GeekBrains

Пользовательский интерфейс iOS-приложений

Проектирование интерфейса. Часть II

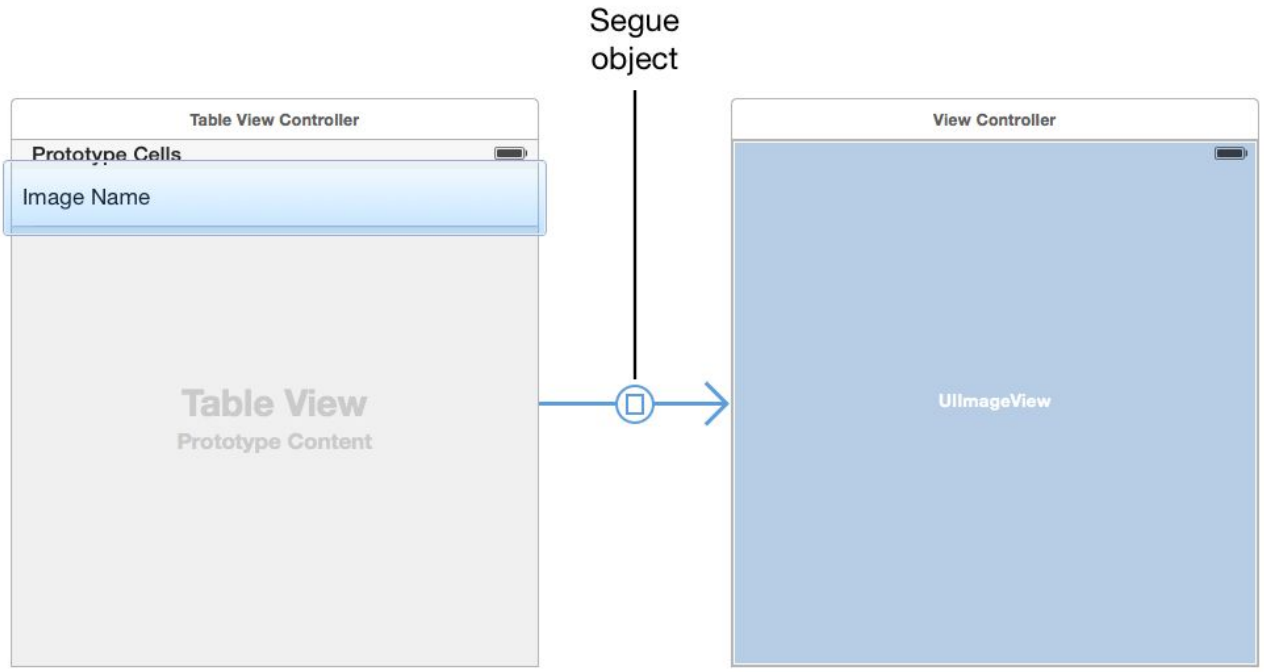
Добавление экранов. Переходы между экранами. UINavigationController. Segue.

Оглавление

Segue

В большинстве приложений несколько экранов — значит, должен быть способ переходить от одного к другому и обратно. В **storyboard** за это отвечает **segue**.

Segue — переход между двумя экранами. Начальной точкой **segue** может быть кнопка, строка в таблице или жест, а конечной — контроллер, который необходимо показать при переходе.

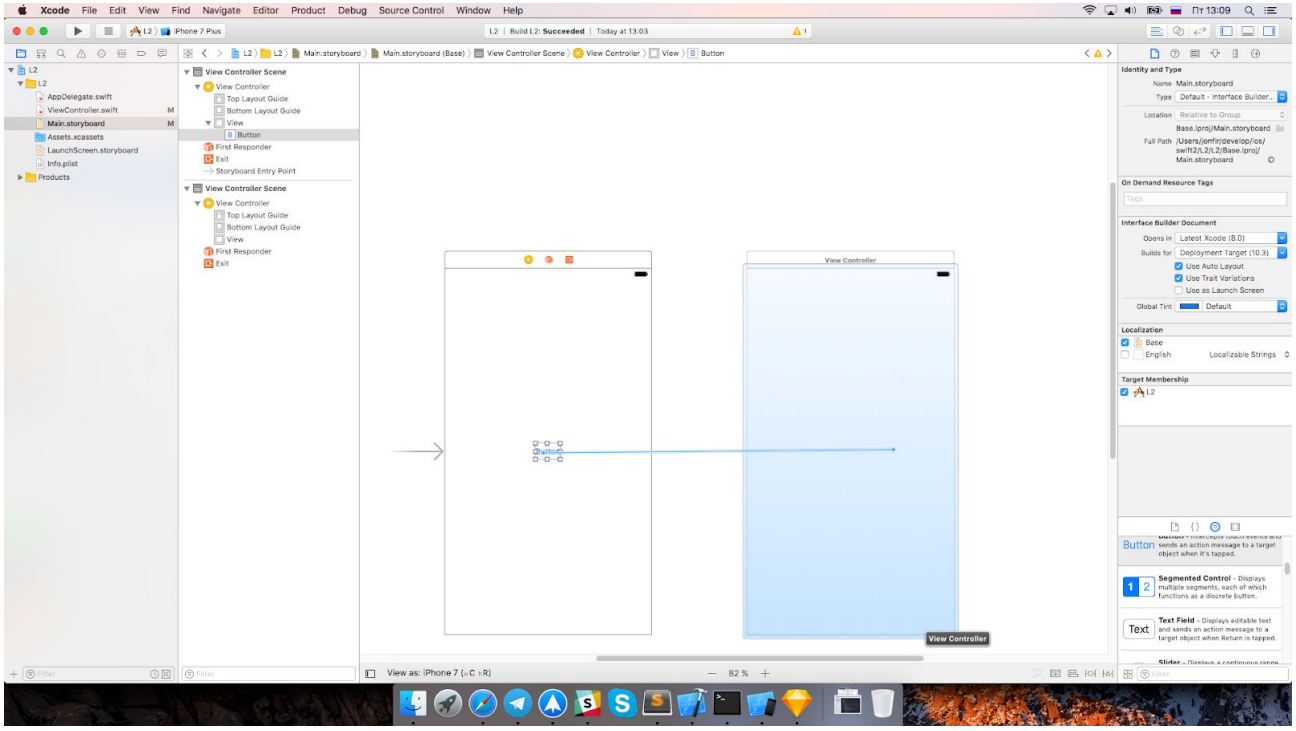


При клике на элемент, являющийся начальной точкой **segue**, переход выполнится автоматически.

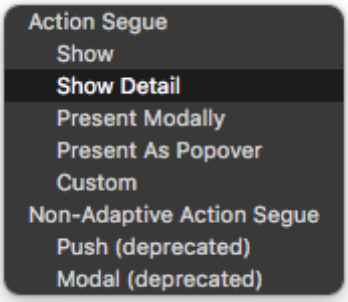
Иногда бывает необходимо запустить переход программно — например, после завершения интернет-запроса. В таком случае можно использовать в качестве начальной точки не кнопку, а сам контроллер, с которого совершается переход, и вызвать его из кода с помощью метода **performSegue(withIdentifier: String, sender: Any?)**.

Чтобы создать **segue** между двумя контроллерами в **storyboard**, щелкните правой кнопкой мыши по элементу на первом контроллере, перетащите появившуюся линию на второй контроллер и отпустите.

Виды переходов



Как только отпустите кнопку мыши, появится таблица с выбором типа перехода. Всего их семь: пять **action segue** и два **non-adaptive action segue**. Последние два использовать не стоит: они считаются устаревшими и сохранены для обратной совместимости.



