БАНК

Альтернативы XCUITest, или Как и зачем разработчику писать автотесты

19.10.2024

Кирилл Володин | Руководитель отдела

@leoniknik

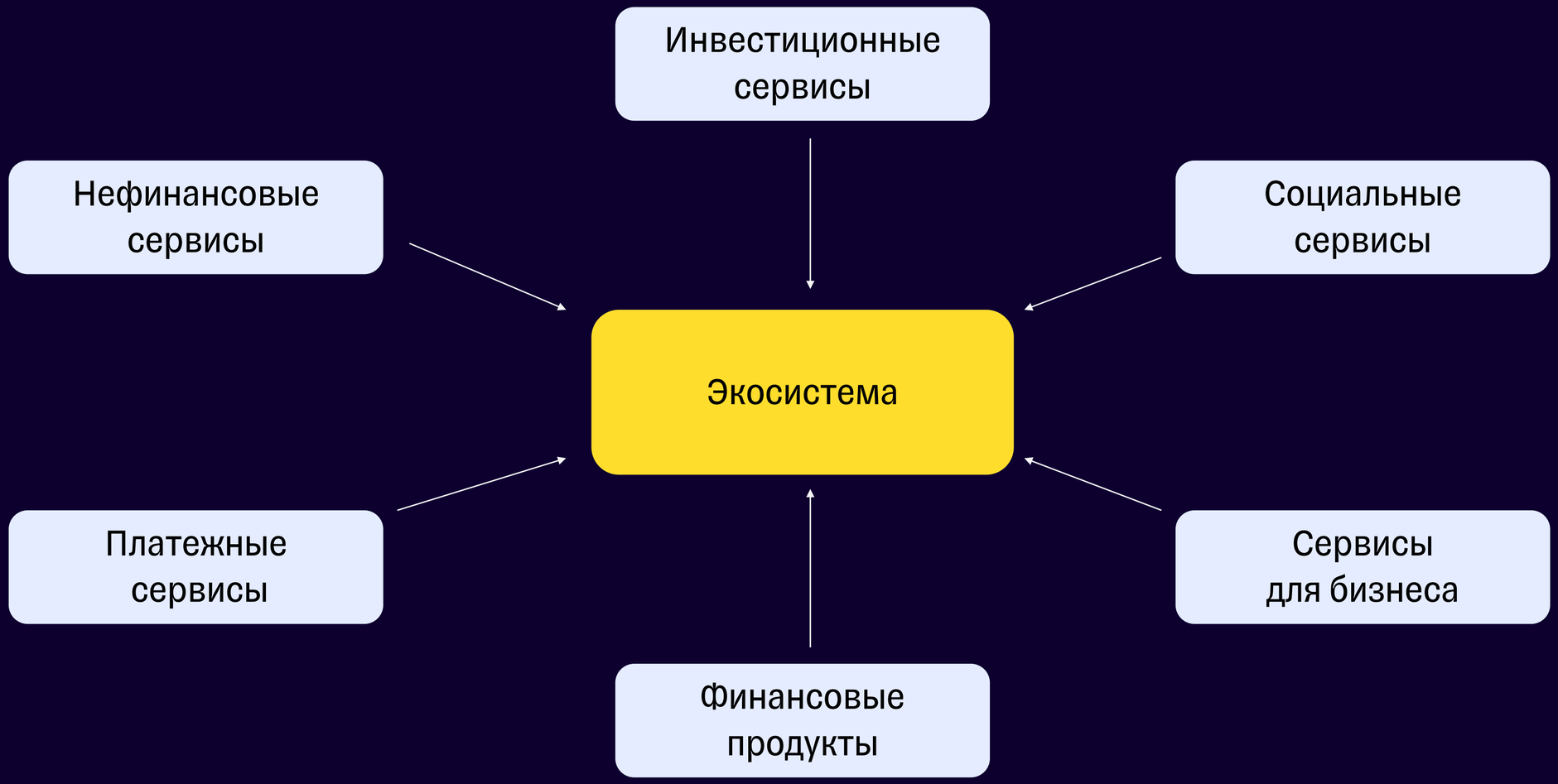
План доклада

1. Почему стоит писать автотесты
2. XCUITest
3. Unit тесты на стероидах
4. KIF
5. Сравнение подходов

План доклада

1. Почему стоит писать автотесты
2. XCUITest
3. Unit тесты на стероидах
4. KIF
5. Сравнение подходов

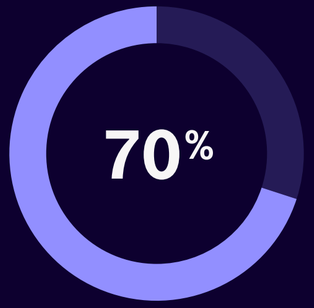
Экосистема Т-Банк



Автоматизация

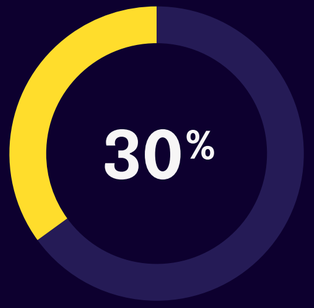
Всего тест-кейсов

80 тысяч



Автоматизированные тест-кейсы

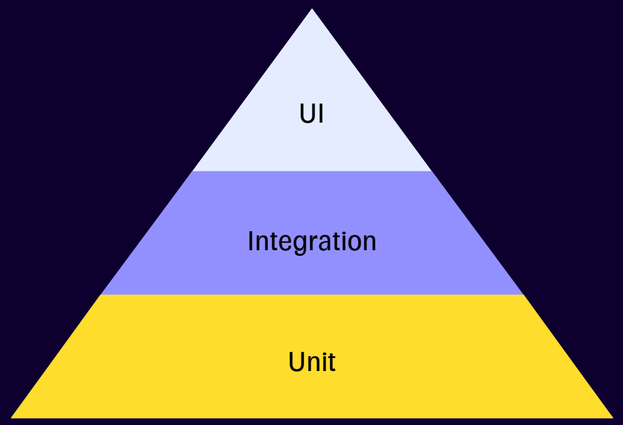
56 тысяч



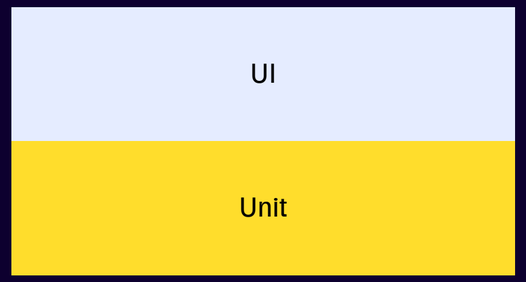
Ручные проверки

24 тысяч

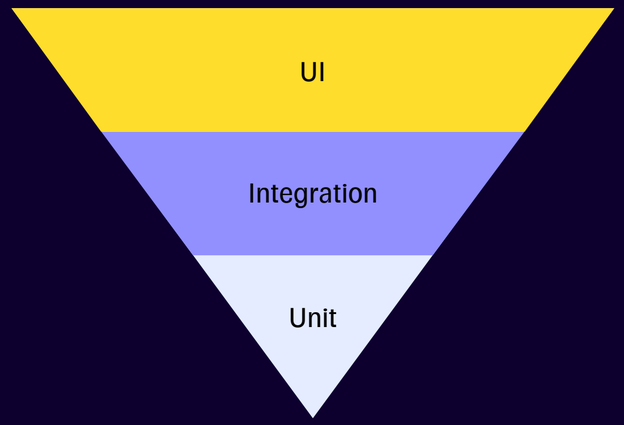
Соотношение автотестов



Пирамида



Что-то среднее

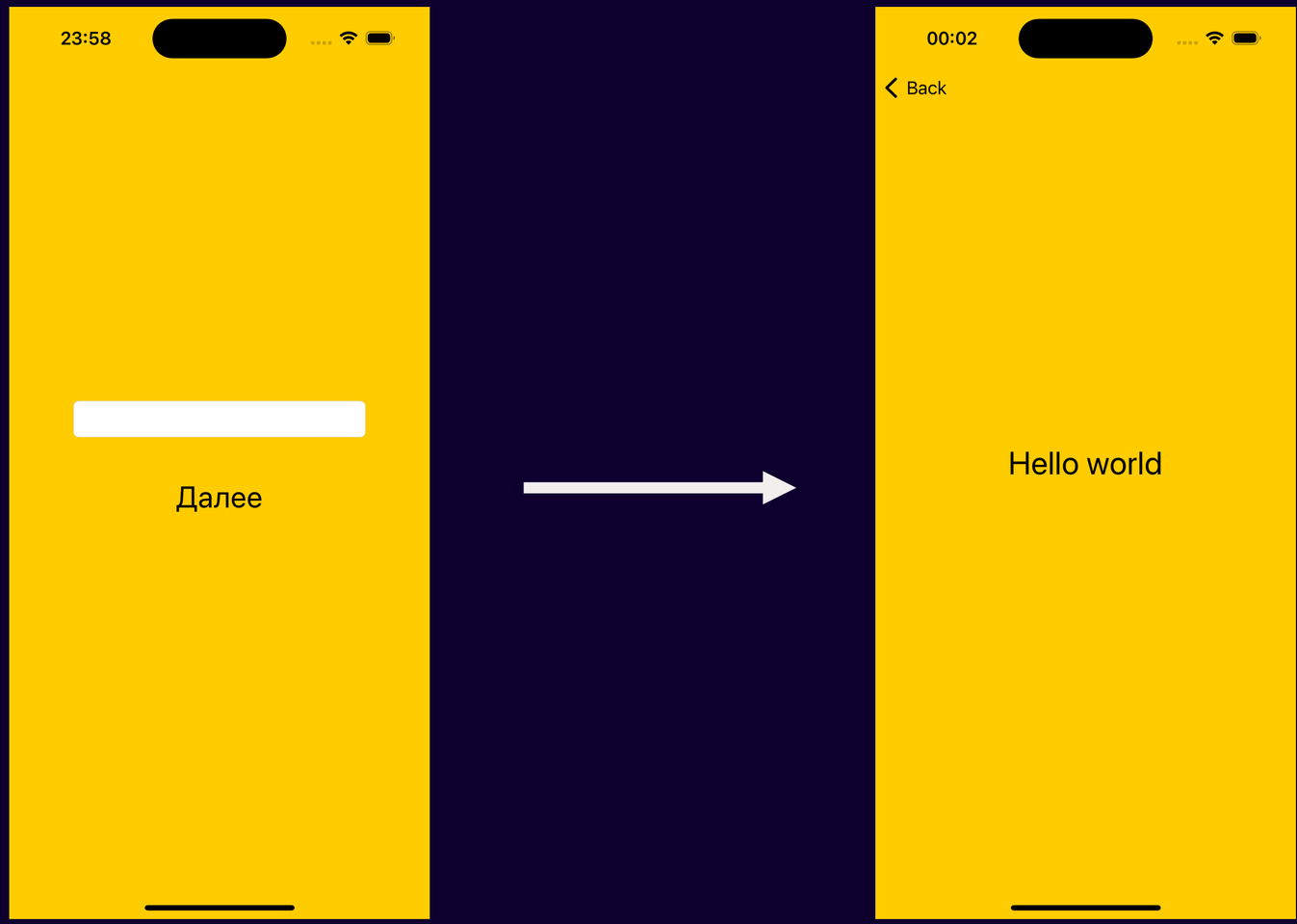


Мороженка

План доклада

1. Почему стоит писать автотесты
2. XCUITest
3. Unit тесты на стероидах
4. KIF
5. Сравнение подходов

Что будем тестировать



Экран логина

Главный экран

Тест-кейс

Предусловие

1. Неавторизованный пользователь

Сценарий

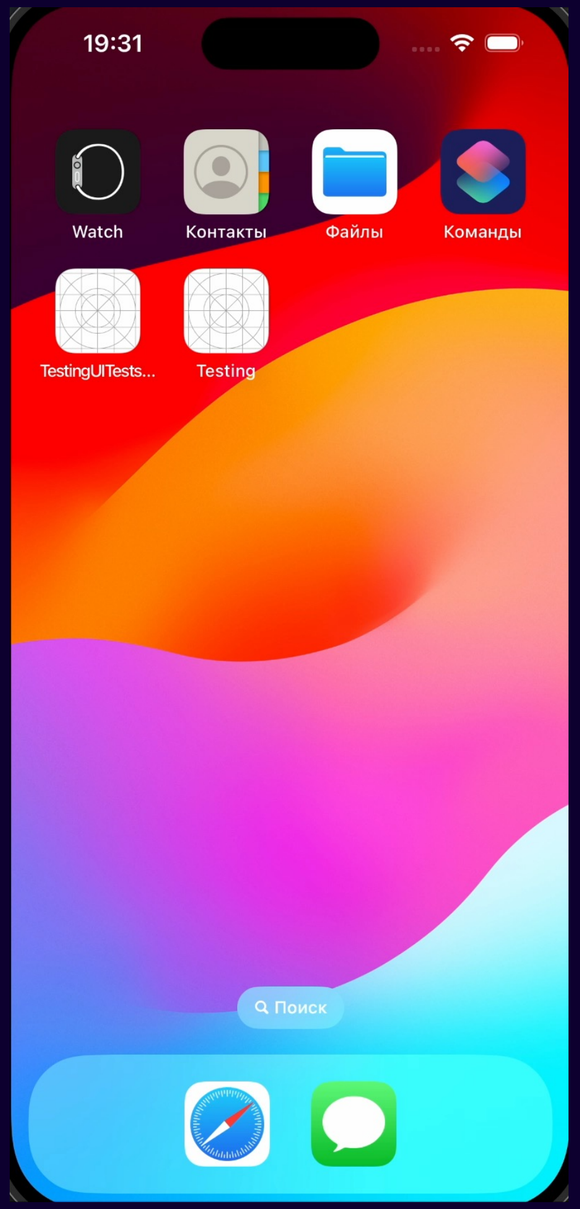
1. На экране логина ввести свой никнейм
2. Нажать кнопку ***Далее***

