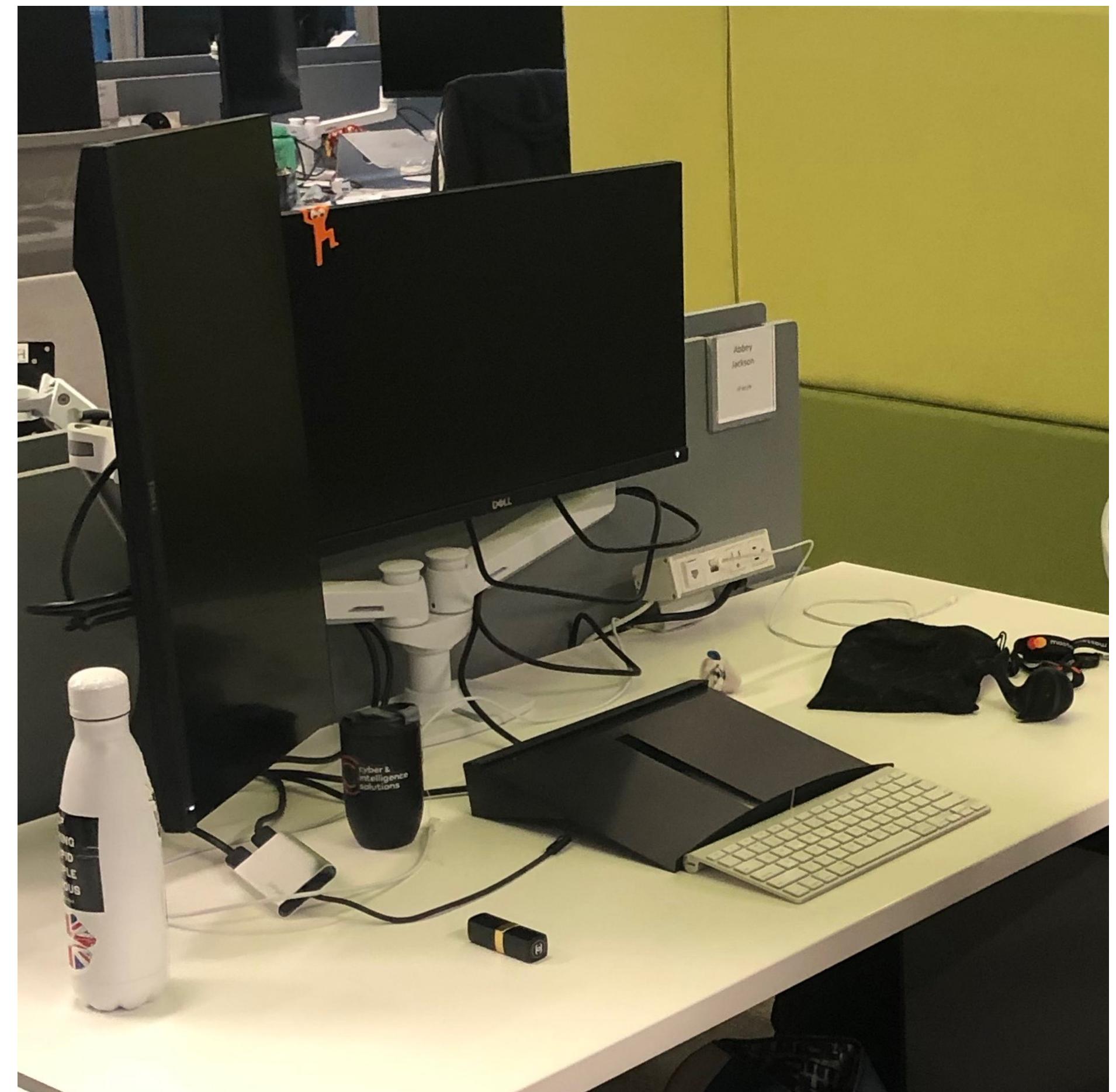
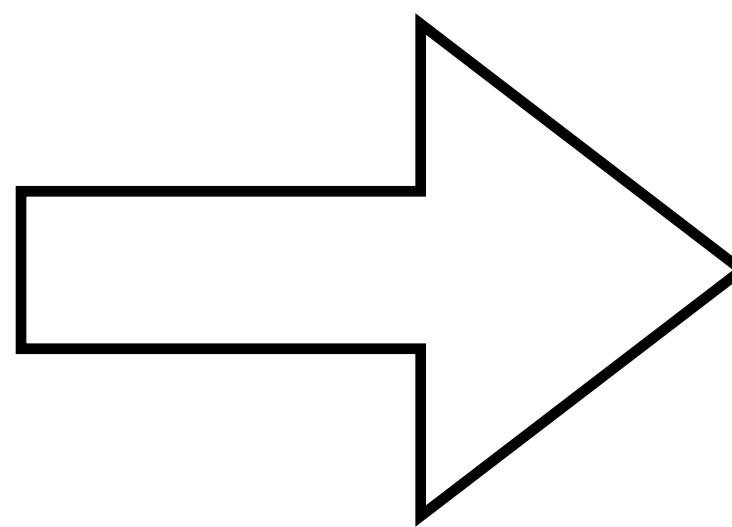




