



Звук кофемашины, или Кэшируем зависимости SPM

Евгений Рыжов, Старший iOS разработчик

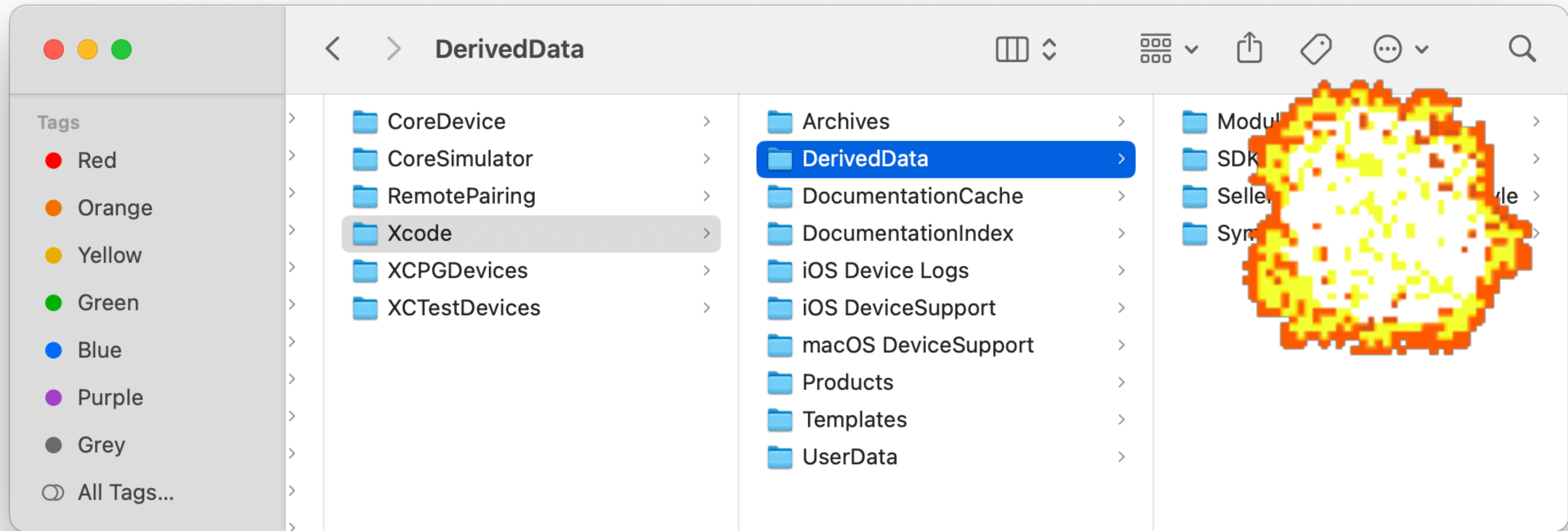
📍 @oiuhr



Shift + ⌘command + K

Знакомо?





Продолжительность, с.



Clean Build

A screenshot of the Xcode interface showing a build log on the left and a build timeline on the right. The build log is titled "SellerCenter (Development Flow) > Build > Prepare packages" and lists various tasks such as "Compile plug-ins and run any prebuild commands", "Run pre-actions", and "Prepare build". The build timeline on the right shows a sequence of tasks represented by colored bars, with a total duration of "5min 46s 444m". The status bar at the top indicates "Build Succeeded | Today at 03:40".

Продолжительность, с.

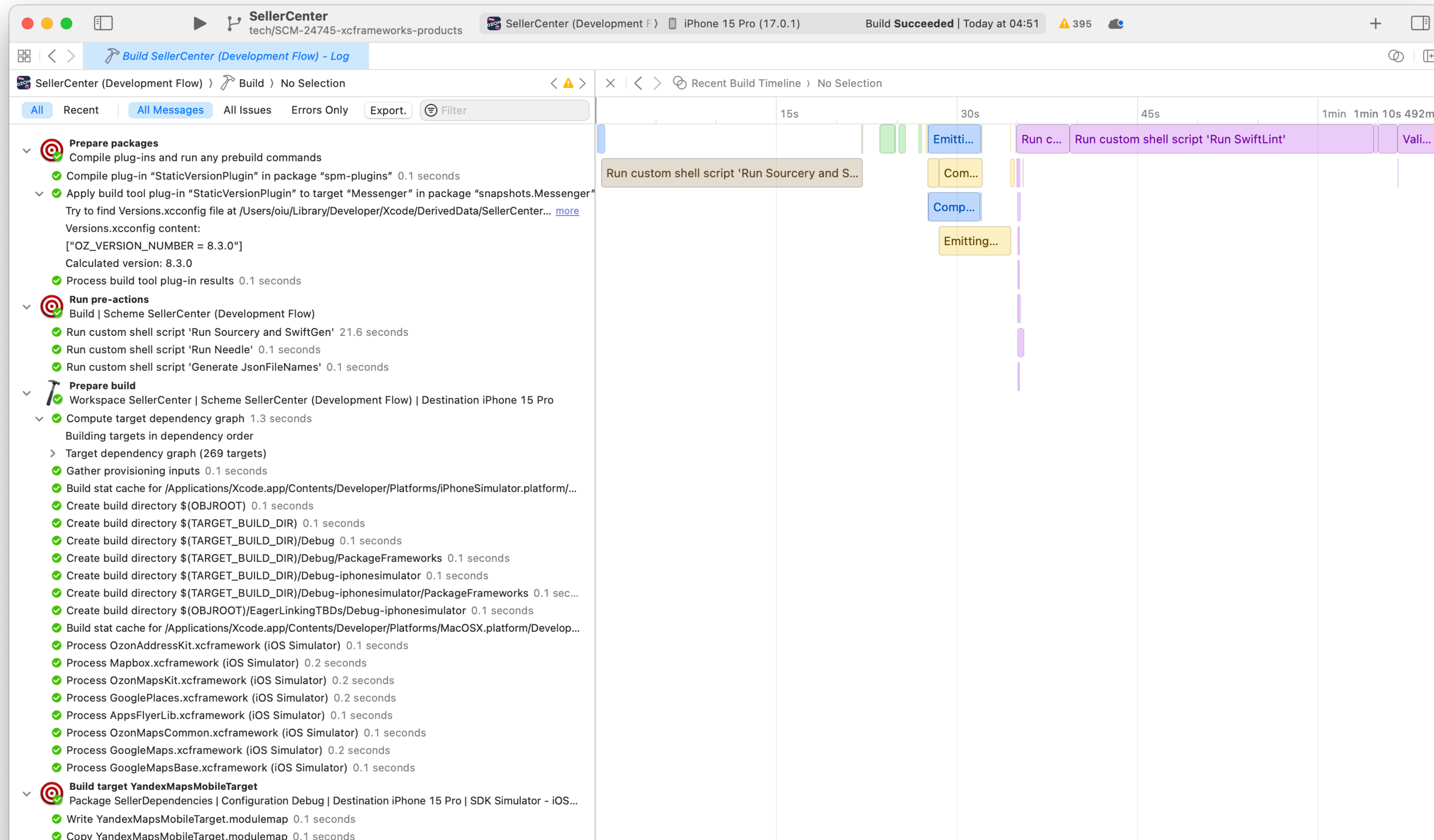


360



70

Clean Build
Iterative Build



SellerCenter (Development Flow) Build No Selection

Build Succeeded | Today at 04:51

Recent Build Timeline No Selection

15s 30s 45s 1min 1min 10s 492ms

- Prepare packages
 - Compile plug-ins and run any prebuild commands
 - Compile plug-in "StaticVersionPlugin" in package "spm-plugins" 0.1 seconds
 - Apply build tool plug-in "StaticVersionPlugin" to target "Messenger" in package "snapshots.Messenger" 0.1 seconds
 - Try to find Versions.xcconfig file at /Users/oiu/Library/Developer/Xcode/DerivedData/SellerCenter... more
 - Versions.xcconfig content:
["OZ_VERSION_NUMBER = 8.3.0"]
 - Calculated version: 8.3.0
 - Process build tool plug-in results 0.1 seconds
- Run pre-actions
 - Build | Scheme SellerCenter (Development Flow)
 - Run custom shell script 'Run Saurcery and SwiftGen' 21.6 seconds
 - Run custom shell script 'Run Needle' 0.1 seconds
 - Run custom shell script 'Generate JsonFileNames' 0.1 seconds
- Prepare build
 - Workspace SellerCenter | Scheme SellerCenter (Development Flow) | Destination iPhone 15 Pro
 - Compute target dependency graph 1.3 seconds
 - Building targets in dependency order
 - Target dependency graph (269 targets)
 - Gather provisioning inputs 0.1 seconds
 - Build stat cache for /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/... 0.1 seconds
 - Create build directory \$(OBJROOT) 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR) 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR)/Debug 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR)/Debug/PackageFrameworks 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR)/Debug-iphonesimulator 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR)/Debug-iphonesimulator/PackageFrameworks 0.1 sec...
 - Create build directory \$(OBJROOT)/EagerLinkingTBDS/Debug-iphonesimulator 0.1 seconds
 - Build stat cache for /Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Develop... 0.1 seconds
 - Process OzonAddressKit.xcframework (iOS Simulator) 0.1 seconds
 - Process Mapbox.xcframework (iOS Simulator) 0.2 seconds
 - Process OzonMapsKit.xcframework (iOS Simulator) 0.2 seconds
 - Process GooglePlaces.xcframework (iOS Simulator) 0.2 seconds
 - Process AppsFlyerLib.xcframework (iOS Simulator) 0.1 seconds
 - Process OzonMapsCommon.xcframework (iOS Simulator) 0.1 seconds
 - Process GoogleMaps.xcframework (iOS Simulator) 0.2 seconds
 - Process GoogleMapsBase.xcframework (iOS Simulator) 0.1 seconds
- Build target YandexMapsMobileTarget
 - Package SellerDependencies | Configuration Debug | Destination iPhone 15 Pro | SDK Simulator - iOS...
 - Write YandexMapsMobileTarget.modulemap 0.1 seconds
 - Copy YandexMapsMobileTarget.modulemap 0.1 seconds

Продолжительность, с.

360

70

690

- Clean Build
- Iterative Build
- Resolve + Clean Build

```
seller-app-ios — zsh — zsh (figterm) ▸ zsh — 68x14
...r-app-ios — zsh — zsh (figterm) ▸ zsh
...rnal — zsh — zsh (figterm) ▸ zsh
Last login: Sat Feb 24 23:21:44 on console
[oiu@mbp-evgeryzhov-BYOD-WHH5P7YX2T seller-app-ios % make go
Заккрытие Xcode
No matching processes belonging to you were found
Установка git hooks
Простановка DI связей
Генерация ресурсов
Генерация проекта
Resolved cache profile 'Development' from Tuist's defaults
Generating workspace SellerCenter.xcworkspace
Generating project SellerCenter
Resolving package dependencies using xcodebuild
Project generated.
Total time taken: 334.816s
```

The screenshot shows the Xcode interface for a build of the SellerCenter project. The top bar indicates the build succeeded. The main area displays a detailed log of the build process, including steps like 'Prepare packages', 'Run pre-actions', and 'Prepare build'. The 'Prepare build' section shows the workspace configuration and the target dependency graph. The right side of the screen shows a 'Recent Build Timeline' with a 15-second duration and various build phases represented by colored bars.

Build Log:

- Prepare packages
 - Compile plug-ins and run any prebuild commands
 - Compile plug-in "StaticVersionPlugin" in package "spm-plugins" 0.1 seconds
 - Apply build tool plug-in "StaticVersionPlugin" to target "Messenger" in package "snapshots.Messenger"
 - Try to find Versions.xcconfig file at /Users/oiu/Library/Developer/Xcode/DerivedData/SellerCenter... more
 - Versions.xcconfig content:
["OZ_VERSION_NUMBER = 8.3.0"]
 - Calculated version: 8.3.0
 - Process build tool plug-in results 0.1 seconds
- Run pre-actions
 - Build | Scheme SellerCenter (Development Flow)
 - Run custom shell script 'Run Sourcery and SwiftGen' 21.6 seconds
 - Run custom shell script 'Run Needle' 0.1 seconds
 - Run custom shell script 'Generate JsonFileNames' 0.1 seconds
- Prepare build
 - Workspace SellerCenter | Scheme SellerCenter (Development Flow) | Destination iPhone 15 Pro
 - Compute target dependency graph 1.3 seconds
 - Building targets in dependency order
 - Target dependency graph (269 targets)
 - Gather provisioning inputs 0.1 seconds
 - Build stat cache for /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/... 0.1 seconds
 - Create build directory \$(OBJROOT) 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR) 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR)/Debug 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR)/Debug/PackageFrameworks 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR)/Debug-iphonesimulator 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR)/Debug-iphonesimulator/PackageFrameworks 0.1 sec...
 - Create build directory \$(OBJROOT)/EagerLinkingTBDs/Debug-iphonesimulator 0.1 seconds
 - Build stat cache for /Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Develop... 0.1 seconds
 - Process OzonAddressKit.xcframework (iOS Simulator) 0.1 seconds
 - Process Mapbox.xcframework (iOS Simulator) 0.2 seconds
 - Process OzonMapsKit.xcframework (iOS Simulator) 0.2 seconds
 - Process GooglePlaces.xcframework (iOS Simulator) 0.2 seconds
 - Process AppsFlyerLib.xcframework (iOS Simulator) 0.1 seconds
 - Process OzonMapsCommon.xcframework (iOS Simulator) 0.1 seconds

Продолжительность, с.

360

70

690

???

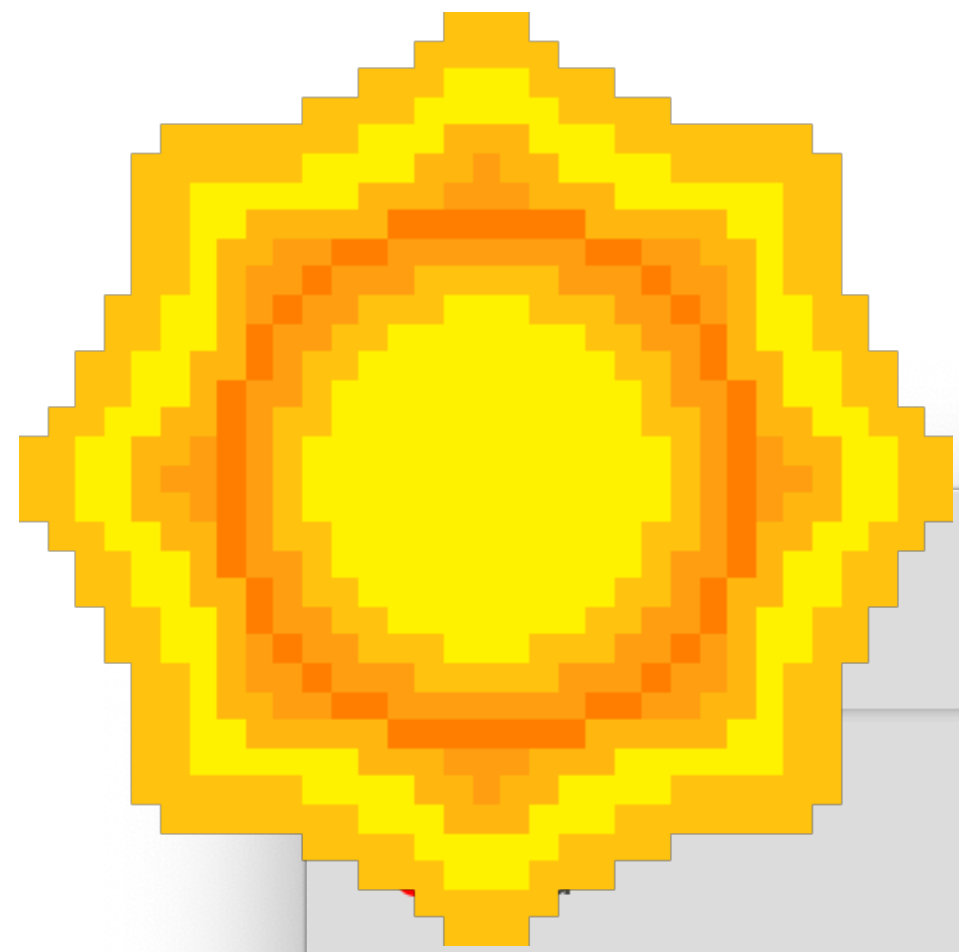
- Clean Build
- Iterative Build
- Resolve + Clean Build
- Cached Clean Build

```
seller-app-ios — zsh — zsh (figterm) ▶ zsh — 68x14
...r-app-ios — zsh — zsh (figterm) ▶ zsh
...rnal — zsh — zsh (figterm) ▶ zsh
Last login: Sat Feb 24 23:21:44 on console
[oiu@mbp-evgeryzhov-BYOD-WHH5P7YX2T seller-app-ios % make go
Заккрытие Xcode
No matching processes belonging to you were found
Установка git hooks
Простановка DI связей
Генерация ресурсов
Генерация проекта
Resolved cache profile 'Development' from Twist's defaults
Generating workspace SellerCenter.xcworkspace
Generating project SellerCenter
Resolving package dependencies using xcodebuild
Project generated.
Total time taken: 334.816s
```

