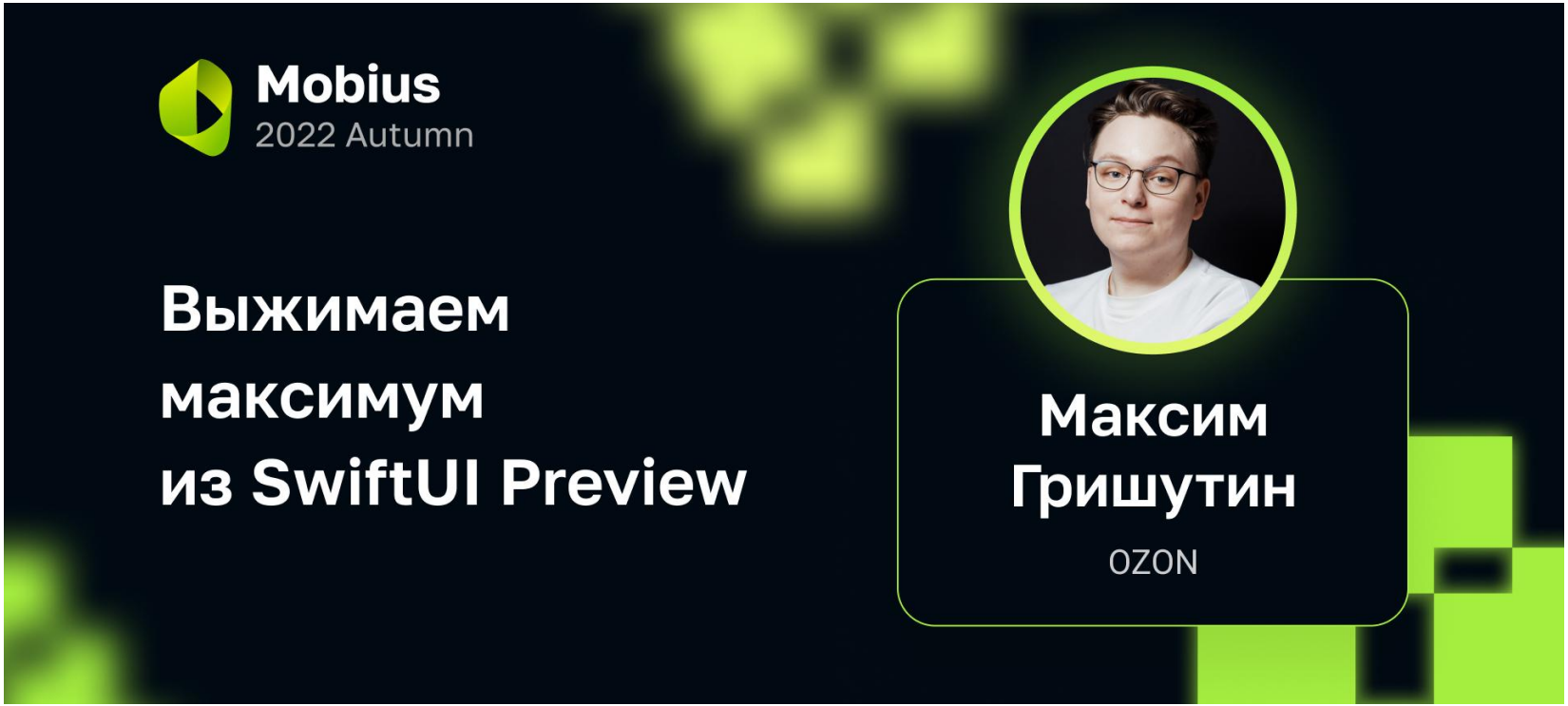
Выжимаем из SwiftUI Preview еще больше

Дмитрий Куркин
xymatic

План

- Mobius 2022: Выжимаем максимум из SwiftUI Preview
- Доступ к Preview в Runtime
- Создание теста в Runtime
- Storybook без кодогенерации
- Отладка Preview
- Preview вместо Micro Apps

