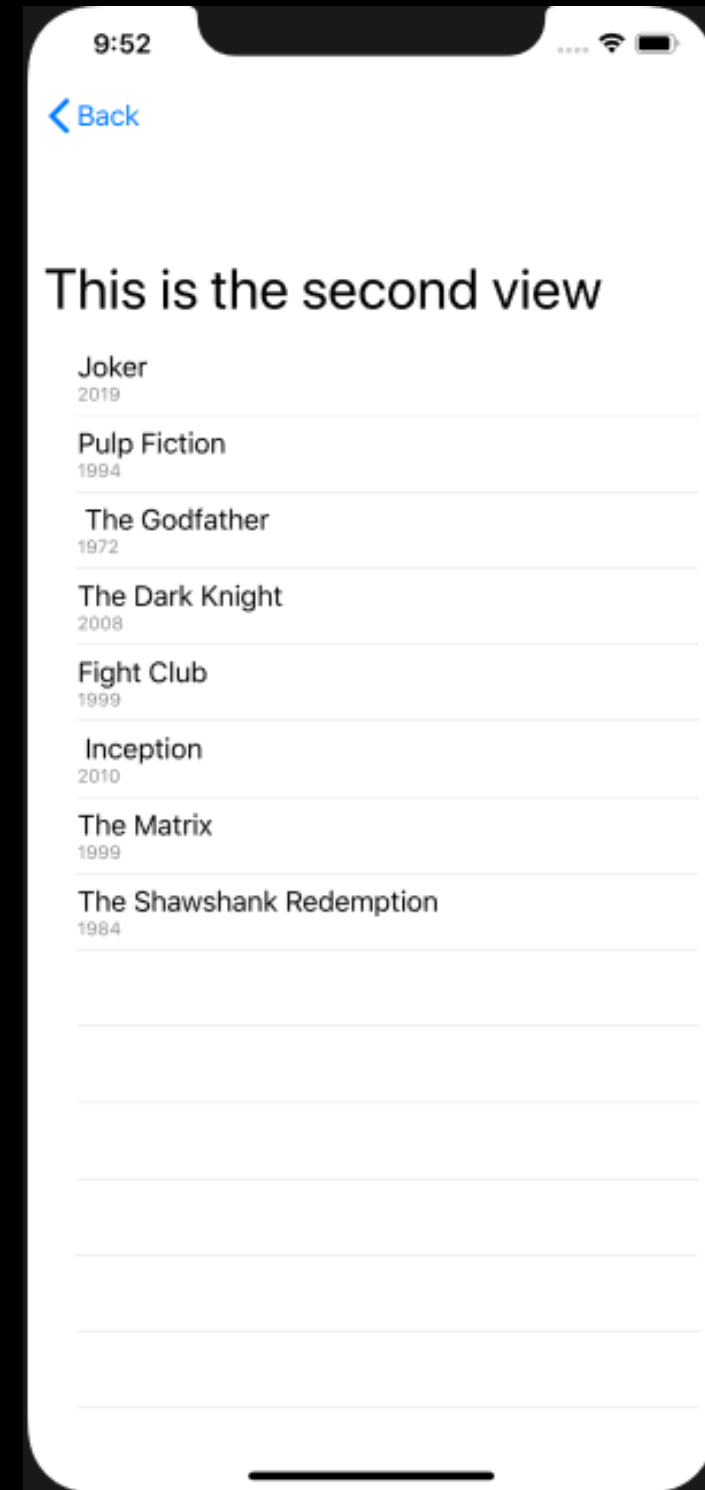
