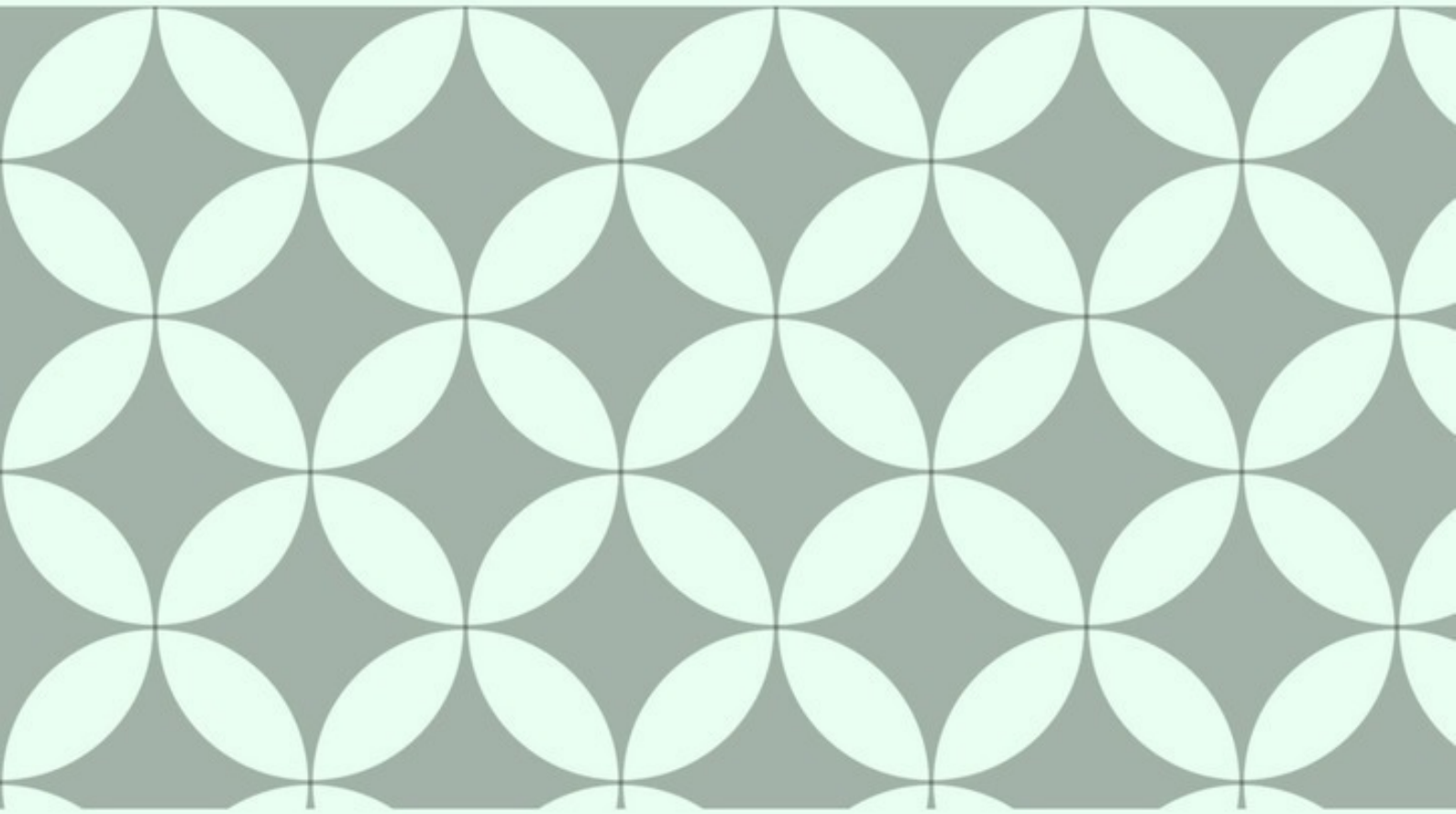
Иван Трещев



# ПРОГРАММИРОВАНИЕ ДЛЯ МОБИЛЬНЫХ ПЛАТФОРМ

## IOS

Иван Трещев

# Программирование для мобильных платформ. IOS

«Издательские решения»

**Трещев И.**

Программирование для мобильных платформ. IOS / И. Трещев — «Издательские решения»,

ISBN 978-5-44-939973-1

Данная книга обобщает опыт работы лаборатории мобильных приложений на базе ФГБОУ ВО КнАГУ, где автор был ее руководителем. Приложения, разработанные в книге, были успешно выложены в магазин приложений. В книге вы найдете описание основных моментов для разработки приложений.

ISBN 978-5-44-939973-1

# Содержание

# Программирование для мобильных платформ IOS

**Иван Трещев**

*Участие в тестировании разработанных приложений* Владислав Андреевич Чусов
*Участие в работе над иллюстрациями* Мария Сергеевна Аксютина
*Участие в верстке и корректуре* Анастасия Сергеевна Ватолина



ISBN 978-5-4493-9973-1
Создано в интеллектуальной издательской системе Ridero

# Введение

Разработка мобильных приложений может приносить и стабильный доход и стать точкой роста для профессионала. Современный рынок мобильных устройств полон различными аппаратами всевозможных форм-факторов, если говорить про IOS то это различные телефоны и планшеты фирмы Apple, умные часы и плееры. Конечно по сравнению с другими платформами (скажем Windows Phone или Android) читатель, для того чтобы начать разрабатывать приложения для проприетарной операционной системы для мобильных устройств Apple, должен по крайней мере приобрести ПЭВМ Mac производства фирмы Apple, поскольку Xcode – среда программирования доступна только для MacOS (что является не дешевым удовольствием) и желательно иметь под рукой для тестирования минимальный набор аппаратных устройств – телефон и планшет, хотя в среде программирования и предусмотрены различные эмуляторы, но опыт показал, что все же желательно тестирование проводить на реальных устройствах. Итого инвестиции в процесс разработки приложений порядка 1500€. Помимо всего вышеперечисленного ежегодно необходимо оплачивать 100 долларов США как сбор для разработчиков (к примеру для платформы Android взнос единоразовый), что осуществить так же не просто, особенно для жителей России, поскольку номер факса для связи с Apple именно из США.