Mobius 2022: Выжимаем максимум из SwiftUI Preview



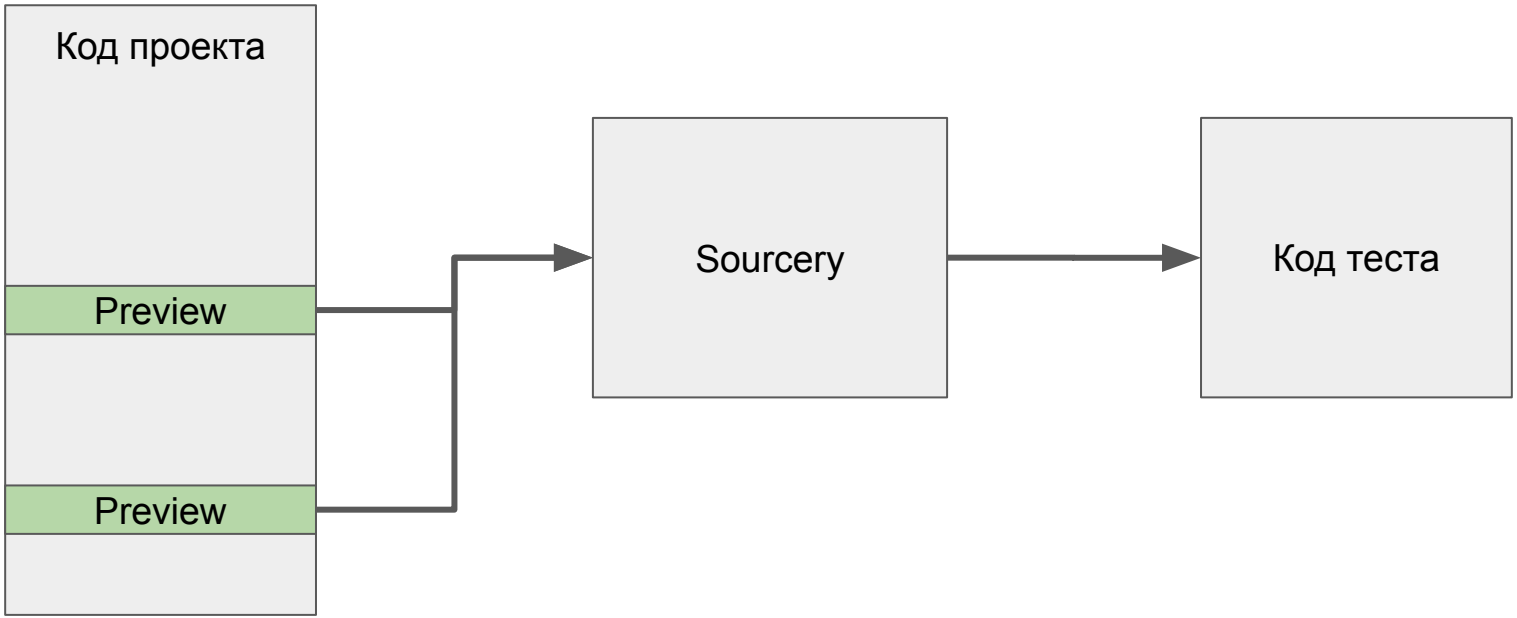
Как обычно используются превью

- Просмотр верстки
- Взаимодействие с View

Как нужно использовать Preview

- Генерация Snapshot-тестов
- Генерация Performance-анализа
- Генерация Playbook.app
- Генерация Accessibility-тестов

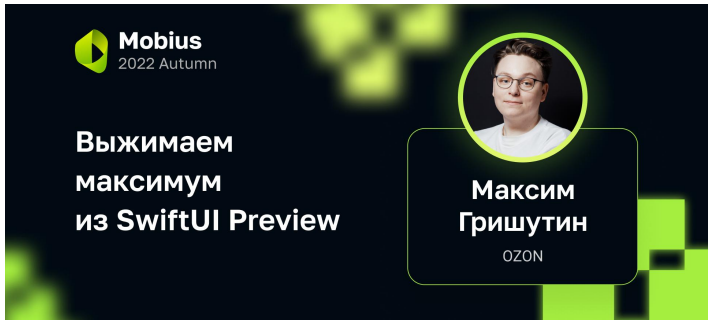
Кодогенерация тестов по Preview



Результаты

- Playbook(Demo) App
- Простой performance-анализ
- Snapshot-тесты
- Accessibility-тесты

Mobius 2022: Выжимаем максимум из SwiftUI Preview



https://www.youtube.com/watch?v=CXDSbJYmMn0

Mobius 2022: Выжимаем максимум из SwiftUI Preview

https://github.com/BarredEwe/Prefire

Mobius 2022 Autumn

https://mobiusconf.com/archive/2022%20Autumn/

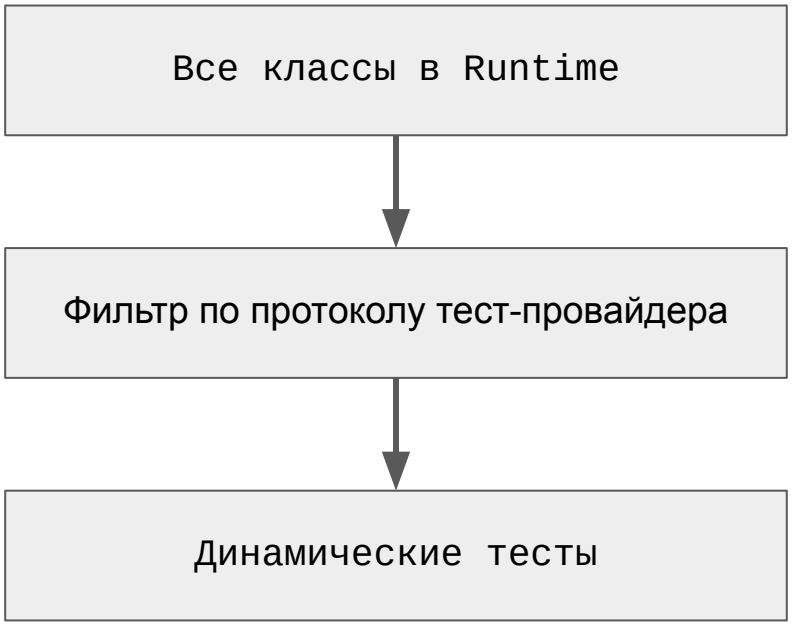
Наш опыт применения подхода

- 380 Snapshot-тестов
- Storybook (Playbook)
- BuildPlugin

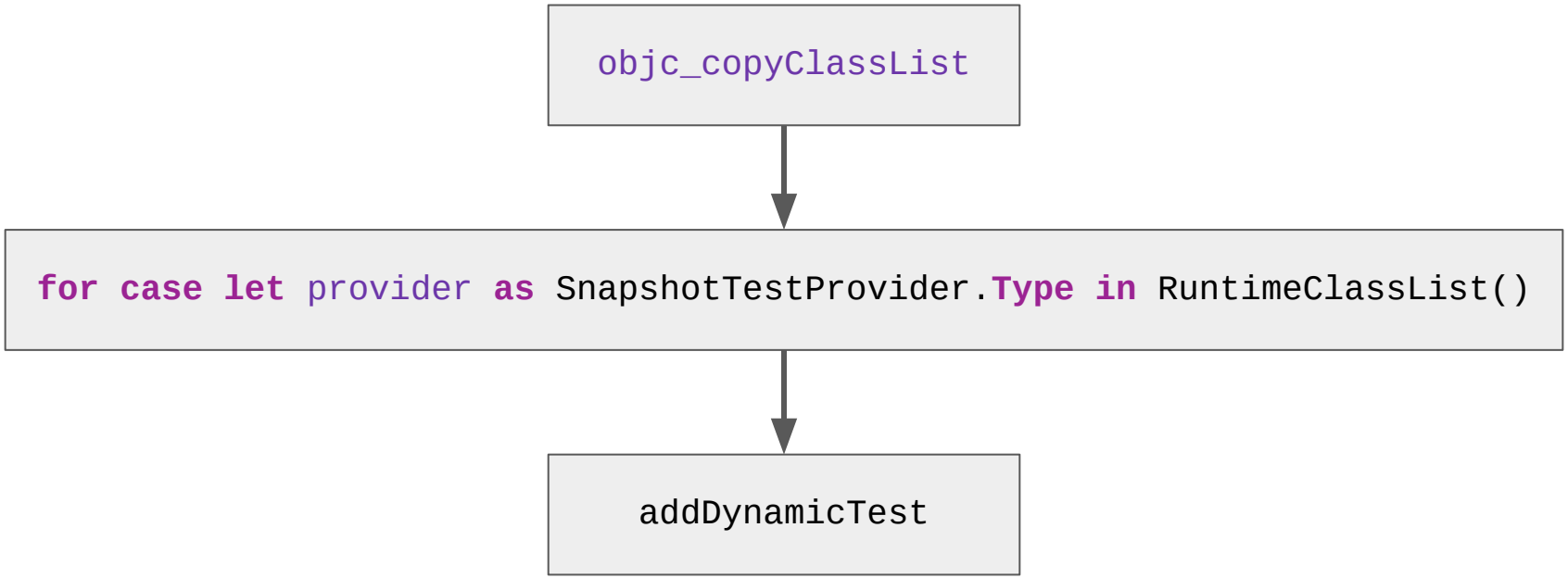
Идея от Grammarly:
Preview можно искать в Runtime