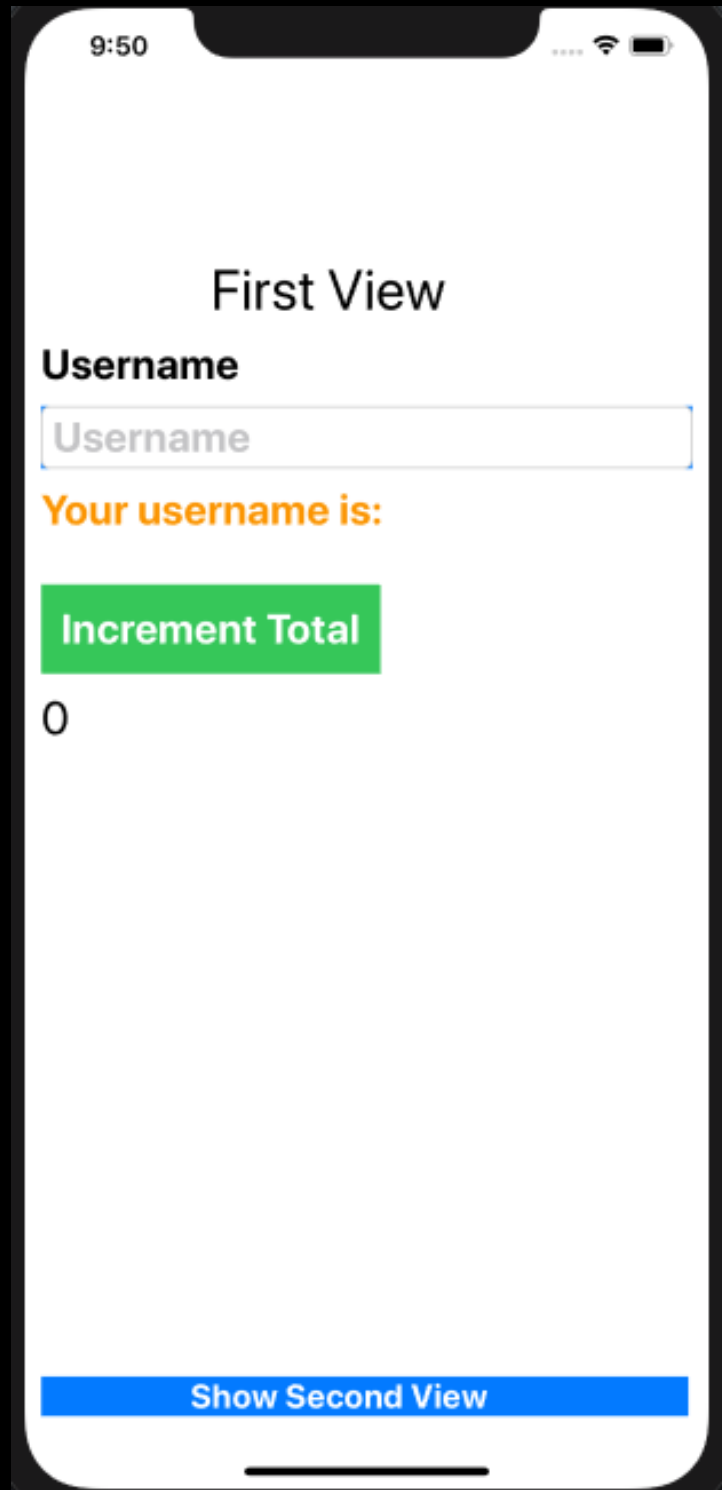