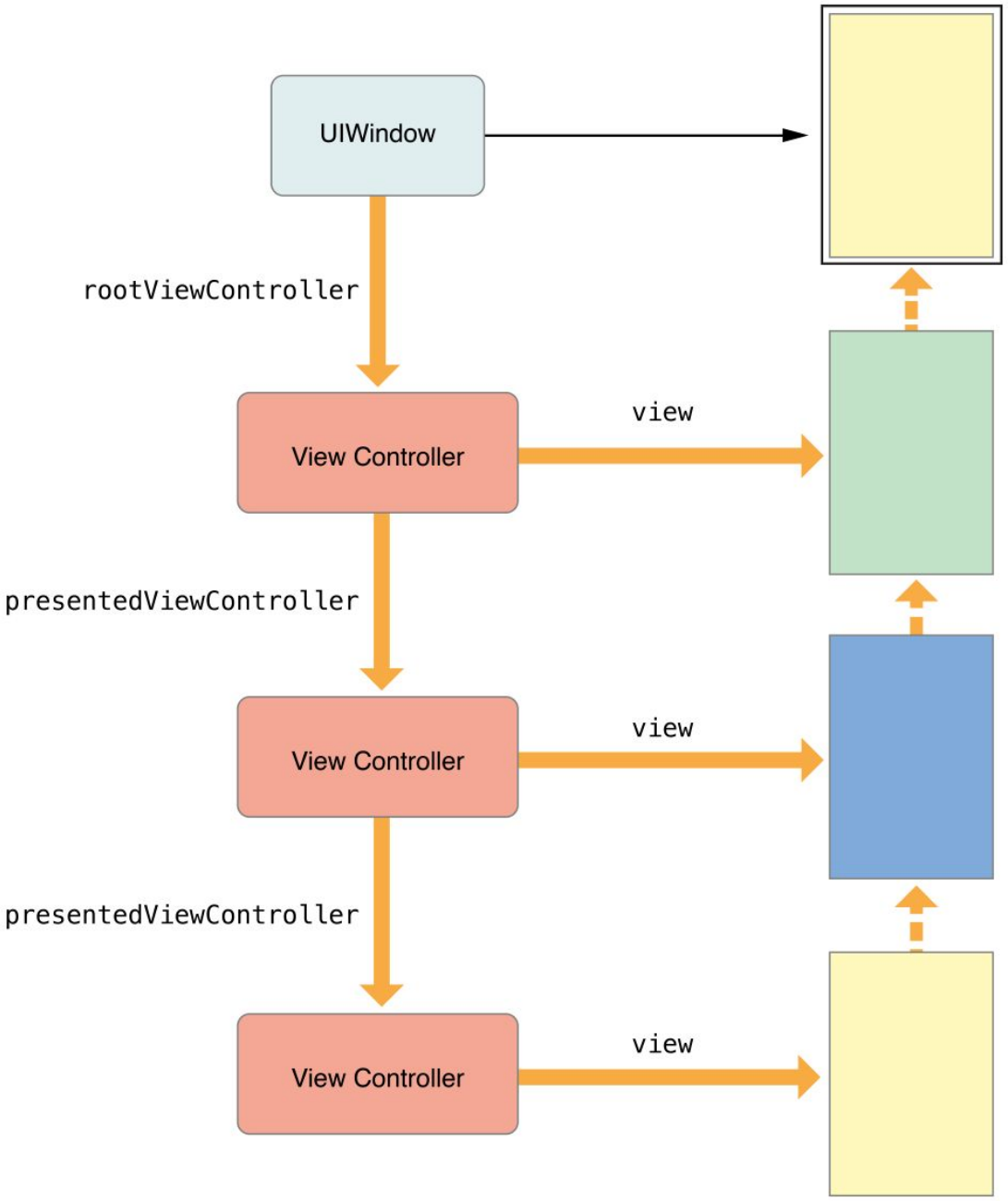
1. **Show (Push)** — показать контроллер. Этот тип перехода зависит от контекста. Если контроллеры находятся в **UINavigationController**, новый будет добавлен в его стек. Если **UINavigationController** не используется, новый контроллер будет показан поверх предыдущего.

2. **Show Detail (Replace)** — работает только при использовании **UISplitViewController**. Заменяет основную часть экрана.

3. **Present Modally** — показать модально. Вне зависимости от контекста показывает новый контроллер поверх предыдущего.

4. **Present as Popover** — показать как всплывающее окно. На телефонах работает, как **Present Modally**. На планшетах новый контроллер будет показан в небольшом всплывающем окне.

5. **Custom** — пользовательский тип перехода. Его поведение определяет разработчик. Чтобы применить его, необходимо создать свой подкласс **UIStoryboardSegue** и определить в нем логику перехода.



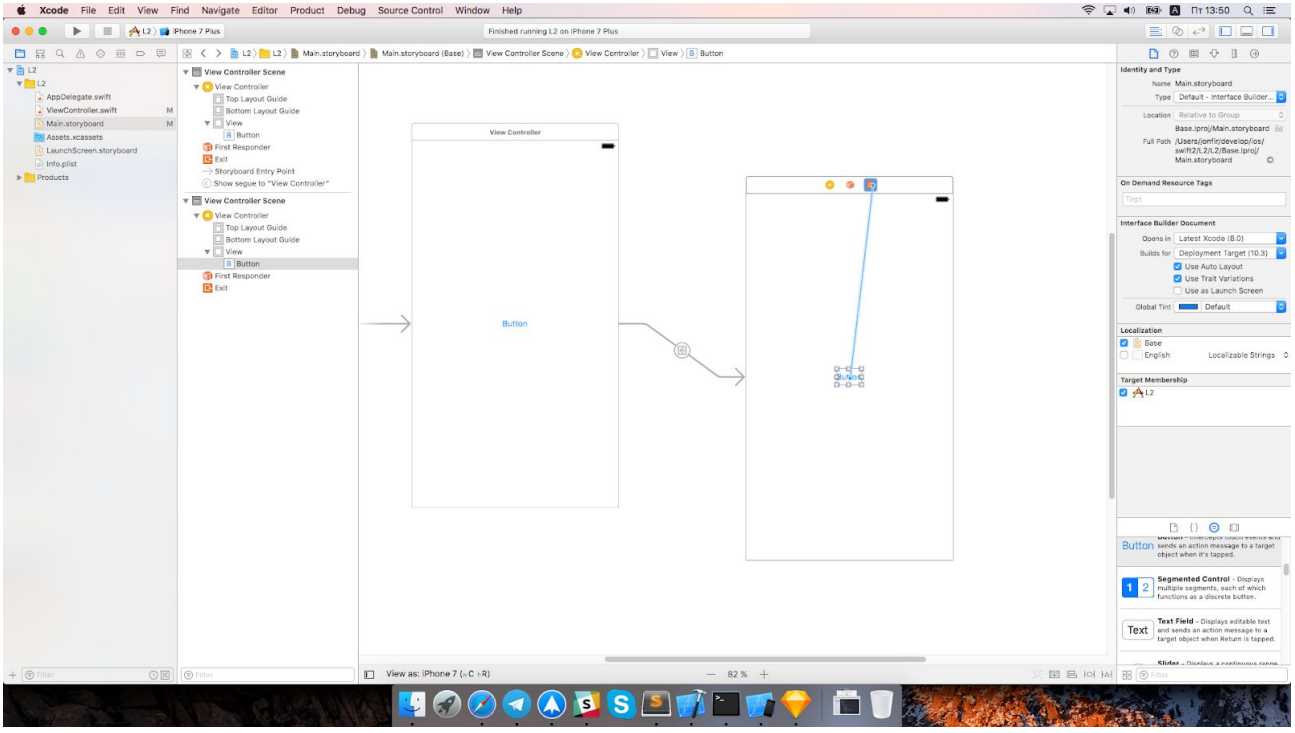
Unwind Segue

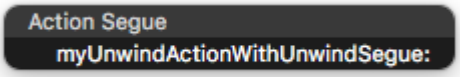
Обычные **segue** используются для показов новых контроллеров. Но существует отдельный тип, предназначенный для возвращения назад, — **Unwind Segue**. Его не получится создать целиком в **storyboard**, потребуется немного кода в контроллере. Чтобы реализовать **Unwind Segue**, выполните следующие шаги:

1. Откройте класс контроллера, на который необходимо вернуться.
2. Добавьте метод, который надо вызвать при обратном переходе.

```
@IBAction func myUnwindAction(unwindSegue: UIStoryboardSegue)
```

3. Откройте **storyboard** и выберите контроллер, с которого будет выполнен переход назад.
4. Выберите элемент, по нажатию на который состоится переход. Нажмите на него правой кнопкой мыши и перетащите полоску на иконку Exit.
5. В открывшейся таблице выберите метод, реализованный ранее.



