Ускорение разработки при помощи Unit-Теста

Дмитрий Куркин
Spark-Networks

План

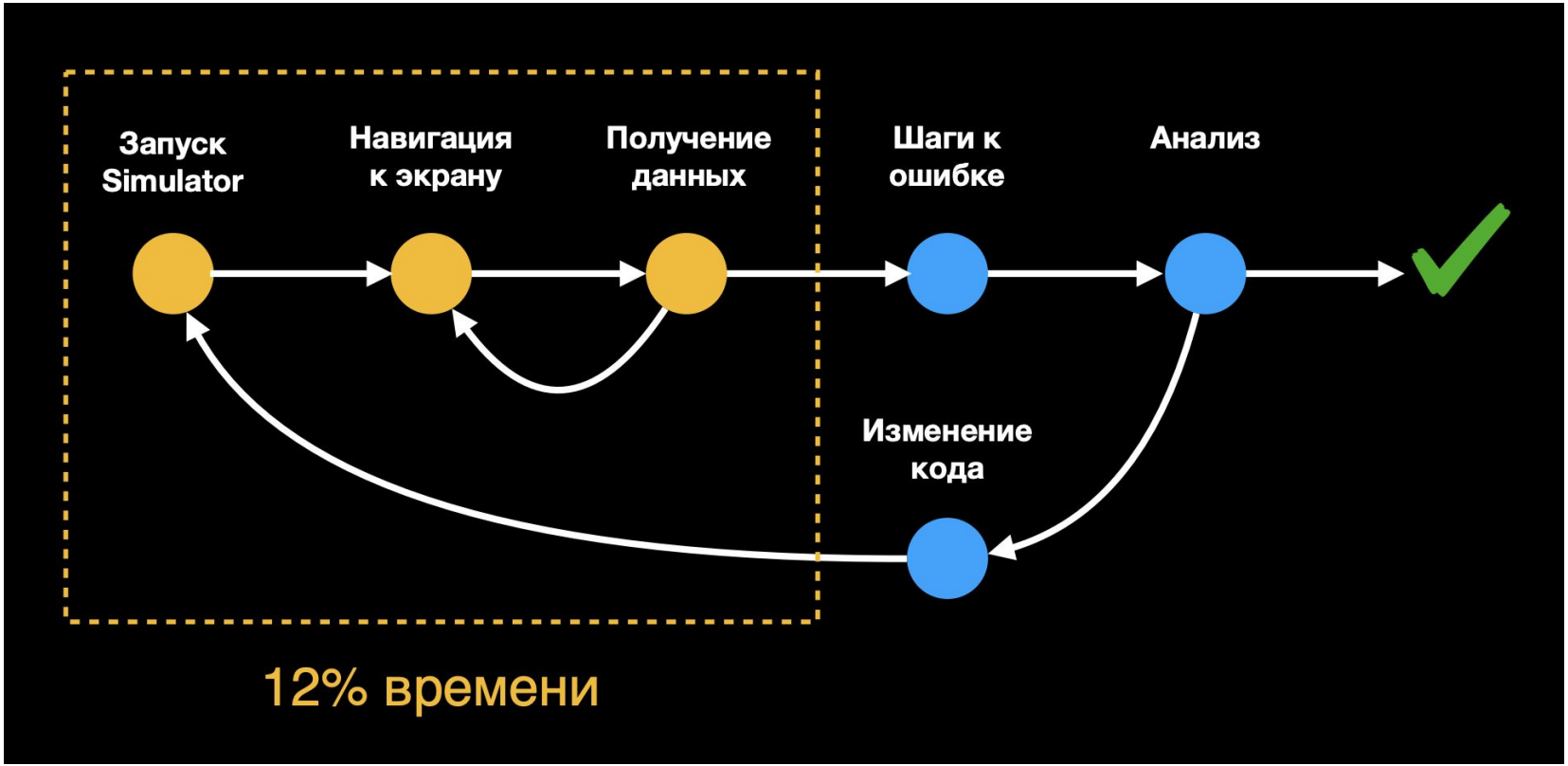
- Где мы зря теряем время
- Известные методы оптимизации доступа к модулю
- Интерактивный тест

