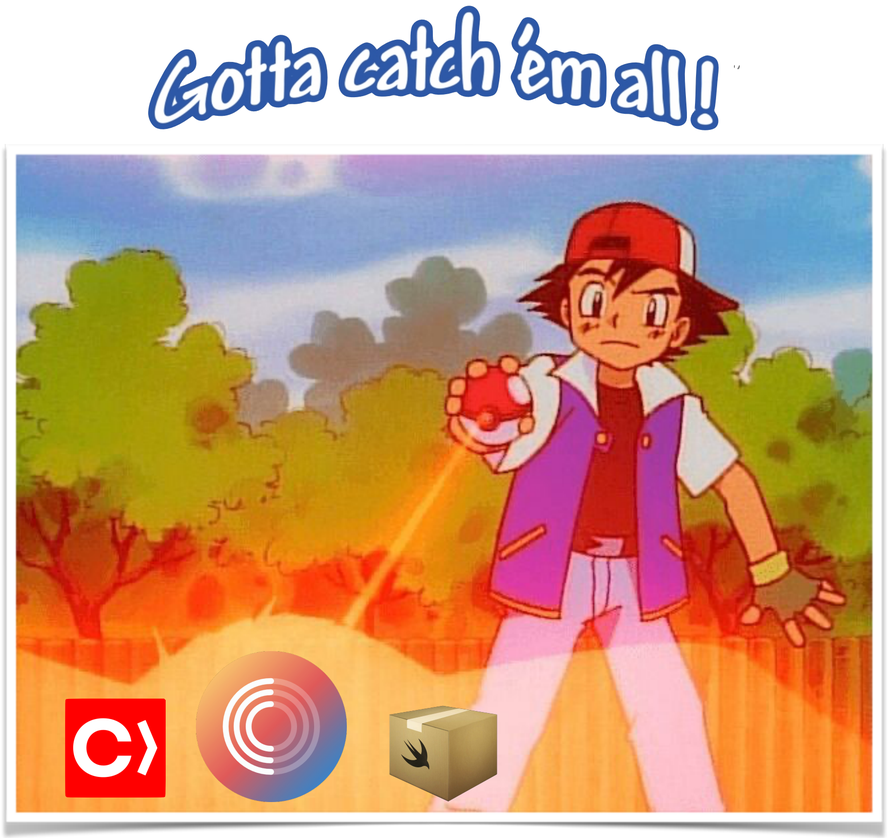
Менеджеры зависимостей – собери их всех



Владислав Фиц

algolia

План

- Введение и постановка задачи

План

- Введение и постановка задачи
- Создание библиотеки

План

- Введение и постановка задачи
- Создание библиотеки
- Версионирование и релизы

План

- Введение и постановка задачи
- Создание библиотеки
- Версионирование и релизы
- Организация многомодульной архитектуры

План

- Введение и постановка задачи
- Создание библиотеки
- Версионирование и релизы
- Организация многомодульной архитектуры
- Нюансы поддержки Linux

План

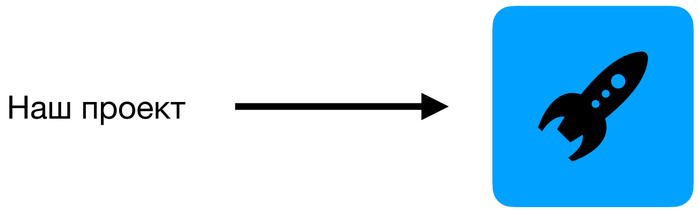
- Введение и постановка задачи
- Создание библиотеки
- Версионирование и релизы
- Организация многомодульной архитектуры
- Нюансы поддержки Linux
- Тестирование

План

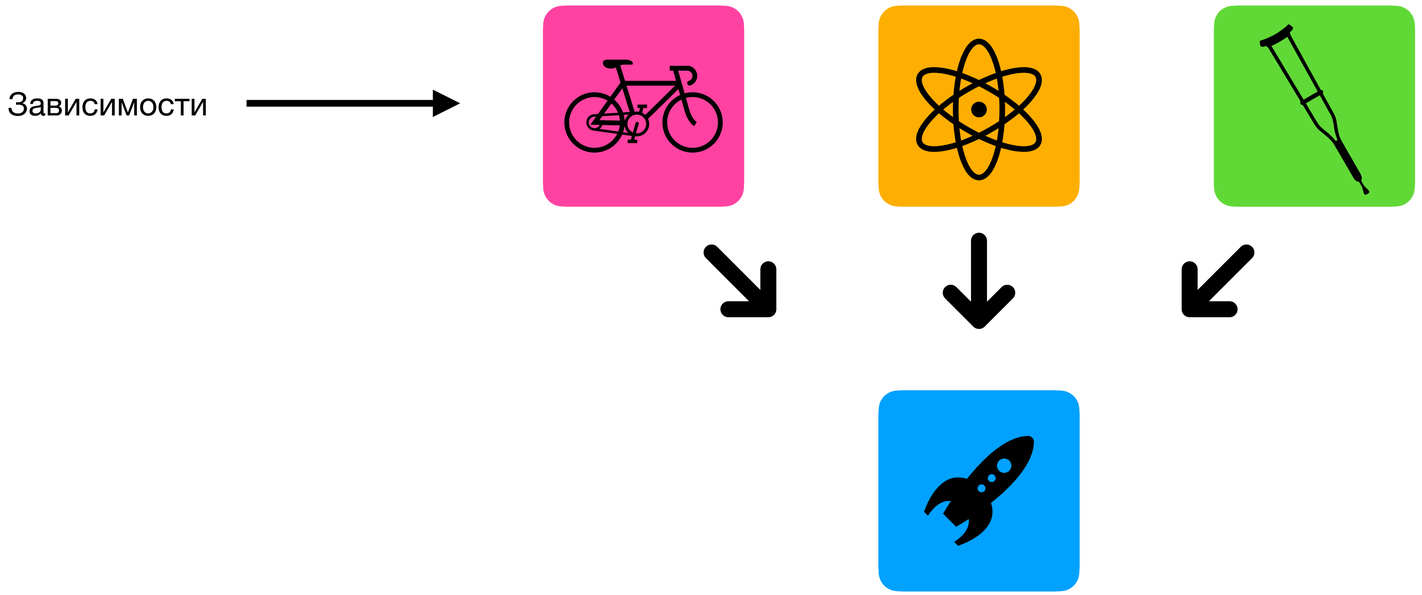
- Введение и постановка задачи
- Создание библиотеки
- Версионирование и релизы
- Организация многомодульной архитектуры
- Нюансы поддержки Linux
- Тестирование
- Заключение

Введение

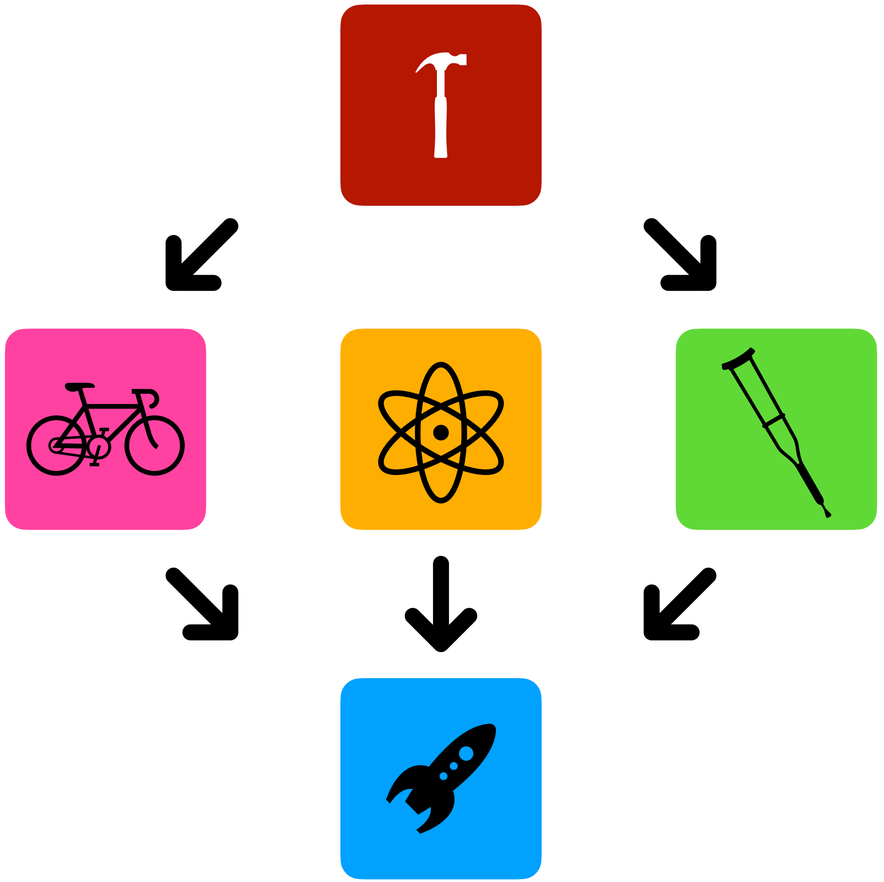
Управление зависимостями



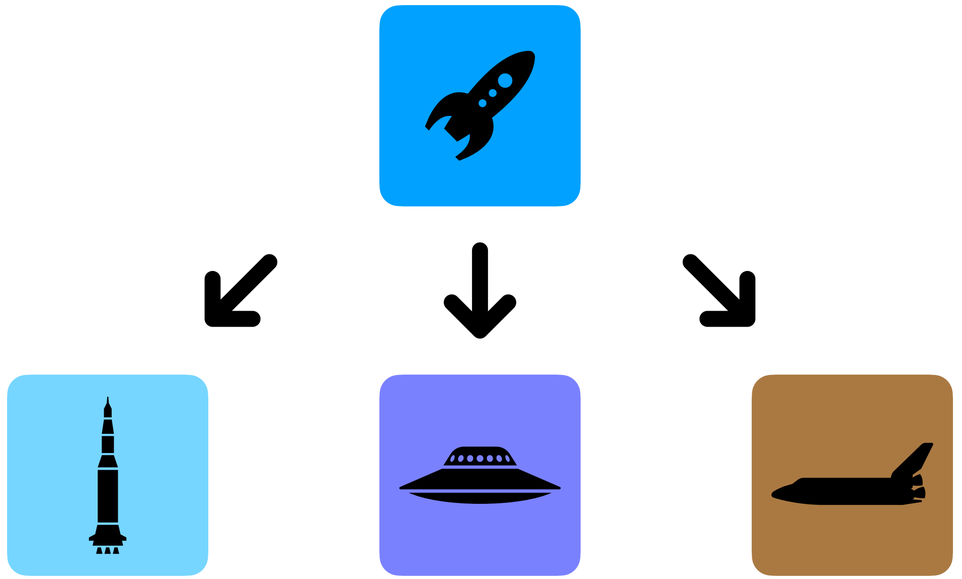
Управление зависимостями



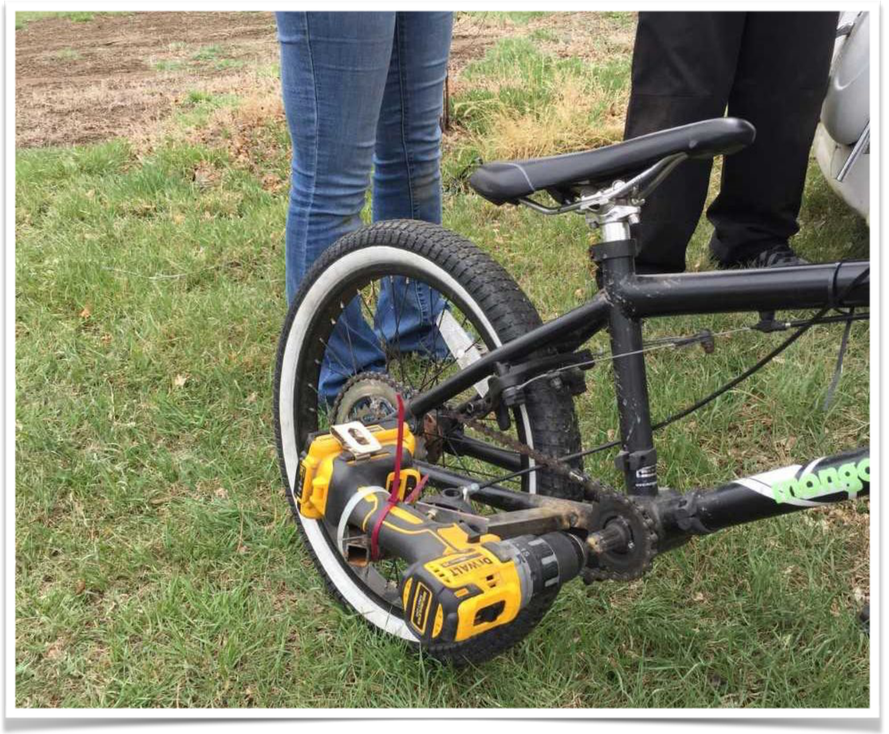
Управление зависимостями



Управление зависимостями



Менеджеры зависимостей

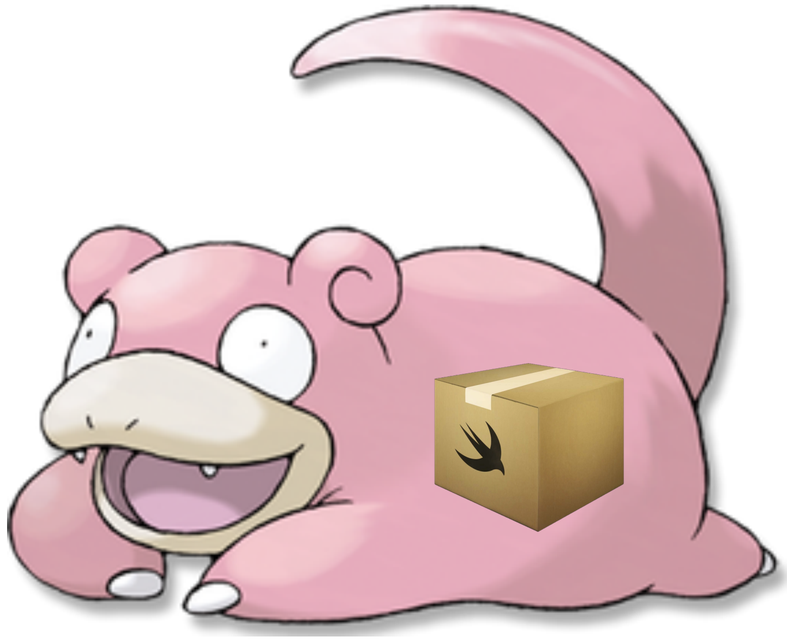


Менеджеры зависимостей

CocoaPods

Carthage

Менеджеры зависимостей



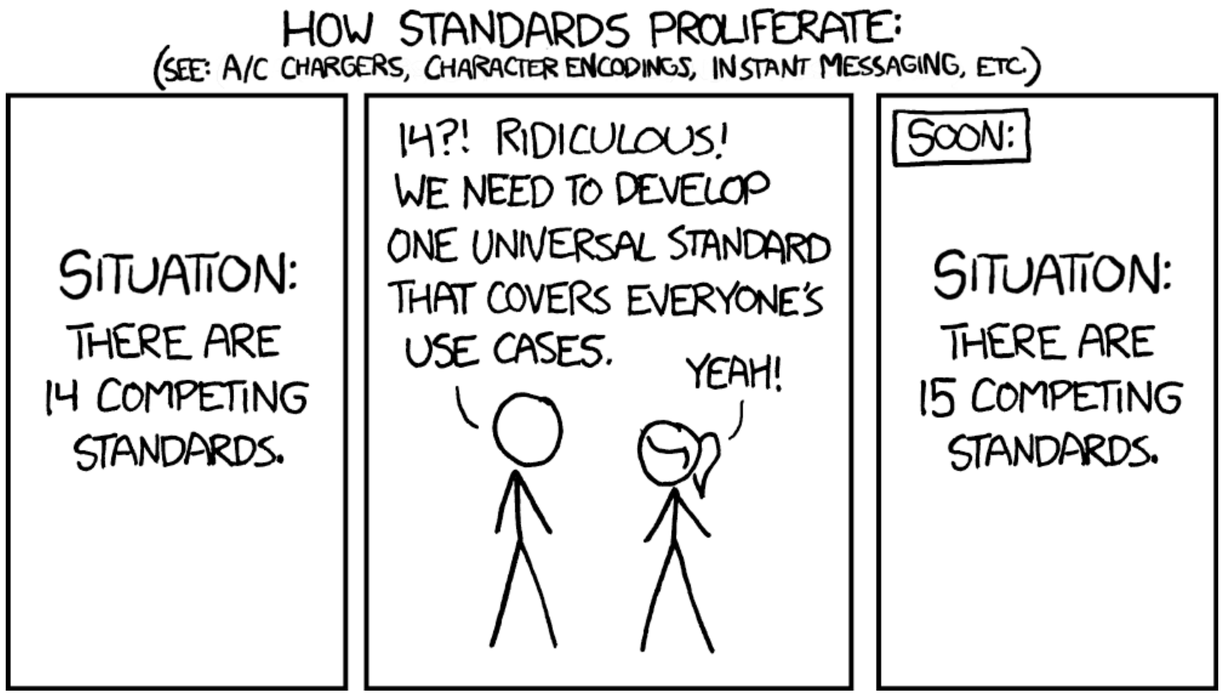
Менеджеры зависимостей

CocoaPods

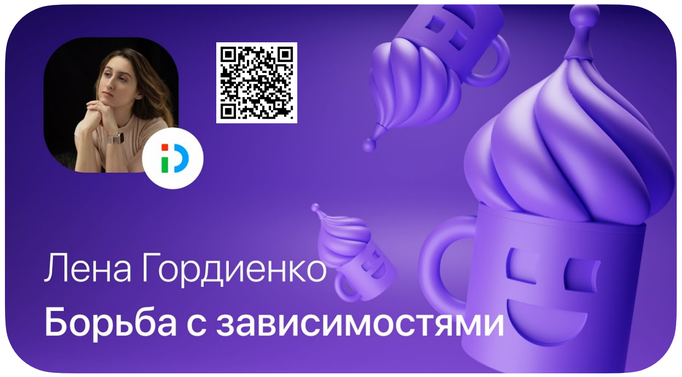
Carthage

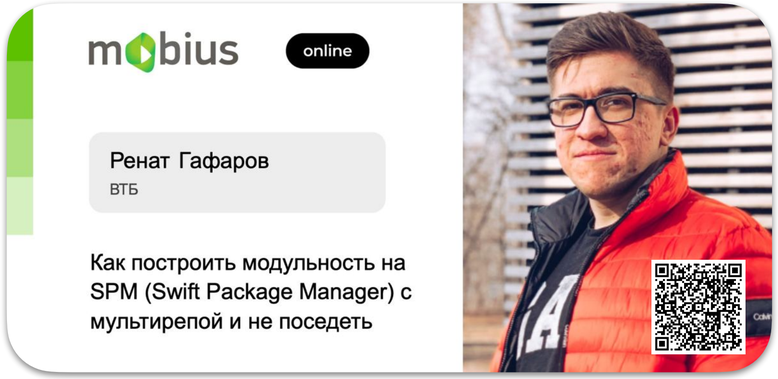
Swift Package Manager

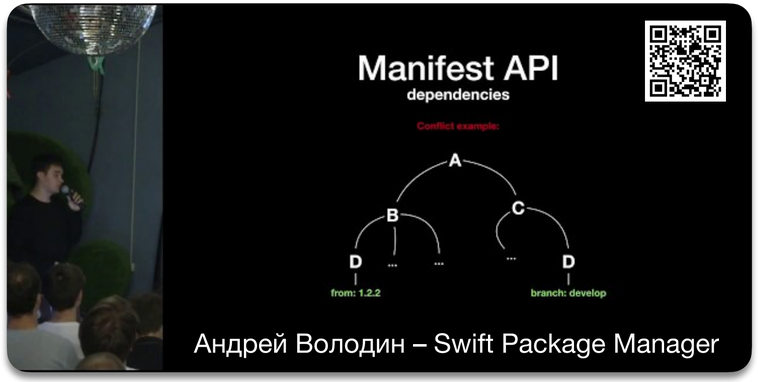
Менеджеры зависимостей

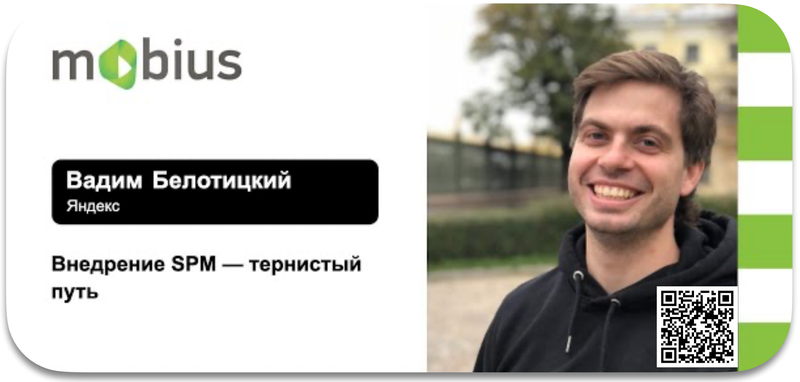


Доклады

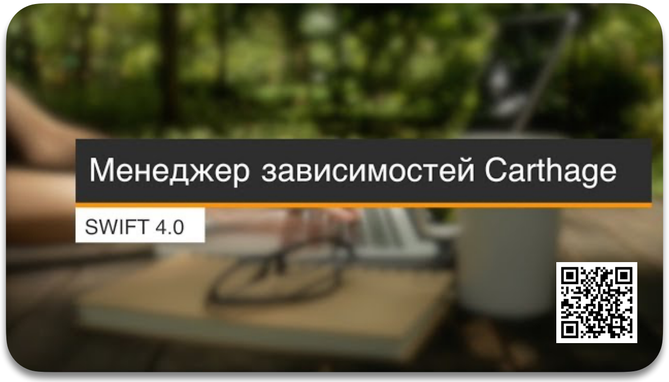

Лена Гордиенко
Борьба с зависимостями


mobius
online
Ренат Гафаров
ВТБ
Как построить модульность на SPM (Swift Package Manager) с мультирепой и не поседеть

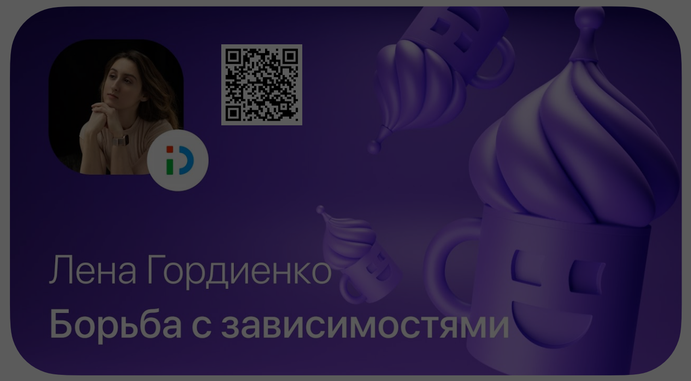

Manifest API
dependencies
Conflict example:
A
B
C
D
D
from: 1.2.2
branch: develop
Андрей Володин – Swift Package Manager