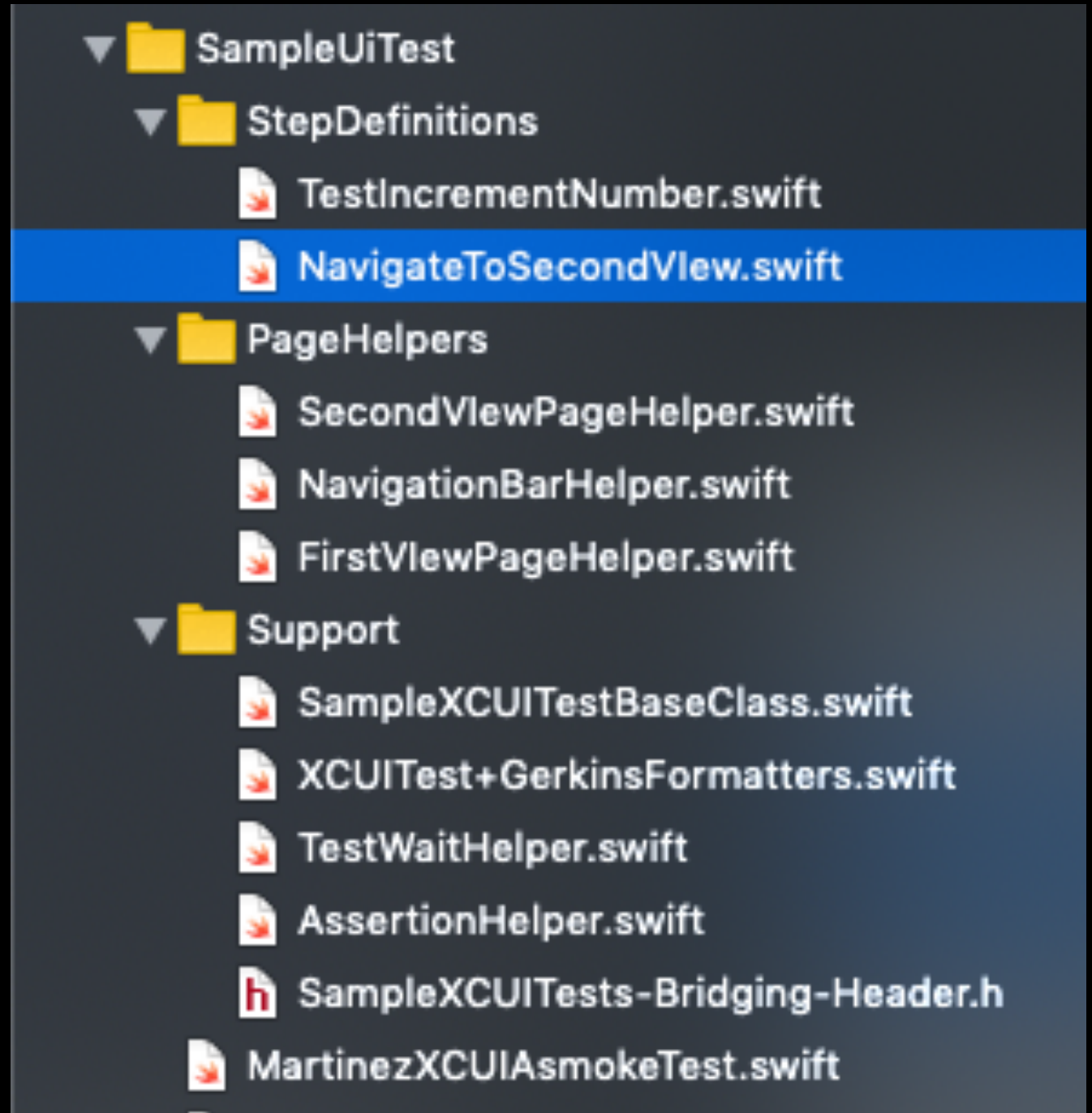
I put together some slides to show you how I architect the XCUITest framework for an iOS project.