https://youtu.be/HFVhM5lV3ww?t=3029

Те же тесты но в Runtime



Те же тесты но в Runtime



RuntimeClassList

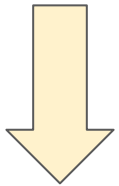
```
public static func allClasses() -> [AnyClass] {
    var count: UInt32 = 0
    let list = objc_copyClassList(&count)
    defer { free(UnsafeMutableRawPointer(list)) }
    let classes = UnsafeBufferPointer(start: list, count: Int(count))
    return classes.compactMap { $0 as AnyClass }
}
```

Фильтр

```
func classes<T>(conformTo _: T.Type) -> [AnyClass] {
    allClasses().filter { $0 is T }
}

classes(conformTo: SnapshotsProvider.Type.self)
```

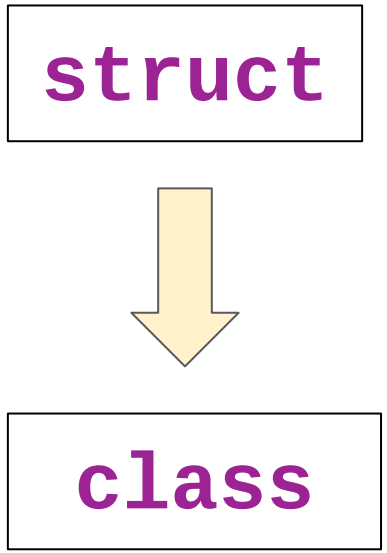
Но есть нюанс

```
struct InputTextFieldPreviews: PreviewProvider
```



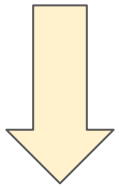
```
class InputTextFieldPreviews: PreviewProvider
```

Но есть нюанс



Вспомогательные протоколы

```
struct InputTextFieldPreviews: PreviewProvider
```



```
class InputTextFieldPreviews:
    PreviewProvider,
    SnapshotsProvider,
    StorybookProvider
```

Вспомогательные протоколы

- Включение/выключение Snapshot-теста
- Добавление/исключение из Storybook
- Расширение с дополнительными методами (allDevices, allLanguages, AnyView, UIView)

Динамический Unit-Test

```
class XCTestCase
    class var defaultTestSuite: XCTestSuite { get }
```

XCTestSuite.h

* **@class** XCTestSuite
* A concrete subclass of XCTest, XCTestSuite is a collection of test cases. Suites
* are usually managed by the IDE, but **XCTestSuite also provides API for dynamic test**
* **and suite management**:

```
XCTestSuite *suite = [XCTestSuite testSuiteWithName:@"My tests"];
[suite addTest:[MathTest testCaseWithSelector:@selector(testAdd)]];
[suite addTest:[MathTest testCaseWithSelector:@selector(testDivideByZero)]];
```

Имя теста

`init(invocation: NSInvocation?)` Initializes a test case with an invocation.

`init(selector: Selector)` Initializes a test case with a selector.

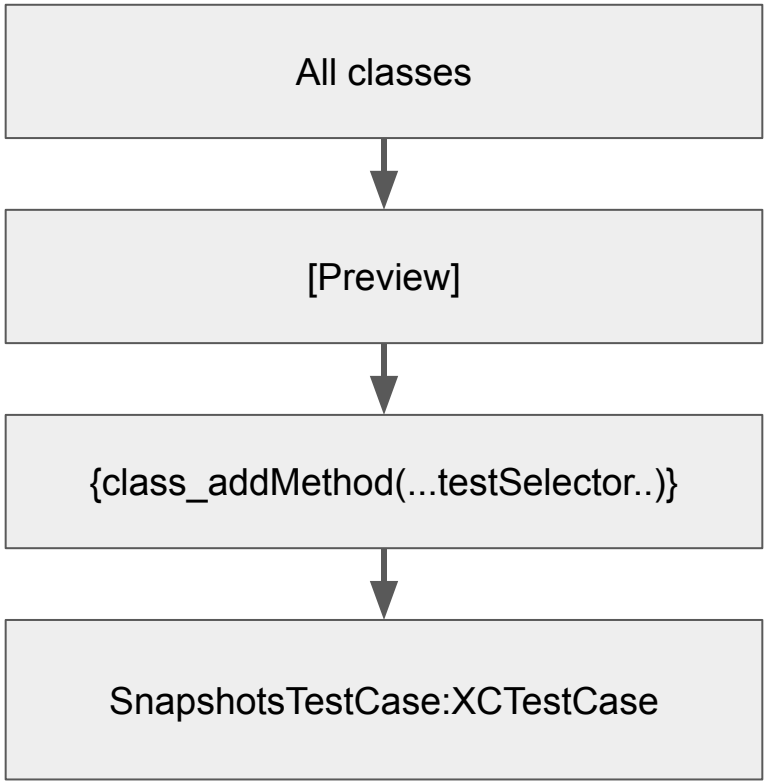
NSObject addMethod

```
extension NSObject {
    static func addM(name: String, block: @escaping () -> Void) -> Selector {
        let impBlock: @convention(block) () -> Void = block
        let blockIMP = imp_implementationWithBlock(
            unsafeBitCast(impBlock, to: AnyObject.self)
        )
        let testSelector = Selector(name)
        class_addMethod(Self.self, testSelector, blockIMP, "v@:" )
        return testSelector
    }
}
```

