MOBILE TALKS

Getting started with iOS modularization

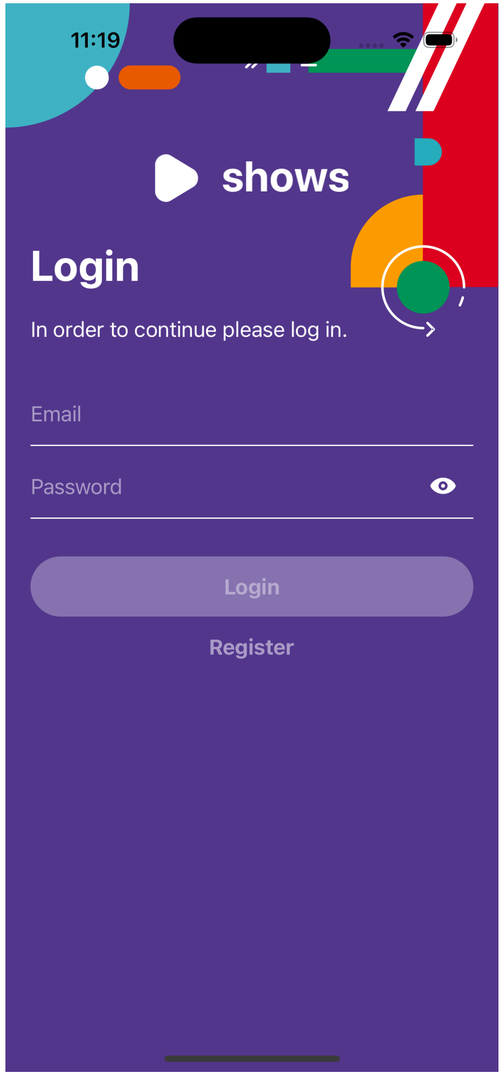
LUKA TERZIĆ
IOS ENGINEER @ INFINUM

Talks

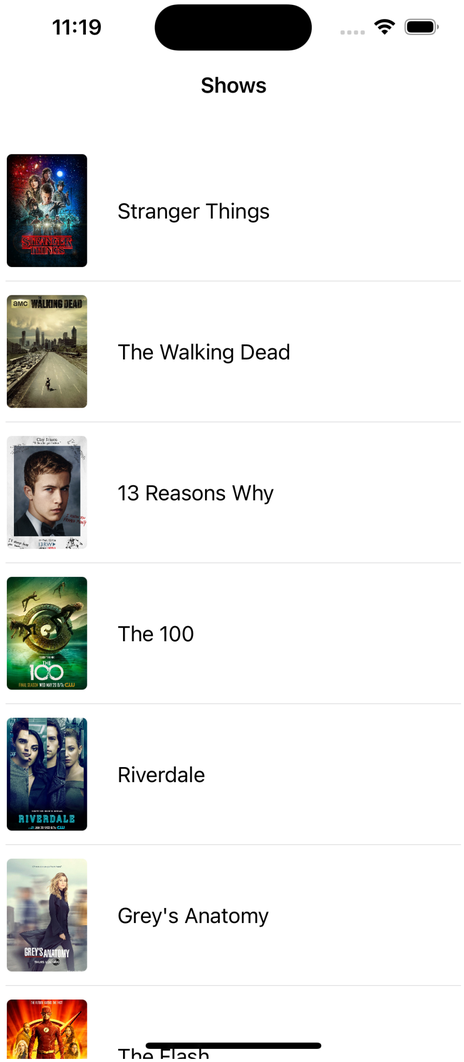
neverstop

01 Why modular architecture?

TVShows



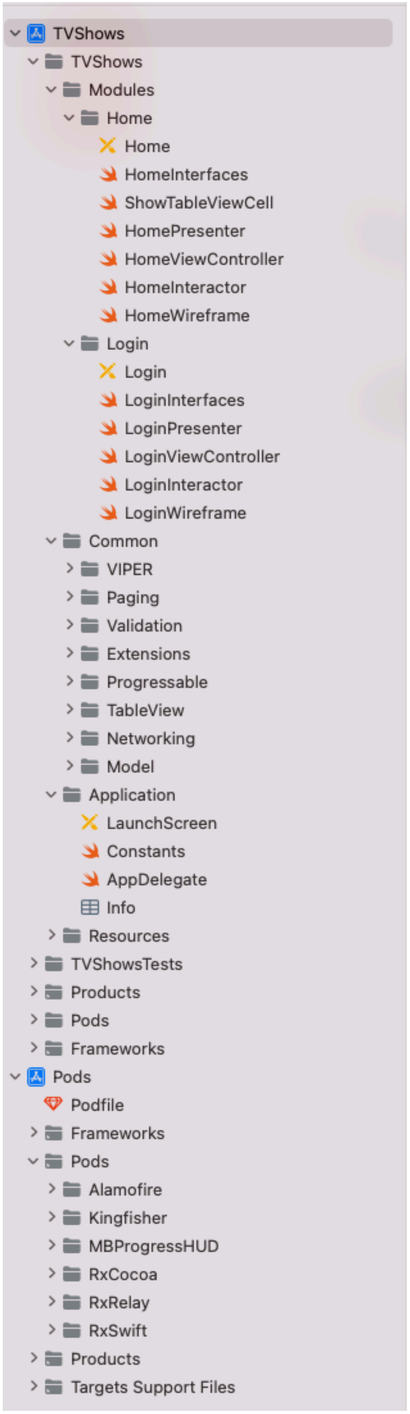
LOGIN



HOME

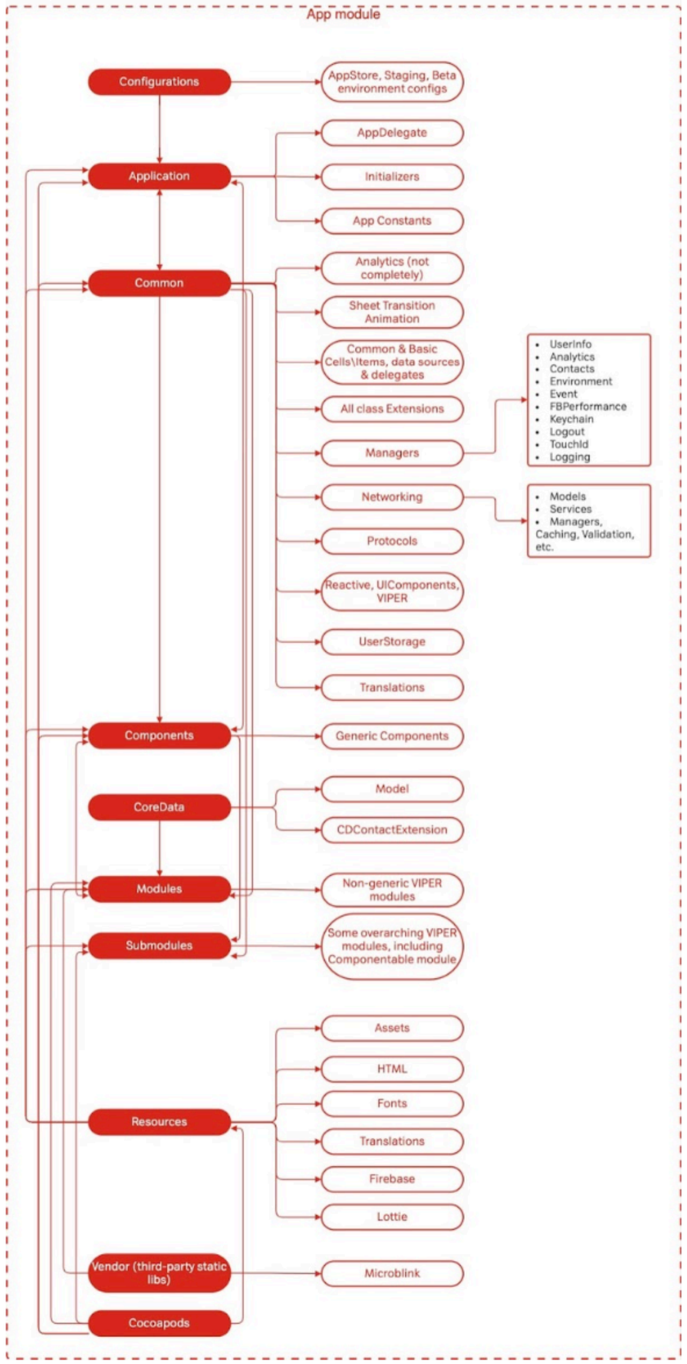
Monolithic architecture

- Entirety of the app is contained in only one module.
- Only one test target.