Мы теряем много времени чтобы проверить свой код

Готовим приложение в Playgrounds

https://mobius-piter.ru/2018/spb/talks/2ie8oi8sreycuecsaq6qeg/

Готовим приложение в Playgrounds

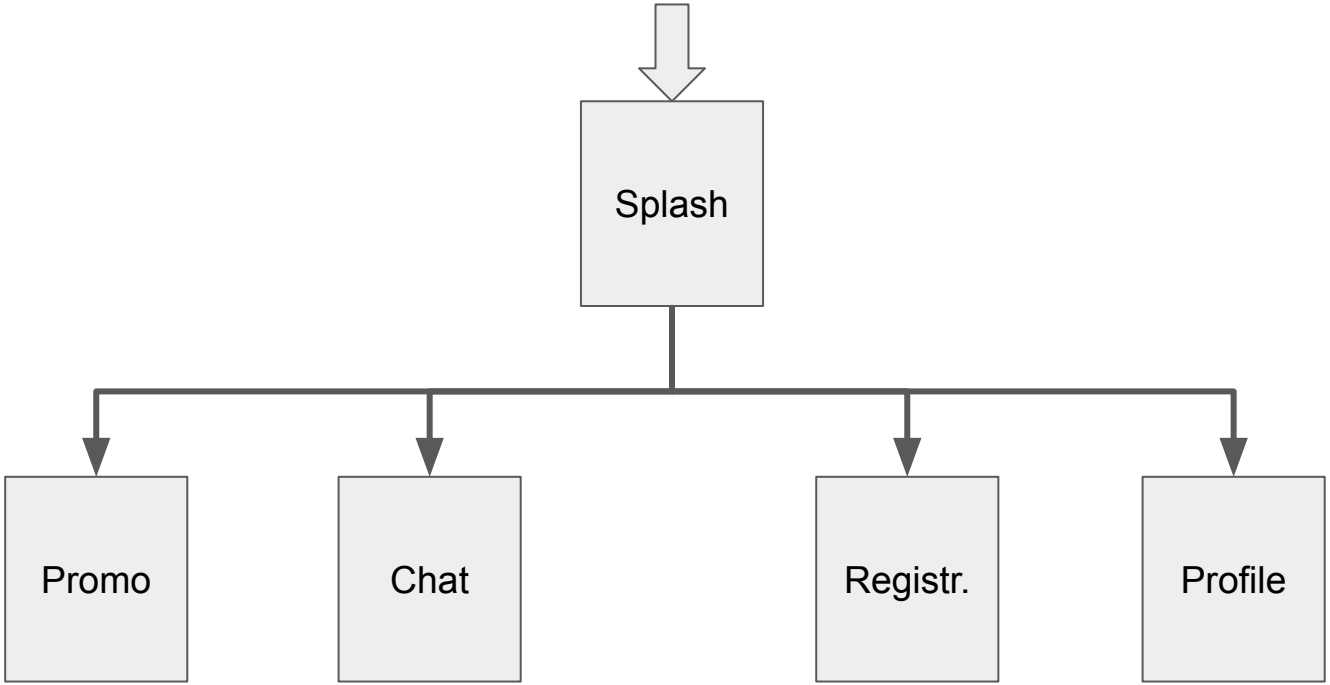


Сложные сценарии

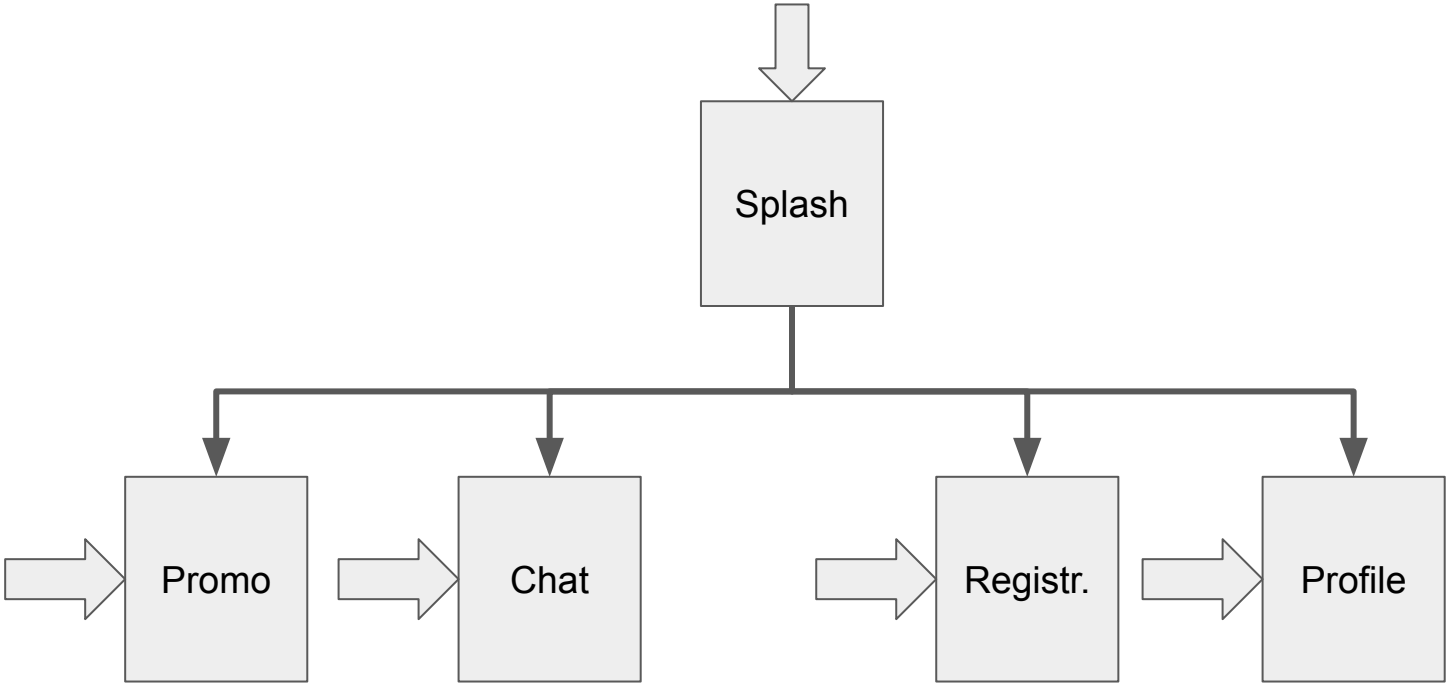
- Требует ответных действий другого пользователя
- Ошибки сервера
- Включает в себя покупку
- Комбинация многих условий

Как упростить?

