ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc452857029)

[ГЛАВА 1. Язык SWIFT 4](#_Toc452857030)

[ГЛАВА 2. интегрированная среда разработки xcode 6](#_Toc452857031)

[2.1. Общая характеристика XCODE 6](#_Toc452857032)

[2.2. Характеристика используемой версии XCODE 6](#_Toc452857033)

[ГЛАВА 3. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 8](#_Toc452857034)

[ГЛАВА 4. РАЗРАБОТКА 9](#_Toc452857035)

[4.1. openweathermap 9](#_Toc452857036)

[4.2. Создание проекта в Xcode 9](#_Toc452857037)

[4.3. Разработка интерфейса 10](#_Toc452857038)

[4.4. Работа с кодом 12](#_Toc452857039)

[4.5 Тестирование приложения на реальном устройстве 19](#_Toc452857040)

[ЗАКЛЮЧЕНИЕ 21](#_Toc452857041)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 22](#_Toc452857042)

ВВЕДЕНИЕ

Объектно-ориентированные языки программирования пользуются в последнее время большой популярностью среди программистов, так как они позволяют использовать преимущества объектно-ориентированного подхода не только на этапах проектирования и конструирования программных систем, но и на этапах их реализации, тестирования и сопровождения.

**Задачи**, которые необходимо решить:

1. Проанализировать объектно-ориентированное программирование в целом
2. Исследовать объектно-ориентированное программирование в Swift
3. Создание мобильного приложения в XCODE на языке Swift

**Актуальность** данной работы заключается в том, что появление нового языка Swift позволяет существенно упростить разработку приложений iOS и OSX за счет упрощенного синтаксиса. Этот язык, сохранив средства ставшего общепризнанным стандартом для написания системных и прикладных программ для iOS и OSX языка Objective-С, ввел в практику программирования возможности нового технологического подхода к разработке программного обеспечения.

# ГЛАВА 1. Язык SWIFT

Swift - это новый язык программирования для разработки приложений на iOS, OSX, watchOS, tvOS, который сочетает в себе все лучшее от C и Objective-C, при этом нет ограничений в совместимости с C. Swift использует паттерны безопасного программирования и содержит современные функции, которые помогают сделать программирование легким, гибким и увлекательным.

Swift разрабатывался несколько лет. Apple заложила в основу Swift существующий компилятор, отладчик и структуру фреймворков. Упростили процесс управления памятью с системой автоматического подсчета ссылок - Automatic Reference Counting (ARC). Фреймворки, основанные на Foundation и Cocoa, также были модернизированы и стандартизированы. Objective-C начал поддерживать блоки, коллекции литералов, и модули, включая заимствование новых возможностей языка фреймворком, без каких-либо проблем. Благодаря этой проделанной работе нам представили новый язык для будущих разработок приложений Apple.

Swift покажется знакомым для разработчиков Objective-C. Он заимствует читабельность именованных параметров Objective-C и мощь динамической модели объектов Objective-C. Он обеспечивает плавный доступ к существующим фреймворкам Cocoa и возможность смешивать код с кодом Objective-C. Построенный на этой общей основе, Swift предоставляет много новых возможностей и унифицирует процедурный и объектно-ориентированный части языка.

Swift дружелюбен для новичков в программировании. Это первый язык программирования промышленного качества, который так же понятен и увлекателен, как скриптовый язык. Он поддерживает playground’ы - инновационная функция, которая позволяет экспериментировать с кодом Swift и видеть результат мгновенно, без необходимости компилировать и запускать приложение.

Swift вобрал в себя лучшие идеи современных языков с мудростью инженерной культуры Apple.

Компилятор оптимизирован для производительности, а язык оптимизирован для разработки, без компромиссов с одной или другой стороны. Он спроектирован для разработки приложений начиная от “Hello, world”, заканчивая масштабами операционной системы. Все это делает Swift перспективным инструментом для разработчиков и самой компании Apple.

Swift - это фантастический способ писать приложения на iOS, OSX, watchOS, tvOS и продолжать знакомиться с новыми функциями и возможностями[1].