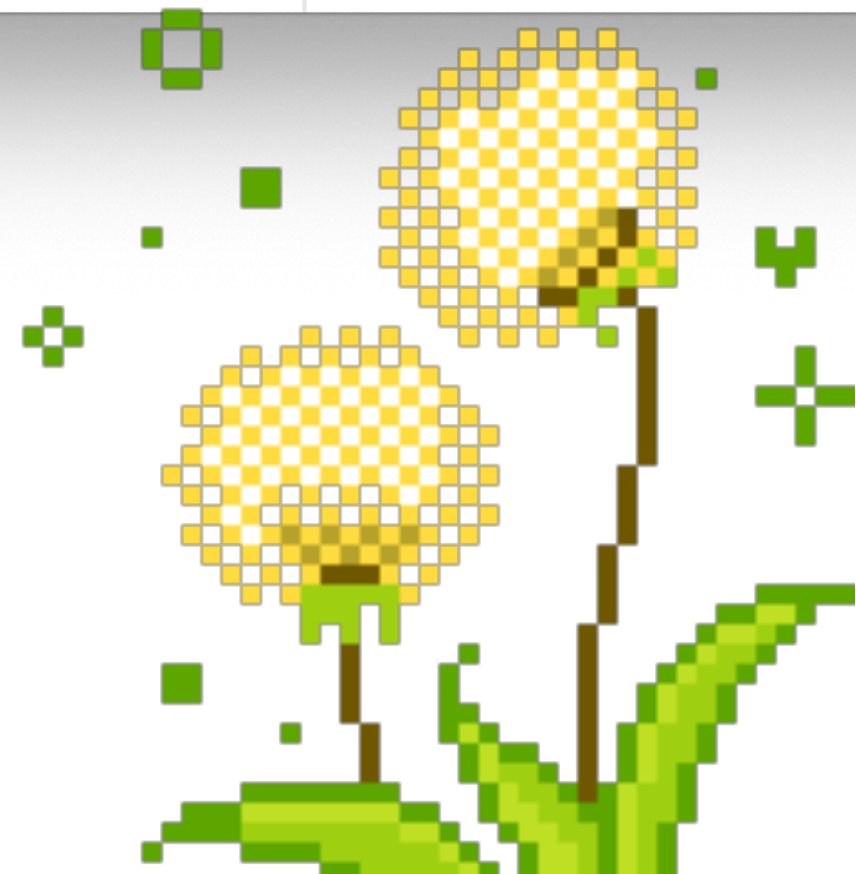
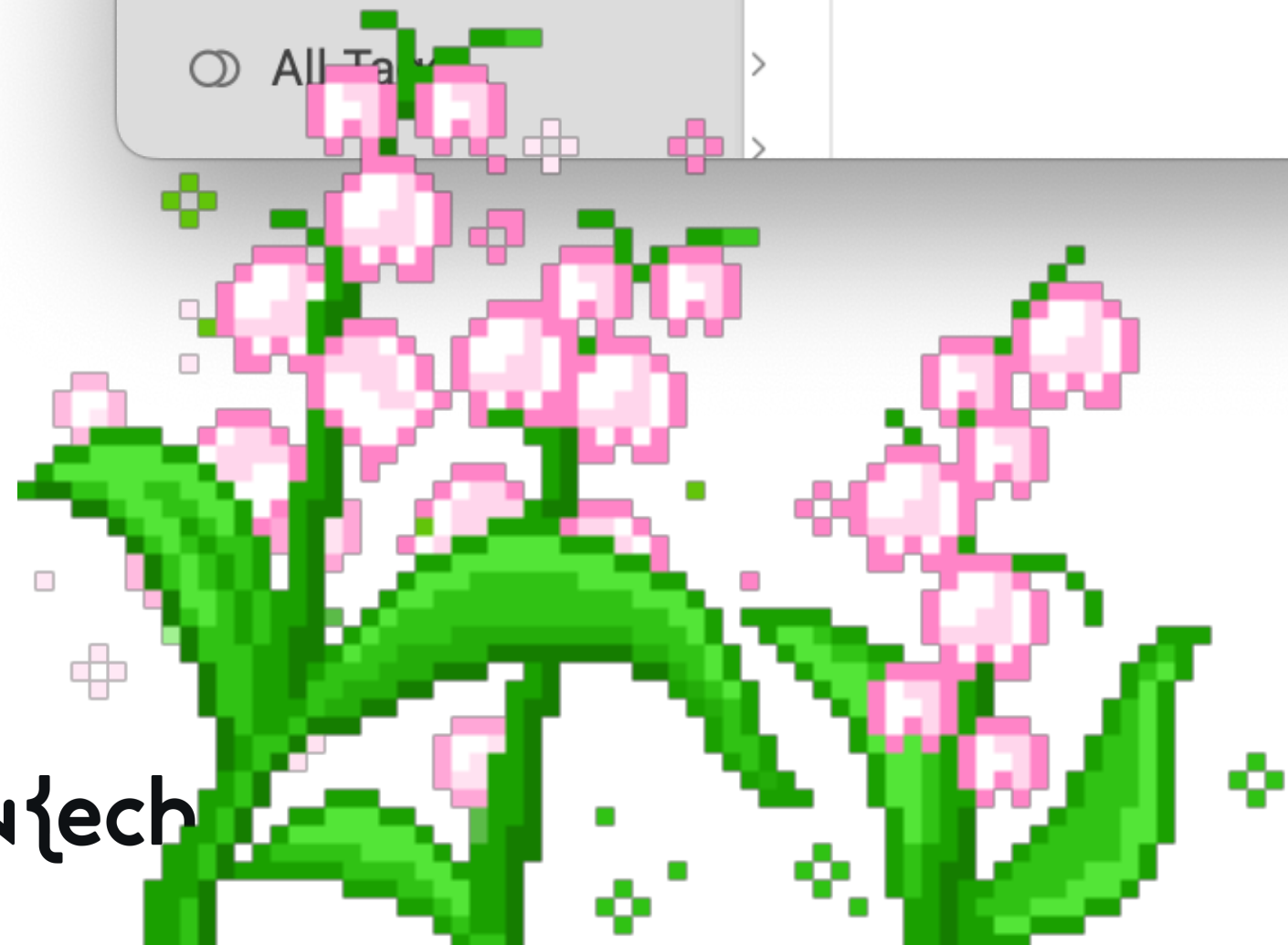
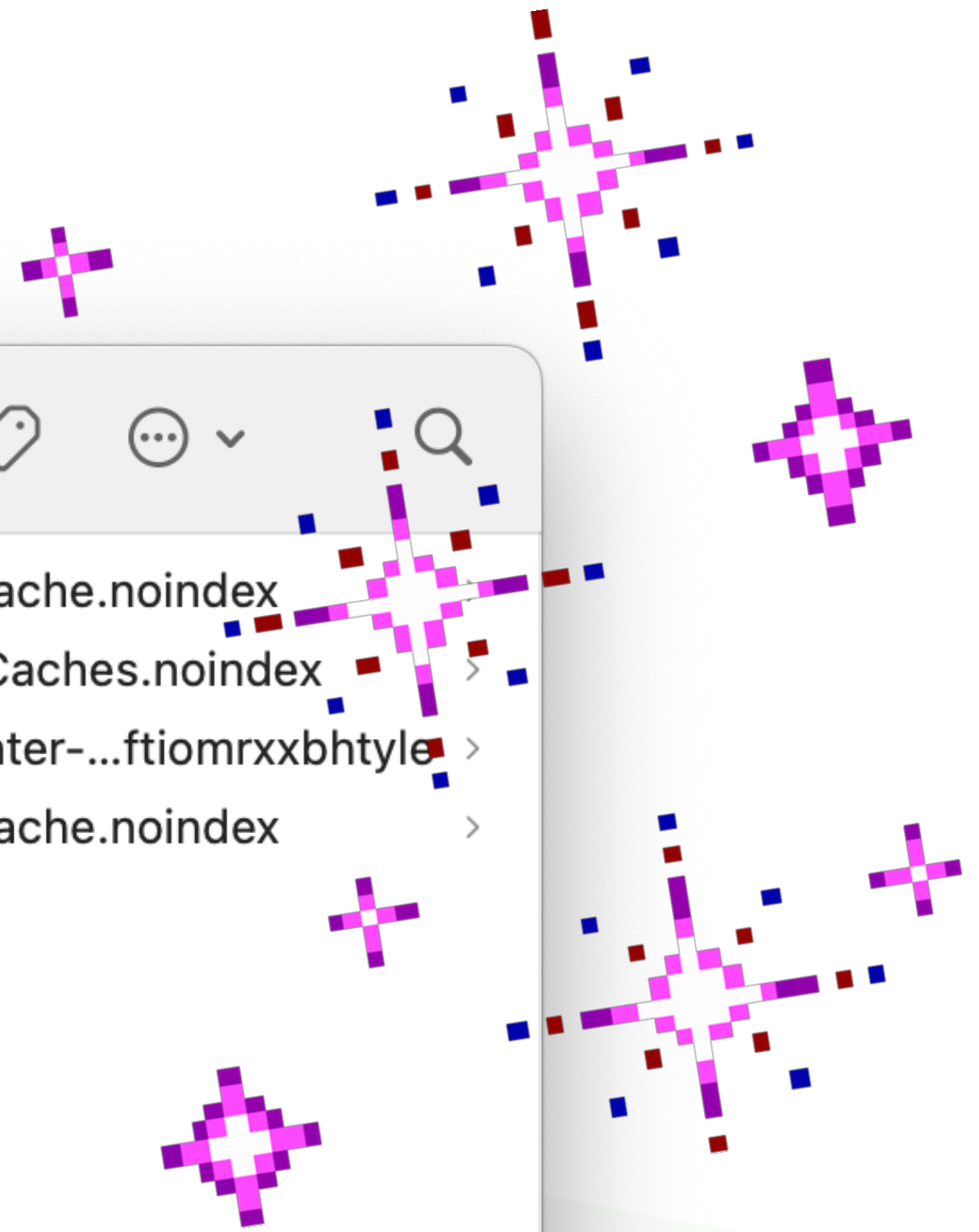
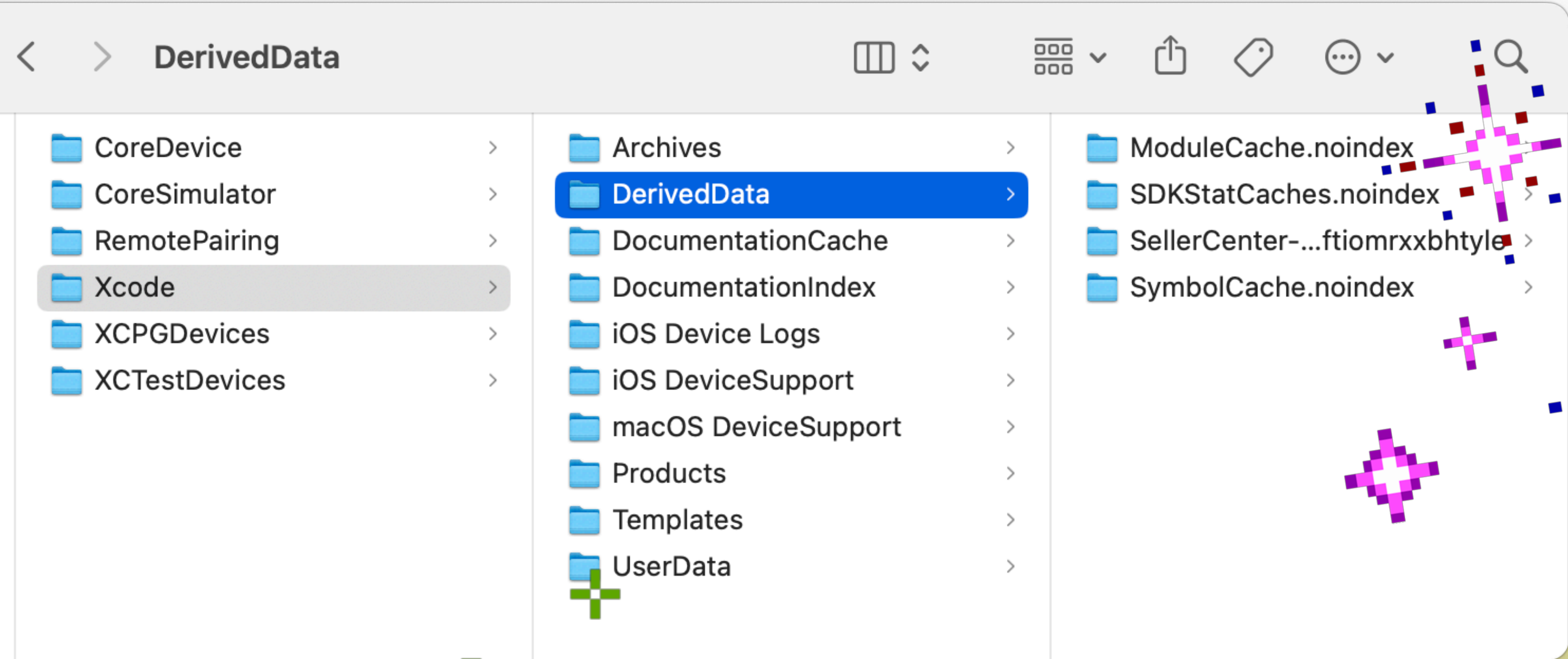
SellerCenter (Development Flow) - Log

SellerCenter (Development Flow) Build No Selection

- Prepare packages
 - Compile plug-ins and run any prebuild commands
 - Compile plug-in "StaticVersionPlugin" in package "spm-plugins" 0.1 seconds
 - Apply build tool plug-in "StaticVersionPlugin" to target "Messenger" in package "snapshots.Messenger"
 - Try to find Versions.xcconfig file at /Users/oiu/Library/Developer/Xcode/DerivedData/SellerCenter... more
 - Versions.xcconfig content:
["OZ_VERSION_NUMBER = 8.3.0"]
 - Calculated version: 8.3.0
 - Process build tool plug-in results 0.1 seconds
- Run pre-actions
 - Build | Scheme SellerCenter (Development Flow)
 - Run custom shell script 'Run Sourcery and SwiftGen' 21.6 seconds
 - Run custom shell script 'Run Needle' 0.1 seconds
 - Run custom shell script 'Generate JsonFileNames' 0.1 seconds
- Prepare build
 - Workspace SellerCenter | Scheme SellerCenter (Development Flow) | Destination iPhone 15 Pro
 - Compute target dependency graph 1.3 seconds
 - Building targets in dependency order
 - Target dependency graph (269 targets)
 - Gather provisioning inputs 0.1 seconds
 - Build stat cache for /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/... 0.1 seconds
 - Create build directory \$(OBJROOT) 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR) 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR)/Debug 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR)/Debug/PackageFrameworks 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR)/Debug-iphonesimulator 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR)/Debug-iphonesimulator/PackageFrameworks 0.1 sec...
 - Create build directory \$(OBJROOT)/EagerLinkingTBDS/Debug-iphonesimulator 0.1 seconds
 - Build stat cache for /Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Develop... 0.1 seconds
 - Process OzonAddressKit.xcframework (iOS Simulator) 0.1 seconds
 - Process Mapbox.xcframework (iOS Simulator) 0.2 seconds
 - Process OzonMapsKit.xcframework (iOS Simulator) 0.2 seconds
 - Process GooglePlaces.xcframework (iOS Simulator) 0.2 seconds
 - Process AppsFlyerLib.xcframework (iOS Simulator) 0.1 seconds
 - Process OzonMapsCommon.xcframework (iOS Simulator) 0.1 seconds



- Orange
- Yellow
- Green
- Blue
- Purple
- Grey
- All Tags



О чём разговор



План действий

А может, лучше кофе? ☕

01 Немного о проекте →

02 Base Benchmarking →

03 Build Timeline →

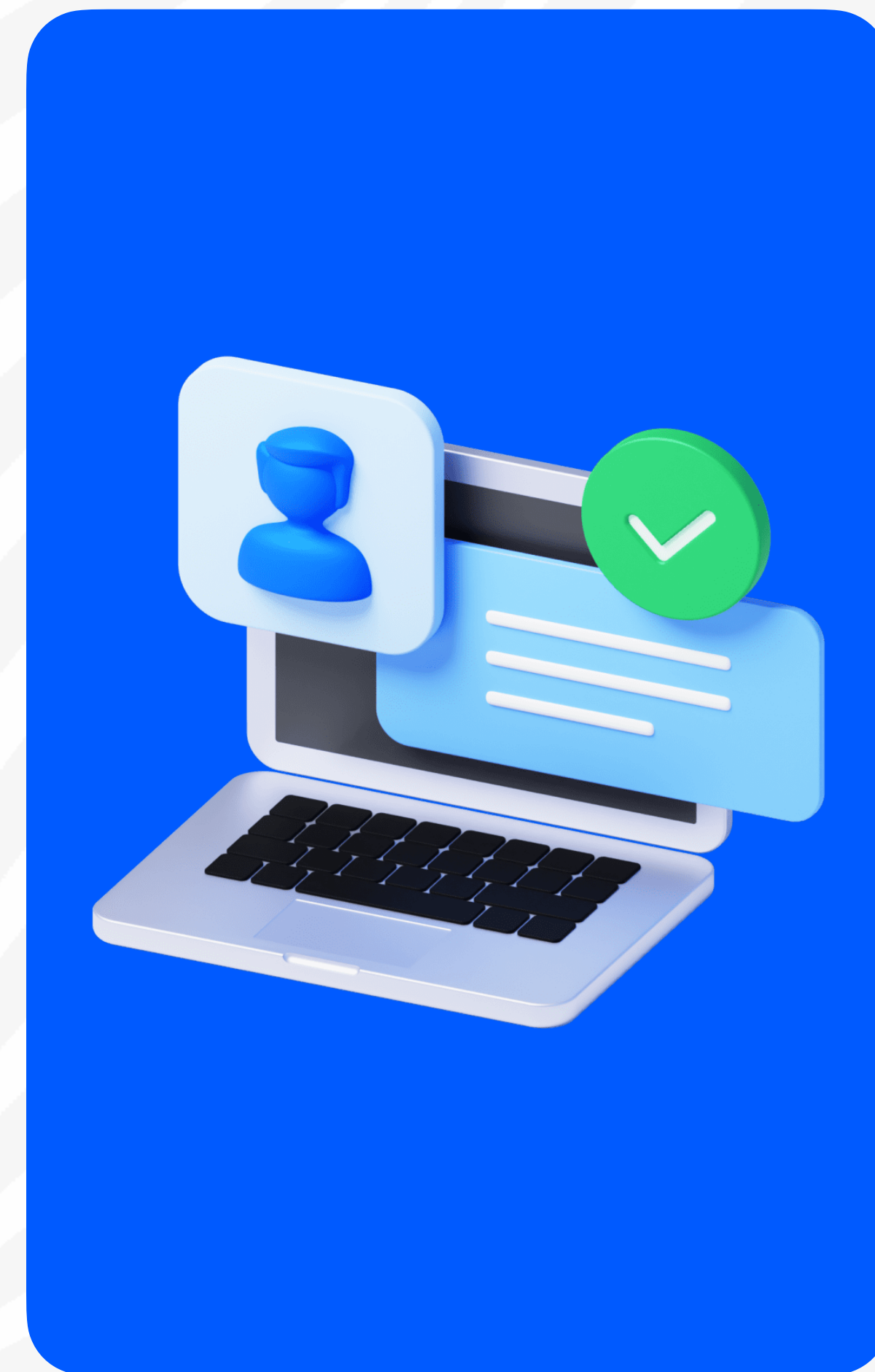
04 Инструменты Build-Cache зависимостей →

05 Накладываем схему на SPM →

06 Финальные бенчмарки →

07 Но какой ценой? →

08 Завершение →



01



Немного
о проекте

Ода людям

Кто мы?



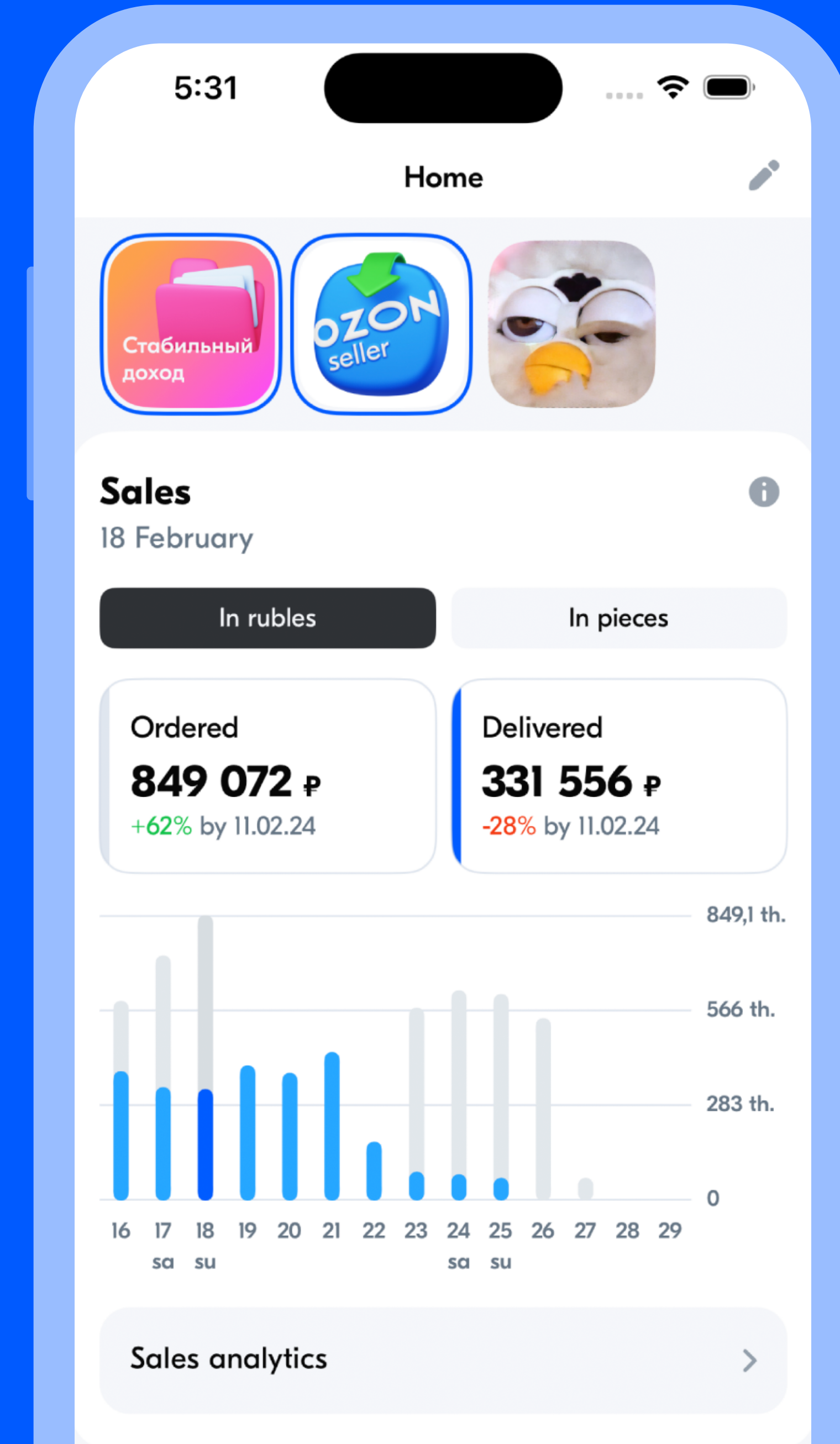
Развиваем приложение
продавца



Перевалили за **1 000 000**
строк кода



В команде **~30** красивых умных
прекрасных iOS-разработчиков,
распределенных между 5
командами, и тестировщики,
пишущие нативные UI-тесты



Ода **техничке**

С чем мы работаем?

- ➔ **iOS 15+, SwiftUI 3, Async/Await**, вставьте своё смузи
- ➔ **SwiftPM** как менеджер зависимостей и как решение для многомодульности проекта
- ➔ **Tuist** для генерации проекта

ozon{tech



Swift

iOS 15+

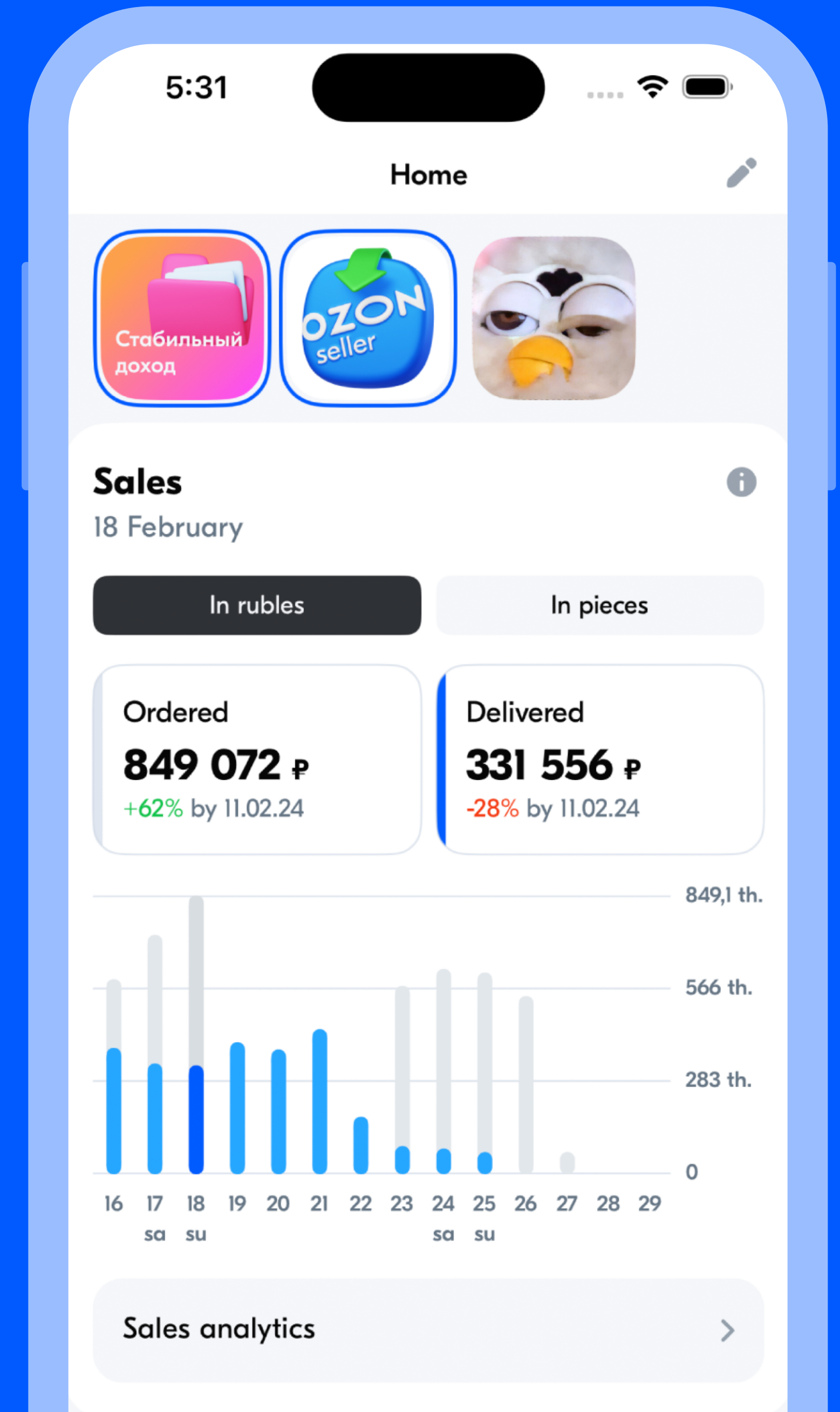
SwiftUI 3



Gitlab

Tuist

SwiftPM



SwiftPM

Менеджер зависимостей



SwiftPM

Историческое решение
для организации
многомодульности

Рулит зависимостями проекта

Нативен!

Проинтегрировали SPM Registry

Tuist

Генератор генерируемого



Tuist

Генерация +
сетап .xcodproj и
остального

Пришел на смену XCodeGen

Убирает возню с мердж-
конфликтами

Облегчает сетапы
таргетов и схем

02



Base
Benchmarking



Base Benchmarking

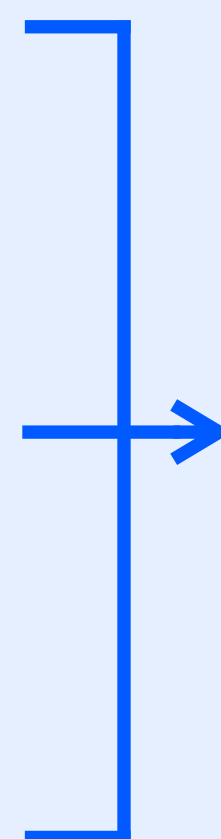
Формула билдов

Merge Request

(стандартный)

Пробегут 3 пайплайна

1. Изначальный коммит 
2. Коммит с фиксом тестов 
3. Merge Train MR`a



По 3 билда

1. Тесты (Unit)
2. Сборка для QA
3. Ещё тесты (UI)



Base Benchmarking

Формула билдов

$$3 \times 3 - 2 = 7$$

Упавший пайп

×

12 мин. 30 сек.