Итоговое решение

```
override class var defaultTestSuite: XCTestSuite {
    let previews = RuntimeTools.classes(conformTo: SnapshotsProvider.Type.self)
    let suite = XCTestSuite(forTestCaseClass: WidgetsDynamicTests.self)
    for previewClass in previews {
        let previewName = String(describing: previewClass)
        let testName = "test\(previewName)"
        let testCase = dynamicTest(name: testName) {
            (previewClass as? SnapshotsProvider.Type)?.snapshots.forEach {
                if let failure = verifySnapshot(
                    of: $0.value,
                    as: .image(precision: 0.99, perceptualPrecision: 0.99, scale: 0.75),
                    snapshotDirectory: snapshotDirectory.absoluteString,
                    testName: testName
                ) {
                    XCTFail(message)
                }
            }
        }
        suite.addTest(testCase)
    }
    return suite
}
```

Итоговое решение



Storybook
(Playbook)

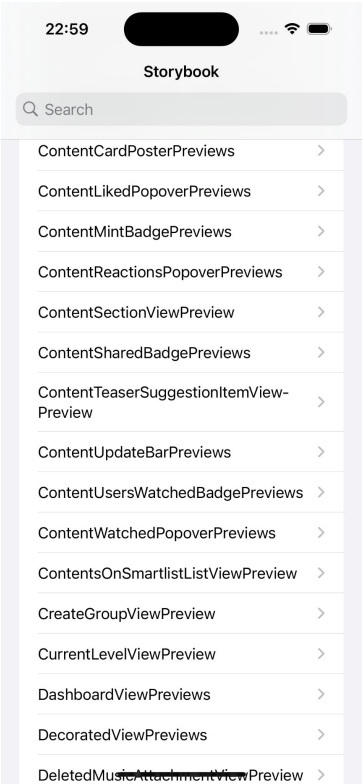
Базовая часть

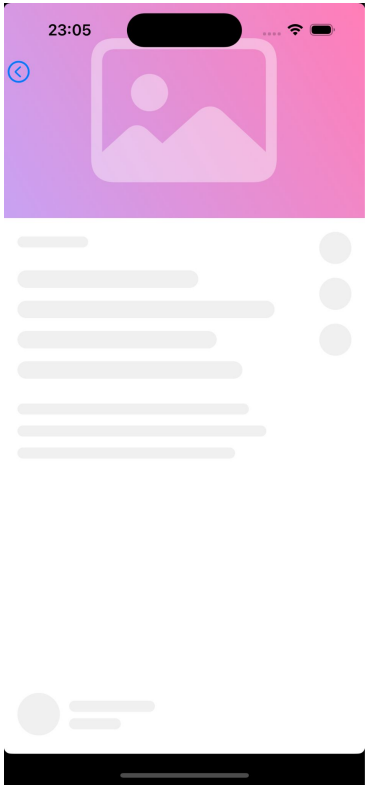
```
NavigationStack {
    Form {
        ForEach(pages, id: \.name.hashValue) { page in
            NavigationLink(page.name) {
                StorybookPageView(
                    deviceConfig: _deviceConfig,
                    page: page
                )
            }
        }
    }
}
```

Подгрузка Preview

```
static var all: [StorybookPage] {
    let pages = RuntimeTools.classes(
        conformTo: StorybookProvider.Type.self
    )
    return pages.map { className in
        (className as? StorybookProvider.Type)
            .pageWithName(String(describing: className))
    }
}
```

Storybook


22:59
Storybook
Search
ContentCardPosterPreviews
ContentLikedPopoverPreviews
ContentMintBadgePreviews
ContentReactionsPopoverPreviews
ContentSectionViewPreview
ContentSharedBadgePreviews
ContentTeaserSuggestionItemView-Preview
ContentUpdateBarPreviews
ContentUsersWatchedBadgePreviews
ContentWatchedPopoverPreviews
ContentsOnSmartlistListViewPreview
CreateGroupViewPreview
CurrentLevelViewPreview
DashboardViewPreviews
DecoratedViewPreviews
DeletedMusicAttachmentViewPreview


23:05

Подходит не всегда

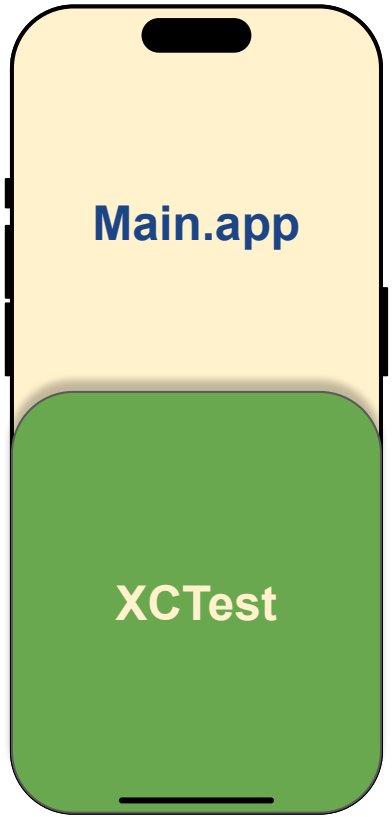
- NavigationStack требует дополнительных элементов
- Маленькие элементы надо дополнительно центрировать
- Побочные эффекты