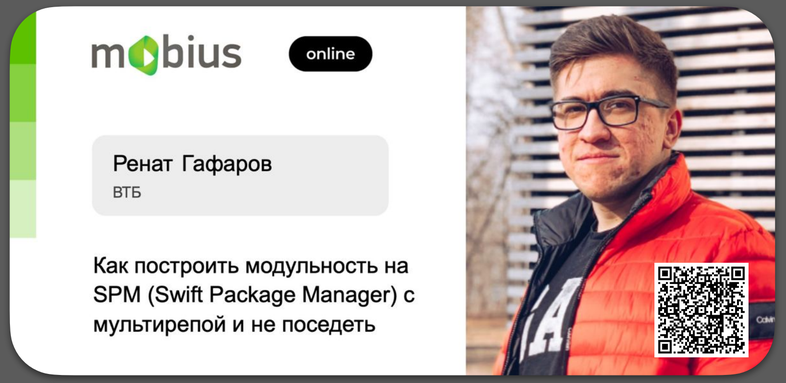

mobius
Вадим Белотицкий
Яндекс
Внедрение SPM — тернистый путь

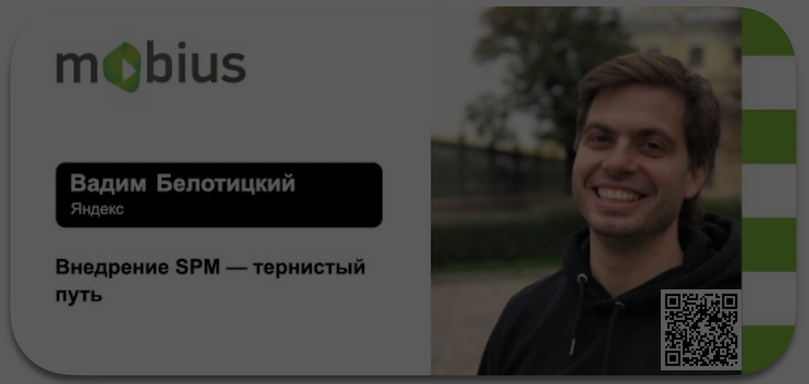

13.09
Модуляризируем свое приложение с помощью собственных Pod'ов
Султан Сейдалин
iOS/Android разработчик, Arbuz.kz
CocoaHeads #2
Kazakhstan

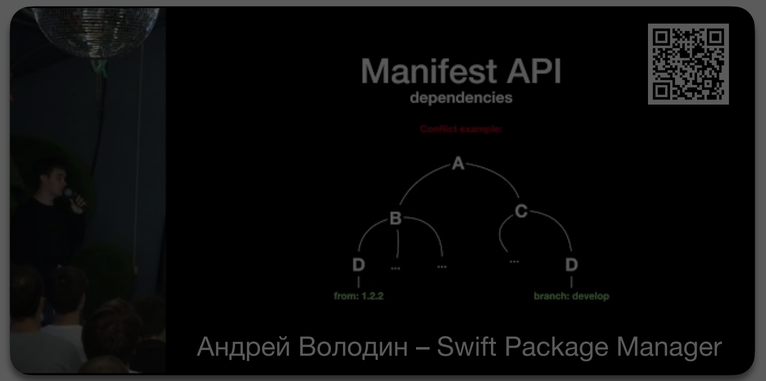

Менеджер зависимостей Carthage
SWIFT 4.0

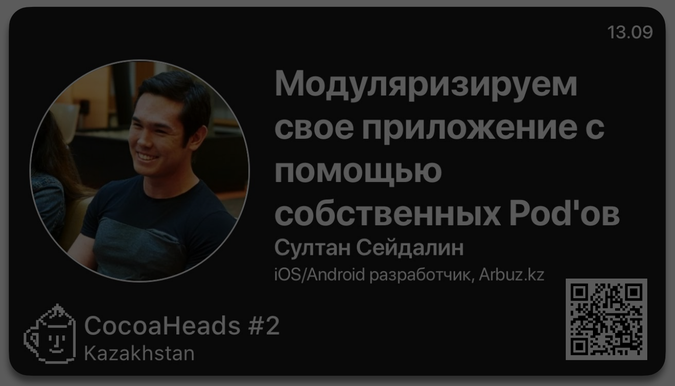
Доклады

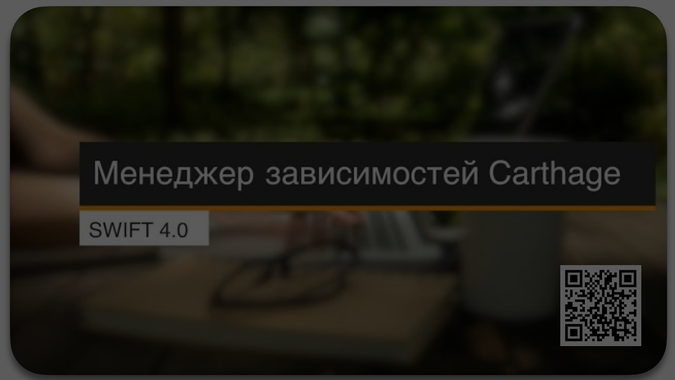

Лена Гордиенко
Борьба с зависимостями


mobius
online
Ренат Гафаров
ВТБ
Как построить модульность на SPM (Swift Package Manager) с мультирепой и не поседеть


mobius
Вадим Белотицкий
Яндекс
Внедрение SPM — тернистый путь

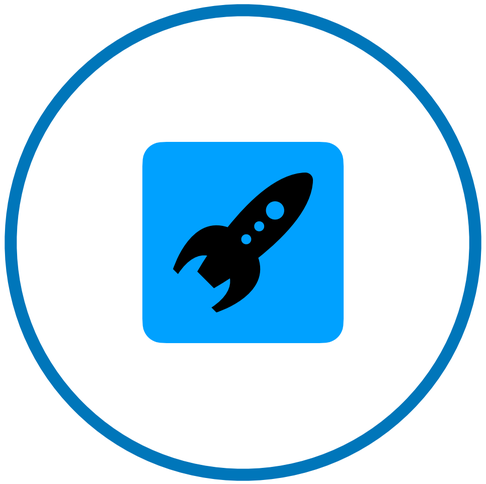

Manifest API
dependencies
A
B
C
D
D
from: 1.2.2
branch: develop
Андрей Володин – Swift Package Manager


13.09
Модуляризируем свое приложение с помощью собственных Pod'ов
Султан Сейдалин
iOS/Android разработчик, Arbuz.kz
CocoaHeads #2
Kazakhstan

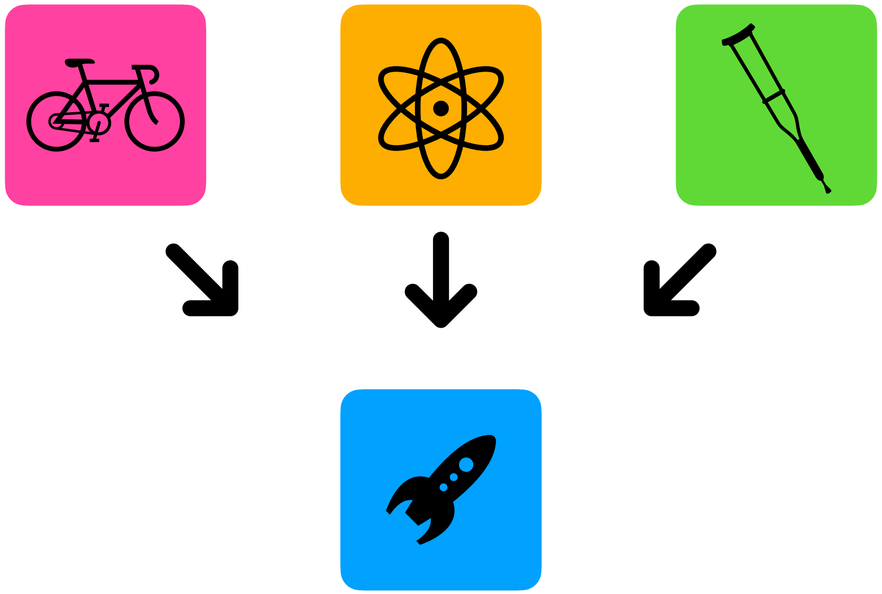

Менеджер зависимостей Carthage
SWIFT 4.0

Менеджеры зависимостей

Зависимости библиотеки



Зависимости библиотеки



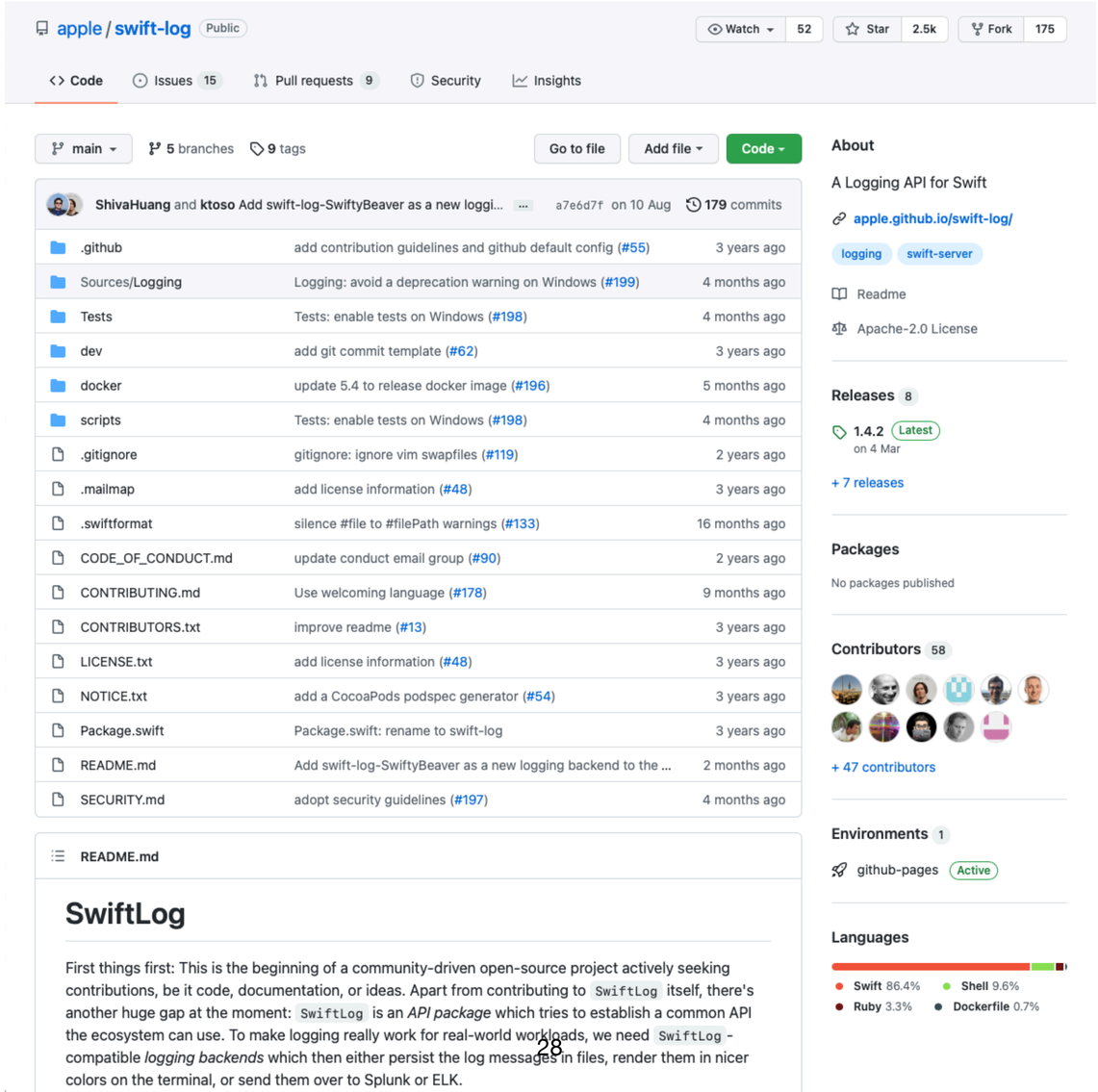
Зависимости библиотеки

Зависимости библиотеки

Зависимости библиотеки

Зависимости библиотеки

SwiftLog



apple / swift-log Public

Watch 52 | Star 2.5k | Fork 175

Code | Issues 15 | Pull requests 9 | Security | Insights

main | 5 branches | 9 tags | Go to file | Add file | Code

ShivaHuang and ktoso Add swift-log-SwiftyBeaver as a new loggi... a7e6d7f on 10 Aug 179 commits

.github	add contribution guidelines and github default config (#55)	3 years ago
Sources/Logging	Logging: avoid a deprecation warning on Windows (#199)	4 months ago
Tests	Tests: enable tests on Windows (#198)	4 months ago
dev	add git commit template (#62)	3 years ago
docker	update 5.4 to release docker image (#196)	5 months ago
scripts	Tests: enable tests on Windows (#198)	4 months ago
.gitignore	gitignore: ignore vim swapfiles (#119)	2 years ago
.mailmap	add license information (#48)	3 years ago
.swiftformat	silence #file to #filePath warnings (#133)	16 months ago
CODE_OF_CONDUCT.md	update conduct email group (#90)	2 years ago
CONTRIBUTING.md	Use welcoming language (#178)	9 months ago
CONTRIBUTORS.txt	improve readme (#13)	3 years ago
LICENSE.txt	add license information (#48)	3 years ago
NOTICE.txt	add a CocoaPods podspec generator (#54)	3 years ago
Package.swift	Package.swift: rename to swift-log	3 years ago
README.md	Add swift-log-SwiftyBeaver as a new logging backend to the ...	2 months ago
SECURITY.md	adopt security guidelines (#197)	4 months ago

README.md

SwiftLog

First things first: This is the beginning of a community-driven open-source project actively seeking contributions, be it code, documentation, or ideas. Apart from contributing to `SwiftLog` itself, there's another huge gap at the moment: `SwiftLog` is an *API package* which tries to establish a common API the ecosystem can use. To make logging really work for real-world workloads, we need `SwiftLog` -compatible *logging backends* which then either persist the log messages in files, render them in nicer colors on the terminal, or send them over to Splunk or ELK.

About

A Logging API for Swift

apple.github.io/swift-log/

logging | swift-server

Readme

Apache-2.0 License

Releases 8

1.4.2 Latest
on 4 Mar

+ 7 releases

Packages

No packages published

Contributors 58

+ 47 contributors

Environments 1

github-pages Active

Languages

Swift 86.4% | Shell 9.6% | Ruby 3.3% | Dockerfile 0.7%

SwiftLog

33 lines (31 sloc) | 918 Bytes — Raw | Blame

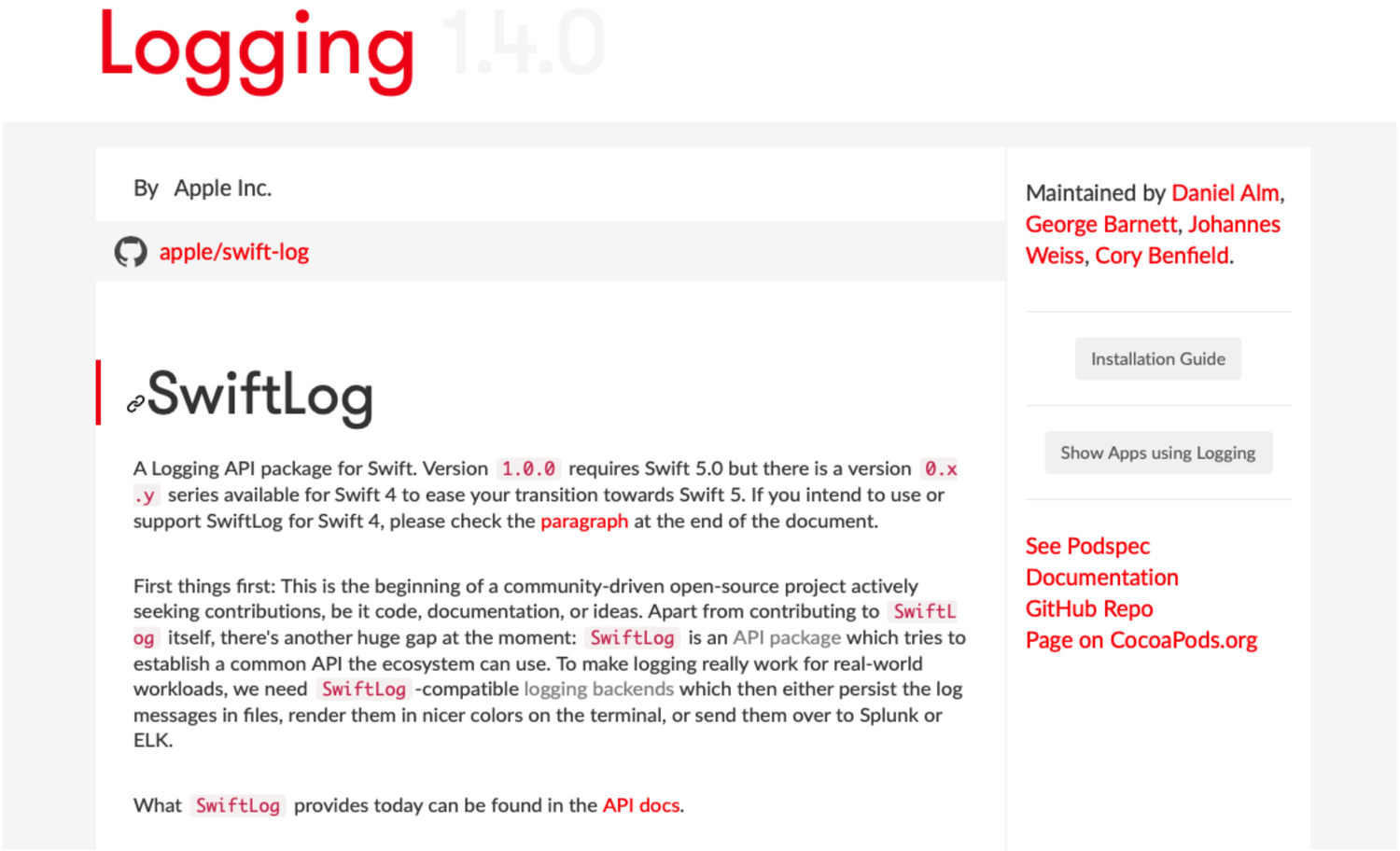
```swift
// swift-tools-version:5.0
//===----------------------------------------------------------------------===//
//
// This source file is part of the Swift Logging API open source project
//
// Copyright (c) 2018-2019 Apple Inc. and the Swift Logging API project authors
// Licensed under Apache License v2.0
//
// See LICENSE.txt for license information
// See CONTRIBUTORS.txt for the list of Swift Logging API project authors
//
// SPDX-License-Identifier: Apache-2.0
//
//===----------------------------------------------------------------------===//

import PackageDescription

let package = Package(
    name: "swift-log",
    products: [
        .library(name: "Logging", targets: ["Logging"]),
    ],
    targets: [
        .target(
            name: "Logging",
            dependencies: []
        ),
        .testTarget(
            name: "LoggingTests",
            dependencies: ["Logging"]
        ),
    ]
)
```

SwiftLog

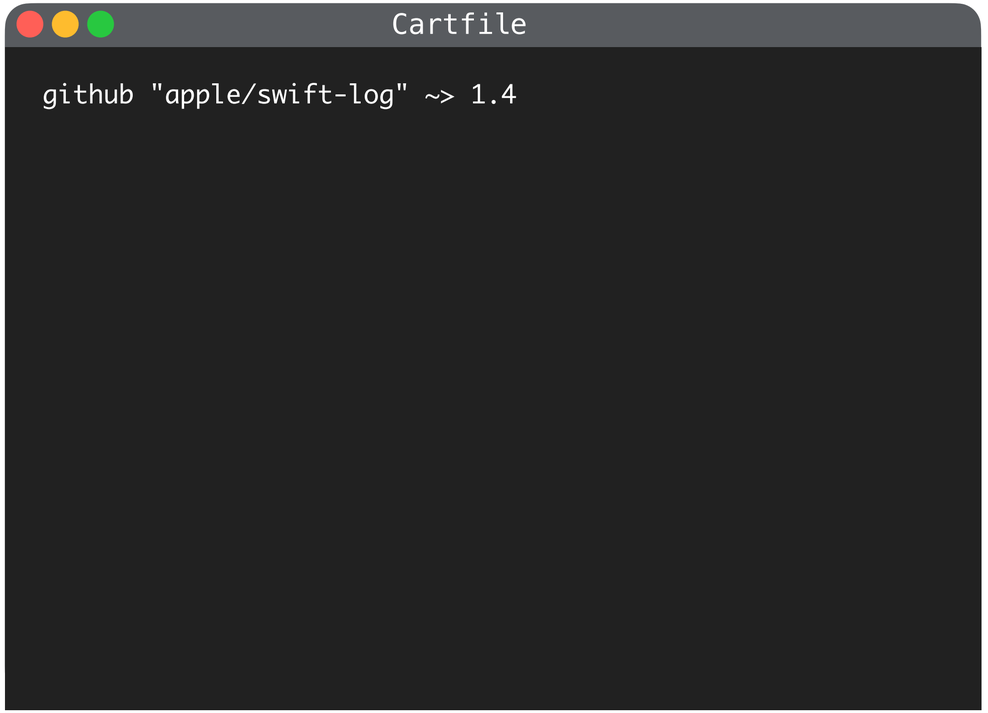
33 lines (31 sloc) | 918 Bytes — Raw | Blame

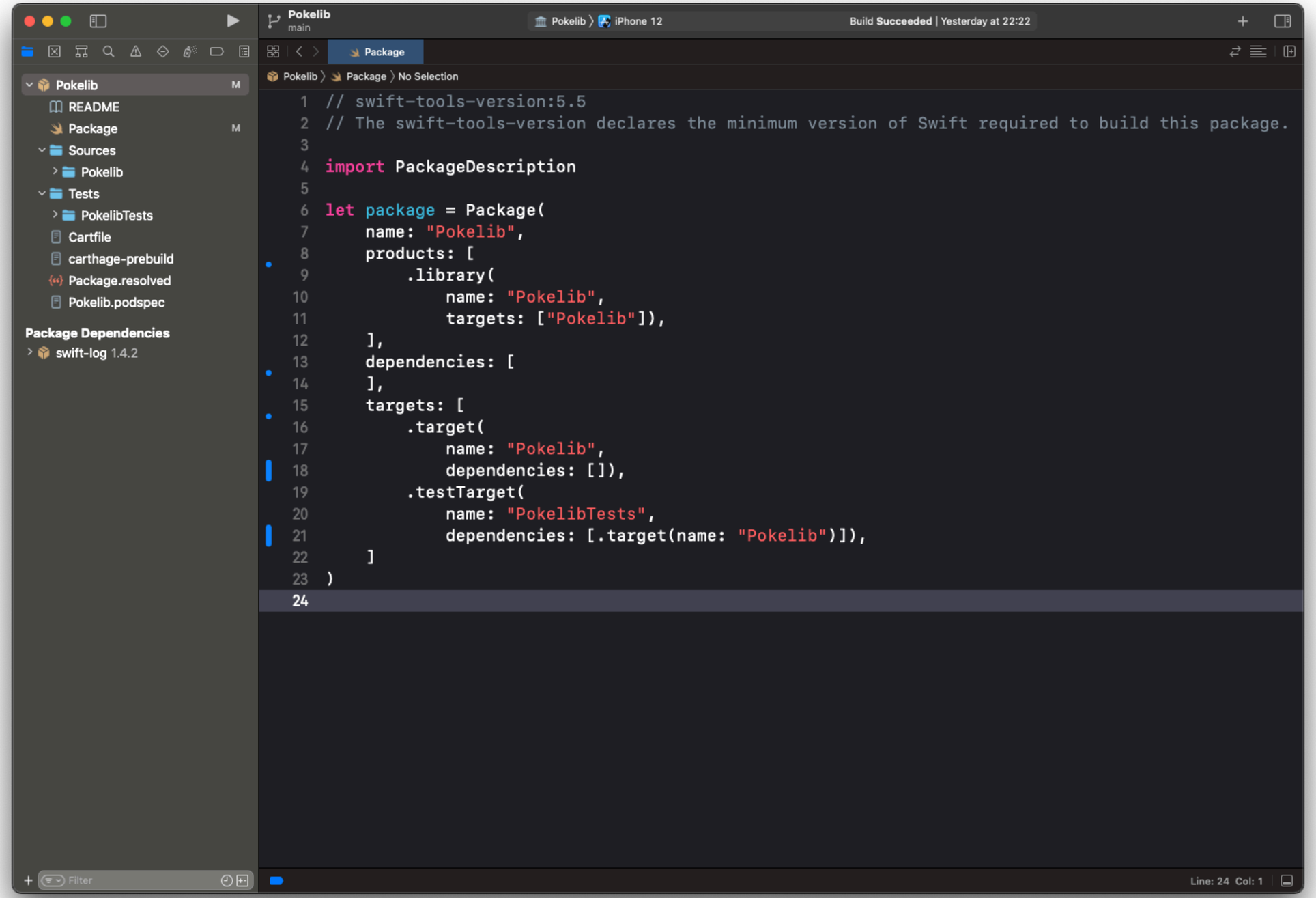
```swift
// swift-tools-version:5.0
//===----------------------------------------------------------------------===//
//
// This source file is part of the Swift Logging API open source project
//
// Copyright (c) 2018-2019 Apple Inc. and the Swift Logging API project authors
// Licensed under Apache License v2.0
//
// See LICENSE.txt for license information
// See CONTRIBUTORS.txt for the list of Swift Logging API project authors
//
// SPDX-License-Identifier: Apache-2.0
//
//===----------------------------------------------------------------------===//

import PackageDescription

let package = Package(
    name: "swift-log",
    products: [
        .library(name: "Logging", targets: ["Logging"]),
    ],
    targets: [
        .target(
            name: "Logging",
            dependencies: []
        ),
        .testTarget(
            name: "LoggingTests",
            dependencies: ["Logging"]
        ),
    ]
)
```