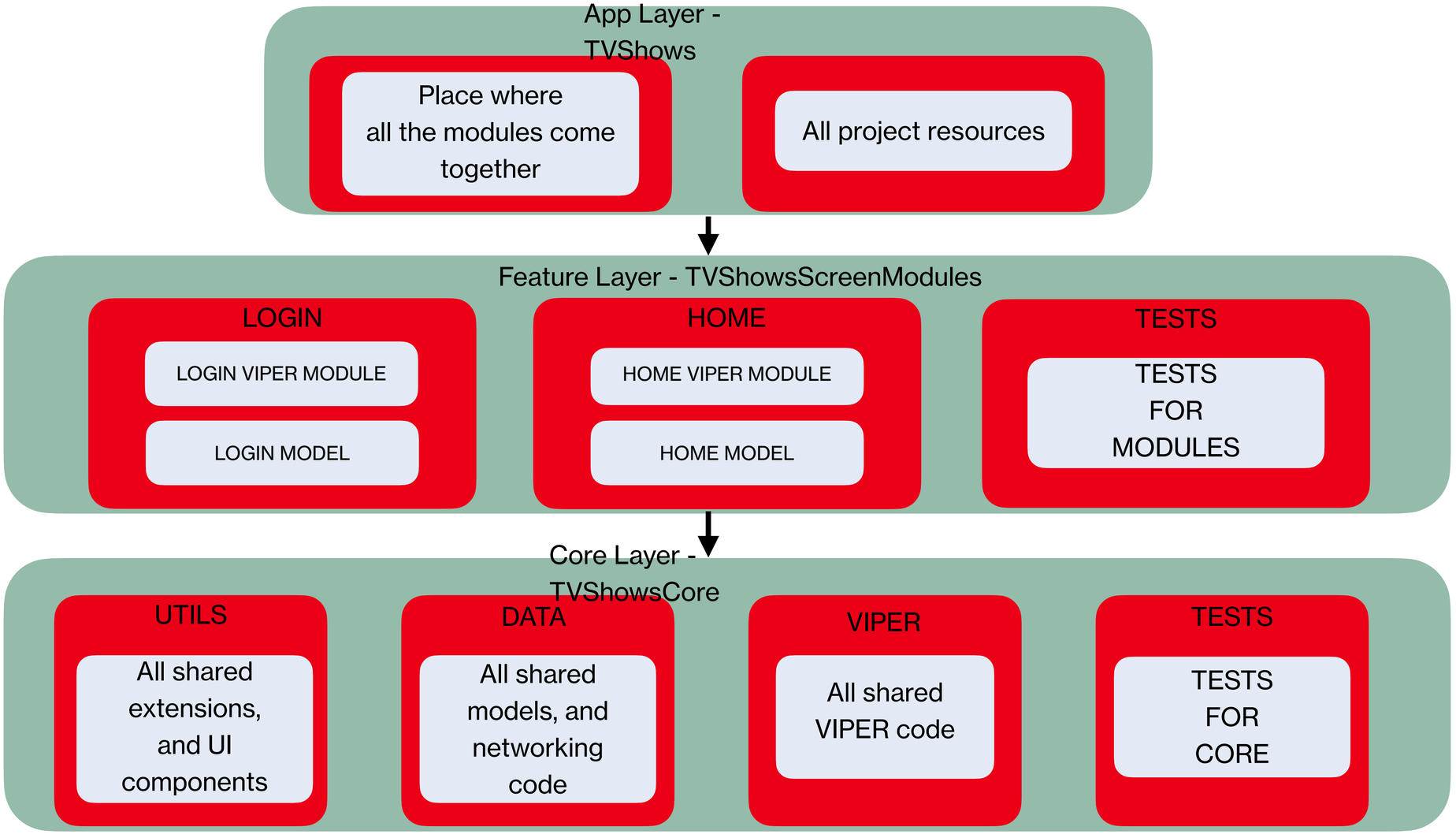


Monolithic architecture

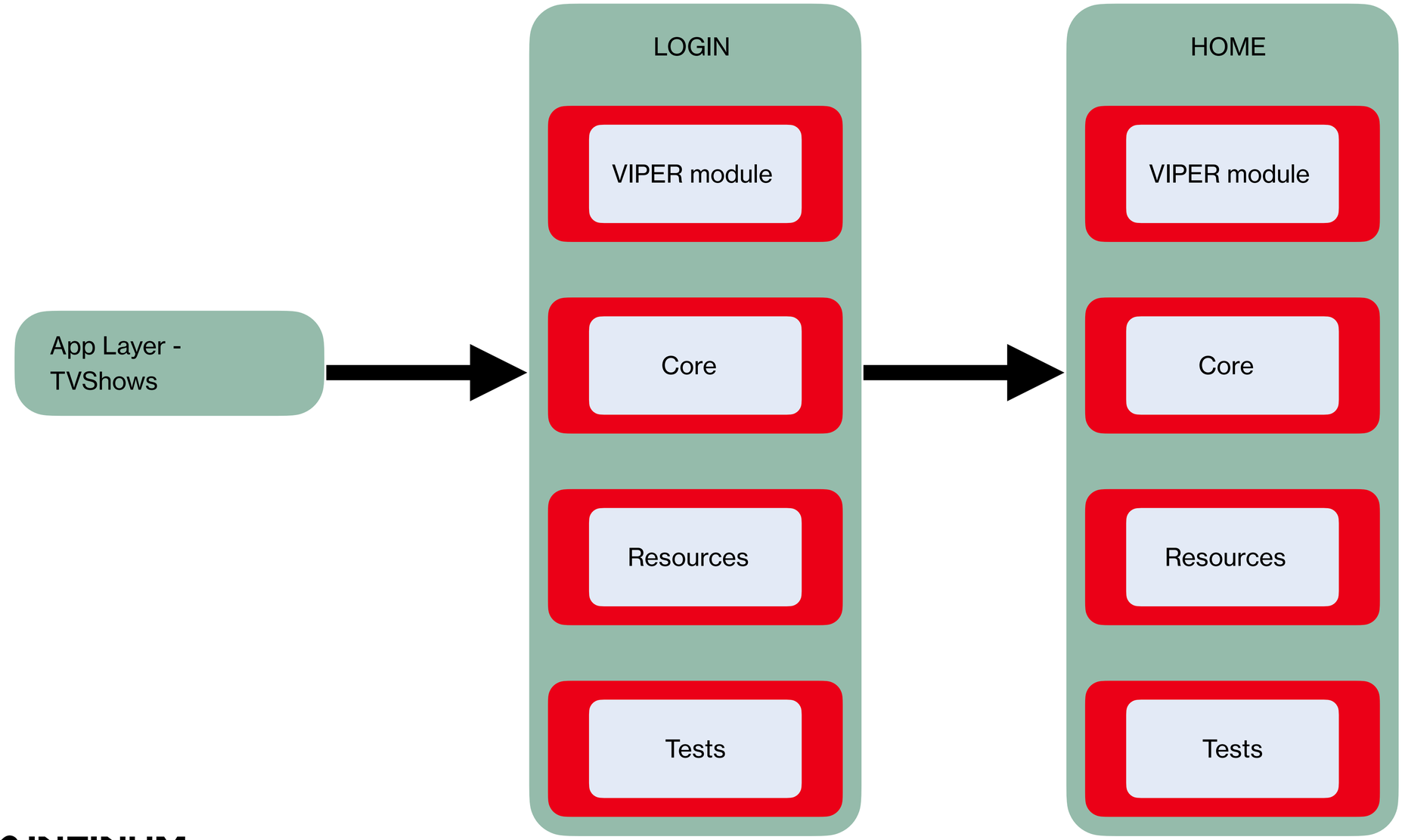
- For an initial scope, the monolithic architecture seems appropriate, but...
- As time passes, it gets difficult to maintain a clean structure.
- The addition itself can become more difficult to implement.