Превью можно использовать в интерактивном тесте

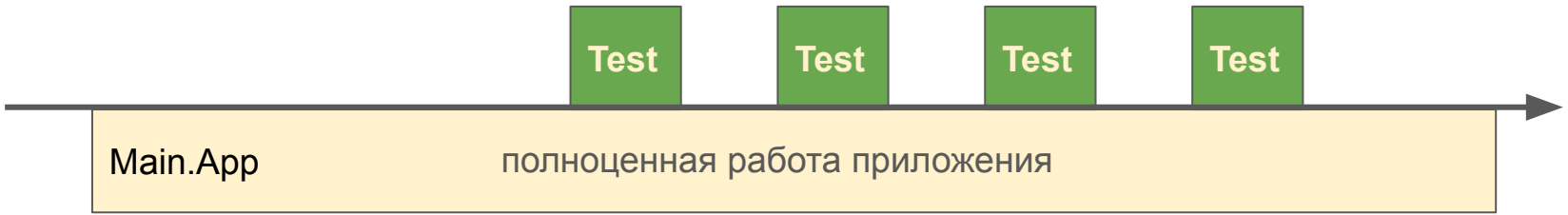
Интерактивный тест - это как Snapshot-тест, но еще можно потыкать

Как потыкать “Snapshot”

Unit-test с Host-Application



Unit-тест с Host-Application



Preview в интерактивном тесте

```
func testPreview() {
    window().show(
        StorybookProvider.find(previewName))
    longWait()
}
```

longWait

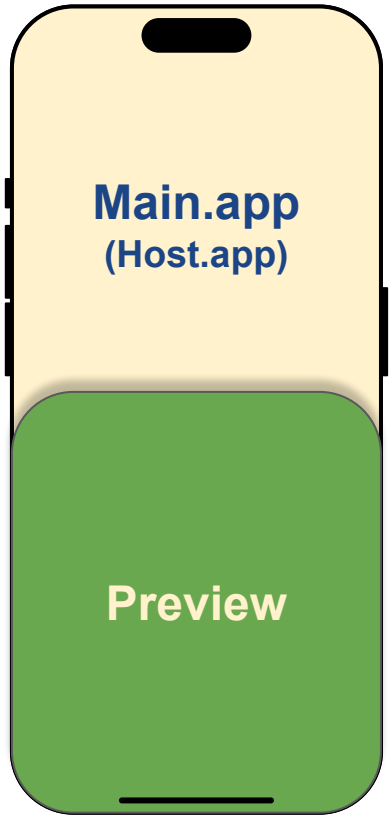
```
let expectation = XCTestExpectation()
wait(for: [expectation], timeout: 1000.0)
```

Интерактивный тест

Unit-test очень долгим ожиданием

🚫 XCUITest

🚫 XCTAssert



Чем это лучше чем само Preview?

Особенность жизненного цикла Preview

- Динамическое обновление
- Частые перезапуски
- Переключения на другие Preview

Сборка и запуск

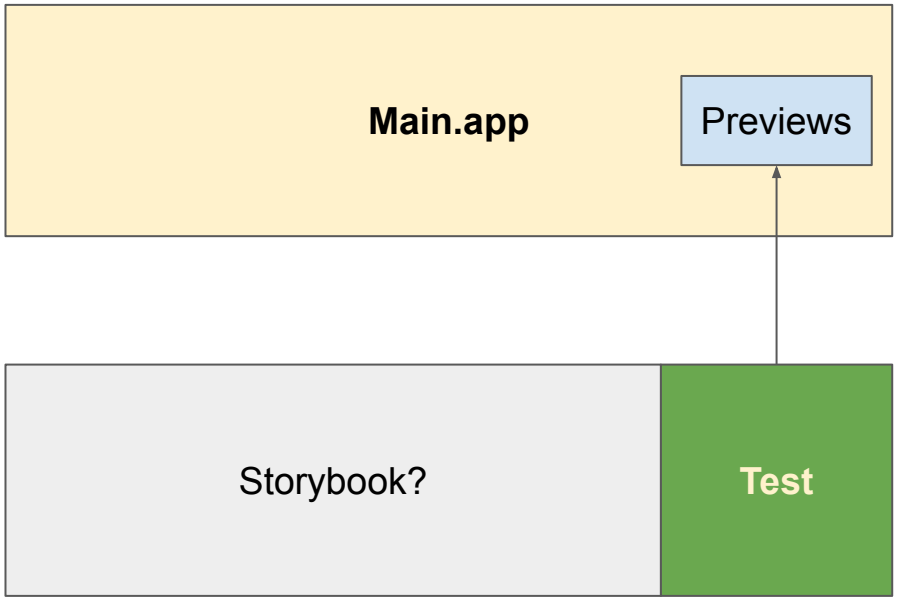
- Специальная папка в DerivedData (PreviewBuild)
- Зависимость от схемы
- Ошибки в маленьком окошке

Ошибки в маленьком окошке

Преимущества Preview в интерактивном тесте

- Обычная сборка приложения и тестов
- Контроль над запуском и остановкой
- Отладчик