Лаборатория которой руководил автор на протяжении 5 лет занималась разработкой различных приложений для самых популярных за последнее пятилетие операционных систем носимых устройств – Android, IOS, Windows Phone. Сегодня платформа IOS насчитывает многомиллионную аудиторию и располагает одним из самых удобных и эргономичных способов для авторов (будь то песни, книги или приложения) для монетизации своих творений при этом не неся затрат на тиражирование, продажу, экспозицию и другие накладные расходы.

По сравнению с другими платформами IOS выделяют несколько немаловажных аспектов. Во-первых эмуляторы есть для всех устройств (не просто для популярных как Windows Phone и просто эмуляторы Android). Во-вторых это конечно то, что ваше приложение действительно будут тестировать специалисты Apple. Причем тестировать будут весьма притязательно. Не стоит надеяться что только разработав приложение и отправив его на модерацию оно тут же будет размещено в магазине (как в случае с Android). Для этого потребуется от двух рабочих дней. В-третьих все же есть единый стиль, единые стандарты программирования, единообразные интерфейсы и механизмы их создания. Member Center, ICloud, синхронизация со всей инфраструктурой Apple. Этот список довольно велик. Причем благодаря постоянным обновлениям операционной системы и «экосистемы» ежегодно появляются новые технологии компании, которые интегрируются в приложения.

В работе не было попытки дать детальное объяснение всех возможных аспектов программирования под IOS. А скорее данная книга будет полезна тем кто все же открыл Xcode приобрел аккаунт разработчика и планирует начать пробираться через тернии и хитросплетения современных технологий программирования.

Данная книга посвящена разработке приложений именно под платформу от Apple и является второй в цикле, над которыми автор работает в настоящее время.

У читателя предполагается опыт программирования на объектно-ориентированном языке, желательно опыт на C#.

Автор хотел бы выразить огромную благодарность Чусову В. А. и Аксютиной М. С. без которых невозможно было работать.

# Основы работы с Xcode

Сегодня мы разберемся, как создать свой проект, познакомимся с основными объектами Фреймворка UIKit.

И так, первым шагом мы запустим Xcode, и создадим проект, после этих действий, должно появиться окно, где можно выбрать уже подготовленный для чего-либо проект, например, выберем «Single View Application» (рис. 1.1). С остальными типами познакомимся позже.

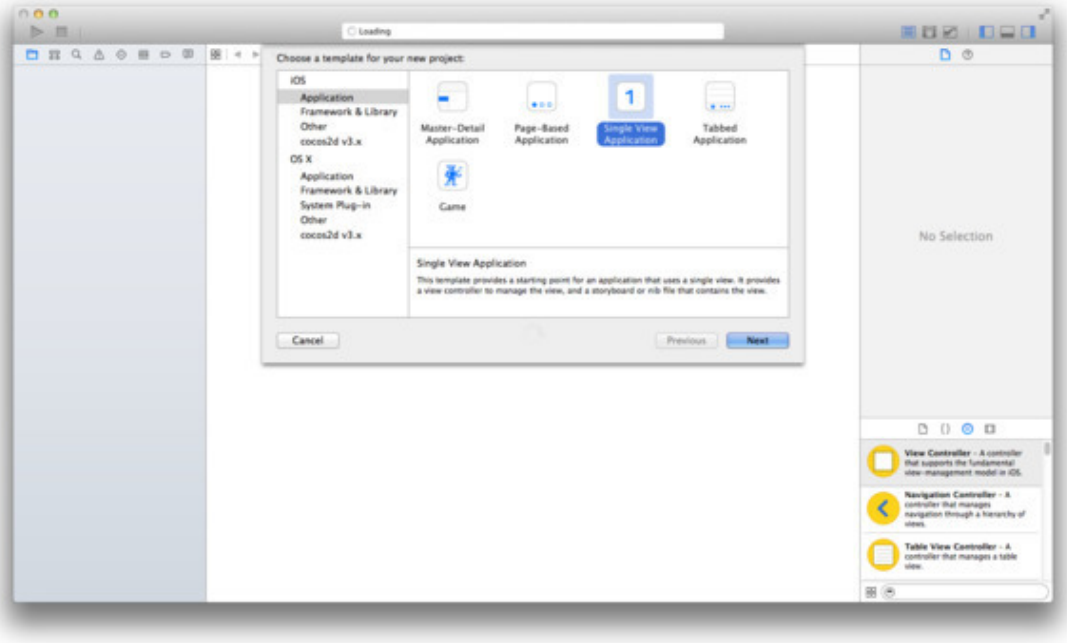



Рисунок 1.1 – Выбор типа приложения

Далее вводим данные, название проекта, название компании, уникальный идентификатор компании, язык на котором мы будем программировать, выбираем Objective-C, далее выбираем для какого типа устройств создаем приложением, и последний пункт – Core data, это некая оболочка/Фреймворк для обработки данных. Пока что, вы можете вводить любые данные, т.к. этот всего лишь тестовый проект.

После проделанных манипуляций, перед нами должно появится окно с настройками нашего проекта (рис. 1.2).