Horizontal Modular architecture

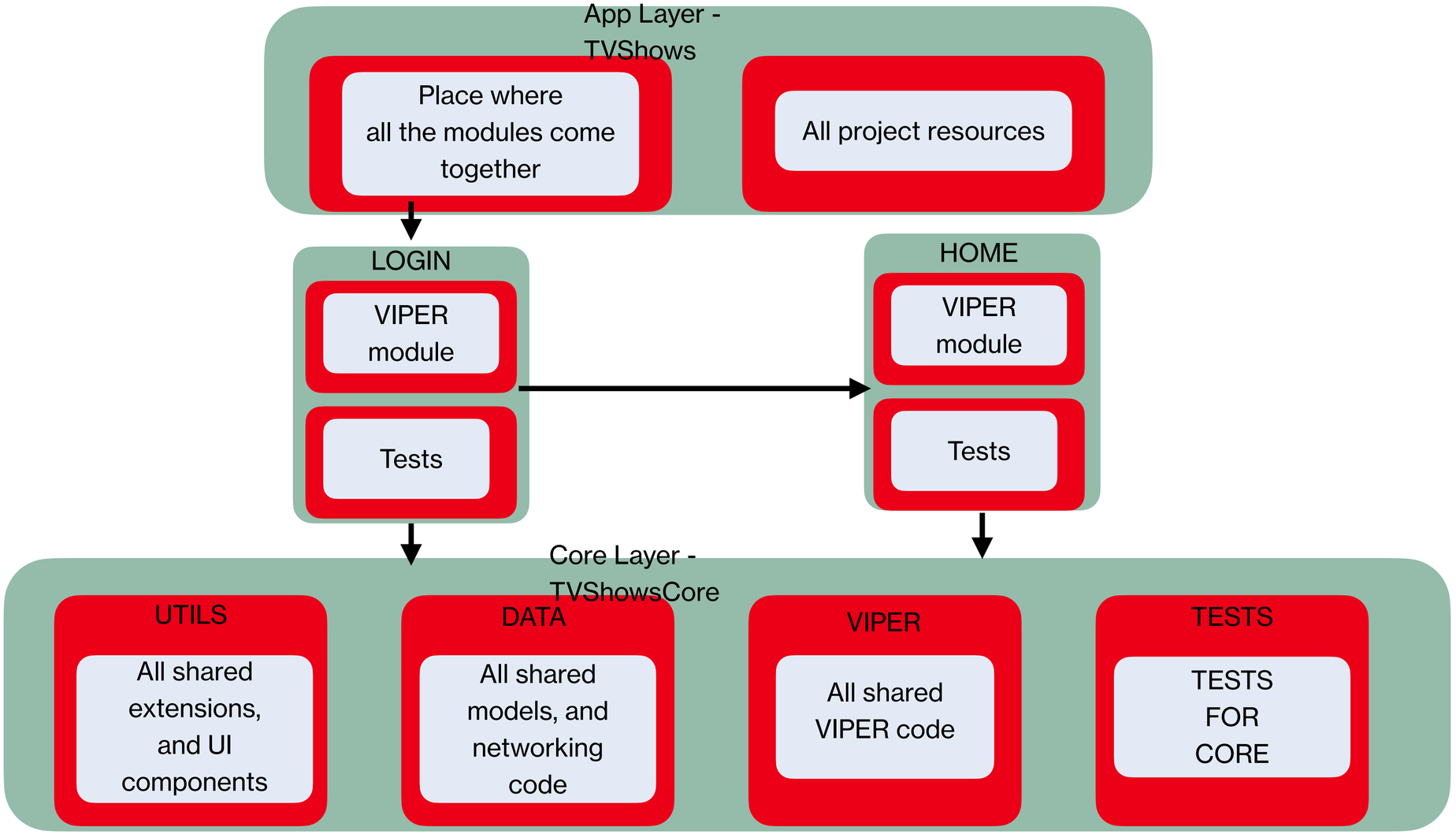


Vertical Modular architecture



We can go hybrid!

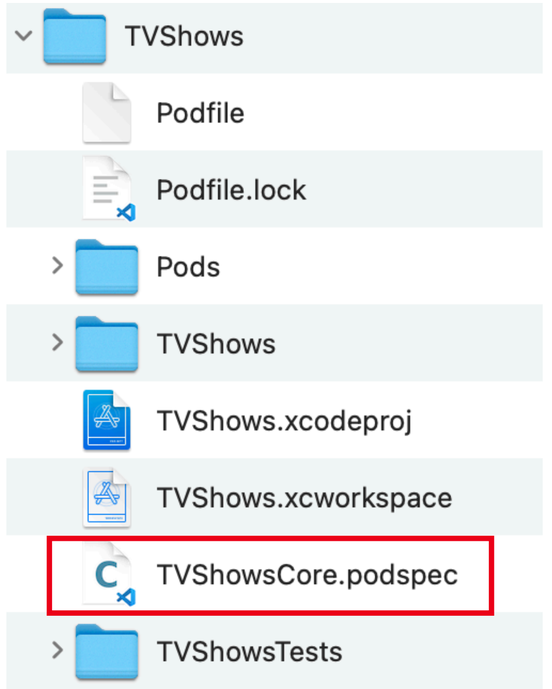
Hybrid Modular architecture



02 Modularization with CocoaPods

TVShowsCore

- Start by adding a ".podspec" file to your project repository.

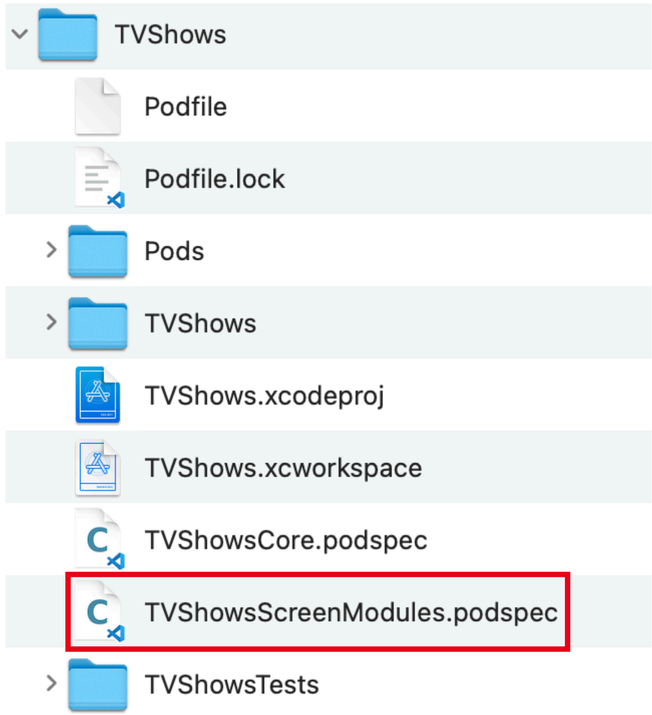


TVShowsCore.podspec

```ruby
Pod::Spec.new do |spec|
  spec.name         = "TVShowsCore"
  spec.version      = "0.0.1"
  spec.summary      = "TVShowsCore module contains all the code that is reused by components and screens"
  spec.homepage     = "https://github.com/terzicluka/iOS-App-Modularization"
  spec.source       = { :git => "https://github.com/terzicluka/iOS-App-Modularization", :tag => "#{spec.version}" }
  spec.license      = "MIT"
  spec.authors      = { "Luka Terzić" => "luka.terzic@infinum.com" }
  spec.platform     = :ios, "16.0"

  spec.subspec 'Utils' do |utils|
    utils.dependency "RxSwift"
    utils.dependency "RxCocoa"
    utils.dependency "Alamofire"
    utils.dependency "Kingfisher"

    utils.source_files =
                        # UI
                        “TVShows/Common/Extensions/UIImageViewExtension.swift”,…
  end

  spec.subspec 'Data' do |data|
    data.dependency "TVShowsCore/Utils"

    data.source_files =
                      "TVShows/Common/Model/APIError.swift",
                      "TVShows/Common/Networking/**/*.swift",
                      “TVShows/Application/Constants.swift”…
  end

  spec.subspec 'VIPER' do |viper|
    viper.dependency "RxSwift"
    viper.dependency "RxCocoa"
    viper.dependency "MBProgressHUD"

    viper.source_files =
                      "TVShows/Common/VIPER/**/*.swift"

    viper.exclude_files =
                      "TVShows/Common/VIPER/UIStoryboardExtension.swift" # Already included in Utils
  end

  spec.test_spec 'Tests' do |test_spec|
    test_spec.source_files = 'TVShowsTests/Extensions/**/*'
  end

end
```

TVShowsScreenModules

- Start by adding a “.podspec” file to your project repository.