Обойтись без Storybook или где взять Host.app



Cocoapods предлагает решение из коробки

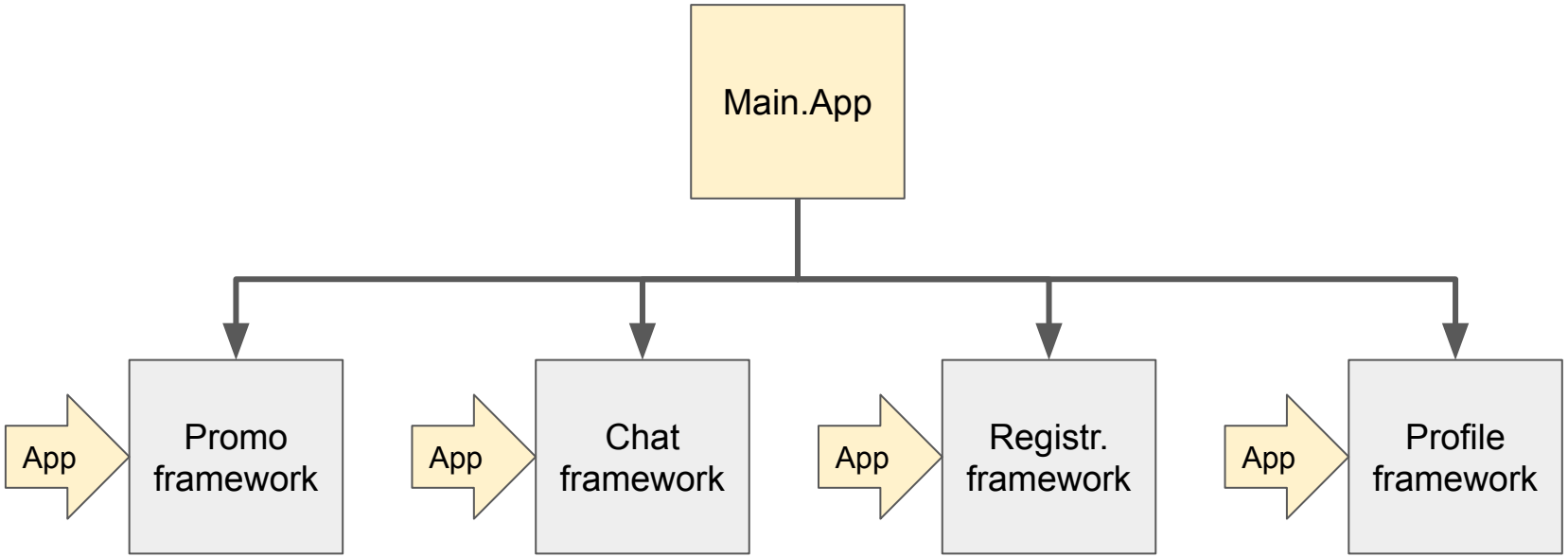
```
s.test_spec 'Tests' do |test_spec|
    test_spec.requires_app_host = true
    test_spec.source_files = 'Tests/*.{h,m}'
end
```

Для SPM придется создать приложение

```
import SwiftUI
@main
struct InteractivePreviewApp: App {
    var body: some Scene {
        WindowGroup {
            Text("Run interactive tests")
        }
    }
}
```

Preview вместо Micro Apps

Micro Apps в многомодульном проекте



Preview это не обязательно одна View

```
NavigationStack {
    Form {
        ForEach(pages, id: \.name.hashValue) { page in
            NavigationLink(page.name) {
                StorybookPageView(
                    deviceConfig: _deviceConfig,
                    page: page
                )
            }
        }
    }
}
```

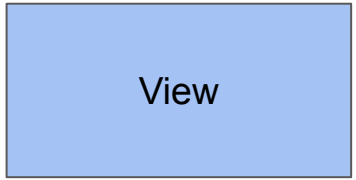
Применение “длинного” Preview

- Весь модуль на Mock-сервисах и Mock-данных
- Модуль с медленным интернетом
- Модуль с ошибками сервисов

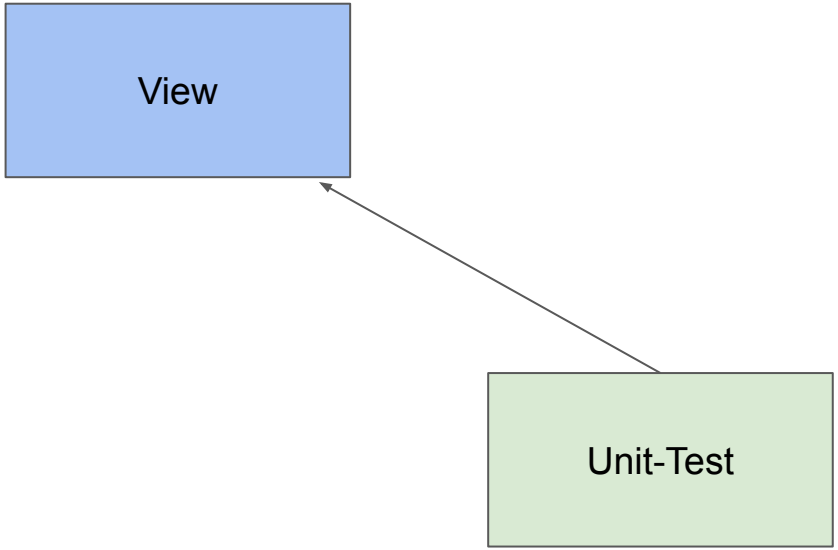
Preview и продакшен-код

```
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
    }
}
```