Точки входа

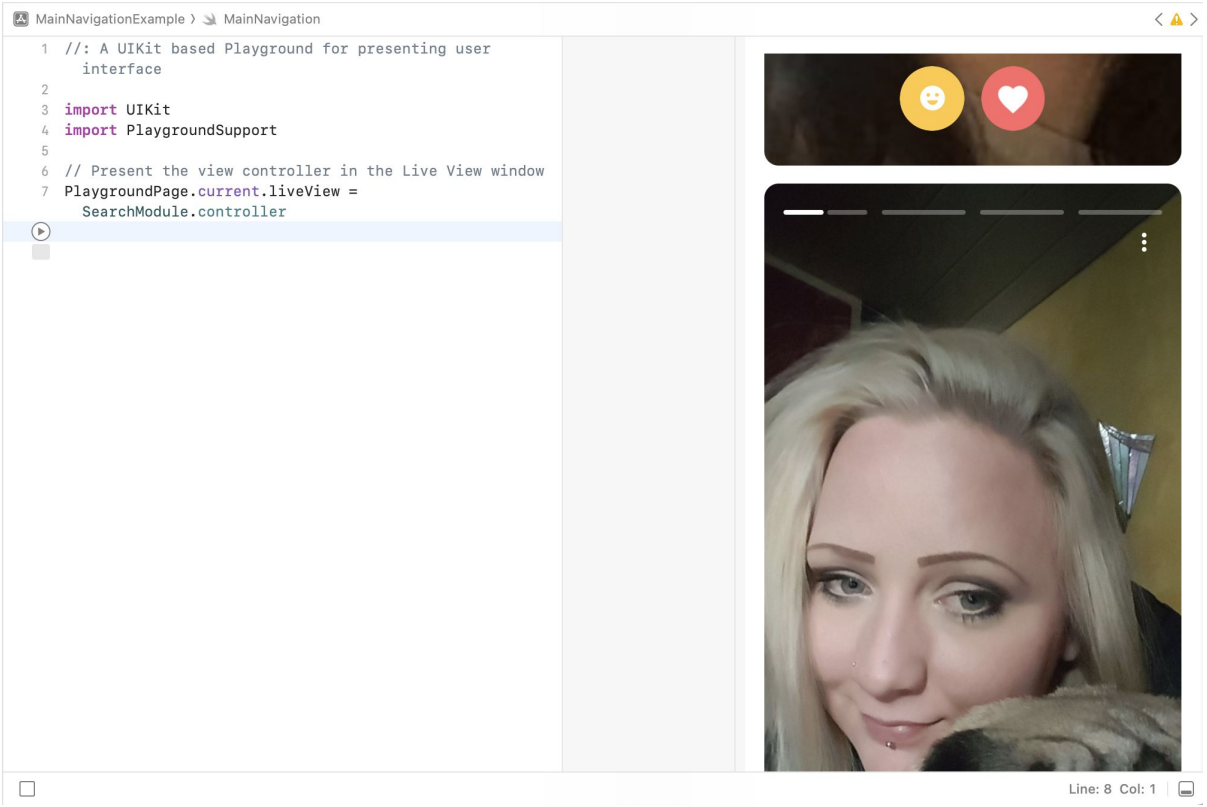


Точки входа



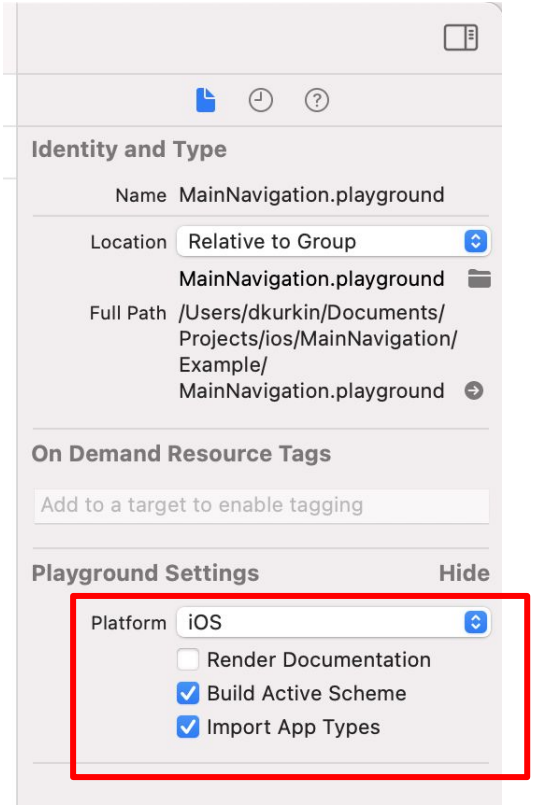
Известные подходы

Playground


MainNavigationExample
MainNavigation
//: A UIKit based Playground for presenting user interface
import UIKit
import PlaygroundSupport
// Present the view controller in the Live View window
PlaygroundPage.current.liveView = SearchModule.controller
Line: 8 Col: 1

Playground стали проще

- Build Active Scheme - собирает код проекта при запуске
- Import App Types - можно вызывать код проекта без import



Playground в workspace

Playground с Mock-ми

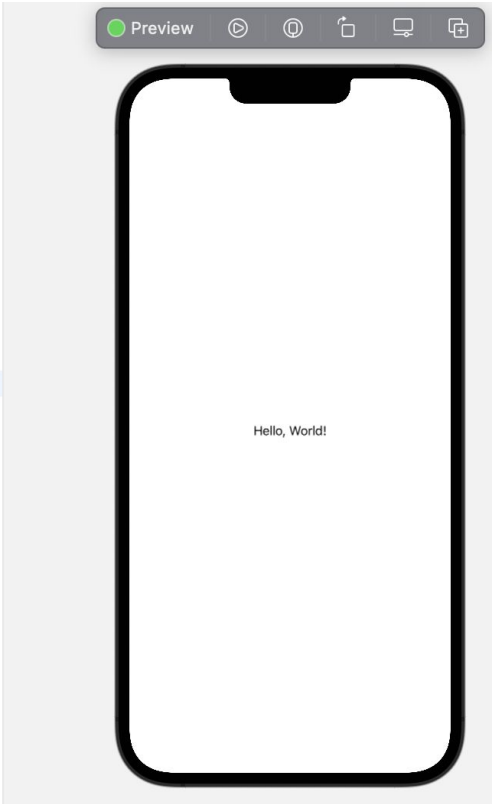
```
import SearchService
import SearchServiceMocks

let serviceMock = ServiceMock(with: .mockData)
let module = SearchModule(service: ServiceMock())

// Present the view controller in the Live View window
PlaygroundPage.current.liveView = module.controller
```

Недостатки Playground

- Нестабильная работа Playground
- Playground никак не связан с устройством
- Не работает Debugger
- Сложный доступ к тестовым данным

Swift UI Preview

```swift
import SwiftUI

struct SearchView: View {
    var body: some View {
        Text("Hello, World!")
    }
}

struct SearchView_Previews: PreviewProvider {
    static var previews: some View {
        SearchView()
    }
}
```



Swift UI Preview

- Интерактивный
- Прогрессивный
- Твой

Swift UI Preview

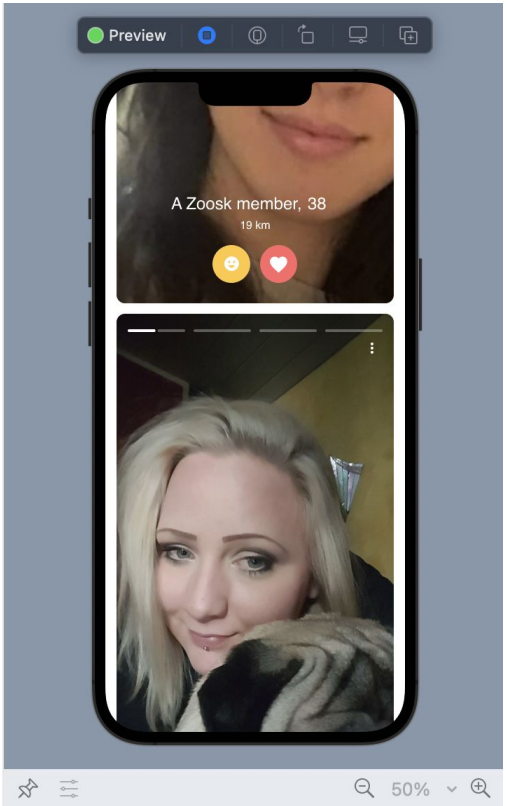
- Привязан к устройству
- Может быть запущен на физическом устройстве
- Поддерживает множество вариантов preview

Preview by default

```
struct SearchView: View {
    var body: some View {
        Text("Hello, World!")
    }
}

struct SearchView_Previews: PreviewProvider {
    static var previews: some View {
        SearchView()
    }
}
```

Preview в жизни

```
struct SearchListView_Previews: PreviewProvider {
    static var previews: some View {
        Group {
            SearchListView<SearchListStateMock>(interactor:
              interactorWithProfiles)
            SearchListView<SearchListStateMock>(interactor:
              interactorWithoutProfiles)
        }
    }

    static var interactorWithProfiles:
      SearchListInteractor<SearchListStateMock> {
        let interactor = SearchListInteractor(
            state: SearchListStateMock(),
            profilesService: SearchProfilesService(),
            appConfigurationService: ConfigurationService()
        )
        interactor.state.listIsEmpty = false
        return interactor
    }

    static var interactorWithoutProfiles:
      SearchListInteractor<SearchListStateMock> {
        let interactor = SearchListInteractor(
            state: SearchListStateMock(),
            profilesService: SearchProfilesService(),
            appConfigurationService: ConfigurationService()
        )
        interactor.state.listIsEmpty = true
        return interactor
    }
}
```



Недостатки в коде

- Дополнительная нагрузка на View
- Отсутствие разделения с production

Технические недостатки Preview

- Сложная и нестабильная технология сборки
- Плохая поддержка framework-ов
- Варианты прибавляют нагрузки на MacBook
- Stateless цикл жизни preview
- Сложность с доступом к тестовым данным