Стандартная сборка

...

87 мин. 30 сек. × 30 человек

Base Benchmarking

Ужасы бытия

≈ 44 часа



Base Benchmarking

Формула билдов

12 мин. **30** сек.

Стандартная сборка

87 мин. **30** сек.

Машинное время билдов MR-a

Base Benchmarking

Формула билдов



12 мин. **30** сек.

Стандартная сборка

87 мин. **30** сек.

Машинное время билдов MP-a



03



Build
Timeline

Build Timeline

Кто, зачем и сколько?

The screenshot displays the Xcode interface for a build of SellerCenter (Development Flow) on an iPhone 15 Pro (17.0.1). The build succeeded at 03:40. The interface is divided into two main sections: a log on the left and a timeline on the right.

Build Log (Left Panel):

- Prepare packages** (1.5 seconds)
 - Compile plug-in "StaticVersionPlugin" in package "spm-plugins"
 - Apply build tool plug-in "StaticVersionPlugin" to target "Messenger" in package "snapshots.Messenger". Try to find Versions.xcconfig file at /Users/oiu/Library/Developer/Xcode/DerivedData/SellerCenter... Versions.xcconfig content: ["OZ_VERSION_NUMBER = 8.3.0"] Calculated version: 8.3.0
 - Process build tool plug-in results (0.1 seconds)
- Run pre-actions**
 - Build | Scheme SellerCenter (Development Flow)
 - Run custom shell script 'Run Sourcery and SwiftGen' (23.7 seconds)
 - Run custom shell script 'Run Needle' (0.1 seconds)
 - Run custom shell script 'Generate JsonFileNames' (0.1 seconds)
- Prepare build**
 - Workspace SellerCenter | Scheme SellerCenter (Development Flow) | Destination iPhone 15 Pro
 - Compute target dependency graph (1.3 seconds)
 - Building targets in dependency order
 - Target dependency graph (269 targets)
 - Gather provisioning inputs (0.1 seconds)
 - Create build description (14.3 seconds)
 - Build stat cache for /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/D...
 - Build stat cache for /Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Develop...
 - Create build directory \$(OBJROOT) (0.1 seconds)
 - Create build directory \$(TARGET_BUILD_DIR) (0.1 seconds)
 - Create build directory \$(TARGET_BUILD_DIR)/Debug (0.1 seconds)
 - Create build directory \$(TARGET_BUILD_DIR)/Debug-iphonesimulator (0.1 seconds)
 - Create build directory \$(OBJROOT)/EagerLinkingTBDS/Debug-iphonesimulator (0.1 seconds)
 - Create build directory \$(OBJROOT)/EagerLinkingTBDS/Debug (0.1 seconds)
 - Create build directory \$(TARGET_BUILD_DIR)/Debug/PackageFrameworks (0.1 seconds)
 - Create build directory \$(TARGET_BUILD_DIR)/Debug-iphonesimulator/PackageFrameworks (0.1 seconds)
 - Process Cronet.xcframework (iOS Simulator) (0.1 seconds)
 - Process YandexMapsMobile.xcframework (iOS Simulator) (0.2 seconds)
 - Process OzonMapsKit.xcframework (iOS Simulator) (0.2 seconds)
 - Process OzonMapsCommon.xcframework (iOS Simulator) (0.1 seconds)
 - Process OzonAddressKit.xcframework (iOS Simulator) (0.1 seconds)
 - Process Mapbox.xcframework (iOS Simulator) (0.2 seconds)
 - Process GooglePlaces.xcframework (iOS Simulator) (0.2 seconds)
 - Process GoogleMapsCore.xcframework (iOS Simulator) (0.1 seconds)
 - Process GoogleMapsBase.xcframework (iOS Simulator) (0.1 seconds)
 - Process AppsFlyerLib.xcframework (iOS Simulator) (0.1 seconds)
 - Process GoogleMaps.xcframework (iOS Simulator) (0.2 seconds)
- Build target ozon_id_sdk_OzonId**
 - Package ozon_id_sdk | Configuration Debug | Destination iPhone 15 Pro | SDK Simulator - iOS 17.2
 - Create directory ozon_id_sdk OzonId.bundle (0.1 seconds)

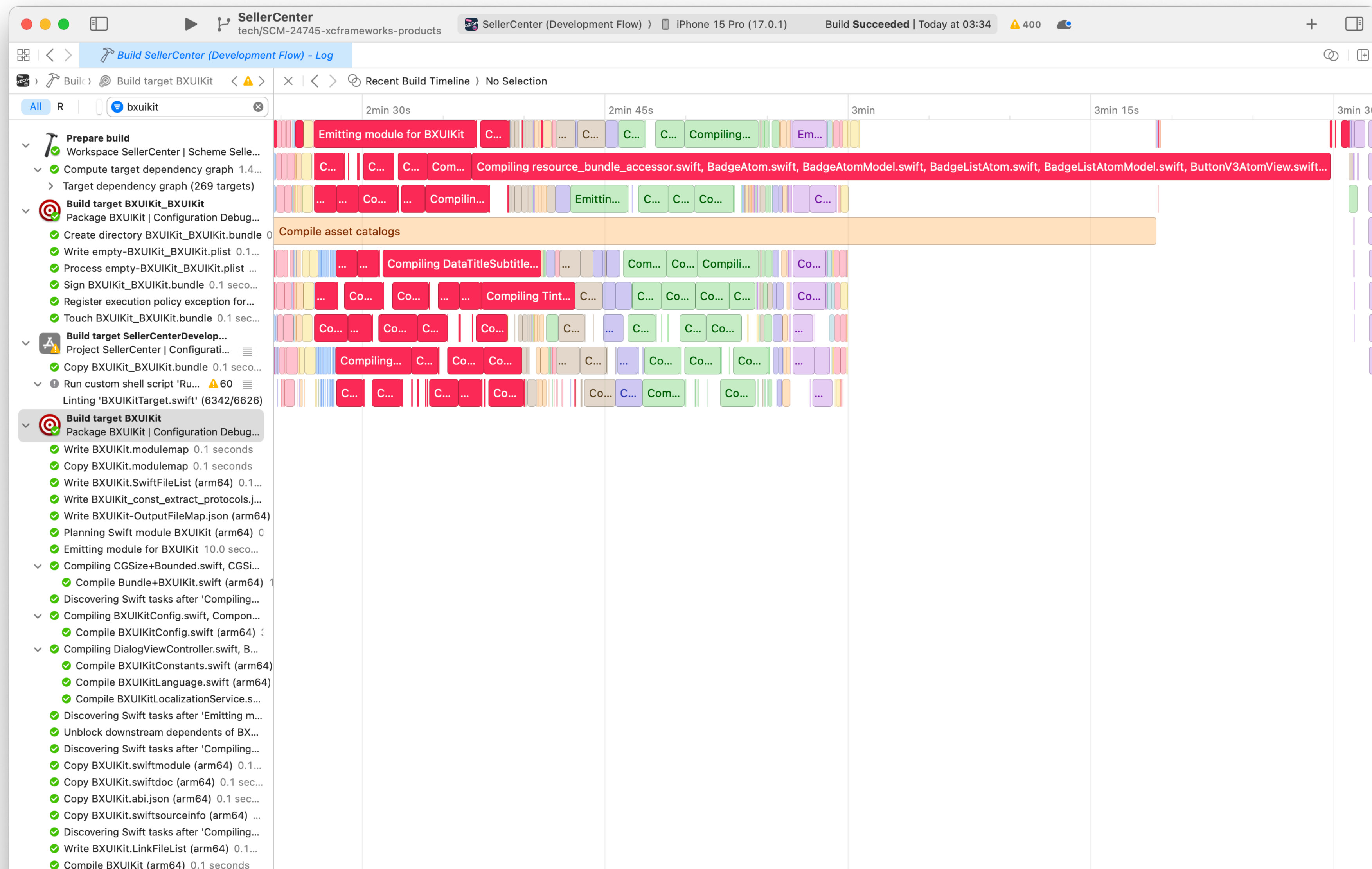
Build Timeline (Right Panel):

The timeline shows a total duration of 5min 46s 444ms. Key tasks include:

- Run cu...
- Compile asset catalogs
- Compiling resource_b...
- Emit...
- Compi...
- Compil...
- C...

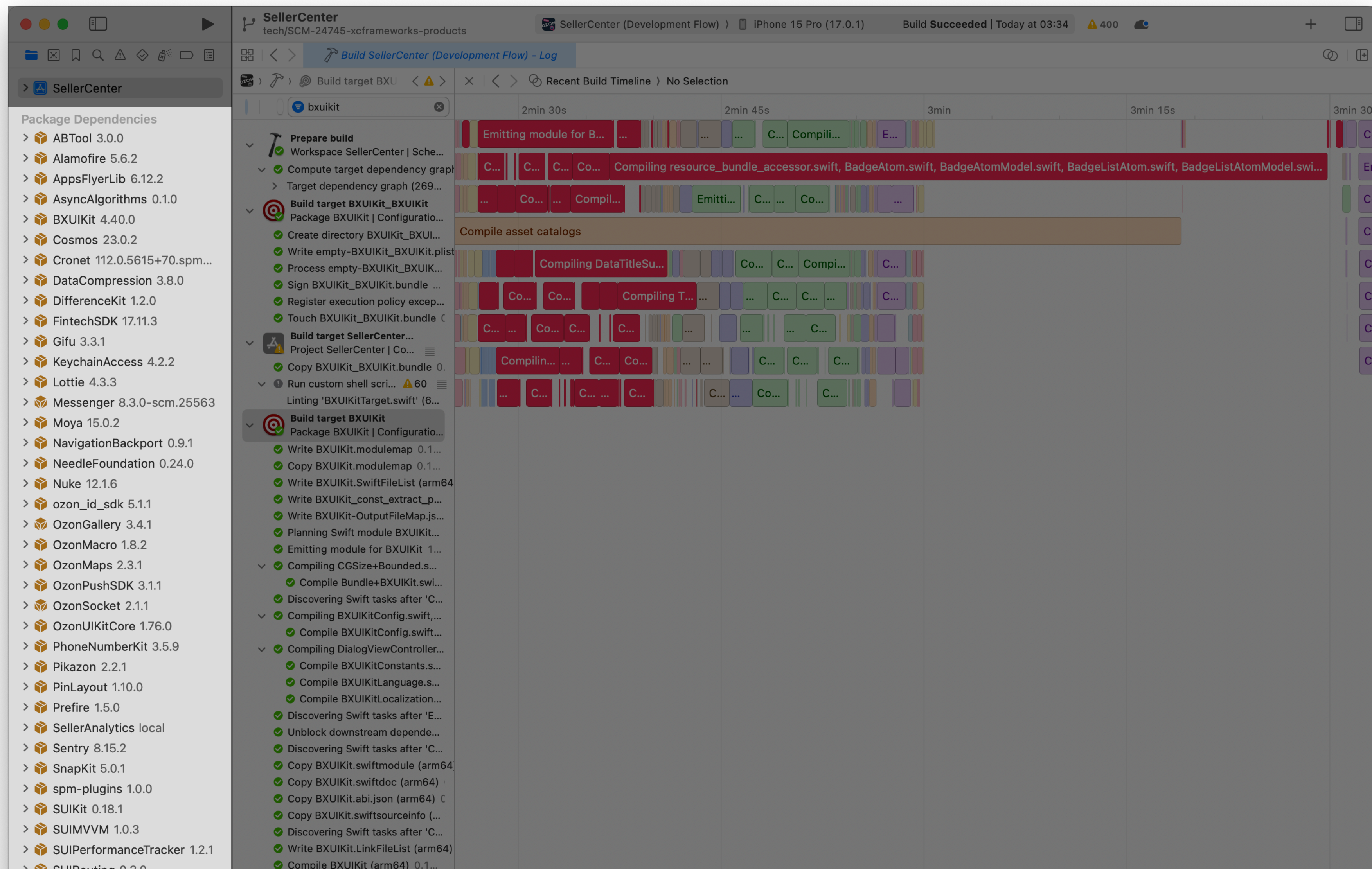
Build Timeline

Кто, зачем и сколько?



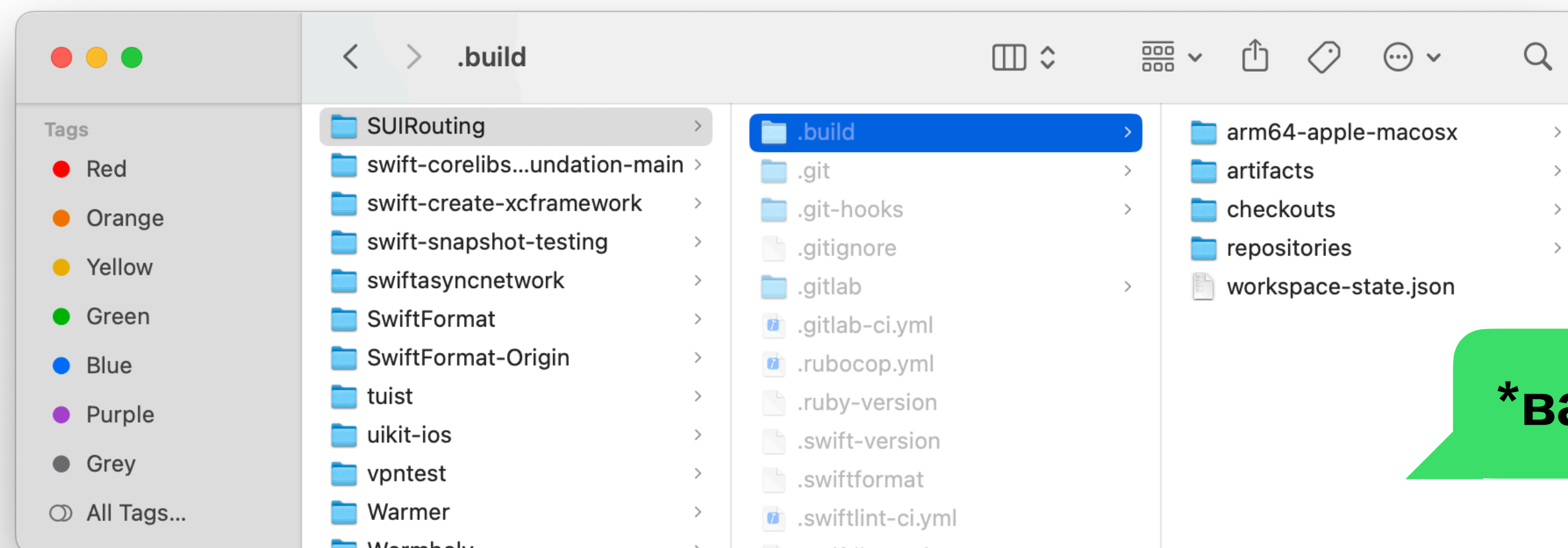
Build Timeline

Кто, зачем и сколько?



Вычеркиваем чекаут зависимостей

— Для plain Package.swift-файла



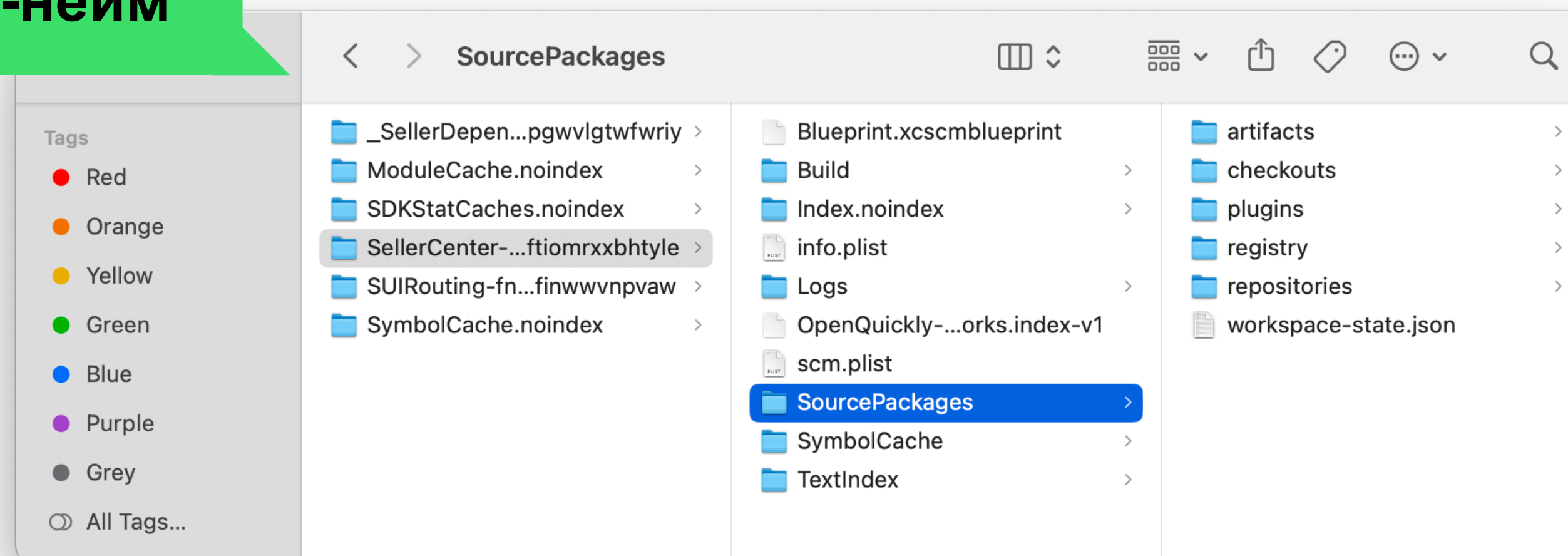
```
1 swift package resolve \  
2   --build-path *ваша-папка-нейм* \  
3   ...
```

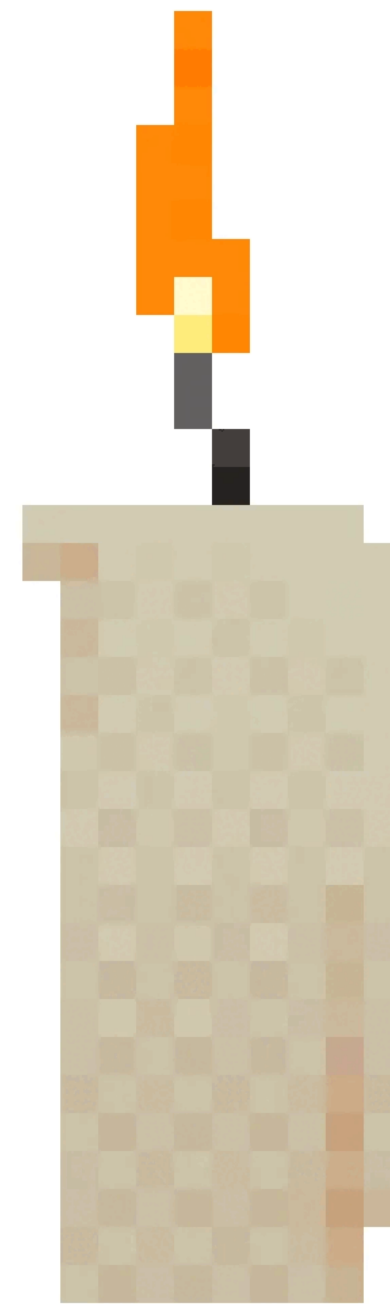
Вычеркиваем чекаут зависимостей

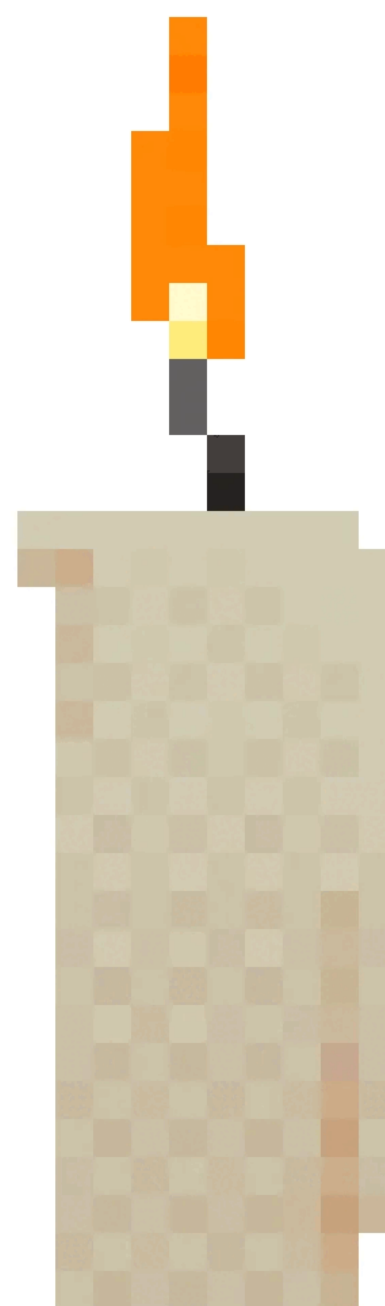
— Для *.xcodproj-файла

```
1 xcodebuild build \  
2   -scheme *ваша-схема-нейм* \  
3   -clonedSourcePackagesDirPath *ваша-папка-нейм* \  
4   ...
```

ваша-папка-нейм







Вычеркиваем билд сурсов зависимостей

Вычеркиваем же, правда?



```
1 // swift-tools-version:5.3
2 import PackageDescription
3
4 let package = Package(
5     name: "AppsFlyerLib",
6     products: [
7         .library(
8             name: "AppsFlyerLib",
9             targets: ["AppsFlyerLib"]
10        )
11    ],
12    targets: [
13        .binaryTarget(
14            name: "AppsFlyerLib",
15            url: "https://github.com/AppsFlyerSDK/AppsFlyerFramework/releases/download/"
16                + "6.14.3/AppsFlyerLib-Static-SPM.xcframework.zip",
17            checksum: "53c824272e8729967af6b22e013577eccbc6e412b7e2da2805ddf3c1ecf7cfc5"
18        )
19    ]
20 )
```

Вычеркиваем билд сурсов зависимостей

Вычеркиваем же, правда?