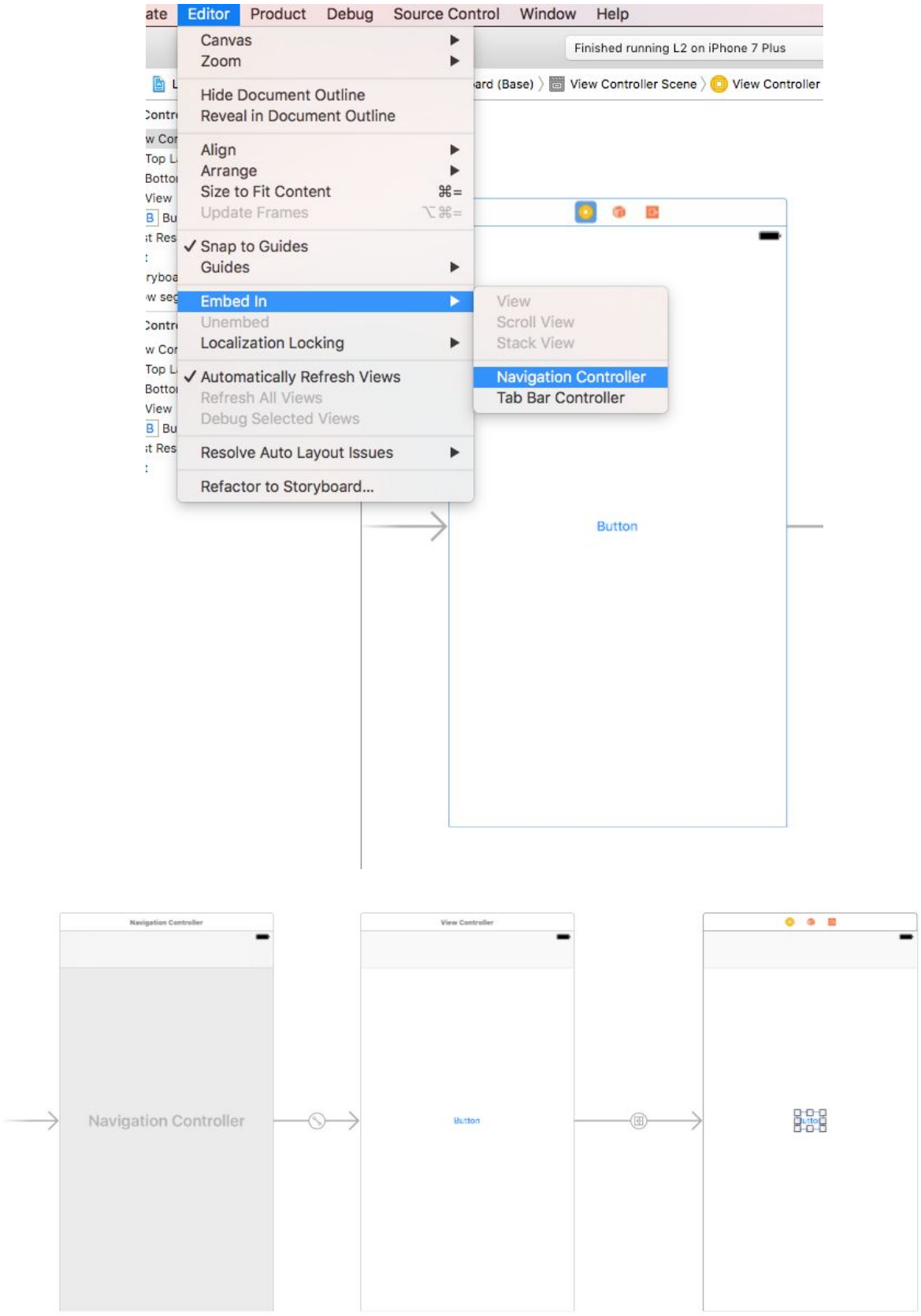


UINavigationController

Для переходов между экранами достаточно использовать **segue**. Но при этом необходимо самостоятельно заботиться об элементах навигации. Чтобы было легче, добавлен **UINavigationController**.

Он выполняет несколько задач. Отображает **UINavigationBar** — панель навигации вверху экрана, автоматически добавляет кнопку «Назад», по которой пользователь может вернуться на предыдущий экран. Управляет переходами так, чтобы контроллеры не отображались один поверх другого, а заменялись на экране. При этом **UINavigationController** хранит всю цепочку контроллеров, которые были показаны. В любой момент можно вернуться на один или несколько экранов назад или к самому первому контроллеру в цепочке. Можно мгновенно изменить их все.

Чтобы добавить в проект **UINavigationController**, выберите первый контроллер приложения, откройте меню **Editor — Embed in — Navigation Controller**.

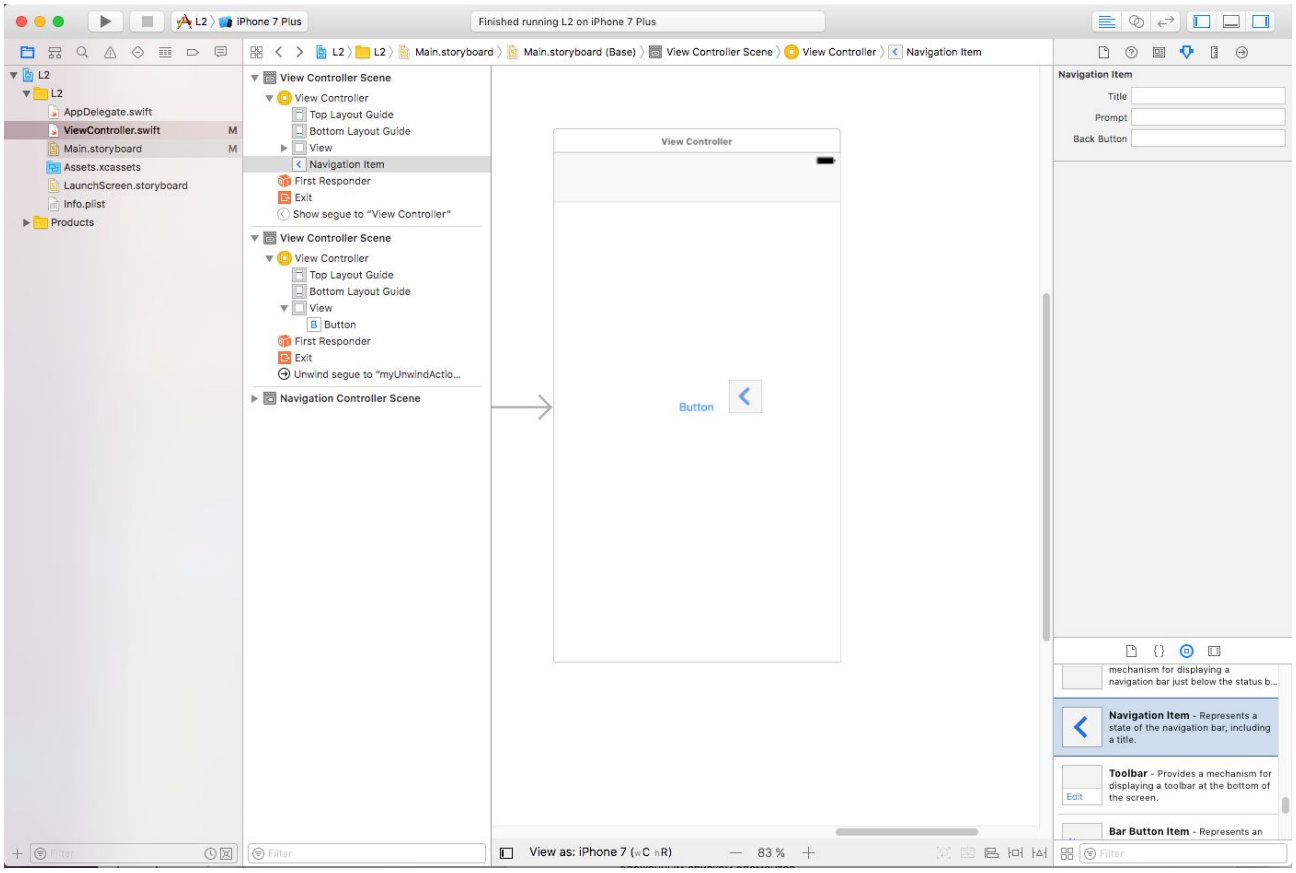


На макеты добавилась панель навигации.