TVShowsScreenModules.podspec

```
Pod::Spec.new do |spec|
  spec.name         = "TVShowsScreenModules"
  spec.version      = "0.0.1"
  spec.summary      = "TVShowsScreenModules contains all screens"
  spec.homepage     = "https://github.com/terzicluka/iOS-App-Modularization"
  spec.source       = { :git => "https://github.com/terzicluka/iOS-App-Modularization", :tag =>
"#{spec.version}" }
  spec.license      = "MIT"
  spec.authors      = { "Luka Terzić"   => "luka.terzic@infinum.com" }
  spec.platform     = :ios, "16.0"

  spec.subspec 'Common' do |common|
    common.dependency "TVShowsCore"
  end

  spec.subspec 'Login' do |login|
    login.dependency "TVShowsScreenModules/Common"
    login.source_files =
                              "TVShows/Modules/Login/**/*.swift",
                              "TVShows/Common/Model/User.swift",
                              "TVShows/Common/Model/MetaData.swift"
    login.resources =
                              "TVShows/Modules/Login/**/*.{xib,storyboard}"
  end

  spec.subspec 'Home' do |home|
    home.dependency "TVShowsScreenModules/Common"
    home.source_files =
                              "TVShows/Modules/Home/**/*.swift",
                              "TVShows/Common/Model/Show.swift"
    home.resources =
                              "TVShows/Modules/Home/**/*.{xib,storyboard}"
  end

  spec.test_spec 'Tests' do |test_spec|
    test_spec.source_files = 'TVShowsTests/Modules/**/*'
  end

end
```

What now?...

- Lets edit our podfile!

Old podfile

```
platform :ios, '16.0'
use_frameworks!

def ui
  pod 'MBProgressHUD', '~> 1.2.0'
end

def reactive
  pod 'RxSwift', '~> 6.5'
  pod 'RxCocoa', '~> 6.5'
end

def networking
  pod 'Kingfisher', '~> 7.4'
  pod 'Alamofire', '~> 5.6'
end

def shared
  ui
  reactive
  networking
end

target 'TVShows' do
  shared
end

post_install do |installer|
    installer.generated_projects.each do |project|
          project.targets.each do |target|
              target.build_configurations.each do |config|
                  config.build_settings['IPHONEOS_DEPLOYMENT_TARGET'] = '16.0'
               end
          end
   end
end
```

New podfile

```
platform :ios, '16.0'
use_frameworks!

def ui
  pod 'MBProgressHUD', '~> 1.2.0'
end

def reactive
  pod 'RxSwift', '~> 6.5'
  pod 'RxCocoa', '~> 6.5'
end

def networking
  pod 'Kingfisher', '~> 7.4'
  pod 'Alamofire', '~> 5.6'
end

def local
  pod 'TVShowsCore', :path => './', :testspecs => ['Tests']
  pod 'TVShowsScreenModules', :path => './', :testspecs => ['Tests']
end

def shared
  ui
  reactive
  networking
  local
end

target 'TVShows' do
  shared
end

post_install do |installer|
    installer.generated_projects.each do |project|
          project.targets.each do |target|
              target.build_configurations.each do |config|
                  config.build_settings['IPHONEOS_DEPLOYMENT_TARGET'] = '16.0'
               end
          end
   end
end
```

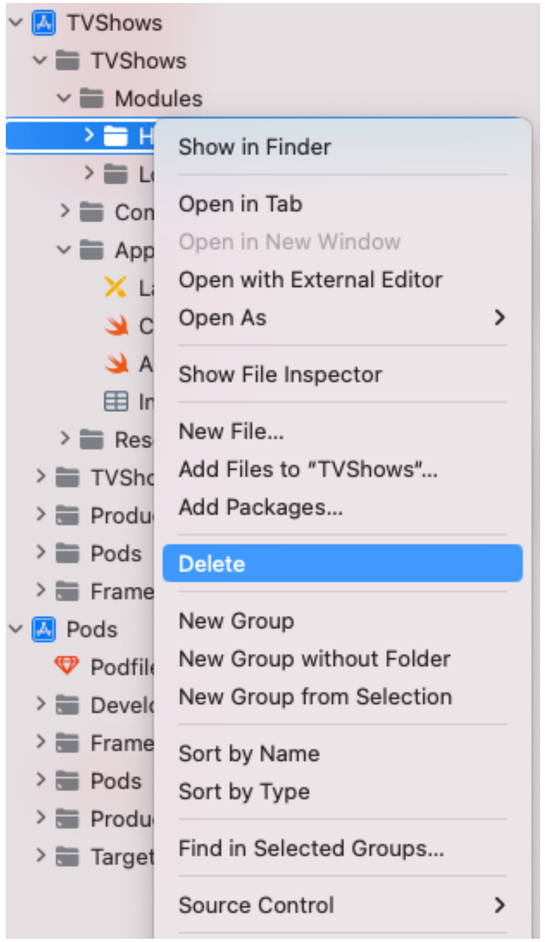
Are we finished?

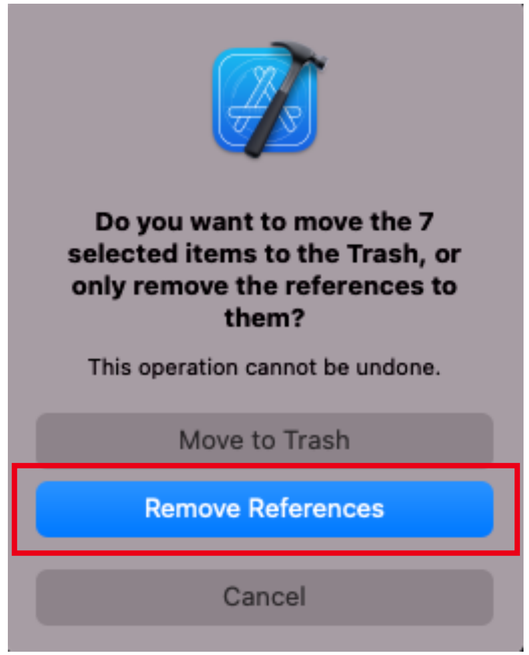
- Run pod install!

```
$ pod install
Analyzing dependencies
Downloading dependencies
Installing TVShowsCore (0.0.1)
Installing TVShowsScreenModules (0.0.1)
Generating Pods project
Integrating client project
Pod installation complete! There are 9 dependencies from the Podfile and 8 total
 pods installed.
```

Are we finished?

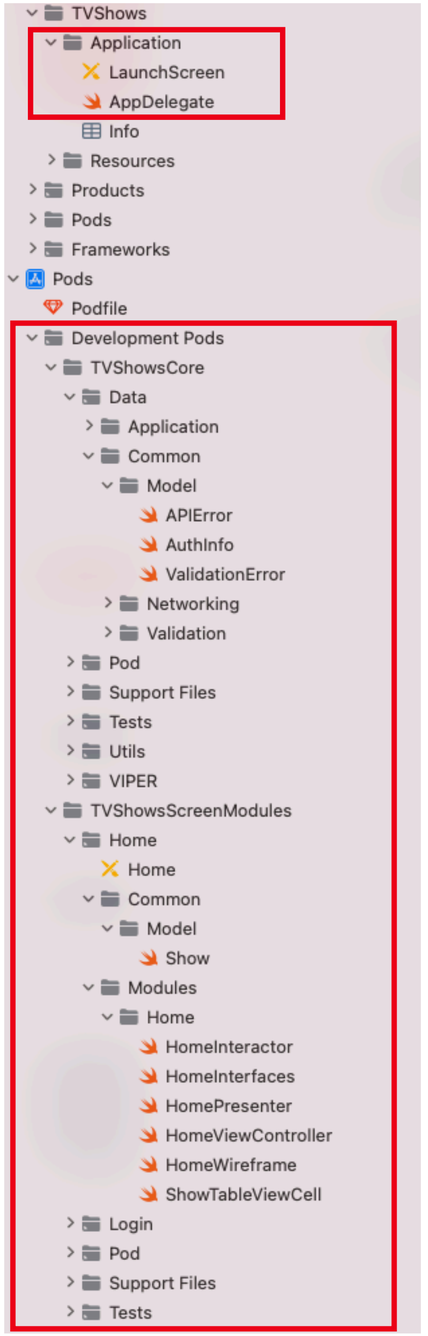
- Not quite...
- Lets remove all references to files from the base project.





Are we finished?