The screenshot shows a GitHub pull request interface. At the top, the repository is identified as 'apple / swift-evolution'. The pull request is titled '0272-swiftpm-binary-dependencies.md' and was created by user 'benrimington' 4 years ago. The pull request description includes the following information:

- Proposal: [SE-0272](#)
- Authors: [Braden Scothern](#), [Daniel Dunbar](#), [Franz Busch](#)
- Review Manager: [Boris Bügling](#)
- Status: **Implemented (Swift 5.3)**
- Implementation: [apple/swift-package-manager#2509](#), [apple/swift-package-manager#2511](#), [apple/swift-package-manager#2514](#), [apple/swift-package-manager#2588](#)
- Decision Notes: [First Review](#), [Second Review](#)

Below the description, there is a 'Contents' section with the following links:

- [Introduction](#)
- [Motivation](#)
- [Proposed solution](#)
- [Detailed design](#)
- [New PackageDescription API](#)
- [New Package.resolved Behavior](#)
- [Binary Target Artifact Format](#)

The interface also shows a file explorer on the left with a list of proposal files under the 'proposals' directory, and a navigation bar at the top with options like 'Code', 'Pull requests', 'Projects', 'Security', and 'Insights'.

Вычеркиваем билд сурсов зависимостей

Вычеркиваем же, правда?

```
1 // swift-tools-version:5.3
2 import PackageDescription
3
4 let package = Package(
5     name: "AppsFlyerLib",
6     products: [
7         .library(
8             name: "AppsFlyerLib",
9             targets: ["AppsFlyerLib"]
10        )
11    ],
12    targets: [
13        .binaryTarget(
14            name: "AppsFlyerLib",
15            url: "https://github.com/AppsFlyerSDK/AppsFlyerFramework/releases/download/"
16                + "6.14.3/AppsFlyerLib-Static-SPM.xcframework.zip",
17            checksum: "53c824272e8729967af6b22e013577eccbc6e412b7e2da2805ddf3c1ecf7cfc5"
18        )
19    ]
20 )
```




~~Вычеркиваем билд сурсов зависимостей~~

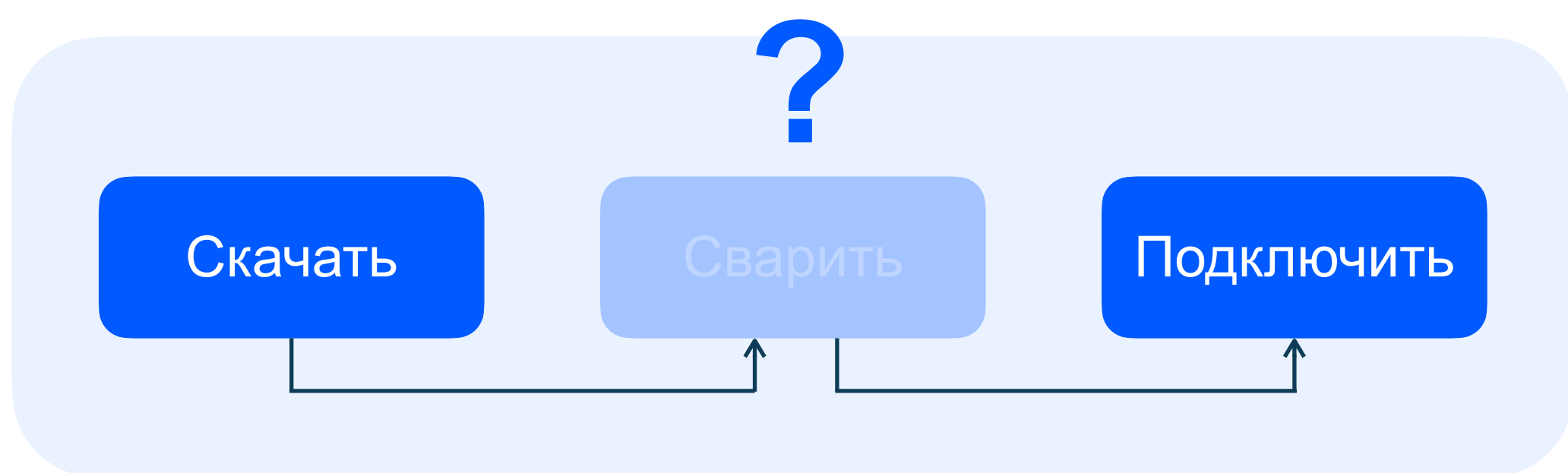
Так просто не выйдет 😭

```
1 // swift-tools-version:5.10
2 import PackageDescription
3
4 let package = Package(
5     name: "Alamofire",
6     products: [
7         .library(name: "Alamofire", targets: ["Alamofire"]),
8         .library(name: "AlamofireDynamic", type: .dynamic, targets: ["Alamofire"])
9     ],
10    targets: [
11        .target(
12            name: "Alamofire",
13            path: "Source",
14            exclude: ["Info.plist"],
15            resources: [
16                .process("PrivacyInfo.xcprivacy")
17            ],
18            linkerSettings: [
19                .linkedFramework(
20                    "CFNetwork",
21                    .when(platforms: [.iOS, .macOS, .tvOS, .watchOS])
22                ),
23            ],
24        ),
25    ]
26 )
```




Нормальные инструменты

SPM



```
1 swift package generate-xcodeproj ...
```

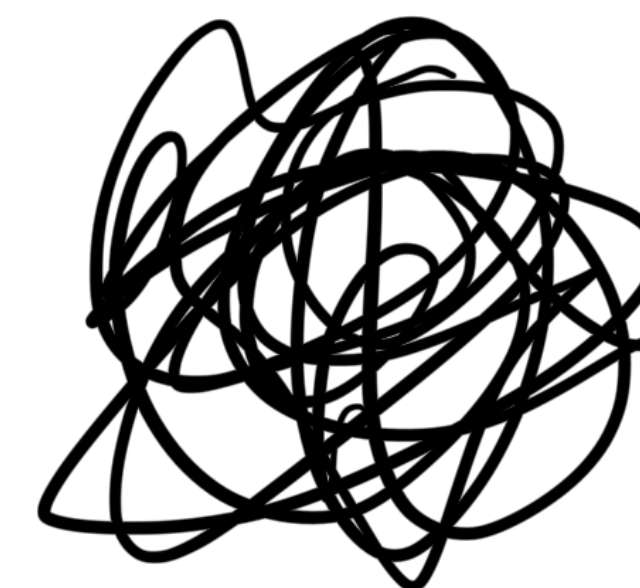
 **swift-create-xcframework** Public

 **spm-to-xcframework** Public

Вычеркиваем билд сурсов зависимостей

Так просто не выйдет 😭

```
1 // swift-tools-version:5.10
2 import PackageDescription
3
4 let package = Package(
5     name: "Alamofire",
6     products: [
7         .library(name: "Alamofire", targets: ["Alamofire"]),
8         .library(name: "AlamofireDynamic", type: .dynamic, targets: ["Alamofire"])
9     ],
10    targets: [
11        .target(
12            name: "Alamofire",
13            path: "Source",
14            exclude: ["Info.plist"],
15            resources: [
16                .process("PrivacyInfo.xcprivacy")
17            ],
18            linkerSettings: [
19                .linkedFramework(
20                    "CFNetwork",
21                    .when(platforms: [.iOS, .macOS, .tvOS, .watchOS])
22                ),
23            ],
24        ),
25    ],
26 )
```



Вычеркиваем билд сурсов зависимостей

Так просто не выйдет 😭

```
1 // swift-tools-version:5.3
2
3 import PackageDescription
4
5 let package = Package(
6     name: "Moya",
7     platforms: [
8         .macOS(.v10_12),
9         .iOS(.v10),
10        .tvOS(.v10),
11        .watchOS(.v3)
12    ],
13    products: [
14        .library(name: "Moya", targets: ["Moya"]),
15        .library(name: "CombineMoya", targets: ["CombineMoya"]),
16        .library(name: "ReactiveMoya", targets: ["ReactiveMoya"]),
17        .library(name: "RxMoya", targets: ["RxMoya"])
18    ],
19    dependencies: [
20        .package(url: "https://github.com/Alamofire/Alamofire.git", .upToNextMajor(from: "5.0.0")),
21        .package(url: "https://github.com/ReactiveCocoa/ReactiveSwift.git", .upToNextMajor(from: "6.0.0")),
22        .package(url: "https://github.com/ReactiveX/RxSwift.git", .upToNextMajor(from: "6.0.0")),
23        .package(url: "https://github.com/Quick/Quick.git", .upToNextMajor(from: "4.0.0")), // dev
24        .package(url: "https://github.com/Quick/Nimble.git", .upToNextMajor(from: "9.0.0")), // dev
25        .package(url: "https://github.com/AlSoftware/OHHTTPStubs.git", .upToNextMajor(from: "9.0.0")) //
26    ],
27    dev
28    ],
29    targets: [
30        .target(
31            name: "Moya",
32            dependencies: [
33                .product(name: "Alamofire", package: "Alamofire")
34            ],
35            exclude: [
36                "Supporting Files/Info.plist"
37            ]
38        ),
39        .target(
40            name: "CombineMoya",
41            dependencies: [
42                "Moya"
```



04



Инструменты Build-Cache зависимостей

Нормальные инструменты

Carthage



Скачать

Сварить

Подключить



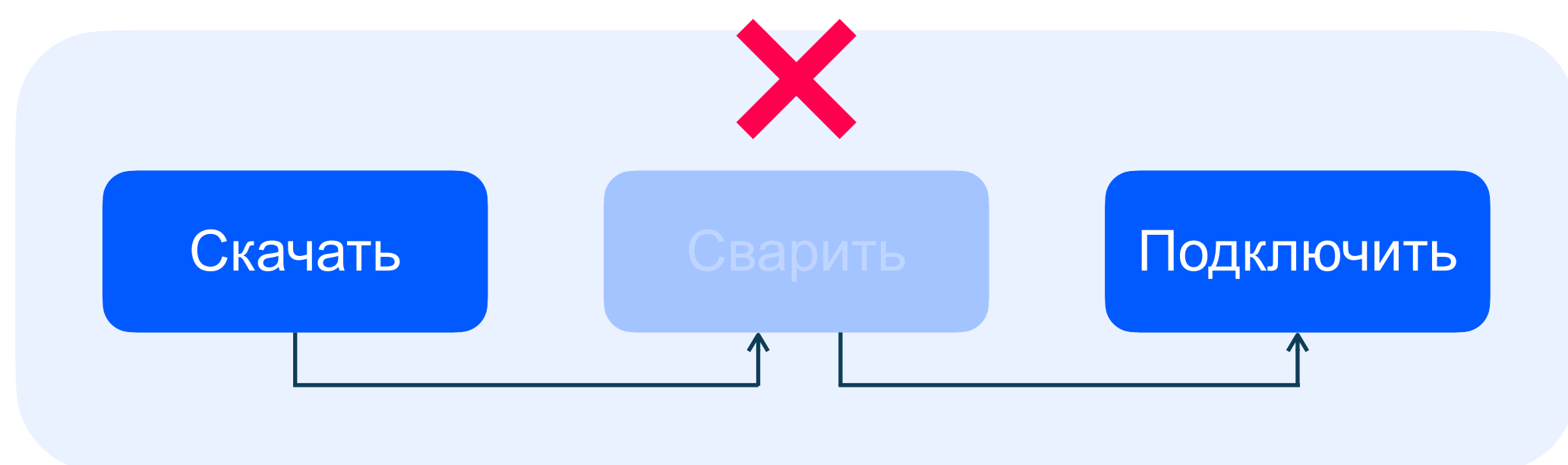
```
1 # Cartfile
2
3 github "Alamofire/Alamofire" ~> 5.5
```



```
1 oiu@kalkulator oiu % carthage update \
2     --use-xcframeworks
```

Нормальные инструменты

CocoaPods



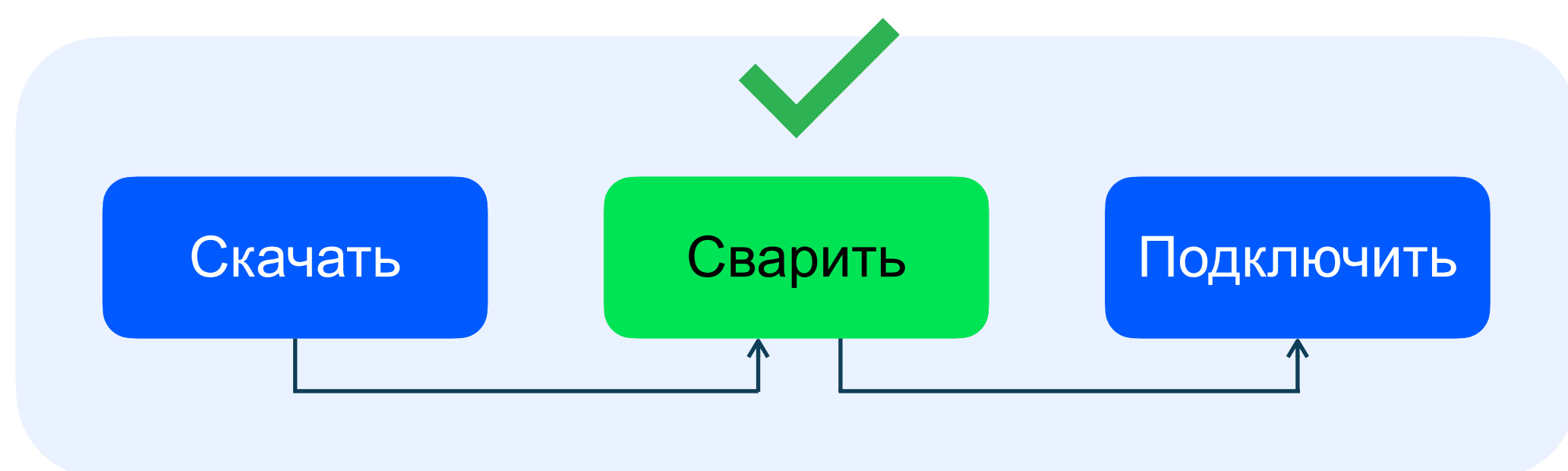
```
1 workspace 'Examples.xcworkspace'  
2 project 'iOS Example.xcodeproj'  
3  
4 target 'iOS Example' do  
5   platform :ios, '8.0'  
6   use_frameworks!  
7   pod 'Alamofire', :path => './Alamofire'  
8 end
```



```
1 oiu@kalkulator oiu % pod install
```


Нормальные инструменты

CocoaPods Rugby



```
1 workspace 'Examples.xcworkspace'
2 project 'iOS Example.xcodeproj'
3
4 target 'iOS Example' do
5   platform :ios, '8.0'
6   use_frameworks!
7   pod 'Alamofire', :path => './Alamofire'
8 end
```



```
1 oiu@kalkulator oiu % pod install
2 oiu@kalkulator oiu % rugby
```

Нормальные инструменты

Tuist



Нормальные инструменты

Tuist



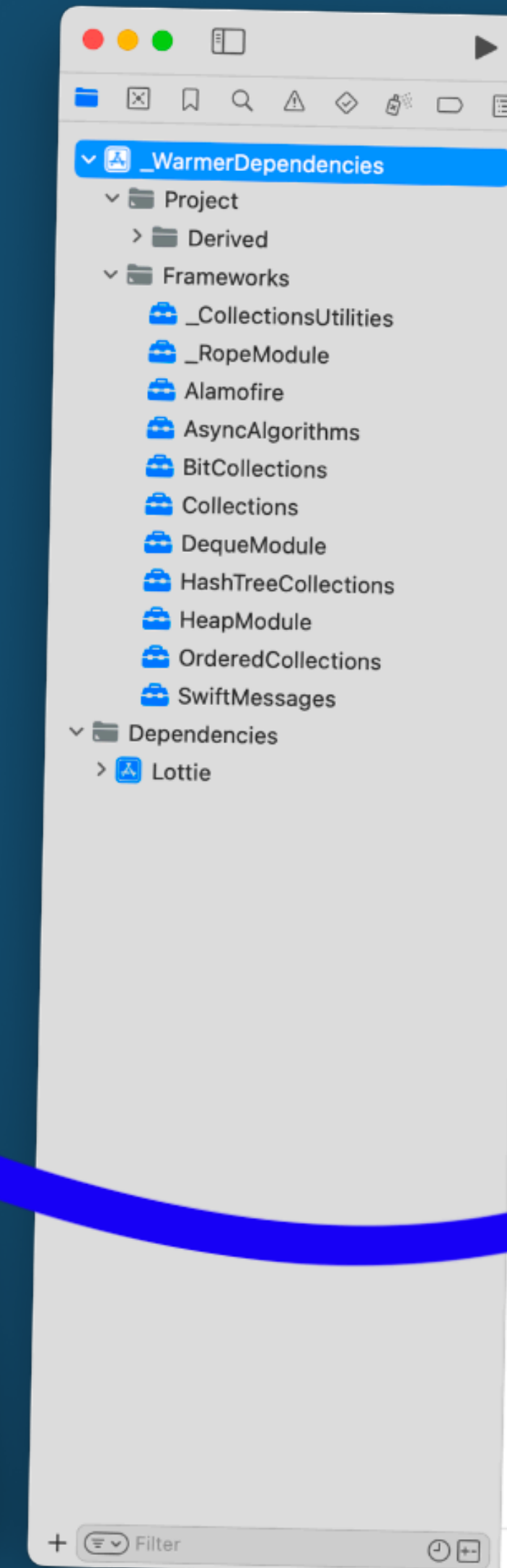
```
1 oiu@kalkulator oiu % tuist install
2 oiu@kalkulator oiu % tuist cache
3 oiu@kalkulator oiu % tuist generate
```

Скачать

Сварить

Подключить

```
1 // Package.swift
2 // swift-tools-version: 5.9
3
4 import PackageDescription
5
6 let package = Package(
7     name: "my-package",
8     platforms: [.iOS("15.0")],
9     dependencies: [
10         .package(
11             url: "swift-async-algorithms.git",
12             exact: "1.0.0"
13         ),
14         .package(
15             url: "Alamofire.git",
16             exact: "5.9.1"
17         ),
18         .package(
19             url: "lottie-ios.git",
20             exact: "4.4.3"
21         ),
22         .package(
23             url: "SwiftMessages.git",
24             exact: "10.0.0"
25         ),
26     ]
27 )
28
```



Сводная таблица инструментов

И их возможностей

	Скачать	Закэшировать	Подключить
Carthage	✓	✓	✓⊖
CocoaPods	✓	⊖	✓
Rugby	✓	✓	✓
Tuist	✓	✓	✓
Swift Package Manager	✓	⊖	✓

Почему не использовать готовое?

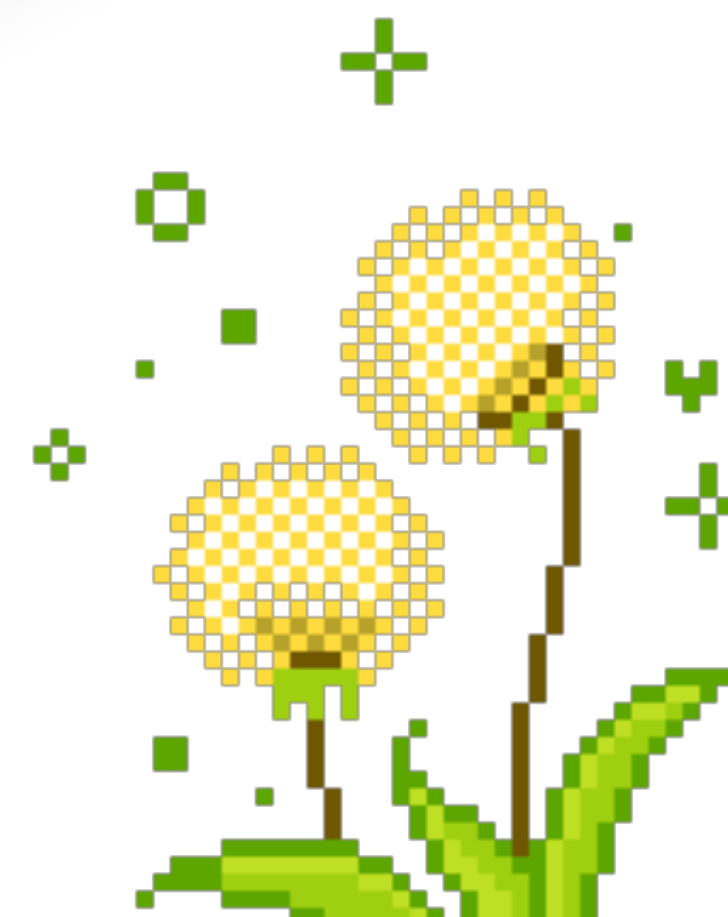
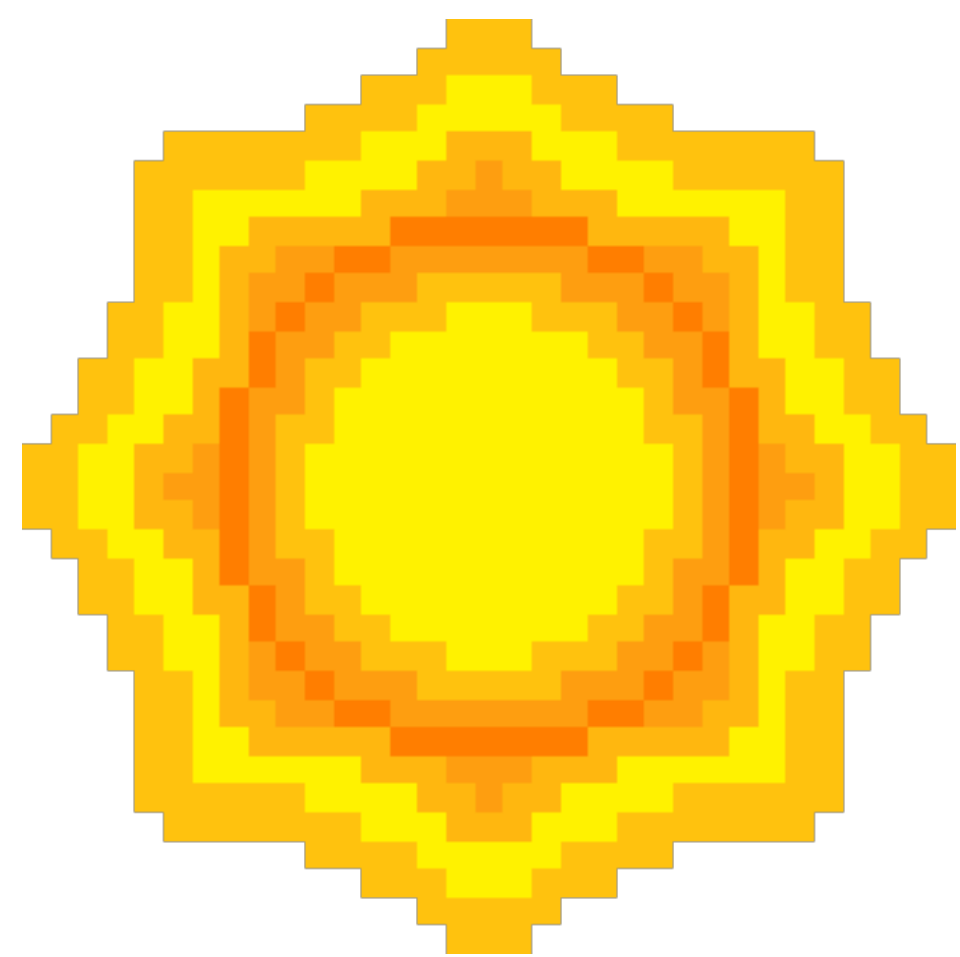
Overview