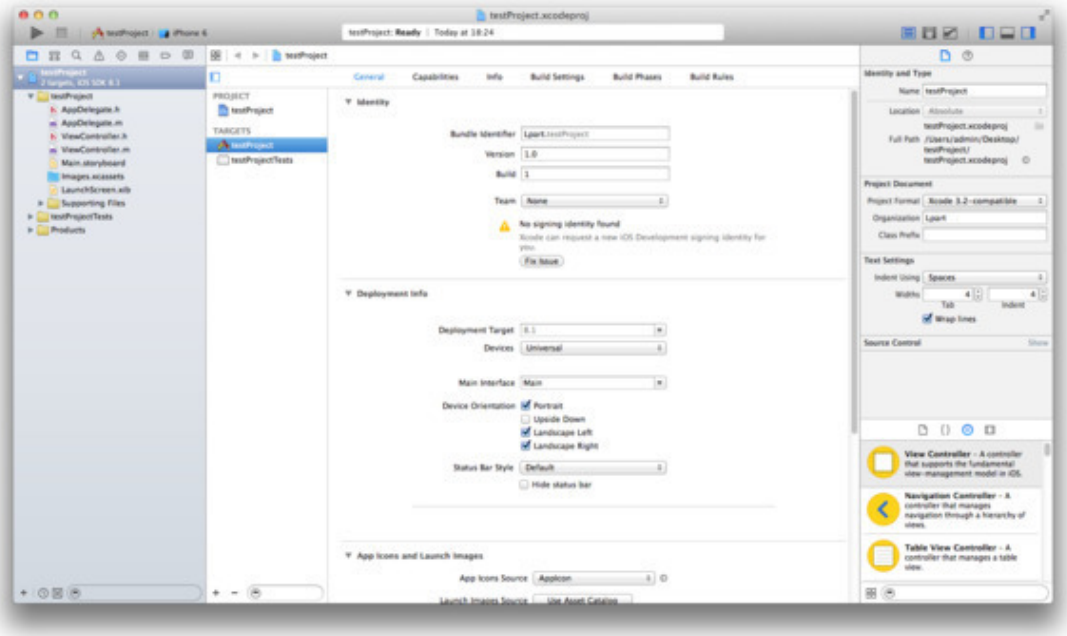


Рисунок 1.2 – Настройки приложения

Разберемся с targets. Грубо говоря, один target – один вид нашего проекта. В каждом target мы можем добавлять или убирать какие-либо объекты, данные. К примеру, мы создаем приложение для двух компаний, но с разными картинками, мы в одном target вставляем одни картинки, а в другом иные картинки. Так же есть еще такое понятие Scheme – это схема, которую мы можем изменять, создавать свою новую или создать автоматическую Она позволяет нам настраивает запуск приложения, его сборку и т. д.

Далее идут основные параметры проекта (рис. 1.3). Identity. В нем предоставляется информация о проекте и команде разработчиков. Deployment info – здесь мы указываем, какие версии iOS будет поддерживать наше приложение, для какого устройства разработка ведется, какие допустимы ориентации устройства. Status Bar – это строка состояние устройства.

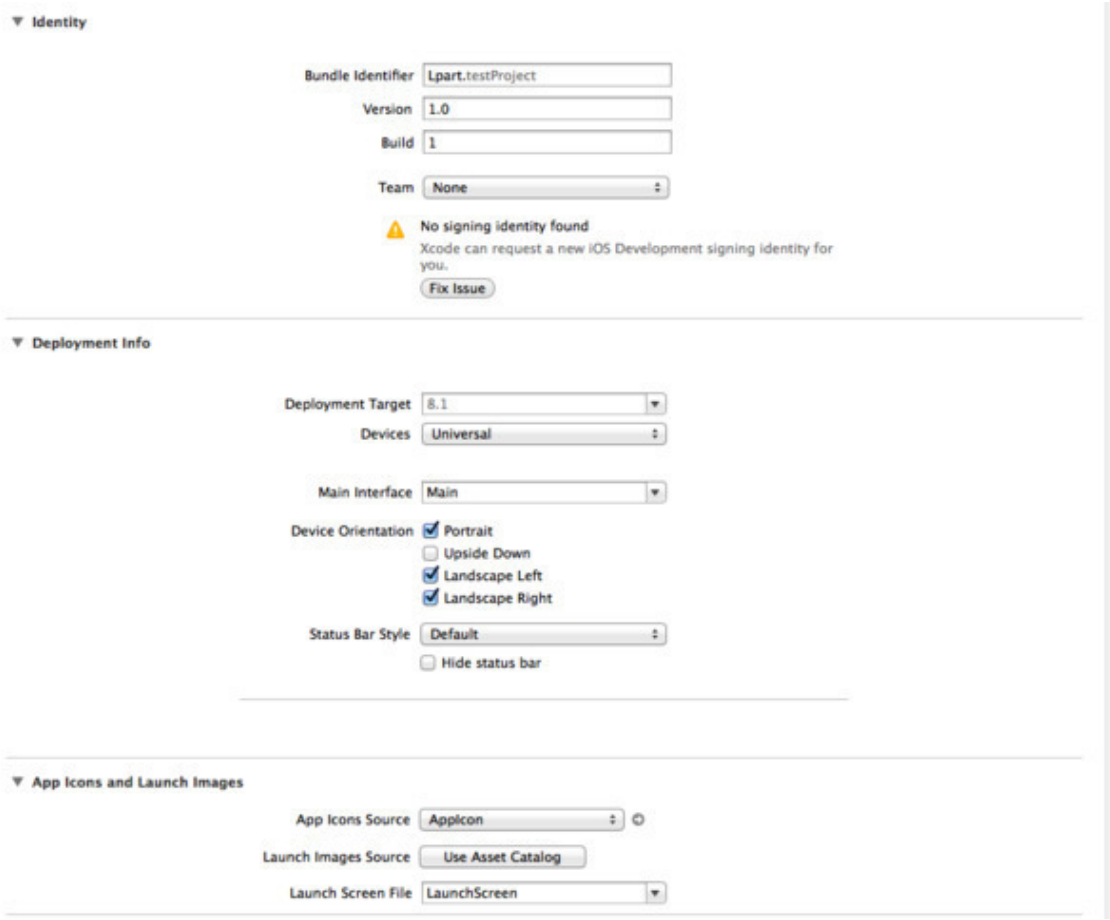



Рисунок 1.3 – Основные параметры проекта

Затем указывается галерея (рис. 1.4) иконок и загрузочных экранов приложения.