- We have our desired architecture!!!



Are we finished?

- But wait...

```swift
@main
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
        [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        window = UIWindow(frame: UIScreen.main.bounds)

        let navigationController = UINavigationController()
        let wireframe = LoginWireframe()
        navigationController.setRootWireframe(wireframe)

        window?.rootViewController = navigationController
        window?.makeKeyAndVisible()

        return true
    }

}
```

Cannot find 'LoginWireframe' in scope

Value of type 'UINavigationController' has no member 'setRootWiref...

Are we finished?

- Okay, simple fix...

```
import UIKit
import TVShowsScreenModules

@main
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
        [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        window = UIWindow(frame: UIScreen.main.bounds)

        let navigationController = UINavigationController()
        let wireframe = LoginWireframe()                                  Cannot find 'LoginWireframe' in scope
        navigationController.setRootWireframe(wireframe)  Value of type 'UINavigationController' has no member 'setRootWiref...

        window?.rootViewController = navigationController
        window?.makeKeyAndVisible()

        return true
    }

}
```

Are we finished?

- Not quite simple...

```
import UIKit
import RxSwift
import RxCocoa

final class LoginWireframe: BaseWireframe<LoginViewController> {          Cannot find type 'BaseWireframe' in sc

    // MARK: - Constants -

    private enum Constants {
        enum Storyboard {
            static let login = "Login"
        }
    }

    // MARK: - Private properties -

    private let storyboard = UIStoryboard(name: Constants.Storyboard.login, bundle: nil)

    // MARK: - Module setup -

    init() {
        let moduleViewController = storyboard.instantiateViewController(ofType: LoginViewController.self)  2
        super.init(viewController: moduleViewController)          'super' members cannot be referenced in a root c

        let interactor = LoginInteractor()
        let presenter = LoginPresenter(view: moduleViewController, interactor: interactor, wireframe: self)
        moduleViewController.presenter = presenter
    }

}
```

Are we finished?

Are we finished?

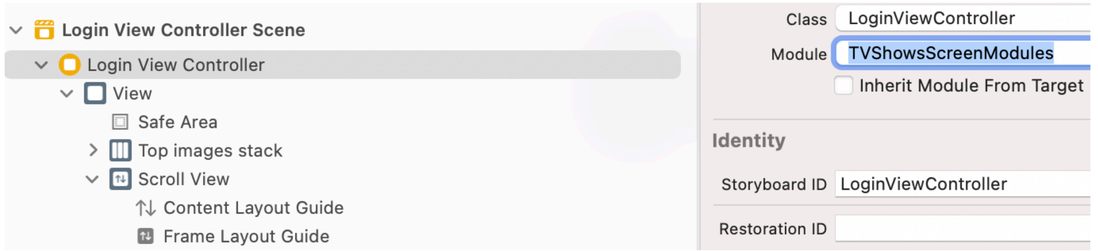
- App crashes on startup...
- The storyboard files don't belong to the main bundle anymore.

```
private let storyboard = UIStoryboard(
    name: Constants.Storyboard.login,
    bundle: Bundle(for: LoginWireframe.self)
    )
```

```
import UIKit
import TVShowsScreenModules

@main  ≡  Thread 1: "Could not find a storyboard named 'Login' in bundle NSBun...
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    func application(_ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions:
        [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        window = UIWindow(frame: UIScreen.main.bounds)

        let navigationController = UINavigationController()
        let wireframe = LoginWireframe()
        navigationController.setRootWireframe(wireframe)

        window?.rootViewController = navigationController
        window?.makeKeyAndVisible()

        return true
    }

}
```

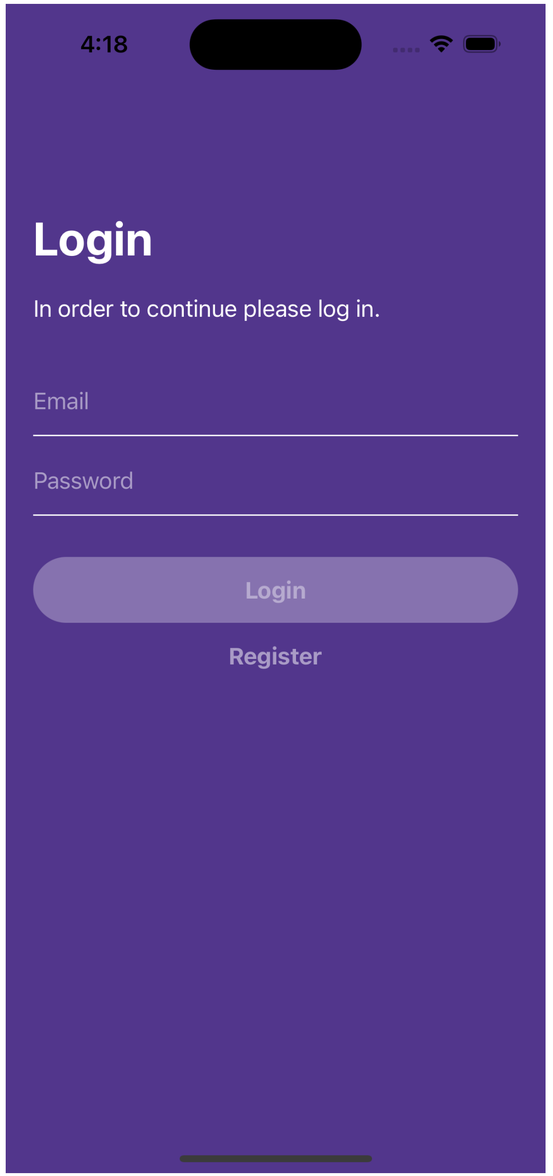
Are we finished?

- Wait we didn’t fix the issue again!?
- Make sure to change the module in all of the storyboard files.



Are we finished?

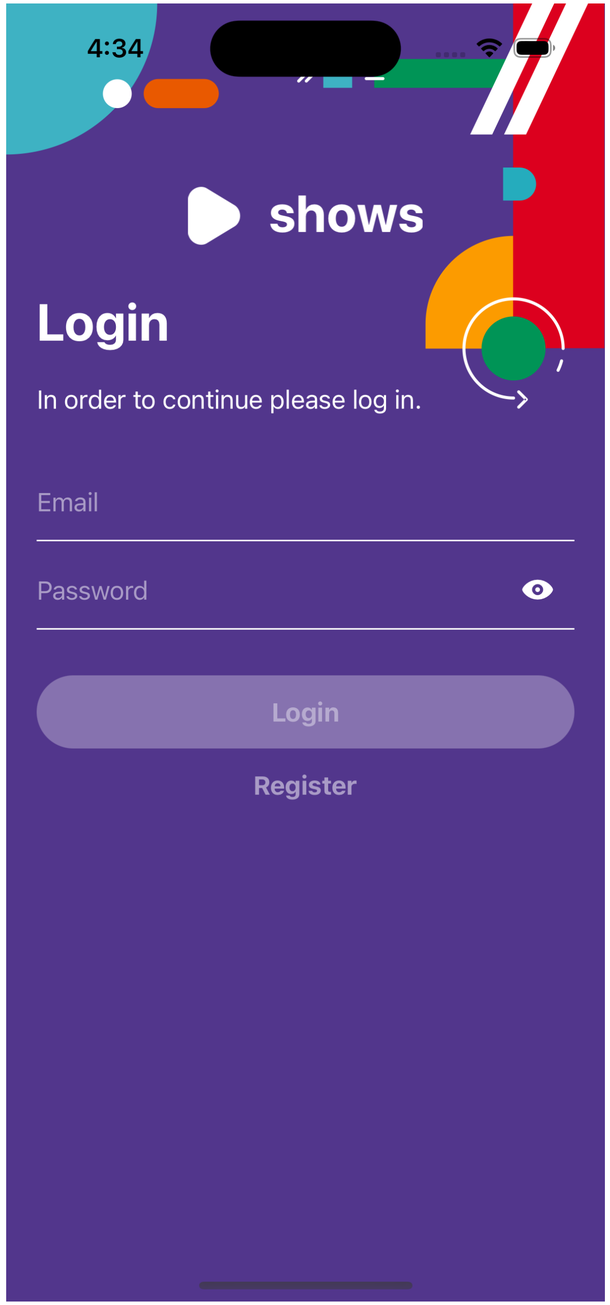
- But now the images aren’t showing...
- Image needs to be located in the same bundle as the xib/storyboard!
- Set all of your images through code!!!