*Eric Martínez*

# Using SwiftUI I create a simple app with two views.



# XUITest file structure

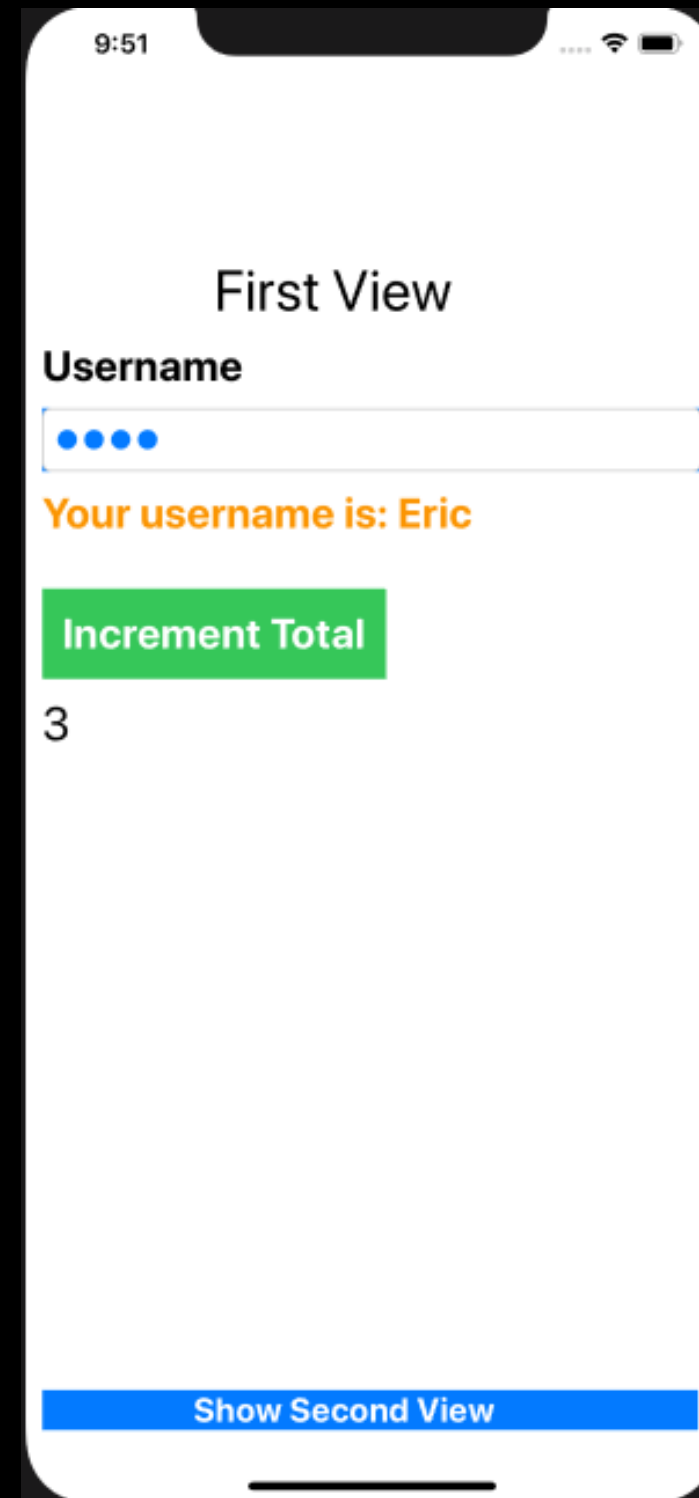


# Created a class for the different test scenarios

*I wrote a smoke test using Gherkin's syntax.*

```
11 class MartinezXCUIASmokeTest: MartinezBaseXCUITestHelper {
12
13     func testIncrementTotal() {
14
15         given("I launch the App I validate all first view UI element appear") {
16             incrementScreenElementValidation()
17         }
18
19         when("I enter the default username for testing") {
20             typeDefaultUsername()
21         }
22
23         then("I increment the number to three") {
24             incrementNumberTest()
25         }
26
27         and("I validate that the number has been increased") {
28             defaultDataPersist()
29         }
30     }
31 }
32
33 func testNavigateToSeconview() {
34
35     given("I'm on the First View") {
36         firstViewValidation()
37     }
38
39     when("I navigate to the second view") {
40         navigateToSecondView()
41     }
42
43     then("I validate that the list of movies appear") {
44         validateMoviesInTableView()
45     }
46
47     and("I navigate back to the first view") {
48         navigateBackToFirstView()
49     }
50 }
51 }
```