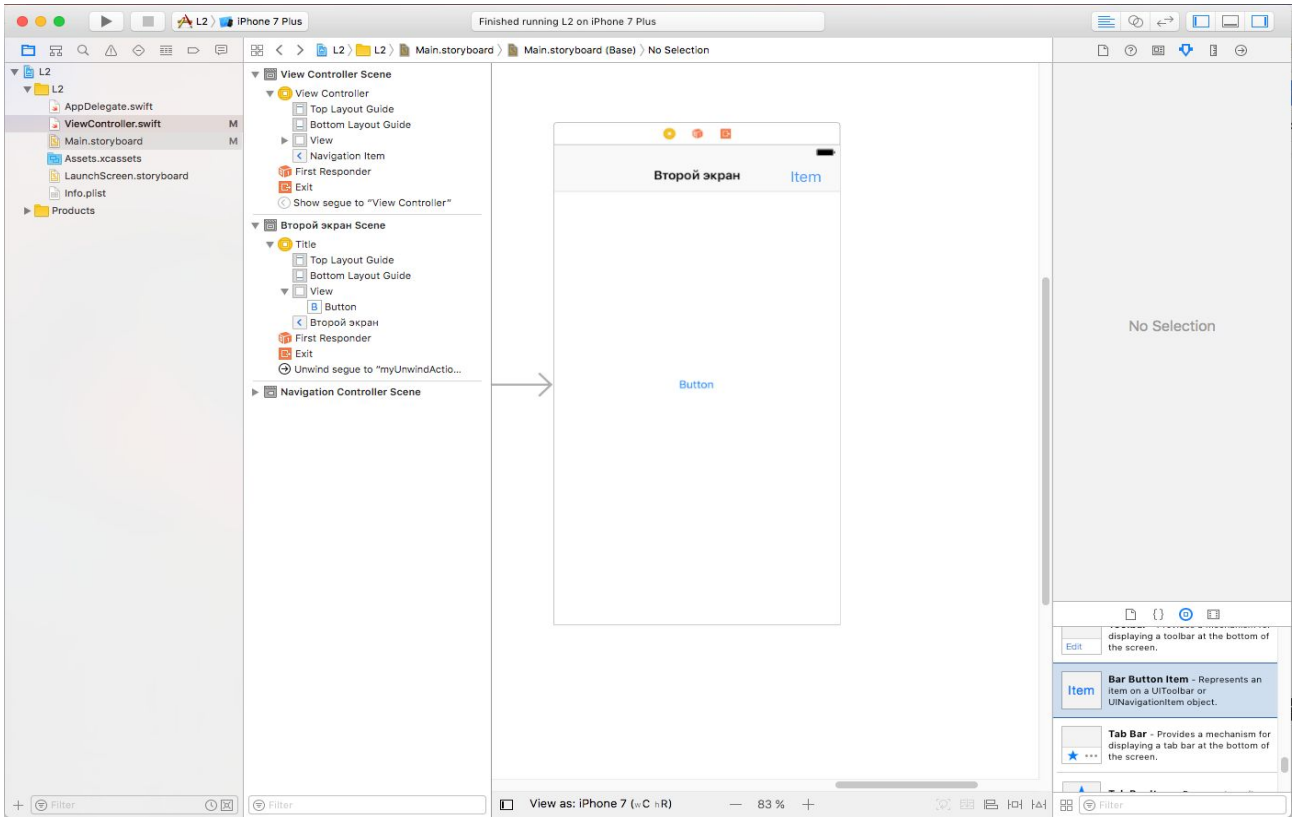
Панель навигации

Как правило, на панель навигации требуется добавить кнопки и установить лейбл. Для этого используется элемент **UINavigationItem**.

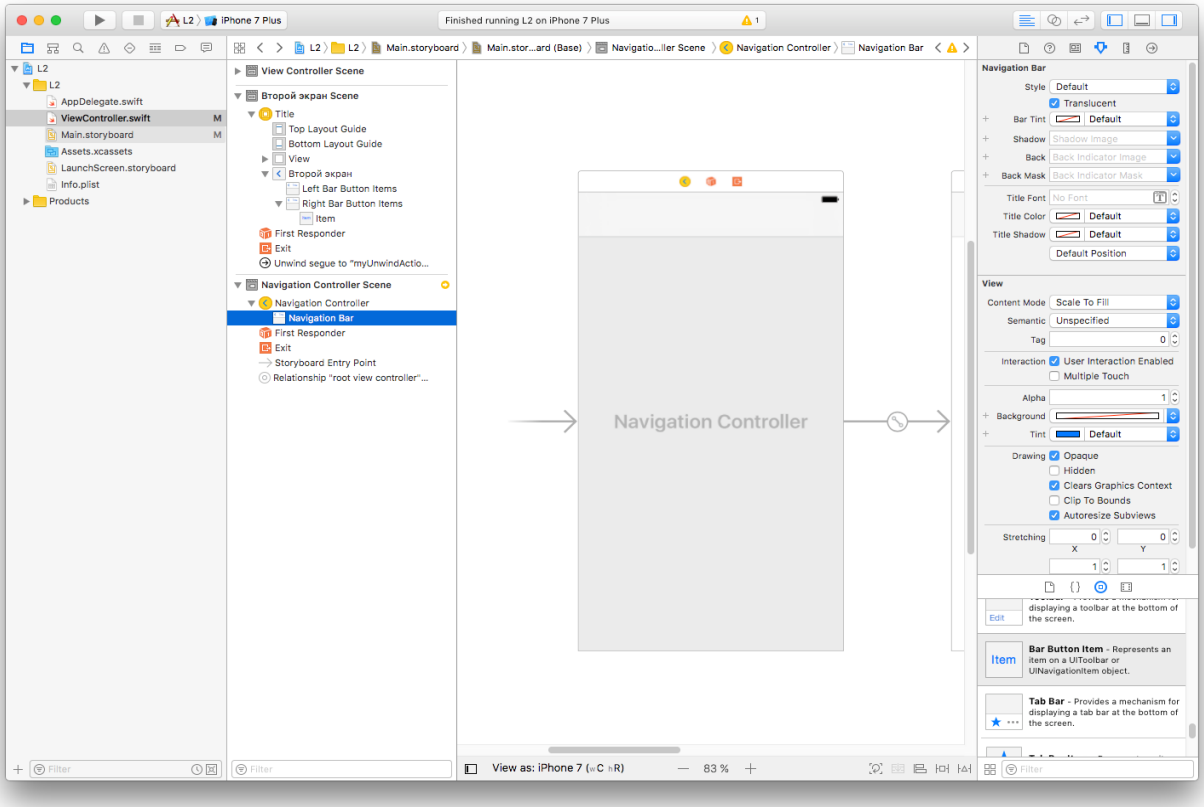
На первом контроллере в цепочке навигации он уже есть. На все остальные необходимо добавлять его вручную.

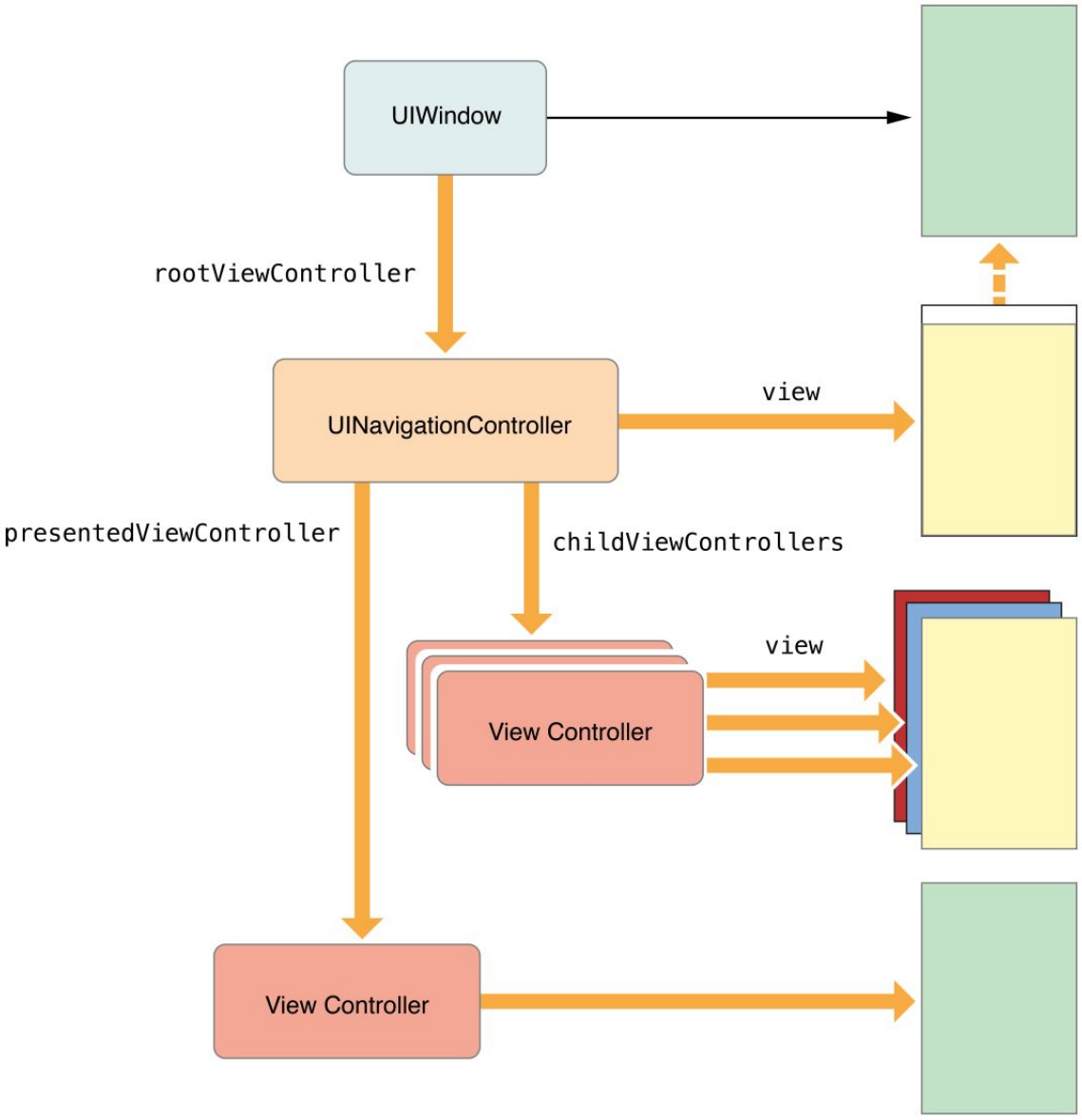


Когда **UINavigationItem** добавлен, на панели атрибутов можно установить заголовок экрана. Чтобы добавить кнопки, используется **UIBarButtonItem**. Есть один нюанс: если добавить кнопку в левую часть панели, пропадет кнопка «Назад».