Ожидаемый результат

1. Открыт главный экран приложения
2. По центру экрана находится текст ***Hello world***

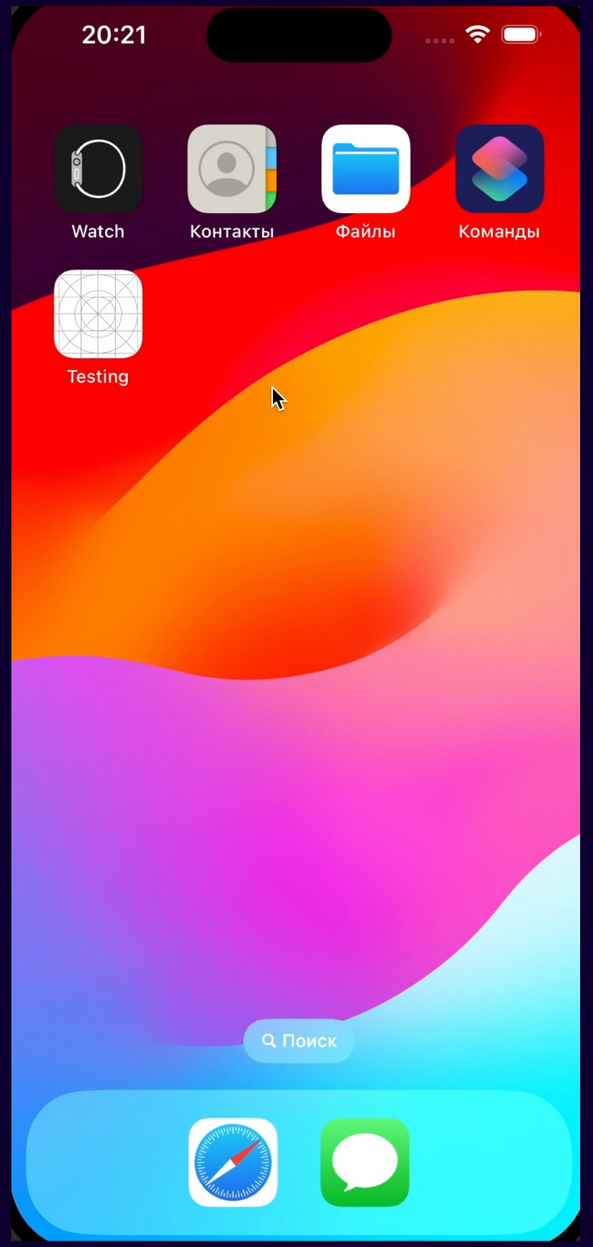
XCUI тест. Скорость

```
final class TestingUITests: XCTestCase {
    func testExample() throws {
        // given
        let app = XCUIApplication()
        app.launch()

        // when
        let element = app.textFields["loginField"]
        element.tap()
        element.typeText("Nickname")
        app.buttons["nextButton"].tap()

        // then
        XCTAssert(app.staticTexts["Hello world"].exists)
    }
}
```



XCUI тест. Нестабильность

```
    func testExample() throws {
19      let app = XCUIApplication()
20      app.launch()
21
22      let element = app.textFields["loginField"]
23      element.tap()
24      element.typeText("Nickname")
25      app.buttons["nextButton"].tap()
26
27      XCTAssert(app.staticTexts["Hello world"].exists)   XCTAssertTrue failed
28  }
```



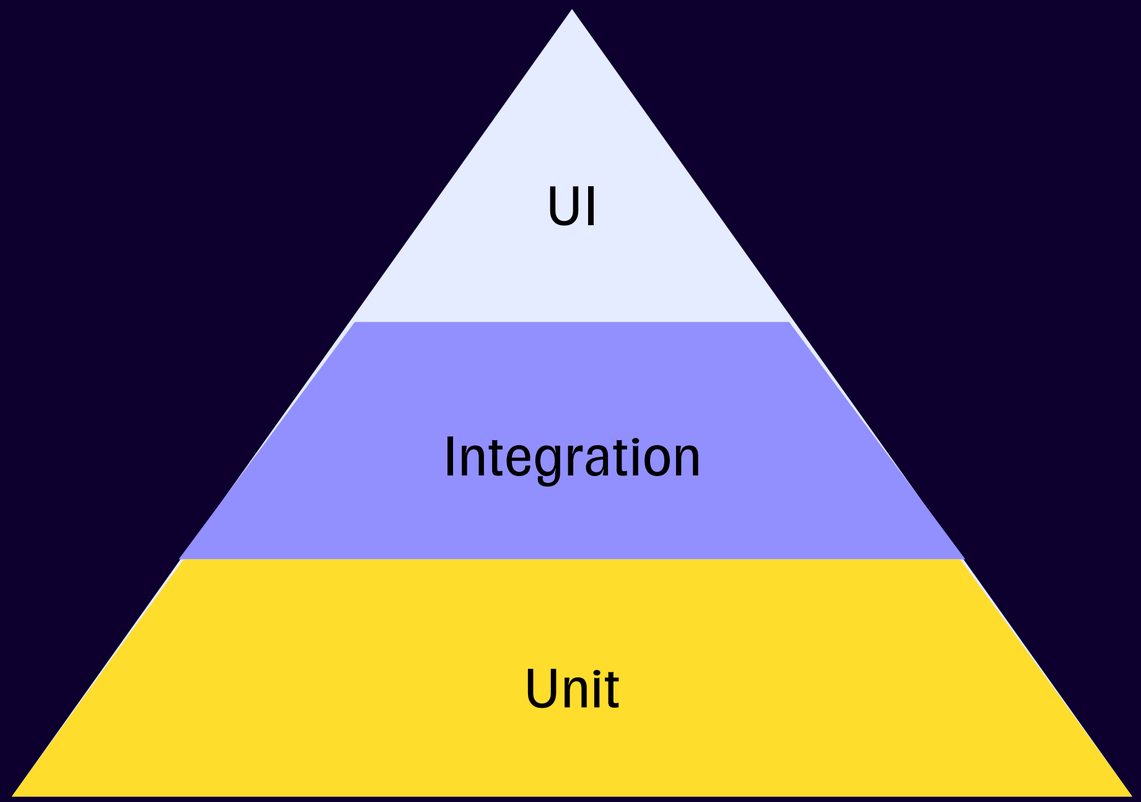
Проблемы XCUITest

- Нестабильность
- Долгое выполнение
- Нельзя просто протестировать только кусочек UI или начать с середины флоу

Практики FIRST

- Быстрота (Fast) ✗
- Независимость (Independent) ✓
- Повторяемость (Repeatable) ✗
- Очевидность (Self-Validating) ✓
- Своевременность (Timely) ✗

Пирамида

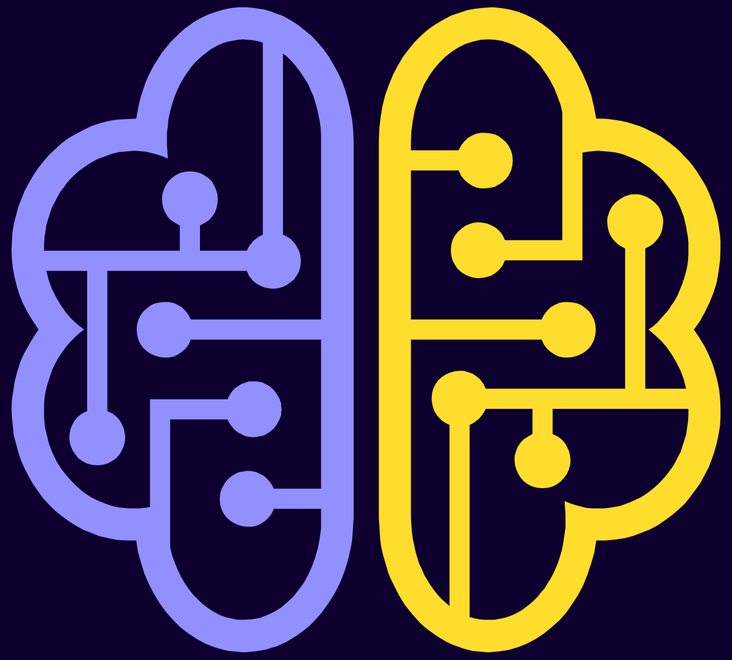


План доклада

1. Почему стоит писать автотесты
2. XCUITest
3. Unit тесты на стероидах
4. KIF
5. Сравнение подходов