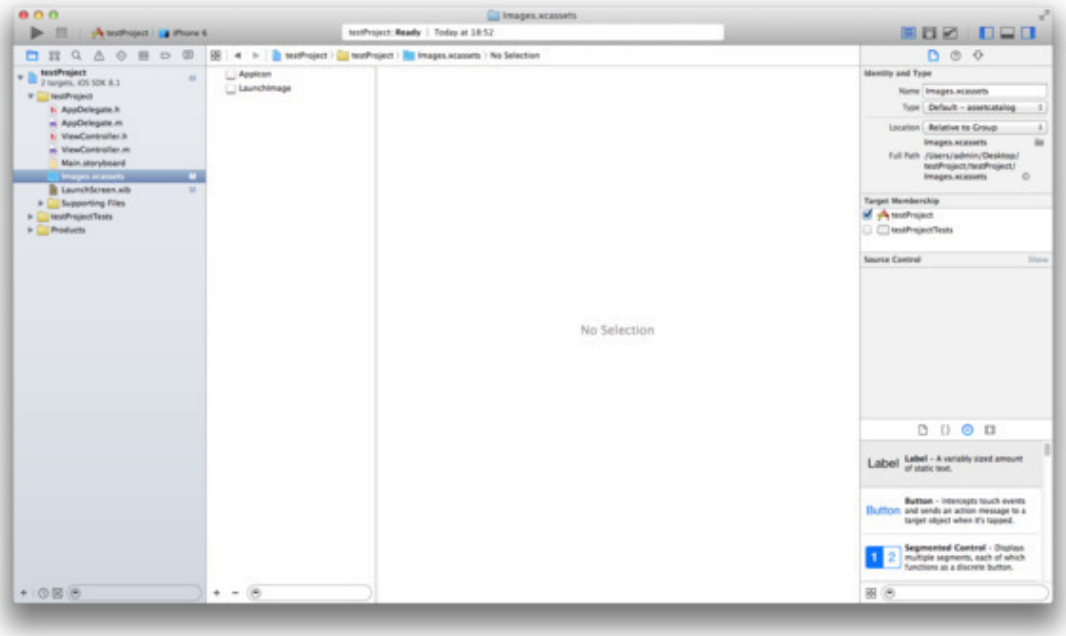



Рисунок 1.4 – Images. xcassets

Загрузочный экран указывается не изображение, а xib-файл (рис. 1.5), в котором у нас находится один объект UIKit – View, попозже разберемся что это такое. Какая разница между xib и storyboard, второе используется в версиях iOS 5 и позднее, xib является устаревшим механизмом разработки интерфейса программы, еще одно отличие – это то, что значительно упростилась работа по созданию интерфейса, различие в представление иерархии объектов, также значительно сократилось кол-во строчек xml-кода.

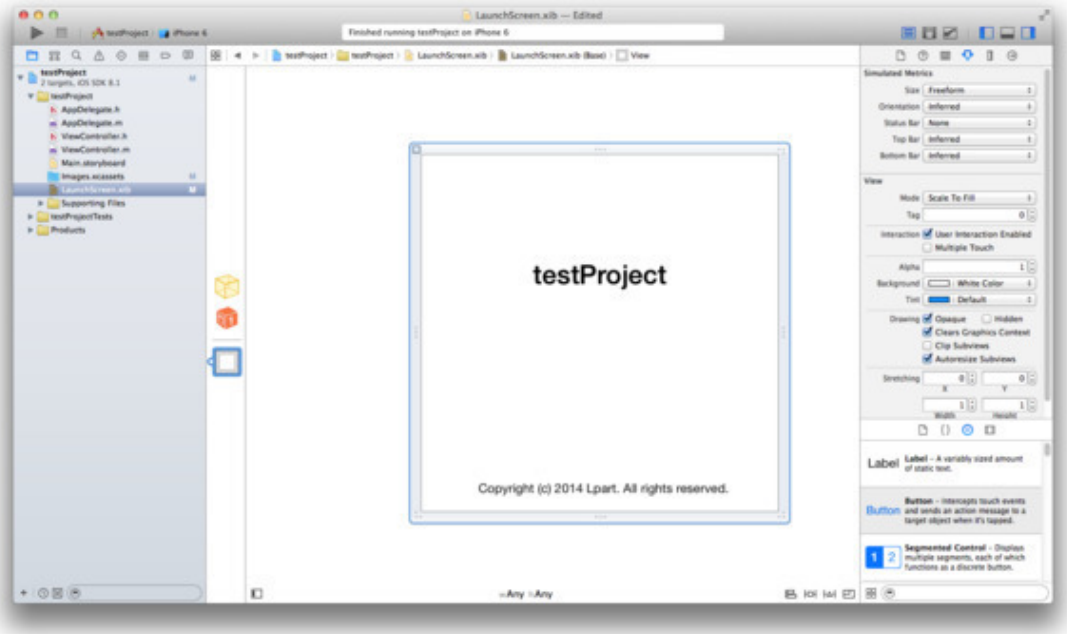



Рисунок 1.5 – Launch screen

Вернемся к настройкам проекта, рассмотрим еще параметр info (рис. 1.6). В нем содержаться основные данные о проекте, которые хранятся в файле "Info.plist», в нем мы можем указывать параметры нашего проекта. Вкладка Capabilities, содержит возможности приложения, такие как: Game Center, iCloud, Maps и т. д. В этой кладке мы подключаем дополнительные возможности. Далее Build Settings – настройка сборки приложения. Build Phases – позволяет нам настраивать сборку приложения для текущего target. Build Rules – позволяют нам задавать правила сборки приложения, к примеру, в момент сборки, сжимать текстовые файлы с помощью определенных скриптов.