Устарели!

```
1 // swift-tools-version: 5.9
2
3 import PackageDescription
4
5 let package = Package(
6     name: "my-feature",
7     platforms: [.iOS(.v15)],
8     products: [
9         .library(
10             name: "my-feature",
11             targets: ["my-feature"]
12         ),
13     ],
14     targets: [
15         .target(
16             name: "my-feature",
17             dependencies: [
18                 .target(name: "Nuke"),
19             ]
20         ),
21         .binaryTarget(
22             name: "Nuke",
23             path: "Sources/Nuke.xcframework"
24         ),
25     ]
26 )
```





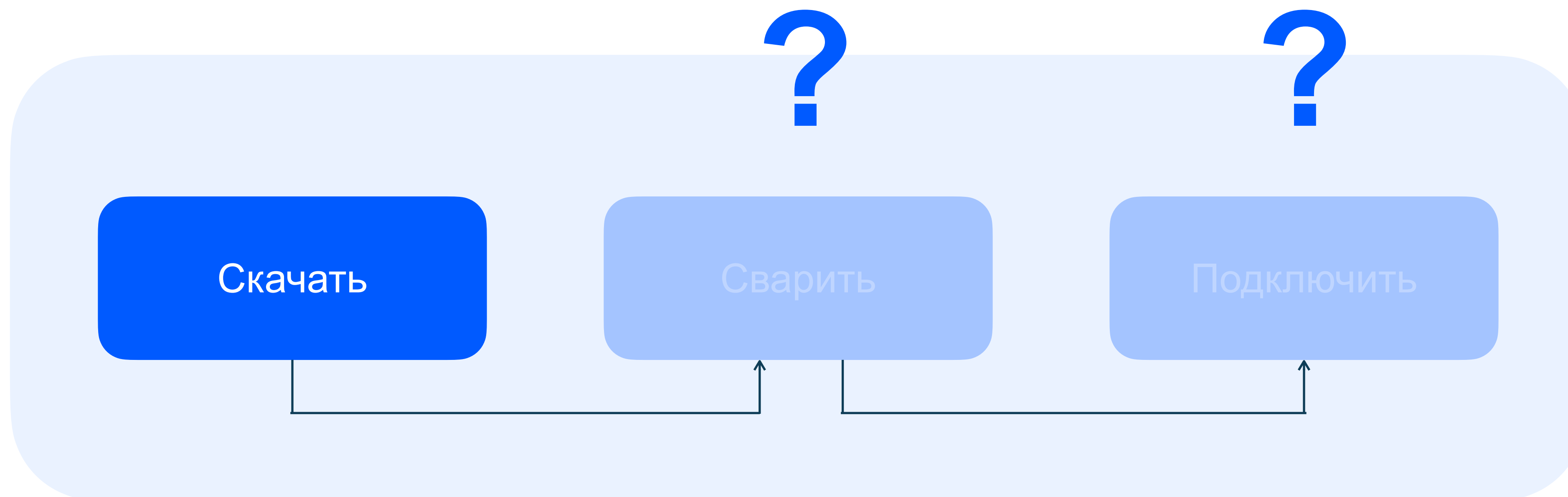
05



Накладываем
схему на SPM

Восстанавливаем схему для SPM

Чего нам не хватает?



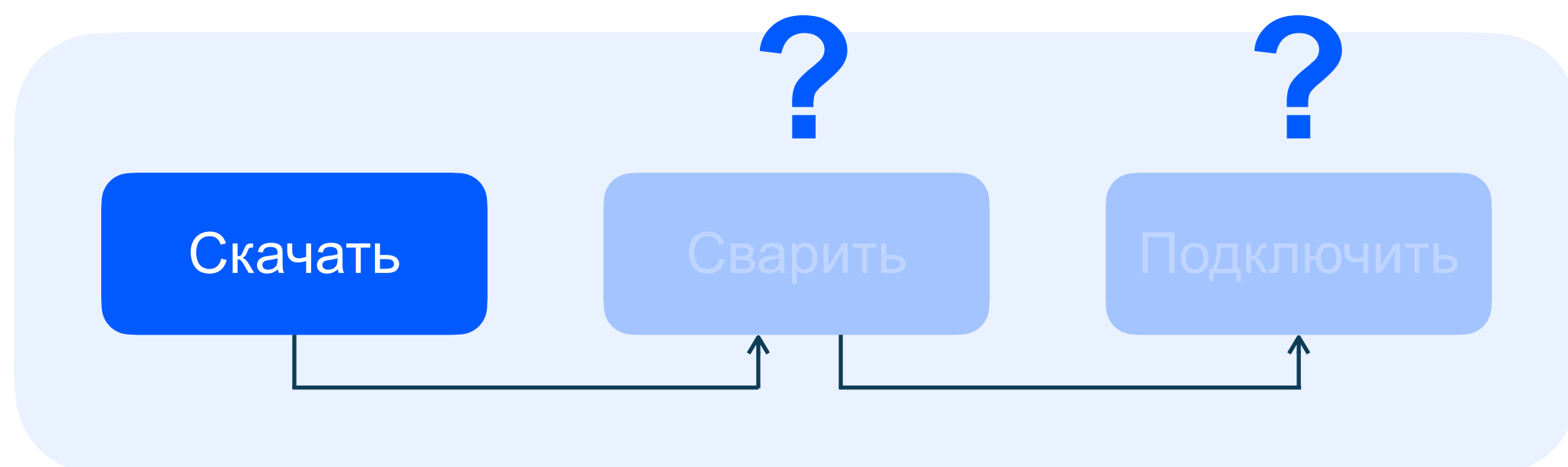
Инструмент может быть любим

Подключение бинарных таргетов в манифест в любом случае придется делать руками, а откуда возьмутся сами бинари — абсолютно неважно



Выборы кандидатов

*в случае нашего проекта



Зачем?



Уже в проекте

подставьте свой инструмент-нейм



Кусочки недостающего пазла

Pen Pineapple Apple Pen



1



```
1 oiu@kalkulator oiu % tuist install
```



2



```
1 oiu@kalkulator oiu % tuist cache
```



3

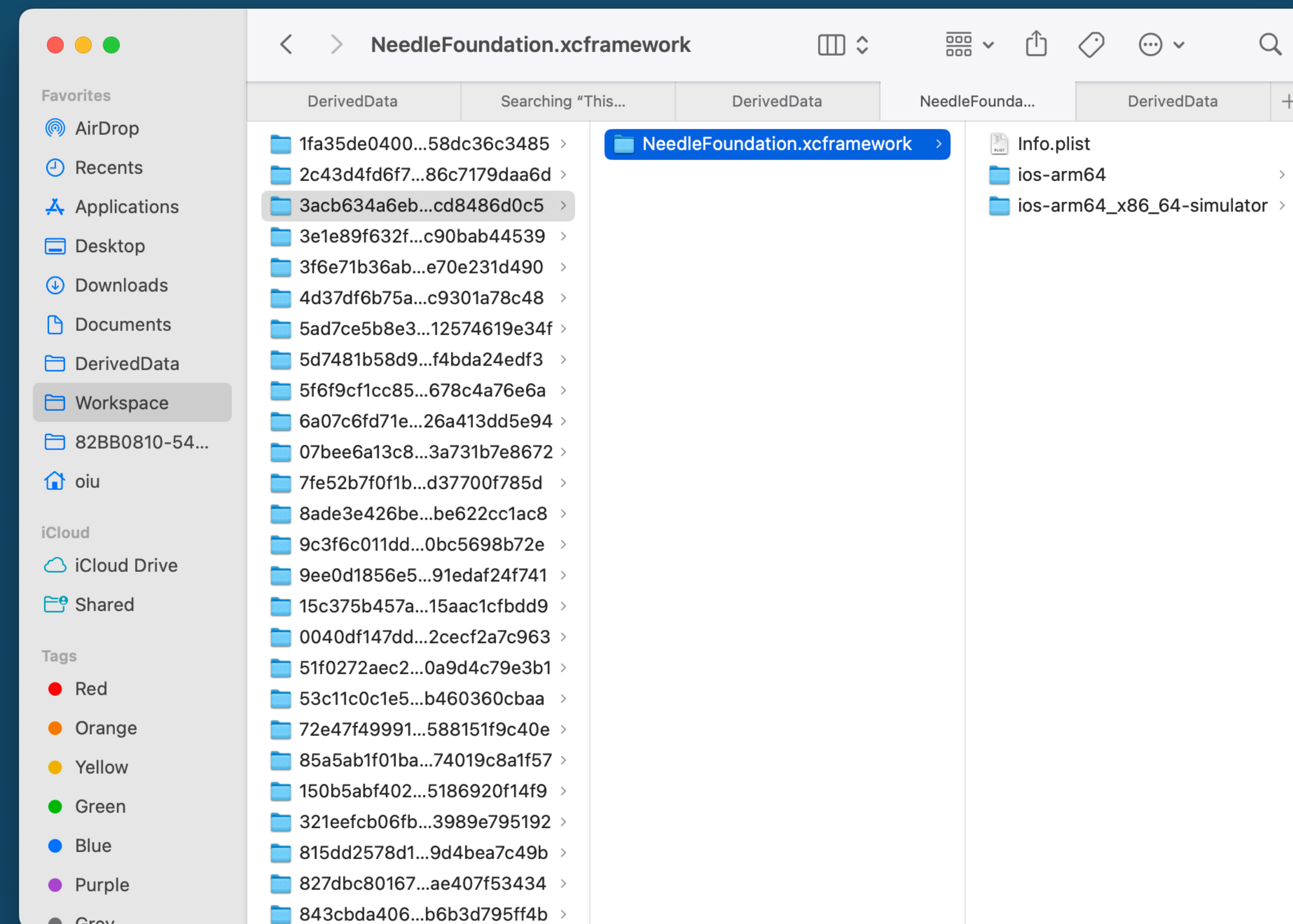
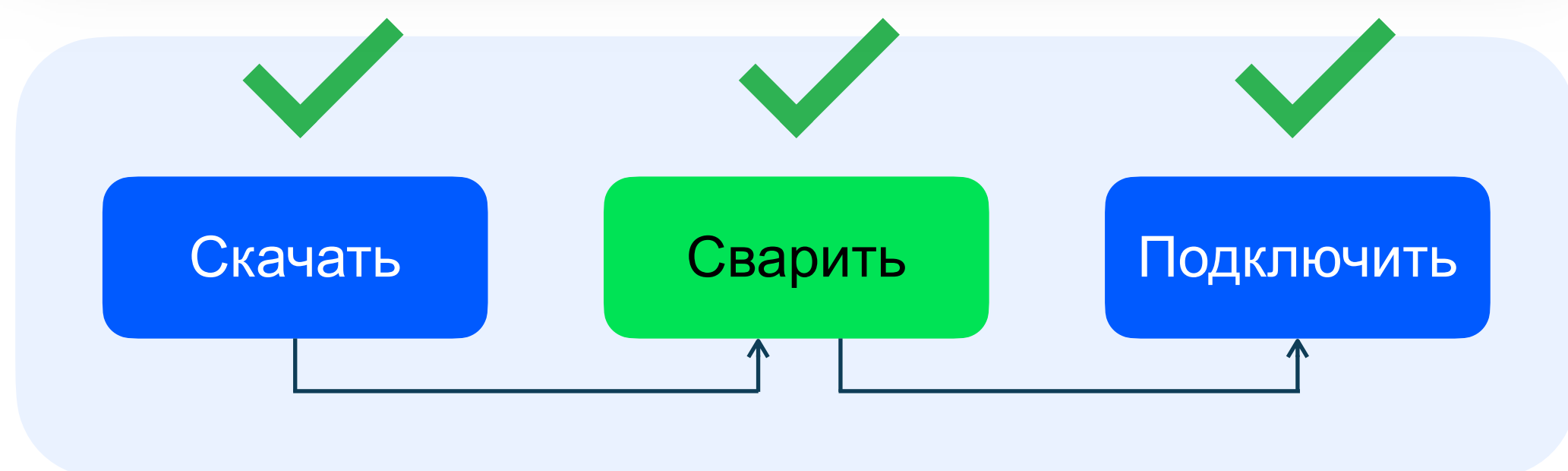


```
1 oiu@kalkulator oiu % tuist generate
```

Нормальные инструменты

Tuist

```
1 oiu@kalkulator oiu % tuist install
2 oiu@kalkulator oiu % tuist cache
3 oiu@kalkulator oiu % tuist generate
```



Кусочки недостающего пазла

Pen Pineapple Apple Pen



1



```
1 oiu@kalkulator oiu % tuist install
```



2



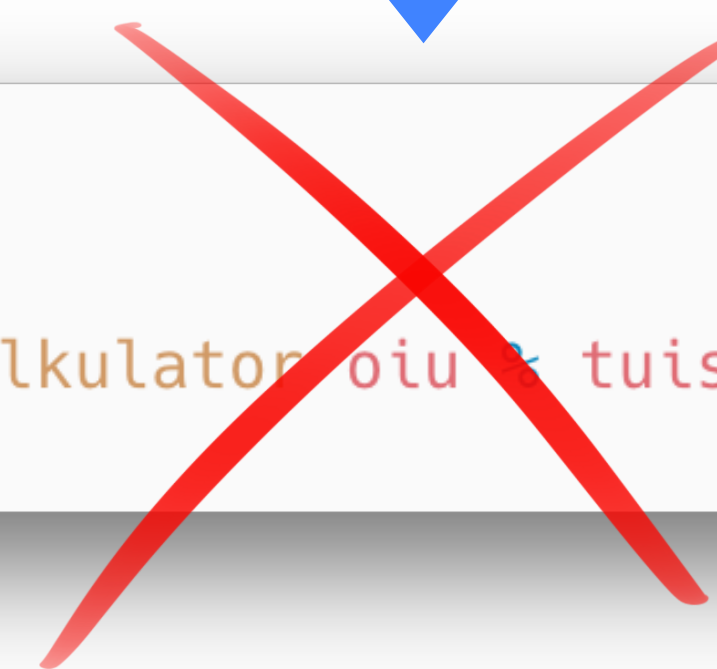
```
1 oiu@kalkulator oiu % tuist cache
```



3



```
1 oiu@kalkulator oiu % tuist generate
```



Варианты подключения

Первый: **попроще**



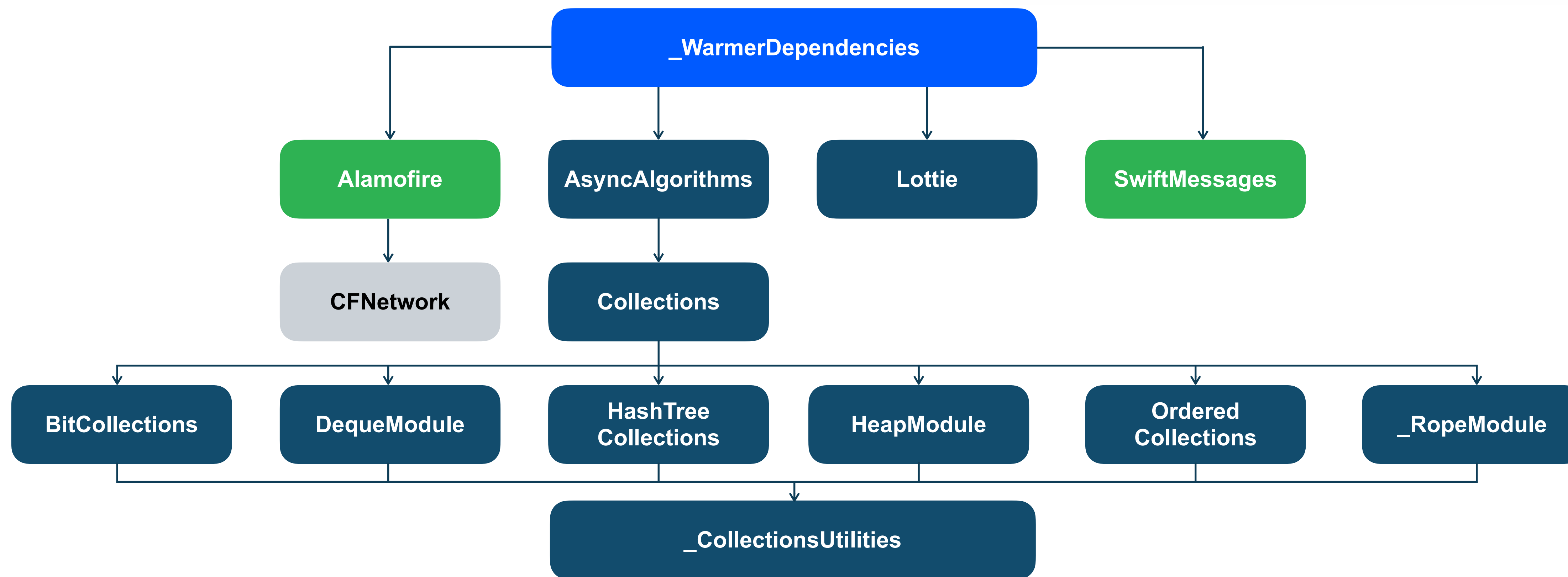
```
1 // swift-tools-version: 5.9
2
3 import PackageDescription
4
5 let package = Package(
6     name: "my-feature",
7     platforms: [.iOS(.v15)],
8     products: [
9         .library(
10             name: "my-feature",
11             targets: ["my-feature"]
12         ),
13     ],
14     targets: [
15         .target(
16             name: "my-feature",
17             dependencies: [
18                 .target(name: "Nuke"), ←
19             ]
20         ),
21         .binaryTarget(
22             name: "Nuke",
23             path: "Sources/Nuke.xcframework" ←
24         ),
25     ]
26 )
```

Варианты подключения

Второй: **вариативнее**



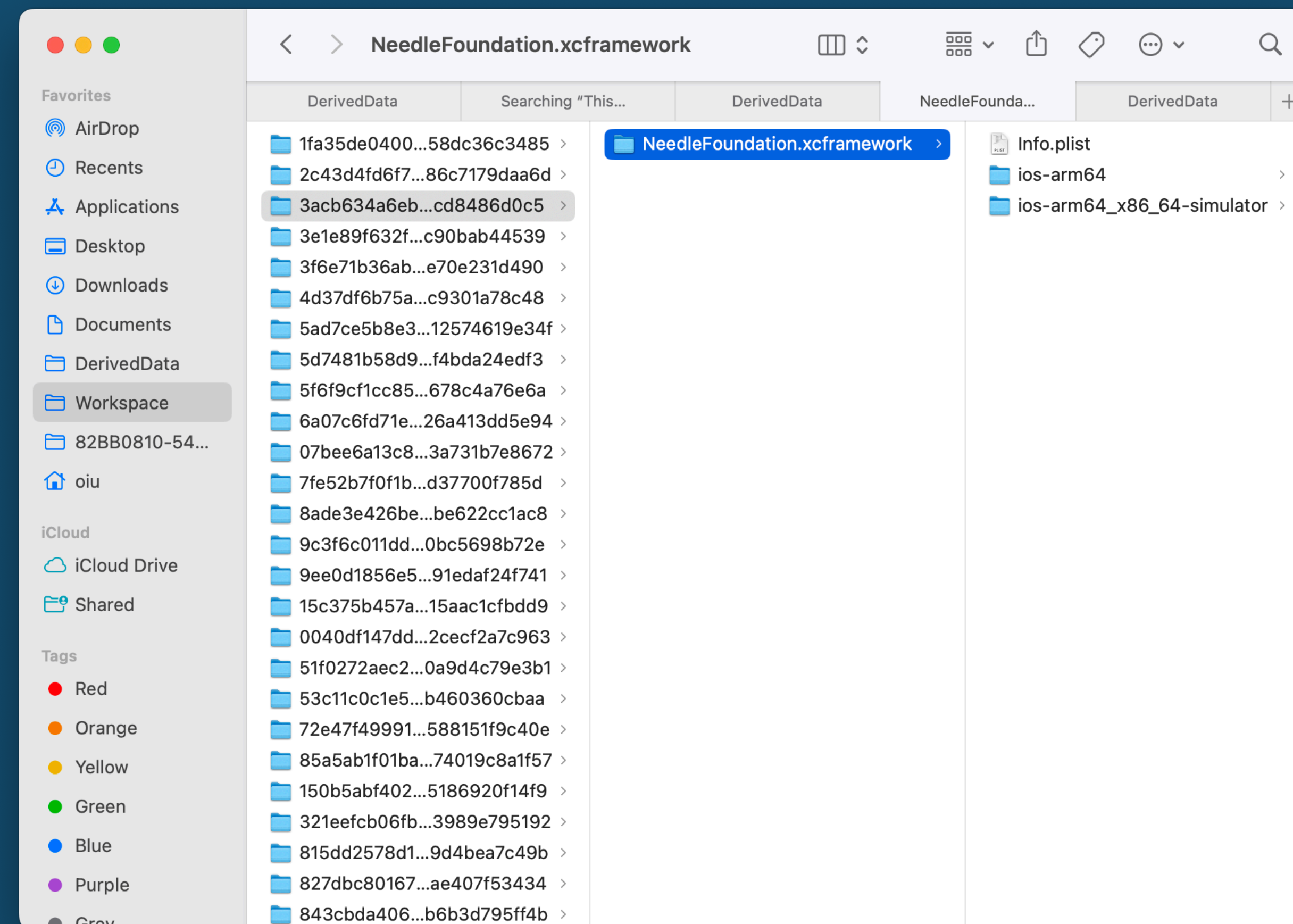
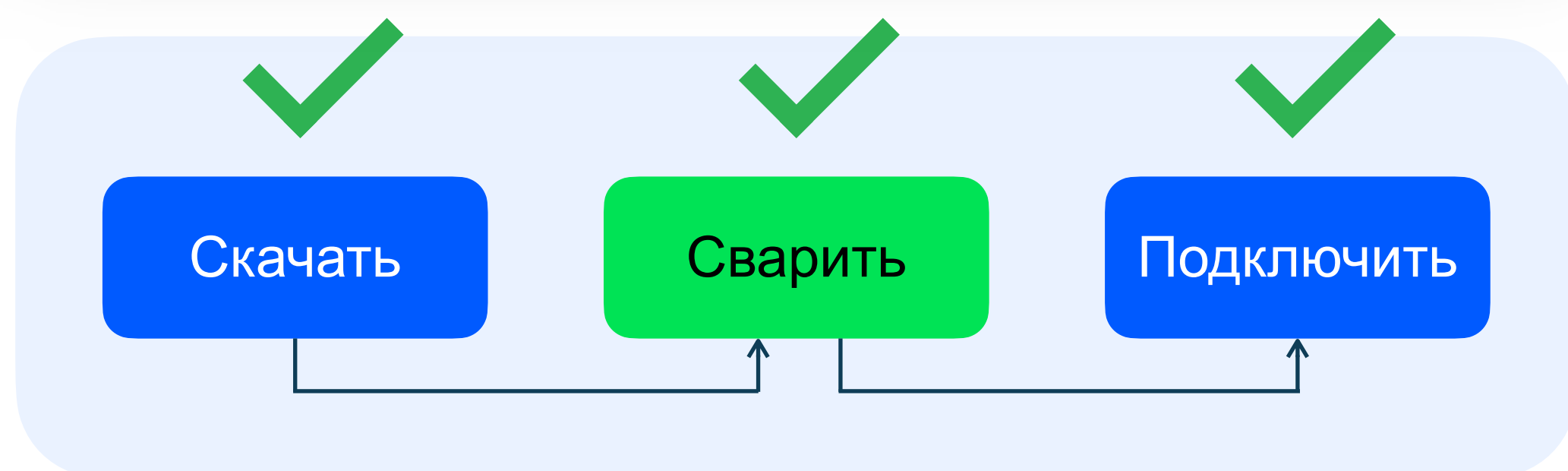
```
1 oiu@kalkulator oiu % tuist graph  
2 // опционально: -f json
```