# First scenario



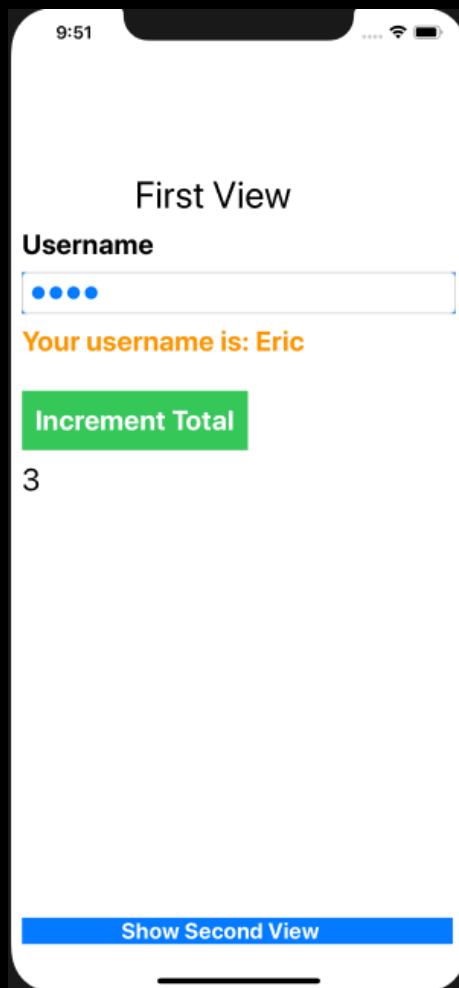
# First view step definitions

*In this file you will nest the necessary functions, and methods from different pages to complete the desired user journey.*

```
5 // Created by Eric Martinez on 1/17/20.
6 // Copyright © 2020 emobile. All rights reserved.
7 //
8
9 import Foundation
10 import XCTest
11
12 extension XCTestCase {
13     func typeDefaultUsername() {
14         application.tapfirstViewTextField(.userName)
15         application.typeText(FirstViewPageHelper.defaultNameEntry.rawValue)
16         application.buttons["Return"].tap() // Soft keyboard event needs its own helper
17         assertTrueStaticTexts(text: FirstViewPageHelper.defaultNameEntryResult.rawValue)
18     }
19
20     func incrementScreenElementValidation() {
21         assertTrueStaticTexts(text: FirstViewPageHelper.defaultNumber.rawValue)
22         assertTrueButtons(name: FirstViewPageHelper.incrementTotal.rawValue)
23     }
24
25     func incrementNumberTest() {
26         application.tapfirstViewButton(.incrementTotal)
27         application.tapfirstViewButton(.incrementTotal)
28         application.tapfirstViewButton(.incrementTotal)
29         waiting(for: application.staticTexts["3"])
30     }
31
32     func defaultDataPersist() {
33         application.tapfirstViewButton(.showSecondView)
34         self.waitForElementToAppear(NavigationBarHelper.navBarBackButton.navBarButton(in: application.self!))
35         application.tapNavBarButton(.navBarBackButton)
36         assertTrueStaticTexts(text: FirstViewPageHelper.defaultNameEntryResult.rawValue)
37         waiting(for: application.staticTexts["3"])
38     }
39 }
```

# First view test page helper

When there are changes in the view this format makes it easy to update



```
10
11 enum FirstViewPageHelper: String {
12     case viewTitle = "                First View"
13     case showSecondView = "                Show Second View"
14     case incrementTotal = "Increment Total"
15     case defaultNumber = "0"
16     case userName = "Username"
17     case defaultNameEntry = "Eric"
18     case defaultNameEntryResult = "Your username is: Eric"
19
20     func firstViewStaticTexts(in application: XCUIApplication) -> XCUIElement? {
21         return application.staticTexts[self.rawValue]
22     }
23
24     func firstViewButton(in application: XCUIApplication) -> XCUIElement? {
25         return application.buttons[self.rawValue]
26     }
27
28     func firstViewTextField(in application: XCUIApplication) -> XCUIElement? {
29         return application.secureTextFields[self.rawValue]
30     }
31 }
32
33 extension XCUIApplication {
34     func tapfirstViewStaticTexts(_ firstViewStaticTexts: FirstViewPageHelper) {
35         guard let element = firstViewStaticTexts.firstViewStaticTexts(in: self) else {
36             XCTFail("failed to tap static text in Sign In screen")
37             return
38         }
39         element.tap()
40     }
41
42     func tapfirstViewButton(_ firstViewButton: FirstViewPageHelper) {
43         guard let element = firstViewButton.firstViewButton(in: self) else {
44             XCTFail("failed to tap button in Sign In screen")
45             return
46         }
47         element.tap()
48     }
49
50     func tapfirstViewTextField(_ firstViewTextField: FirstViewPageHelper) {
51         guard let element = firstViewTextField.firstViewTextField(in: self) else {
52             XCTFail("failed to tap text field in Sign In screen")
53             return
54         }
55         element.tap()
56     }
57 }
58
```