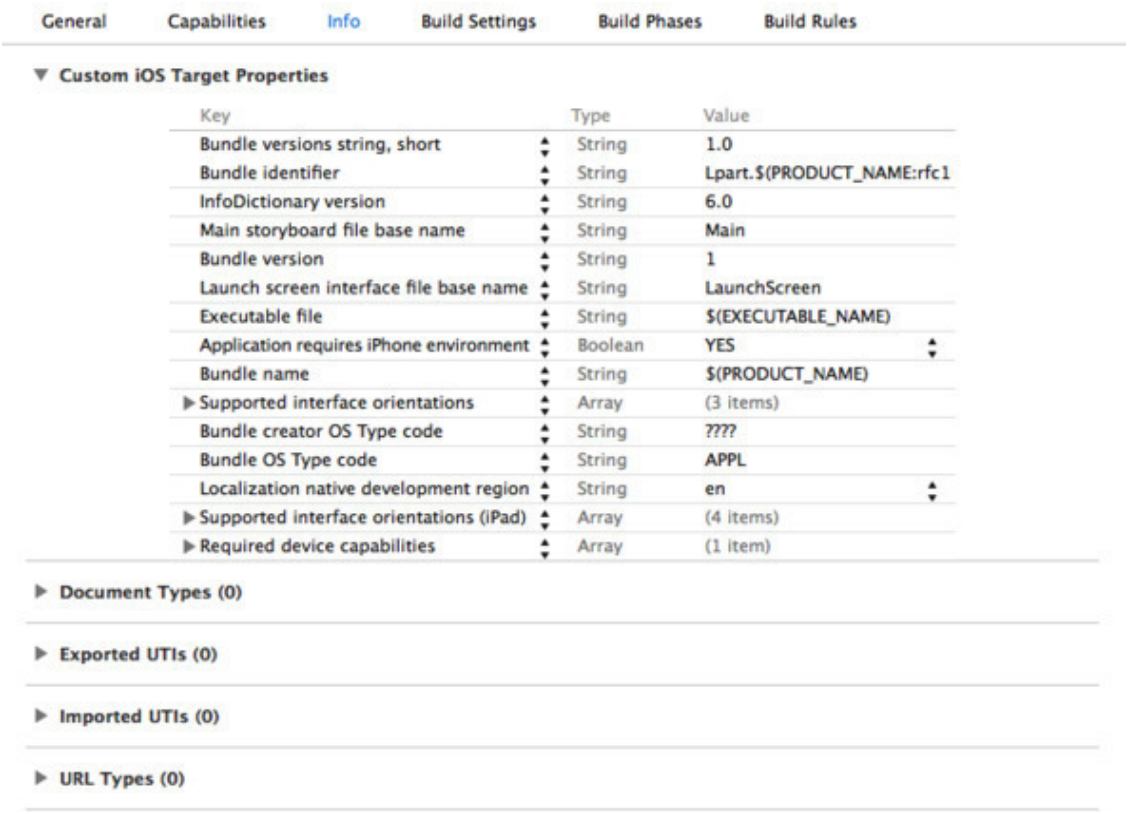



Рисунок 1.6 – Информация о проекте

Теперь перейдем в Storyboard. Как уже упоминалось ранее, это есть механизм разработки интерфейса приложения, также мы можем создавать интерфейса приложения в коде.

Перед нами должен появится View Controller (рис. 1.7) – это фундаментальный объект в UIKit, на котором можно отображать различные объекты UIKit. Сам View Controller добавлен на объект UIWindow, который поддерживает отображение графических элементов на экран. Как вы можете заметить у вью есть три элемента. Первый элемент – означает, что выбран сам вью. Второй элемент – открывает нам список готовых действий. Третий элемент – высвобождает из стека предыдущий вью, тем самым возвращает предыдущий «экран».


Рисунок 1.7 – Storyboard

Рассмотрим элементы Storyboard, в нижней части экрана (рис. 1.8) указываются размеры экранов, с которыми можно работать. Начиная с 3.5-дюймового телефона в портретной ориентации, и заканчивая 12.9-дюймовым планшетом в любой ориентации.

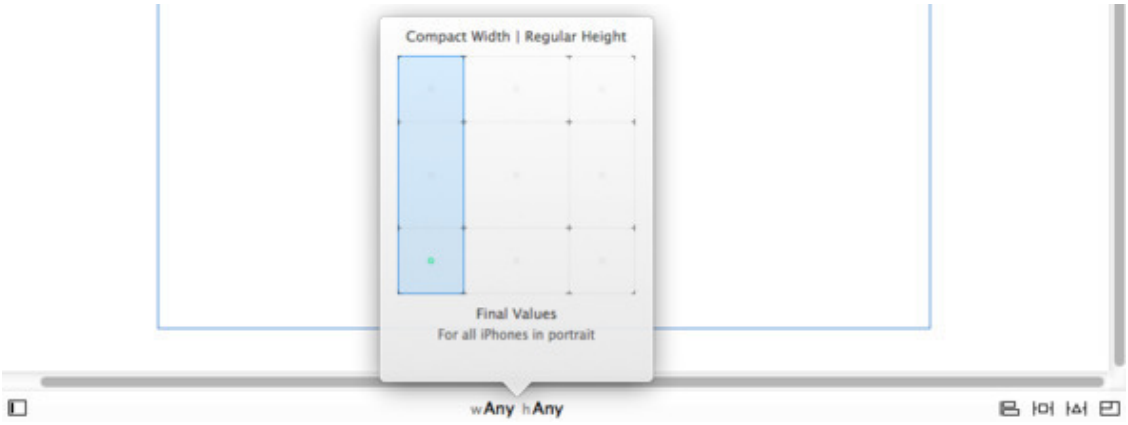



Рисунок 1.8 – Настройка представления

В левой нижней части есть кнопка, которая открывает нам иерархию объектов на View Controller, в правой нижней части, начиная от середины, первая кнопка позволяет выравнивать объекты на представление. Вторая кнопка позволяет нам закреплять объекты на определенной позиции экрана. То, что мы добавляем выравнивание или закрепляем объект – называется добавлением constraint. Третья кнопка обновляет constraint, либо изменяет положение объектов в соответствие с их constraints. Последняя кнопка обновляет constraints, если изменяются размеры экрана.

Теперь рассмотрим правый блок Xcode (рис. 1.9).

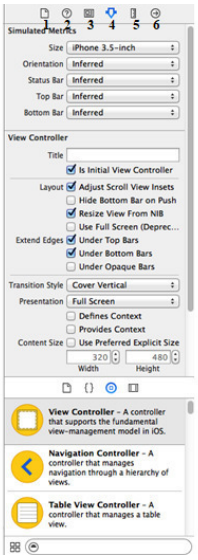



Рисунок 1.9 – Утилиты

Этот блок содержит сведения и параметры, выделенного объекта, снизу, находится библиотека объектов, список различных конструкций кода, файлов, добавляемых в проект, и медиа библиотека.