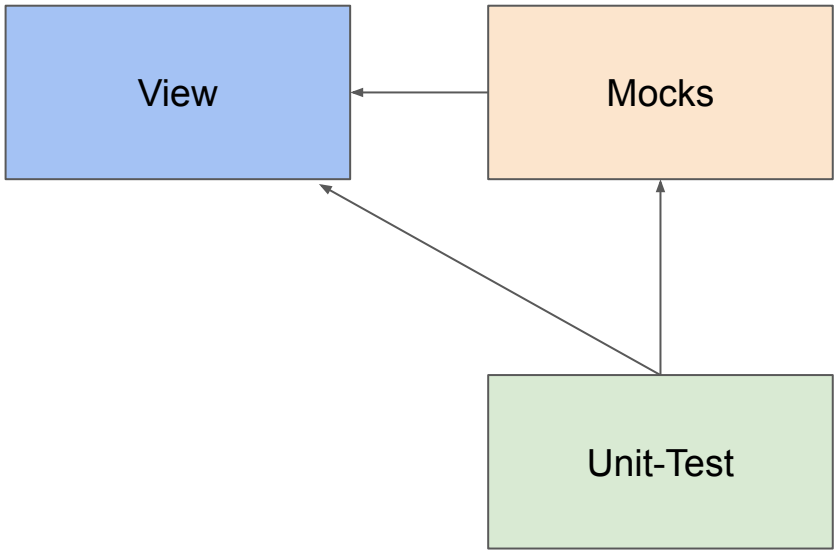
Mocks, Unit-тест и Preview



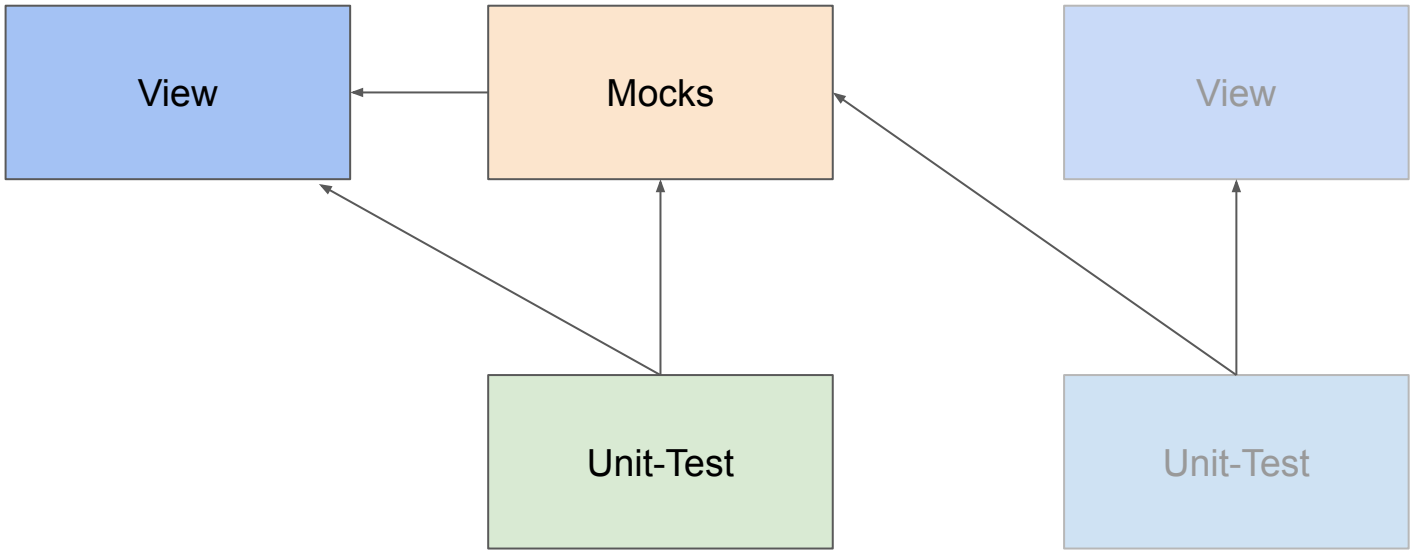
Mocks, Unit-тест и Preview



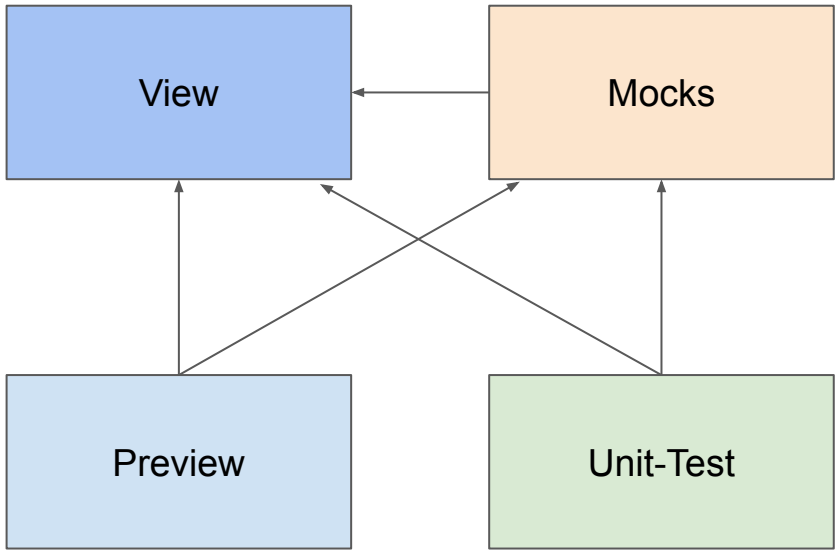
Mocks, Unit-тест и Preview



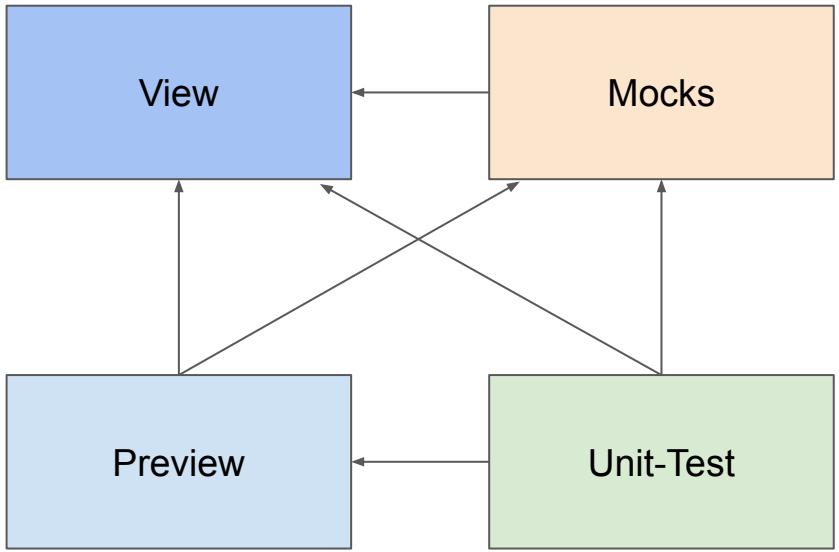
Mocks, Unit-тест и Preview



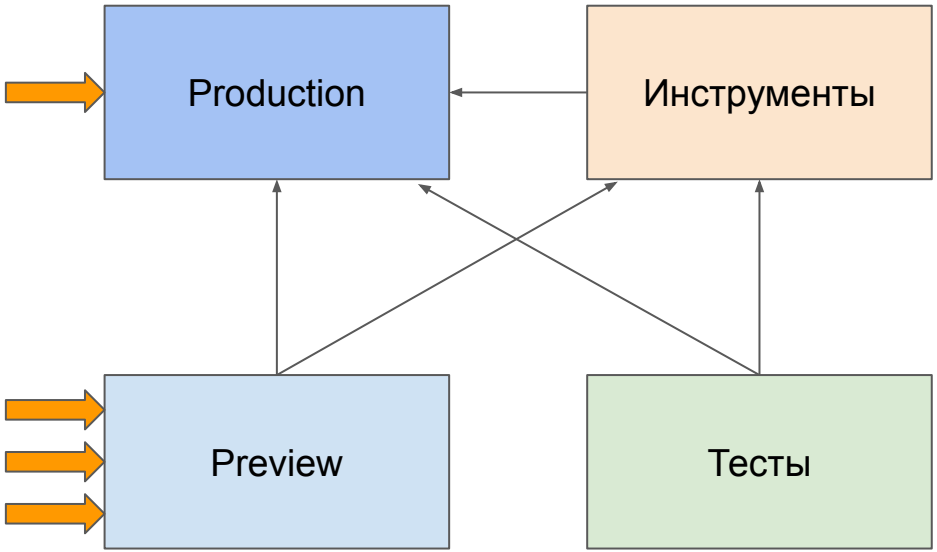
Mocks, Unit-тест и Preview



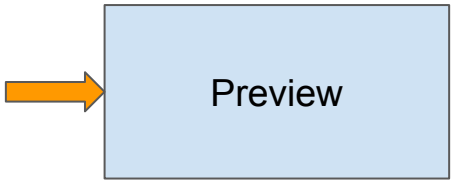
Mocks, Unit-тест и Preview