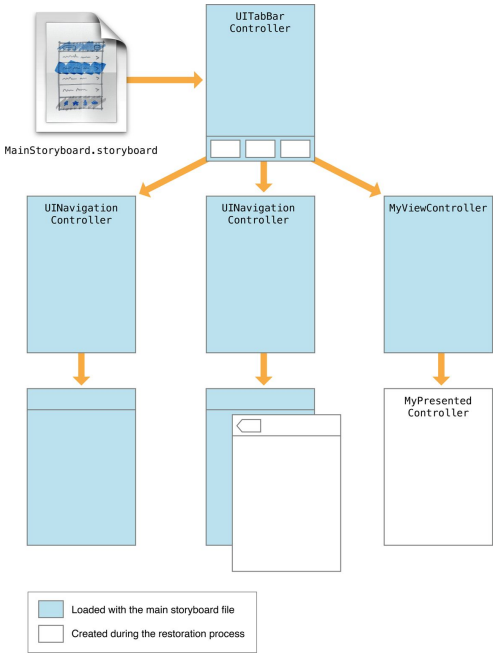
Если выбрать **UINavigationController** и его панель навигации, можно изменить ее атрибуты.




UIWindow
rootViewController
UINavigationController
view
presentedViewController
childViewControllers
view
View Controller
View Controller

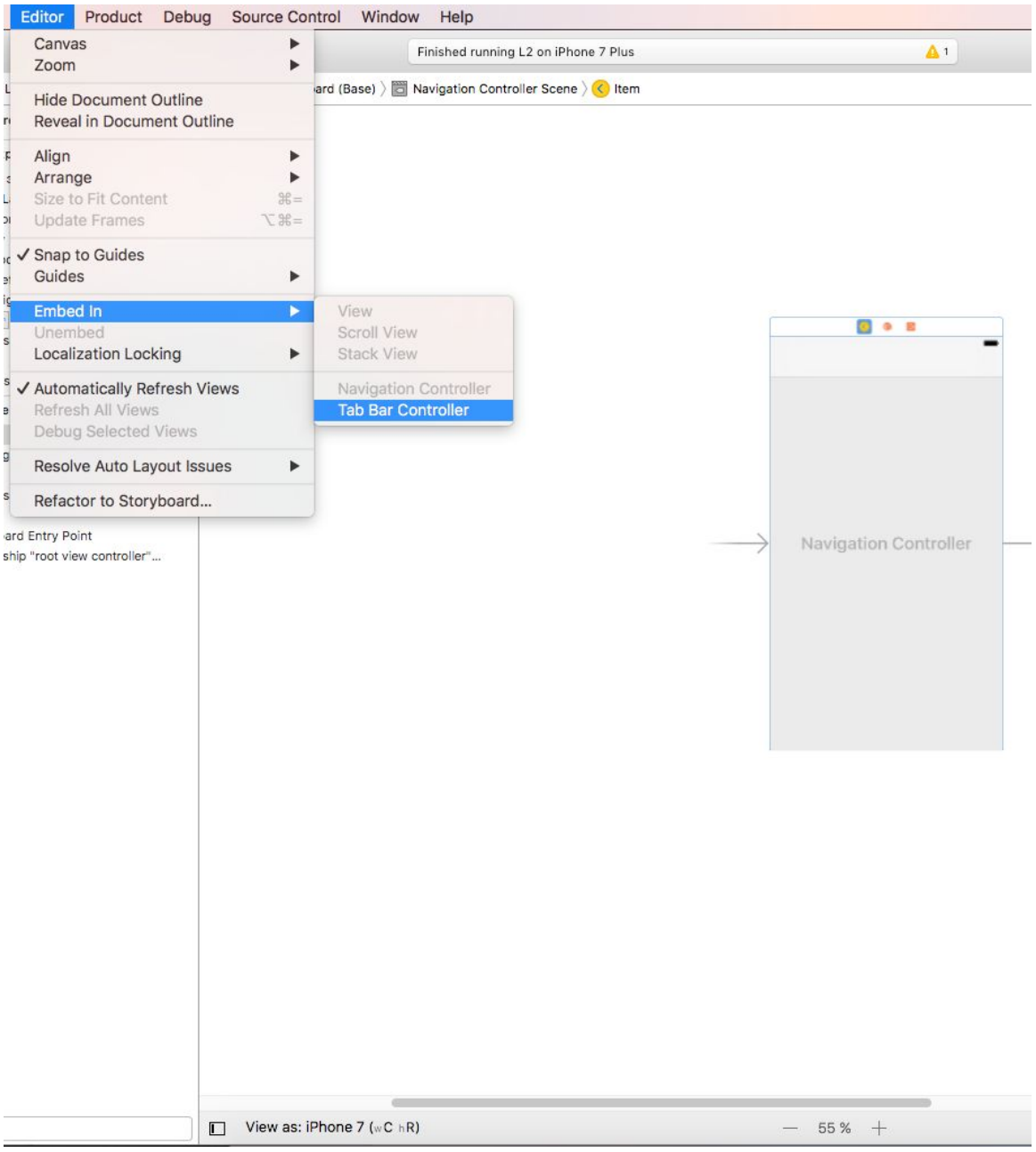
UITabbarController

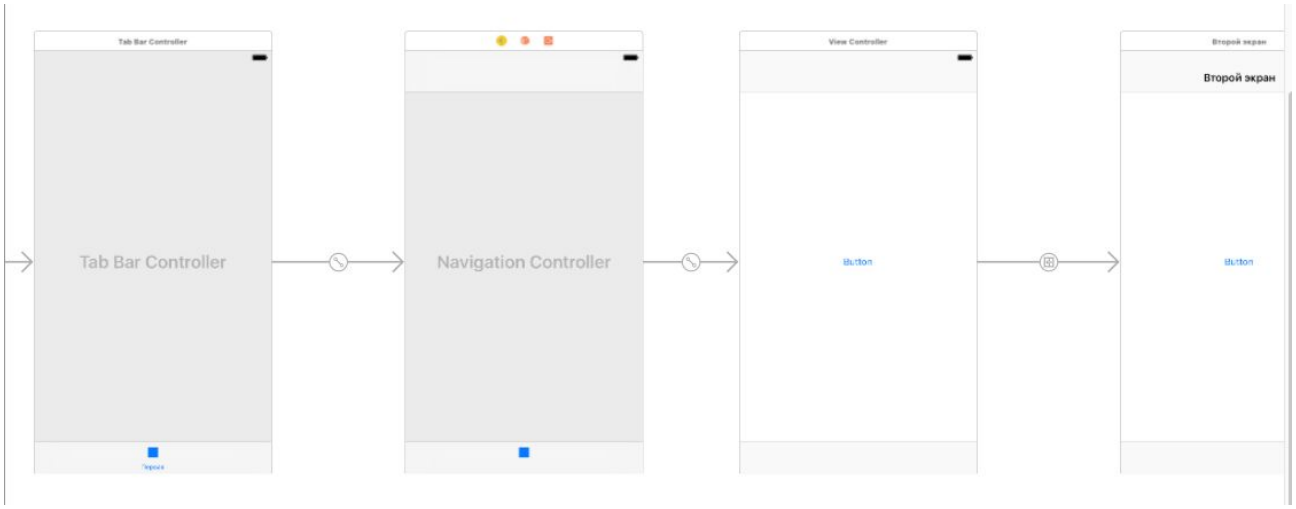
Еще один вариант построения переходов между экранами — **UITabbarController**. Он позволяет реализовать несколько параллельных цепочек навигации, разделенных по вкладкам.



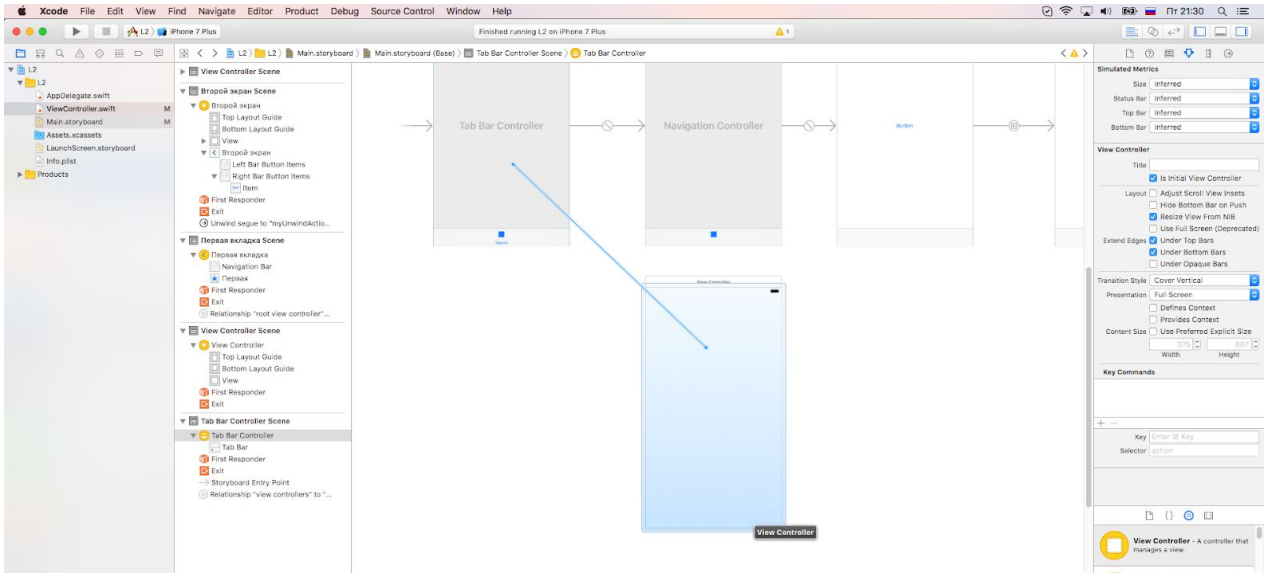
В каждой вкладе может быть один контроллер или несколько, может присутствовать даже **UINavigationController**, управляющий навигацией внутри вкладки.