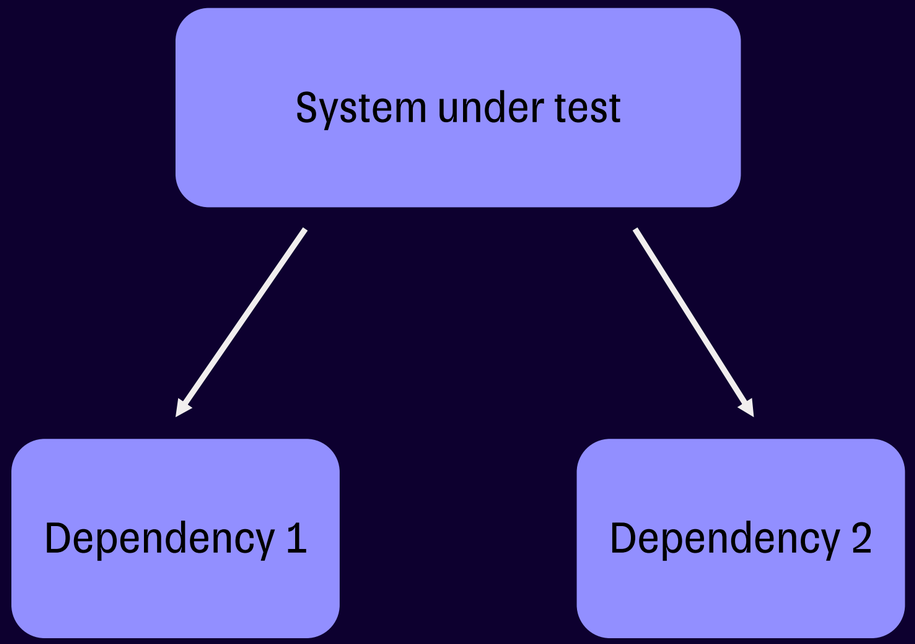
Unit тесты бывают разными

Лондонская школа

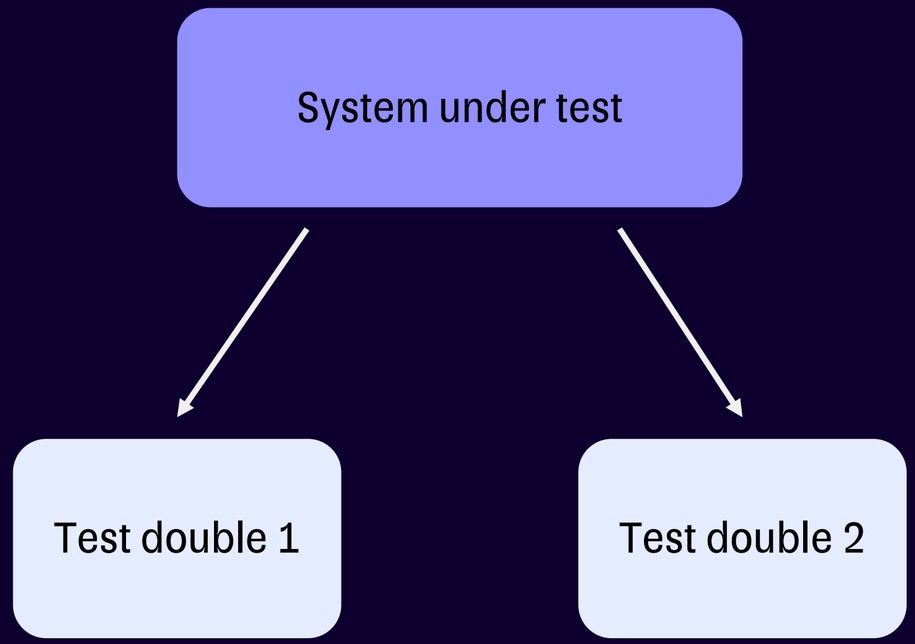


Классическая школа

Unit тесты лондонской школы

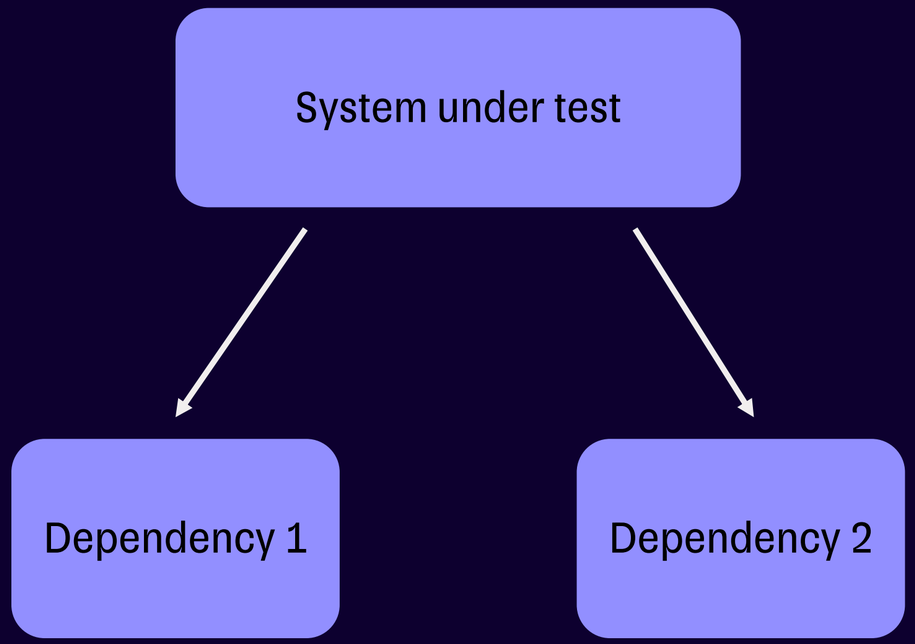


Продакшен код

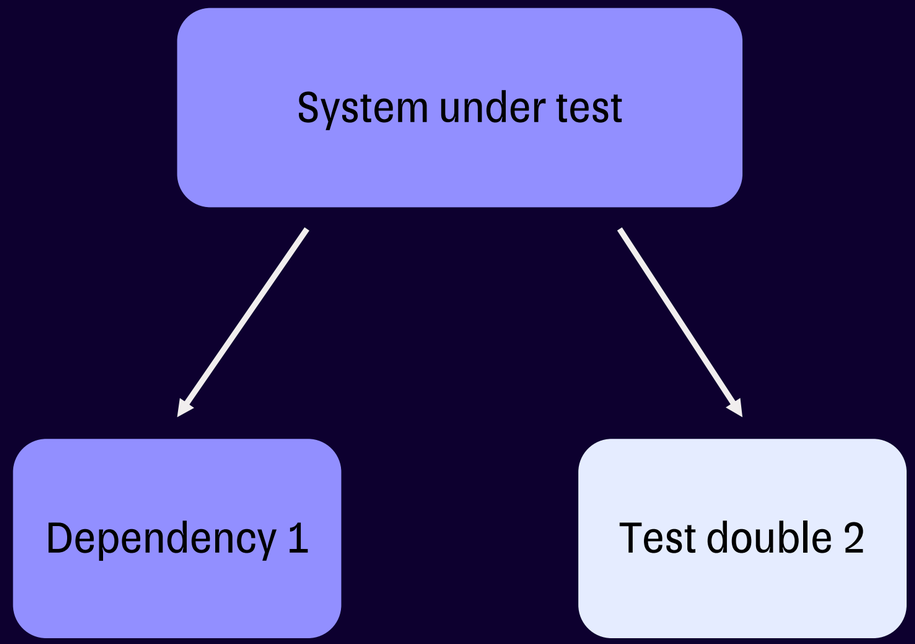


Тестовый код

Unit тесты классической школы



Продакшен код



Тестовый код

Тест-кейс

Предусловие

1. Неавторизованный пользователь

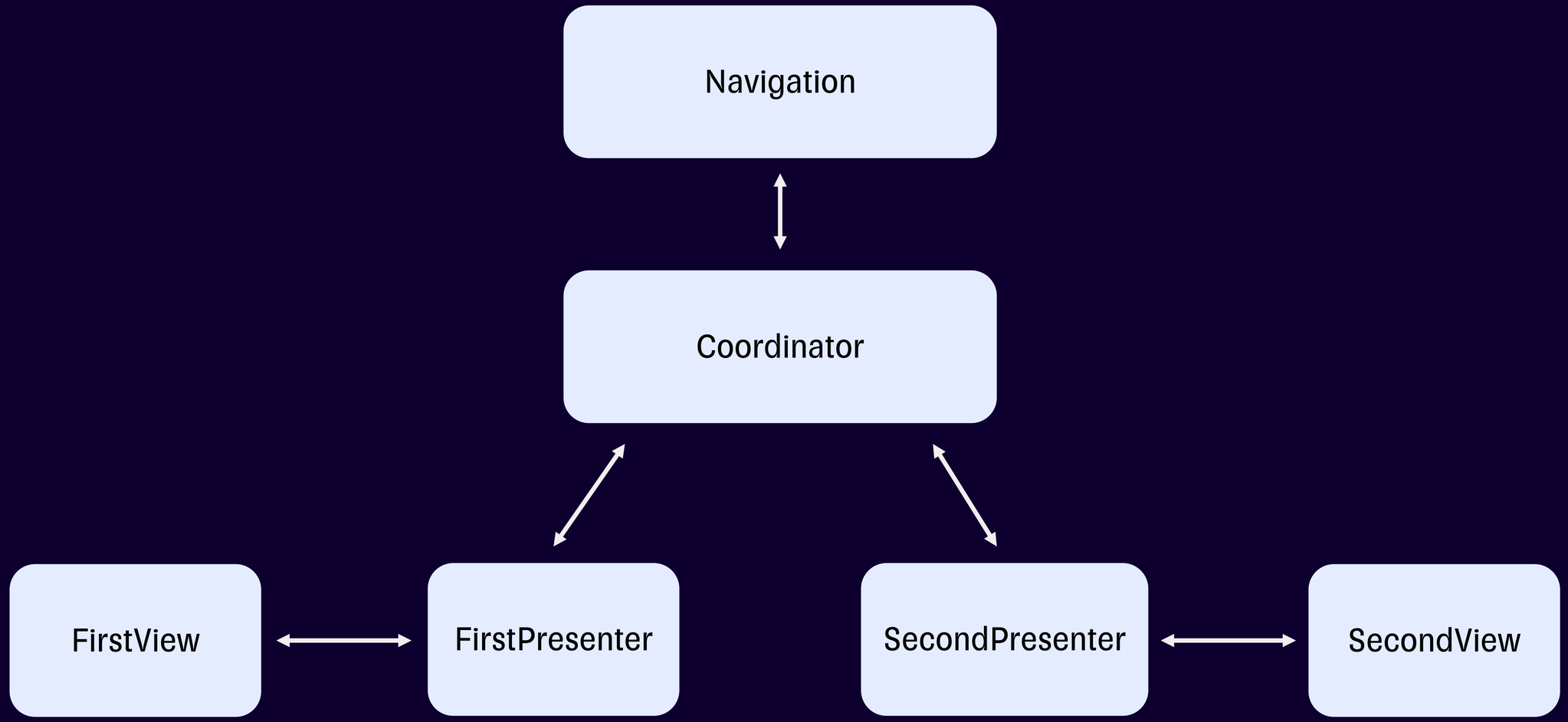
Сценарий

1. На экране логина ввести свой никнейм
2. Нажать кнопку ***Далее***

Ожидаемый результат

1. Открыт главный экран приложения
2. По центру экрана находится текст ***Hello world***

Сущности


Navigation
Coordinator
FirstView
FirstPresenter
SecondPresenter
SecondView

FirstPresenter

```
class FirstPresenter: IFirstPresenter {
    // ...
    private var nickname = ""

    func textFieldValueChanged(value: String) {
        nickname = value
    }

    func didTap() {
        guard !nickname.isEmpty else { return }
        output?.didProcess()
    }
}
```

FirstPresenter

```
class FirstPresenter: IFirstPresenter {
    // ...
    private var nickname = ""

    func textFieldValueChanged(value: String) {
        nickname = value
    }

    func didTap() {
        guard !nickname.isEmpty else { return }
        output?.didProcess()
    }
}
```

SecondPresenter

```
class SecondPresenter: ISecondPresenter {
    // ...

    func viewDidAppear() {
        view?.showText("Hello world")
    }
}
```

Coordinator

```
protocol INavigation: AnyObject {
    func pushViewController(_ viewController: UIViewController, animated: Bool)
}

class Coordinator {
    private let firstAssembly: IFirstAssembly
    private let secondAssembly: ISecondAssembly
    private let navigation: INavigation

    func start() {
        let view = firstAssembly.assemble(output: self)
        navigation.pushViewController(view, animated: true)
    }

    private func showSecond() {
        let view = secondAssembly.assemble(output: self)
        navigation.pushViewController(view, animated: true)
    }
}

extension Coordinator: IFirstOutput {
    func didProcess() {
        showSecond()
    }
}
```

Coordinator

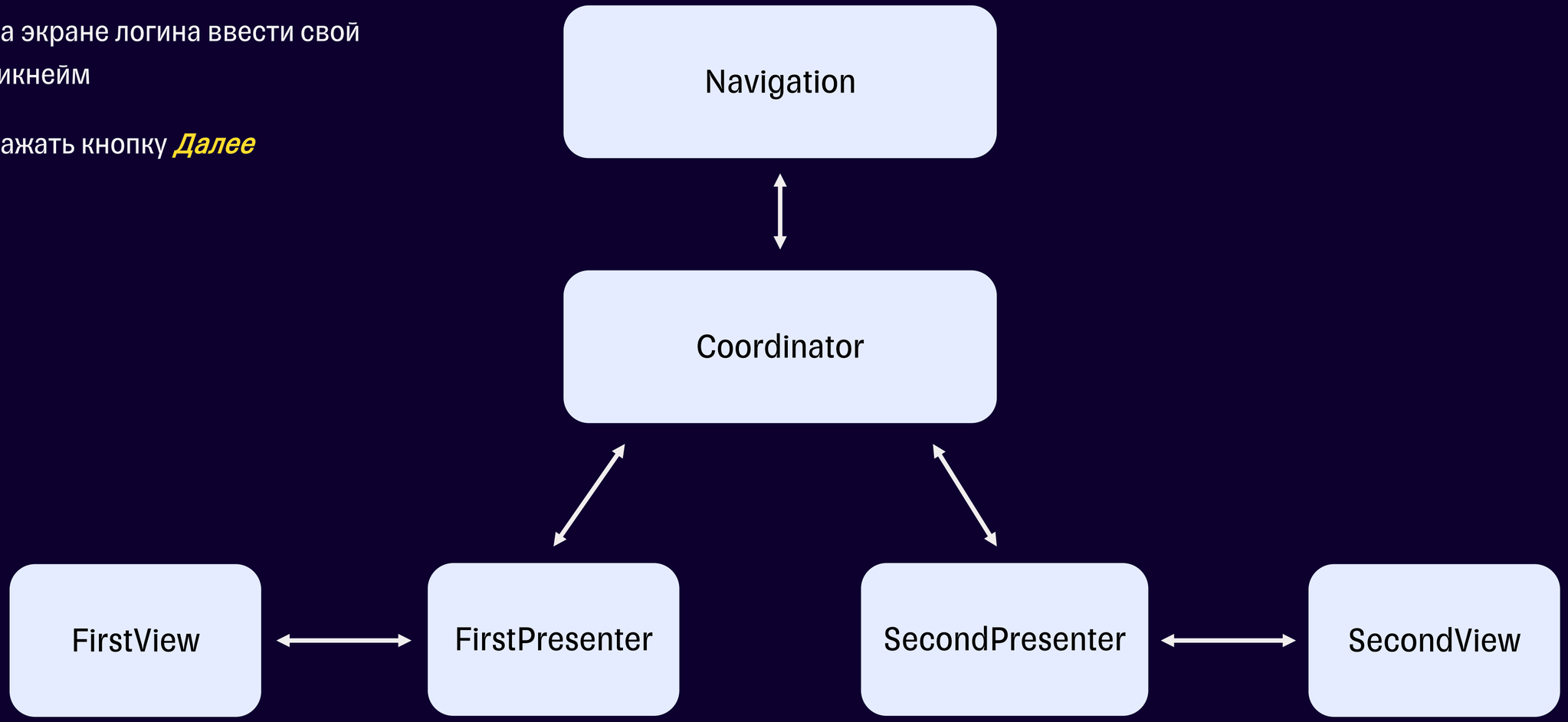
```
protocol INavigation: AnyObject {
    func pushViewController(_ viewController: UIViewController, animated: Bool)
}

class Coordinator {
    private let firstAssembly: IFirstAssembly
    private let secondAssembly: ISecondAssembly
    private let navigation: INavigation

    func start() {
        let view = firstAssembly.assemble(output: self)
        navigation.pushViewController(view, animated: true)
    }

    private func showSecond() {
        let view = secondAssembly.assemble(output: self)
        navigation.pushViewController(view, animated: true)
    }
}

extension Coordinator: IFirstOutput {
    func didProcess() {
        showSecond()
    }
}
```