Разберем пронумерованные вкладки (рис 1.9). Первая – файловый инспектор, в котором указаны характеристики storyboard, такие как: где расположен файл, под какую версию разрабатывается и т. д. Вторая – быстрый помощник. Третья – идентификатор выбранного объекта, в нем задается идентификатор объекту, указывается класс. Четвертая вкладка – параметры объекта. В пятой вкладке, настраиваются размеры объекта. Шестая – показывается существующие связи между объектами.

Рассмотри навигацию проекта (рис. 1.10). Первая вкладка – файловый инспектор проекта. Вторая – показывает нам классы, а при их раскрытие, показывает методы классов. Третья – поисковая строка, ищет во всем проекте, также можно искать и в самом классе (cmd+f). Четвертая – показывает список предупреждений и ошибок, если таковые имеются. Пятая – список существующих тестов для приложения. Шестая вкладка – показывает нагрузку на процессор, память, сеть, файловую систему телефона. Седьмая – список точек остановок. Восьмая – показывается выполненные действия над проектом.

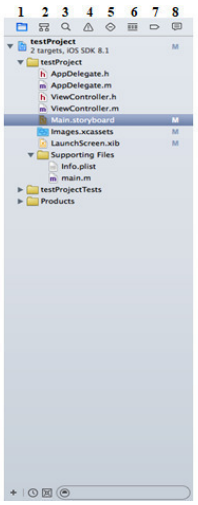



Рисунок 1.10 – Навигация проекта

Теперь добавим кнопку в наш проект. Для этого, нам необходимо найти библиотеку объектов, и в ней найти Button (рис 1.11).

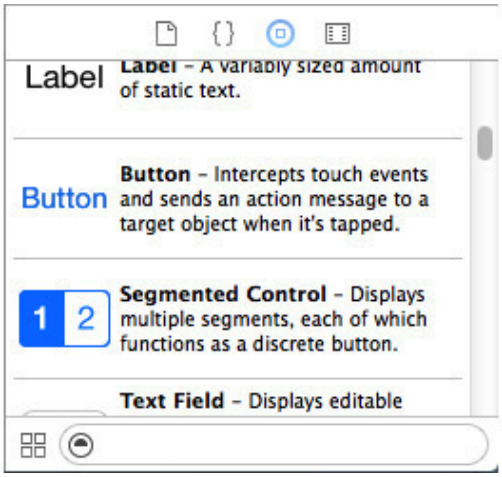



Рисунок 1.11 – Button

Затем, с помощью метода «drag and drop», добавляем на наш View Controller. Затем таким же способом добавим Label. Обоям объектам установить размер 200x40, в инспекторе размера. А нашему вью в инспекторе параметров, установите для size значение – iPhone 3.5 inch.

Теперь мы должны установить связи, между кнопкой и вью, и между надписью и вью. Нам нужно разделить экран Xcode на две части, нажмем на соответствующую кнопку в правом верхнем углу (рис. 1.12).

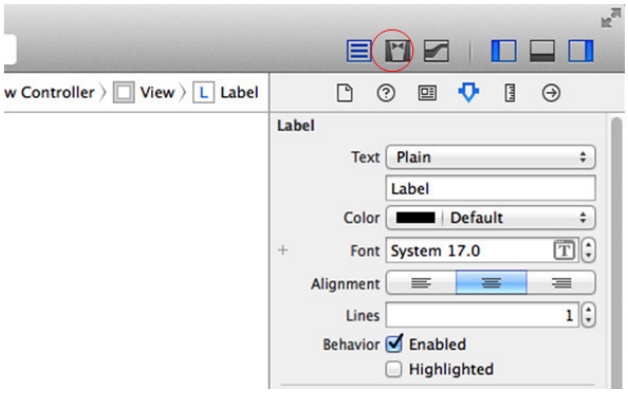



Рисунок 1.12 – Assistant Editor

Теперь у нас есть два окна, в левой части storyboard, в правой части какой-либо класс. Теперь в правую часть поместим ViewController. h, из файлового инспектора перетянем этот класс в правое окно (рис. 1.13).

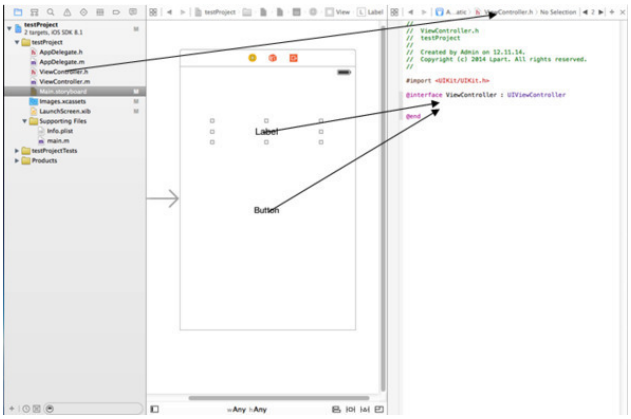



Рисунок 1.13 – Assistant Editor II

И добавим связи в представление вью. Правой кнопкой мыши перетяните стрелку с Label и Button в ViewController. h. Перед вами появится окно с параметрами (рис. 1.14), с ними мы разберемся позже, а пока что введите имя надписи.

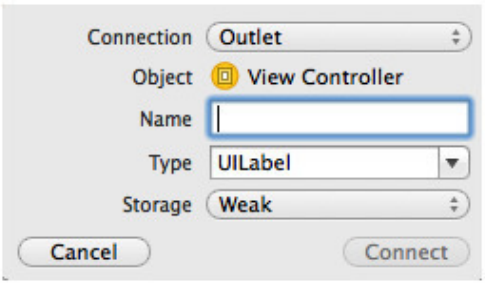



Рисунок 1.14 – Параметры свойства

Для Button нужно будет дважды проделать такое действие, первое мы добавляем свойство для кнопки, а второе мы добавляем для него метод, чтобы добавить метод, нужно указать в параметре connection – action (рис 1.15).