Третий кусочек пазла

Подключение

```
1 oiu@kalkulator oiu % tuist install
2 oiu@kalkulator oiu % tuist cache
3 oiu@kalkulator oiu % tuist generate
```



Третий кусочек пазла

Подключение



```
1 oiu@kalkulator oiu % tuist install
2 oiu@kalkulator oiu % tuist cache
3 oiu@kalkulator oiu % tuist generate
```

Скачать

Сварить

Подключить

```
18 targets: [
19     .target(
20         name: "WarmerDependencies",
21         dependencies: [
22             .target(name: "HeapModuleFramework"),
23             .target(name: "OrderedCollectionsFramework"),
24             .target(name: "LottieFramework"),
25             .target(name: "_CollectionsUtilitiesFramework"),
26             .target(name: "CollectionsFramework"),
27             .target(name: "AlamofireFramework"),
28             .target(name: "DequeModuleFramework"),
29             .target(name: "HashTreeCollectionsFramework"),
30             .target(name: "BitCollectionsFramework"),
31             .target(name: "AsyncAlgorithmsFramework"),
32             .target(name: "_RopeModuleFramework"),
33             .target(name: "SwiftMessagesFramework"),
34         ]
35     ),
36     .binaryTarget(
37         name: "HeapModuleFramework",
38         path: "./Caches/tuist-cloud/BinaryCache/c8e5b5a3ba7d0d9",
39     ),
40     .binaryTarget(
41         name: "OrderedCollectionsFramework",
42         path: "./Caches/tuist-cloud/BinaryCache/65102715afc23f4",
43     ),
44     .binaryTarget(
45         name: "LottieFramework",
46         path: "./Caches/tuist-cloud/BinaryCache/4de729d39b10270",
47     ),
48     .binaryTarget(
49         name: "_CollectionsUtilitiesFramework",
50         path:
```

ИСТОЧНИКИ

Вместо лайв-кодинга 🤔

- 1 **Изначальный коммит** →
- 2 **Заводим “пакет с пакетами”** →
- 3 **Обвязка Tuist`а — собираем фреймворки** →
- 4 **Скрипты генерации** →
- 5 **Хэширование версий зависимостей** →
- 6 **Бандлы SPM-зависимостей** →
- 7 **Генерация проекта исходниками** →



07



Финальные
бенчмарки

Базовые планки

Бъём?



12 мин. **30** сек.

Стандартная сборка

87 мин. **30** сек.

Машинное время билдов MP-а



Базовые планки

Рвём

12 мин. **30** сек.

Стандартная сборка



6 мин. **42** сек.

Стандартная сборка

87 мин. **30** сек.

Машинное время билдов MP-а



46 мин. **50** сек.

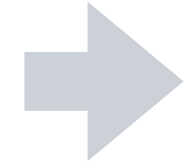
Машинное время билдов MP-а

Итоговые результаты

Ну по большей части рвём

12 мин. **30** сек.

Стандартная сборка



6 мин. **42** сек.

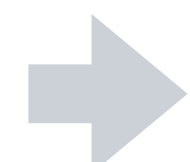
Стандартная сборка

19 мин. **10** сек.

Стандартная сборка
(cacheless)

87 мин. **30** сек.

Машинное время билдов MP-a



46 мин. **50** сек.

Машинное время билдов MP-a

Продолжительность, с.

360

70

690

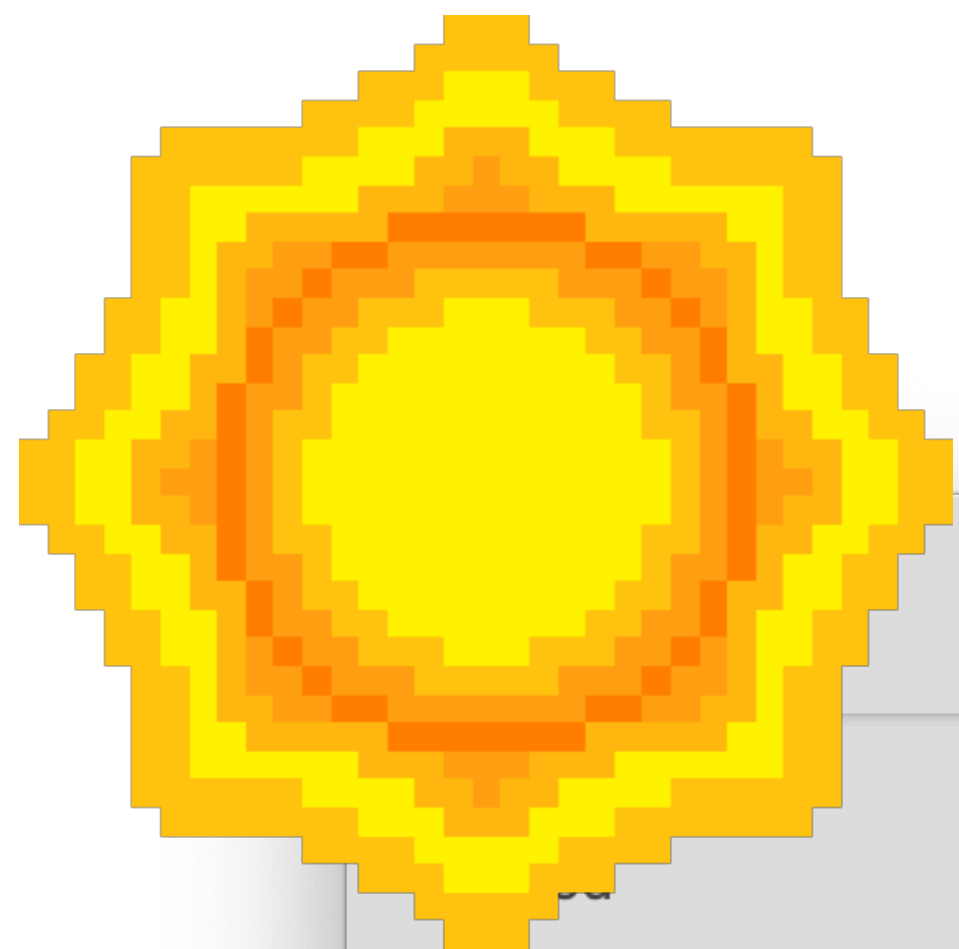
260

- Clean Build
- Iterative Build
- Resolve + Clean Build
- Cached Clean Build

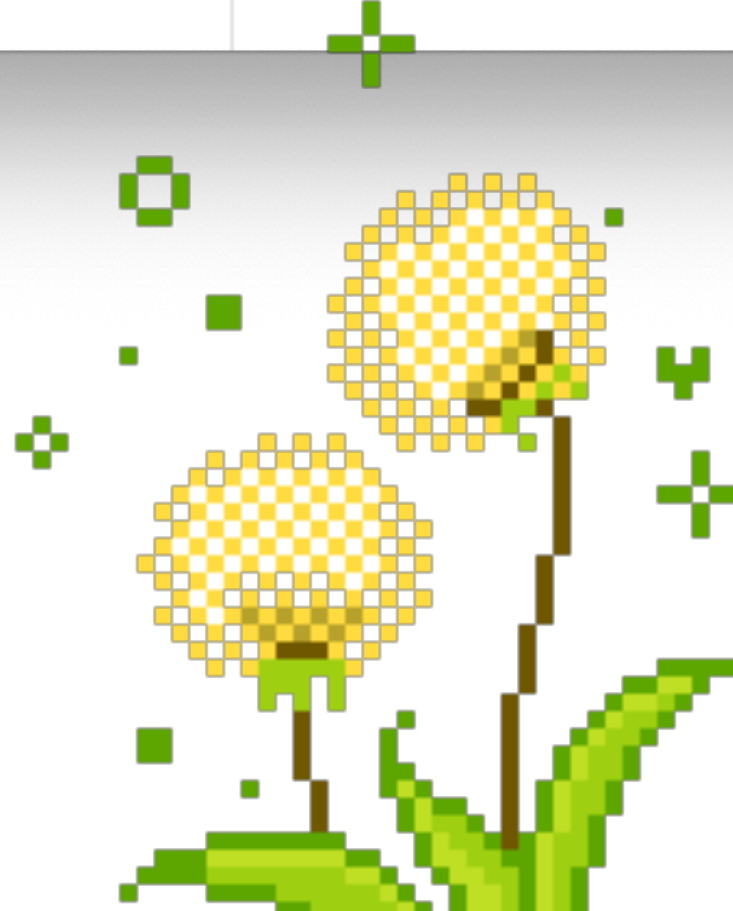
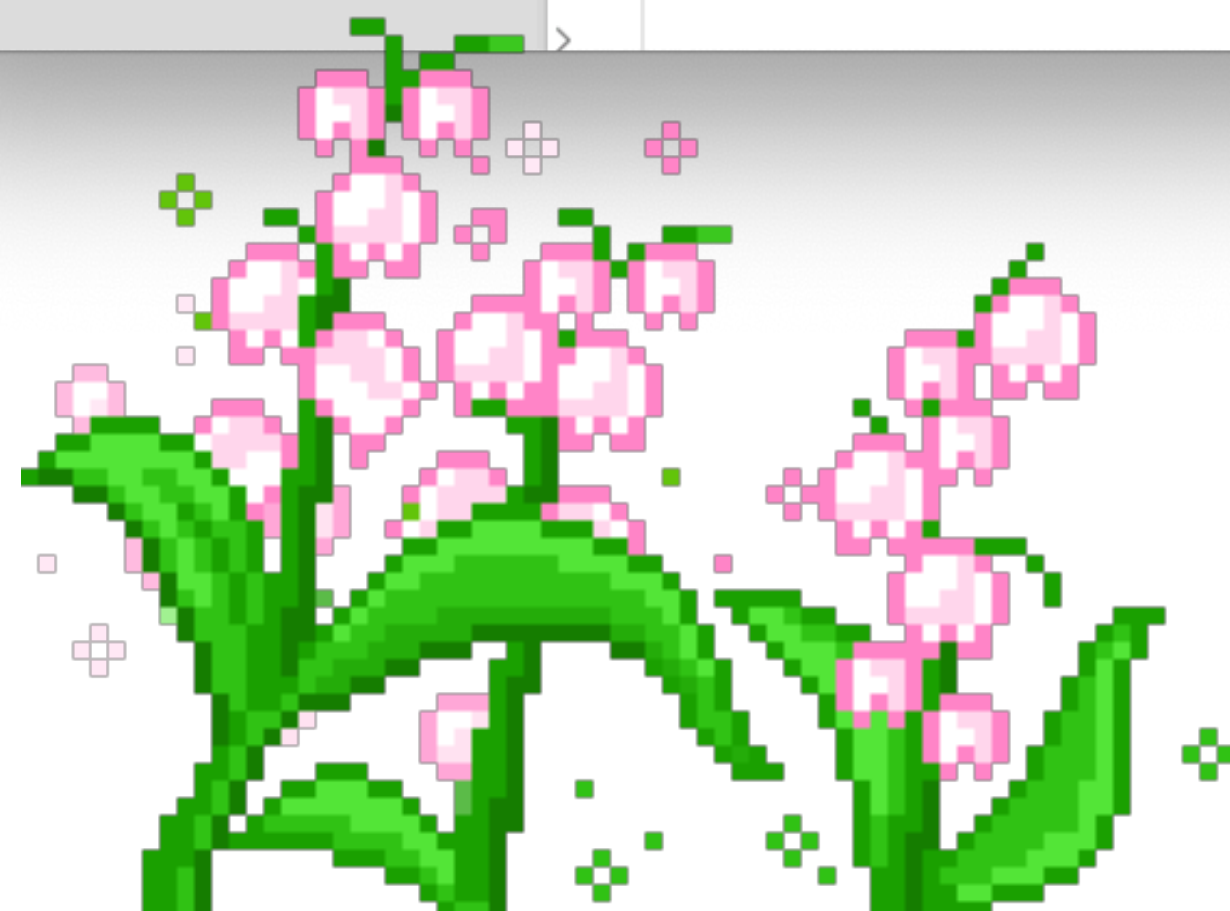
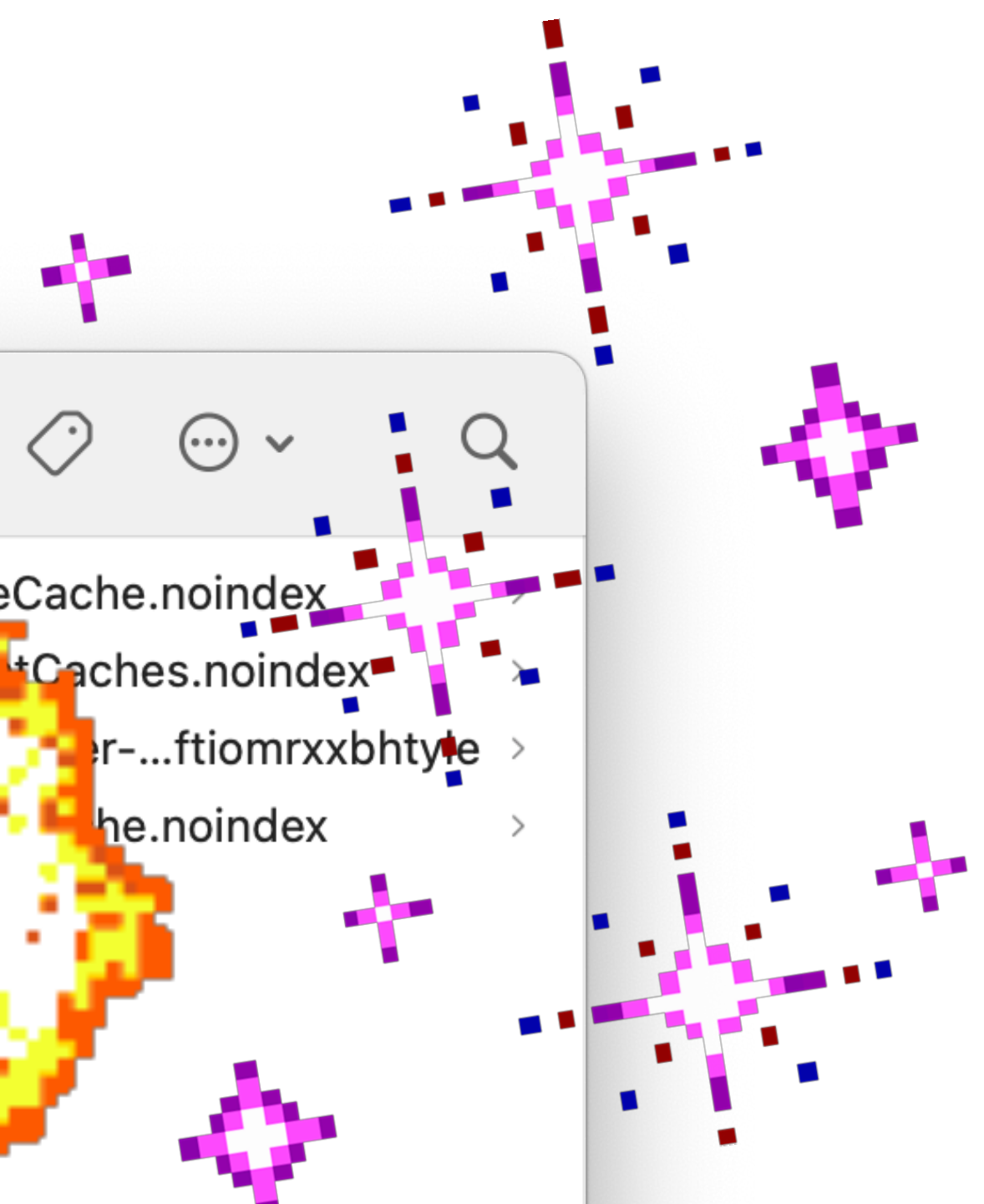
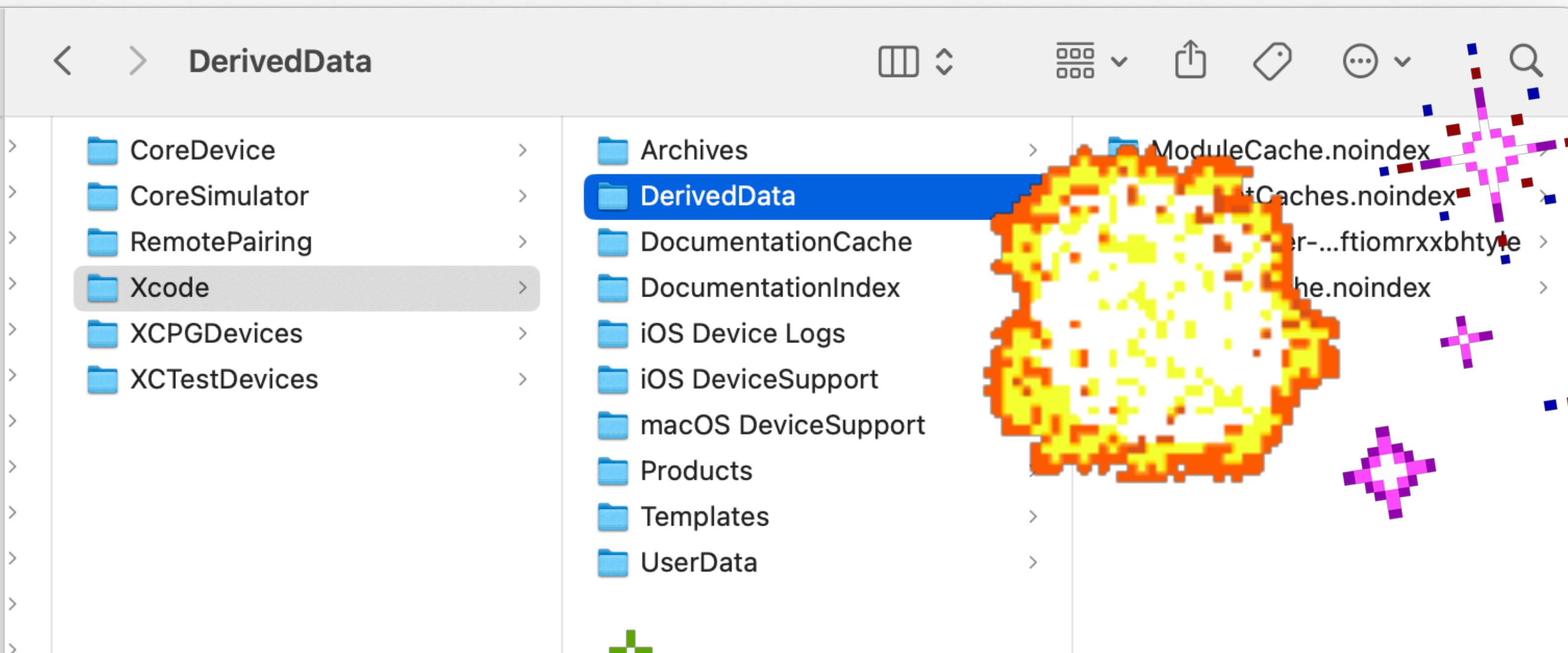
```
seller-app-ios — zsh — zsh (figterm) ▸ zsh — 68x14
...r-app-ios — zsh — zsh (figterm) ▸ zsh
...rnal — zsh — zsh (figterm) ▸ zsh ... +
Last login: Sat Feb 24 23:21:44 on console
[oiu@mbp-evgeryzhov-BYOD-WHH5P7YX2T seller-app-ios % make go
Заккрытие Xcode
No matching processes belonging to you were found
Установка git hooks
Простановка DI связей
Генерация ресурсов
Генерация проекта
Resolved cache profile 'Development' from Tuist's defaults
Generating workspace SellerCenter.xcworkspace
Generating project SellerCenter
Resolving package dependencies using xcodebuild
Project generated.
Total time taken: 334.816s
```

SellerCenter (Development Flow) Build

- Prepare packages
 - Compile plug-ins and run any prebuild commands
 - Compile plug-in "StaticVersionPlugin" in package "spm-plugins" 0.1 seconds
 - Apply build tool plug-in "StaticVersionPlugin" to target "Messenger" in package "snapshots.Messenger"
 - Try to find Versions.xcconfig file at /Users/oiu/Library/Developer/Xcode/DerivedData/SellerCenter... more
 - Versions.xcconfig content:
["OZ_VERSION_NUMBER = 8.3.0"]
 - Calculated version: 8.3.0
 - Process build tool plug-in results 0.1 seconds
- Run pre-actions
 - Build | Scheme SellerCenter (Development Flow)
 - Run custom shell script 'Run Sourcery and SwiftGen' 21.6 seconds
 - Run custom shell script 'Run Needle' 0.1 seconds
 - Run custom shell script 'Generate JsonFileNames' 0.1 seconds
- Prepare build
 - Workspace SellerCenter | Scheme SellerCenter (Development Flow) | Destination iPhone 15 Pro
 - Compute target dependency graph 1.3 seconds
 - Building targets in dependency order
 - Target dependency graph (269 targets)
 - Gather provisioning inputs 0.1 seconds
 - Build stat cache for /Applications/Xcode.app/Contents/Developer/Platforms/iPhoneSimulator.platform/... 0.1 seconds
 - Create build directory \$(OBJROOT) 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR) 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR)/Debug 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR)/Debug/PackageFrameworks 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR)/Debug-iphonesimulator 0.1 seconds
 - Create build directory \$(TARGET_BUILD_DIR)/Debug-iphonesimulator/PackageFrameworks 0.1 sec...
 - Create build directory \$(OBJROOT)/EagerLinkingTBDs/Debug-iphonesimulator 0.1 seconds
 - Build stat cache for /Applications/Xcode.app/Contents/Developer/Platforms/MacOSX.platform/Develop... 0.1 seconds
 - Process OzonAddressKit.xcframework (iOS Simulator) 0.1 seconds
 - Process Mapbox.xcframework (iOS Simulator) 0.2 seconds
 - Process OzonMapsKit.xcframework (iOS Simulator) 0.2 seconds



- Orange
- Yellow
- Green
- Blue
- Purple
- Grey
- All Tags...