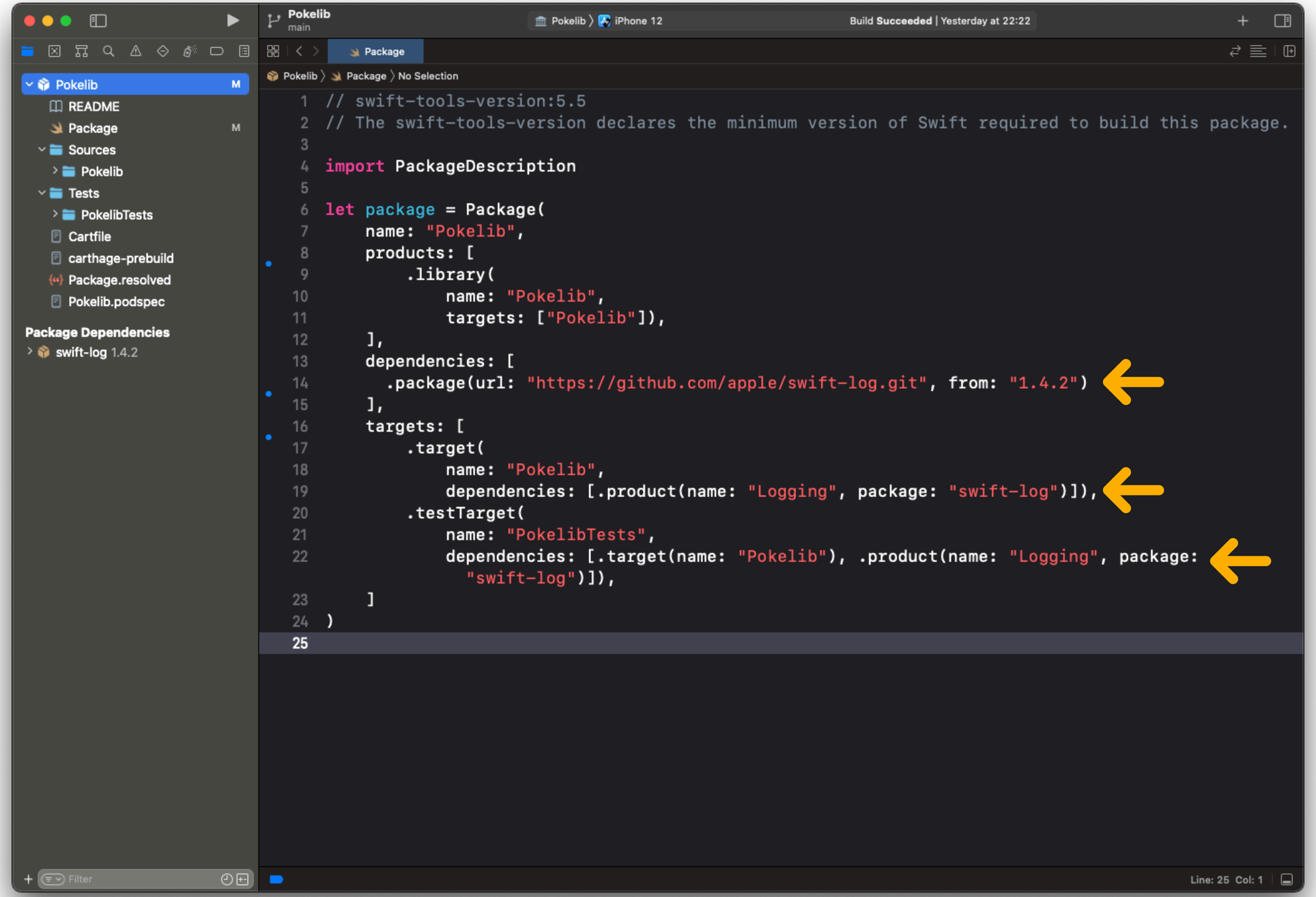
SwiftLog + CocoaPods



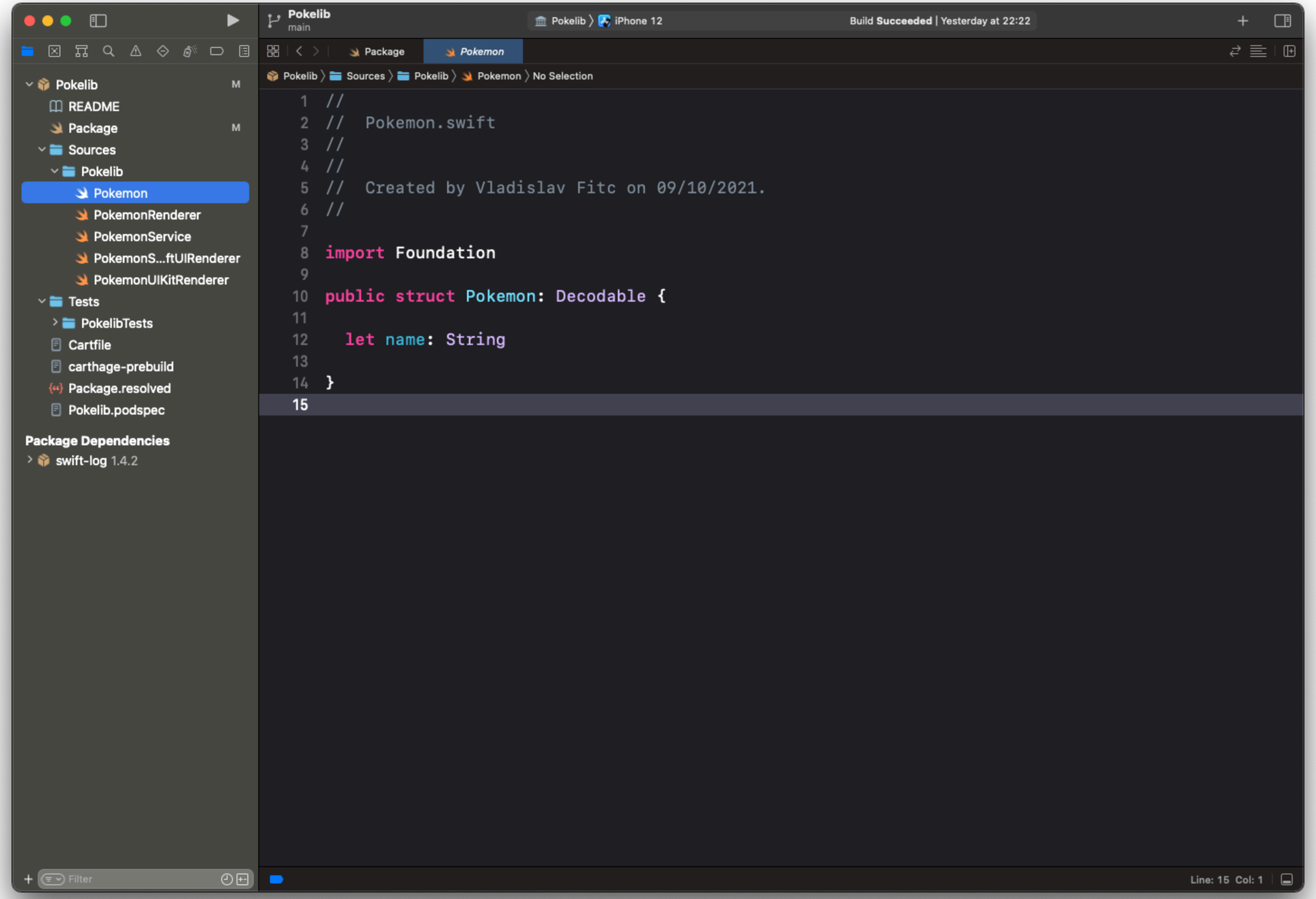
SwiftLog + Carthage

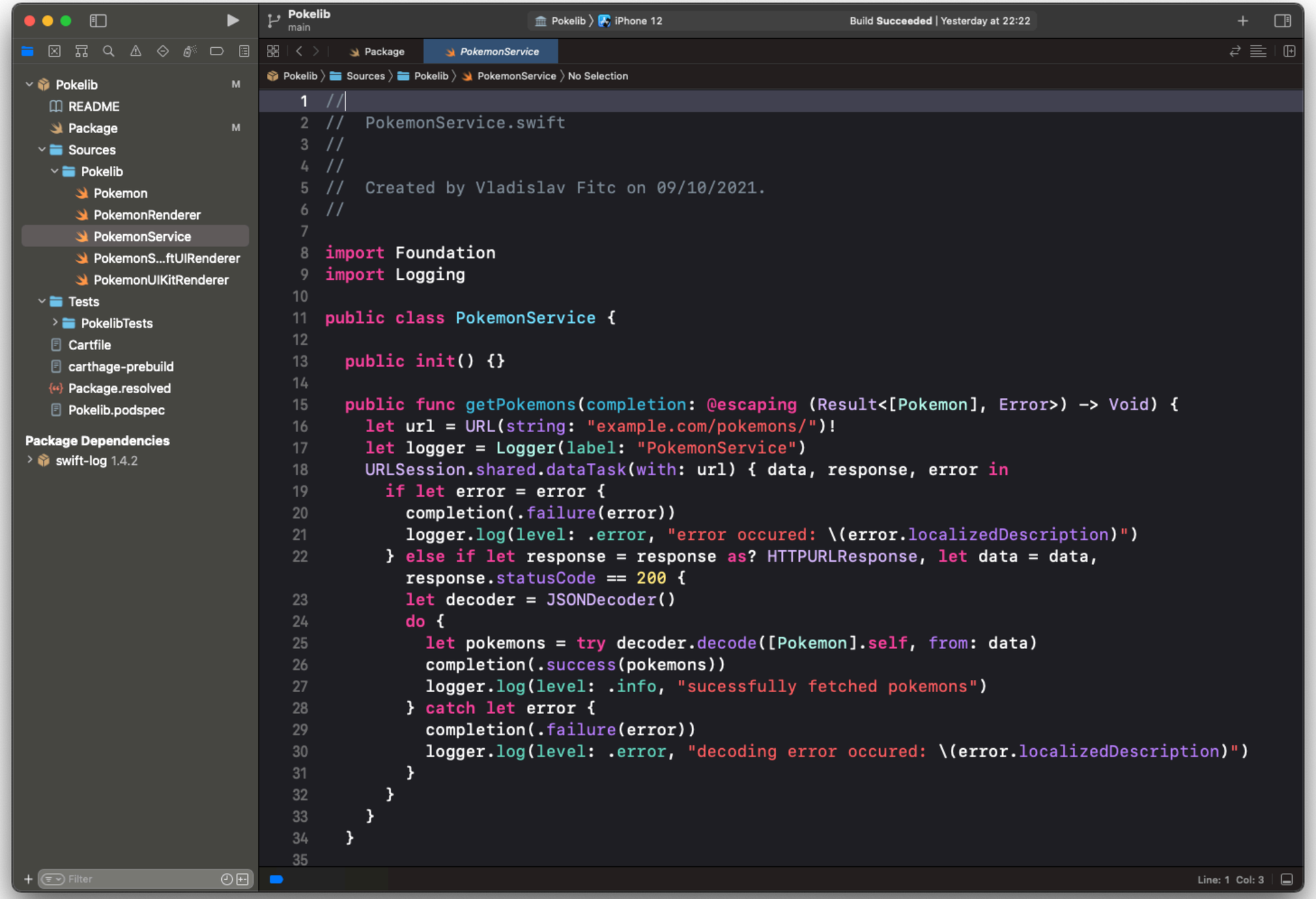



Pokelib
main
Pokelib 〉 iPhone 12
Build Succeeded | Yesterday at 22:22
Package
Pokelib 〉 Package 〉 No Selection
Pokelib M
README
Package M
Sources
Pokelib
Tests
PokelibTests
Cartfile
carthage-prebuild
Package.resolved
Pokelib.podspec
Package Dependencies
swift-log 1.4.2
Filter
1 // swift-tools-version:5.5
2 // The swift-tools-version declares the minimum version of Swift required to build this package.
3
4 import PackageDescription
5
6 let package = Package(
7 name: "Pokelib",
8 products: [
9 .library(
10 name: "Pokelib",
11 targets: ["Pokelib"]),
12],
13 dependencies: [
14],
15 targets: [
16 .target(
17 name: "Pokelib",
18 dependencies: []),
19 .testTarget(
20 name: "PokelibTests",
21 dependencies: [.target(name: "Pokelib")]),
22]
23)
24
Line: 24 Col: 1

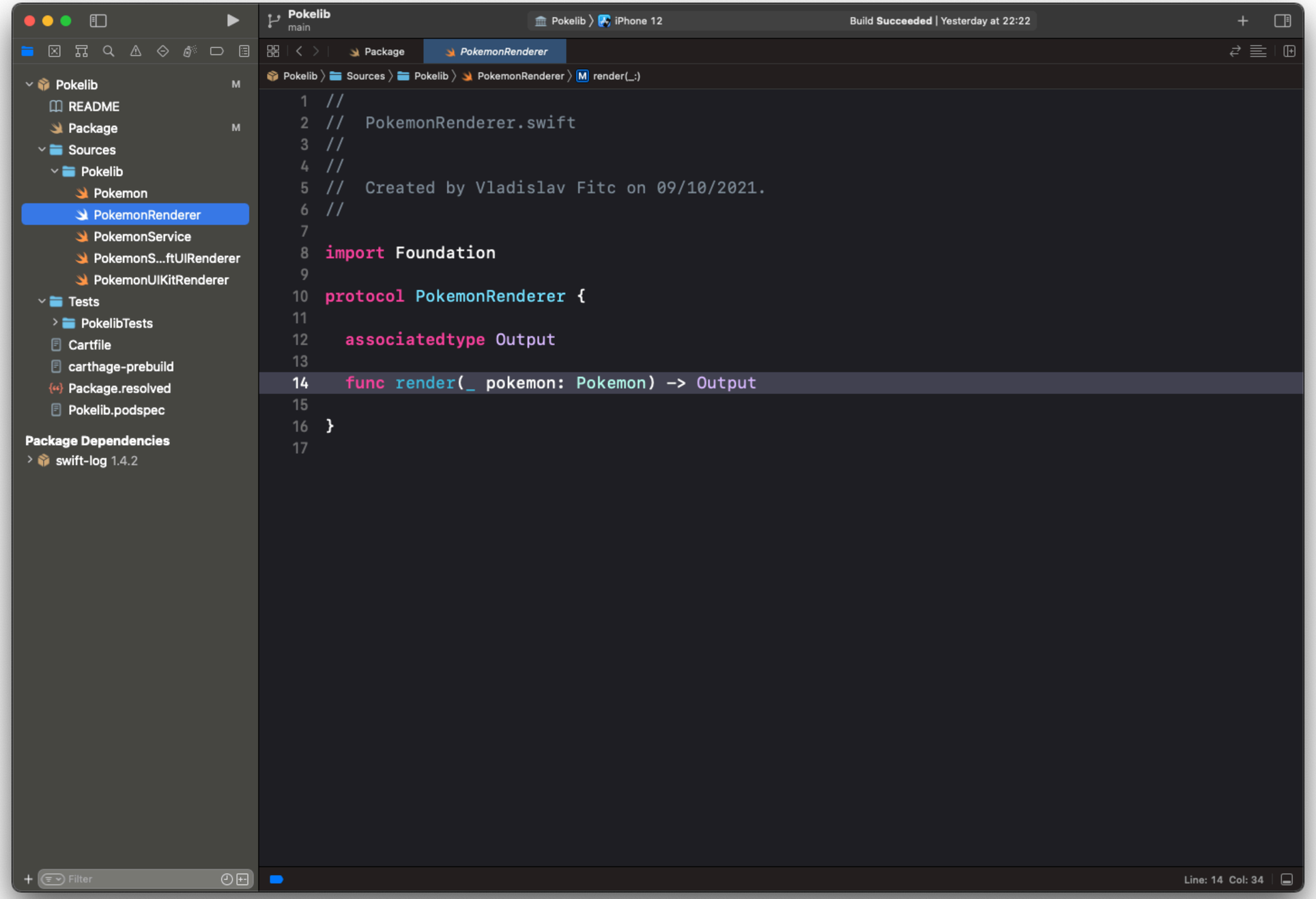


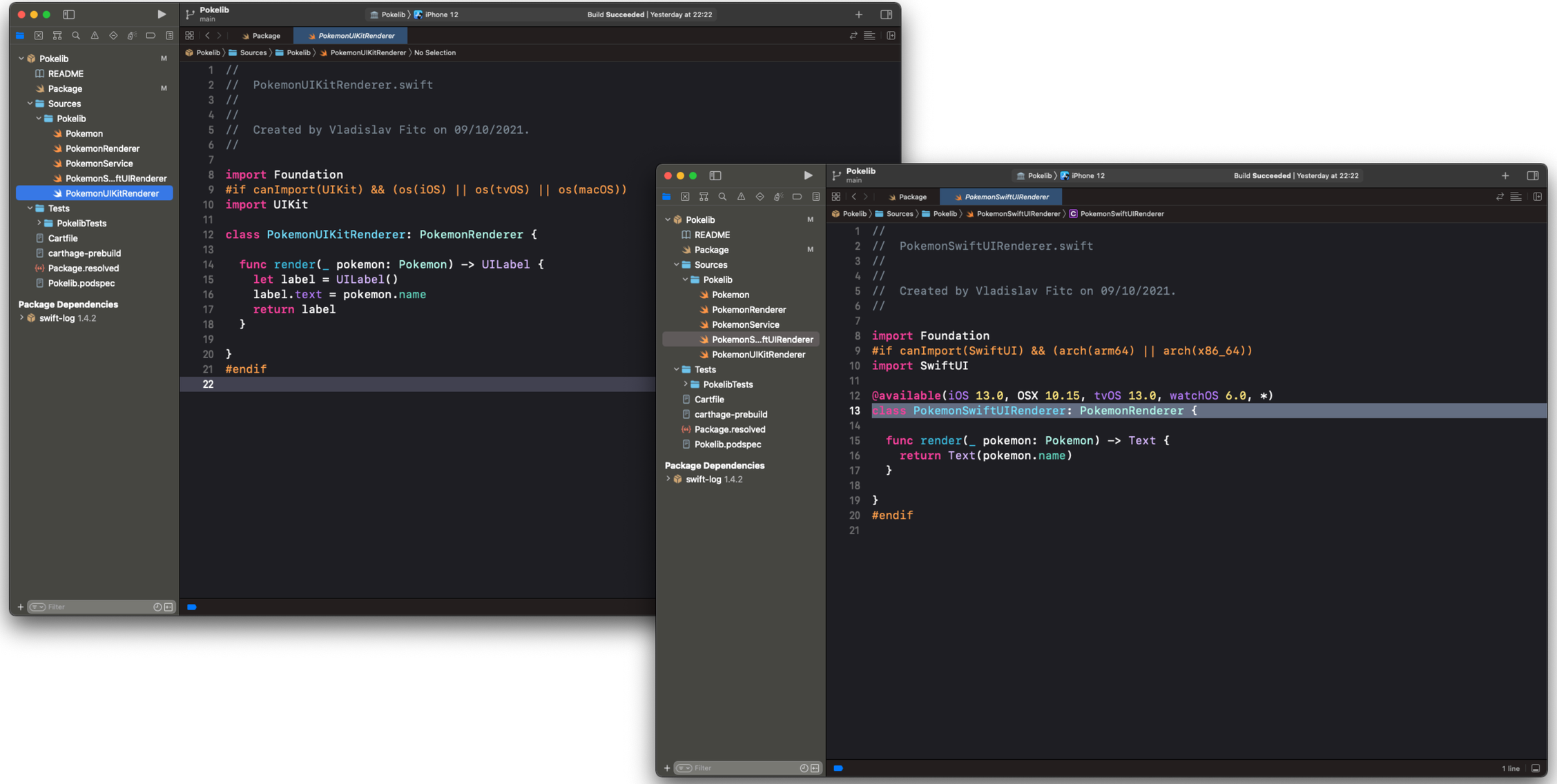

```
1  // swift-tools-version:5.5
2  // The swift-tools-version declares the minimum version of Swift required to build this package.
3
4  import PackageDescription
5
6  let package = Package(
7      name: "Pokelib",
8      products: [
9          .library(
10             name: "Pokelib",
11             targets: ["Pokelib"]),
12     ],
13     dependencies: [
14       .package(url: "https://github.com/apple/swift-log.git", from: "1.4.2")
15     ],
16     targets: [
17         .target(
18             name: "Pokelib",
19             dependencies: [.product(name: "Logging", package: "swift-log")]),
20         .testTarget(
21             name: "PokelibTests",
22             dependencies: [.target(name: "Pokelib"), .product(name: "Logging", package:
                 "swift-log")]),
23     ]
24 )
25
```

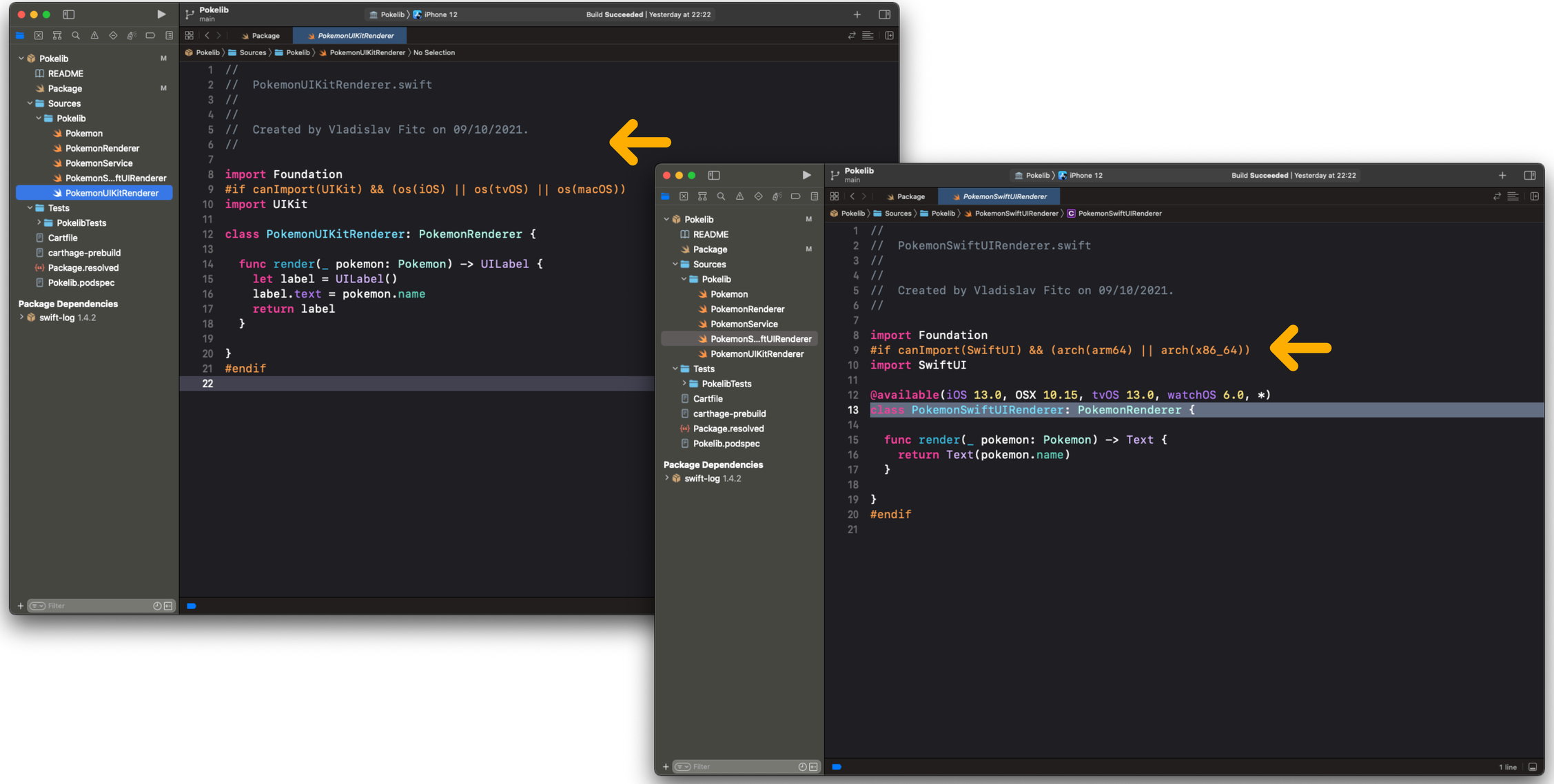

Pokelib
main
Pokelib 〉 iPhone 12
Build Succeeded | Yesterday at 22:22
Package
Pokemon
Pokelib 〉 Sources 〉 Pokelib 〉 Pokemon 〉 No Selection
Pokelib M
README
Package M
Sources
Pokelib
Pokemon
PokemonRenderer
PokemonService
PokemonS...ftUIRenderer
PokemonUIKitRenderer
Tests
PokelibTests
Cartfile
carthage-prebuild
Package.resolved
Pokelib.podspec
Package Dependencies
swift-log 1.4.2
Filter
1 //
2 // Pokemon.swift
3 //
4 //
5 // Created by Vladislav Fitc on 09/10/2021.
6 //
7
8 import Foundation
9
10 public struct Pokemon: Decodable {
11
12 let name: String
13
14 }
15
Line: 15 Col: 1