# Navigation bar Helper

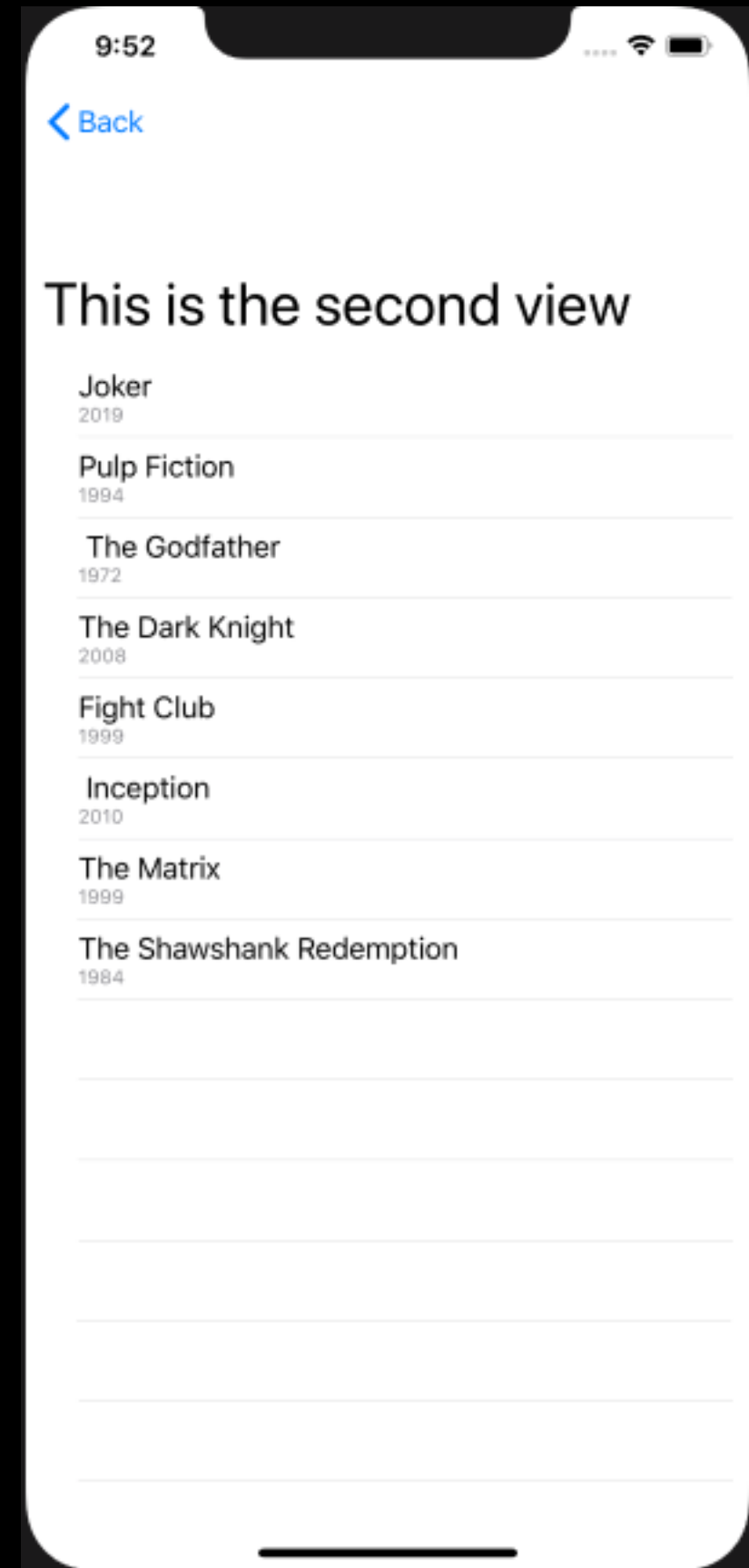


*The navBar can appear on all pages with the same buttons so I feel is best to give it its own page helper and use it across all test.*