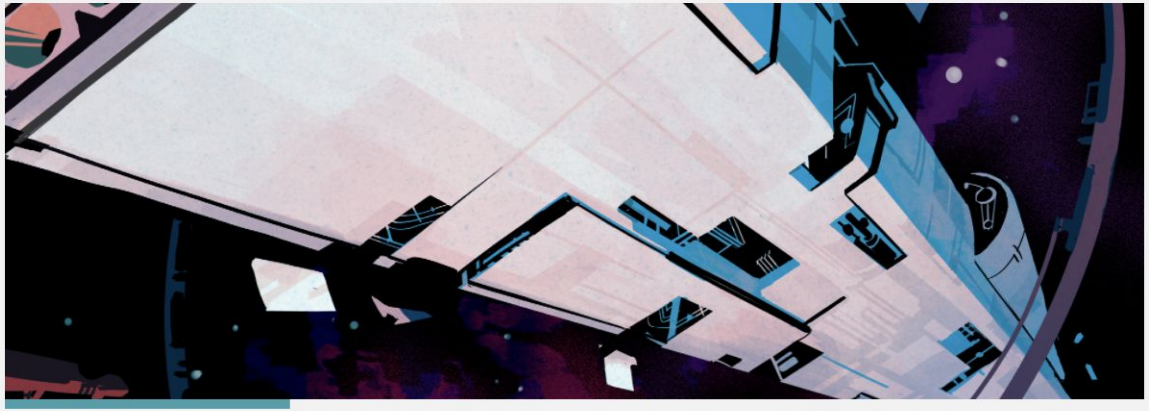
Microapps

increment

ISSUE 18

AUGUST 2021

Mobile

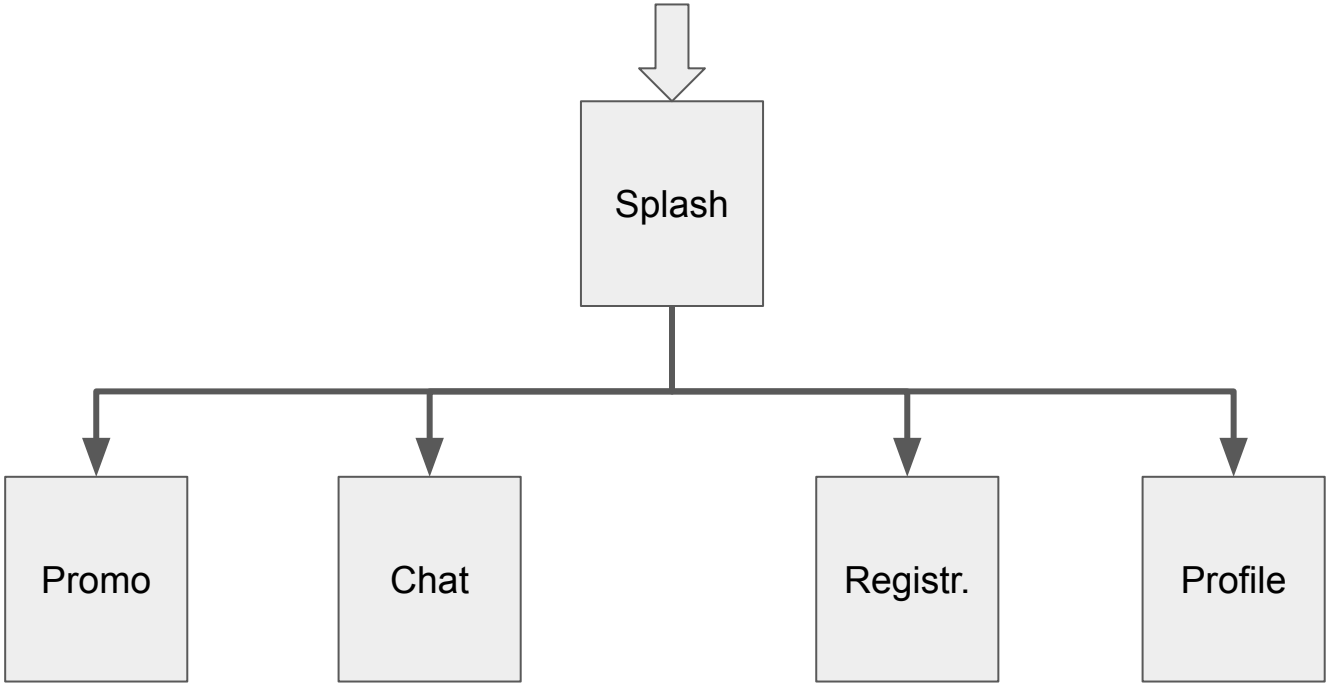


GIO LODI

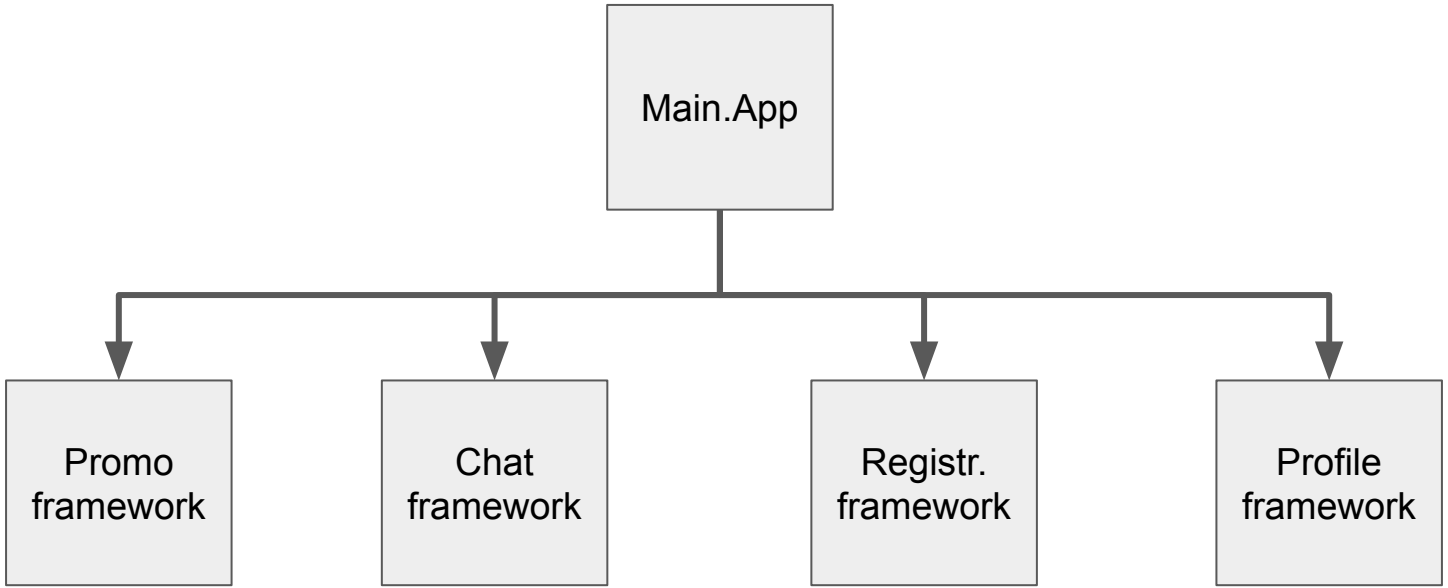
Meet the microapps architecture

How an emerging architecture pattern inspired by microservices can invigorate feature development and amplify developer velocity.