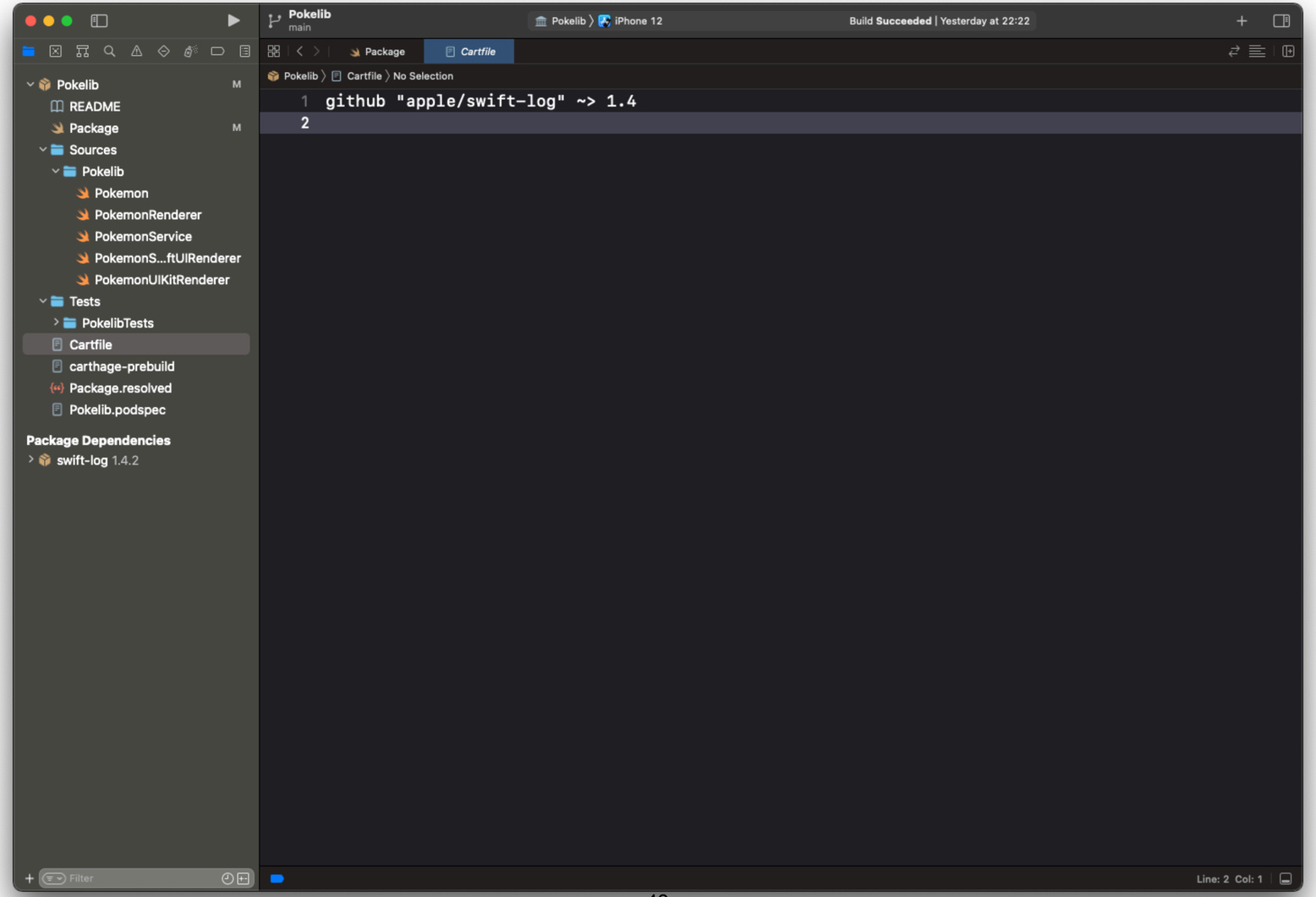

```swift
//
//  PokemonService.swift
//
//
//  Created by Vladislav Fitc on 09/10/2021.
//

import Foundation
import Logging

public class PokemonService {

  public init() {}

  public func getPokemons(completion: @escaping (Result<[Pokemon], Error>) -> Void) {
    let url = URL(string: "example.com/pokemons/")!
    let logger = Logger(label: "PokemonService")
    URLSession.shared.dataTask(with: url) { data, response, error in
      if let error = error {
        completion(.failure(error))
        logger.log(level: .error, "error occured: \(error.localizedDescription)")
      } else if let response = response as? HTTPURLResponse, let data = data,
        response.statusCode == 200 {
        let decoder = JSONDecoder()
        do {
          let pokemons = try decoder.decode([Pokemon].self, from: data)
          completion(.success(pokemons))
          logger.log(level: .info, "sucessfully fetched pokemons")
        } catch let error {
          completion(.failure(error))
          logger.log(level: .error, "decoding error occured: \(error.localizedDescription)")
        }
      }
    }
  }

```

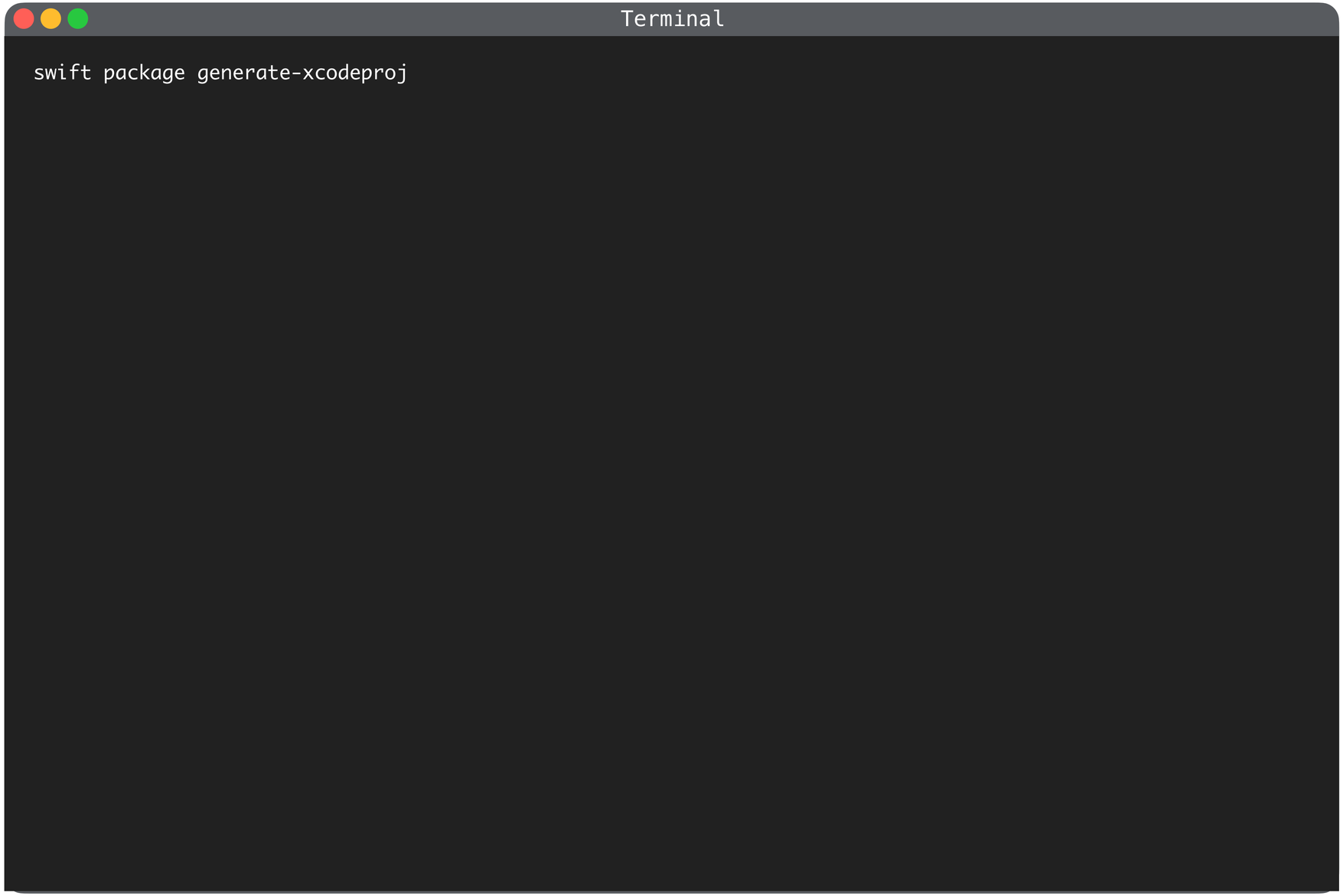

Pokelib
main
Pokelib 〉 iPhone 12
Build Succeeded | Yesterday at 22:22
Package
PokemonRenderer
Pokelib 〉 Sources 〉 Pokelib 〉 PokemonRenderer 〉 render(_:)
Pokelib M
README
Package M
Sources
Pokelib
Pokemon
PokemonRenderer
PokemonService
PokemonS...ftUIRenderer
PokemonUIKitRenderer
Tests
PokelibTests
Cartfile
carthage-prebuild
Package.resolved
Pokelib.podspec
Package Dependencies
swift-log 1.4.2
Filter
1 //
2 // PokemonRenderer.swift
3 //
4 //
5 // Created by Vladislav Fitc on 09/10/2021.
6 //
7
8 import Foundation
9
10 protocol PokemonRenderer {
11
12 associatedtype Output
13
14 func render(_ pokemon: Pokemon) -> Output
15
16 }
17
Line: 14 Col: 34


Pokelib
main
Pokelib 〉 iPhone 12
Build Succeeded | Yesterday at 22:22
Package
PokemonUIKitRenderer
Pokelib 〉 Sources 〉 Pokelib 〉 PokemonUIKitRenderer 〉 No Selection
//
// PokemonUIKitRenderer.swift
//
//
// Created by Vladislav Fitc on 09/10/2021.
//
import Foundation
#if canImport(UIKit) && (os(iOS) || os(tvOS) || os(macOS))
import UIKit
class PokemonUIKitRenderer: PokemonRenderer {
func render(_ pokemon: Pokemon) -> UILabel {
let label = UILabel()
label.text = pokemon.name
return label
}
}
#endif
Pokelib
README
Package
Sources
Pokelib
Pokemon
PokemonRenderer
PokemonService
PokemonS...ftUIRenderer
PokemonUIKitRenderer
Tests
PokelibTests
Cartfile
carthage-prebuild
Package.resolved
Pokelib.podspec
Package Dependencies
swift-log 1.4.2
Filter
Pokelib
main
Pokelib 〉 iPhone 12
Build Succeeded | Yesterday at 22:22
Package
PokemonSwiftUIRenderer
Pokelib 〉 Sources 〉 Pokelib 〉 PokemonSwiftUIRenderer 〉 PokemonSwiftUIRenderer
//
// PokemonSwiftUIRenderer.swift
//
//
// Created by Vladislav Fitc on 09/10/2021.
//
import Foundation
#if canImport(SwiftUI) && (arch(arm64) || arch(x86_64))
import SwiftUI
@available(iOS 13.0, OSX 10.15, tvOS 13.0, watchOS 6.0, *)
class PokemonSwiftUIRenderer: PokemonRenderer {
func render(_ pokemon: Pokemon) -> Text {
return Text(pokemon.name)
}
}
#endif
1 line


Pokelib
main
Pokelib 〉 iPhone 12
Build Succeeded | Yesterday at 22:22
Package
PokemonUIKitRenderer
Pokelib 〉 Sources 〉 Pokelib 〉 PokemonUIKitRenderer 〉 No Selection
Pokelib
README
Package
Sources
Pokelib
Pokemon
PokemonRenderer
PokemonService
PokemonS...ftUIRenderer
PokemonUIKitRenderer
Tests
PokelibTests
Cartfile
carthage-prebuild
Package.resolved
Pokelib.podspec
Package Dependencies
swift-log 1.4.2
Filter
//
// PokemonUIKitRenderer.swift
//
//
// Created by Vladislav Fitc on 09/10/2021.
//
import Foundation
#if canImport(UIKit) && (os(iOS) || os(tvOS) || os(macOS))
import UIKit
class PokemonUIKitRenderer: PokemonRenderer {
func render(_ pokemon: Pokemon) -> UILabel {
let label = UILabel()
label.text = pokemon.name
return label
}
}
#endif
Pokelib
main
Pokelib 〉 iPhone 12
Build Succeeded | Yesterday at 22:22
Package
PokemonSwiftUIRenderer
Pokelib 〉 Sources 〉 Pokelib 〉 PokemonSwiftUIRenderer 〉 PokemonSwiftUIRenderer
Pokelib
README
Package
Sources
Pokelib
Pokemon
PokemonRenderer
PokemonService
PokemonS...ftUIRenderer
PokemonUIKitRenderer
Tests
PokelibTests
Cartfile
carthage-prebuild
Package.resolved
Pokelib.podspec
Package Dependencies
swift-log 1.4.2
Filter
//
// PokemonSwiftUIRenderer.swift
//
//
// Created by Vladislav Fitc on 09/10/2021.
//
import Foundation
#if canImport(SwiftUI) && (arch(arm64) || arch(x86_64))
import SwiftUI
@available(iOS 13.0, OSX 10.15, tvOS 13.0, watchOS 6.0, *)
class PokemonSwiftUIRenderer: PokemonRenderer {
func render(_ pokemon: Pokemon) -> Text {
return Text(pokemon.name)
}
}
#endif
1 line

Поддержка Carthage




Terminal
swift package generate-xcodeproj

carthage_prebuild

```
packages=(
  swift-log
  Pokelib
)
```

carthage_prebuild

```
packages=(
  swift-log
  Pokelib
)
cd Carthage/Checkouts/
```

carthage_prebuild

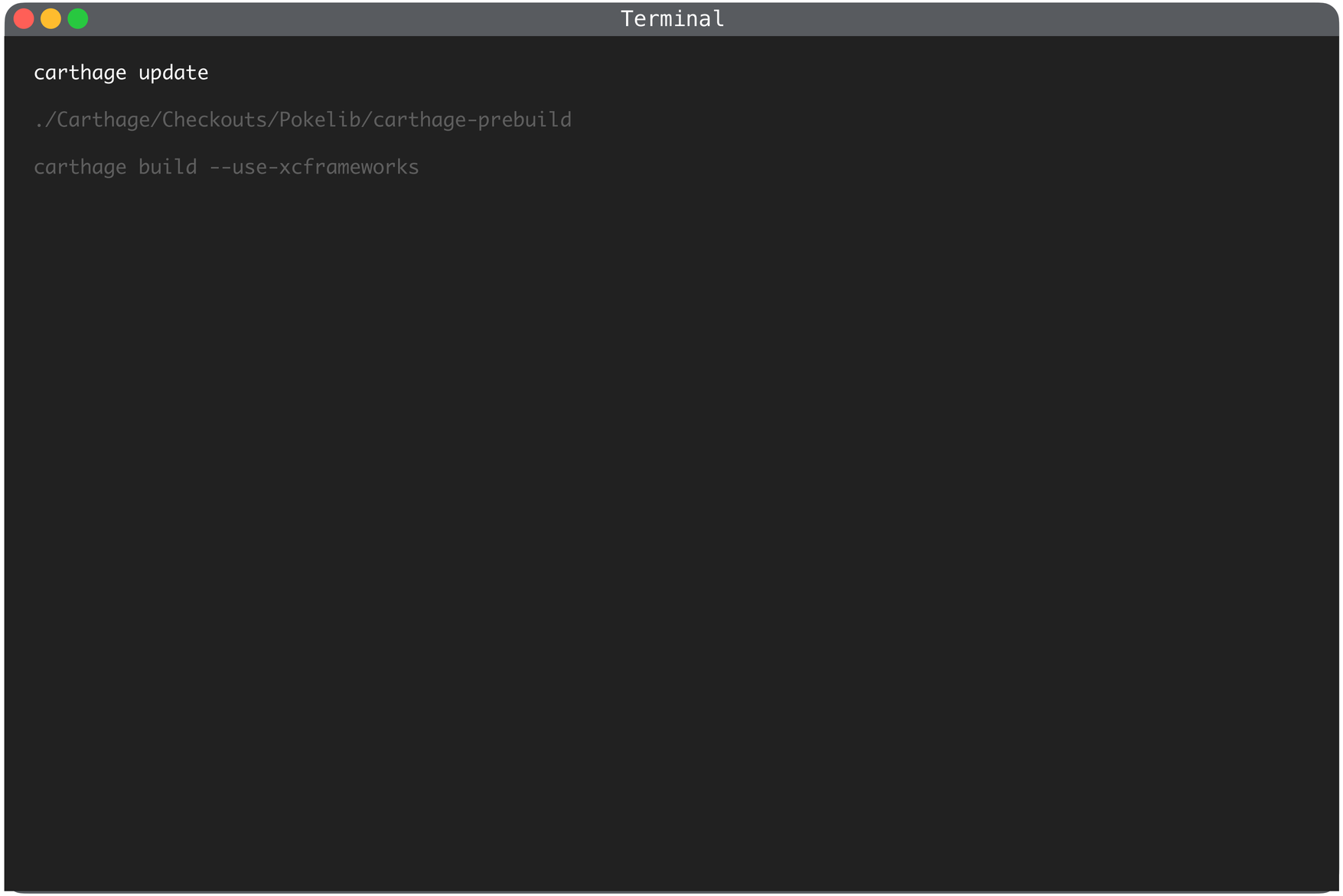
```
packages=(
  swift-log
  Pokelib
)
cd Carthage/Checkouts/
for package in "${packages[@]}"; do
  cd ./$package
  swift package generate-xcodeproj
  cd ..
done
```

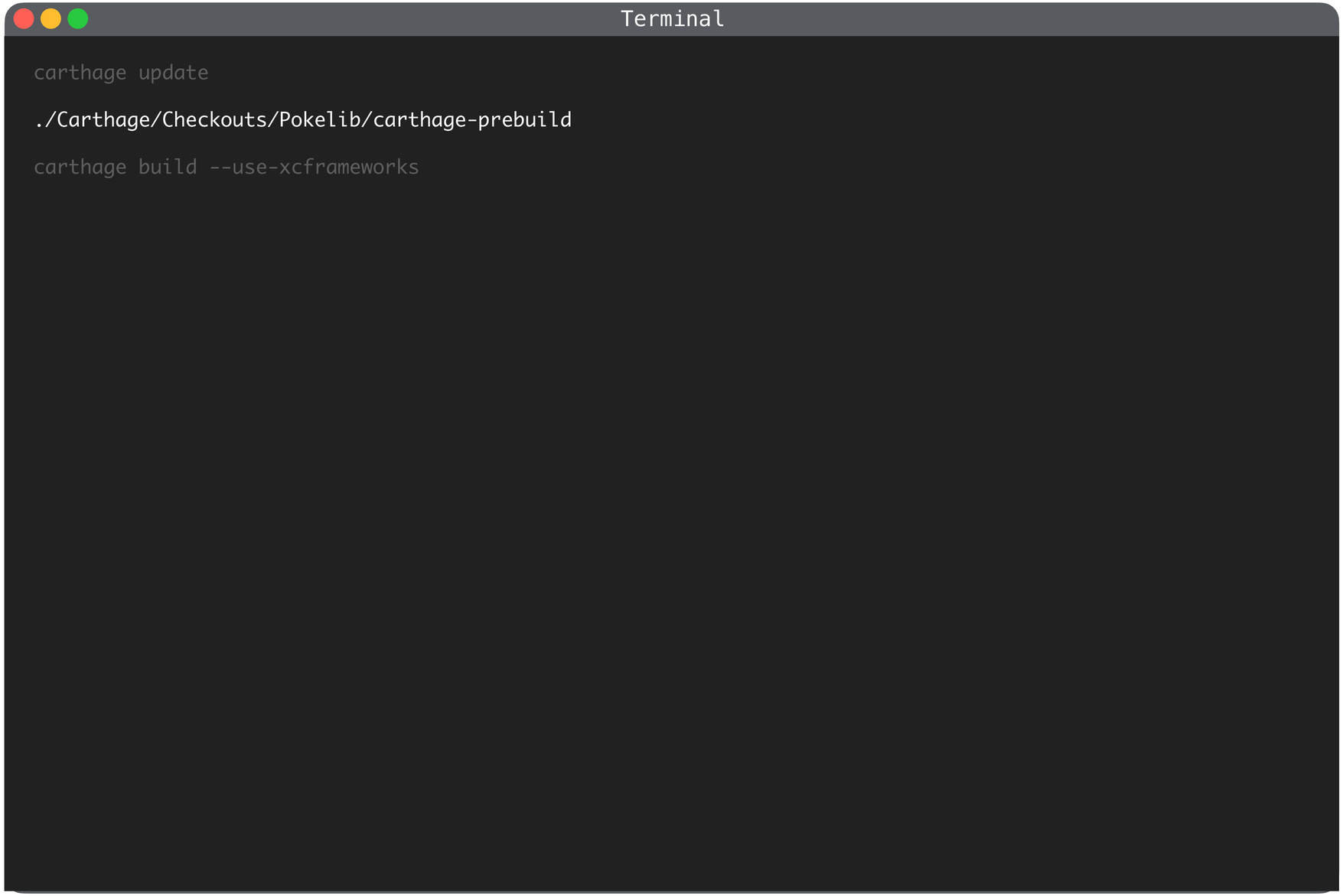
carthage_prebuild

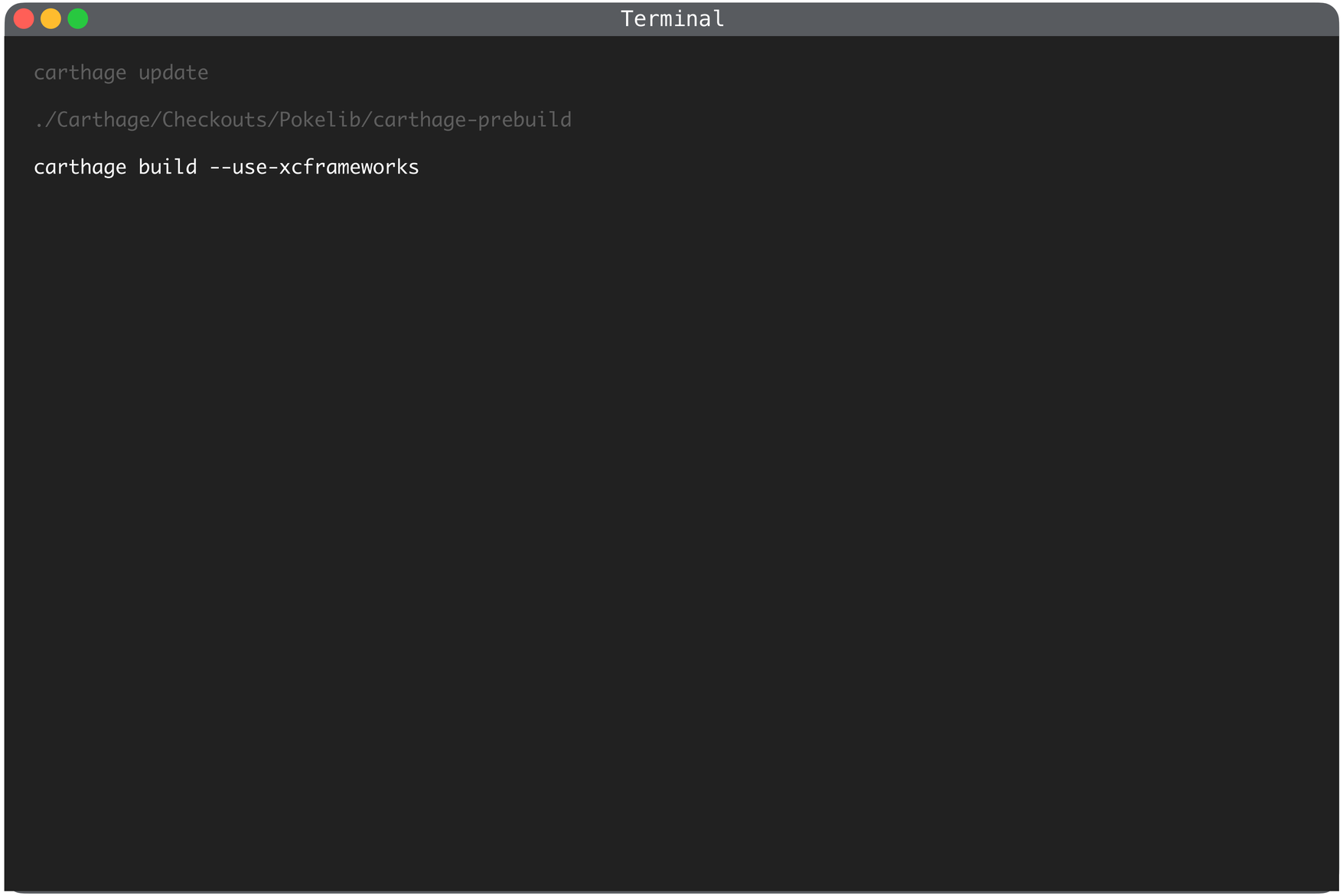
```
packages=(
  swift-log
  Pokelib
)
cd Carthage/Checkouts/
for package in "${packages[@]}"; do
  cd ./$package
  swift package generate-xcodeproj
  cd ..
done
cd ../../..
```