08



Но какой
ценой?

Зачем всё это?

А главное — какой ценой

→ Затраты на разработку



Зачем всё это?

А главное — какой ценой

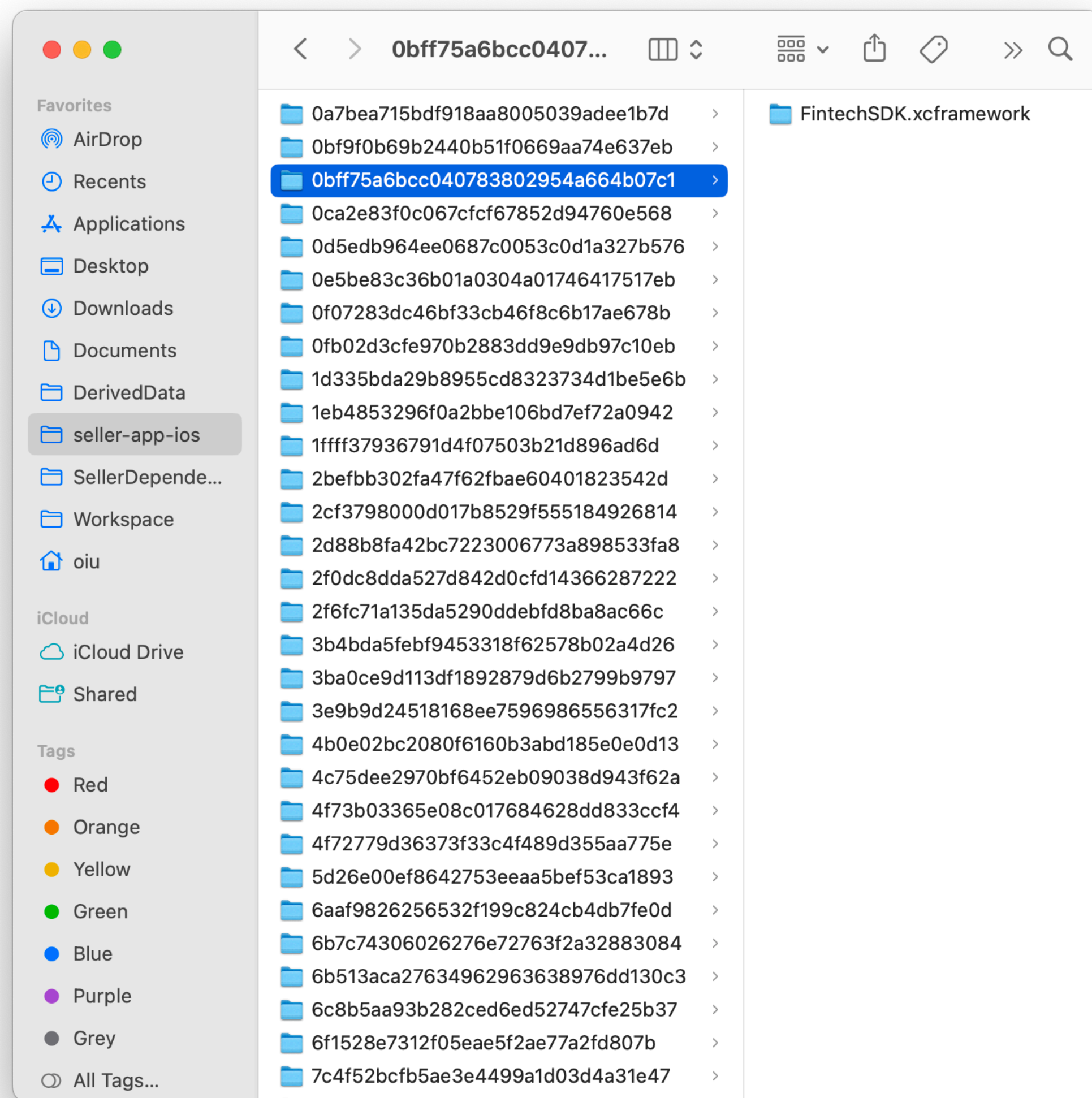
➔ Затраты на разработку

➔ Затраты на поддержку



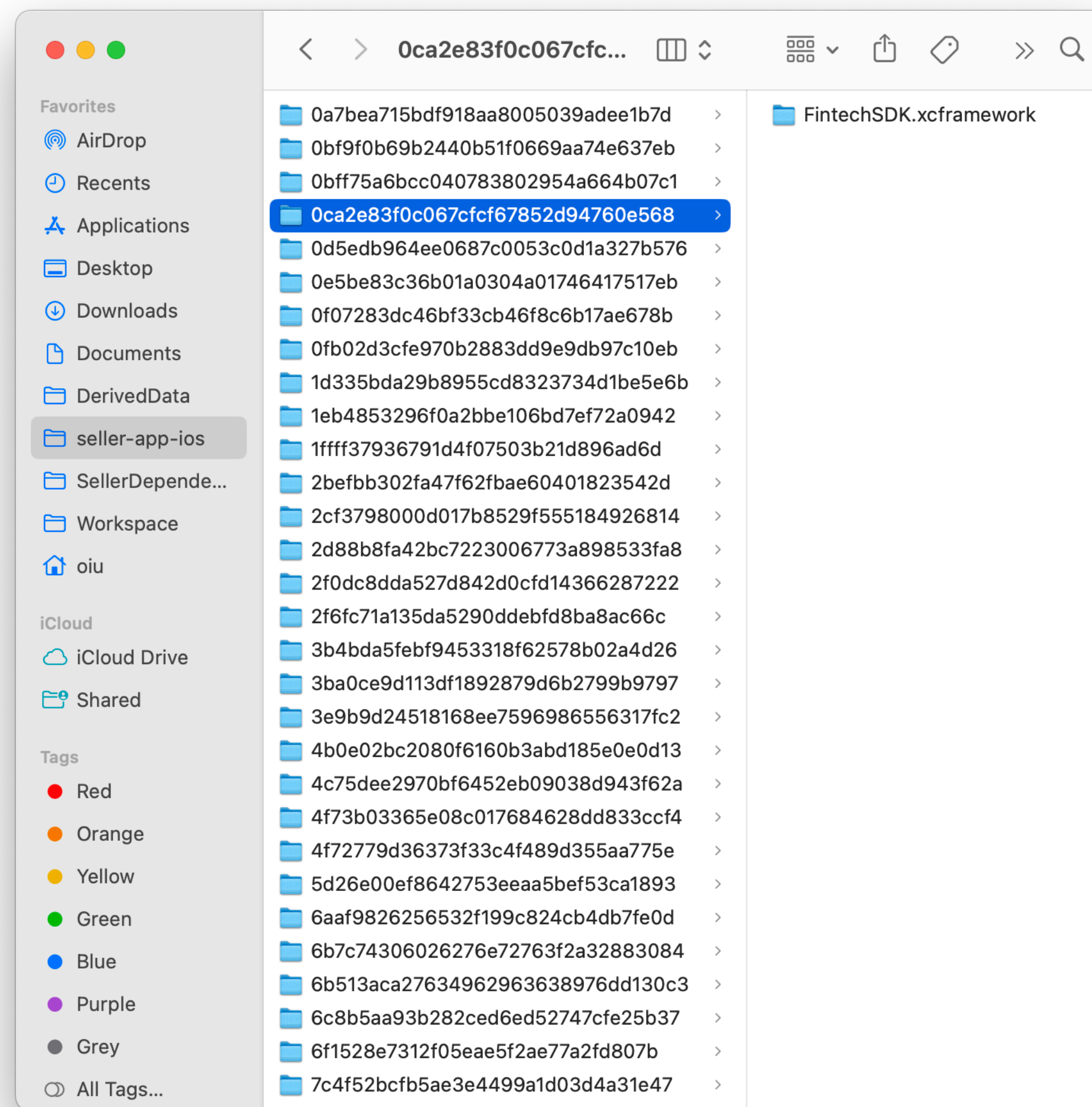
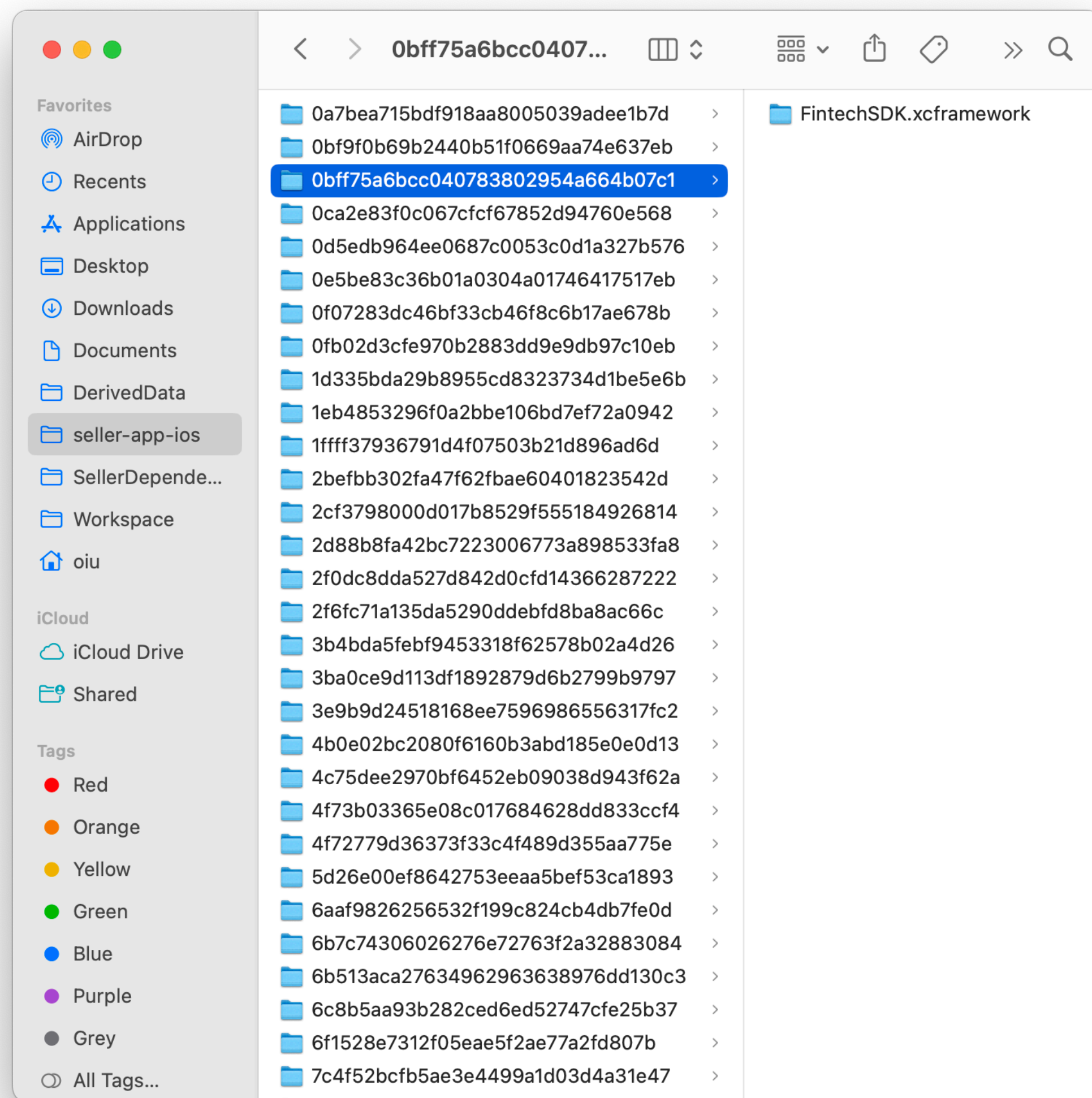
Хэши прогретых зависимостей

И их дубликаты



Хэши прогретых зависимостей

И их дубликаты



Хэши прогретых зависимостей

И их дубликаты

- 1 Изначальный коммит →
- 2 Заводим “пакет с пакетами” →
- 3 Обвязка Tuist`а — собираем фреймворки →
- 4 Скрипты генерации →
- 5 Хэширование версий зависимостей →**
- 6 Бандлы SPM-зависимостей →
- 7 Генерация проекта исходниками →



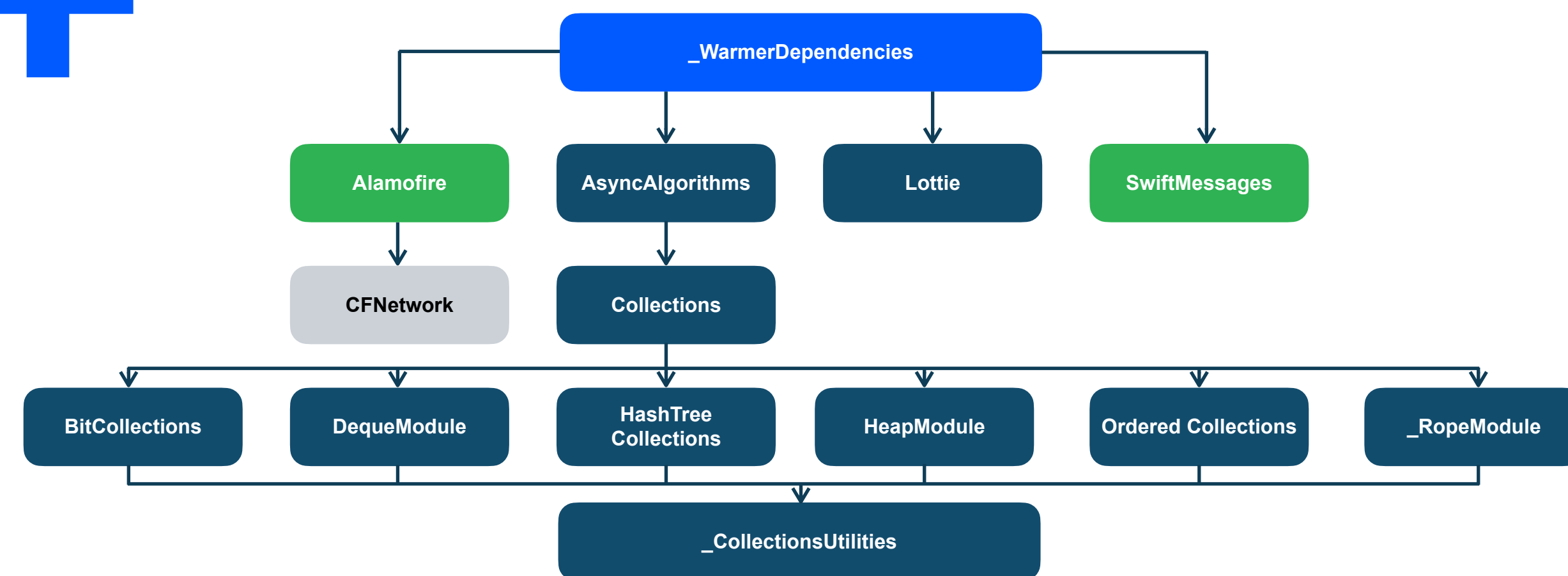
 @oiuhr

Хэши прогретых зависимостей

И их дубликаты

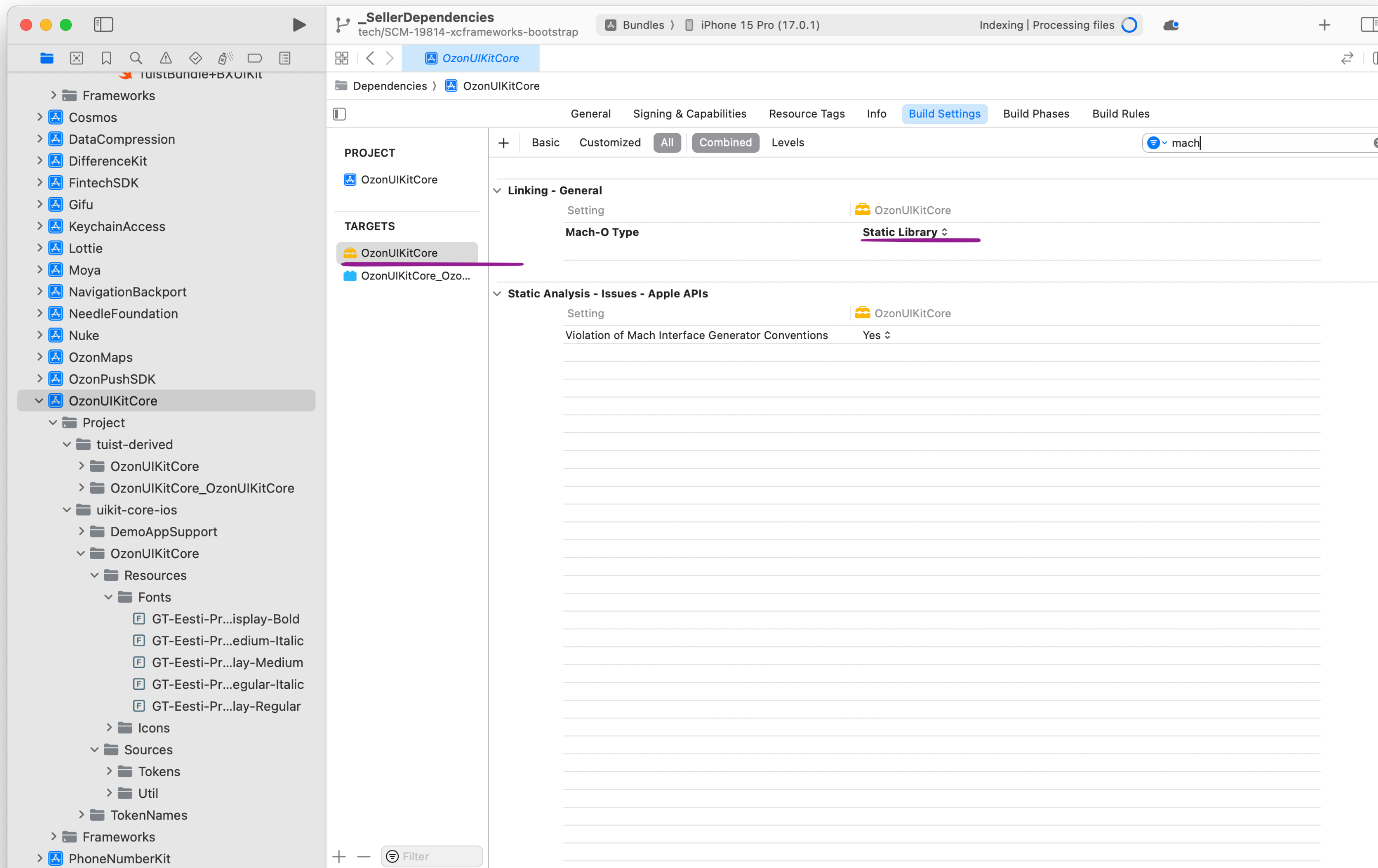
```
1 oiu@kalkulator .internal % tuist cache --print-hashes
2 Loading and constructing the graph
3 It might take a while if the cache is empty
4 Alamofire - c435572b0a7ad22c114f8d51a31ae436
5 AsyncAlgorithms - 720331ea3ae0fe949c993e79ce0189e8
6 BitCollections - 28a87950d21c2bd32a186e8af9b18ae3
7 Collections - 8092a7cf321e5baa360287afd327c627
8 DequeModule - 80aba64f69a2fabe9f88b498ce53a4ce
9 HashTreeCollections - fd1f6a880f83254e0eef63c6f05cc70b
10 HeapModule - c8e5b5a3ba7d0d90bf949a80ab4c6fa6
11 Lottie - 4de729d39b102705ae7e5cec6f575324
12 Lottie_Lottie - 5624029cc39cae4926d23ad86a16fec4
13 OrderedCollections - 65102715afc23f47960497675b59e8a1
14 SwiftMessages - df635f71fc370f397c7943196c0f3069
15 _CollectionsUtilities - 285fcd1cd13e3e5e246a86e7e03bb625
16 _RopeModule - 145ab29585d0c66ba147747e3319ba6f
17 Total time taken: 1.533s
```

```
1 oiu@kalkulator oiu % tuist graph
2 // опционально: -f json
```



Бандлы сторонних проектов

А точнее — их отсутствие



Бандлы сторонних проектов

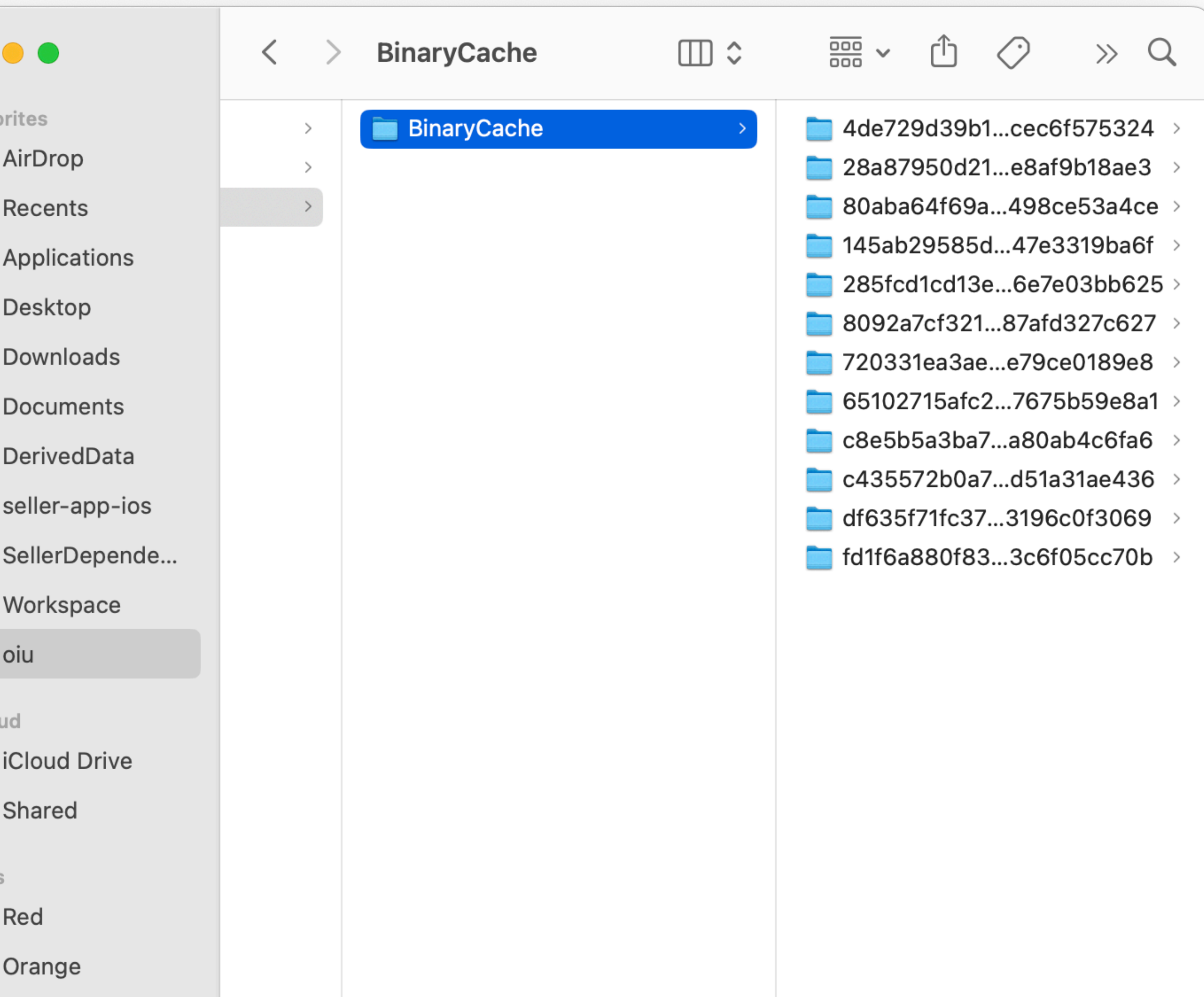
А точнее — их отсутствие



```
1 oiu@kalkulator .internal % tuist cache --print-hashes
2 Loading and constructing the graph
3 It might take a while if the cache is empty
4 Alamofire - c435572b0a7ad22c114f8d51a31ae436
5 AsyncAlgorithms - 720331ea3ae0fe949c993e79ce0189e8
6 BitCollections - 28a87950d21c2bd32a186e8af9b18ae3
7 Collections - 8092a7cf321e5baa360287afd327c627
8 DequeModule - 80aba64f69a2fabe9f88b498ce53a4ce
9 HashTreeCollections - fd1f6a880f83254e0eef63c6f05cc70b
10 HeapModule - c8e5b5a3ba7d0d90bf949a80ab4c6fa6
11 Lottie - 4de729d39b102705ae7e5cec6f575324
12 Lottie_Lottie - 5624029cc39cae4926d23ad86a16fec4
13 OrderedCollections - 65102715atc23t47960497675b59e8a1
14 SwiftMessages - df635f71fc370f397c7943196c0f3069
15 _CollectionsUtilities - 285fcd1cd13e3e5e246a86e7e03bb625
16 _RopeModule - 145ab29585d0c66ba147747e3319ba6f
17 Total time taken: 1.533s
```