# ГЛАВА 2. интегрированная среда разработки xcode

## 2.1. Общая характеристика XCODE

Xcode — интегрированная среда разработки программного обеспечения под OSX, iOS, watchOS и tvOS разработанная корпорацией Apple. Первая версия выпущена в 2001 году. Стабильные версии распространяется бесплатно через Mac App Store. Зарегистрированные разработчики также имеют доступ к бета-сборкам через сайт Apple Developer.

Xcode включает в себя большую часть документации разработчика от Apple и Interface Builder — приложение, использующееся для создания графических интерфейсов.

Пакет Xcode включает в себя изменённую версию свободного набора компиляторов GNU Compiler Collection и поддерживает языки C, C++, Objective-C, Swift, Java, AppleScript, Python и Ruby с различными моделями программирования, включая (но не ограничиваясь) Cocoa, Carbon и Java. Сторонними разработчиками реализована поддержка GNU Pascal, Free Pascal, Ada, C#, Perl, Haskell. Пакет Xcode использует GDB в качестве back-end’а для своего отладчика.

В августе 2006 Apple объявила о том, что DTrace, фреймворк динамической трассировки от Sun Microsystems, выпущенный как часть OpenSolaris, будет интегрирован в Xcode под названием Xray. Позже Xray был переименован в Instruments[2].

## 2.2. Характеристика используемой версии XCODE

Для выполнения курсовой работы используется последняя версия Xcode 7.3.1 build 7D1014, доступная для всех в Mac App Store или с сайта разработчиков Apple Developer.

Используемый формат проекта – xcodeproj или xcworkspace (в том случае, если в проект интегрированы сторонние библиотеки, например, cocoapods.

Основные преимущества Xcode 7.3:

- система кода завершения стала более "умной" и срабатывает по первой букве слова.

- возможность переключения между окнами двух часов Apple Watch, подключённых к одному iPhone.

- добавлены новые иконки уведомлений и других активностей.

- исправлена ошибка, когда не удавалось провести тест UI на tvOS.

- статический анализатор теперь может обнаружить больше потенциальных проблем.

- встроенный отладчик больше не будет автоматически оценивать структуру кода.

ГЛАВА 3. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

Цель:

Требуется создать мобильное приложение на языке Swift для платформы iOS.

Задачи:

1. Поддерживаемое устройство: iPhone
2. Используемая версия iOS: 9.3.
3. Использование только портретной ориентации.
4. Использование API openwethermap.
5. Реализация геолокации.
6. Реализация поиска по городу.
7. Вывод информации о погоде пользователю.

ГЛАВА 4. РАЗРАБОТКА

## 4.1. openweathermap

Для реализации приложения берем информацию с openweathermap

(рис 4.1).