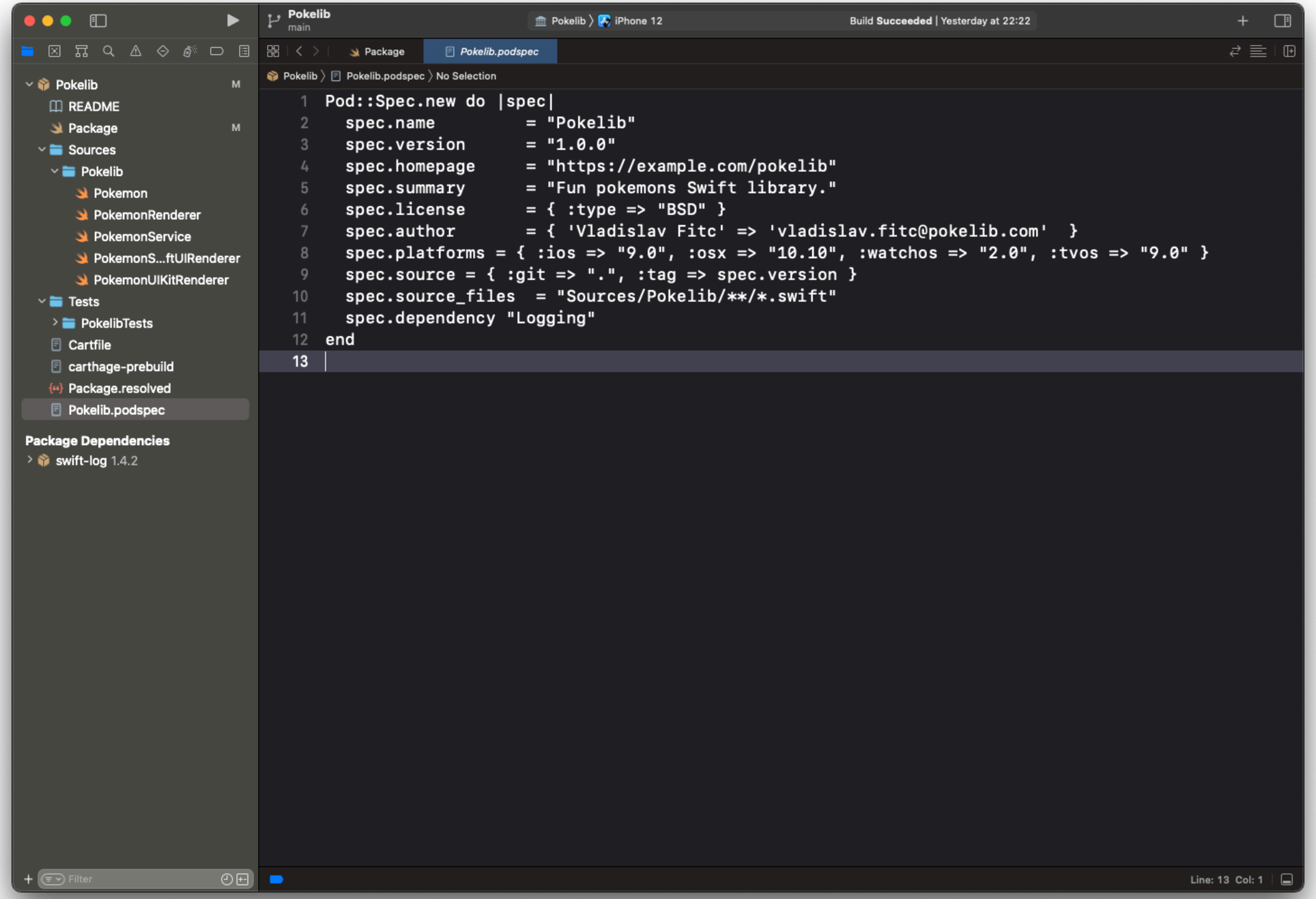
Terminal

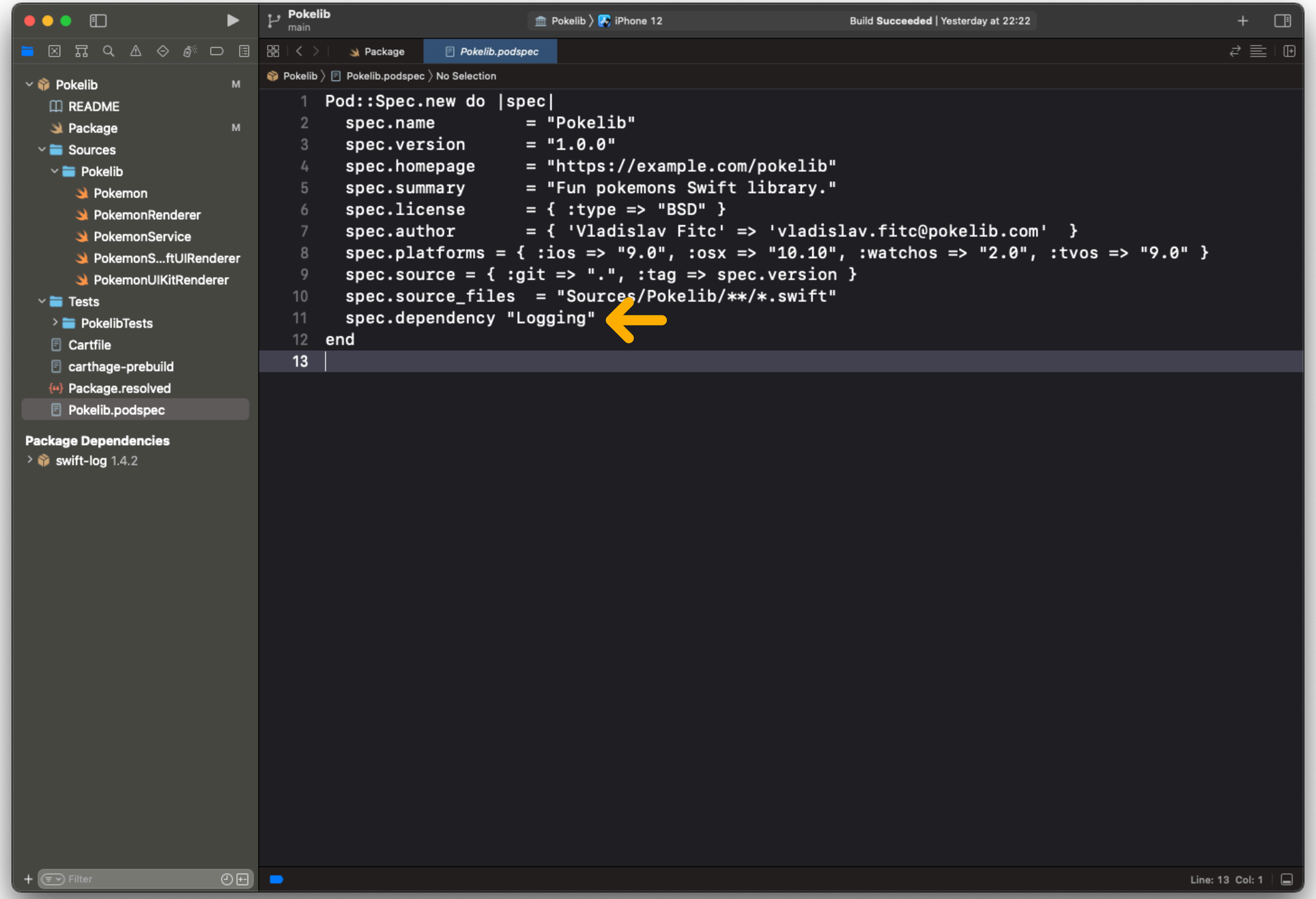
```
carthage update

./Carthage/Checkouts/Pokelib/carthage-prebuild

carthage build --use-xcframeworks
```


Terminal
carthage update
./Carthage/Checkouts/Pokelib/carthage-prebuild
carthage build --use-xcframeworks


Terminal
carthage update
./Carthage/Checkouts/Pokelib/carthage-prebuild
carthage build --use-xcframeworks

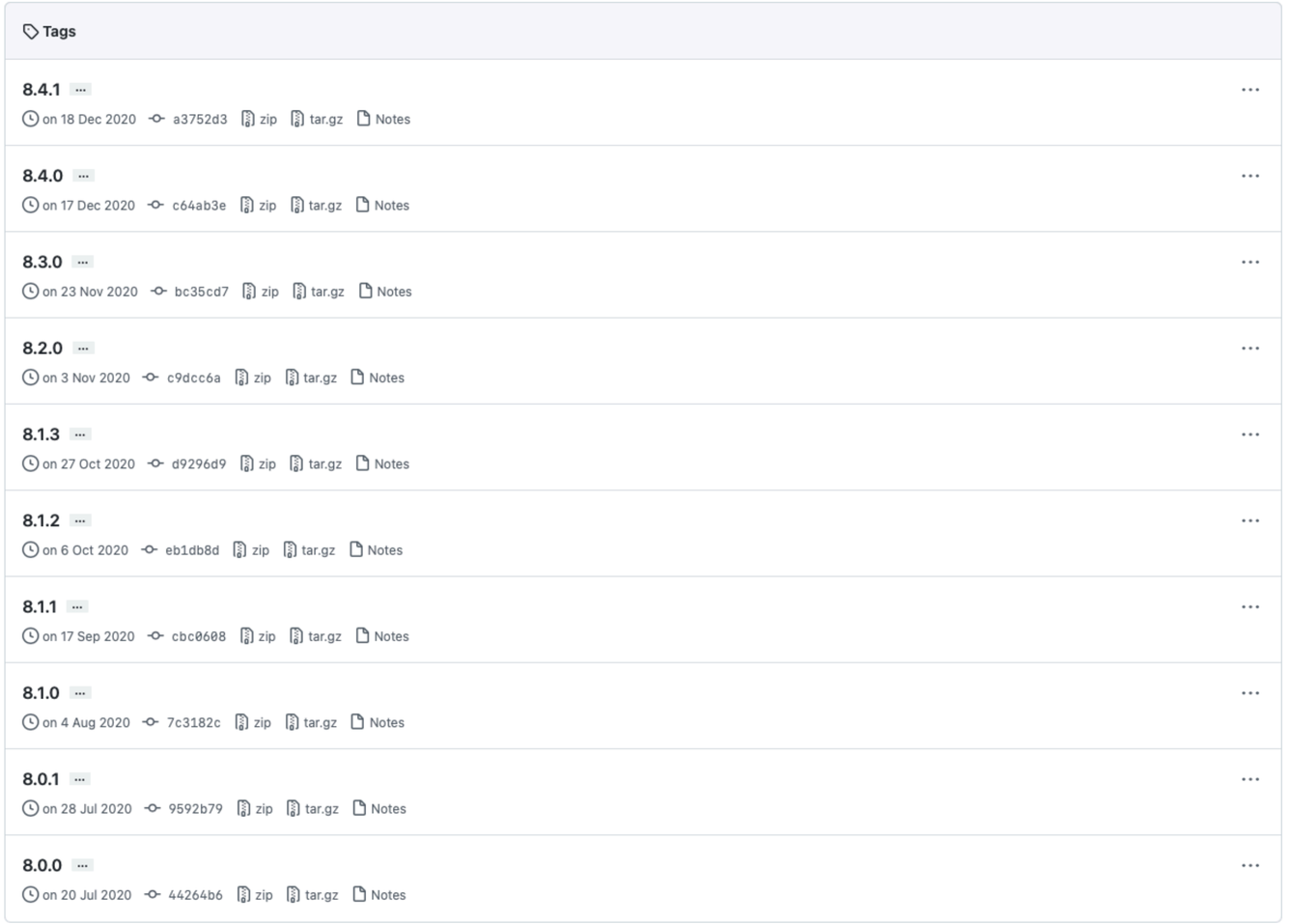

Terminal
carthage update
./Carthage/Checkouts/Pokelib/carthage-prebuild
carthage build --use-xcframeworks


Pokelib
main
Pokelib 〉 iPhone 12
Build Succeeded | Yesterday at 22:22
Package
Pokelib.podspec
Pokelib 〉 Pokelib.podspec 〉 No Selection
Pokelib M
README
Package M
Sources
Pokelib
Pokemon
PokemonRenderer
PokemonService
PokemonS...ftUIRenderer
PokemonUIKitRenderer
Tests
PokelibTests
Cartfile
carthage-prebuild
Package.resolved
Pokelib.podspec
Package Dependencies
swift-log 1.4.2
Filter
Pod::Spec.new do |spec|
spec.name = "Pokelib"
spec.version = "1.0.0"
spec.homepage = "https://example.com/pokelib"
spec.summary = "Fun pokemons Swift library."
spec.license = { :type => "BSD" }
spec.author = { 'Vladislav Fitc' => 'vladislav.fitc@pokelib.com' }
spec.platforms = { :ios => "9.0", :osx => "10.10", :watchos => "2.0", :tvos => "9.0" }
spec.source = { :git => ".", :tag => spec.version }
spec.source_files = "Sources/Pokelib/**/*.swift"
spec.dependency "Logging"
end
Line: 13 Col: 1

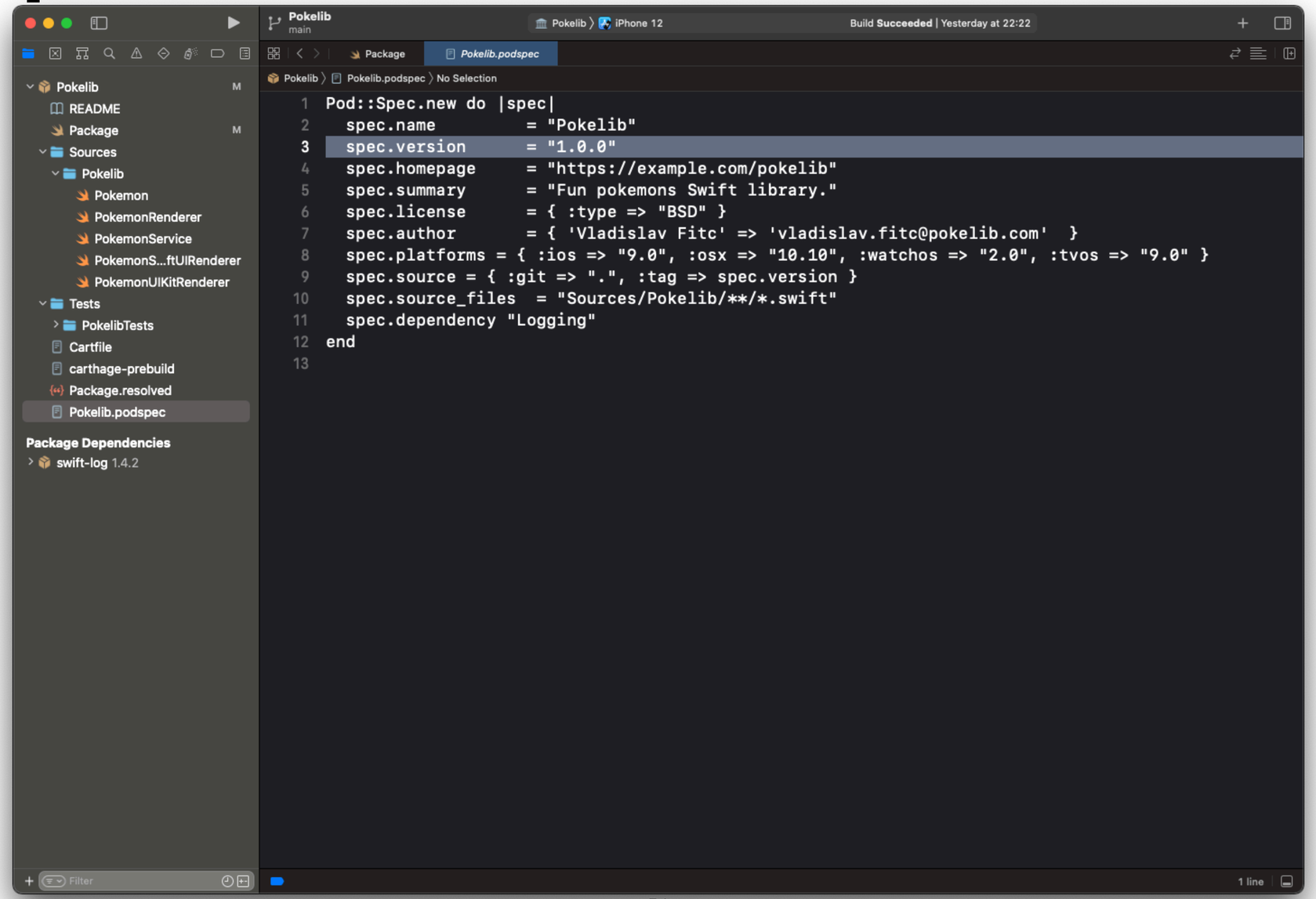

Pokelib
main
Pokelib 〉 iPhone 12
Build Succeeded | Yesterday at 22:22
Package
Pokelib.podspec
Pokelib 〉 Pokelib.podspec 〉 No Selection
Pokelib M
README
Package M
Sources
Pokelib
Pokemon
PokemonRenderer
PokemonService
PokemonS...ftUIRenderer
PokemonUIKitRenderer
Tests
PokelibTests
Cartfile
carthage-prebuild
Package.resolved
Pokelib.podspec
Package Dependencies
swift-log 1.4.2
Filter
Pod::Spec.new do |spec|
spec.name = "Pokelib"
spec.version = "1.0.0"
spec.homepage = "https://example.com/pokelib"
spec.summary = "Fun pokemons Swift library."
spec.license = { :type => "BSD" }
spec.author = { 'Vladislav Fitc' => 'vladislav.fitc@pokelib.com' }
spec.platforms = { :ios => "9.0", :osx => "10.10", :watchos => "2.0", :tvos => "9.0" }
spec.source = { :git => ".", :tag => spec.version }
spec.source_files = "Sources/Pokelib/**/*.swift"
spec.dependency "Logging"
end
Line: 13 Col: 1

Версионирование и релизы

SPM & Carthage



Cocoapods




```
Pod::Spec.new do |spec|
  spec.name         = "Pokelib"
  spec.version      = "1.0.0"
  spec.homepage     = "https://example.com/pokelib"
  spec.summary      = "Fun pokemons Swift library."
  spec.license      = { :type => "BSD" }
  spec.author       = { 'Vladislav Fitc' => 'vladislav.fitc@pokelib.com'  }
  spec.platforms = { :ios => "9.0", :osx => "10.10", :watchos => "2.0", :tvos => "9.0" }
  spec.source = { :git => ".", :tag => spec.version }
  spec.source_files  = "Sources/Pokelib/**/*.swift"
  spec.dependency "Logging"
end
```

Fastlane release

Fastfile

```
release_type = options[:type] # patch | minor | major
```

Fastlane release

Fastfile

```
release_type = options[:type] # patch | minor | major

new_version = version_bump_podspec(
  bump_type: release_type,
  path: podspec_path
)
```

Fastlane release

Fastfile

```
release_type = options[:type] # patch | minor | major

new_version = version_bump_podspec(
  bump_type: release_type,
  path: podspec_path
)

branch_name = "version-#{new_version}"
sh("git checkout -b #{branch_name}")
```

Fastlane release

Fastfile

```
release_type = options[:type] # patch | minor | major

new_version = version_bump_podspec(
  bump_type: release_type,
  path: podspec_path
)

branch_name = "version-#{new_version}"
sh("git checkout -b #{branch_name}")

git_commit(
  path: [podspec_path],
  message: "chore: update version to #{new_version} [skip ci]"
)
```

Fastlane release

Fastfile

```
release_type = options[:type] # patch | minor | major

new_version = version_bump_podspec(
  bump_type: release_type,
  path: podspec_path
)

branch_name = "version-#{new_version}"
sh("git checkout -b #{branch_name}")

git_commit(
  path: [podspec_path],
  message: "chore: update version to #{new_version} [skip ci]"
)

add_git_tag(
  build_number: new_version,
  tag: new_version
)
```

Fastlane release

Fastfile

```
release_type = options[:type] # patch | minor | major

new_version = version_bump_podspec(
  bump_type: release_type,
  path: podspec_path
)

branch_name = "version-#{new_version}"
sh("git checkout -b #{branch_name}")

git_commit(
  path: [podspec_path],
  message: "chore: update version to #{new_version} [skip ci]"
)

add_git_tag(
  build_number: new_version,
  tag: new_version
)

push_to_git_remote(remote: "origin")
```