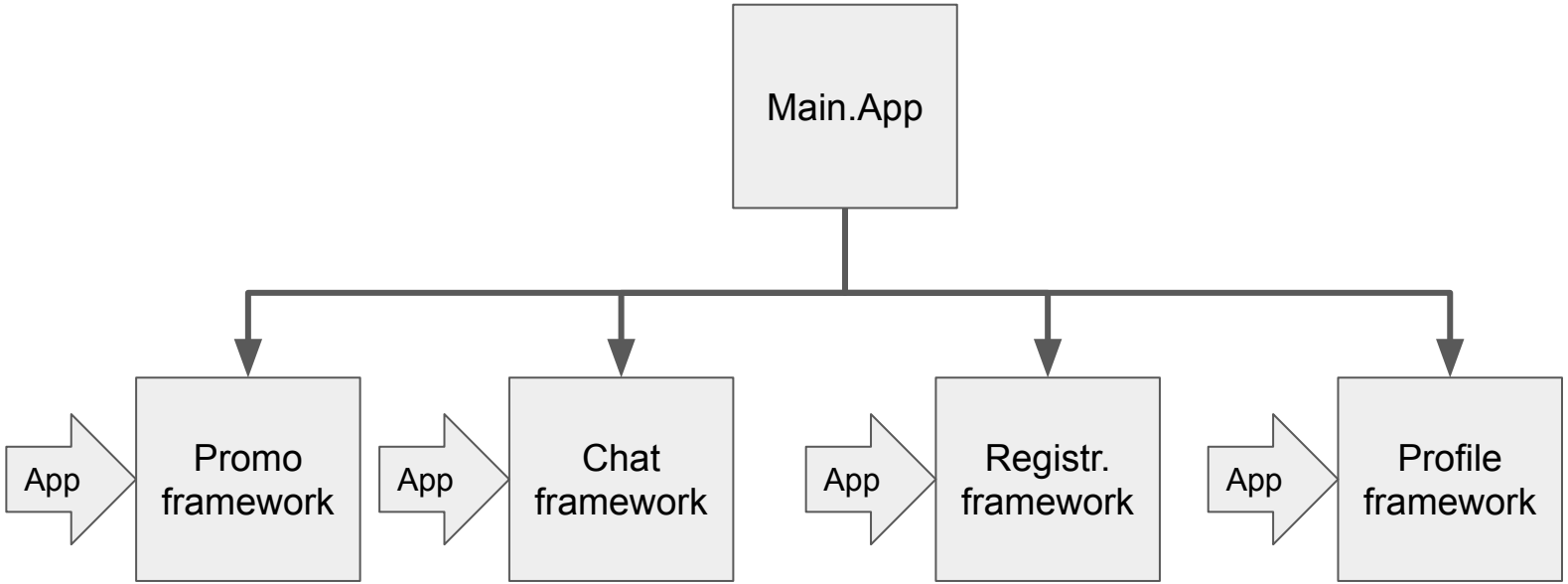
Microapps



Microapps


Main.App
Promo framework
Chat framework
Registr. framework
Profile framework

Microapps

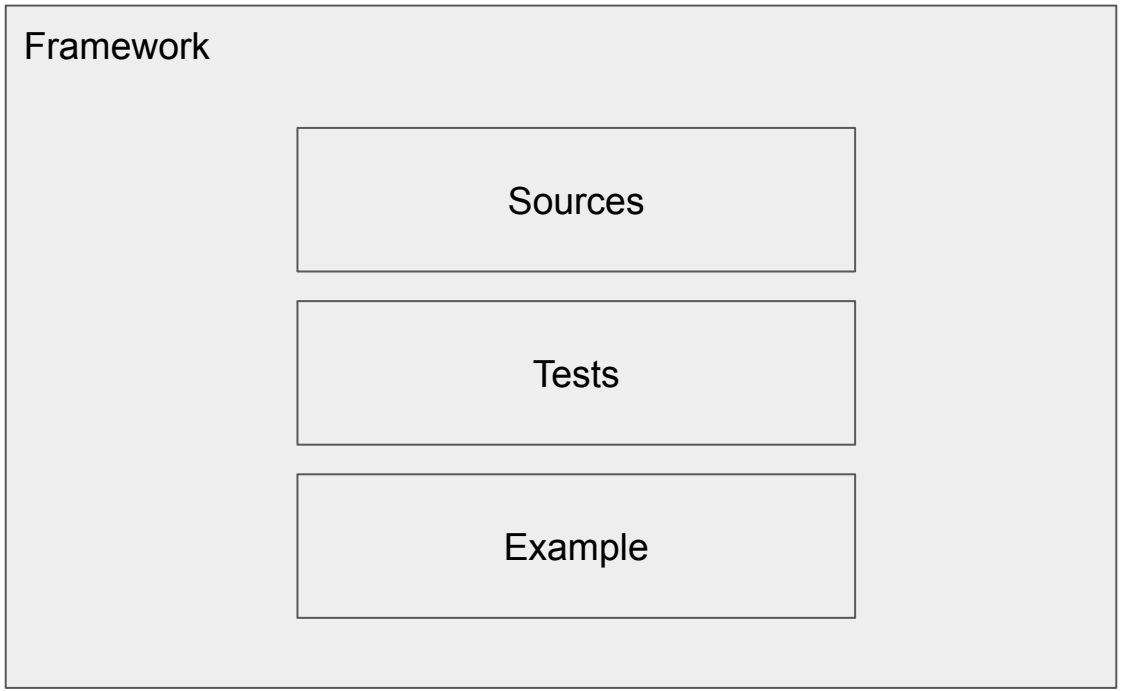

Main.App
App
Promo framework
App
Chat framework
App
Registr. framework
App
Profile framework

Microapp

MicroApp.xcodeproj

```
import MyFramework

let window = UIWindow().with(MyModule.controller)
window.makeKeyAndVisible()
```

Microapp как пример для фреймворка



Microapps на Example-ax

- Никакого пижонства - стабильная работа
- Хорошее дополнение к Framework-у
- Приложение целиком вне production

Недостатки Microapps

- Отдельный проект добавляет много лишнего
- Сложно поддерживать в актуальном состоянии
- Одно приложение - один вариант запуска

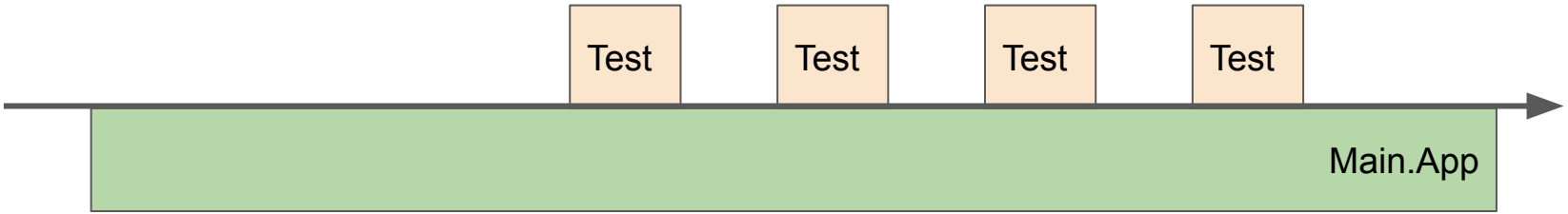
Интерактивный тест

Основная идея

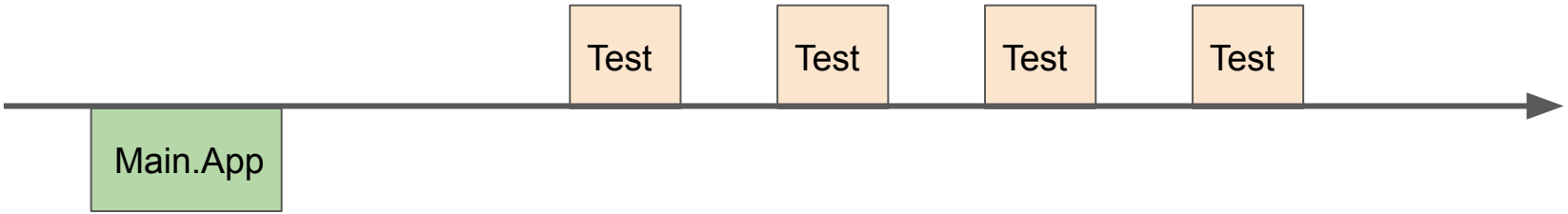
```
let expectation = XCTestExpectation()
wait(for: [expectation], timeout: 1000.0)
```