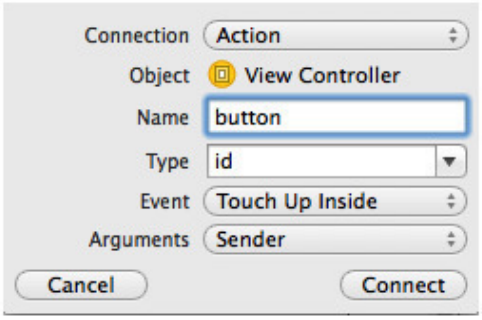



Рисунок 1.15 – Action button

Должно получится следующим образом (рис. 1.16).

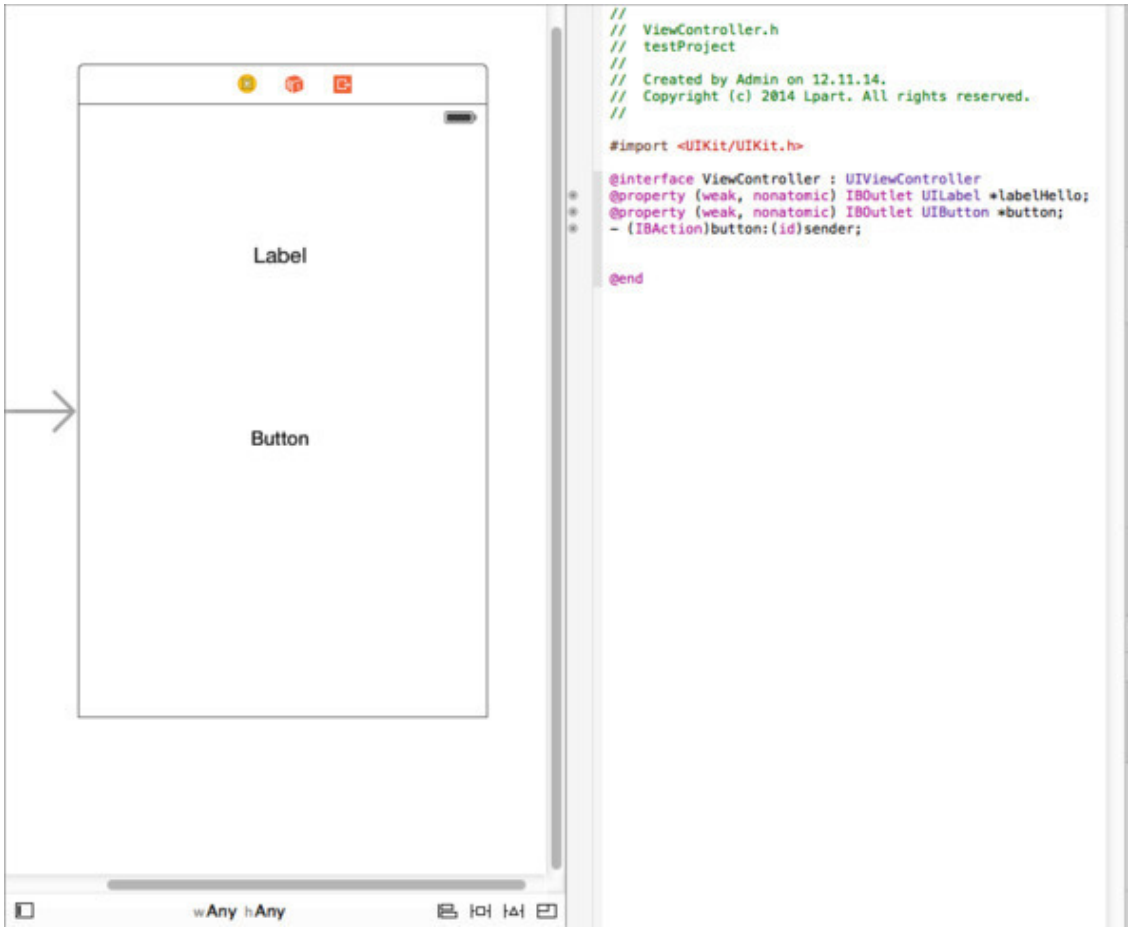



Рисунок 1.16 – Связь объектов и вью

Теперь закроем второе окно и перейдем в класс ViewController. m. Найдем в нем метод – (IBAction) button: (id) sender. Это метод кнопки, который вызывается каждый раз, когда мы нажимаем на кнопку. Пропишем в этом методе следующий код:

– (IBAction) button: (id) sender {
_labelHello. text = @«Hello World!»;
}

Теперь, когда будете нажимать на кнопку, надпись будет менять свой текст на «Hello World», результат работы – рисунок 1.17.