**my name is
Abbey
Jackson**



2015



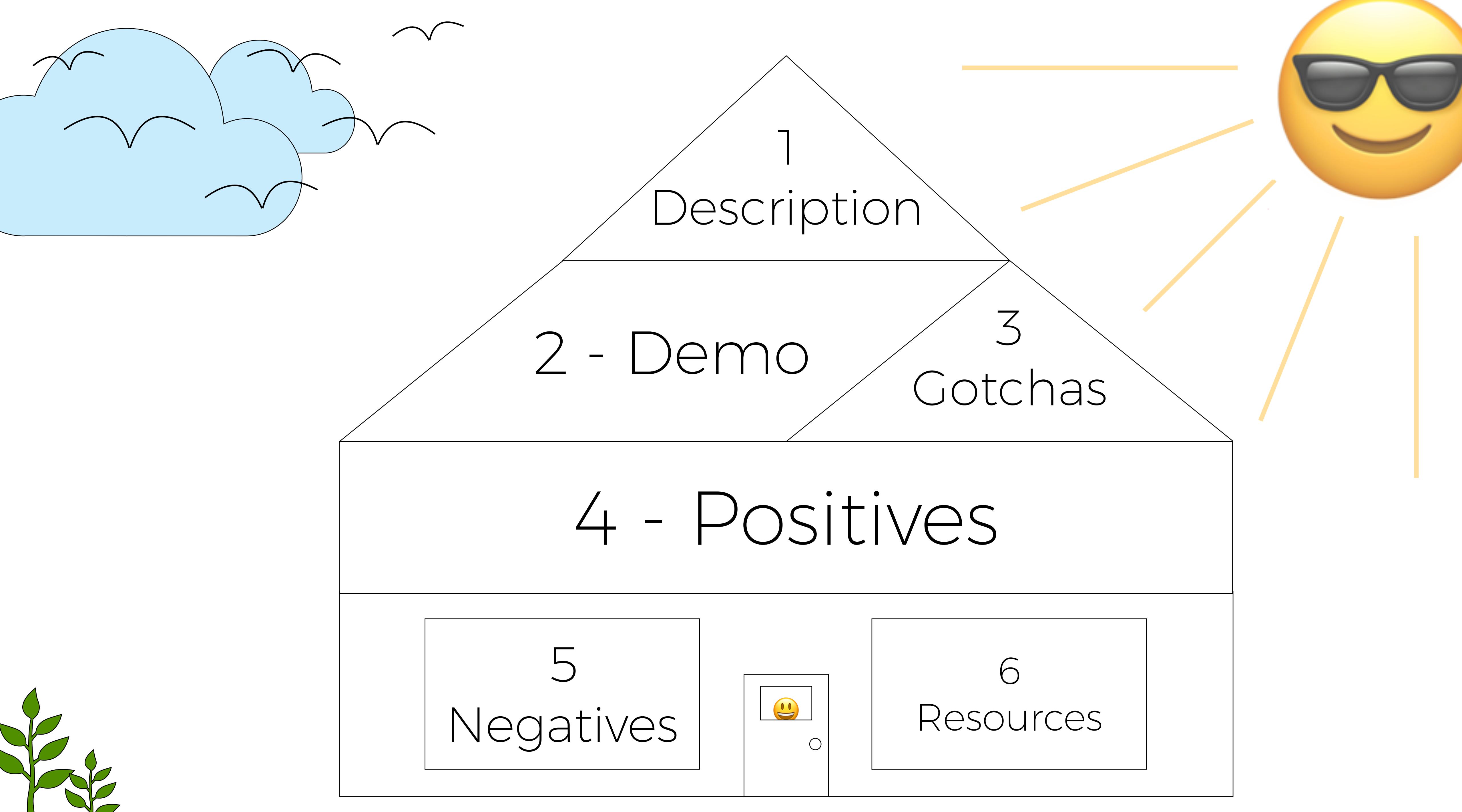
2019

MODULARIZE ALL THE THINGS!