Coordinator

```
protocol INavigation: AnyObject {
    func pushViewController(_ viewController: UIViewController, animated: Bool)
}

class Coordinator {
    private let firstAssembly: IFirstAssembly
    private let secondAssembly: ISecondAssembly
    private let navigation: INavigation

    func start() {
        let view = firstAssembly.assemble(output: self)
        navigation.pushViewController(view, animated: true)
    }

    private func showSecond() {
        let view = secondAssembly.assemble(output: self)
        navigation.pushViewController(view, animated: true)
    }
}

extension Coordinator: IFirstOutput {
    func didProcess() {
        showSecond()
    }
}
```

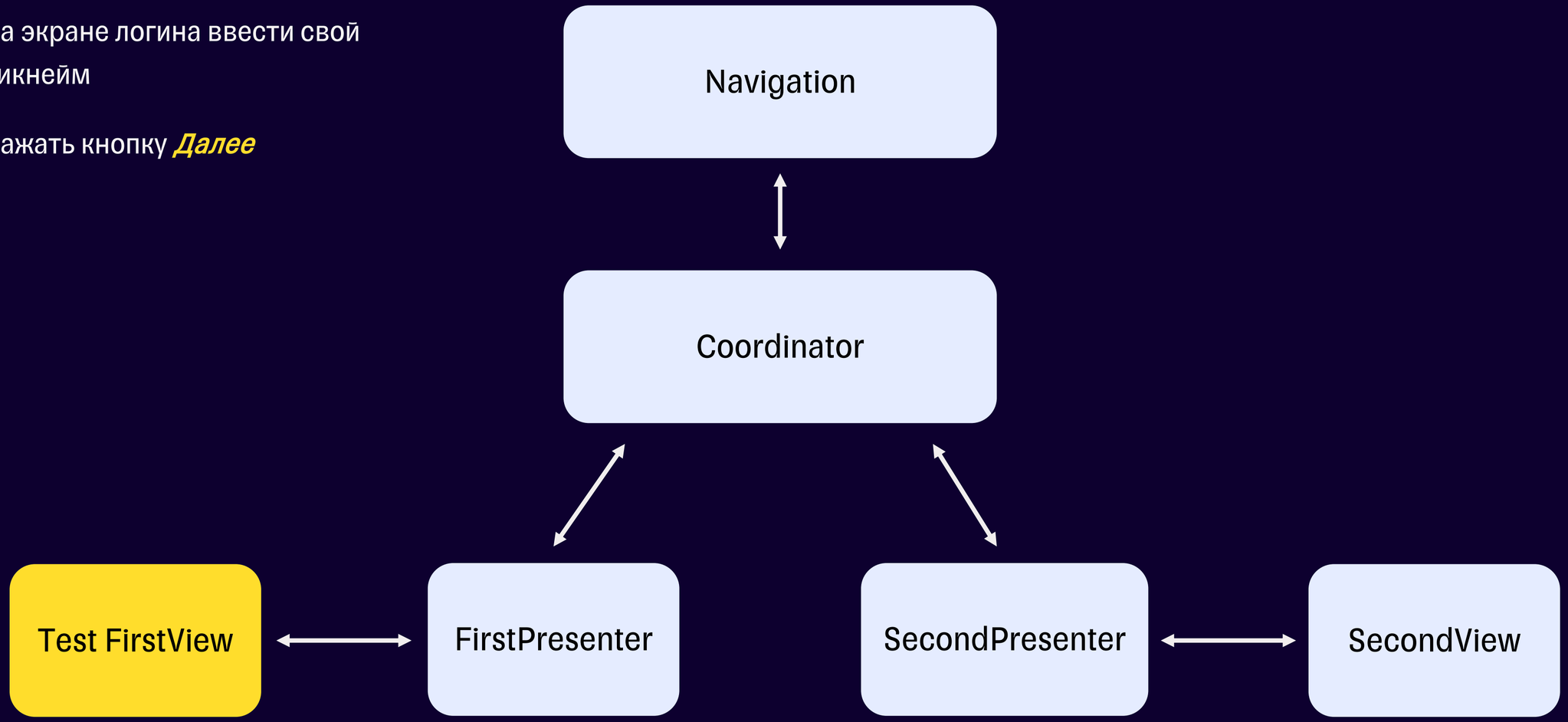
Сущности

1. На экране логина ввести свой никнейм
2. Нажать кнопку ***Далее***



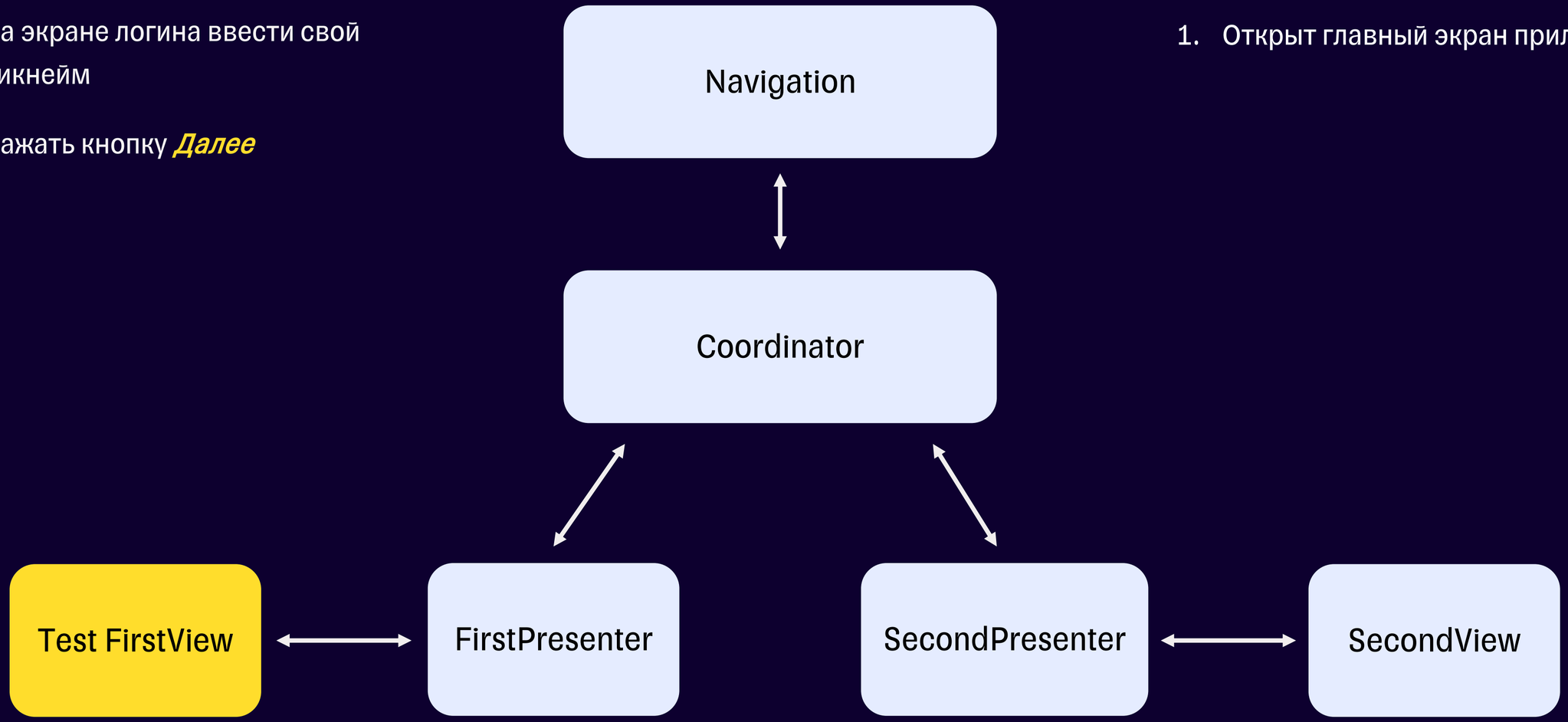
Сущности

1. На экране логина ввести свой никнейм
2. Нажать кнопку ***Далее***



Сущности

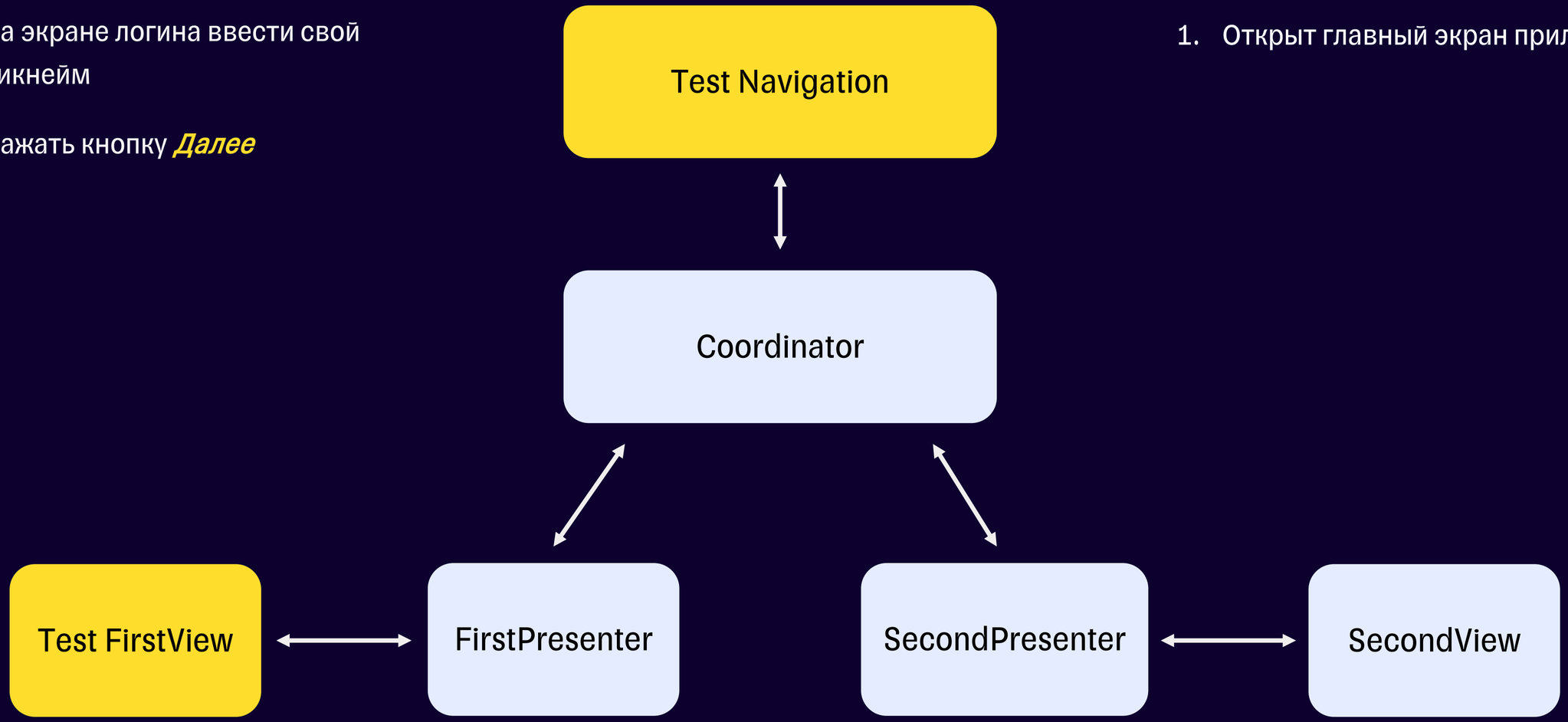
1. На экране логина ввести свой никнейм
2. Нажать кнопку ***Далее***