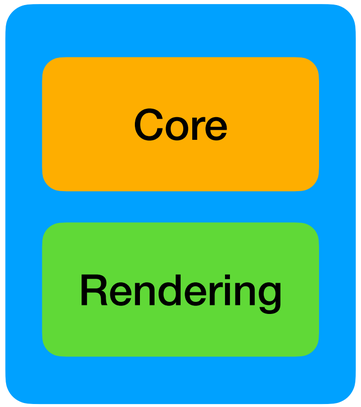
Fastlane release

Fastfile

```
release_type = options[:type] # patch | minor | major

new_version = version_bump_podspec(
  bump_type: release_type,
  path: podspec_path
)

branch_name = "version-#{new_version}"
sh("git checkout -b #{branch_name}")

git_commit(
  path: [podspec_path],
  message: "chore: update version to #{new_version} [skip ci]"
)

add_git_tag(
  build_number: new_version,
  tag: new_version
)

push_to_git_remote(remote: "origin")

pod_push(path: podspec_path)
```

Модульная архитектура

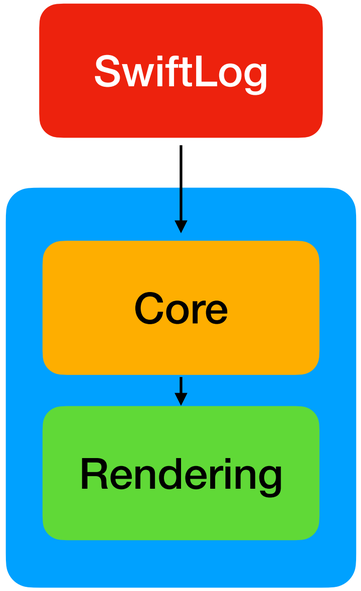
Модульная архитектура



- `Pokemon.swift`
- `PokemonService.swift`

- `PokemonRenderer.swift`
- `PokemonUIKitRenderer.swift`
- `PokemonSwiftUIRenderer.swift`

Модульная архитектура



Модульная архитектура + SPM

Package.swift

```
// swift-tools-version:5.5

let package = Package(
    name: "Pokelib",
    products: [
        .library(
            name: "PokelibCore",
            targets: ["PokelibCore"]),
        .library(
          name: "PokelibRendering",
          targets: ["PokelibRendering"])
    ],
    dependencies: [
      .package(url: "https://github.com/apple/swift-log.git", from: "1.4.2")
    ],
    targets: [
        .target(
            name: "PokelibCore",
            dependencies: [.product(name: "Logging", package: "swift-log")]),
        .target(
            name: "PokelibRendering",
            dependencies: ["PokelibCore"]),
        .testTarget(
            name: "PokelibCoreTests",
            dependencies: [.target(name: "PokelibCore")]),
        .testTarget(
            name: "PokelibRenderingTests",
            dependencies: [.target(name: "PokelibRendering")]),
    ]
)
```

Модульная архитектура + SPM

Package.swift

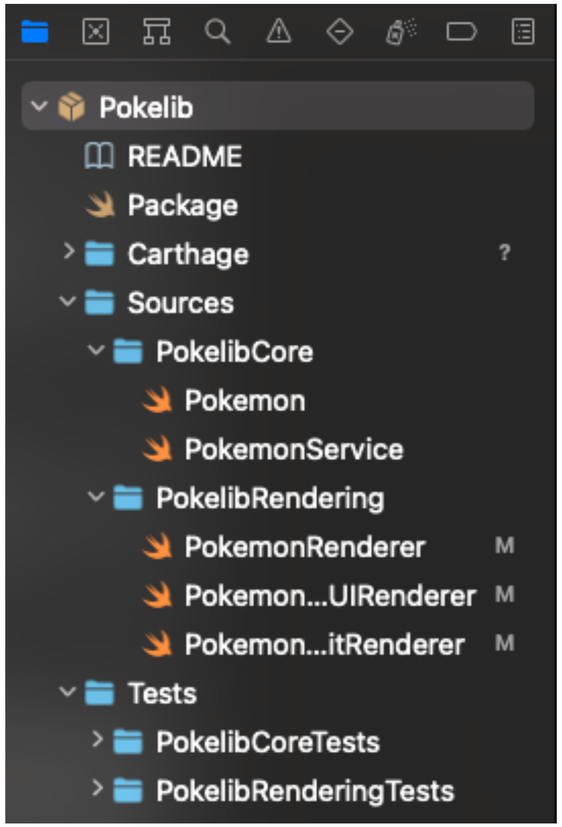
```
// swift-tools-version:5.5

let package = Package(
    name: "Pokelib",
    products: [
        .library(
            name: "PokelibCore",
            targets: ["PokelibCore"]),
        .library(
          name: "PokelibRendering",
          targets: ["PokelibRendering"])
    ],
    dependencies: [
      .package(url: "https://github.com/apple/swift-log.git", from: "1.4.2")
    ],
    targets: [
        .target(
            name: "PokelibCore",
            dependencies: [.product(name: "Logging", package: "swift-log")]),
        .target(
            name: "PokelibRendering",
            dependencies: ["PokelibCore"]),
        .testTarget(
            name: "PokelibCoreTests",
            dependencies: [.target(name: "PokelibCore")]),
        .testTarget(
            name: "PokelibRenderingTests",
            dependencies: [.target(name: "PokelibRendering")]),
    ]
)
```

Модульная архитектура + SPM

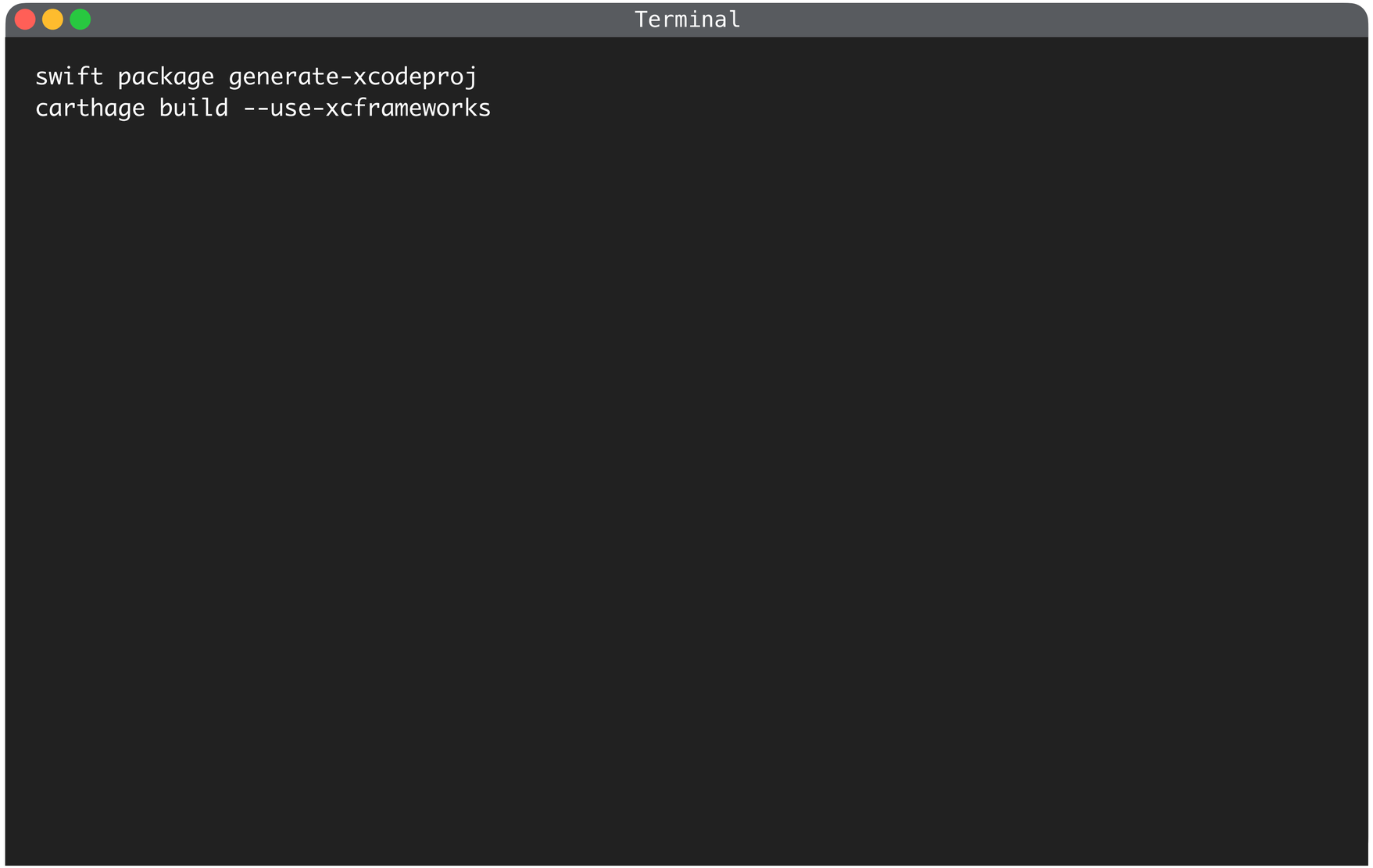
Package.swift

```
// swift-tools-version:5.5

let package = Package(
    name: "Pokelib",
    products: [
        .library(
            name: "PokelibCore",
            targets: ["PokelibCore"]),
        .library(
          name: "PokelibRendering",
          targets: ["PokelibRendering"])
    ],
    dependencies: [
      .package(url: "https://github.com/apple/swift-log.git", from: "1.4.2")
    ],
    targets: [
        .target(
            name: "PokelibCore",
            dependencies: [.product(name: "Logging", package: "swift-log")]),
        .target(
            name: "PokelibRendering",
            dependencies: ["PokelibCore"]),
        .testTarget(
            name: "PokelibCoreTests",
            dependencies: [.target(name: "PokelibCore")]),
        .testTarget(
            name: "PokelibRenderingTests",
            dependencies: [.target(name: "PokelibRendering")]),
    ]
)
```

Модульная архитектура + SPM



Модульная архитектура + Carthage



Модульная архитектура + Cocoapods

Pokelib.podspec

```
Pod::Spec.new do |spec|
  spec.name         = "Pokelib"
  spec.version      = "1.0.0"
  spec.homepage     = "https://example.com/pokelib"
  spec.summary      = "Fun pokemons Swift library."
  spec.license      = { :type => "BSD" }
  spec.author       = { 'Vladislav Fitc' => 'vladislav.fitc@pokelib.com'  }
  spec.platforms = { :ios => "9.0", :osx => "10.10", :watchos => "2.0", :tvos => "9.0" }
  spec.source = { :git => ".", :tag => spec.version }

  spec.default_subspec = 'Core'

  spec.subspec "Core" do |ss|
      ss.source_files = 'Sources/PokelibCore/**/*.{swift}'
      ss.dependency 'Logging', '~> 1.4'
  end

  spec.subspec "Rendering" do |ss|
      ss.source_files = 'Sources/PokelibRendering/**/*.{swift}'
      ss.dependency 'Pokelib/Core'
 end

end
```

Модульная архитектура + Cocoapods

Pokelib.podspec

```
Pod::Spec.new do |spec|
  spec.name         = "Pokelib"
  spec.version      = "1.0.0"
  spec.homepage     = "https://example.com/pokelib"
  spec.summary      = "Fun pokemons Swift library."
  spec.license      = { :type => "BSD" }
  spec.author       = { 'Vladislav Fitc' => 'vladislav.fitc@pokelib.com'  }
  spec.platforms = { :ios => "9.0", :osx => "10.10", :watchos => "2.0", :tvos => "9.0" }
  spec.source = { :git => ".", :tag => spec.version }

  spec.default_subspec = 'Core'

  spec.subspec "Core" do |ss|
      ss.source_files = 'Sources/PokelibCore/**/*.{swift}'
      ss.dependency 'Logging', '~> 1.4'
  end

  spec.subspec "Rendering" do |ss|
      ss.source_files = 'Sources/PokelibRendering/**/*.{swift}'
      ss.dependency 'Pokelib/Core'
 end

end
```

Модульная архитектура + Cocoapods

PokemonRenderer.swift

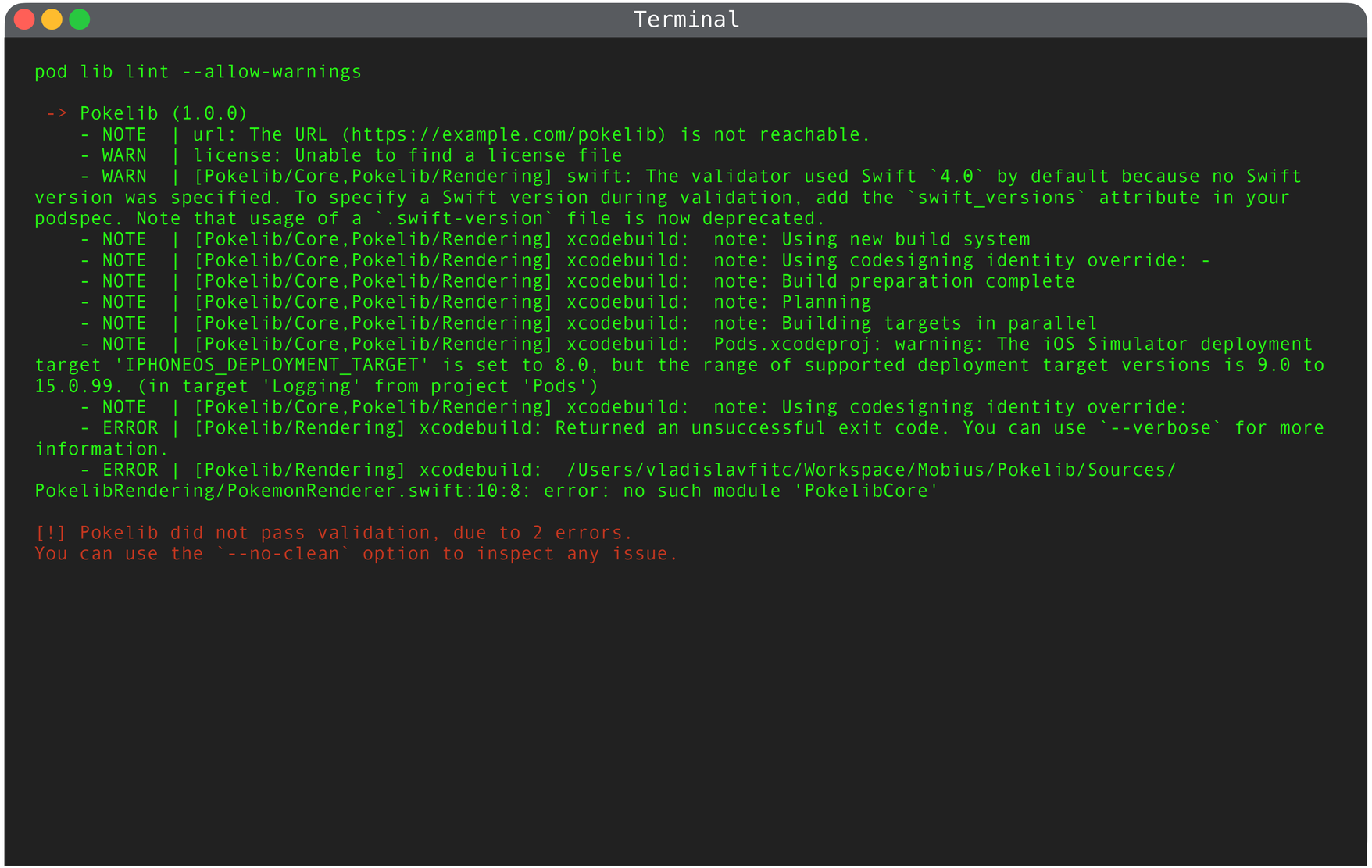
```swift
import Foundation
import PokelibCore

public protocol PokemonRenderer {

  associatedtype Output

  func render(_ pokemon: Pokemon) -> Output

}
```

Модульная архитектура + Cocoapods

PokemonRenderer.swift

```swift
import Foundation
import PokelibCore

public protocol PokemonRenderer {

  associatedtype Output

  func render(_ pokemon: Pokemon) -> Output

}
```

Модульная архитектура + Cocoapods



Модульная архитектура + Cocoapods

Pokelib.podspec

```
Pod::Spec.new do |spec|
  spec.name         = "Pokelib"
  spec.version      = "1.0.0"
  spec.homepage     = "https://example.com/pokelib"
  spec.summary      = "Fun pokemons Swift library."
  spec.license      = { :type => "BSD" }
  spec.author       = { 'Vladislav Fitc' => 'vladislav.fitc@pokelib.com'  }
  spec.platforms = { :ios => "9.0", :osx => "10.10", :watchos => "2.0", :tvos => "9.0" }
  spec.source = { :git => ".", :tag => spec.version }

  spec.default_subspec = 'Core'

  spec.subspec "Core" do |ss|
      ss.source_files = 'Sources/PokelibCore/**/*.{swift}'
      ss.dependency 'Logging', '~> 1.4'
  end

  spec.subspec "Rendering" do |ss|
      ss.source_files = 'Sources/PokelibRendering/**/*.{swift}'
      ss.dependency 'Pokelib/Core'
      ss.pod_target_xcconfig = { 'OTHER_SWIFT_FLAGS' => '-DPokelibCocoaPods' }
  end

end
```

Модульная архитектура + Cocoapods

PokemonRenderer.swift

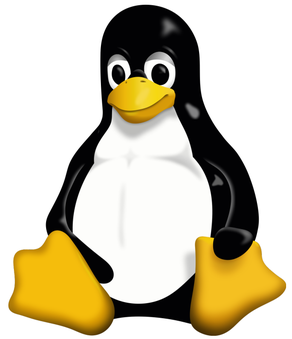
```swift
import Foundation
#if !PokelibCocoaPods
import PokelibCore
#endif

public protocol PokemonRenderer {

  associatedtype Output

  func render(_ pokemon: Pokemon) -> Output

}
```

Поддержка Linux 

Поддержка Linux

Swift Package Manager

+



Linux

Поддержка Linux

PokemonService.swift

```swift
import Foundation
import Logging

public class PokemonService {

  public init() {}

  public func getPokemons(completion: @escaping (Result<[Pokemon], Error>) -> Void) {
    let url = URL(string: "example.com/pokemons/")!
    let logger = Logger(label: "PokemonService")
    URLSession.shared.dataTask(with: url) { data, response, error in
      ...
   }
  }

}
```

Pokemon.swift

```swift
import Foundation

public struct Pokemon: Decodable {

  public let name: String

}
```

PokemonRenderer.swift

```swift
import Foundation
#if !PokelibCocoaPods
import PokelibCore
#endif
public protocol PokemonRenderer {

  associatedtype Output

  func render(_ pokemon: Pokemon) -> Output

}
```

Поддержка Linux

Поддержка Linux

PokemonService.swift

```swift
import Foundation
import Logging

public class PokemonService {

  public init() {}

  public func getPokemons(completion: @escaping (Result<[Pokemon], Error>) -> Void) {
    let url = URL(string: "example.com/pokemons/")!
    let logger = Logger(label: "PokemonService")
    URLSession.shared.dataTask(with: url) { data, response, error in
      ...
   }
  }

}
```

Поддержка Linux

PokemonRenderer.swift

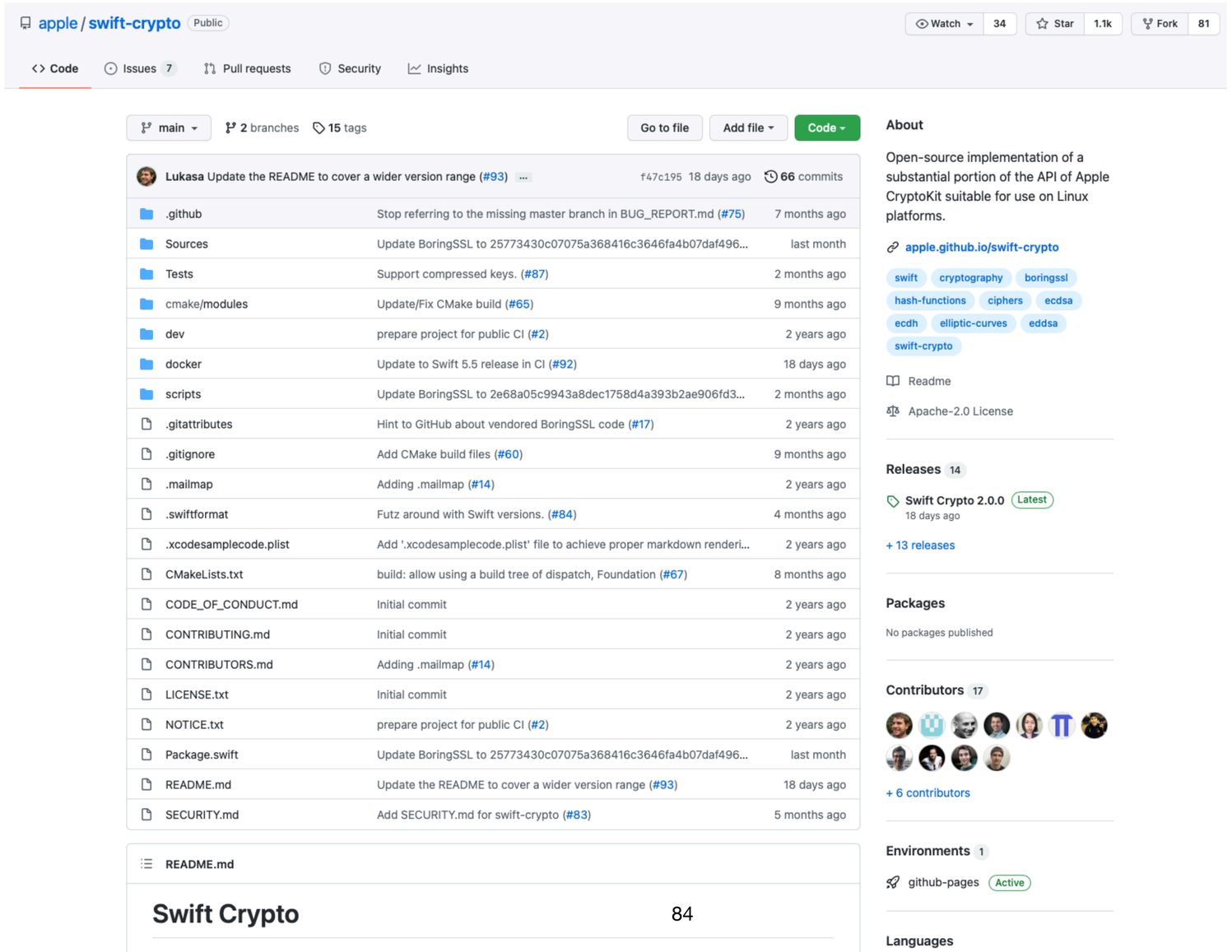
```swift
import Foundation
import Logging
#if canImport(FoundationNetworking)
import FoundationNetworking
#endif

public class PokemonService {

  public init() {}

  public func getPokemons(completion: @escaping (Result<[Pokemon], Error>) -> Void) {
    let url = URL(string: "example.com/pokemons/")!
    let logger = Logger(label: "PokemonService")
    URLSession.shared.dataTask(with: url) { data, response, error in
      ...
   }
  }

}
```

Поддержка Linux

Package.swift

```swift
import PackageDescription

let package = Package(
    name: "Pokelib",
    products: [
        .library(
            name: "PokelibCore",
            targets: ["PokelibCore"]),
        .library(
          name: "PokelibRendering",
          targets: ["PokelibCore"])
    ],
    dependencies: [
      .package(url: "https://github.com/apple/swift-log.git", from: "1.4.2")
    ],
    targets: [
        .target(
            name: "PokelibCore",
            dependencies: [.product(name: "Logging", package: "swift-log")]),
        .target(
            name: "PokelibRendering",
            dependencies: ["PokelibCore"]),
        .testTarget(
            name: "PokelibCoreTests",
            dependencies: [.target(name: "PokelibCore")]),
        .testTarget(
            name: "PokelibRenderingTests",
            dependencies: [.target(name: "PokelibRendering")]),
    ]
)
```

Поддержка Linux



apple / swift-crypto Public

Watch 34 | Star 1.1k | Fork 81

Code | Issues 7 | Pull requests | Security | Insights

main | 2 branches | 15 tags | Go to file | Add file | Code

Lukasa Update the README to cover a wider version range (#93) ... f47c195 18 days ago 66 commits

