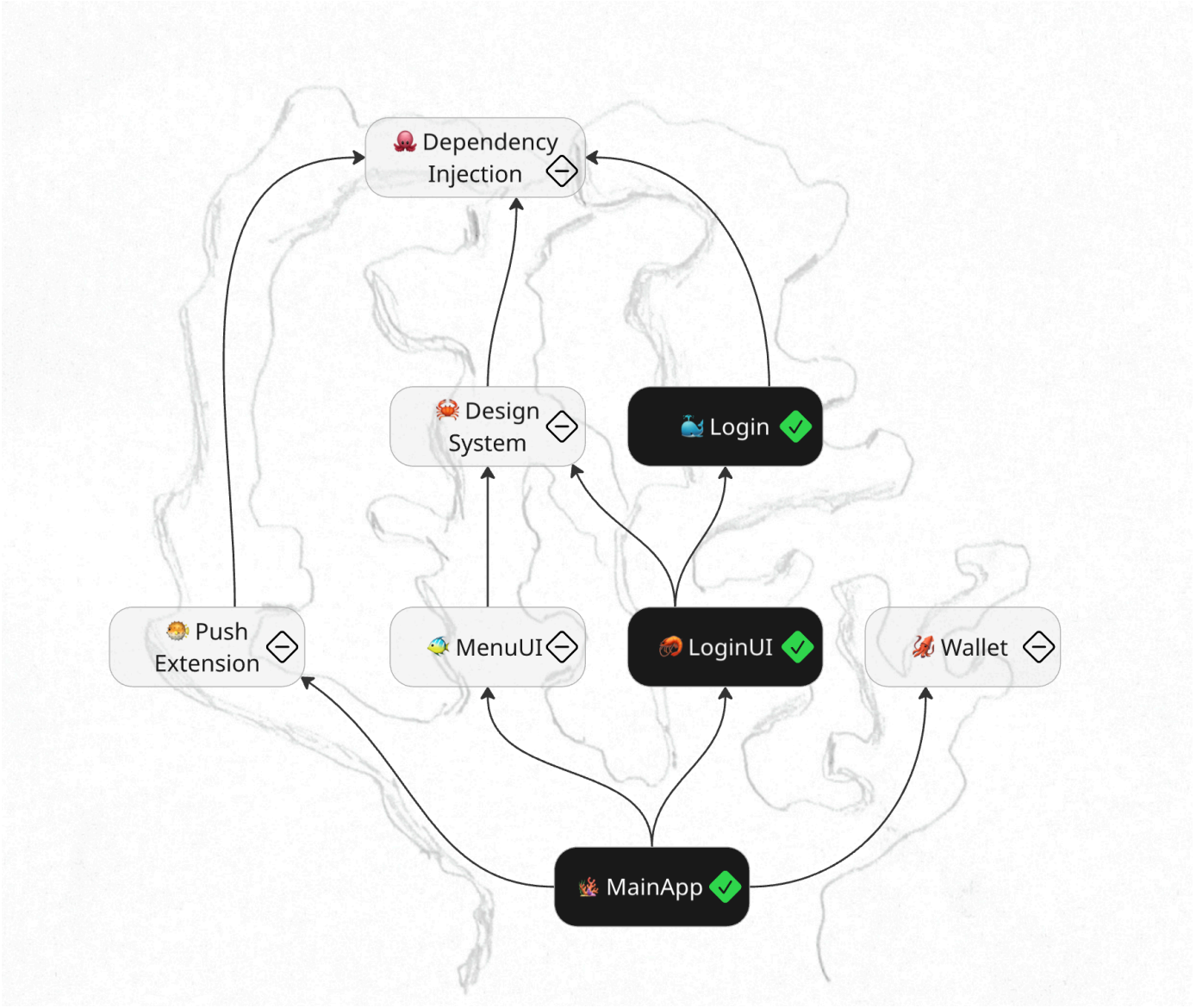
And the 📱 *MainApp*...





Does it make sense to test all the modules, if we know only the 📦 *Login* module is changed?

We can only run 50% of the tests and get the same results



But how can we know?

1. Detecting what is changed

Git allows us to find what files were touched in the changeset.

```
Root
├── Dependencies
│   └── Login
│       ├── Package.swift
│       ├── Sources
│       │   └── ❗LoginAssembly.swift
│       └── ...
├── MyProject.xcodeproj
└── Sources
```

2. Build the dependency graph, save the list of files for each dependency

Going from the project to its dependencies, to its dependencies, to dependencies of the dependencies, ...

This can be achieved with the *XcodeProj* package from Tuist.

Dependencies between packages can be parsed with `swift package dump-package`.

This is the most challenging part. We are dealing with obscure Xcode formats. But if we get that far, we will not be scared by that.