1. Открыт главный экран приложения

Сущности

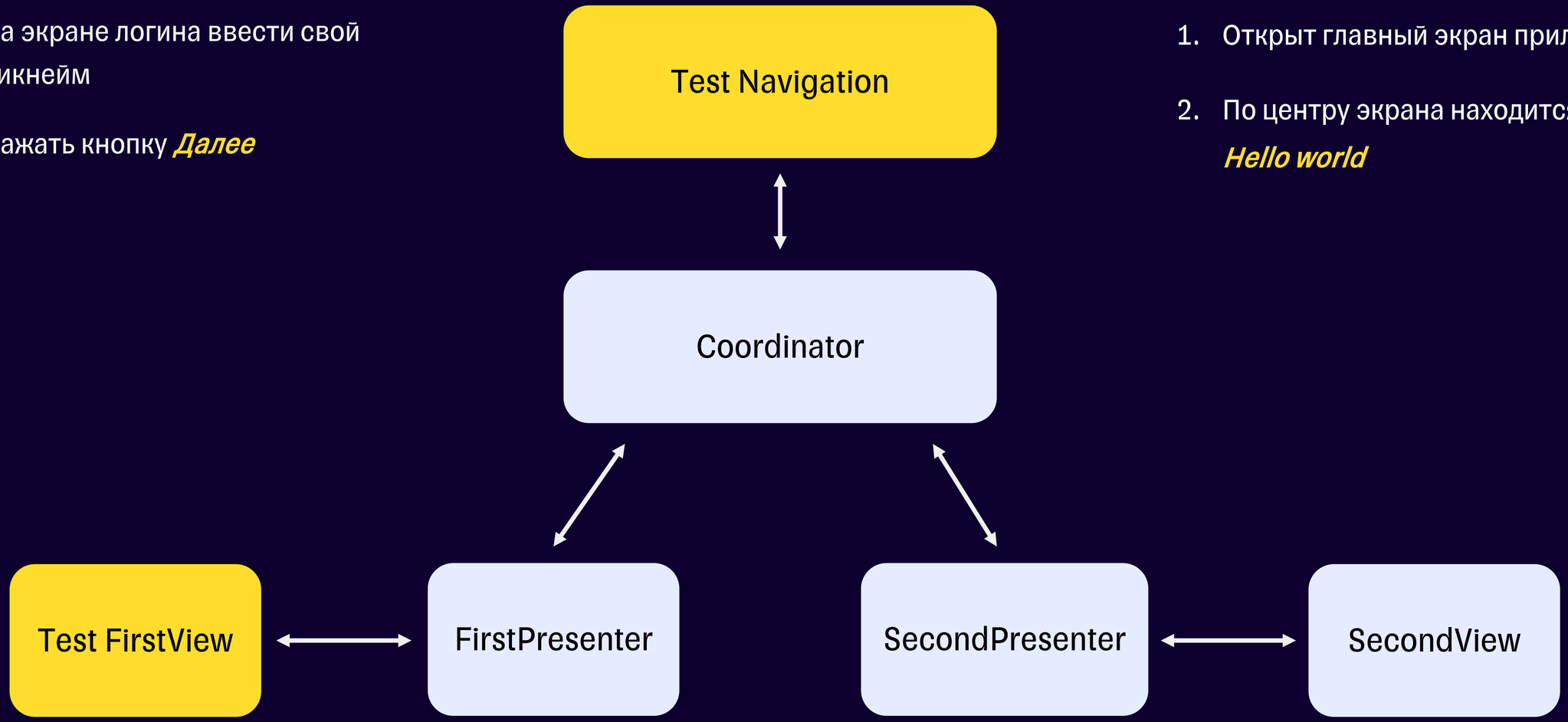
1. На экране логина ввести свой никнейм
2. Нажать кнопку ***Далее***



1. Открыт главный экран приложения

Сущности

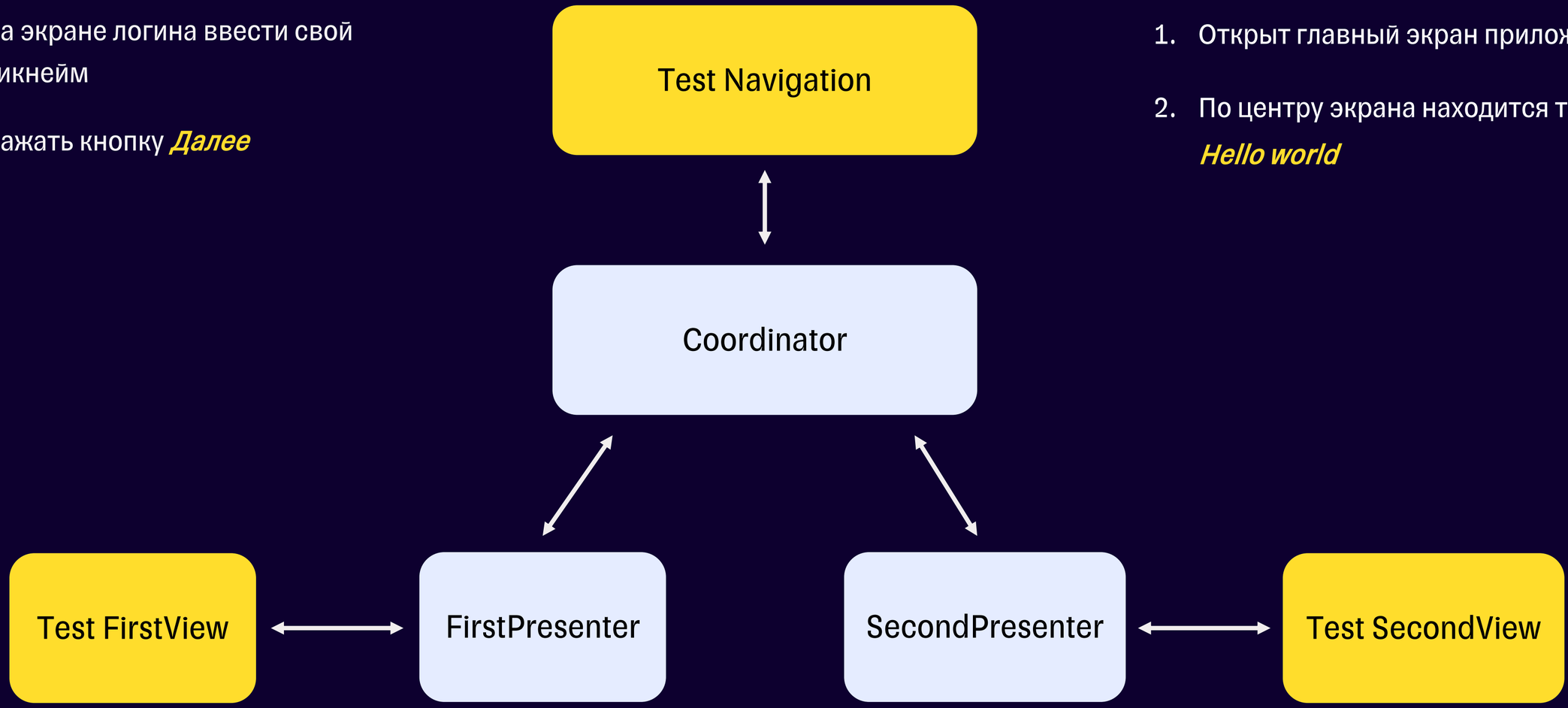
1. На экране логина ввести свой никнейм
2. Нажать кнопку ***Далее***



1. Открыт главный экран приложения
2. По центру экрана находится текст ***Hello world***

Сущности

1. На экране логина ввести свой никнейм
2. Нажать кнопку ***Далее***



1. Открыт главный экран приложения
2. По центру экрана находится текст ***Hello world***

Подготовка моков

```
class MockFirstController: UIViewController, IFirstView {
    let presenter: FirstPresenter

    init(presenter: FirstPresenter) {
        self.presenter = presenter
        super.init(nibName: nil, bundle: nil)
    }

    required init?(coder: NSCoder) {
        fatalError("init(coder:) has not been implemented")
    }
}
```

Подготовка моков

```
class MockFirstAssembly: IFirstAssembly {
    var presenter: FirstPresenter?
    var view: MockFirstController?

    func assemble(output: IFirstOutput) -> UIViewController {
        let presenter = FirstPresenter(output: output)
        let view = MockFirstController(presenter: presenter)
        presenter.view = view
        self.presenter = presenter
        self.view = view
        return view
    }
}
```

Подготовка моков

```
class MockNavigation: INavigation {
    var viewControllers: [UIViewController] = []

    func pushViewController(_ viewController: UIViewController, animated: Bool) {
        viewControllers.append(viewController)
    }
}
```

Автоматизация тест-кейса

```
func testFlow() throws {
    // given
    let firstAssembly = MockFirstAssembly()
    let secondAssembly = MockSecondAssembly()
    let navigation = MockNavigation()
    let coordinator = Coordinator(firstAssembly: firstAssembly, secondAssembly: secondAssembly, navigation: navigation

    // when
    coordinator.start()

    firstAssembly.presenter?.textFieldValueChanged(value: "Nickname")
    firstAssembly.presenter?.didTap()

    secondAssembly.presenter?.viewDidAppear()

    // then
    XCTAssertTrue(navigation.viewControllers.last is MockSecondController)
    let view = try XCTUnwrap(secondAssembly.view)
    XCTAssertEqual(view.text, "Hello world")
}
```

Автоматизация тест-кейса

```
func testFlow() throws {
    // given
    let firstAssembly = MockFirstAssembly()
    let secondAssembly = MockSecondAssembly()
    let navigation = MockNavigation()
    let coordinator = Coordinator(firstAssembly: firstAssembly, secondAssembly: secondAssembly, navigation: navigation)

    // when
    coordinator.start()

    firstAssembly.presenter?.textFieldValueChanged(value: "Nickname")
    firstAssembly.presenter?.didTap()

    secondAssembly.presenter?.viewDidAppear()

    // then
    XCTAssertTrue(navigation.viewControllers.last is MockSecondController)
    let view = try XCTUnwrap(secondAssembly.view)
    XCTAssertEqual(view.text, "Hello world")
}
```

Автоматизация тест-кейса

```
func testFlow() throws {
    // given
    let firstAssembly = MockFirstAssembly()
    let secondAssembly = MockSecondAssembly()
    let navigation = MockNavigation()
    let coordinator = Coordinator(firstAssembly: firstAssembly, secondAssembly: secondAssembly, navigation: navigation)

    // when
    coordinator.start()

    firstAssembly.presenter?.textFieldValueChanged(value: "Nickname")
    firstAssembly.presenter?.didTap()

    secondAssembly.presenter?.viewDidAppear()

    // then
    XCTAssertTrue(navigation.viewControllers.last is MockSecondController)
    let view = try XCTUnwrap(secondAssembly.view)
    XCTAssertEqual(view.text, "Hello world")
}
```

Автоматизация тест-кейса

```
func testFlow() throws {
    // given
    let firstAssembly = MockFirstAssembly()
    let secondAssembly = MockSecondAssembly()
    let navigation = MockNavigation()
    let coordinator = Coordinator(firstAssembly: firstAssembly, secondAssembly: secondAssembly, navigation: navigation)

    // when
    coordinator.start()

    firstAssembly.presenter?.textFieldValueChanged(value: "Nickname")
    firstAssembly.presenter?.didTap()

    secondAssembly.presenter?.viewDidAppear()

    // then
    XCTAssertTrue(navigation.viewControllers.last is MockSecondController)
    let view = try XCTUnwrap(secondAssembly.view)
    XCTAssertEqual(view.text, "Hello world")
}
```

А что делать с UI тестами?

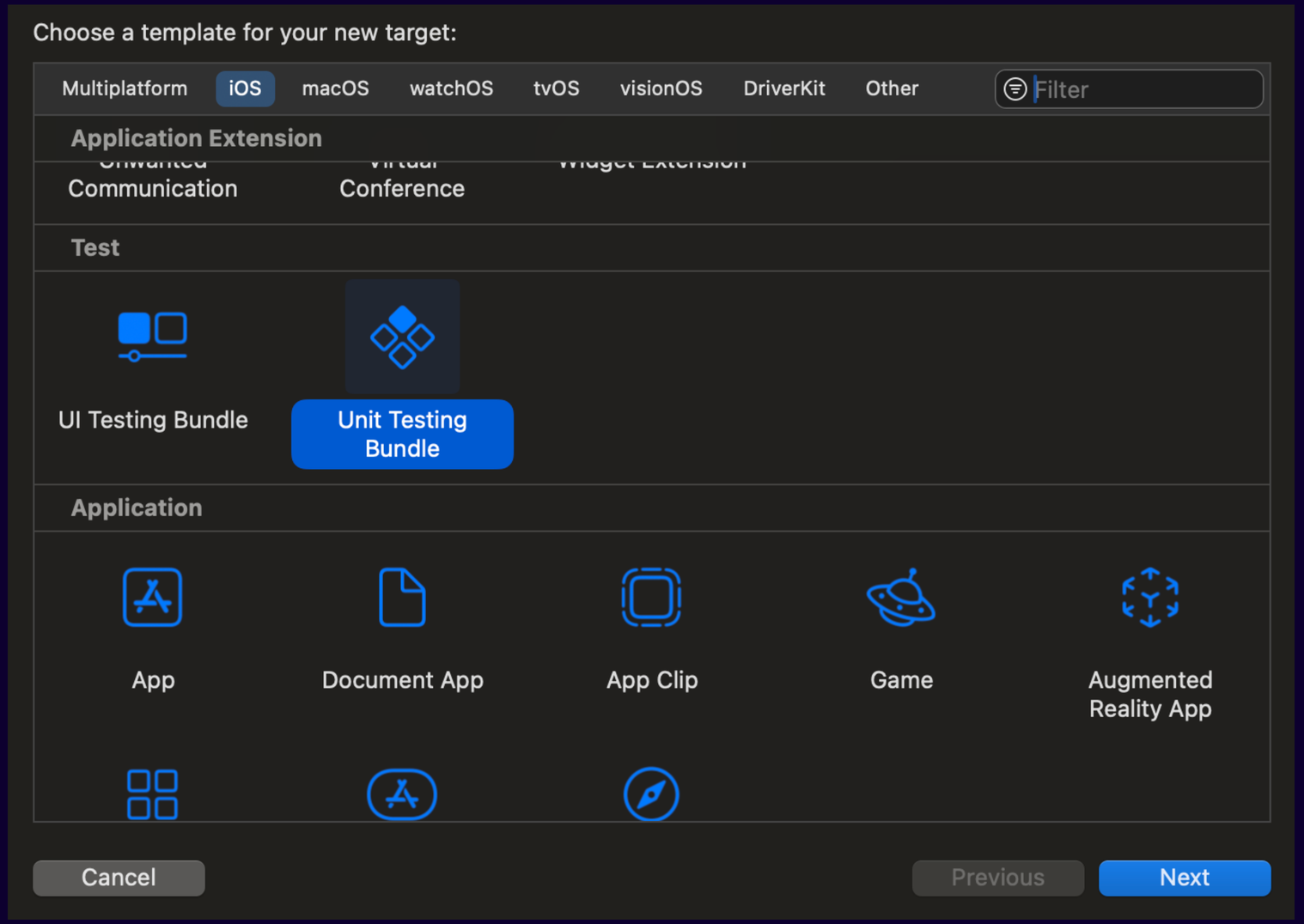
План доклада

1. Почему стоит писать автотесты
2. XCUITest
3. Unit тесты на стероидах
4. KIF
5. Сравнение подходов