.github	Stop referring to the missing master branch in BUG_REPORT.md (#75)	7 months ago
Sources	Update BoringSSL to 25773430c07075a368416c3646fa4b07daf496...	last month
Tests	Support compressed keys. (#87)	2 months ago
cmake/modules	Update/Fix CMake build (#65)	9 months ago
dev	prepare project for public CI (#2)	2 years ago
docker	Update to Swift 5.5 release in CI (#92)	18 days ago
scripts	Update BoringSSL to 2e68a05c9943a8dec1758d4a393b2ae906fd3...	2 months ago
.gitattributes	Hint to GitHub about vendored BoringSSL code (#17)	2 years ago
.gitignore	Add CMake build files (#60)	9 months ago
.mailmap	Adding .mailmap (#14)	2 years ago
.swiftformat	Futz around with Swift versions. (#84)	4 months ago
.xcodesamplecode.plist	Add '.xcodesamplecode.plist' file to achieve proper markdown renderi...	2 years ago
CMakeLists.txt	build: allow using a build tree of dispatch, Foundation (#67)	8 months ago
CODE_OF_CONDUCT.md	Initial commit	2 years ago
CONTRIBUTING.md	Initial commit	2 years ago
CONTRIBUTORS.md	Adding .mailmap (#14)	2 years ago
LICENSE.txt	Initial commit	2 years ago
NOTICE.txt	prepare project for public CI (#2)	2 years ago
Package.swift	Update BoringSSL to 25773430c07075a368416c3646fa4b07daf496...	last month
README.md	Update the README to cover a wider version range (#93)	18 days ago
SECURITY.md	Add SECURITY.md for swift-crypto (#83)	5 months ago

README.md

Swift Crypto

About

Open-source implementation of a substantial portion of the API of Apple CryptoKit suitable for use on Linux platforms.

apple.github.io/swift-crypto

swift | cryptography | boringssl | hash-functions | ciphers | ecdsa | ecdh | elliptic-curves | eddsa | swift-crypto

Readme

Apache-2.0 License

Releases 14

Swift Crypto 2.0.0 Latest
18 days ago

+ 13 releases

Packages

No packages published

Contributors 17

+ 6 contributors

Environments 1

github-pages Active

Languages

Поддержка Linux

Package.swift

```
import PackageDescription

#if os(Linux)
let extraPackageDependencies: [Package.Dependency] = [
  .package(url: "https://github.com/apple/swift-crypto.git", from: "1.1.2")
]
#else
let extraPackageDependencies: [Package.Dependency] = []
#endif

let package = Package(
    name: "Pokelib",
    products: [
        .library(
            name: "PokelibCore",
            targets: ["PokelibCore"]),
        .library(
          name: "PokelibRendering",
          targets: ["PokelibCore"])
    ],
    dependencies: [
      .package(url: "https://github.com/apple/swift-log.git", from: "1.4.2")
    ],
    targets: [
        .target(
            name: "PokelibCore",
            dependencies: [.product(name: "Logging", package: "swift-log")]),
        .target(
            name: "PokelibRendering",
            dependencies: ["PokelibCore"]),
        .testTarget(
            name: "PokelibCoreTests",
            dependencies: [.target(name: "PokelibCore")]),
        .testTarget(
            name: "PokelibRenderingTests",
            dependencies: [.target(name: "PokelibRendering")]),
    ]
)
```

Поддержка Linux

Package.swift

```
import PackageDescription

#if os(Linux)
let extraPackageDependencies: [Package.Dependency] = [
  .package(url: "https://github.com/apple/swift-crypto.git", from: "1.1.2")
]
#else
let extraPackageDependencies: [Package.Dependency] = []
#endif

let package = Package(
    name: "Pokelib",
    products: [
        .library(
            name: "PokelibCore",
            targets: ["PokelibCore"]),
        .library(
          name: "PokelibRendering",
          targets: ["PokelibCore"])
    ],
    dependencies: [
      .package(url: "https://github.com/apple/swift-log.git", from: "1.4.2")
    ] + extraPackageDependencies,
    targets: [
        .target(
            name: "PokelibCore",
            dependencies: [.product(name: "Logging", package: "swift-log")] + extraPackageDependencies),
        .target(
            name: "PokelibRendering",
            dependencies: ["PokelibCore"]),
        .testTarget(
            name: "PokelibCoreTests",
            dependencies: [.target(name: "PokelibCore")] + extraPackageDependencies),
        .testTarget(
            name: "PokelibRenderingTests",
            dependencies: [.target(name: "PokelibRendering")]),
    ]
)
```

Поддержка Linux

String+HMAC.swift

```swift
import Foundation

#if os(Linux)
import Crypto
#else
import CommonCrypto
#endif

extension String {

  func hmac256(withKey key: String) -> String {
    #if os(Linux)
    let key = SymmetricKey(data: Data(key.utf8))
    let digest = Array(HMAC<SHA256>.authenticationCode(for: Data(utf8), using: key))
    #else
    let algorithm = HmacAlgorithm.sha256
    var digest = [UInt8](repeating: 0, count: algorithm.digestLength)
    CCHmac(algorithm.algorithm, key, key.count, self, self.count, &digest)
    #endif
    let data = Data(digest)
    return data.map { String(format: "%02hhx", $0) }.joined()
  }

}
```

Поддержка Linux

String+HMAC.swift

```swift
import Foundation

#if os(Linux)
import Crypto
#else
import CommonCrypto
#endif

extension String {

  func hmac256(withKey key: String) -> String {
    #if os(Linux)
    let key = SymmetricKey(data: Data(key.utf8))
    let digest = Array(HMAC<SHA256>.authenticationCode(for: Data(utf8), using: key))
    #else
    let algorithm = HmacAlgorithm.sha256
    var digest = [UInt8](repeating: 0, count: algorithm.digestLength)
    CCHmac(algorithm.algorithm, key, key.count, self, self.count, &digest)
    #endif
    let data = Data(digest)
    return data.map { String(format: "%02hhx", $0) }.joined()
  }

}
```

Поддержка Linux

String+HMAC.swift

```swift
import Foundation

#if os(Linux)
import Crypto
#else
import CommonCrypto
#endif

extension String {

  func hmac256(withKey key: String) -> String {
    #if os(Linux)
    let key = SymmetricKey(data: Data(key.utf8))
    let digest = Array(HMAC<SHA256>.authenticationCode(for: Data(utf8), using: key))
    #else
    let algorithm = HmacAlgorithm.sha256
    var digest = [UInt8](repeating: 0, count: algorithm.digestLength)
    CCHmac(algorithm.algorithm, key, key.count, self, self.count, &digest)
    #endif
    let data = Data(digest)
    return data.map { String(format: "%02hhx", $0) }.joined()
  }

}
```

Поддержка Linux

String+HMAC.swift

```swift
import Foundation

#if os(Linux)
import Crypto
#else
import CommonCrypto
#endif

extension String {

  func hmac256(withKey key: String) -> String {
    #if os(Linux)
    let key = SymmetricKey(data: Data(key.utf8))
    let digest = Array(HMAC<SHA256>.authenticationCode(for: Data(utf8), using: key))
    #else
    let algorithm = HmacAlgorithm.sha256
    var digest = [UInt8](repeating: 0, count: algorithm.digestLength)
    CCHmac(algorithm.algorithm, key, key.count, self, self.count, &digest)
    #endif
    let data = Data(digest)
    return data.map { String(format: "%02hhx", $0) }.joined()
  }

}
```

Поддержка Linux

String+HMAC.swift

```swift
import Foundation

#if os(Linux)
import Crypto
#else
import CommonCrypto
#endif

extension String {

  func hmac256(withKey key: String) -> String {
    #if os(Linux)
    let key = SymmetricKey(data: Data(key.utf8))
    let digest = Array(HMAC<SHA256>.authenticationCode(for: Data(utf8), using: key))
    #else
    let algorithm = HmacAlgorithm.sha256
    var digest = [UInt8](repeating: 0, count: algorithm.digestLength)
    CCHmac(algorithm.algorithm, key, key.count, self, self.count, &digest)
    #endif
    let data = Data(digest)
    return data.map { String(format: "%02hhx", $0) }.joined()
  }

}
```

Тестирование 

Тестирование

- Заводим тестовый проект, использующий нашу зависимость

Тестирование

- Заводим тестовый проект, использующий нашу зависимость
- Подключаем нашу зависимость при помощи одного из менеджеров

Тестирование

- Заводим тестовый проект, использующий нашу зависимость
- Подключаем нашу зависимость при помощи одного из менеджеров
- Проверяем корректность сборки

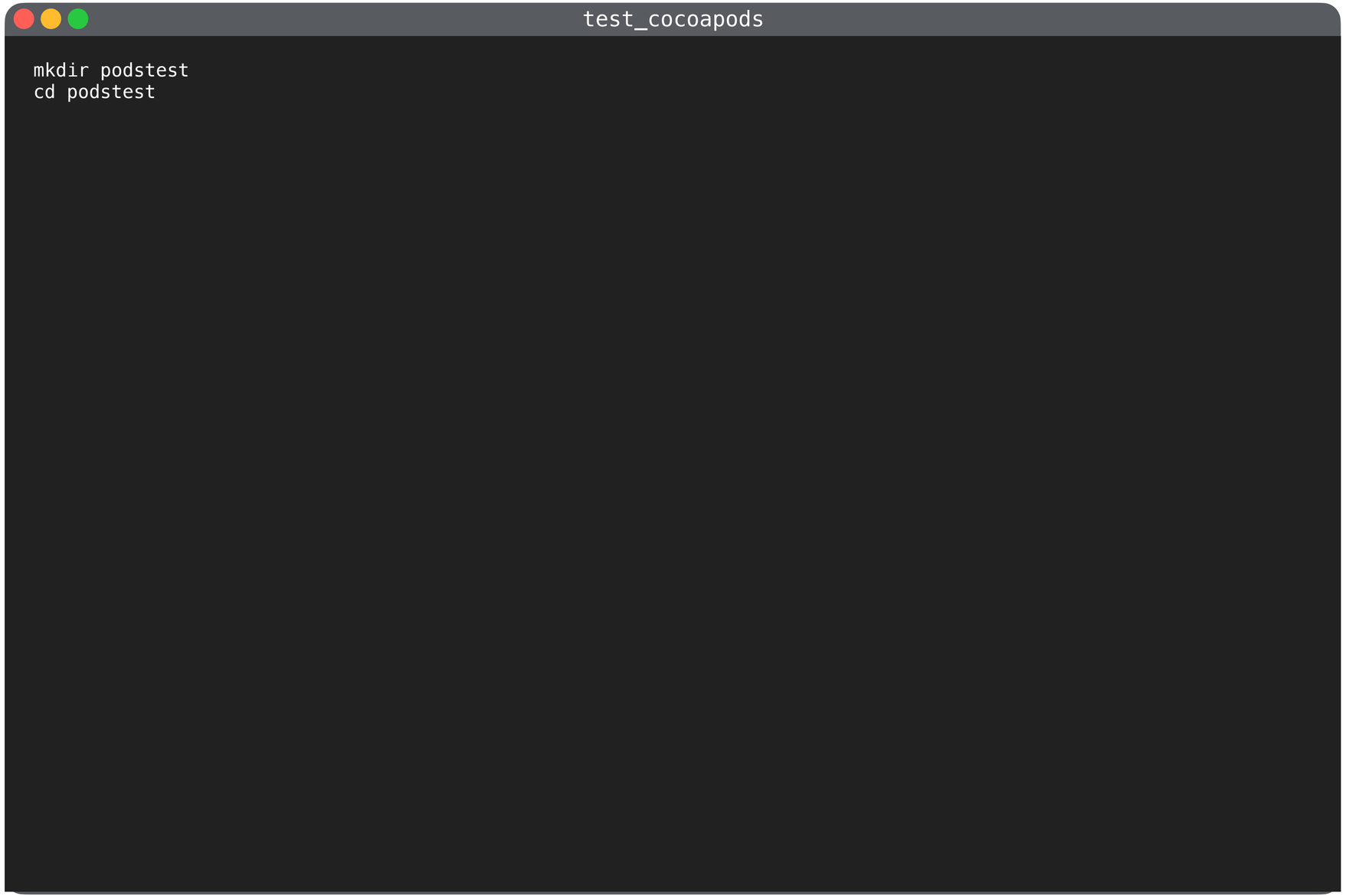
Тестирование

- Заводим тестовый проект, использующий нашу зависимость
- Подключаем нашу зависимость при помощи одного из менеджеров
- Проверяем корректность сборки
- Добавляем эти шаги в CI

Тестирование

- Заводим тестовый проект, использующий нашу зависимость
- Подключаем нашу зависимость при помощи одного из менеджеров
- Проверяем корректность сборки
- Добавляем эти шаги в CI
- Тестировать работу Linux можно в Docker-е

Тестирование Cocoapods



Тестирование Cocoapods

test_cocoapods

```
mkdir podstest
cd podstest

cat > Podfile <<EOL
target 'podstest' do
  use_frameworks!
  pod 'Pokelib', '~> $1'
end
EOL
```

Тестирование Cocoapods

test_cocoapods

```
mkdir podstest
cd podstest

cat > Podfile <<EOL
target 'podstest' do
  use_frameworks!
  pod 'Pokelib', '~> $1'
end
EOL
```

Тестирование Cocoapods

test_cocoapods

```
mkdir podstest
cd podstest

cat > Podfile <<EOL
target 'podstest' do
  use_frameworks!
  pod 'Pokelib', '~> $1'
end
EOL

swift package init --type library
swift package generate-xcodeproj
```

Тестирование Cocoapods

test_cocoapods

```
mkdir podstest
cd podstest

cat > Podfile <<EOL
target 'podstest' do
  use_frameworks!
  pod 'Pokelib', '~> $1'
end
EOL

swift package init --type library
swift package generate-xcodeproj

pod repo update
pod install
```

Тестирование Cocoapods

test_cocoapods

```
mkdir podstest
cd podstest

cat > Podfile <<EOL
target 'podstest' do
  use_frameworks!
  pod 'Pokelib', '~> $1'
end
EOL

swift package init --type library
swift package generate-xcodeproj

pod repo update
pod install

cat > Sources/podstest/podstest.swift <<EOL
import Pokelib

public struct podstest {
    public private(set) var text = "Hello, World!"

    public init() {}
}
EOL
```

Тестирование Cocoapods

test_cocoapods

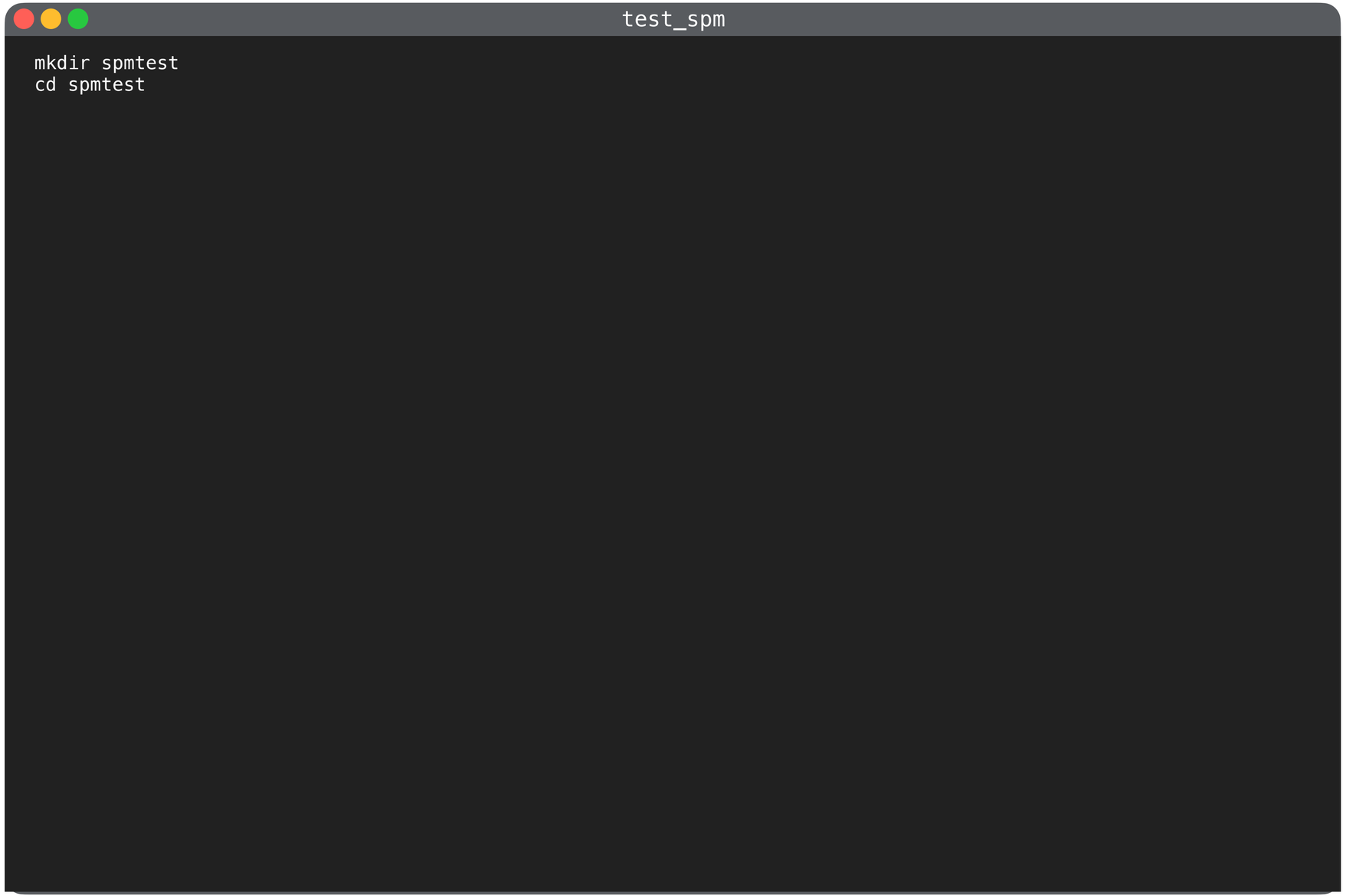
```
mkdir podstest
cd podstest

cat > Podfile <<EOL
target 'podstest' do
  use_frameworks!
  pod 'Pokelib', '~> $1'
end
EOL

swift package init --type library
swift package generate-xcodeproj

pod repo update
pod install

cat > Sources/podstest/podstest.swift <<EOL
import Pokelib

public struct podstest {
    public private(set) var text = "Hello, World!"

    public init() {}
}
EOL
```

Тестирование Cocoapods

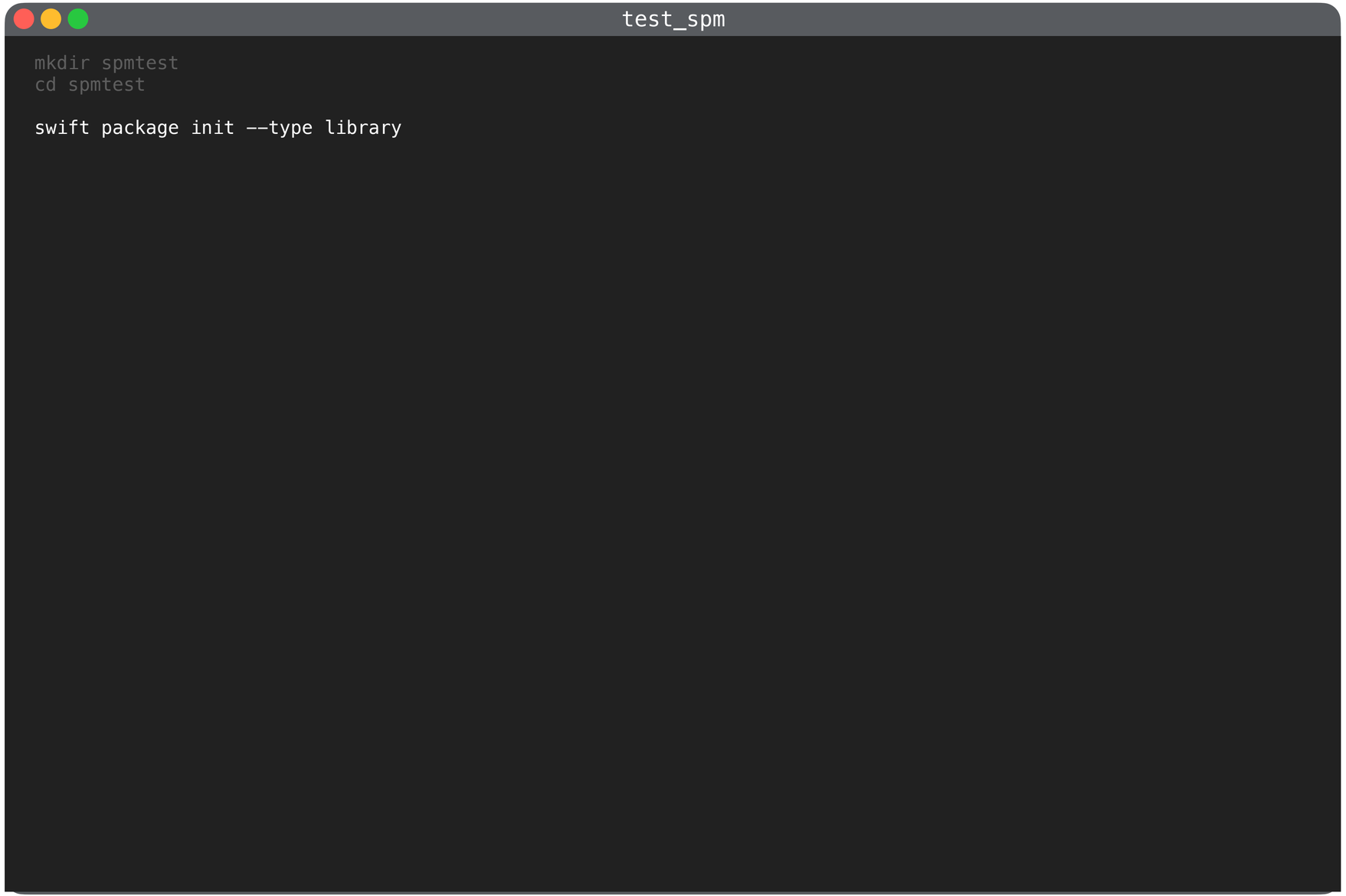
test_cocoapods

```
mkdir podstest
cd podstest

cat > Podfile <<EOL
target 'podstest' do
  use_frameworks!
  pod 'Pokelib', '~> $1'
end
EOL

swift package init --type library
swift package generate-xcodeproj

pod repo update
pod install

cat > Sources/podstest/podstest.swift <<EOL
import Pokelib

public struct podstest {
    public private(set) var text = "Hello, World!"

    public init() {}
}
EOL

xcodebuild \
  -workspace podstest.xcworkspace \
  -scheme podstest-Package \
  -destination 'platform=iOS Simulator,name=iPhone 12,OS=15.0'
```

Тестирование SPM



Тестирование SPM



Тестирование SPM

test_spm

```
mkdir spmtest
cd spmtest

swift package init --type library

cat > Package.swift <<EOL
// swift-tools-version:5.5
import PackageDescription
let package = Package(
    name: "spmtest",
    products: [
        .library(
            name: "spmtest",
            targets: ["spmtest"]),
    ],
    dependencies: [
      .package(name: "Pokelib",
               url: "https://github.com/pokemons/pokelib",
               from: "$1")
    ],
    targets: [
        .target(
            name: "spmtest",
            dependencies: ["Pokelib"]),
        .testTarget(
            name: "spmtestTests",
            dependencies: ["spmtest"]),
    ]
)
EOL
```

Тестирование SPM

test_spm

```
mkdir spmtest
cd spmtest

swift package init --type library

cat > Package.swift <<EOL
// swift-tools-version:5.5
import PackageDescription
let package = Package(
    name: "spmtest",
    products: [
        .library(
            name: "spmtest",
            targets: ["spmtest"]),
    ],
    dependencies: [
      .package(name: "Pokelib",
               url: "https://github.com/pokemons/pokelib",
               from: "$1")
    ],
    targets: [
        .target(
            name: "spmtest",
            dependencies: ["Pokelib"]),
        .testTarget(
            name: "spmtestTests",
            dependencies: ["spmtest"]),
    ]
)
EOL
```