Бандлы сторонних проектов

А точнее — их отсутствие



Бандлы сторонних проектов

А точнее — их отсутствие

- 1 Изначальный коммит →
- 2 Заводим “пакет с пакетами” →
- 3 Обвязка Tuist`а — собираем фреймворки →
- 4 Скрипты генерации →
- 5 Хэширование версий зависимостей →
- 6 Бандлы SPM-зависимостей →**
- 7 Генерация проекта исходниками →



 @oiuhr

Артефакты линковки

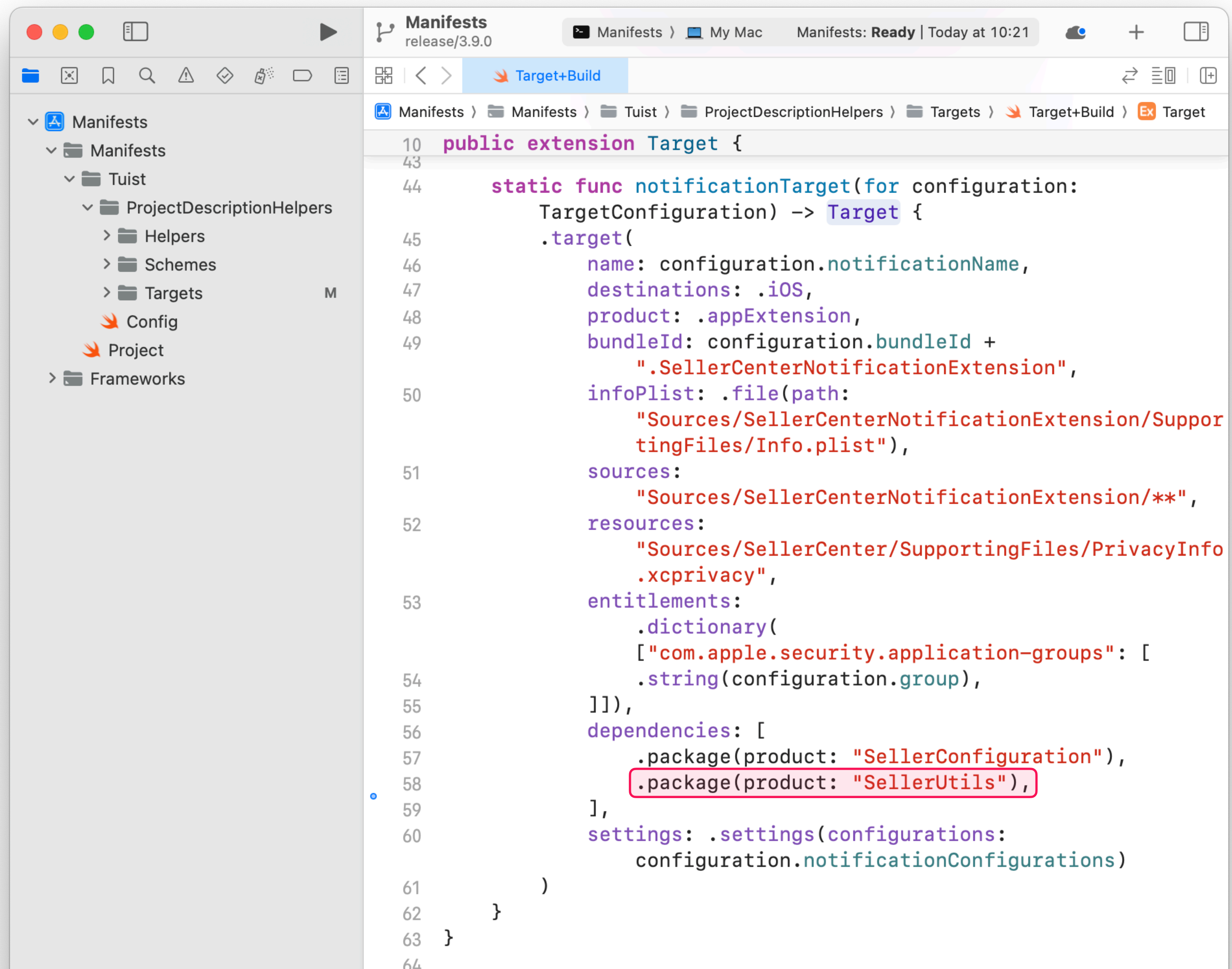
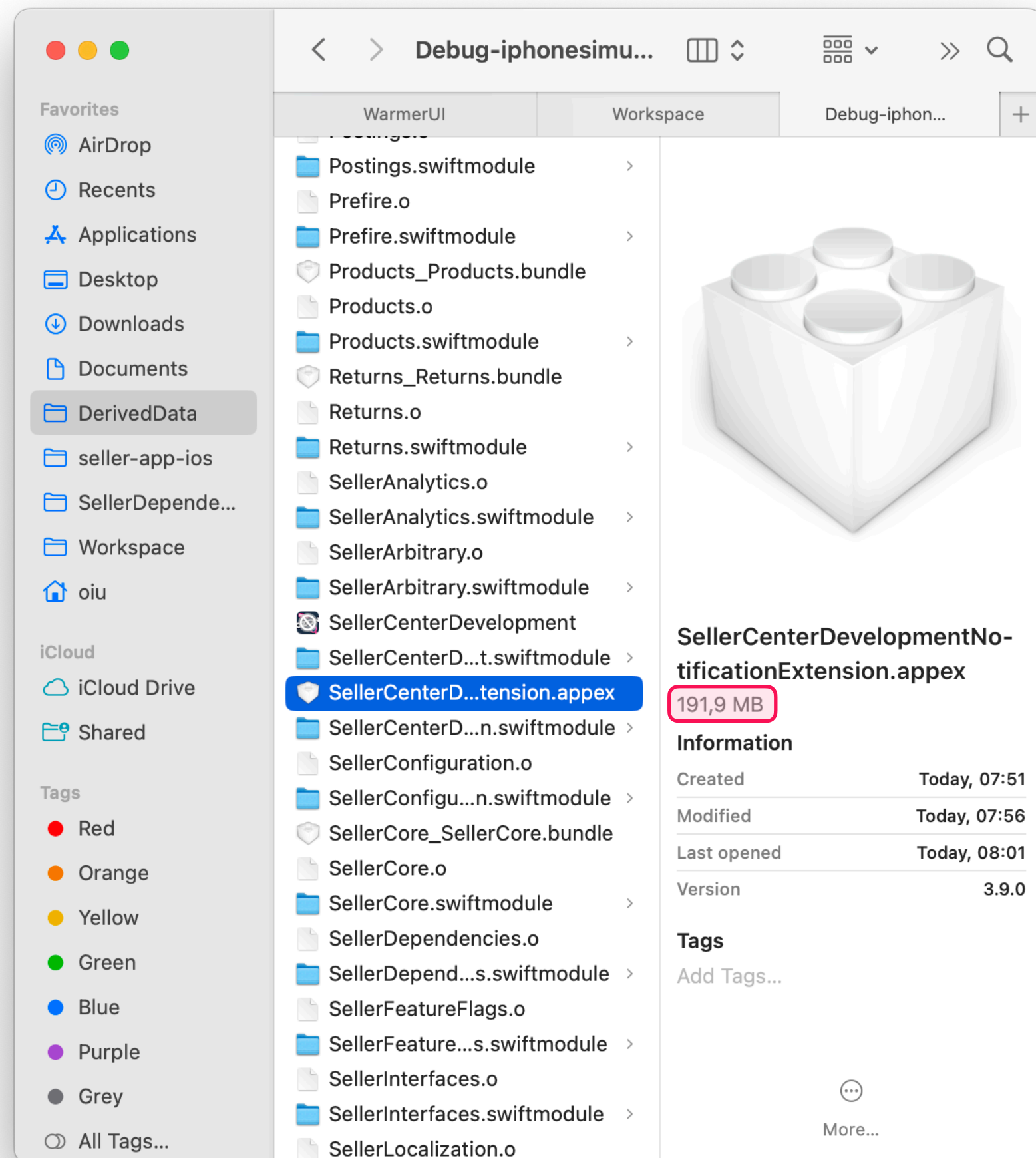
На этот раз действительно без костылей

```
_ = UIApplicationMain(  
    CommandLine.argc,  
    UnsafeMutableRawPointer(C  
        .bindMemory(to: UnsafeMutablePointer?.self, capacity: Int(CommandLine.argc)),  
    nil,  
    appDelegateClass
```

Thread 1: Fatal error: NSArray element failed to match the Swift Array Element type
Expected YMKsuggestItem but found YMKsuggestItem

Артефакты линковки

На этот раз действительно без костылей

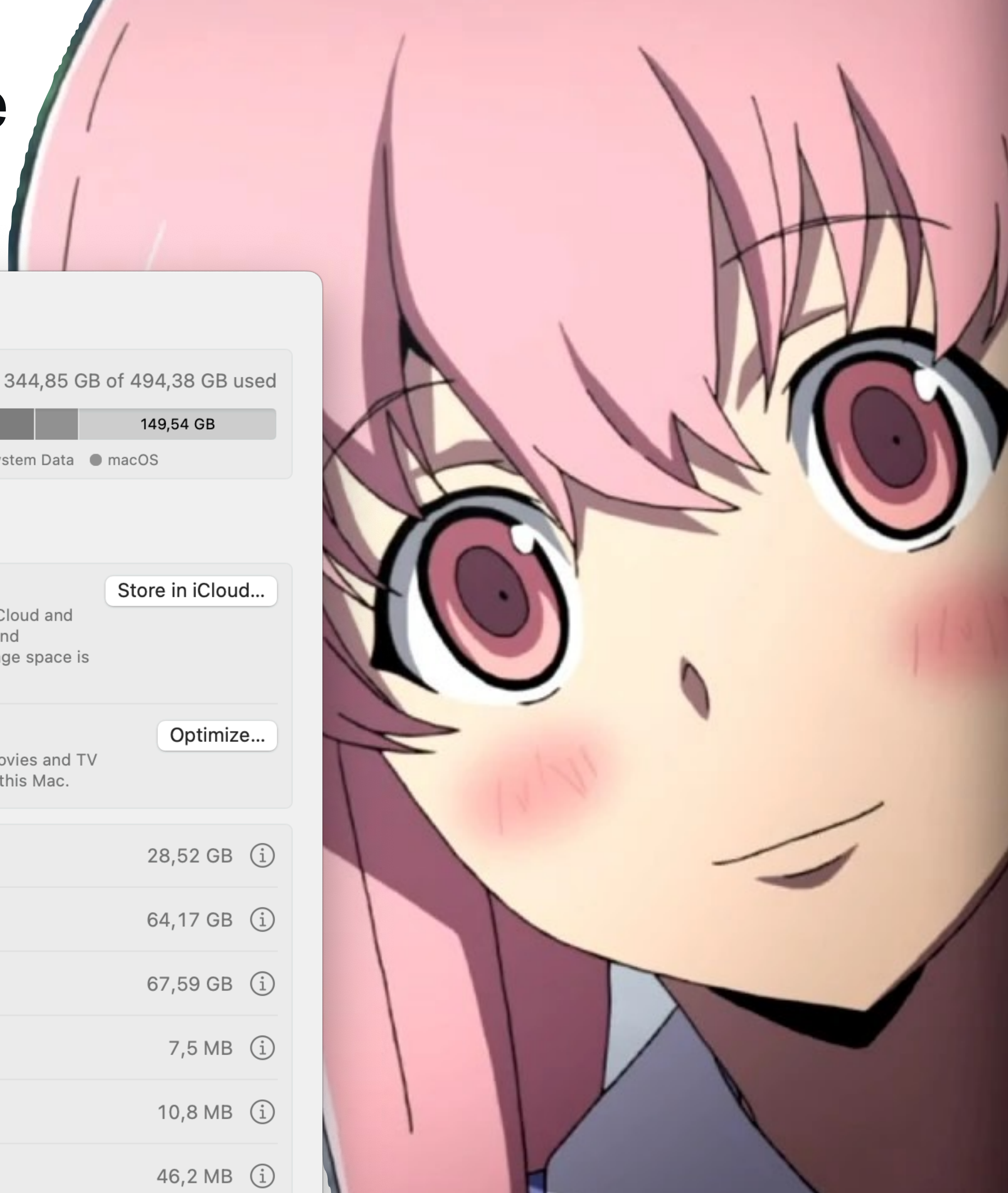


Взрыв на макаронной фабрике

Место на машинках CI

The screenshot shows the macOS Storage settings window. At the top, it indicates that 344,85 GB of 494,38 GB is used on the Macintosh HD. A progress bar shows the usage, with 149,54 GB of space remaining. Below this, there are recommendations to 'Store in iCloud' and 'Optimize Storage'. At the bottom, a list of storage categories is shown with their respective sizes.

Category	Size
Macintosh HD	344,85 GB of 494,38 GB used
Documents	67,59 GB
Developer	64,17 GB
Applications	28,52 GB
iCloud Drive	7,5 MB
Mail	10,8 MB
Photos	46,2 MB



Генерация проекта исходниками

На семь бед — один ответ

```
1 f "SPM_WARMER_GENERATE_WITH_SOURCES" in os.environ:
2     print(
3         f'👁️ Поймано значение окружения для генерации из
4         сурсов: используем внутренний манифест',
5         flush=True
6     )
7     subprocess.call(['cp', f'{manifest_path}', './'])
8     exit(0)
```

Генерация проекта исходниками

На семь бед — один ответ

- 1 **Изначальный коммит** →
- 2 **Заводим “пакет с пакетами”** →
- 3 **Обвязка Tuist`а — собираем фреймворки** →
- 4 **Скрипты генерации** →
- 5 **Хэширование версий зависимостей** →
- 6 **Бандлы SPM-зависимостей** →
- 7 **Генерация проекта исходниками** →



 @oiuhr

Зачем всё это?

А главное — какой ценой

➔ Затраты на разработку

➔ Затраты на поддержку

➔ **Целесообразность
написания собственного
решения**



Зачем кастом?

Или почему не проприетарное решение?

Pros



- Кэширование бинарных артефактов из коробки

Cons



- Огромный пласт работ на переезд
- Очень частые обновления
 - Так себе документация
 - Стабильность релизов
- Закрытые (пейволлом) исходники Tuist Cloud



Зачем всё это?

А главное — какой ценой

➔ Затраты на разработку

➔ Затраты на поддержку

➔ **Целесообразность
написания собственного
решения**



09



Всё!

Сегодня ты узнал...

... что лучше бы выбрал кофе вместо прослушки доклада

01

Что билдтайм можно измерять
в граммах (зерна кофе) и часах (сна)



Сегодня ты узнал...

... что лучше бы выбрал кофе вместо прослушки доклада

01

Что билдтайм можно измерять в граммах (зерна кофе) и часах (сна)

02

Что SPM, в общем-то, ничего не умеет



Сегодня ты узнал...

... что лучше бы выбрал кофе вместо прослушки доклада

01

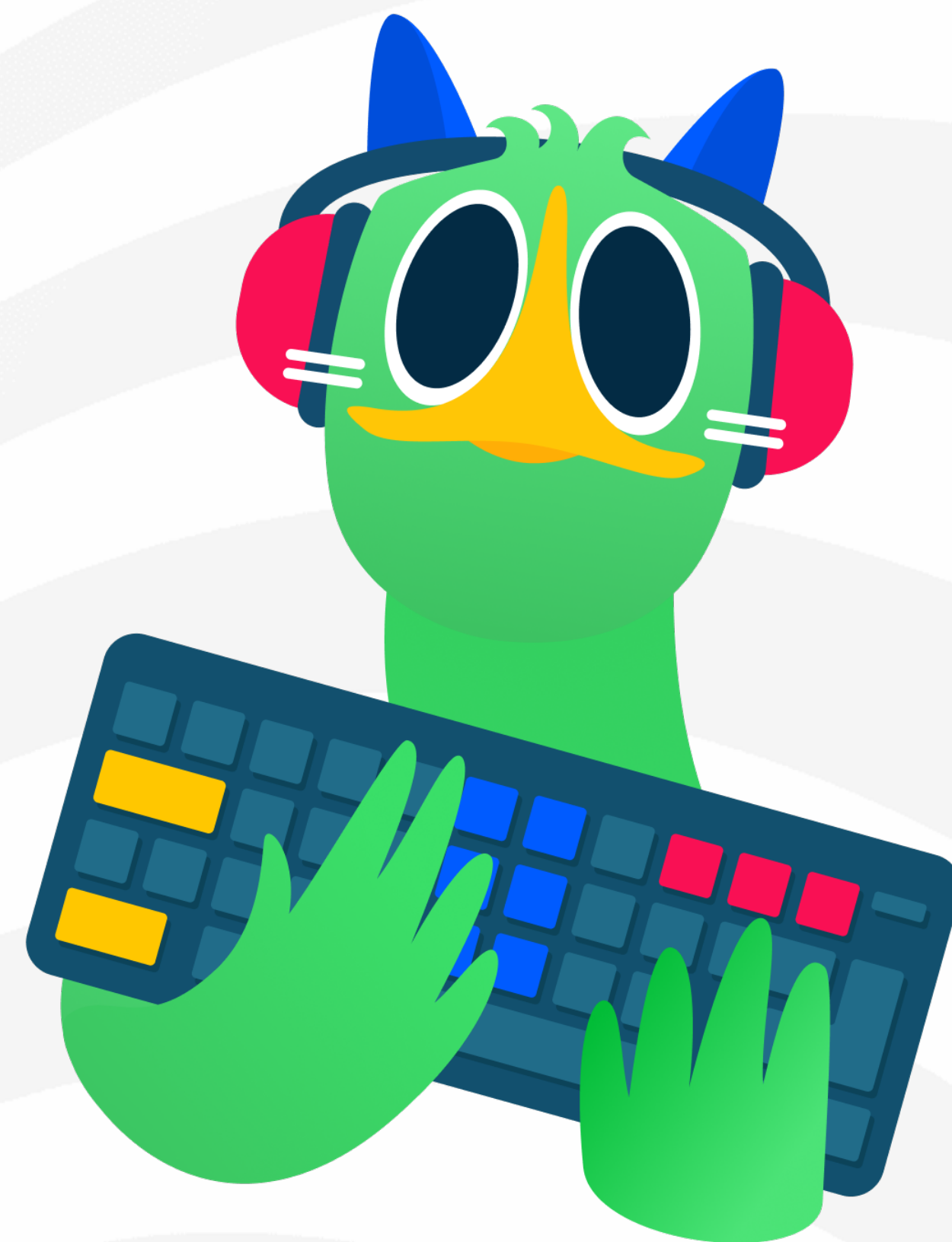
Что билдтайм можно измерять в граммах (зерна кофе) и часах (сна)

02

Что SPM, в общем-то, ничего не умеет

03

Как использовать готовый инструмент в виде костыля



Сегодня ты узнал...

... что лучше бы выбрал кофе вместо прослушки доклада

01

Что билдтайм можно измерять в граммах (зерна кофе) и часах (сна)

02

Что SPM, в общем-то, ничего не умеет

03

Как использовать готовый инструмент в виде костыля

67

Сборку SPM-проекта с использованием сурсов и бинарных артефактов

Сегодня ты узнал...

... что лучше бы выбрал кофе вместо прослушки доклада

01

Что билдтайм можно измерять в граммах (зерна кофе) и часах (сна)

02

Что SPM, в общем-то, ничего не умеет

03

Как использовать готовый инструмент в виде костыля

67

Сборку SPM-проекта с использованием сурсов и бинарных артефактов

68

Варианты подключения бинарных артефактов к фичевым пакетам с помощью собственного инструмента

Сегодня ты узнал...

... что лучше бы выбрал кофе вместо прослушки доклада

01

Что билдтайм можно измерять в граммах (зерна кофе) и часах (сна)

02

Что SPM, в общем-то, ничего не умеет

03

Как использовать готовый инструмент в виде костыля

67

Сборку SPM-проекта с использованием сурсов и бинарных артефактов

68

Варианты подключения бинарных артефактов к фичевым пакетам с помощью собственного инструмента

69

Результаты нашего аттракциона, включая всю боль

Что дальше?

Наш роадмап

- ~~Разнести всё из одного пакета по разным (вес, инкапсуляция)~~ **Сделано за время подготовки!**
- ~~Убить постоянный ребилд кэша~~ **Сделано за время подготовки!**
- **Remote build cache**
- **Кэширование локальных фичевых пакетов**
- **Интерфейс с кнопками**
- **???**





А каков **ваш** роадмап?

Евгений Рыжов, Старший iOS разработчик

 @oiuhr

Спасибо! ❤️