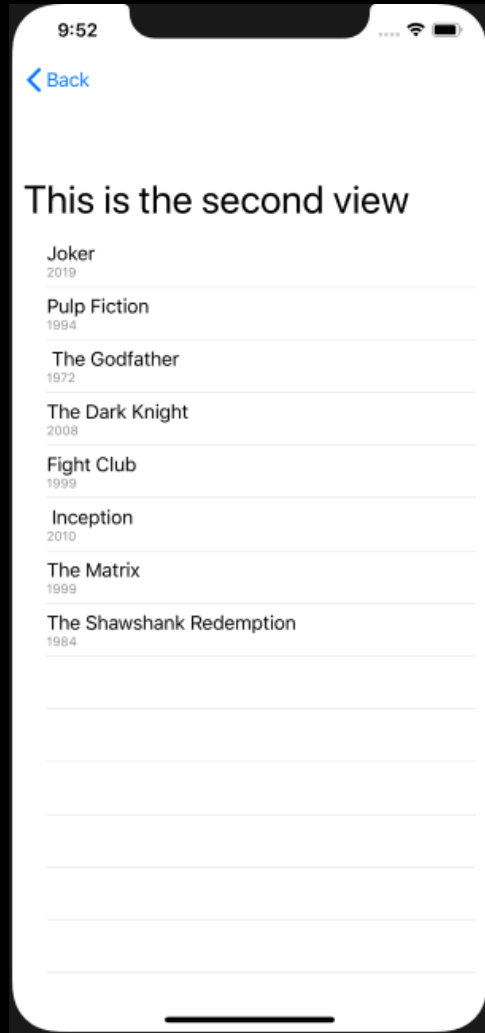
```
1 //
2 //  NavigationBarHelper.swift
3 //  SampleUITest
4 //
5 //  Created by Eric Martinez on 1/18/20.
6 //  Copyright © 2020 emobile. All rights reserved.
7 //
8
9 import Foundation
10 import XCTest
11
12 enum NavigationBarHelper: String {
13
14     case navBarBackButton = "Back"
15
16     func navBarButton(in application: XCUIApplication) -> XCUIElement? {
17         return application.buttons[self.rawValue]
18     }
19 }
20
21 extension XCUIApplication {
22
23     func tapNavBarButton(_ navBarButton: NavigationBarHelper) {
24         guard let element = navBarButton.navBarButton(in: self) else {
25             XCTFail("failed to tap Navigation bar button")
26             return
27         }
28         element.tap()
29     }
30 }
31
```



# Navigate to second view test



## Second view page helper



*You can write loops to iterate over Ui elements on the view or other functions related the page*

```
11
12 enum SecondViewPageHelper: String {
13     case secondViewTitle = "This is the second view"
14     case propertyJoker = "Joker"
15     case propertyJokerRelease = "2019"
16
17     func secondViewStaticText(in application: XCUIApplication) -> XCUIElement? {
18         return application.staticTexts[self.rawValue]
19     }
20
21     func secondViewTableStaticText(in application: XCUIApplication) -> XCUIElement? {
22         return application.tables.staticTexts[self.rawValue]
23     }
24 }
25
26 extension XCUIApplication {
27
28     func tapSecondViewStaticText(_ secondViewStaticText: SecondViewPageHelper) {
29         guard let element = secondViewStaticText.secondViewStaticText(in: self) else {
30             XCTFail("failed to tap Static text")
31             return
32         }
33         element.tap()
34     }
35
36     func findSecondViewTableStaticText(_ secondViewTableStaticText: SecondViewPageHelper) {
37         guard let element = secondViewTableStaticText.secondViewTableStaticText(in: self) else {
38             XCTFail("failed to find Table Static text")
39             return
40         }
41         element.tap()
42     }
43 }
44
45 func assertTrueTablesStaticText(_ movieInTable: String) {
46     XCTAssertTrue(application.tables.staticTexts[movieInTable].exists)
47 }
48
49 func movieListInTable() {
50     let moviesList = ["Joker", "Pulp Fiction", " The Godfather ", "The Dark Knight ",
51                     "Fight Club", " Inception", "The Matrix ", "The Shawshank Redemption "]
52     print(moviesList)
53     for movie in moviesList {
54         assertTrueTablesStaticText(movie)
55     }
56 }
57
```