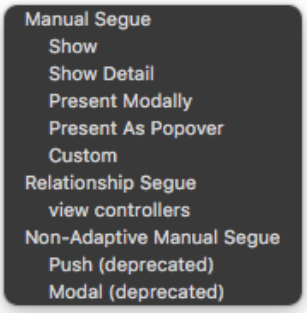
Чтобы добавить в проект **UITabbarController**, выберите первый контроллер приложения, откройте меню **Editor — Embed in — Tab Bar Controller**.

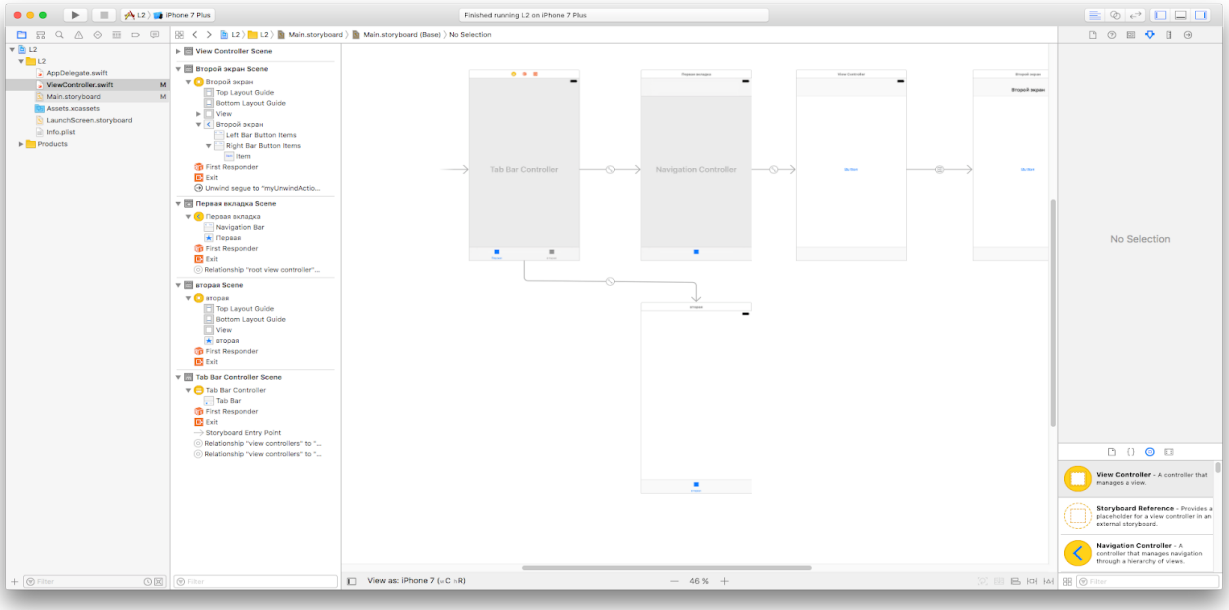

Editor
Product
Debug
Source Control
Window
Help
Canvas
Zoom
Hide Document Outline
Reveal in Document Outline
Align
Arrange
Size to Fit Content
Update Frames
Snap to Guides
Guides
Embed In
Unembed
Localization Locking
Automatically Refresh Views
Refresh All Views
Debug Selected Views
Resolve Auto Layout Issues
Refactor to Storyboard...
View
Scroll View
Stack View
Navigation Controller
Tab Bar Controller
Finished running L2 on iPhone 7 Plus
Navigation Controller Scene
Item
Navigation Controller
View as: iPhone 7
55 %


Tab Bar Controller
Navigation Controller
Второй экран

Чтобы добавить еще одну вкладку, выберите **UITabbarController**, нажмите на него правой кнопкой мыши и перетащите линию на контроллер, который должен отображаться в новой вкладке. В появившемся меню выберите **relationship segue — view controller**.



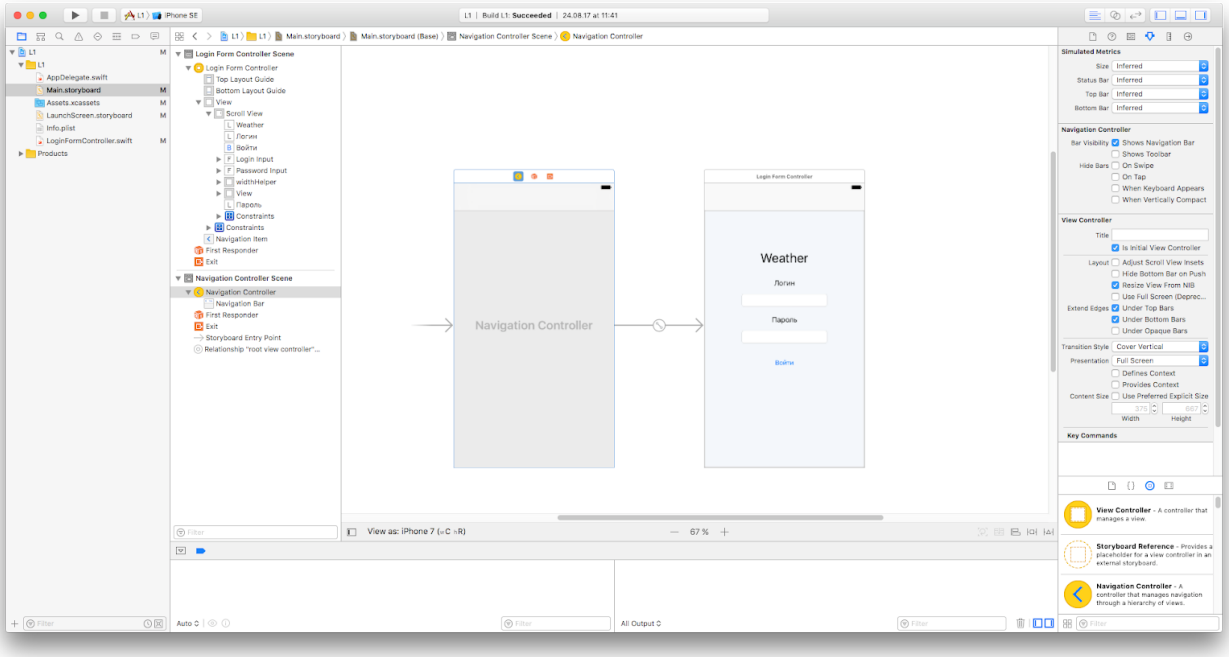




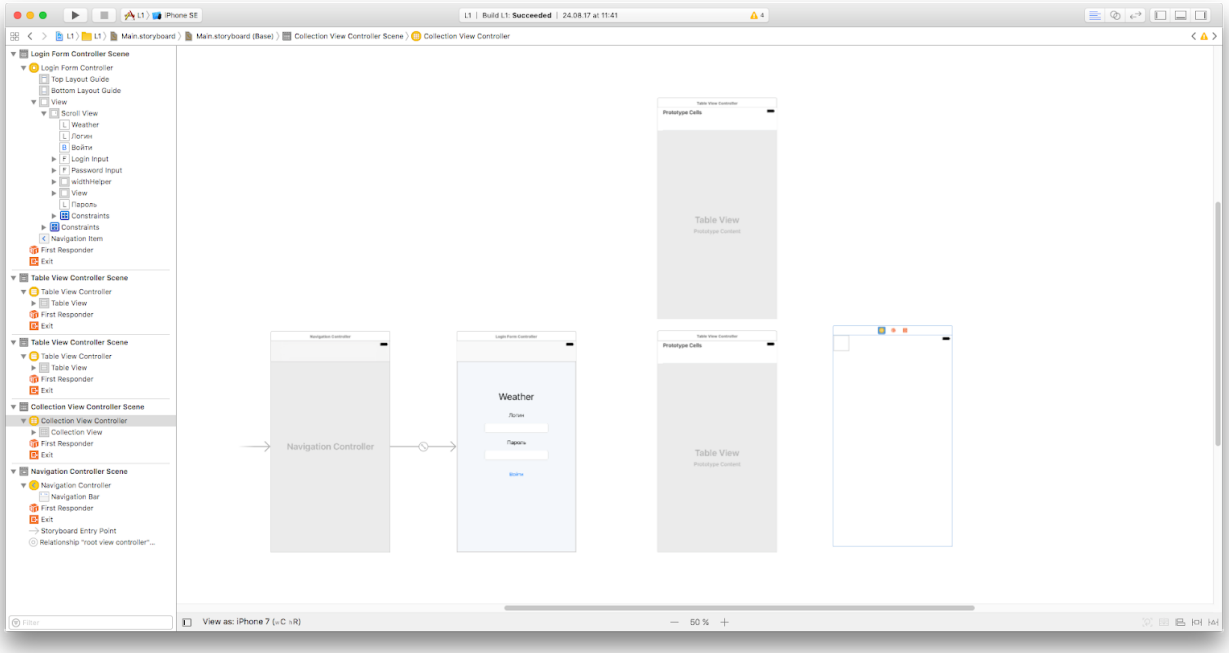
Чтобы изменить атрибуты панели вкладок, необходимо выбрать ее непосредственно на **UITabbarController**. Чтобы поменять имя вкладки, следует выбрать ее на контроллере, который в ней отображается.