 @earthabbey

 @abbeyjackson





**What is
Modular App
Development?**



Networking

User Data

User Interface

Authentication

Request

Response

Persistence

Profile Info

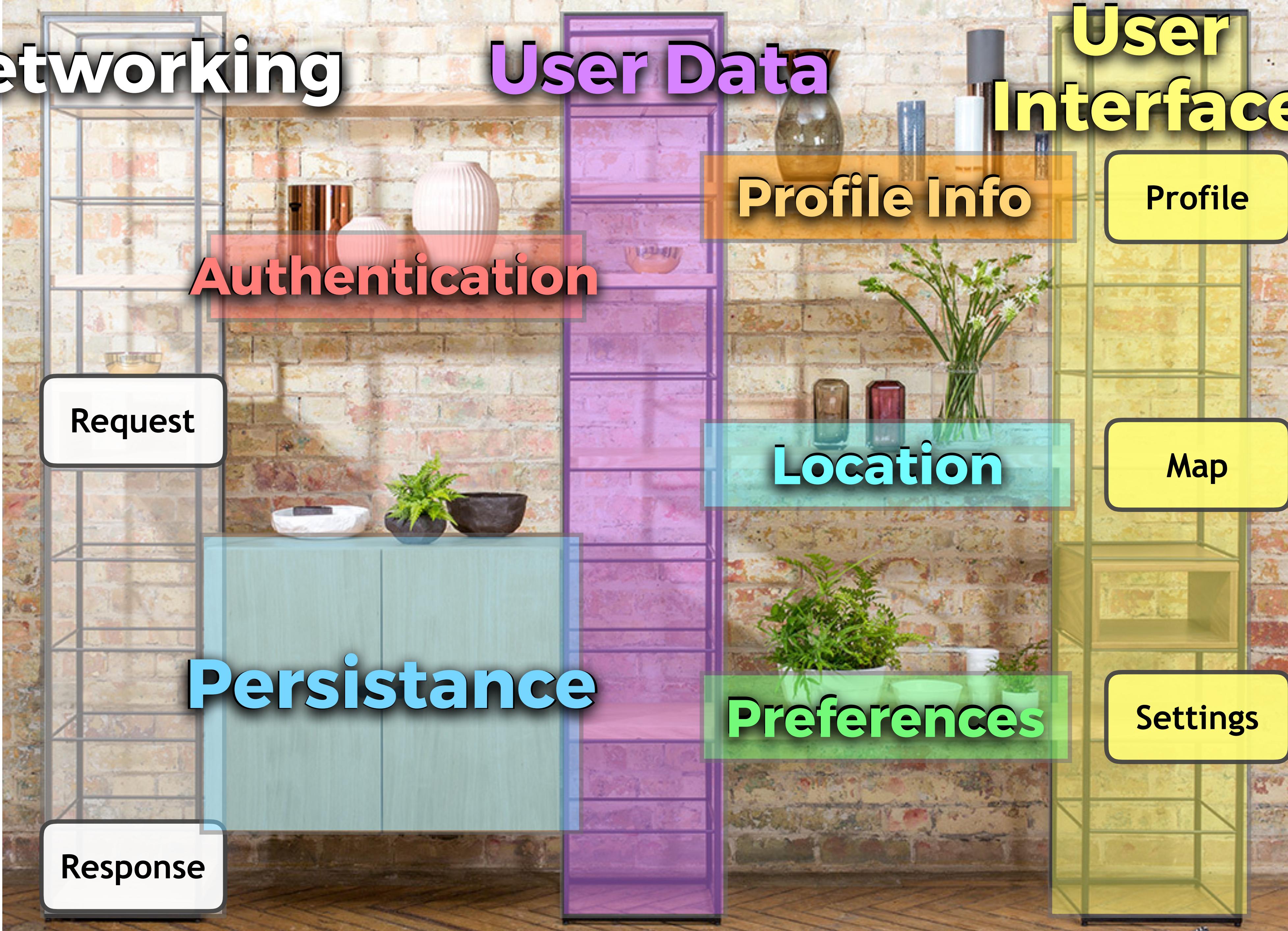
Location

Preferences

Profile

Map

Settings





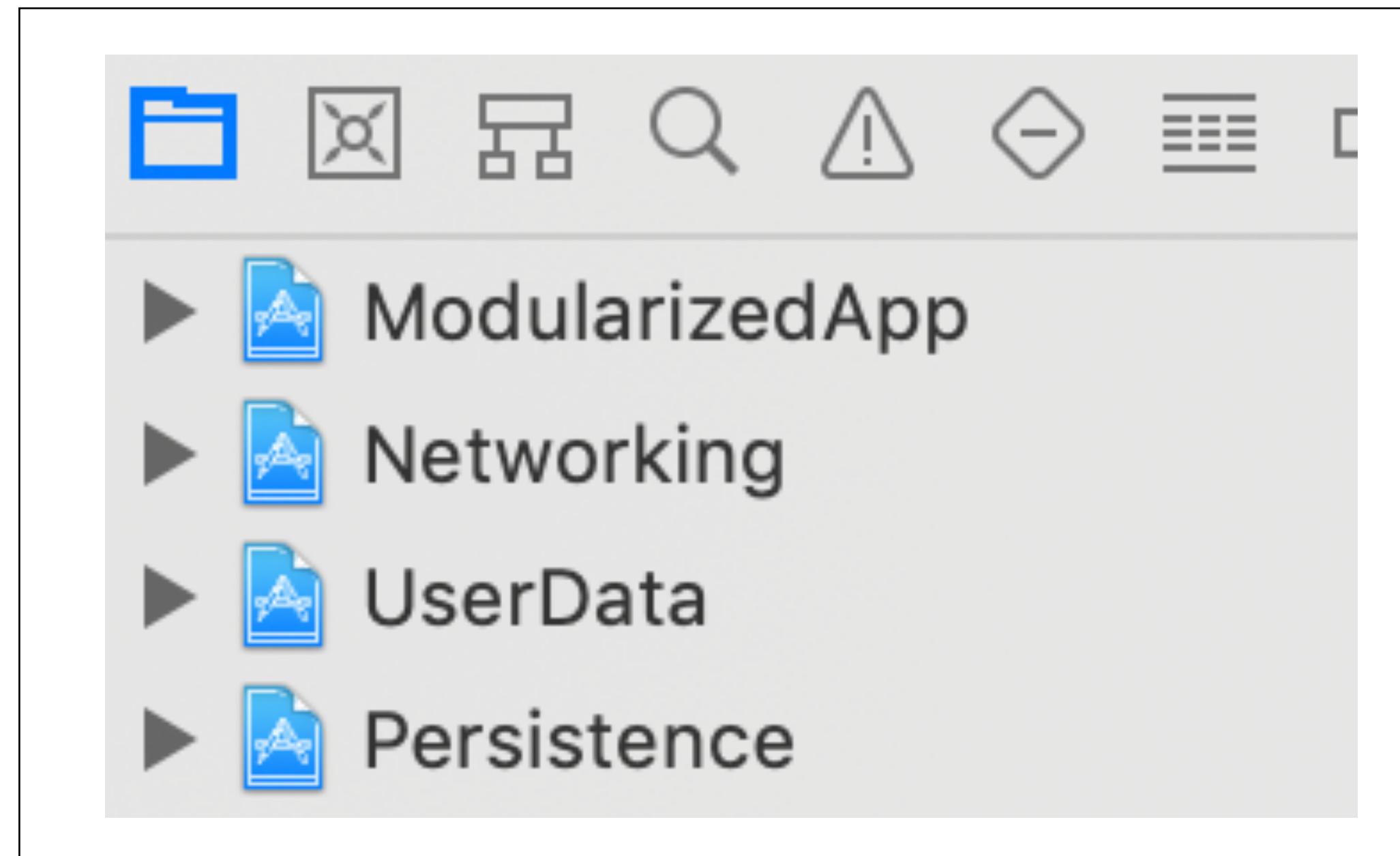
**How to
modularize
your projects**

Dynamic or Static?