Основная идея

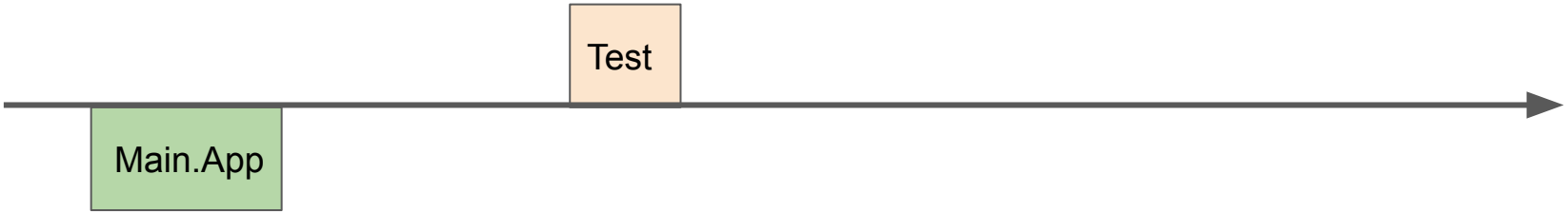
Как это работает



Как это работает



Как это работает



Как это работает

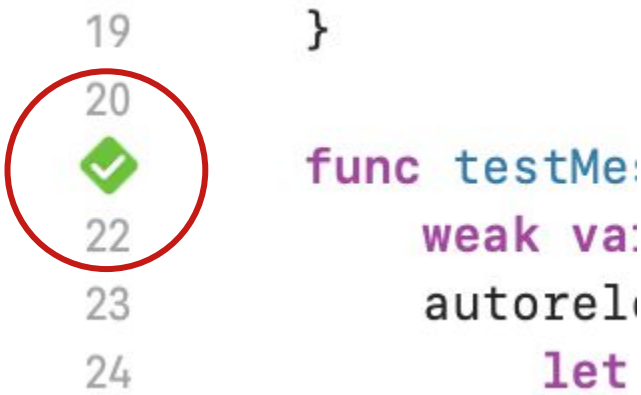
Точка входа

```
let serviceMock = ServiceMock(with: .mockData)
let module = SearchModule(service: ServiceMock())

let window = UIWindow()
window.rootViewController = module.controller
window.makeKeyAndVisible()
```

Тестовый экран

Инфраструктура

Запуск нужного теста



Запуск последнего теста

^ ⌥ ⌘ G

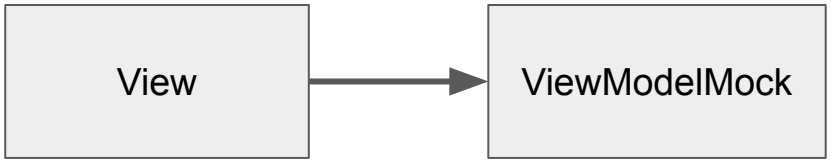
Преимущества

- Стабильная работа
- Тестовые данные из тестового окружения
- Варианты запуска через разные тесты