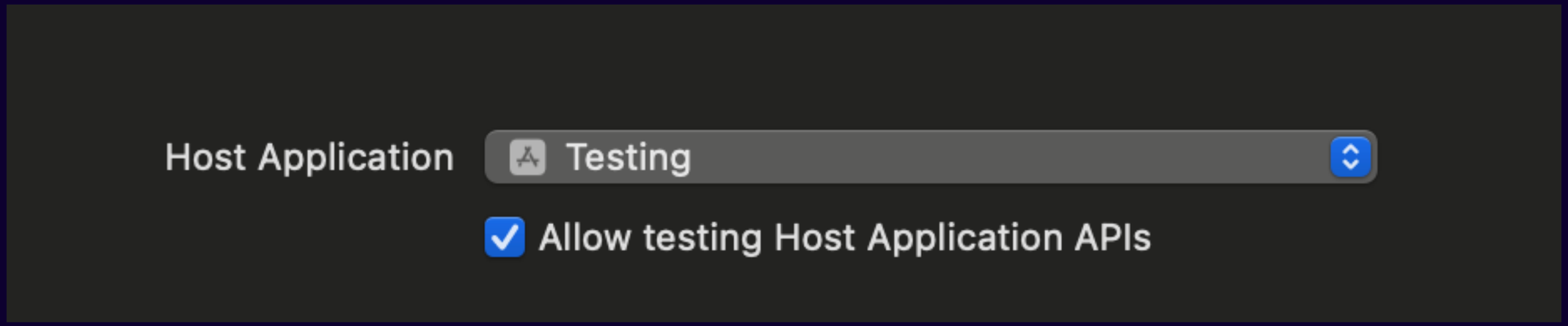
KIF

- Тестирование в белом ящике
- Выполняются в том же процессе
- Симулирует взаимодействие с UI элементами
- Имеет ограничения по сравнению с XCUI
- Позволяет писать быстрые UI тесты

Настройка в проекте



Настройка в проекте



Вспоминаем XCUITest

```
    func testExample() throws {
19      let app = XCUIApplication()
20      app.launch()
21
22      let element = app.textFields["loginField"]
23      element.tap()
24      element.typeText("Nickname")
25      app.buttons["nextButton"].tap()
26
27      XCTAssert(app.staticTexts["Hello world"].exists)   XCTAssertTrue failed
28  }
```

Напишем KIF тест

```
func testFlow() throws {
    // given
    let firstAssembly = FirstAssembly()
    let secondAssembly = SecondAssembly()
    let navigationController = UINavigationController()
    setRootVC(navigationController)
    let coordinator = Coordinator(
        firstAssembly: firstAssembly,
        secondAssembly: secondAssembly,
        navigation: navigationController
    )

    // when
    coordinator.start()
    tester().usingIdentifier("loginField").enterText("Nickname")
    tester().usingIdentifier("nextButton").tap()
    tester().waitForAnimationsToFinish()

    // then
    let label = try XCTUnwrap(tester().usingIdentifier("mainLabel").view as? UILabel)
    XCTAssertEqual(label.text, "Hello world")
}
```

Напишем KIF тест

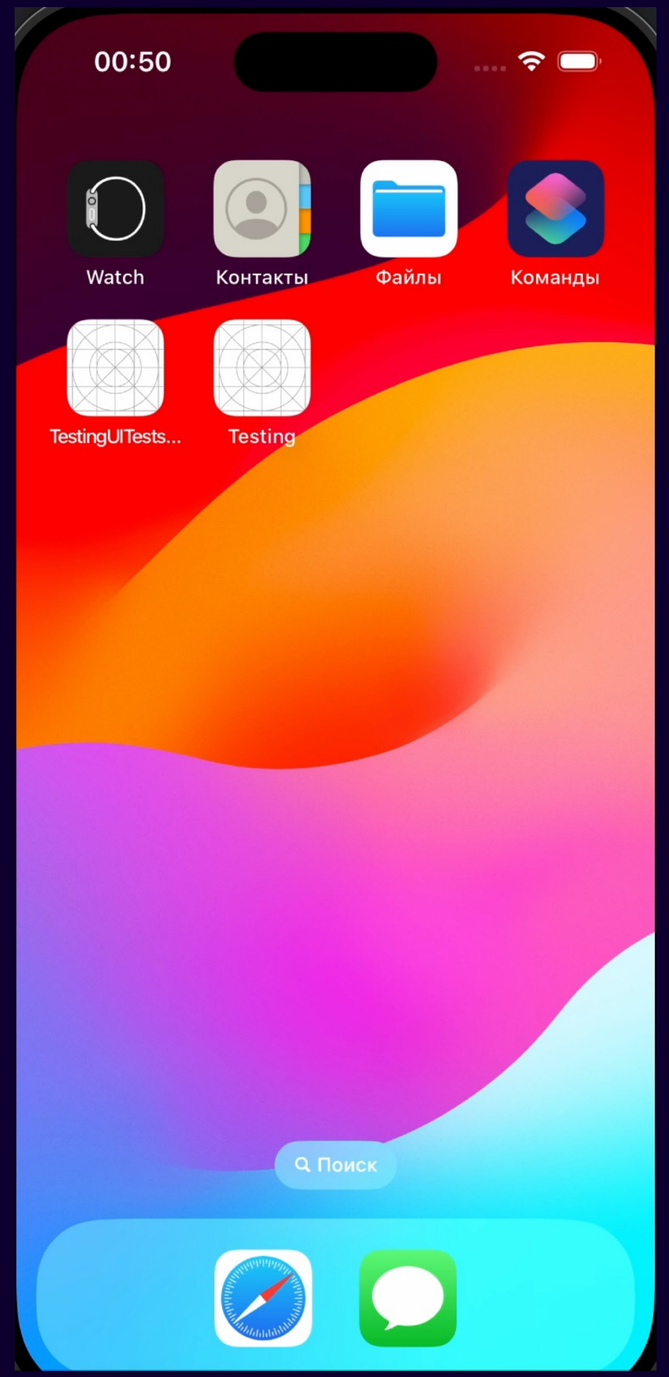
```
func testFlow() throws {
    // given
    let firstAssembly = FirstAssembly()
    let secondAssembly = SecondAssembly()
    let navigationController = UINavigationController()
    setRootVC(navigationController)
    let coordinator = Coordinator(
        firstAssembly: firstAssembly,
        secondAssembly: secondAssembly,
        navigation: navigationController
    )

    // when
    coordinator.start()
    tester().usingIdentifier("loginField").enterText("Nickname")
    tester().usingIdentifier("nextButton").tap()
    tester().waitForAnimationsToFinish()

    // then
    let label = try XCTUnwrap(tester().usingIdentifier("mainLabel").view as? UILabel)
    XCTAssertEqual(label.text, "Hello world")
}
```

Напишем KIF тест

```
func testFlow() throws {
    // given
    let firstAssembly = FirstAssembly()
    let secondAssembly = SecondAssembly()
    let navigationController = UINavigationController()
    setRootVC(navigationController)
    let coordinator = Coordinator(
        firstAssembly: firstAssembly,
        secondAssembly: secondAssembly,
        navigation: navigationController
    )

    // when
    coordinator.start()
    tester().usingIdentifier("loginField").enterText("Nickname")
    tester().usingIdentifier("nextButton").tap()
    tester().waitForAnimationsToFinish()

    // then
    let label = try XCTUnwrap(tester().usingIdentifier("mainLabel").view as? UILabel)
    XCTAssertEqual(label.text, "Hello world")
}
```

Напишем KIF тест

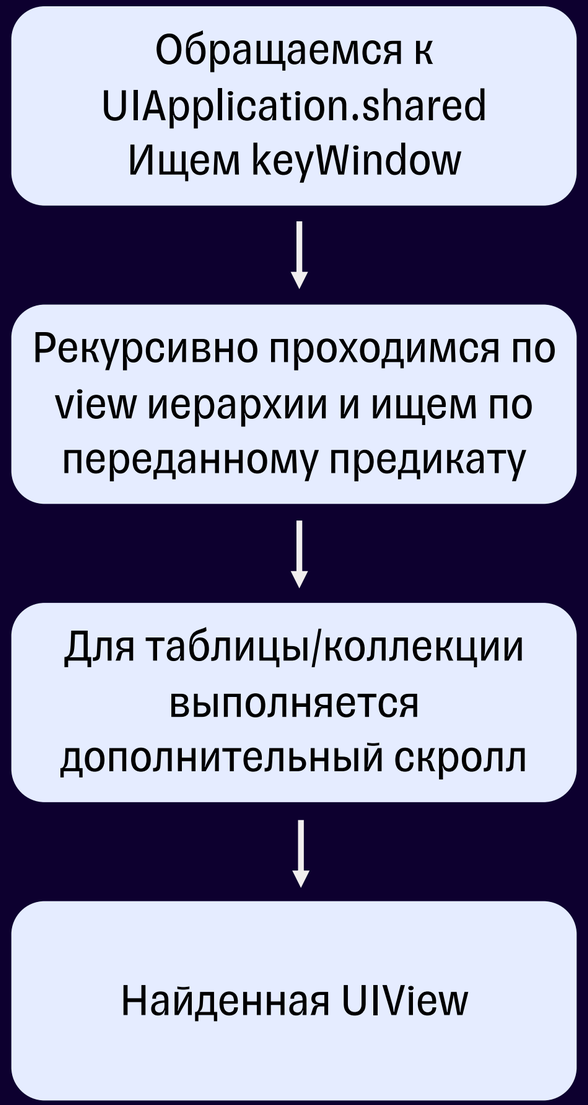
```
func testFlow() throws {
    // given
    let firstAssembly = FirstAssembly()
    let secondAssembly = SecondAssembly()
    let navigationController = UINavigationController()
    setRootVC(navigationController)
    let coordinator = Coordinator(
        firstAssembly: firstAssembly,
        secondAssembly: secondAssembly,
        navigation: navigationController
    )

    // when
    coordinator.start()
    tester().usingIdentifier("loginField").enterText("Nickname")
    tester().usingIdentifier("nextButton").tap()
    tester().waitForAnimationsToFinish()

    // then
    let label = try XCTUnwrap(tester().usingIdentifier("mainLabel").view as? UILabel)
    XCTAssertEqual(label.text, "Hello world")
}
```



Алгоритм работы

```
func testFlow() throws {
    // given
    let firstAssembly = FirstAssembly()
    let secondAssembly = SecondAssembly()
    let navigationController = UINavigationController()
    setRootVC(navigationController)
    let coordinator = Coordinator(
        firstAssembly: firstAssembly,
        secondAssembly: secondAssembly,
        navigation: navigationController
    )

    // when
    coordinator.start()
    tester().usingIdentifier("loginField").enterText("Nickname")
    tester().usingIdentifier("nextButton").tap()
    tester().waitForAnimationsToFinish()

    // then
    let label = try XCTUnwrap(tester().usingIdentifier("mainLabel").view as? UILabel)
    XCTAssertEqual(label.text, "Hello world")
}
```

Поиск


Обращаемся к UIApplication.shared Ищем keyWindow
Рекурсивно проходимся по view иерархии и ищем по переданному предикату
Для таблицы/коллекции выполняется дополнительный скролл
Найденная UIView

Нажатие


isUserInteractionEnabled
Рассчитываем CGPoint для нажатия (условно середину)
Проверяем не надо ли подскролить для нажатия в середину
Создаем UITouch и UIEvent
UIApplication.sendEvent
Эмулируем поведение firstRespinder, так как event отправлен программно