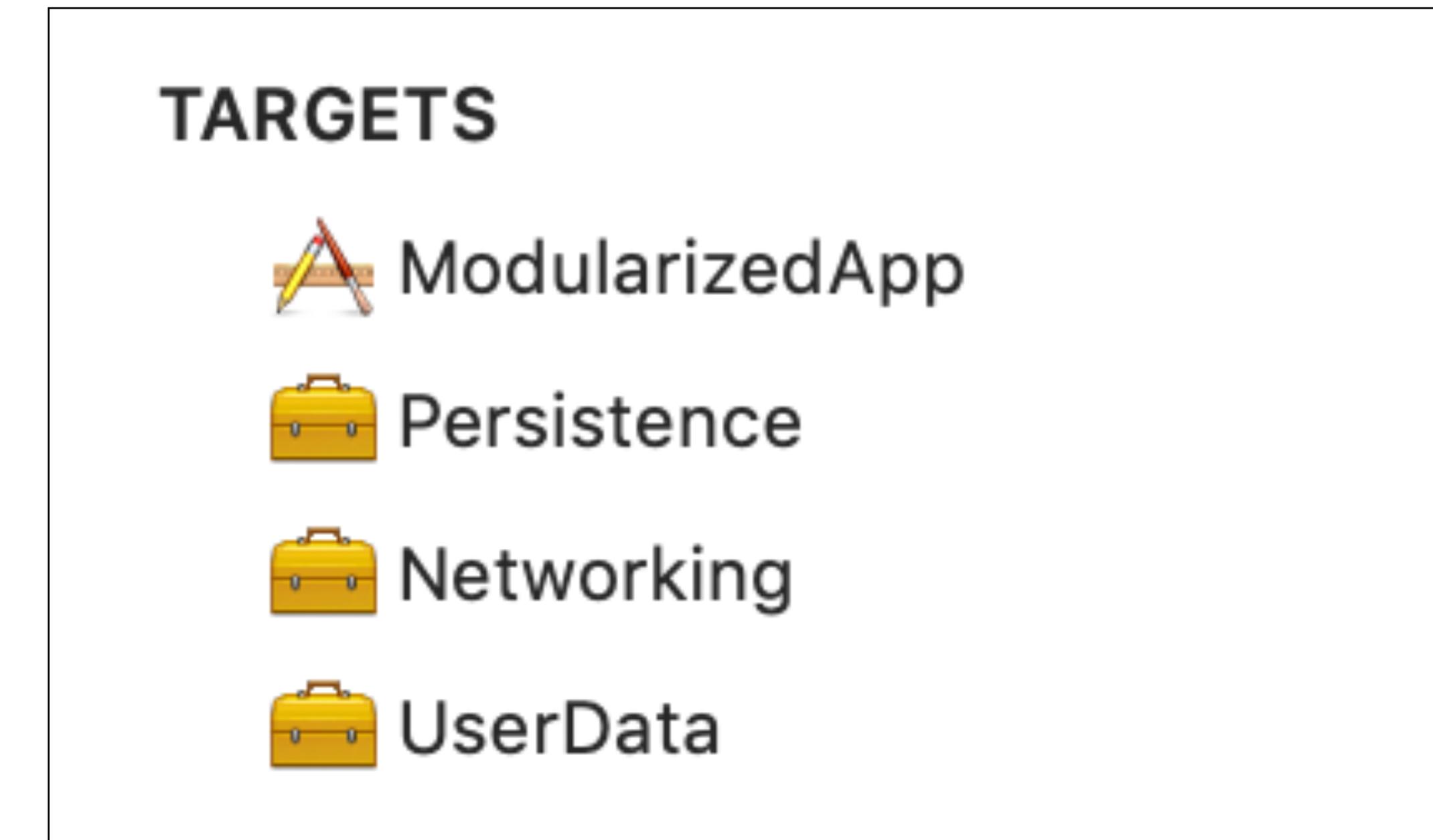
Set Up Multiple...

There are two different approaches you can take

... projects?



... targets?



... projects?

Can open separately



... targets?

Can not open separately

Files can only belong
to one project



Files can belong to
multiple targets

Visual separation in
filetree is automatic



Visual separation has to
be maintained manually

Build settings are not
related to parent app



Build settings can be
affected by inheritance

Demo - A working project

- 1 Set up your workspace
- 2 Link your frameworks
- 3 Use your frameworks

- 4 Add a new framework
- 5 Use that framework

Example: Passing Data (protocol method)

Persistence

```
public protocol DataObjectProtocol {  
    var objectId: String {get}  
    var objectData: Data? {get}  
}  
  
public struct DataObject: DataObjectProtocol { ... }  
  
public class Database {  
    public func store(_ object: DataObject) {  
        objects[object.objectId] = object  
    }  
    public func get(_ objectId: String) -> DataObjectProtocol? {  
        return objects[objectId]  
    }  
}
```

Example: Passing Data

(protocol method)

Modularized App

```
import Persistence
import UserData

extension User: DataObjectProtocol {
    public var objectId: String { ... }
    public var objectData: Data? { ... }
}

let user = User( ... )
let userDataObject = DataObject(objectId: user.objectId,
                                objectData: user.objectData)
let database = Database( ... )
database.store(userDataObject)
```