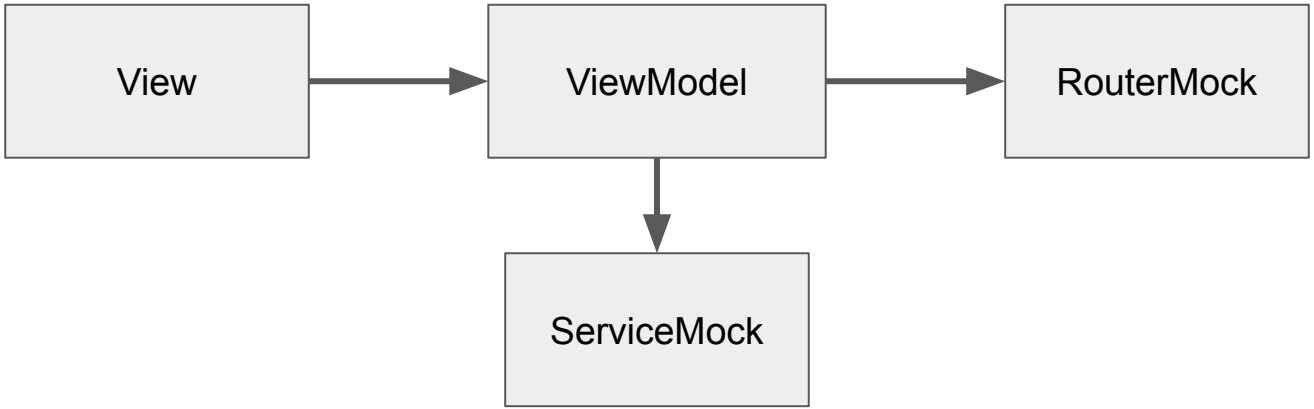
Недостатки

- Не прогрессивно
- Не интерактивно
- Тесты с Host Application

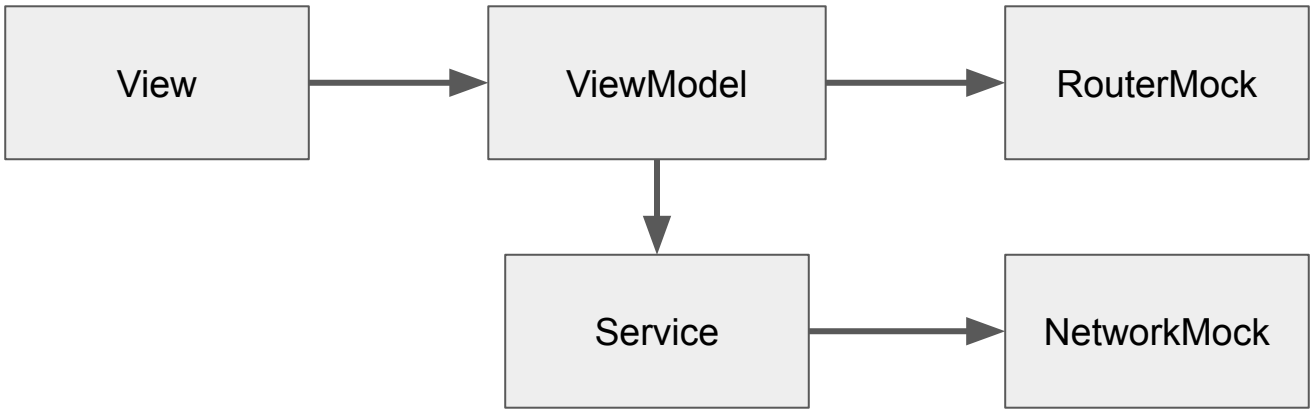
Гибкость Dependency Injection



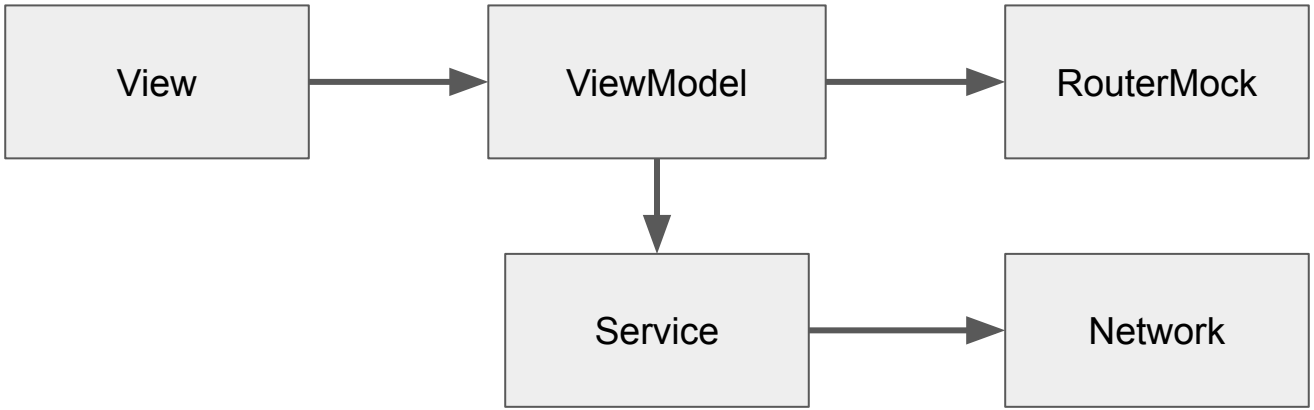
Гибкость Dependency Injection



Гибкость Dependency Injection



Гибкость Dependency Injection



Даже в самой плохой архитектуре

UIApplication.delegate.showPromo()

Чемодан с чудо инструментами

LLDB: expression controller.fillTestData(forProfileID: 322323233)

Детали реализации

```
wait(for: [expectation], timeout: 1000.0)
```

не сместимо с CI

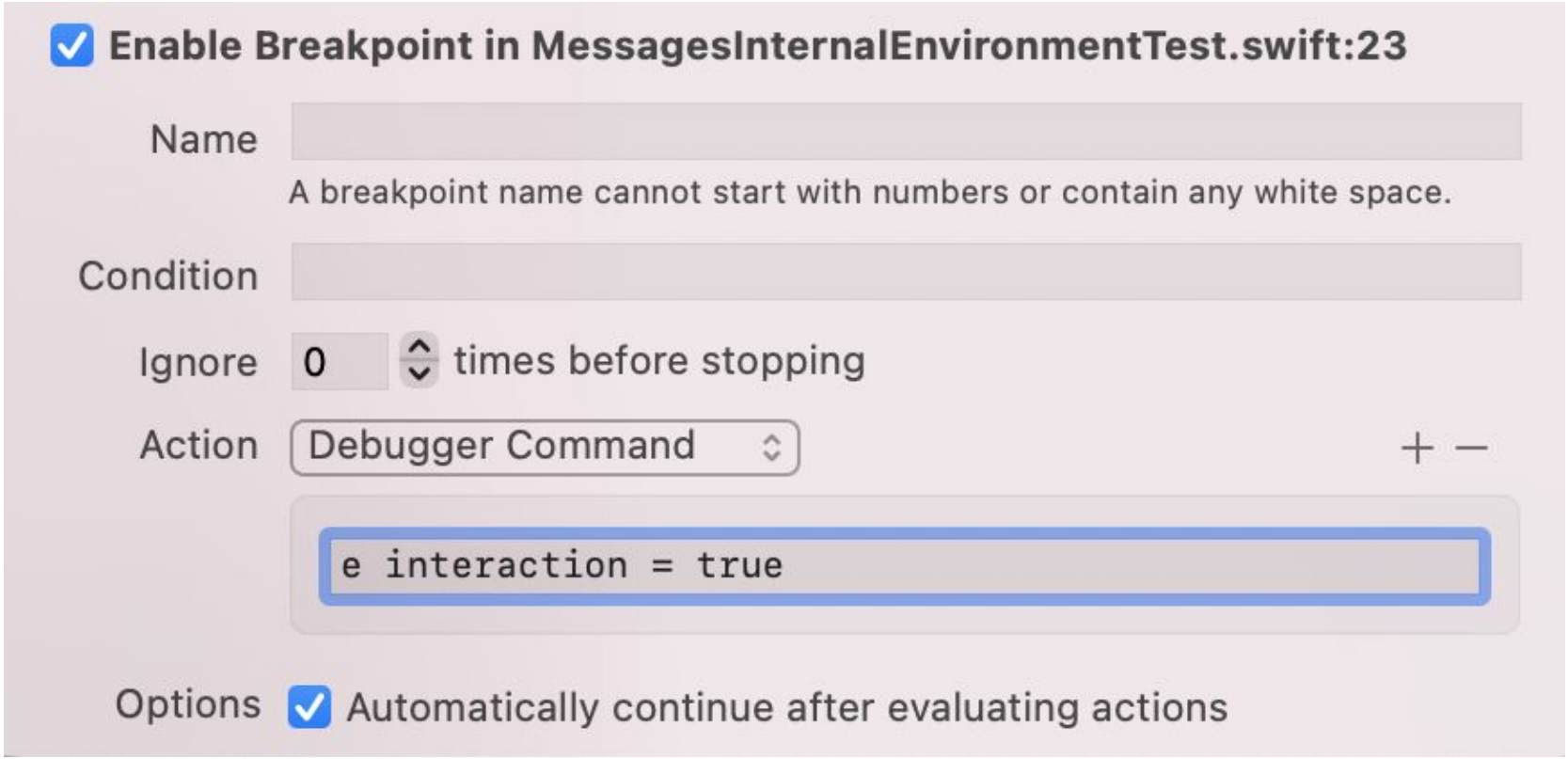
Сложности

- Отключение/включение теста можно закоммитить
- Изменения кода можно закоммитить
- Должен собираться

Решение для Breakpoint

```
var interaction = false
UIWindow().show(controller)
if interaction {
    let expectation = XCTestExpectation()
    wait(for: [expectation], timeout: 1000.0)
}
```

Решения для Breakpoint



Избежать Warning

```
var interaction = false
```

```
let switcher = DebugSwitcher()

final class DebugSwitcher {
    private var enabled = false

    var isOn: Bool { enabled }
    func enable() {
        enabled = true
    }
}
```

Code coverage

```
func testPhotos() {
    profile = .nataliePhotos
    injectTest(in: environment.container)
    coordinator.navigate(
        route: .main(shouldPush: false)
    )
    if switcher.isOn {
        longWait()
    }
}
```

Code coverage для UI
Безопасно для CI

Code coverage

```
func testPhotos() {
    profile = .nataliePhotos
    injectTest(in: environment.container)
    coordinator.navigate(
        route: .main(shouldPush: false)
    )
    if switcher.isOn {
        longWait()
    }
}
```