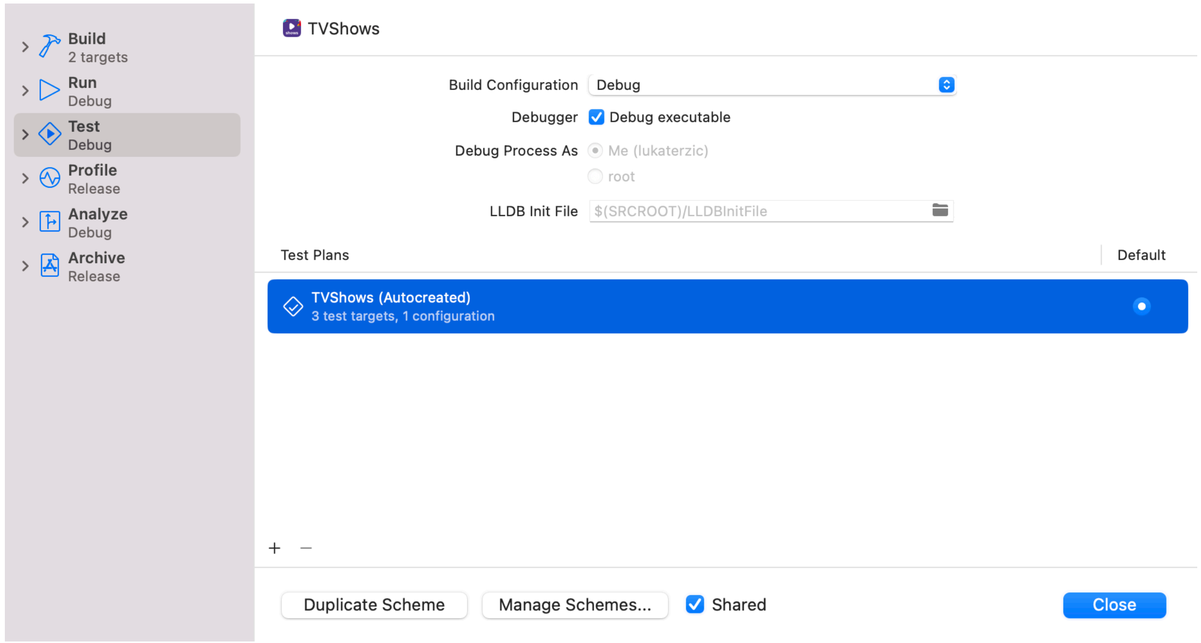
```swift
public extension UIImageView {

    private enum Constants {
        static var imageKey = "TVShowsResourceBundleImage"
    }

    @IBInspectable var tvShowsImageName: String? {
        get {
            objc_getAssociatedObject(self, &Constants.imageKey) as? String
        }

        set(imageName) {
            objc_setAssociatedObject(self, &Constants.imageKey, imageName, .OBJC_ASSOCIATION_RETAIN_NONATOMIC)

            guard let imageName = imageName,
                  let image = UIImage(named: imageName, in: .main, compatibleWith: nil)
            else {
                self.image = nil
                return
            }

            self.image = image
        }
    }
}
```

Are we finished?