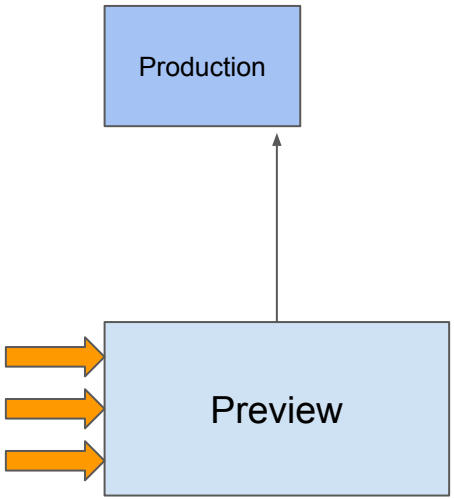
Разработка через Preview



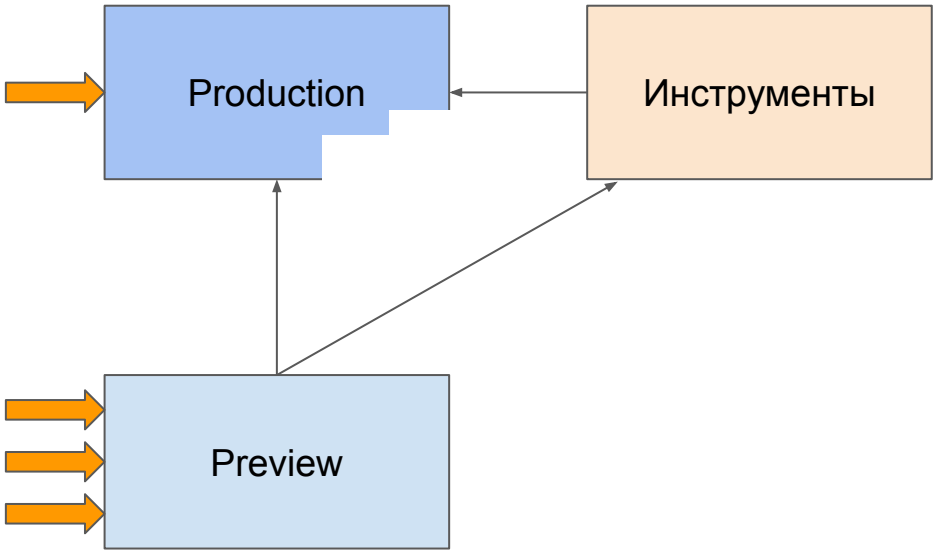
Разработка через Preview



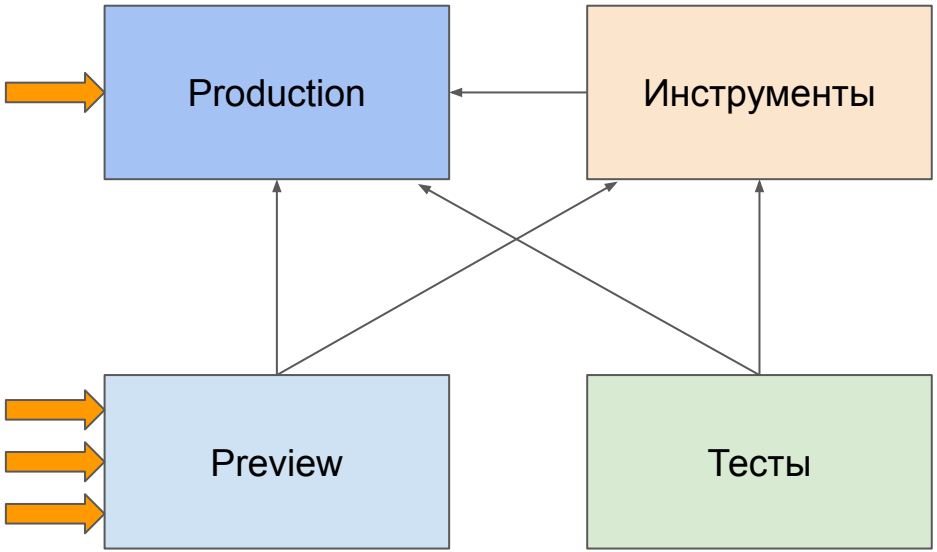
Разработка через Preview



Разработка через Preview



Разработка через Preview



Итого без кодогенерации отжато:

- Snapshot-тесты
- Storybook
- Интерактивный тест
- TDD - Превью драйвен девелопмент

Вопросы

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