Создание клиента для сервиса openweathermap.org

Добавим в приложение немного навигации. Начнем с **UINavigationController**.

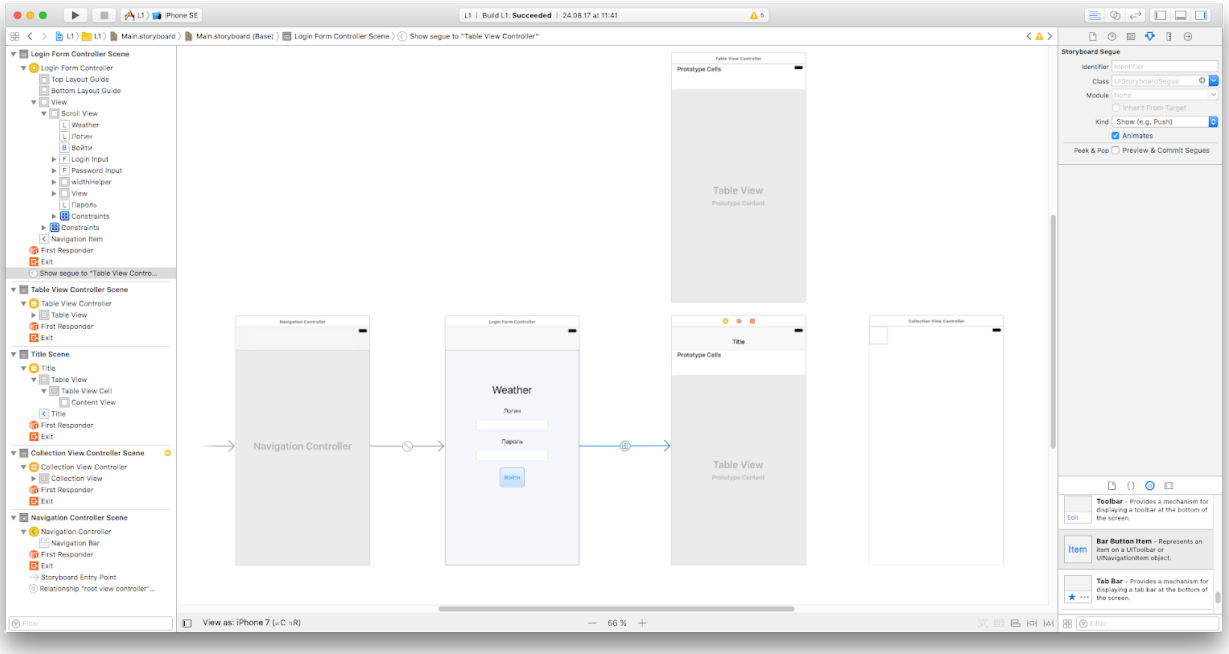


Теперь добавим два **UITableViewController** и один **UICollectionViewController**.



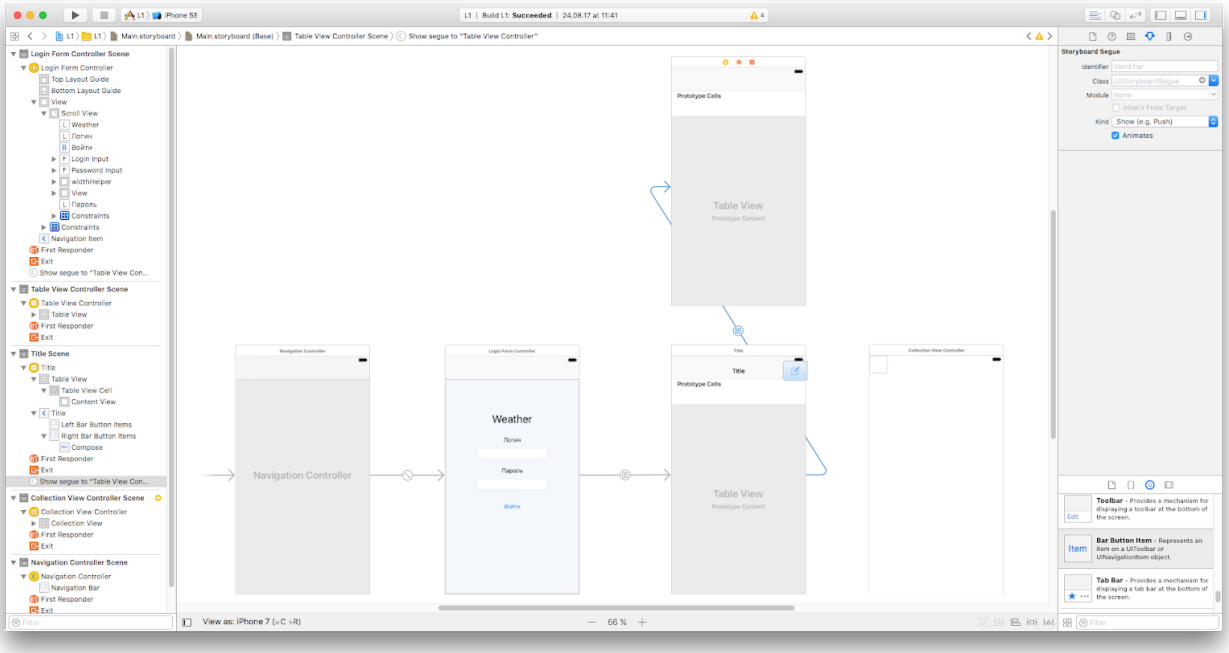
Один **UITableViewController** будет отображать доступные для просмотра погоды города, второй — все города с возможностью добавления в доступные. **UICollectionViewController** будет отображать погоду в конкретном городе.

Настроим навигацию в проекте. Кнопка «Войти» будет инициировать переход на первый **UITableViewController**.

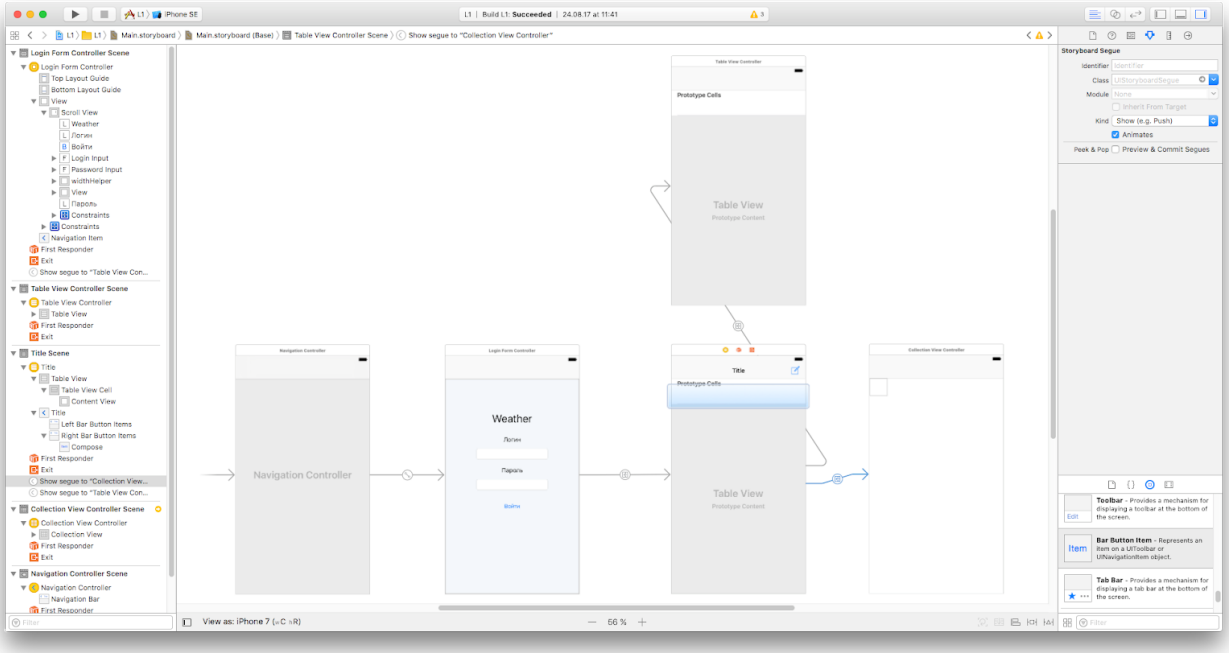


На первый **UITableViewController** добавим **UINavigationItem**, кнопку и изменим атрибут **system item** на **add**.