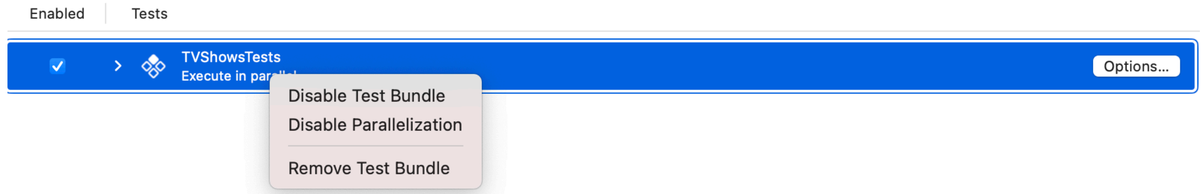
- Yes!!! We are!!!! 🥳



Testing

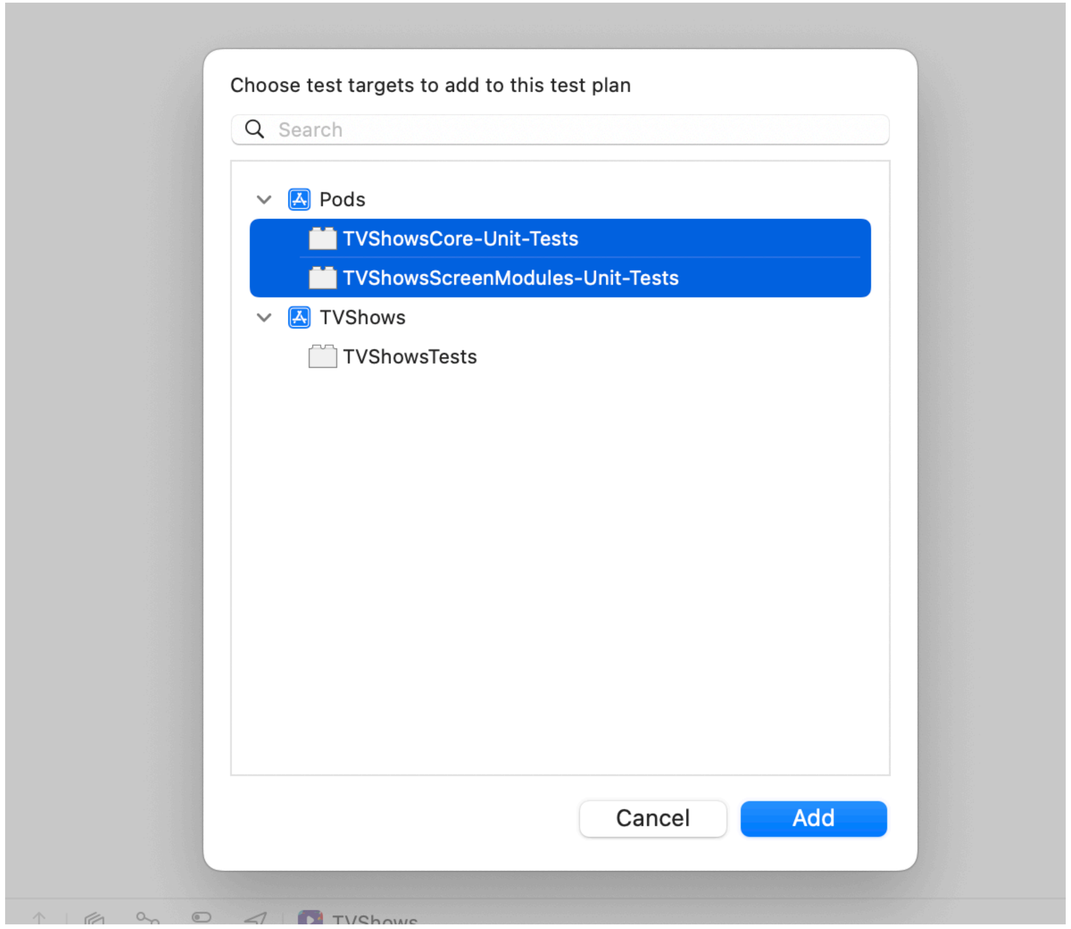


Step 1. Edit test scheme



Step 2. Delete old test target

Testing

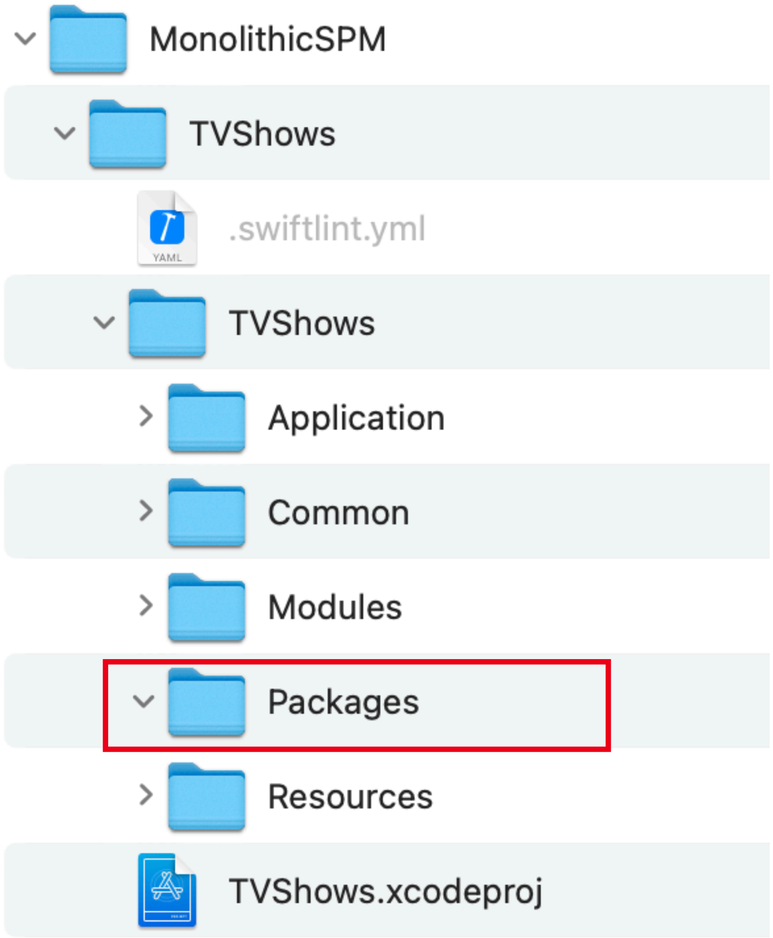


Step 3. Add new test targets

03 Modularization with SPM

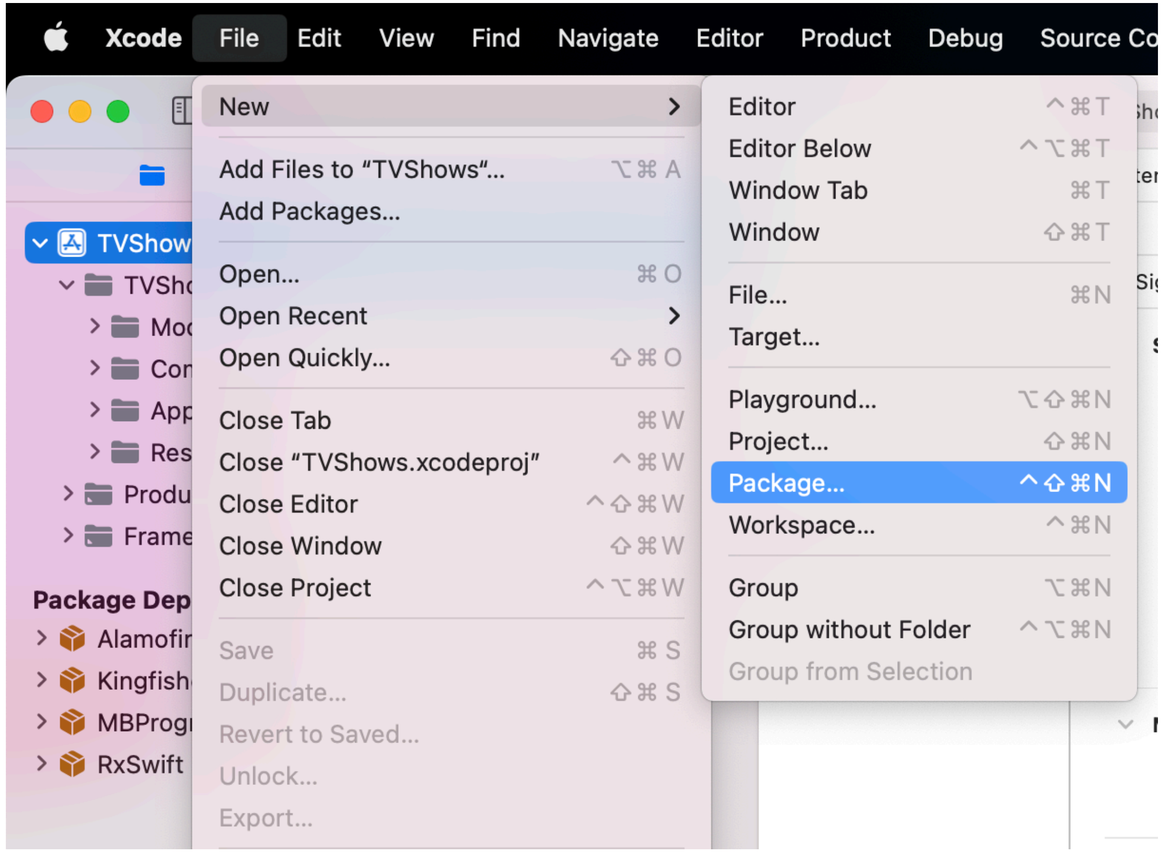
Modularization with SPM

- STEP 1 - Create Packages directory



Modularization with SPM

- STEP 2 - Create new package

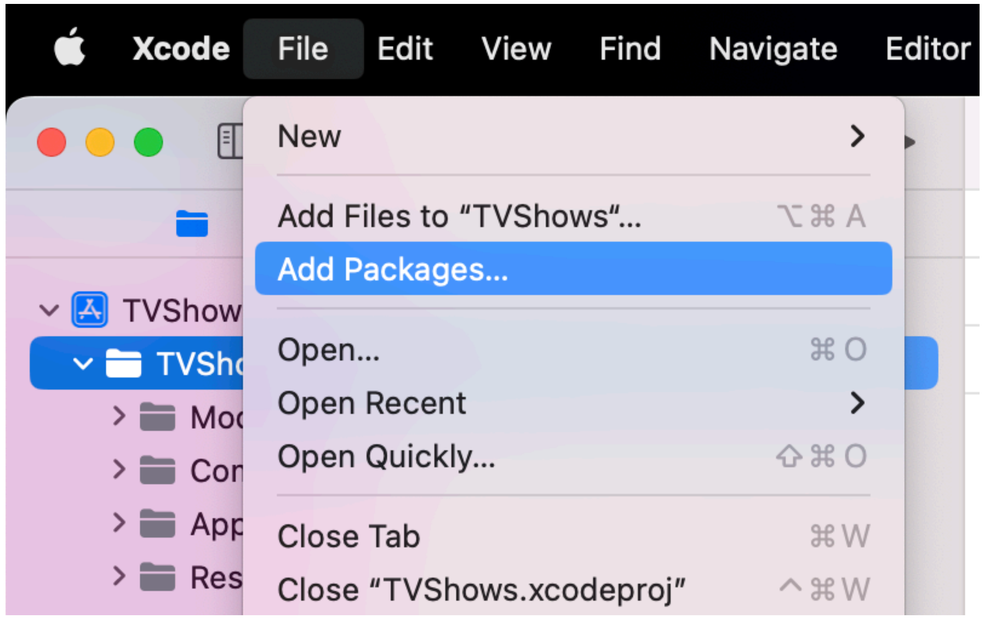


Package.swift for TVShowsCore

```
// swift-tools-version:5.7.1
// The swift-tools-version declares the minimum version of Swift required to build this package.

import PackageDescription

let package = Package(
    name: "TVShowsCore",
    platforms: [.iOS(.v16)],
    products: [
        // Products define the executables and libraries a package produces, and make them visible to other
packages.
        .library(
            name: "TVShowsCore",
            targets: ["TVShowsCore"]),
    ],
    dependencies: [
        .package(url: "https://github.com/ReactiveX/RxSwift.git", .upToNextMajor(from: "5.0.0")),
        .package(url: "https://github.com/Alamofire/Alamofire.git", .upToNextMajor(from: "5.6.4")),
        .package(url: "https://github.com/jdg/MBProgressHUD.git", .upToNextMajor(from: "1.2.0")),
        .package(url: "https://github.com/onevcat/Kingfisher.git", .upToNextMajor(from: "7.0.0")),
    ],
    targets: [
        // Targets are the basic building blocks of a package. A target can define a module or a test suite.
        // Targets can depend on other targets in this package, and on products in packages this package depends
on.
        .target(
            name: "TVShowsCore",
            dependencies: [
                .product(name: "RxSwift", package: "RxSwift"),
                .product(name: "RxCocoa", package: "RxSwift"),
                .product(name: "Alamofire", package: "Alamofire"),
                .product(name: "MBProgressHUD", package: "MBProgressHUD"),
                .product(name: "Kingfisher", package: "Kingfisher"),
            ]),
        .testTarget(
            name: "TVShowsCoreTests",
            dependencies: ["TVShowsCore"]),
    ]
)
```