Многопоточность

Многопоточность

```
class SecondPresenter {
    weak var view: ISecondView?
    weak var output: ISecondOutput?

    init(output: ISecondOutput?) {
        self.output = output
    }
}

extension SecondPresenter: ISecondPresenter {
    func viewDidAppear() {
        DispatchQueue.main.asyncAfter(deadline: .now() + .seconds(3)) {
            self.view?.showText("Hello world")
        }
    }
}
```

Пример с моком очереди

```
protocol IDispatchQueue: AnyObject {
    var context: DispatchQueueContext { get }

    func sync(execute block: () -> Void)
    func sync(execute item: DispatchWorkItem)
    func async(execute item: DispatchWorkItem)
    func async(
        qos: DispatchQoS,
        flags: DispatchWorkItemFlags,
        execute work: @escaping @convention(block) () -> Void
    )
    func asyncAfter(
        deadline: DispatchTime,
        qos: DispatchQoS,
        flags: DispatchWorkItemFlags,
        execute work: @escaping @convention(block) () -> Void
    )

    // ...
}
```

Пример моком очереди

```
struct TestWorkItem {
    let id: Int
    let deadline: DispatchTime
    let work: () -> Void
}
```

Пример моком очереди

```
final class TestDispatchQueue {

    private(set) var currentTime: DispatchTime = .now()
    private var scheduled: [TestWorkItem] = []

    func run() {
        while let date = scheduled.first?.deadline {
            let distanceTime = currentTime.distance(to: date)
            advance(by: distanceTime)
        }
    }

    func advance(by timeInterval: DispatchTimeInterval) {
        // ...
    }
}
```

Пример моком очереди

```
final class TestDispatchQueue {

    private(set) var currentTime: DispatchTime = .now()
    private var scheduled: [TestWorkItem] = []

    func run() {
        while let date = scheduled.first?.deadline {
            let distanceTime = currentTime.distance(to: date)
            advance(by: distanceTime)
        }
    }

    func advance(by timeInterval: DispatchTimeInterval) {
        // ...
    }
}
```

Пример моком очереди

```
public func advance(by timeInterval: DispatchTimeInterval) {
    let finalDate = advancedCurrentTime(by: timeInterval)

    while currentTime <= finalDate {
        sortScheduledTasks()

        guard let nextDate = scheduled.first?.deadline,
              finalDate >= nextDate
        else {
            currentTime = finalDate
            return
        }

        currentTime = nextDate

        while let item = scheduled.first, item.deadline == nextDate {
            scheduled.removeFirst()
            item.work()
        }
    }
}
```

Пример моком очереди

```
public func asyncAfter(
    deadline: DispatchTime,
    qos: DispatchQoS,
    flags: DispatchWorkItemFlags,
    execute work: @escaping @convention(block) () -> Void
) {
    schedule(deadline: deadline, work: work)
}

public func schedule(deadline: DispatchTime, work: @escaping () -> Void) {
    let newItem = TestWorkItem(
        id: nextTaskId(),
        deadline: deadline,
        work: work
    )

    scheduled.append(newItem)
}
```

Пример моком очереди

```
class SecondPresenter {
    weak var view: ISecondView?
    weak var output: ISecondOutput?
    private let dispatchQueueFactory: IDispatchQueueFactory

    init(output: ISecondOutput?, dispatchQueueFactory: IDispatchQueueFactory) {
        self.output = output
        self.dispatchQueueFactory = dispatchQueueFactory
    }
}

extension SecondPresenter: ISecondPresenter {
    func viewDidAppear() {
        dispatchQueueFactory.main().asyncAfter(deadline: .now() + .seconds(3)) {
            self.view?.showText("Hello world")
        }
    }
}
```

Пример моком очереди

```
func testFlow() throws {
    // given
    let firstAssembly = FirstAssembly()

    let testQueue = TestDispatchQueue()
    let testQueueFactory = TestDispatchQueueFactory(testQueue: testQueue)
    let secondAssembly = KIFMockSecondAssembly(dispatchQueueFactory: testQueueFactory)

    let navigationController = UINavigationController()
    setRootVC(navigationController)
    let coordinator = Coordinator(
        firstAssembly: firstAssembly,
        secondAssembly: secondAssembly,
        navigation: navigationController
    )

    // when
    coordinator.start()
    tester().usingIdentifier("loginField").enterText("Nickname")
    tester().usingIdentifier("nextButton").tap()
    tester().waitForAnimationsToFinish()
    testQueue.run()

    // then
    let label = try XCTUnwrap(tester().usingIdentifier("mainLabel").view as? UILabel)
    XCTAssertEqual(label.text, "Hello world")
}
```

Пример моком очереди

```
func testFlow() throws {
    // given
    let firstAssembly = FirstAssembly()

    let testQueue = TestDispatchQueue()
    let testQueueFactory = TestDispatchQueueFactory(testQueue: testQueue)
    let secondAssembly = KIFMockSecondAssembly(dispatchQueueFactory: testQueueFactory)

    let navigationController = UINavigationController()
    setRootVC(navigationController)
    let coordinator = Coordinator(
        firstAssembly: firstAssembly,
        secondAssembly: secondAssembly,
        navigation: navigationController
    )

    // when
    coordinator.start()
    tester().usingIdentifier("loginField").enterText("Nickname")
    tester().usingIdentifier("nextButton").tap()
    tester().waitForAnimationsToFinish()
    testQueue.run()

    // then
    let label = try XCTUnwrap(tester().usingIdentifier("mainLabel").view as? UILabel)
    XCTAssertEqual(label.text, "Hello world")
}
```

PageObject

LoginPage

```
final class LoginPage {
    func enter(login: String) {
        tester().usingIdentifier("loginField").enterText("Nickname")

    }

    func tapNextButton() {
        tester().usingIdentifier("nextButton").tap()
    }
}
```

MainPage

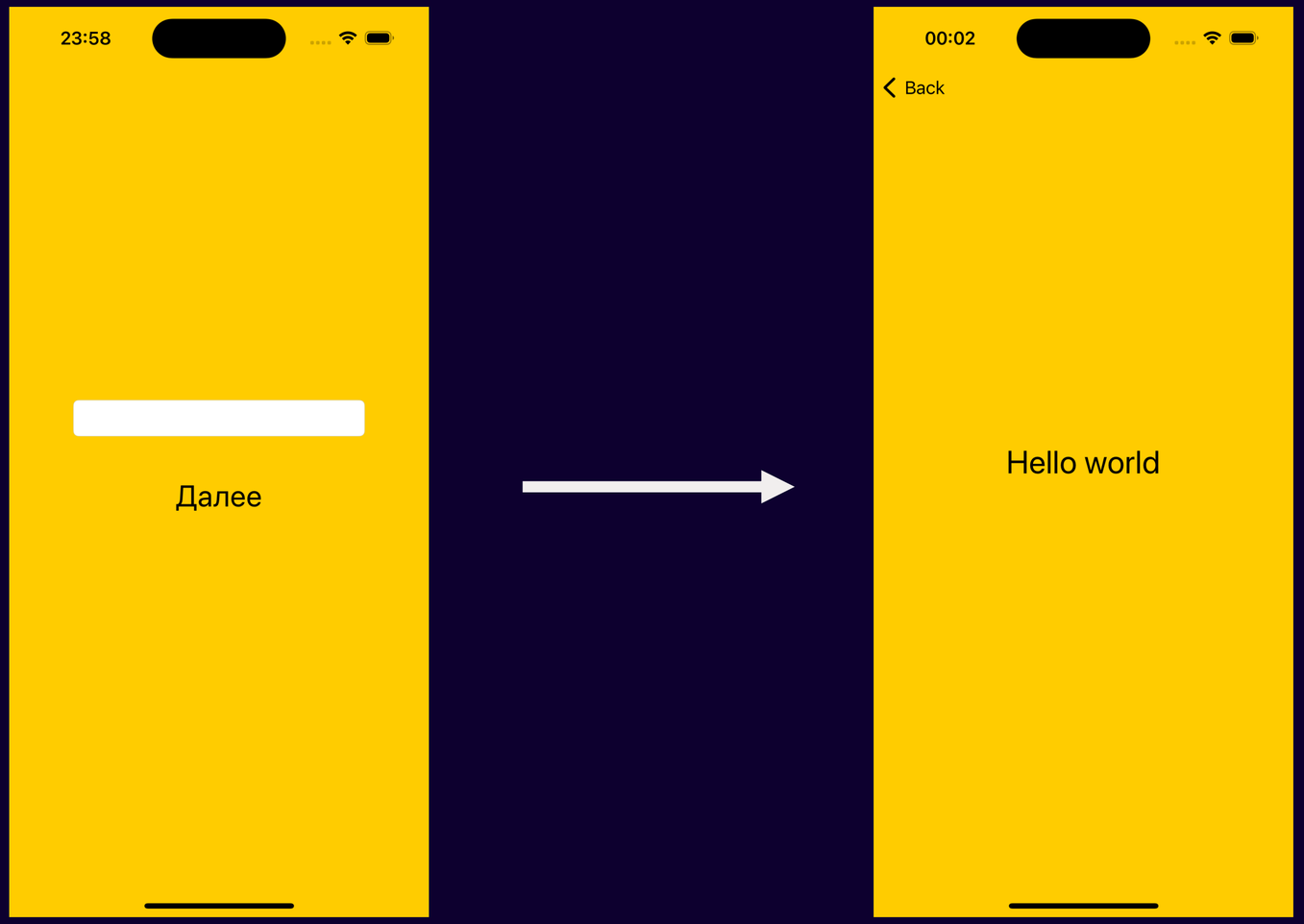
```
final class MainPage {
    func assertMainLabelText(_ text: String) throws {
        let label = try XCTUnwrap(tester().usingIdentifier("mainLabel").view as? UILabel)
        XCTAssertEqual(label.text, text)
    }
}
```

Как теперь выглядит тест

```
func testFlowWithPages() throws {
    // given
    let firstAssembly = FirstAssembly()

    let testQueue = TestDispatchQueue()
    let testQueueFactory = TestDispatchQueueFactory(testQueue: testQueue)
    let secondAssembly = KIFMockSecondAssembly(dispatchQueueFactory: testQueueFactory)

    let navigationController = UINavigationController()
    setRootVC(navigationController)
    let coordinator = Coordinator(
        firstAssembly: firstAssembly,
        secondAssembly: secondAssembly,
        navigation: navigationController
    )
    let loginPage = LoginPage()
    let mainPage = MainPage()

    // when
    coordinator.start()
    loginPage.enter(login: "Nickname")
    loginPage.tapNextButton()

    tester().waitForAnimationsToFinish()
    testQueue.run()

    // then
    try mainPage.assertMainLabelText("Hello world")
}
```