# Second view step definitions

```
4 //
5 // Created by Eric Martinez on 1/18/20.
6 // Copyright © 2020 emobile. All rights reserved.
7 //
8
9 import Foundation
10 import XCTest
11
12 extension XCTestCase {
13
14     func firstViewValidation() {
15         self.waitForElementToAppear(FirstViewPageHelper.showSecondView.firstViewButton(in: application.self!))
16     }
17
18     func navigateToSecondView() {
19         application.tapfirstViewButton(.showSecondView)
20         self.waitForElementToAppear(NavigationBarHelper.navBarBackButton.navBarButton(in: application.self!))
21         assertTrueStaticTexts(text: SecondViewPageHelper.secondViewTitle.rawValue)
22     }
23
24     func validateMoviesInTableView() {
25         movieListInTable()
26         application.findSecondViewTableStaticText(.propertyJoker)
27         application.findSecondViewTableStaticText(.propertyJokerRelease)
28     }
29
30     func navigateBackToFirstView() {
31         application.tapNavBarButton(.navBarBackButton)
32         self.waitForElementToAppear(FirstViewPageHelper.showSecondView.firstViewButton(in: application.self!))
33     }
34 }
35
```

# Makefile commands

make test-ui runs all the test in the Ui test framework














make ui-test-report creates a test report and opens it on a browser

```
94  test-ui:
95      rm -rf TestResults
96      $(XCODEBUILD) test -scheme "SampleUiTest" \
97      -sdk iphonesimulator \
98      -destination '${SIMULATOR_UITEST}' \
99      -resultBundlePath TestResults
100
101  ui-test-report:
102      xcrun simctl shutdown all
103      xhtmlreport -r TestResults
104      echo "Your Test Report test report is launching on our default browser"
105      open index.html
106
```

In CI system we can generate xcpretty reports with colorful graphs for displaying on dashboard


# HTML Report

As you can see in this image the gherkins syntax prints out nicely

XCTestHTMLReport	
	Tests Logs
Devices	All (2) Passed (2) Failed (0)
<b>iPhone 11 Pro Max</b> iOS 13.3  Model: iPhone 11 Pro Max Identifier: 8F8AE9A6-BF57-4...	Status   Tests
	Selected tests - 2 tests (35s)
	SampleUITest.xctest - 2 tests (35s)
	MartinezXCUIAsmokeTest - 2 tests (35s)
	✓ ▾  testIncrementTotal() (18s)
	▶ Start Test at 2020-01-20 22:13:13.750 (0s) 
	▶ Set Up (5s) 
	▶ Given: I launch the App I validate all first view UI element appear (0s) 
	▶ When: I enter the default username for testing (4s) 
	▶ Then: I increment the number to three (3s) 
	▶ And: I validate that the number has been increased (5s) 
	Tear Down (0s)
	✓ ▾  testNavigateToSeconview() (16s)
	▶ Start Test at 2020-01-20 22:13:32.381 (0s) 
	▶ Set Up (5s) 
	▶ Given: I'm on the First View (1s)
	▶ When: I navigate to the second view (2s) 
	▶ Then: I validate that the list of movies appear (4s) 
	▶ Then: I navigate back to the first view (2s) 
	Tear Down (0s)

# HTML report

During the run Xcode captures a screenshot for each step

 XCTestHTMLReport

Tests

Logs

Devices

iPhone 11 Pro Max

IOS 13.3

Model: iPhone 11 Pro Max

Identifier: 8F8AE9A6-BF57-4...

All (2) Passed (2) Failed (0)

Status Tests

Selected tests - 2 tests (35s)

SampleUITest.xctest - 2 tests (35s)

MartinezXCUIAsmokeTest - 2 tests (35s)

testIncrementTotal() (18s)

▶ Start Test at 2020-01-20 22:13:13.750 (0s)

▶ Set Up (5s)

▶ Given: I launch the App I validate all first view UI element appear (0s)

▶ When: I enter the default username for testing (4s)

▶ Then: I increment the number to three (3s)

▼ And: I validate that the number has been increased (5s)

▼ Tap " Show Second View " Button (0s)

▼ Wait for emobile.UI-automation-Sample to idle (0s)

Screenshot

Find the " Show Second View " Button (0s)

Check for interrupting elements affecting " Show Second View " Button (0s)

▶ Synthesize event (0s)

▶ Wait for emobile.UI-automation-Sample to idle (0s)

▶ Capture Localization Screenshots (0s)

▶ Checking `Expect predicate `exists == 1` for object "Back" Button` (0s)

▶ Tap "Back" Button (1s)

Report an issue

10:13

First View

Username

Your username is: Eric

Increment Total

3

**Thanks!**