Example: Passing Data

Protocols

Delegates

Reactive Programming

Property Observers

**Where to
look to avoid
moments like this**



Not Embedding the Binary

▼ Embedded Binaries



Persistence.framework ...in build/Debug-iphoneos



~~UserData.framework~~ ...in build/Debug-iphoneos



▼ Linked Frameworks and Libraries

Name

Status



Persistence.framework

Required ▲



Networking.framework

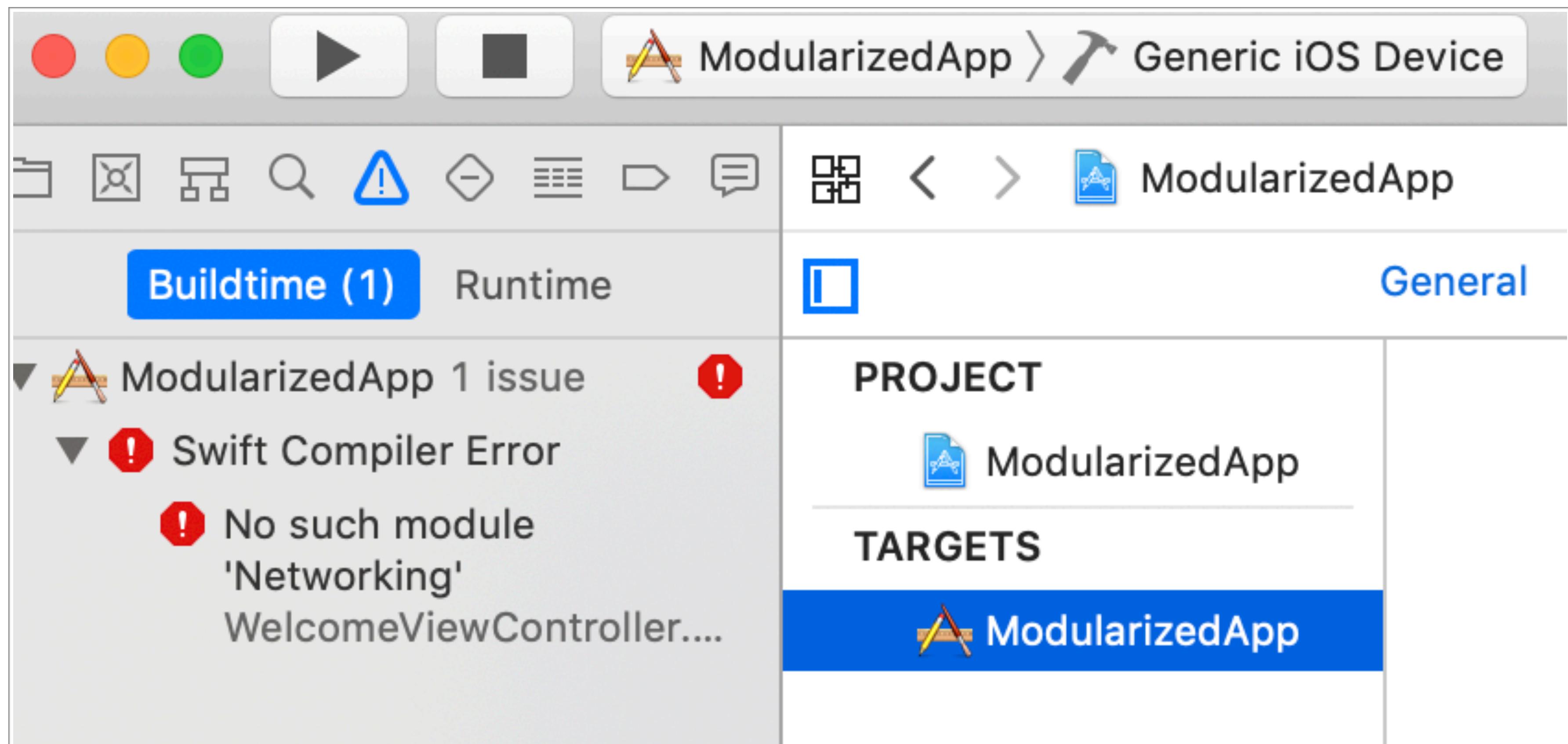
Required ▲



UserData.framework

Required ▲





Not Embedding the Binary

Not Linking in Target Dependencies

▼ Target Dependencies (2 items)



Persistence (Persistence)



UserData (UserData)

+

-

Running ModularizedApp on iPhone XR

etworking > Networking.swift > M getUserResponse()

```
private func getUserResponse() ->
UserresponseData? {  
  
let profileDictionary: [String: Any] =  
    "firstName": "Abbey" as AnyObject,  
    "lastName": "Bob" as AnyObject,  
    "email": "billy@bob.com" as AnyObject  
]  
  
let locationDictionary: [String: Any] =  
    "address": "123 Smith Street" as AnyObject,  
    "city": "Toronto" as AnyObject,
```

Billy Bob

123 Smith

Not Embedding the Binary

Not Linking in Target Dependencies

Using Common Module Name

Basic

Customized

All

Combined

Levels

+

Q name

▼ Packaging

Setting



Networking

▶ Product Module Name

Networking

Product Name

Networkinga

Strings File Output Encoding

\$(PRODUCT_NAME:c99extidentifier)

▼ Search Paths

Setting

Basic

Customized

All

Combined

Levels

+

Q name

▼ Packaging

Setting



Networking

► Product Module Name

Networking

Product Name

Networkinga

Strings File Output Encoding

\$(APP_PREFIX)\$(PRODUCT_NAME:c99extid

▼ Search Paths

Not Embedding the Binary

Not Linking in Target Dependencies

Using Common Module Name

Test Target Greyed Out

The screenshot shows a modularized application interface with the title "ModularizedApp". The top bar includes status indicators (red, yellow, green circles), a play button, a square button, a pencil icon, and a "ModularizedApp" title with a dropdown arrow. Below the top bar is a toolbar with icons for folder, search, and other functions. The main area displays two test suites: "ModularizedAppTests" (1 test) and "PersistenceTests" (1 test). The "ModularizedAppTests" suite has a green checkmark icon. To the right of the test suites are numerical values (1 and 2) and green double-slash symbols (//).

Test Suite	Test Count	Status
ModularizedAppTests	1	Success (green checkmark)
PersistenceTests	1	Success (green checkmark)

The screenshot shows a persistence test interface with the title "Persistence". The top bar includes status indicators (red, yellow, green circles), a play button, a square button, a briefcase icon, and a "Persistence" title with a dropdown arrow. Below the top bar is a toolbar with icons for folder, search, and other functions. The main area displays two test suites: "ModularizedAppTests" (1 test) and "PersistenceTests" (1 test). Both suites have green checkmark icons. To the right of the test suites are numerical values (1 and 2) and green double-slash symbols (//).

Test Suite	Test Count	Status
ModularizedAppTests	1	Success (green checkmark)
PersistenceTests	1	Success (green checkmark)