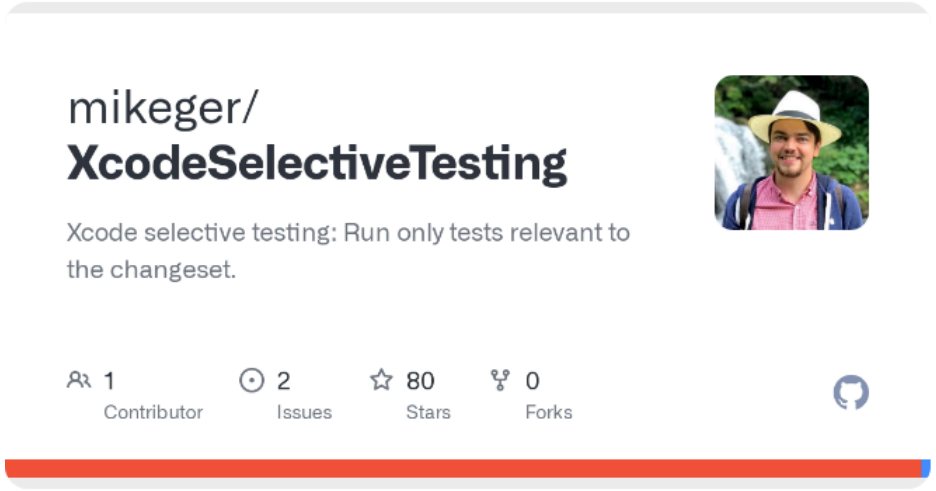
3. Traverse the graph, disable tests that can be skipped in the scheme/test plan

Go from every changed dependency all the way up, and save a set of dependencies you've touched.

Sounds like fun, Mike

But I am not going to implement it now.

Available Today!



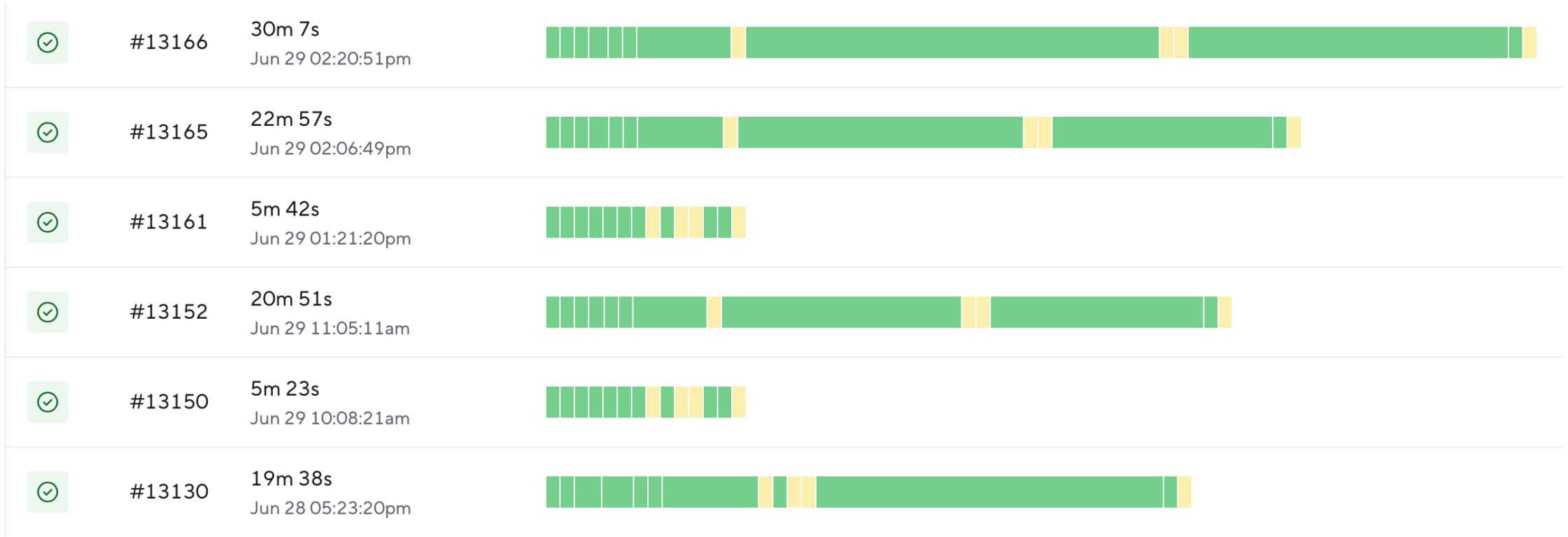
GitHub

GitHub - mikeger/XcodeSelectiveTesting: Xcode selective testing: Run only tests relevant to the chan…

Xcode selective testing: Run only tests relevant to the changeset. - GitHub - mikeger/XcodeSelectiveTesting: Xcode selective testing: Run only tests relevant to the changeset.

github.com/mikeger/XcodeSelectiveTesting

What does it bring?