От него будет идти **segue** на второй **UITableViewController**.



Последний **segue** будет выполнять переход на **UICollectionViewController**.



Можно запустить проект и убедиться, что переходы работают. Но при нажатии на кнопку «Войти» переход будет совершен без проверки логина и пароля. Добавим эту проверку. Чтобы подтвердить или отменить переход, в **UIViewController** есть специальный метод.

```
func shouldPerformSegue(withIdentifier identifier: String, sender: Any?) -> Bool
```

Откроем **LoginFormController** и переопределим этот метод.

```
override func shouldPerformSegue(withIdentifier identifier: String, sender: Any?) -> Bool {
        let login = loginInput.text!
        let password = passwordInput.text!

        if login == "admin" && password == "123456" {
            return true
        } else {
            return false
        }
    }
```

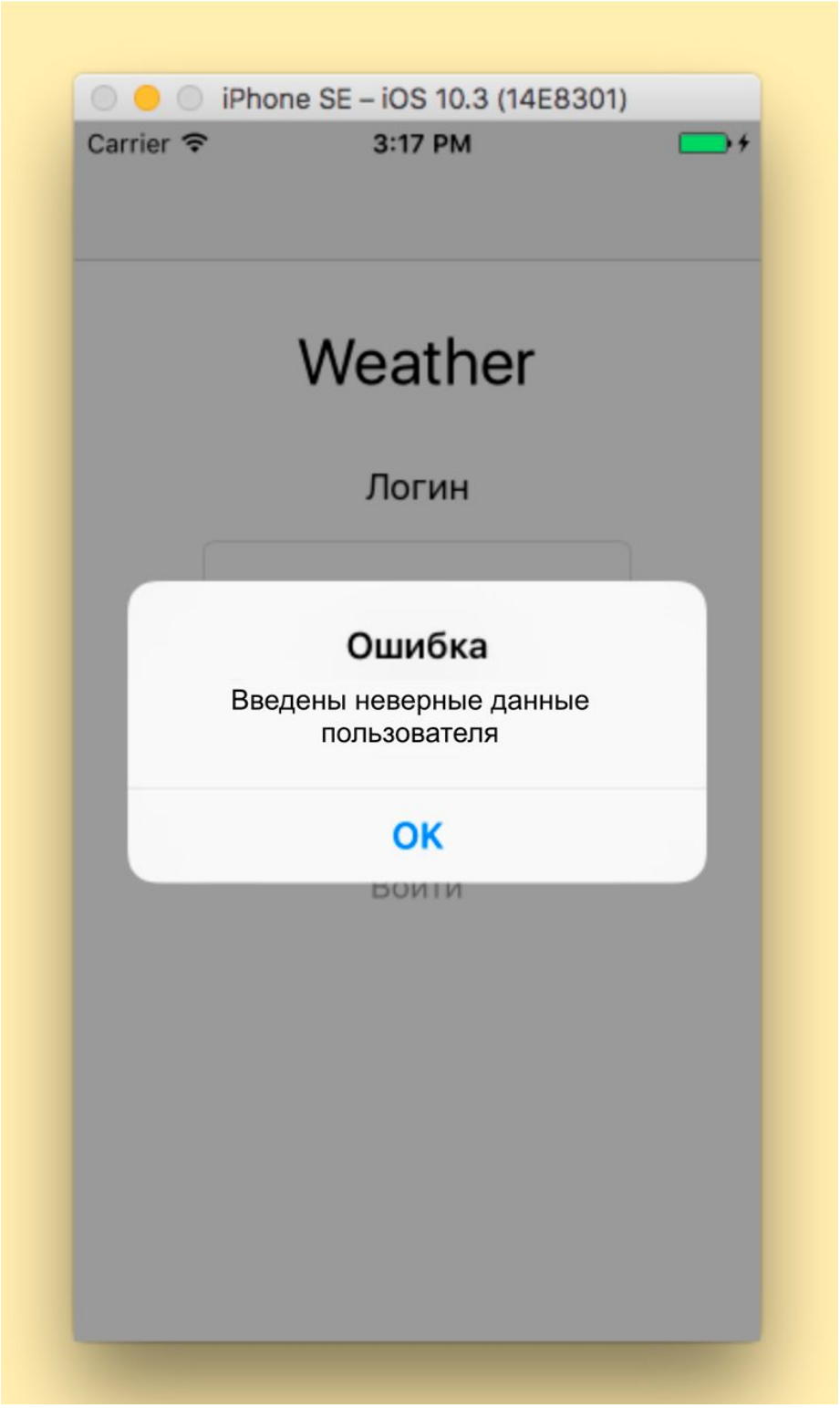
Прежде чем совершить переход, **UIKit** вызовет метод **shouldPerformSegue**. Если метод вернет **true**, переход выполнится, а если **false** — будет отменен. Проверяем введенные пользователем данные: если они верны — возвращаем **true**, в противном случае — **false**.

Теперь проверка данных осуществляется, но если они не верны, не видно никаких действий — кажется, что кнопка просто не нажимается. Покажем пользователю сообщение с помощью специального класса **UIAlertController**. Изменим метод проверки:

```
override func shouldPerformSegue(withIdentifier identifier: String, sender: Any?) -> Bool {
        let login = loginInput.text!
        let password = passwordInput.text!

        if login == "admin" && password == "123456" {
            return true
        } else {
            // Создаем контроллер
               let alert = UIAlertController(title: "Ошибка", message: "Введены неверные данные пользователя", preferredStyle: .alert)
            // Создаем кнопку для UIAlertController
               let action = UIAlertAction(title: "OK", style: .cancel, handler: nil)
            // Добавляем кнопку на UIAlertController
            alert.addAction(action)
            // Показываем UIAlertController
            present(alert, animated: true, completion: nil)

            return false
        }
    }
```



Теперь если пользователь ошибется при вводе логина и/или пароля, он об этом узнает.

Осталось привести код в подобающий вид. Метод проверки перехода выглядит ужасно — в нем слишком много кода. Вынесем показ ошибки и проверку данных в отдельные методы.

```
override func shouldPerformSegue(withIdentifier identifier: String, sender: Any?) -> Bool {
        // Проверяем данные
        let checkResult = checkUserData()

        // Если данные не верны, покажем ошибку
        if !checkResult {
            showLoginError()
        }

        // Вернем результат
        return checkResult
    }

    func checkUserData() -> Bool {
        guard let login = loginInput.text,
            let password = passwordInput.text else { return false }

        if login == "admin" && password == "123456" {
            return true
        } else {
            return false
        }
    }

    func showLoginError() {
        // Создаем контроллер
           let alter = UIAlertController(title: "Ошибка", message: "Введены не верные данные пользователя", preferredStyle: .alert)
        // Создаем кнопку для UIAlertController
        let action = UIAlertAction(title: "OK", style: .cancel, handler: nil)
        // Добавляем кнопку на UIAlertController
        alter.addAction(action)
        // Показываем UIAlertController
        present(alter, animated: true, completion: nil)
    }
```

Логика не поменялась сильно, зато код стало проще читать.

Практическое задание

На основе ПЗ из предыдущего урока:

1. Добавить в приложение **UITabbarViewController**, три **UITableViewController** и один **UICollectionViewController**.
2. После того как пользователь ввел верные логин и пароль, перейти на **UITabbarViewController**.
3. Добавить две вкладки в **UITabbarViewController**.

4. На первой вкладке настроить переходы в следующем порядке: **UINavigationController — UITableViewController — UICollectionViewController.** Это будущая вкладка для отображения друзей пользователя «ВКонтакте» и его фотографий. Переход с таблицы на коллекцию должен происходить по нажатию на ячейку.

5. На второй вкладке — в таком порядке: **UINavigationController — UITableViewController — UITableViewController**. Первый контроллер для отображения групп пользователя, второй — для отображения глобального поиска групп, которые могут быть интересны пользователю. Для перехода с первой таблицы на вторую на **NavigationBar** необходимо создать **Bar Button Item**.

*В этих цепочках **UINavigationController** не является отдельным экраном, он нужен для управления переходами.*

Дополнительные материалы

1. UIStoryboardSegue.
2. View Controller Programming Guide for iOS.

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. UIStoryboardSegue.
2. View Controller Programming Guide for iOS.