Изолированное тестирование UI

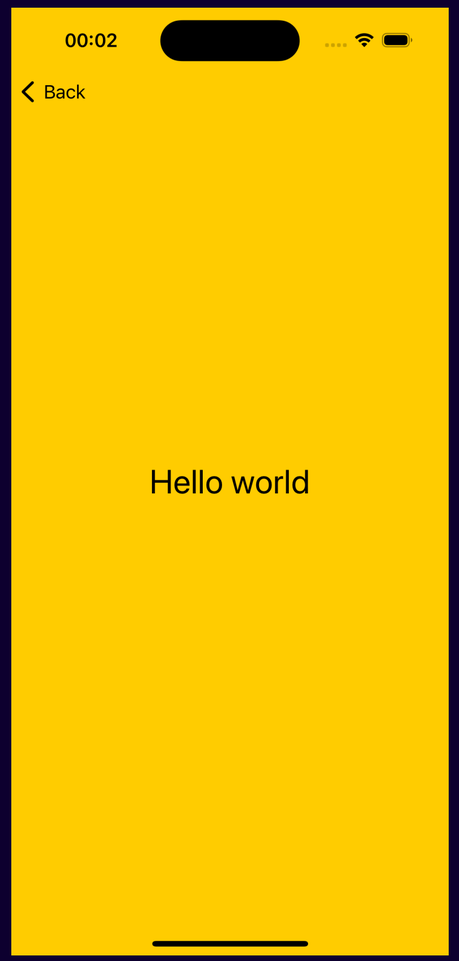
Изолированное тестирование UI



Экран логина

Главный экран

Изолированное тестирование UI



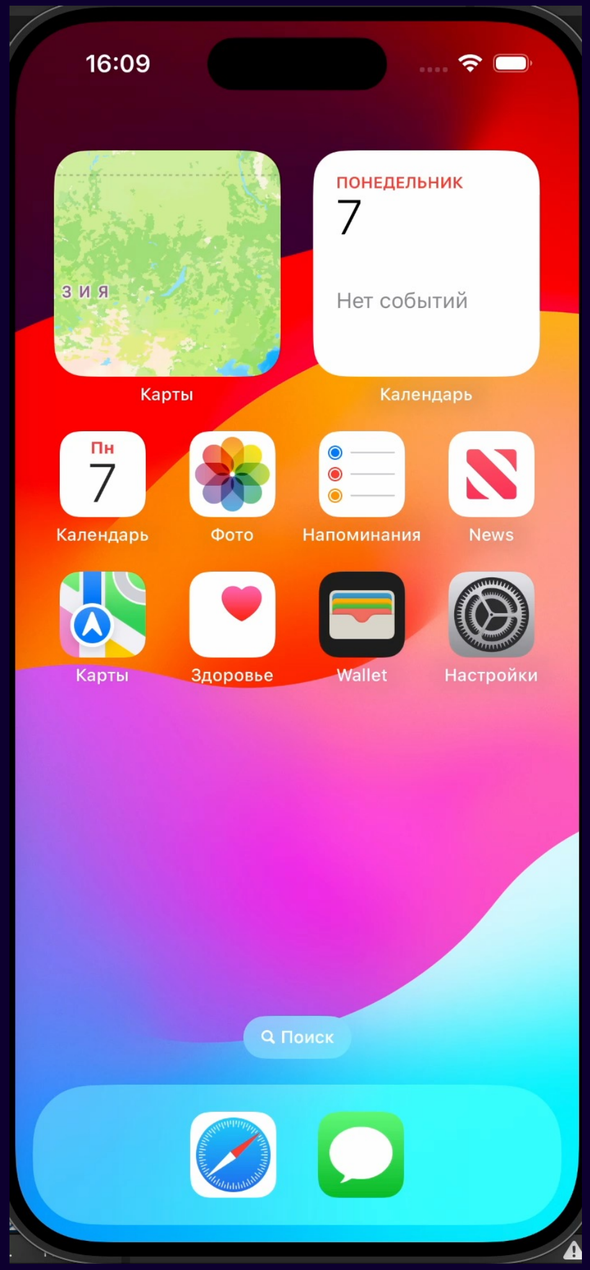
Главный экран

Изолированное тестирование UI

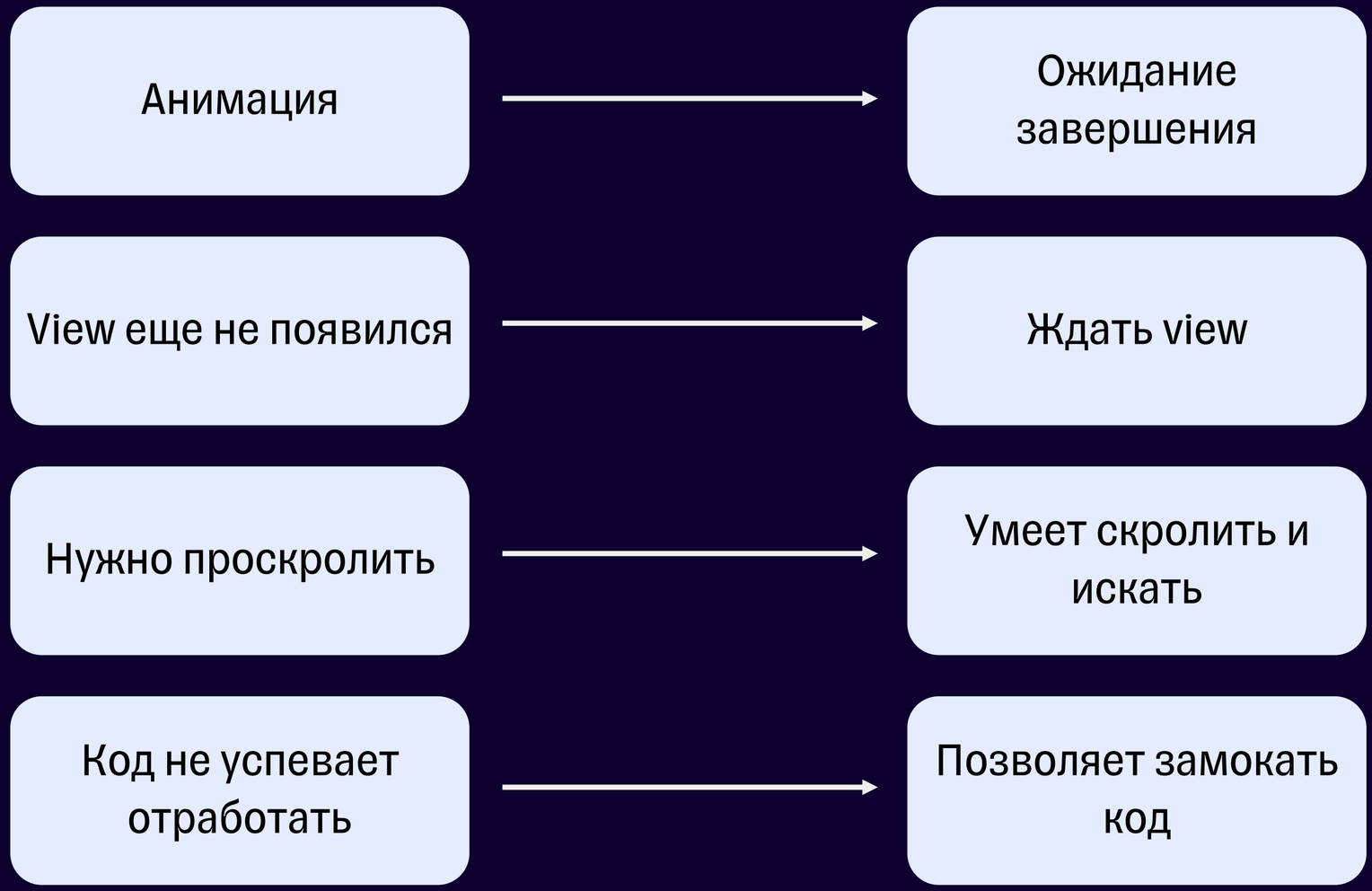
```
func testSecondPage() throws {
    // given
    let navigationController = UINavigationController()
    setRootVC(navigationController)
    let secondAssembly = SecondAssembly()
    let secondOutputMock = SecondOutputMock()
    let mainPage = MainPage()

    // when
    let controller = secondAssembly.assemble(output: secondOutputMock)
    navigationController.setViewControllers([controller], animated: false)

    // then
    try mainPage.assertMainLabelText("Hello world")
}
```

Изолированное тестирование UI

```
func testSecondPage() throws {
    // given
    let navigationController = UINavigationController()
    setRootVC(navigationController)
    let secondAssembly = SecondAssembly()
    let secondOutputMock = SecondOutputMock()
    let mainPage = MainPage()

    // when
    let controller = secondAssembly.assemble(output: secondOutputMock)
    navigationController.setViewControllers([controller], animated: false)

    // then
    try mainPage.assertMainLabelText("Hello world")
}
```



Почему KIF стабильный



KIF и SwiftUI

SwiftUI и KIF

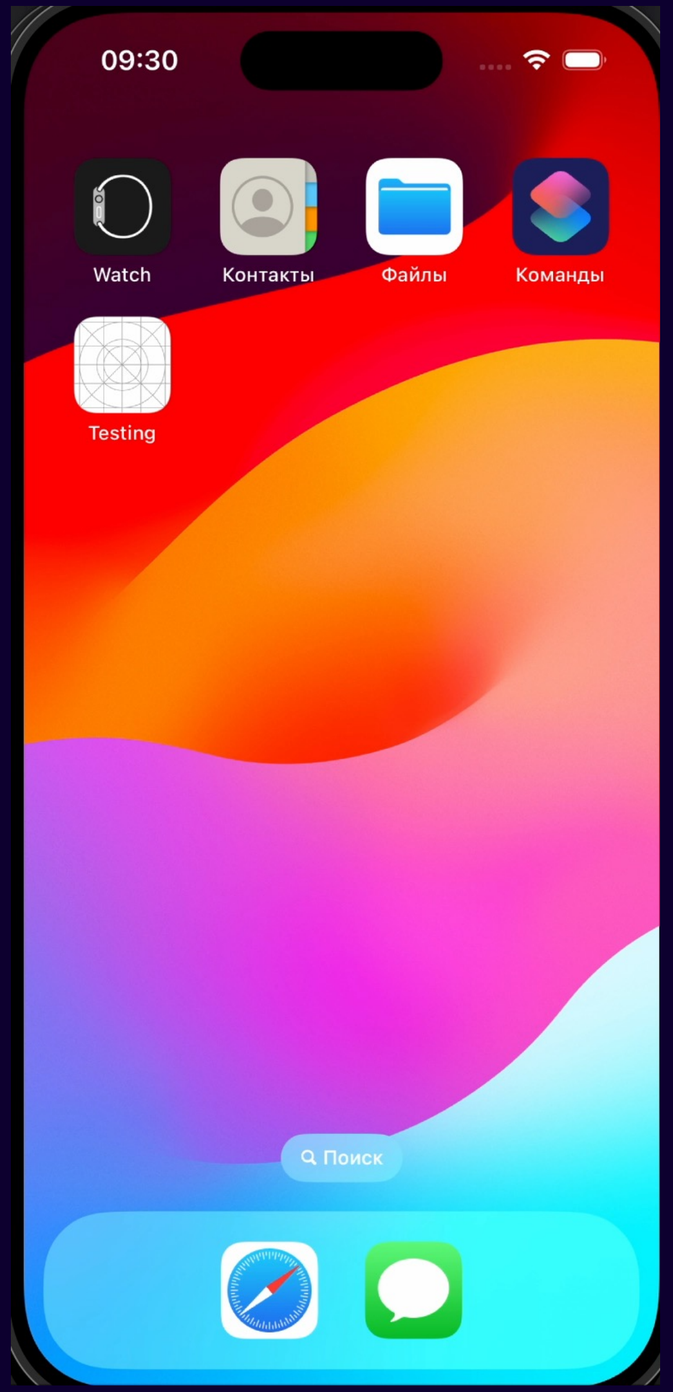
```
struct MainView: View {
    @ObservedObject var viewModel: MainViewModel
    @State var text = ""
    @State var isActive = false

    init(viewModel: MainViewModel) {
        self.viewModel = viewModel
    }

    var body: some View {
        NavigationView {
            VStack(alignment: .center, spacing: 16) {
                NavigationLink(
                    destination: SecondSwftUIView(),
                    isActive: $isActive
                ) { EmptyView() }
                TextField("", text: $text)
                    .padding(EdgeInsets(top: .zero, leading: 64, bottom: .zero, trailing: 64))
                    .background(Color.white)
                    .accessibilityIdentifier("loginField")

                Button("Далее") {
                    isActive.toggle()
                }.accessibilityIdentifier("nextButton")
            }
        }
    }
}
```

SwiftUI и KIF

```
struct SecondSwftUIView: View {
    var body: some View {
        Text("Hello world")
            .accessibilityIdentifier("mainLabel")
    }
}
```

SwiftUI и KIF

```
func testSwiftUI() {
    tester().usingIdentifier("loginField").enterText("Nickname")
    tester().usingIdentifier("nextButton").tap()
    tester().waitForAnimationsToFinish()

    tester().usingIdentifier("mainLabel").waitForView()
}
```



Со SwiftUI очень много нюансов...

Что учитывать, чтобы KIF работал лучше

Ускорить/отключить анимацию

Что учитывать, чтобы KIF работал лучше

Ускорить/отключить анимацию

Не искать View каждый раз от rootviewcontainer

Что учитывать, чтобы KIF работал лучше

Ускорить/отключить анимацию

Не искать View каждый раз от rootviewcontainer

Избегать поиска через scroll

Что учитывать, чтобы KIF работал лучше

Ускорить/отключить анимацию

Не искать View каждый раз от rootviewcontainer

Избегать поиска через scroll

Использовать тестовые диспатчеры

Что учитывать, чтобы KIF работал лучше

Ускорить/отключить анимацию

Не искать View каждый раз от rootviewcontainer

Избегать поиска через scroll

Использовать тестовые диспатчеры

Тестировать только часть флоу

Что учитывать, чтобы KIF работал лучше

Ускорить/отключить анимацию

Не искать View каждый раз от rootviewcontainer

Избегать поиска через scroll

Использовать тестовые диспатчеры

Тестировать только часть флоу

Убрать депрекейтед обращения

Минусы и ограничения

- Не умеет обращаться к элементам из других процессов
- Внешняя зависимость
- Нельзя обратиться к simctl из UI теста
- Использует приватное API
- Развитие остановилось
- Нельзя свернуть и заново открыть приложение
- Не совсем честное взаимодействие с UI
- Не все View поддерживает
- Есть ограничения со SwiftUI
- Следить за сбросом состояния между тестами

План доклада

1. Почему стоит писать автотесты
2. XCUITest
3. Unit тесты на стероидах
4. KIF
5. Сравнение подходов

Тест-кейс

Предусловие

1. Неавторизованный пользователь

Сценарий

1. На экране логина ввести свой никнейм
2. Нажать кнопку ***Далее***

Ожидаемый результат

1. Открыт главный экран приложения
2. По центру экрана находится текст ***Hello world***

Сравнение перформанса (10 повторений)

Unit x2363 раз быстрее

Итоги

1. Пишите автотесты
2. Используйте весь доступный арсенал
3. Unit тесты могут быть полезными даже для автомитизация тест-кейсов
4. UI тесты по методу белого ящика работают в несколько раз быстрее
5. Абстрагируйте многопоточку при помощи виртуального времени
6. Про XCUI не забываем

Т БАНК

Спасибо!