Включается по надобности

Примеры

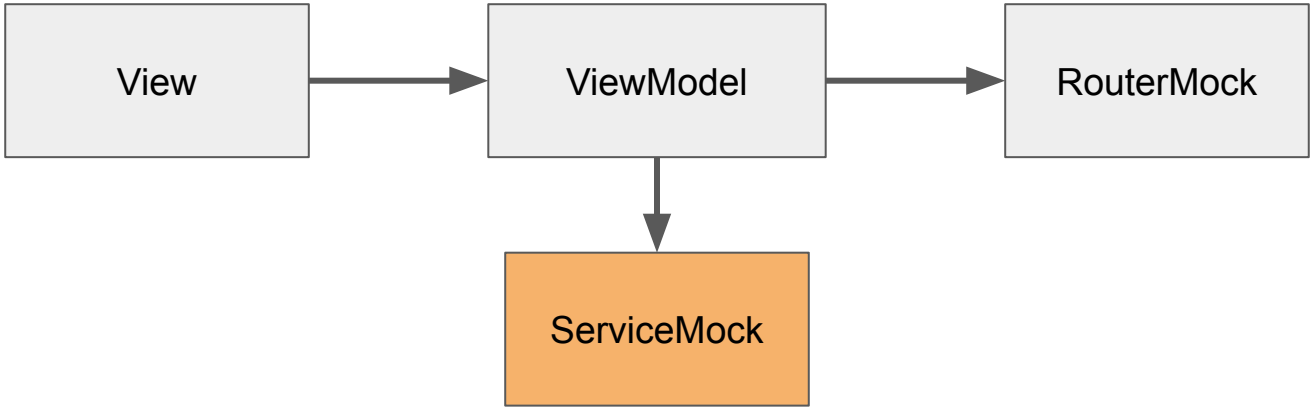
Обработка ошибок

Обработка ошибок

Как создать ошибку

Метод	**Последствия**
Выключить сеть	Stackoverflow не работает
Уложить stage-сервер	Фингал от бэкендера
Сломать запрос	Продакшн в опасности

Подмена сервиса




```
final class SomeServiceMock: SomeServicd {
    var requestResult: Result<Void, Error>?
```

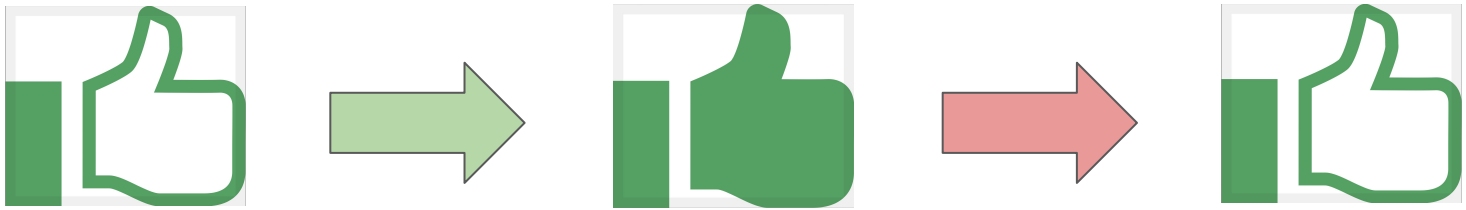
Подстановка ошибки

```
favoriteMock.result = .failure(NSError(domain: "test", code: 0))
```

Красота

- Stackoverflow работает
- Backend не нужен
- Production в безопасности
- Легко показать дизайнеру и менеджеру

Отимистичный переключатель



Сложности

- Промежуточные состояния не видны
- Сложно snapshot-ить
- Network shaper долго настраивать

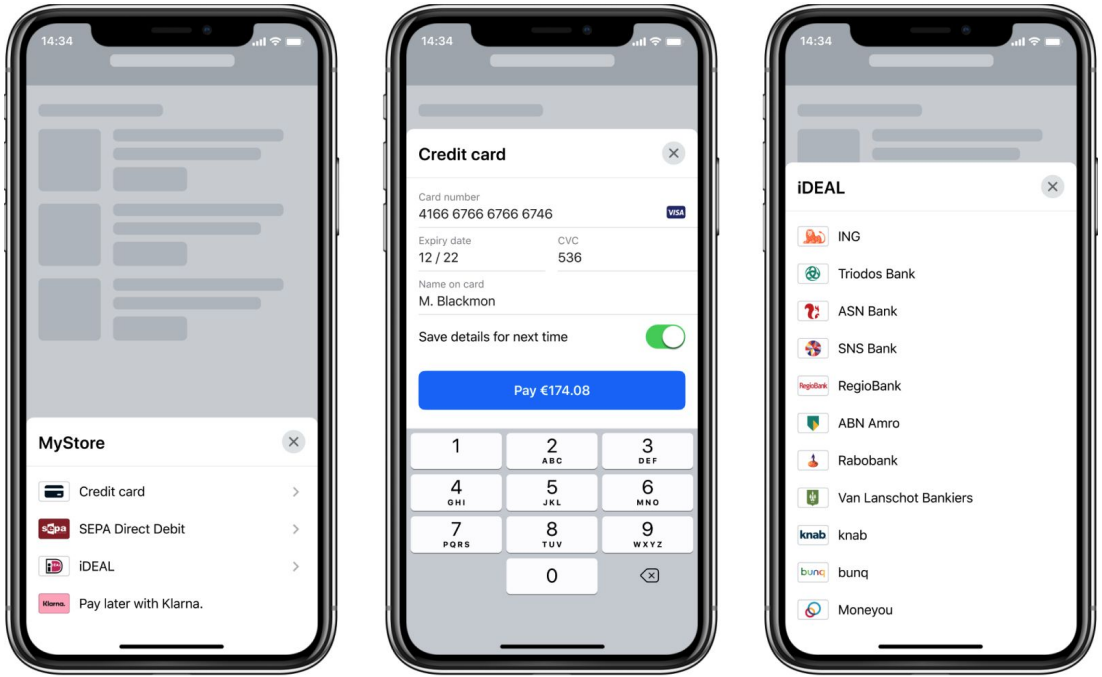
Mock-сервис с задержкой

```
final class SomeServiceMock: SomeServicd {
    public var requestResult: Result<Void, Error>?
    public func with(delay: Int) -> Self {
        self.delay = TimeInterval(delay)
        return self
    }

    DispatchQueue.main.asyncAfter(deadline: .now() + delay) {
```

Красота

- Наглядная работа переключателя
- Production в безопасности
- Легко запустить и потыкать

Система оплаты



https://docs.adyen.com/online-payments/ios/drop-in

Сложности

- Специальные пользователи из разных стран
- Пользователи с заполненными данными
- “Чистые” пользователи

Решение с тестом

- Тест с пустыми данными
- Тест со всеми методами оплаты
- Тест с сохраненными картами

Эпилог

- Не нужно мучаться - используйте короткий путь к своему модулю
- Хорошие тесты ускоряют разработку и отладку
- Используйте тестовое окружение для разработки

Вопросы

Куркин Дмитрий
dmitry.kurkin@spark.net
Telegram: @kurkinm
https://github.com/sclown