Persistence > iPhone XR

► **Build**
1 target

► **Run**
Debug

► **Test**
Debug

► **Profile**
Release

► **Analyze**
Debug

► **Archive**
Release

Info

Build Config

Def

Debug Proc

Tests



 Persistence >  iPhone XR

►  **Build**
1 target

►  **Run**
Debug

►  **Test**
Debug

►  **Profile**
Release

►  **Analyze**
Debug

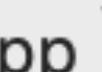
►  **Archive**
Release

Info

Build Configu

Det

D

 ModularizedApp >  iPhone XR

►  **Build**
2 targets

►  **Run**
Debug

►  **Test**
Debug

►  **Profile**
Release

►  **Analyze**
Debug

►  **Archive**
Release

Info

Build Configu

Debu

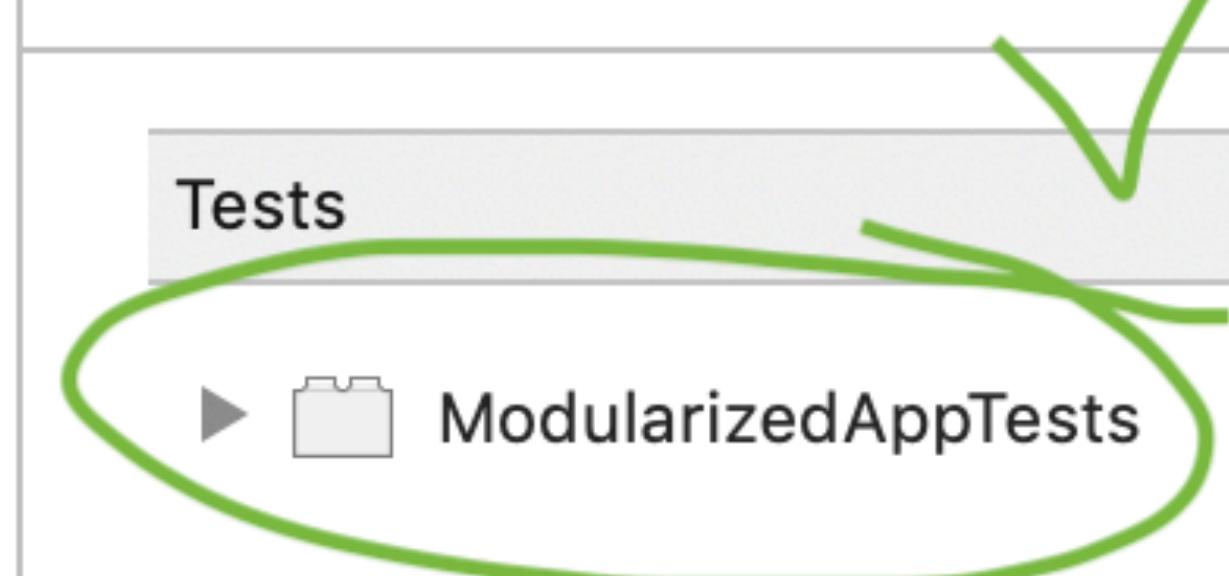
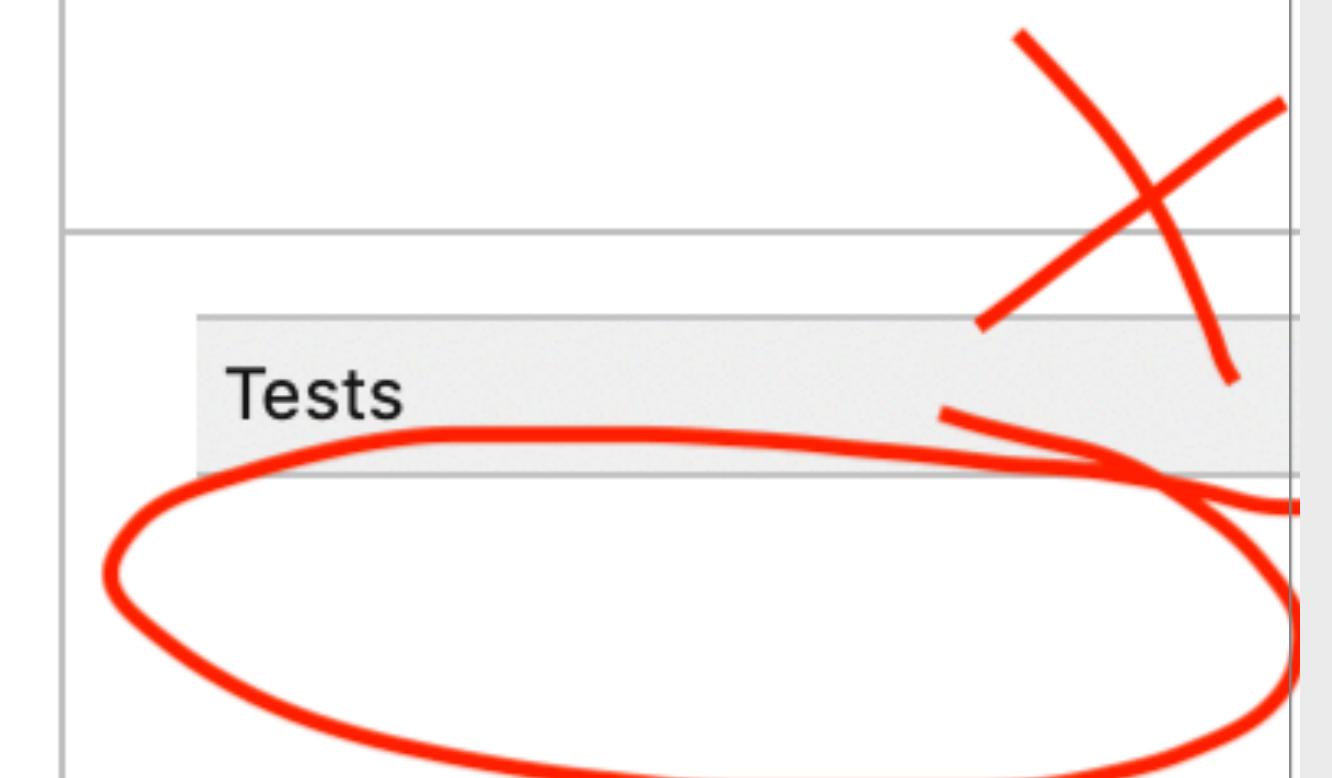
Debug Proce

Tests

Tests

Tests

►  ModularizedAppTests



Not Embedding the Binary

Not Linking in Target Dependencies

Using Common Module Name

Test Target Greyed Out

Violating Proper Unit Testing Principles

The screenshot shows the Xcode interface with the PersistenceTests target selected in the left sidebar. The Build Phases tab is active, displaying the Target Dependencies, Compile Sources, and Link Binary With Libraries sections.