Тестирование SPM

test_spm

```
mkdir spmtest
cd spmtest

swift package init --type library

cat > Package.swift <<EOL
// swift-tools-version:5.5
import PackageDescription
let package = Package(
    name: "spmtest",
    products: [
        .library(
            name: "spmtest",
            targets: ["spmtest"]),
    ],
    dependencies: [
      .package(name: "Pokelib",
               url: "https://github.com/pokemons/pokelib",
               from: "$1")
    ],
    targets: [
        .target(
            name: "spmtest",
            dependencies: ["Pokelib"]),
        .testTarget(
            name: "spmtestTests",
            dependencies: ["spmtest"]),
    ]
)
EOL
```

Тестирование SPM

test_spm

```
swift package init --type library

cat > Package.swift <<EOL
// swift-tools-version:5.5
import PackageDescription
let package = Package(
    name: "spmtest",
    products: [
        .library(
            name: "spmtest",
            targets: ["spmtest"]),
    ],
    dependencies: [
      .package(name: "Pokelib",
               url: "https://github.com/pokemons/pokelib",
               from: "$1")
    ],
    targets: [
        .target(
            name: "spmtest",
            dependencies: ["Pokelib"]),
        .testTarget(
            name: "spmtestTests",
            dependencies: ["spmtest"]),
    ]
)
EOL

cat > Sources/spmtest/spmtest.swift <<EOL
import Pokelib

public struct spmtest {
    public private(set) var text = "Hello, World!"

    public init() {
    }
}
EOL
```

Тестирование SPM

test_spm

```
swift package init --type library

cat > Package.swift <<EOL
// swift-tools-version:5.5
import PackageDescription
let package = Package(
    name: "spmtest",
    products: [
        .library(
            name: "spmtest",
            targets: ["spmtest"]),
    ],
    dependencies: [
      .package(name: "Pokelib",
               url: "https://github.com/pokemons/pokelib",
               from: "$1")
    ],
    targets: [
        .target(
            name: "spmtest",
            dependencies: ["Pokelib"]),
        .testTarget(
            name: "spmtestTests",
            dependencies: ["spmtest"]),
    ]
)
EOL

cat > Sources/spmtest/spmtest.swift <<EOL
import Pokelib

public struct spmtest {
    public private(set) var text = "Hello, World!"

    public init() {
    }
}
EOL
```

Тестирование SPM

test_spm

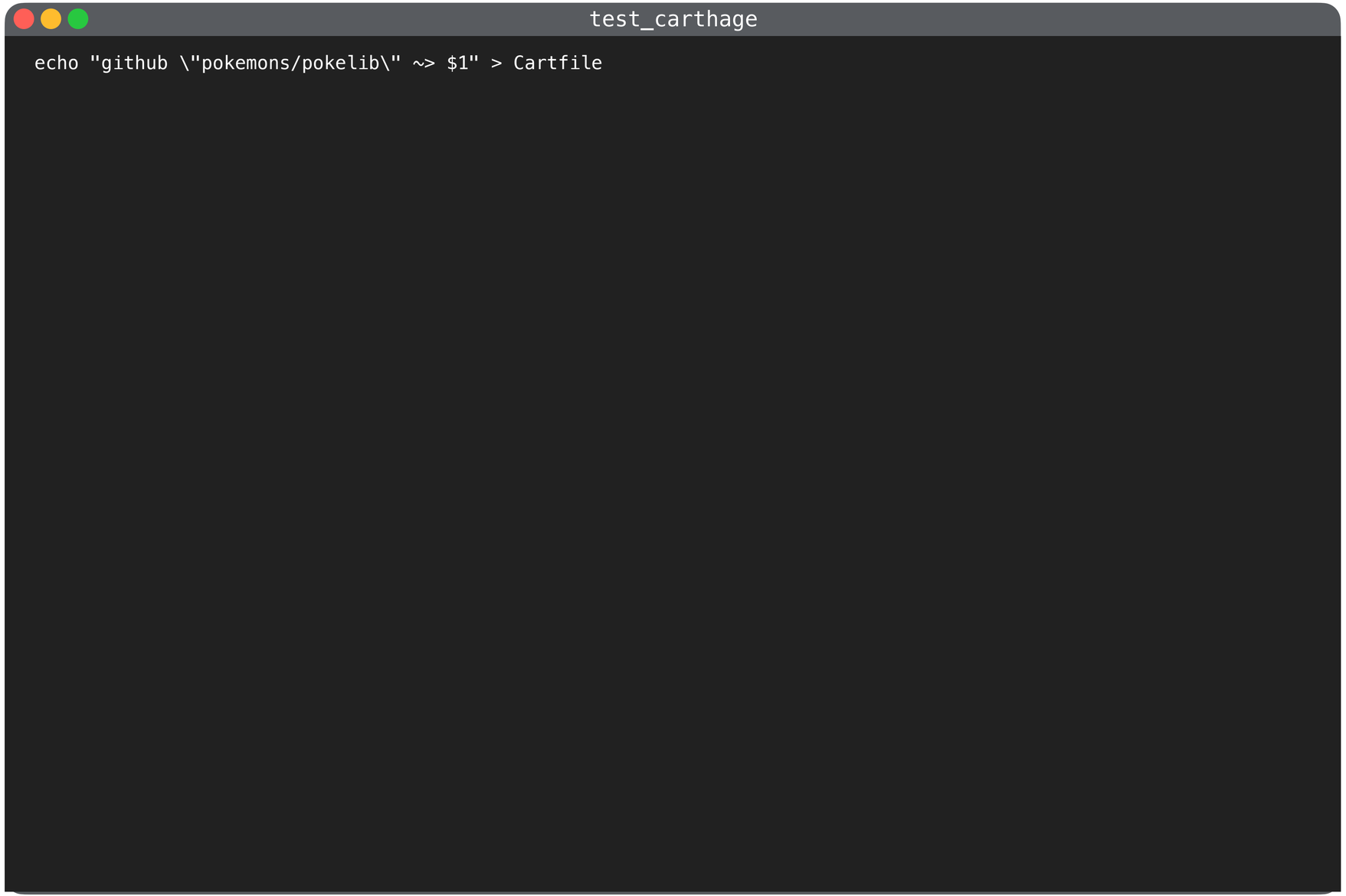
```
cat > Package.swift <<EOL
// swift-tools-version:5.5
import PackageDescription
let package = Package(
    name: "spmtest",
    products: [
        .library(
            name: "spmtest",
            targets: ["spmtest"]),
    ],
    dependencies: [
      .package(name: "Pokelib",
               url: "https://github.com/pokemons/pokelib",
               from: "$1")
    ],
    targets: [
        .target(
            name: "spmtest",
            dependencies: ["Pokelib"]),
        .testTarget(
            name: "spmtestTests",
            dependencies: ["spmtest"]),
    ]
)
EOL

cat > Sources/spmtest/spmtest.swift <<EOL
import Pokelib

public struct spmtest {
    public private(set) var text = "Hello, World!"

    public init() {
    }
}
EOL

swift package resolve
```

Тестирование SPM

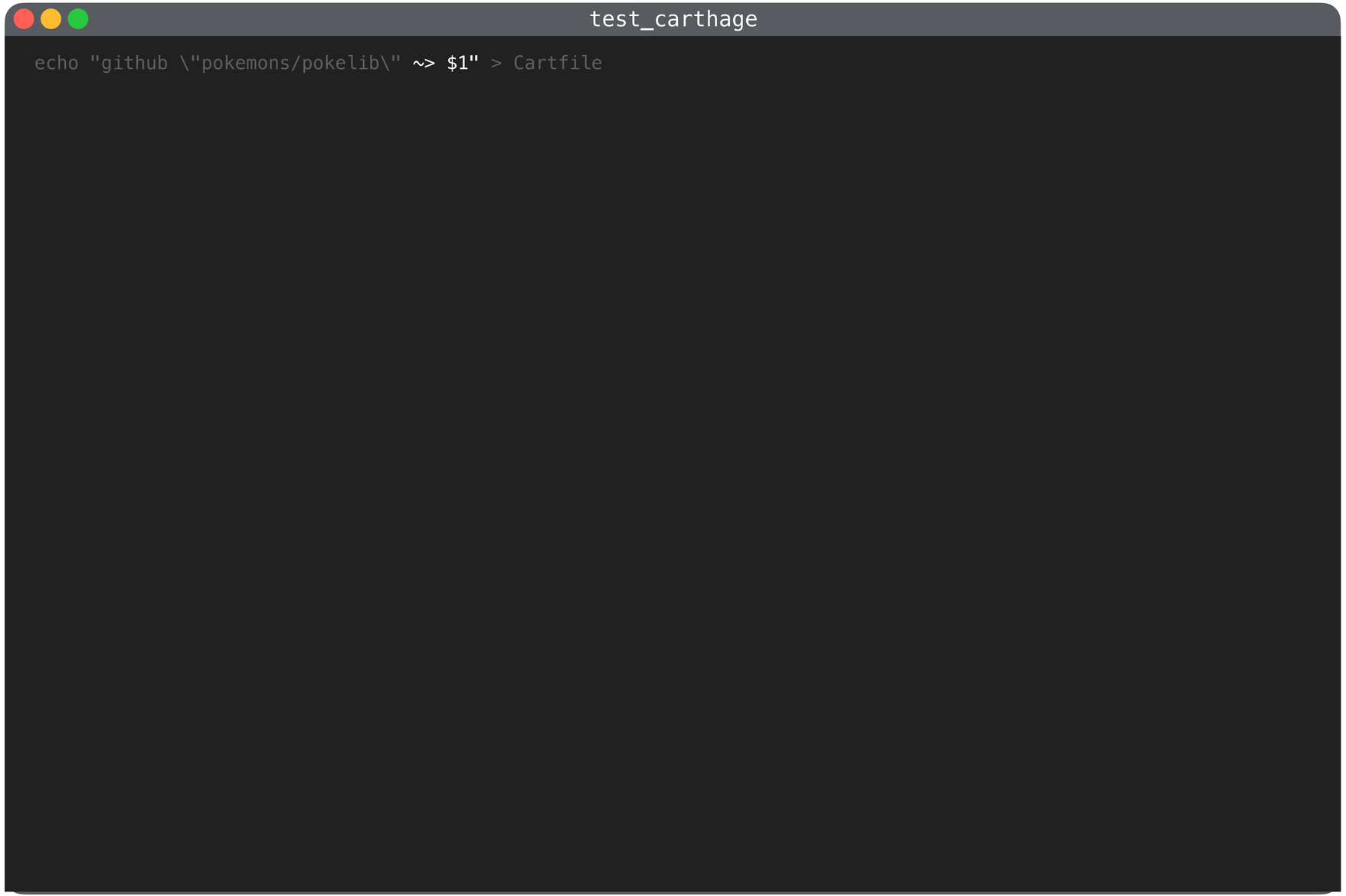
test_spm

```
let package = Package(
    name: "spmtest",
    products: [
        .library(
            name: "spmtest",
            targets: ["spmtest"]),
    ],
    dependencies: [
      .package(name: "Pokelib",
               url: "https://github.com/pokemons/pokelib",
               from: "$1")
    ],
    targets: [
        .target(
            name: "spmtest",
            dependencies: ["Pokelib"]),
        .testTarget(
            name: "spmtestTests",
            dependencies: ["spmtest"]),
    ]
)
EOL

cat > Sources/spmtest/spmtest.swift <<EOL
import Pokelib

public struct spmtest {
    public private(set) var text = "Hello, World!"

    public init() {
    }
}
EOL

swift package resolve

xcodebuild \
 -scheme spmtest \
 -destination 'platform=iOS Simulator,name=iPhone 12,OS=15.0'
```

Тестирование Carthage



Тестирование Carthage



Тестирование Carthage

test_carthage

```
echo "github \"pokemons/pokelib\" ~> $1" > Cartfile

carthage update
./Carthage/Checkouts/pokelib/carthage-prebuild
carthage build --use-xcframeworks --platform ios
```

Тестирование Carthage

test_carthage

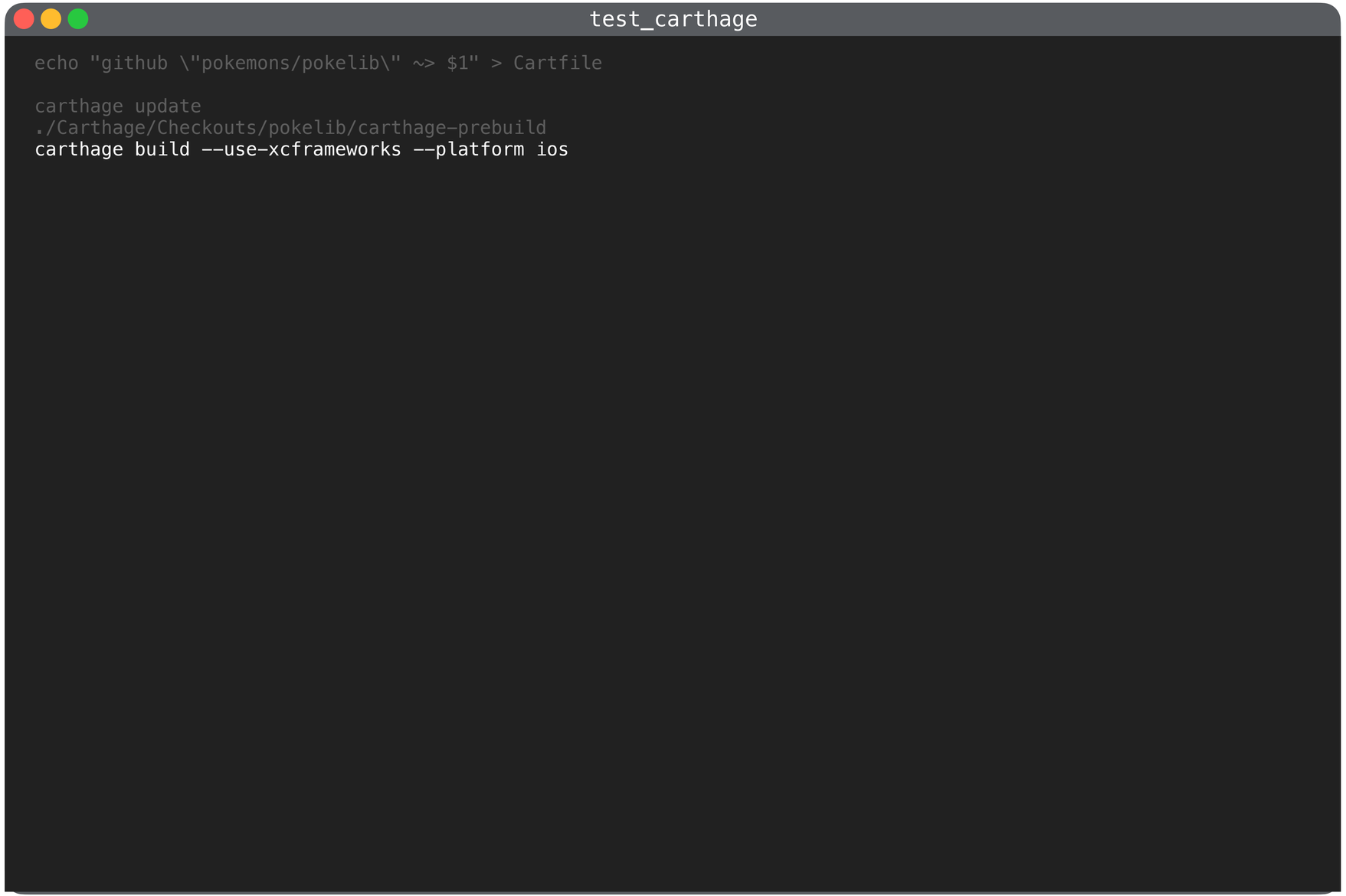
```
echo "github \"pokemons/pokelib\" ~> $1" > Cartfile

carthage update
./Carthage/Checkouts/pokelib/carthage-prebuild
carthage build --use-xcframeworks --platform ios
```

Тестирование Carthage

test_carthage

```
echo "github \"pokemons/pokelib\" ~> $1" > Cartfile

carthage update
./Carthage/Checkouts/pokelib/carthage-prebuild
carthage build --use-xcframeworks --platform ios
```

Тестирование Carthage



Тестирование Carthage

test_carthage

```
echo "github \"pokemons/pokelib\" ~> $1" > Cartfile

carthage update
./Carthage/Checkouts/pokelib/carthage-prebuild
carthage build --use-xcframeworks --platform ios

mkdir carthagetest
```

Тестирование Carthage

test_carthage

```
echo "github \"pokemons/pokelib\" ~> $1" > Cartfile

carthage update
./Carthage/Checkouts/pokelib/carthage-prebuild
carthage build --use-xcframeworks --platform ios

mkdir carthagetest

cp -r Carthage/Build/Logging.xcframework ./carthagetest/Logging.xcframework
cp -r Carthage/Build/Pokelib.xcframework ./carthagetest/Pokelib.xcframework
```

Тестирование Carthage

test_carthage

```
echo "github \"pokemons/pokelib\" ~> $1" > Cartfile

carthage update
./Carthage/Checkouts/pokelib/carthage-prebuild
carthage build --use-xcframeworks --platform ios

mkdir carthagetest

cp -r Carthage/Build/Logging.xcframework ./carthagetest/Logging.xcframework
cp -r Carthage/Build/Pokelib.xcframework ./carthagetest/Pokelib.xcframework

cd carthagetest/
swift package init --type library
```

Тестирование Carthage

test_carthage

```
cp -r Carthage/Build/Logging.xcframework ./carthagetest/Logging.xcframework
cp -r Carthage/Build/Pokelib.xcframework ./carthagetest/Pokelib.xcframework

cd carthagetest/
swift package init --type library

cat > Package.swift <<EOL
// swift-tools-version:5.5
import PackageDescription
let package = Package(
    name: "carthagetest",
    products: [
        .library(
            name: "carthagetest",
            targets: ["carthagetest"]),
    ],
    dependencies: [
    ],
    targets: [
        .binaryTarget(name: "Logging",
                      path: "Logging.xcframework"),
        .binaryTarget(name: "Pokelib",
                      path: "Pokelib.xcframework"),
        .target(
            name: "carthagetest",
            dependencies: ["Logging", "Pokelib"]),
        .testTarget(
            name: "carthagetestTests",
            dependencies: ["carthagetest"]),
    ]
)
EOL
```

Тестирование Carthage

test_carthage

```
cp -r Carthage/Build/Logging.xcframework ./carthagetest/Logging.xcframework
cp -r Carthage/Build/Pokelib.xcframework ./carthagetest/Pokelib.xcframework

cd carthagetest/
swift package init --type library

cat > Package.swift <<EOL
// swift-tools-version:5.5
import PackageDescription
let package = Package(
    name: "carthagetest",
    products: [
        .library(
            name: "carthagetest",
            targets: ["carthagetest"]),
    ],
    dependencies: [
    ],
    targets: [
        .binaryTarget(name: "Logging",
                      path: "Logging.xcframework"),
        .binaryTarget(name: "Pokelib",
                      path: "Pokelib.xcframework"),
        .target(
            name: "carthagetest",
            dependencies: ["Logging", "Pokelib"]),
        .testTarget(
            name: "carthagetestTests",
            dependencies: ["carthagetest"]),
    ]
)
EOL
```

Тестирование Carthage

test_carthage

```
cp -r Carthage/Build/Logging.xcframework ./carthagetest/Logging.xcframework
cp -r Carthage/Build/Pokelib.xcframework ./carthagetest/Pokelib.xcframework

cd carthagetest/
swift package init --type library

cat > Package.swift <<EOL
// swift-tools-version:5.5
import PackageDescription
let package = Package(
    name: "carthagetest",
    products: [
        .library(
            name: "carthagetest",
            targets: ["carthagetest"]),
    ],
    dependencies: [
    ],
    targets: [
        .binaryTarget(name: "Logging",
                      path: "Logging.xcframework"),
        .binaryTarget(name: "Pokelib",
                      path: "Pokelib.xcframework"),
        .target(
            name: "carthagetest",
            dependencies: ["Logging", "Pokelib"]),
        .testTarget(
            name: "carthagetestTests",
            dependencies: ["carthagetest"]),
    ]
)
EOL
```

Тестирование Carthage

test_carthage

```
cat > Package.swift <<EOL
// swift-tools-version:5.5
import PackageDescription
let package = Package(
    name: "carthagetest",
    products: [
        .library(
            name: "carthagetest",
            targets: ["carthagetest"]),
    ],
    dependencies: [
    ],
    targets: [
        .binaryTarget(name: "Logging",
                      path: "Logging.xcframework"),
        .binaryTarget(name: "Pokelib",
                      path: "Pokelib.xcframework"),
        .target(
            name: "carthagetest",
            dependencies: ["Logging", "Pokelib"]),
        .testTarget(
            name: "carthagetestTests",
            dependencies: ["carthagetest"]),
    ]
)
EOL

cat > Sources/carthagetest/carthagetest.swift <<EOL
import Pokelib

public struct carthagetest {
    public private(set) var text = "Hello, World!"

    public init() {
    }
}
EOL
```

Тестирование Carthage

test_carthage

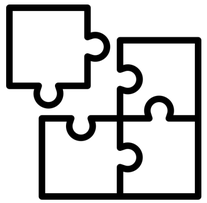
```
cat > Package.swift <<EOL
// swift-tools-version:5.5
import PackageDescription
let package = Package(
    name: "carthagetest",
    products: [
        .library(
            name: "carthagetest",
            targets: ["carthagetest"]),
    ],
    dependencies: [
    ],
    targets: [
        .binaryTarget(name: "Logging",
                      path: "Logging.xcframework"),
        .binaryTarget(name: "Pokelib",
                      path: "Pokelib.xcframework"),
        .target(
            name: "carthagetest",
            dependencies: ["Logging", "Pokelib"]),
        .testTarget(
            name: "carthagetestTests",
            dependencies: ["carthagetest"]),
    ]
)
EOL

cat > Sources/carthagetest/carthagetest.swift <<EOL
import Pokelib

public struct carthagetest {
    public private(set) var text = "Hello, World!"

    public init() {
    }
}
EOL
```

Тестирование Carthage

test_carthage

```
    name: "carthagetest",
    products: [
        .library(
            name: "carthagetest",
            targets: ["carthagetest"]),
    ],
    dependencies: [
    ],
    targets: [
        .binaryTarget(name: "Logging",
                      path: "Logging.xcframework"),
        .binaryTarget(name: "Pokelib",
                      path: "Pokelib.xcframework"),
        .target(
            name: "carthagetest",
            dependencies: ["Logging", "Pokelib"]),
        .testTarget(
            name: "carthagetestTests",
            dependencies: ["carthagetest"]),
    ]
)
EOL

cat > Sources/carthagetest/carthagetest.swift <<EOL
import Pokelib

public struct carthagetest {
    public private(set) var text = "Hello, World!"

    public init() {
    }
}
EOL

xcodebuild \
  -scheme carthagetest
  -destination 'platform=iOS Simulator,name=iPhone 12,OS=15.0'
```

Заключение



Заключение

Теперь мы умеем

- Распространять библиотеку через все популярные менеджеры зависимостей одновременно

Заключение

Теперь мы умеем

- Распространять библиотеку через все популярные менеджеры зависимостей одновременно
- Разделять библиотеку на модули с поддержкой во всех менеджерах

Заключение

Теперь мы умеем

- Распространять библиотеку через все популярные менеджеры зависимостей одновременно
- Разделять библиотеку на модули с поддержкой во всех менеджерах
- Релизить библиотеку с сохранением целостности версий

Заключение

Теперь мы умеем

- Распространять библиотеку через все популярные менеджеры зависимостей одновременно
- Разделять библиотеку на модули с поддержкой во всех менеджерах
- Релизить библиотеку с сохранением целостности версий
- Тестировать корректную работу всех менеджеров зависимостей перед каждым релизом

Заключение

Теперь мы умеем

- Распространять библиотеку через все популярные менеджеры зависимостей одновременно
- Разделять библиотеку на модули с поддержкой во всех менеджерах
- Релизить библиотеку с сохранением целостности версий
- Тестировать корректную работу всех менеджеров зависимостей перед каждым релизом

~~и ждём когда всё переедет в SPM 🙃~~

Контакты

@vfitc

VladislavFitz