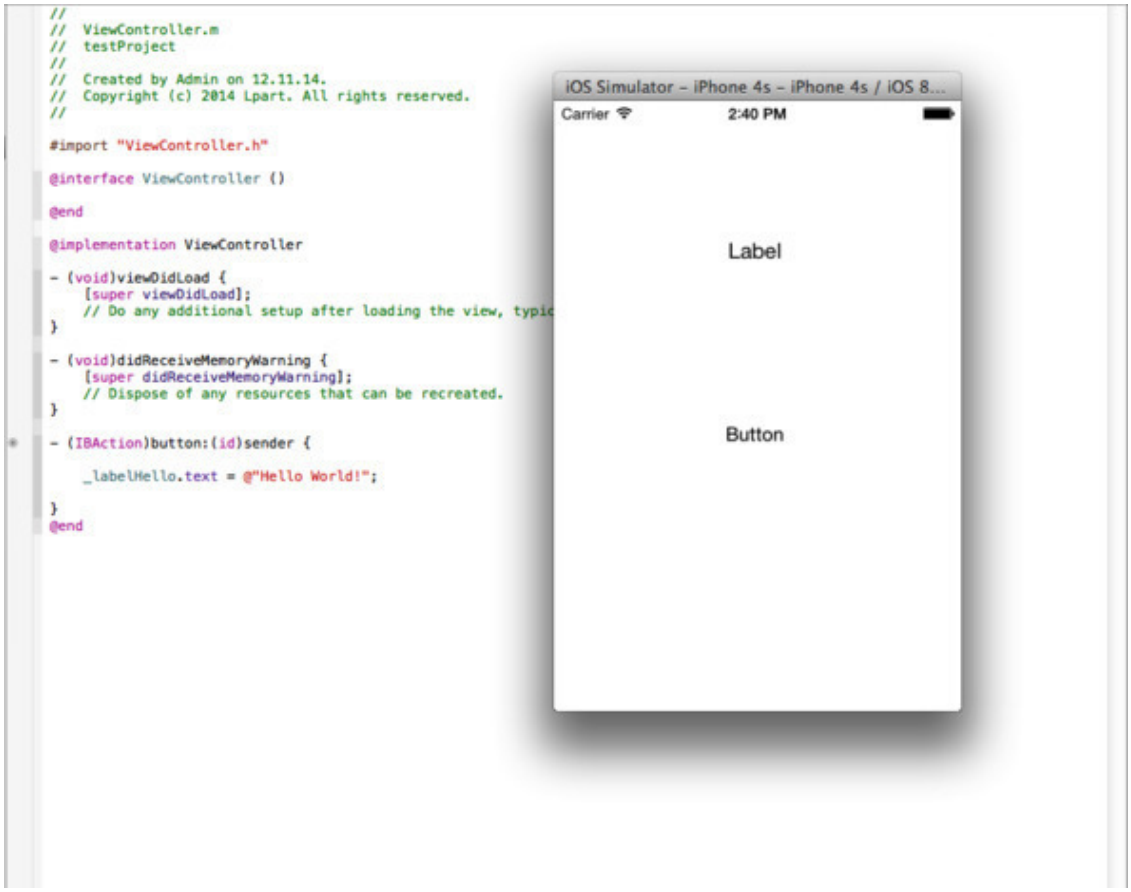



Рисунок 1.17 – Результат работы

И последнее, что осталось – отладчик. Чтобы поставить точку остановки, необходимо нажать в выделенной полосе слева от кода (рис. 1.18).

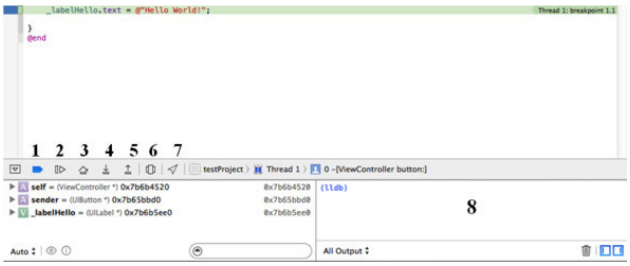



Рисунок 1.18 – Отладчик

Первая кнопка – отключает/включает точки остановки. Вторая – продолжить/приостановить процесс работы приложения. Третья – пошаговая отладка кода. Четвертая кнопка – заход в метод. Пятая – выход из метода. Шестая – позволяет нам просматривать наше приложение в виде 3D-модели. Седьмая кнопка – задает местонахождение. Окно под номером 8 – консоль, в которой выводятся системные сообщения.

# Основы графического дизайна интерфейсов приложений

«Быть дизайнером – значит не просто собирать разрозненные элементы воедино, упорядочивать их или как-то изменять. Тут нужно и создавать некую ценность, и придавать смысл, и освещать, и упрощать, и трансформировать, и облагораживать, и сгущать краски, и убеждать, и даже в какой-то мере развлекать».

– Пол Рэнд (Paul Rand)

Перед тем как начать разрабатывать интерфейс приложения, следует разобраться с его основными задачами. Грамотно созданная графическая составляющая приложения должна способствовать эффективной работе, облегчать взаимодействие пользователя с ПО. Исходя из этого, сформулируем 20 основных тезисов, которые должен знать каждый дизайнер:

1.Обязанность интерфейса – обеспечение взаимодействия.

Интерфейсы служат для обеспечения взаимодействия между людьми и окружающим миром. Они помогают нам прояснять, освещать, реализовывать и наблюдать взаимосвязи; они могут объединять и разъединять нас, влиять на наши ожидания; а кроме того, они дают нам доступ к различным услугам. Интерфейсы призваны выполнять определенные функции.

2. Ясность прежде всего.

Любой интерфейс в первую очередь должен быть понятным. Чтобы эффективно использовать разработанный вами интерфейс, люди должны понимать, что он из себя представляет, зачем им его использовать, какие задачи они смогут с его помощью выполнять, к чему приведет то или иное действие – и тогда они смогут успешно с ним взаимодействовать. Да, разобраться в интерфейсе с первого раза может быть непросто, но двусмысленностям в нем места нет. Понятному интерфейсу пользователи доверяют и поэтому, скорее всего, будут использовать его в дальнейшем. В общем, вместо того чтобы запихивать все на один экран и тем самым запутать пользователей, лучше сделайте сотню понятных экранов.

3. Внимание любой ценой.

Внимание пользователей бесценно. Дайте ему сосредоточиться, не засоряйте приложения элементами, отвлекающими внимание от главного назначения того или иного созданного вами экрана. Если пользователям нужно что-то прочитать, то дайте им достаточно времени для этого, а уж потом показывайте рекламу, если это необходимо. Цените внимание ваших пользователей: от этого в выигрыше будут и они, и вы. Для обеспечения использования продукта внимание пользователей – жизненно важный фактор. Старайтесь удерживать его любой ценой.

4. Под контролем пользователей.

Людям нравится чувствовать контроль над ситуацией. Многие разработчики об этом не задумываются, и в результате пользователи вопреки своему желанию вынуждены совершать операции, которые они не собирались совершать, причем направление их движения оказывается весьма неясным, а результаты действий – неожиданными. Дайте пользователям почувствовать, что ситуация под их контролем, периодически отображая состояние системы, описывая причинно-следственные связи (если вы сделаете это, случится то-то) и помогая им ясно понять, чего можно ожидать от каждой конкретной операции.

# Конец ознакомительного фрагмента.