Target Dependencies: Persistence (Persistence)

Compile Sources: PersistenceTests.swift

Link Binary With Libraries:

Name	Status
Networking.framework	Req... ▾
Persistence.framework	Req... ▾

PersistenceTests.swift Content:

```
// PersistenceTests.swift
// PersistenceTests
//
// Created by Abbey Jackson on 2019-08-12.
// Copyright © 2019 Abbey Jackson. All rights reserved.

import XCTest
@testable import Persistence
@testable import Networking

class PersistenceTests: XCTestCase {

    func testTheIntrnalsOfNetworking() {
        let userResponse1 = Networking().getUserResponse()
        let userResponse2 = Networking().getUserResponse()
        XCTAssertEqual(userResponse1!.dataDictionary.count,
                      userResponse2!.dataDictionary.count)
    }
}
```

Not Embedding the Binary

Not Linking in Target Dependencies

Using Common Module Name

Test Target Greyed Out

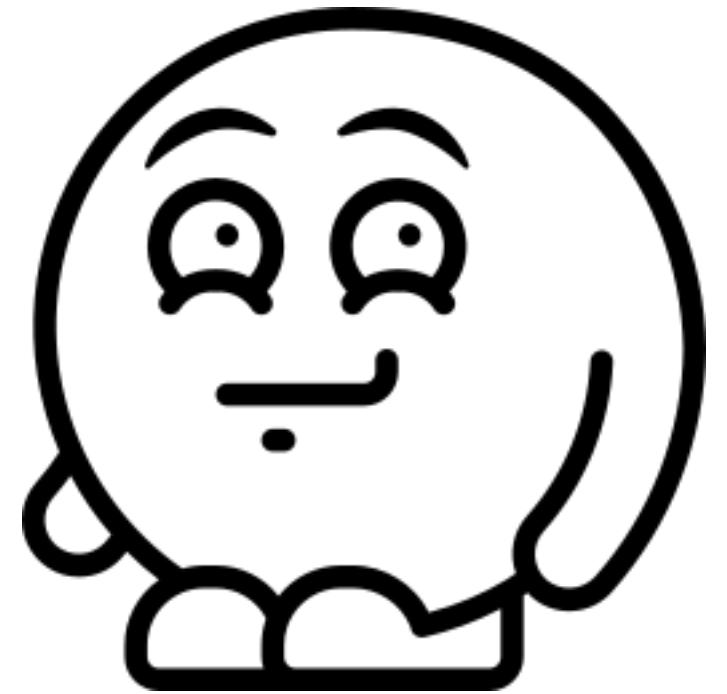
Violating Proper Unit Testing Principles

Inconsistency Between Targets

iOS Deployment Target

Swift Version (if < 5.0)

Active Architectures



**Why might
you want to
modularize?**

Wide-Spread Benefits

Modularizing your app can affect more than just your developers

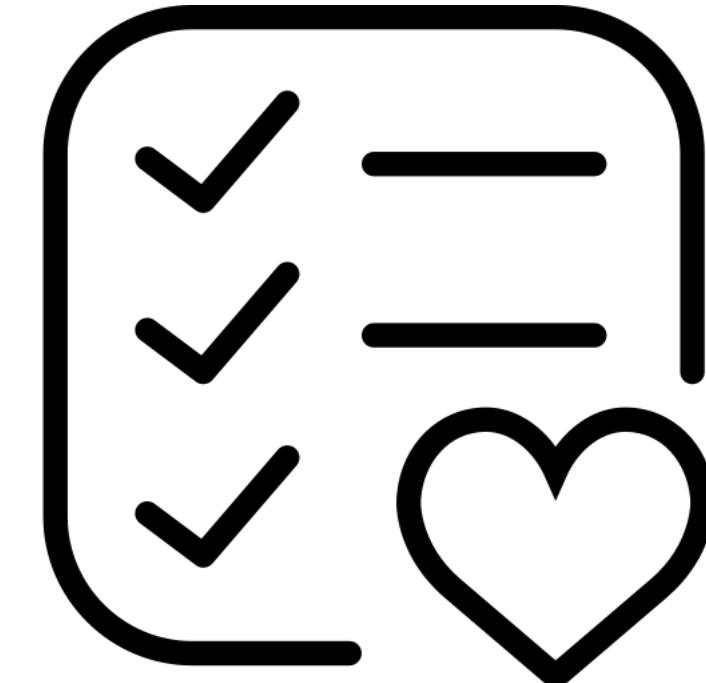
Business



Product



Development



Lower Tooling
Costs

Faster
Releases

Improved
Communication



Less Developer
Resourcing

Happier
Clients

Wide-Spread Benefits

Modularizing your app can affect more than just your developers

Business



Product



Development



Less Bugs

Better
Reviews

Improved
Communication

Faster
Fixes

Fewer
Regressions

Less Support
Resourcing

Happier
Clients



Wide-Spread Benefits

Modularizing your app can affect more than just your developers

Business



Product



Development



Code
Reuse

Uncoupled
Dependencies

Pipeline
Improvements

Better
Metrics

Increased
Code Integrity

Improved
Testing

Faster
Buildtimes

Easier
Debugging

Enforce
Best Practices

Better
Sense of
Ownership



ALL THE THINGS!