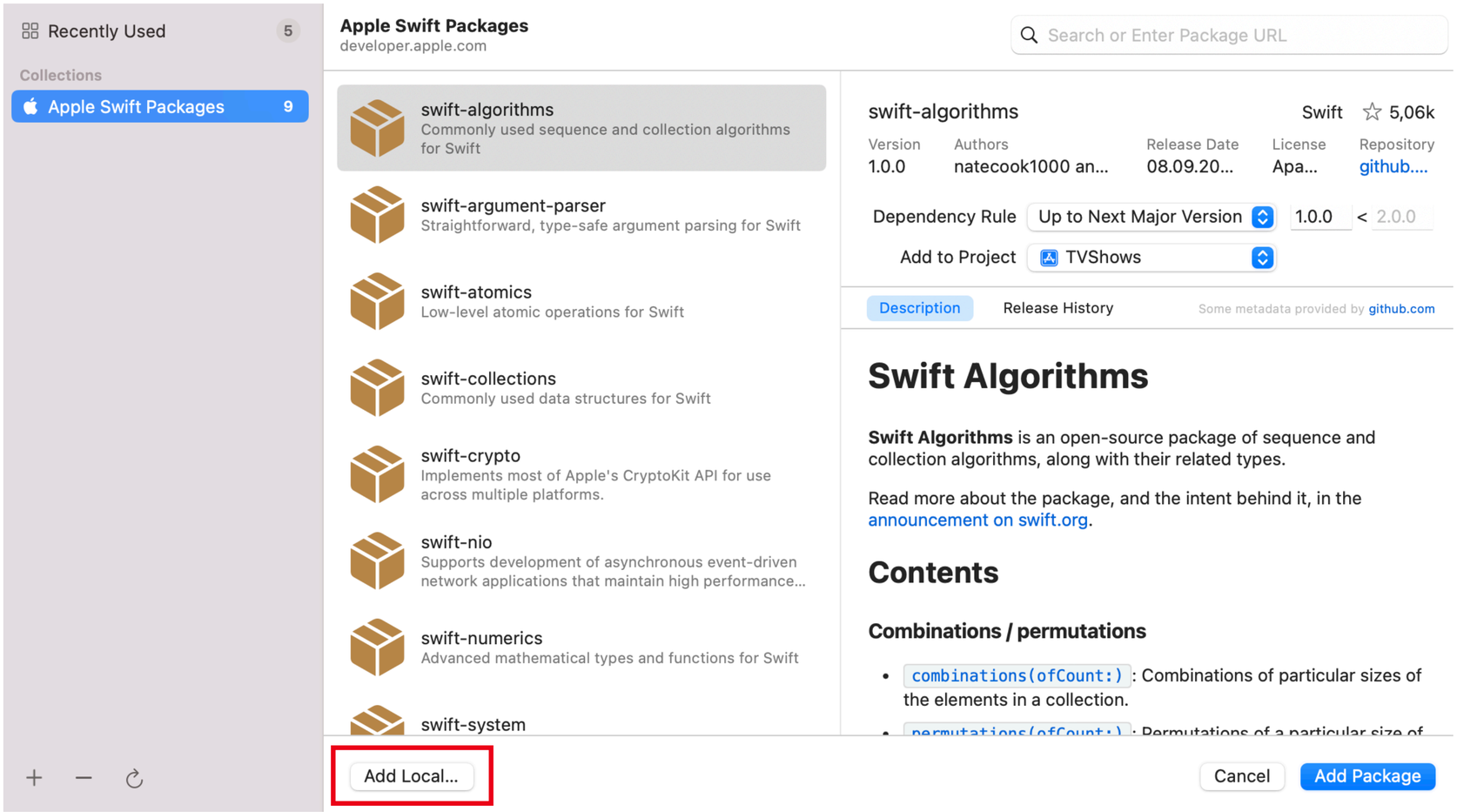
Modularization with SPM

- STEP 3 - Add the package to the project



Modularization with SPM

- STEP 3 - Add the package to the project

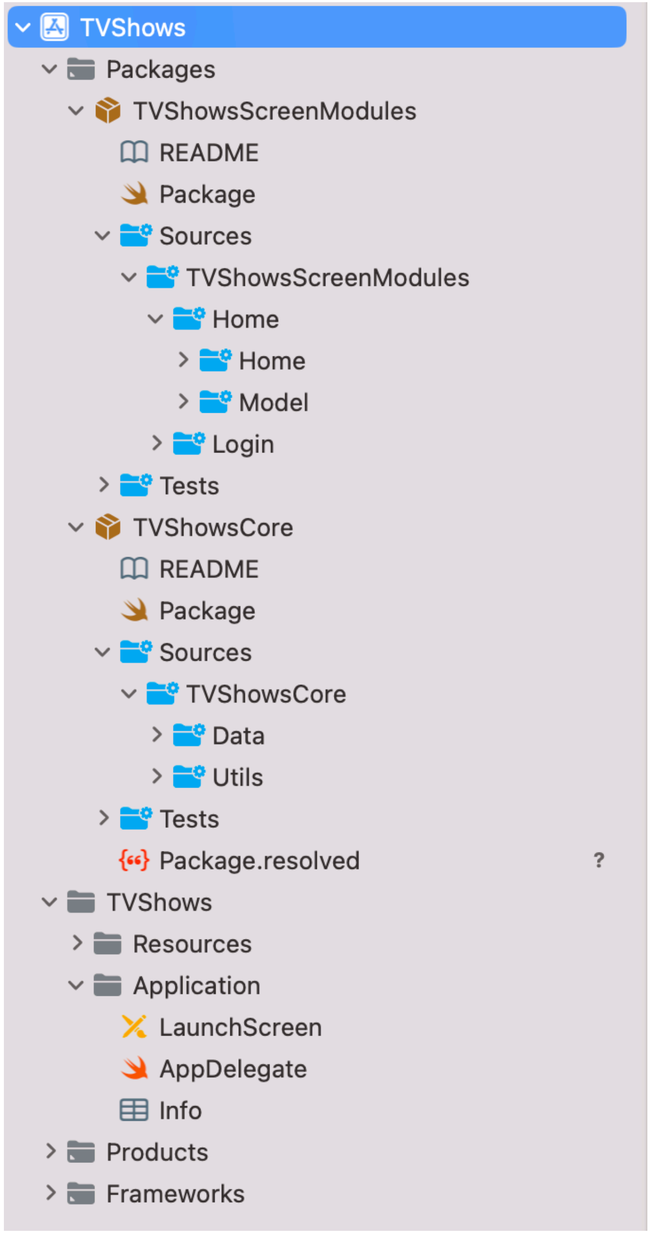


Package.swift for TVShowsScreenModules

```
// swift-tools-version:5.7.1
// The swift-tools-version declares the minimum version of Swift required to build this
package.

import PackageDescription

let package = Package(
    name: "TVShowsScreenModules",
    platforms: [.iOS(.v16)],
    products: [
        // Products define the executables and libraries a package produces, and make them
visible to other packages.
        .library(
            name: "TVShowsScreenModules",
            targets: ["TVShowsScreenModules"]),
    ],
    dependencies: [
        .package(path: "../Packages/TVShowsCore"),
    ],
    targets: [
        // Targets are the basic building blocks of a package. A target can define a module or
a test suite.
        // Targets can depend on other targets in this package, and on products in packages
this package depends on.
        .target(
            name: "TVShowsScreenModules",
            dependencies: ["TVShowsCore"]),
        .testTarget(
            name: "TVShowsScreenModulesTests",
            dependencies: ["TVShowsScreenModules"]),
    ]
)
```

Modularization with SPM

- STEP 4 - Drag and drop all project files to the desired package until you get the wanted structure.



04 Aftermath

Lets talk numbers!

ESTIMATION	REALITY
296 Hours	292 Hours

Was it worth it?

- Much cleaner and more maintainable project architecture.
- Less merge conflicts.
- Easier to add new features.
- NO CIRCULAR DEPENDENCIES!

“The hardest part of programming is naming!”

ANONYMOUS REDDIT USER

```
class FeatureAChoosePaymentOptionViewController: UIViewController
```
❌

```
class FeatureBChoosePaymentOptionViewController: UIViewController
```
❌

```
class ChoosePaymentOptionViewController: UIViewController
```
✅

Sample project

Any questions?

Newsletter

Website

YouTube

Dribbble

FB Community