Support

- Xcode Projects, also with buildable folders
- Swift Testing
- XCTest + UI Tests
- SPM packages

Support

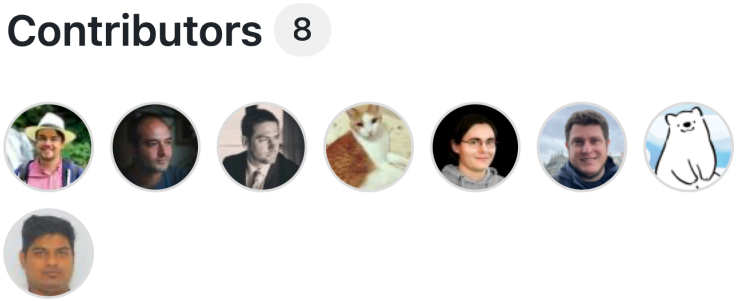
- Xcode Projects, also with buildable folders
- Swift Testing
- XCTest + UI Tests
- SPM packages
- Even Linux! Ask Marcin 

Pitfalls

- Code coverage reports must be uploaded per module. Otherwise, every run you would get different (bad) coverage.
- Flaky tests might be merged and stay in the repo for a while until someone touches them.

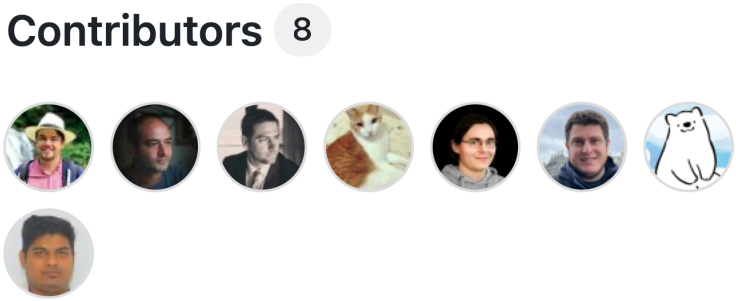
Open-source is awesome

- Make friends.
- Let your future employer already benefit from your code.
- Other people fix your bugs.



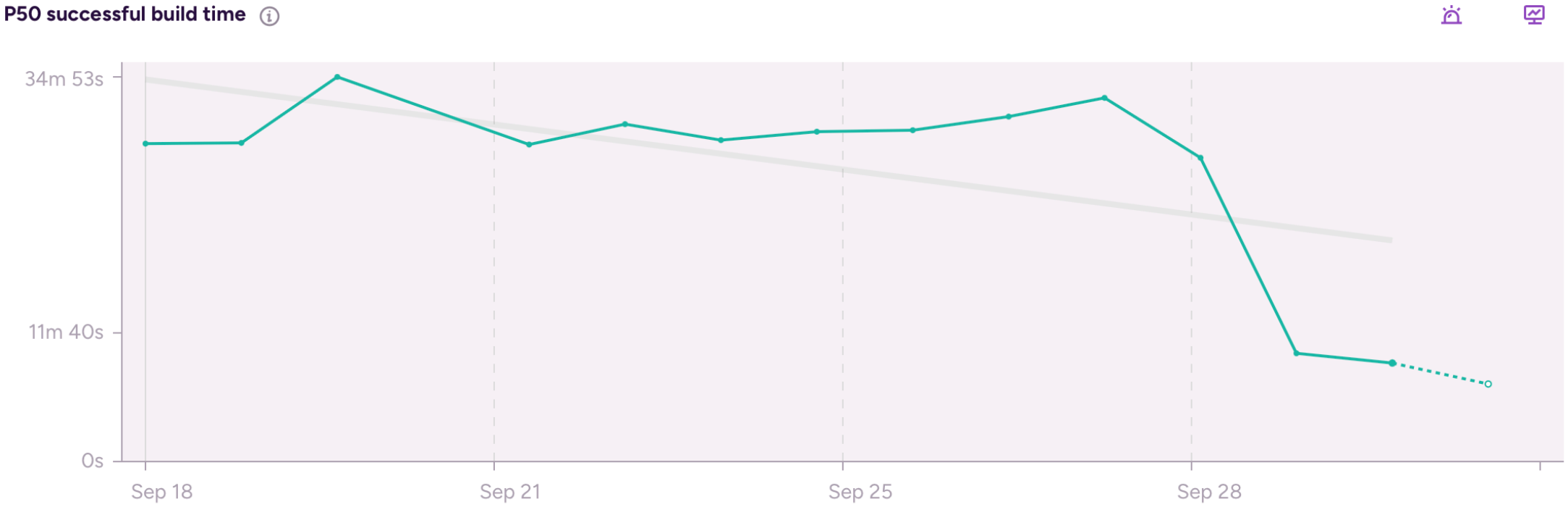
Open-source is awesome

- Make friends.
- Let your future employer already benefit from your code.
- Other people fix your bugs.
- Maybe even speak on stage!
- **I would like to thank all contributors!**



What about other solutions?

- **Tuist** also supports selective testing.
- Has a different principle: Target contents are hashed, and after the test is successful, it's saved on the Tuist.dev cloud.
- Downsides: it is paid.
- Upsides: fewer test runs for feature branches.


P50 successful build time
34m 53s
11m 40s
0s
Sep 18
Sep 21
Sep 25
Sep 28

Fin!

Or not?

Modularization allows Selective Testing.

What else?

Modularization

- Clear ownership
- Foster API design
- AI has a way to reason about your code's context — it's module

Thank you!

Illustrations by Alexander Gerasymenko