Wide-Spread Benefits

Modularizing your app can affect more than just your developers

Business



Product



Development





**When is
modularization
not so great?**

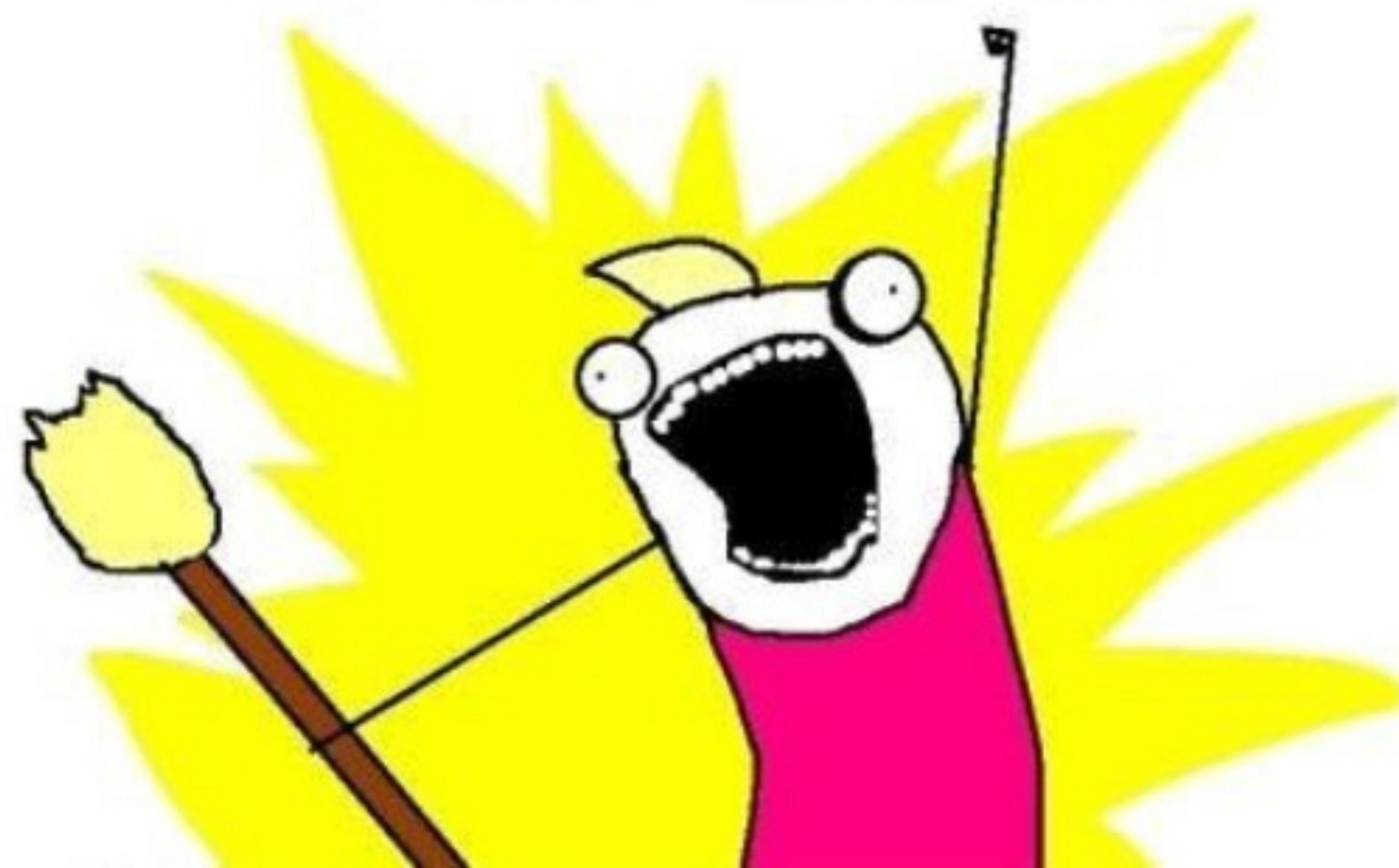
Handling Assets

Where to put stuff

Separate Project Settings

Sharing UI

MODULARIZE ALL THE THINGS!



Sample Code

[Modularized Workspace](http://abgl.ca/mod-basic) (abgl.ca/mod-basic)

Please read the README.md that comes with the sample code as it explains some additional things.

The sample code contains a workspace with an app linked to modules. It also demonstrates passing data and additionally demonstrates the testing issue I mentioned to watch out for.

Recordings

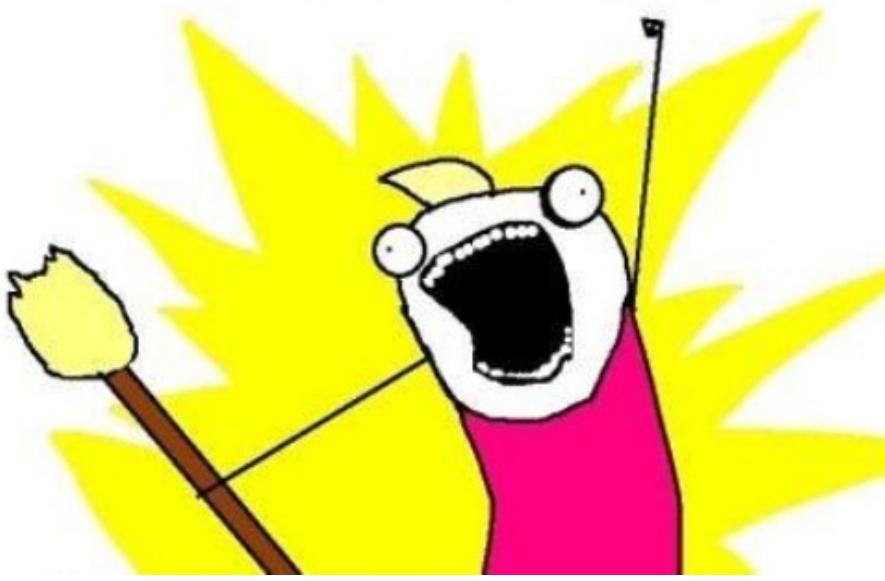
[All Available On My Website](http://abgl.ca/talks) (abgl.ca/talks)

 @earthabbey

 @abbeyjackson



MODULARIZE ALL THE THINGS!



 @earthabbey

 @abbeyjackson

Further Reading

It's always a good idea to learn from those who have come before you, even if everything is working out great and you feel confident about your set up!

- [Modularising iOS apps into powerful reusable kits](#), from *Zomato* (abgl.ca/mod-1)
- [Modular iOS Part 1: Strangling the Monolith](#), from *Sam Dods* (abgl.ca/mod-2)
 - [Modular iOS Part 2: Splitting a Workspace Into Modules](#), from *Sam Dods* (abgl.ca/mod-3)
 - [Modular iOS Part 3: Configuration & Testing of Modules](#), from *Sam Dods* (abgl.ca/mod-4)
 - [Modular iOS Part 4: Sharing Configuration Between Modules](#), from *Sam Dods* (abgl.ca/mod-5)
- [Our iOS modularization story](#), from *wehkamp* (abgl.ca/mod-6)
- [Using CocoaPods to Modularize a Big iOS App](#), from *Hubspot* (abgl.ca/mod-7)
 - [Update: How We're Building iOS Apps Today](#), from *Hubspot* (abgl.ca/mod-8)
- [iOS at Scale: Modularization of the Wayfair App](#), from *Wayfair* (abgl.ca/mod-9)
- [Building a Composable, Flexible, and Functional Layout in our iOS App](#), from *Wayfair* (abgl.ca/mod-10)
- [How to create bridges between frameworks in an iOS app](#), from *Rayane Kurrimboccus* (abgl.ca/mod-11)
- And [this SO post](#) which shows the kinds of confusing troubles you can get into (abgl.ca/mod-12)