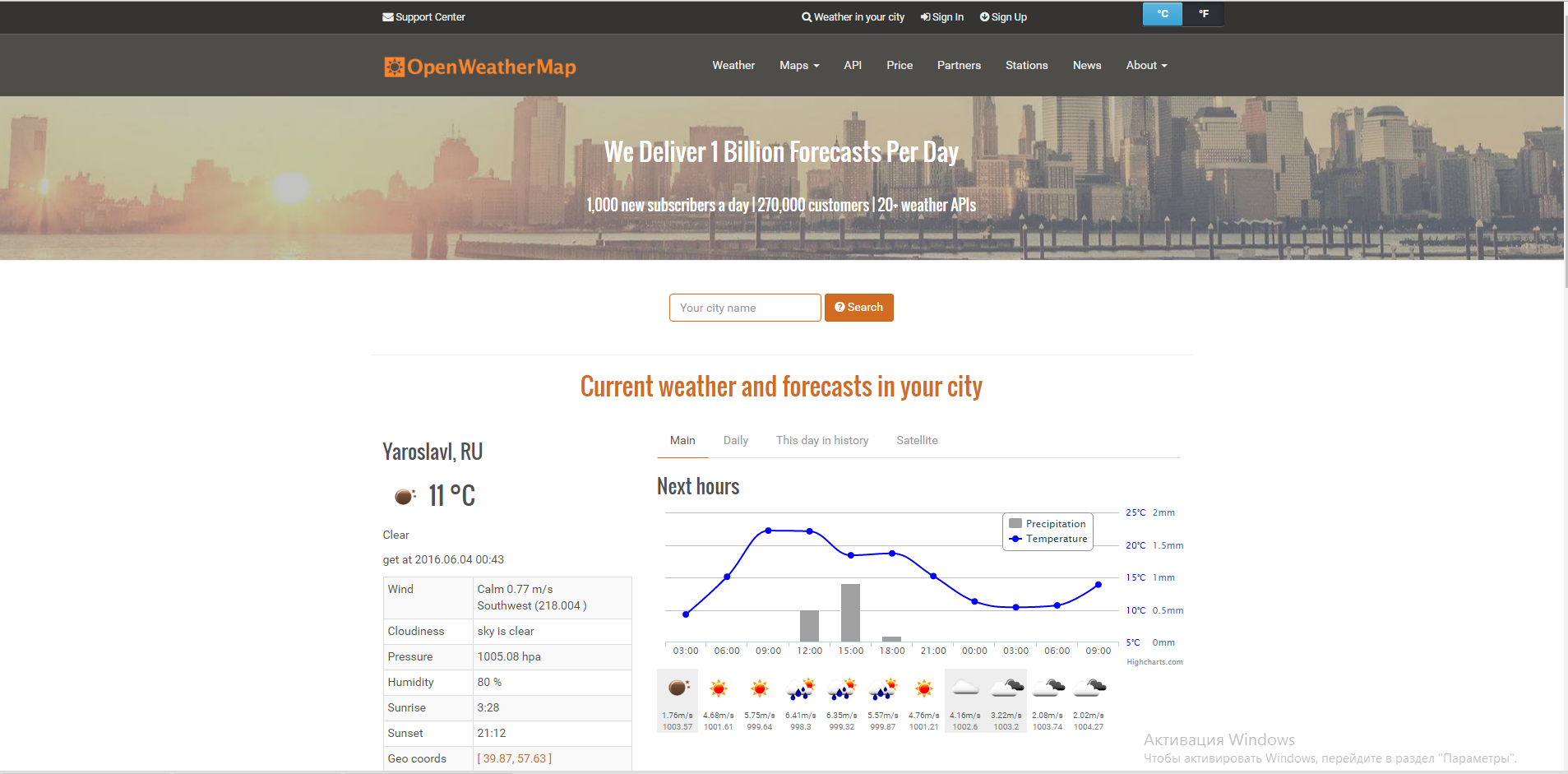


Рис 4.1 openweathermap

## 4.2. Создание проекта в Xcode

Необходимо создать проект с использованием языка Swift. Создание Xcode проекта (рис 4.2).

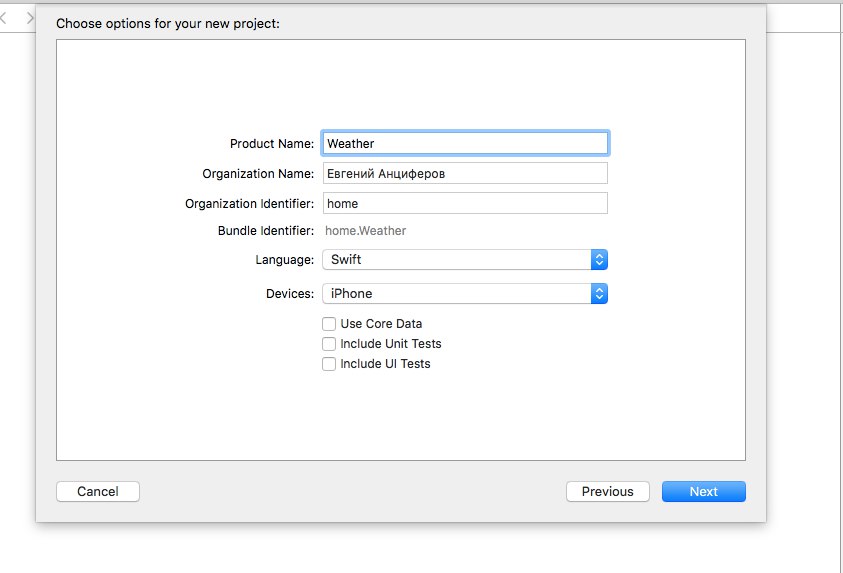


Рис 4.2 Создание Xcode проекта

Настройка проекта по требованиям технического задания. Указываем версию iOS, устройство, для которого будет разрабатываться приложение и ориентацию (рис 4.3).

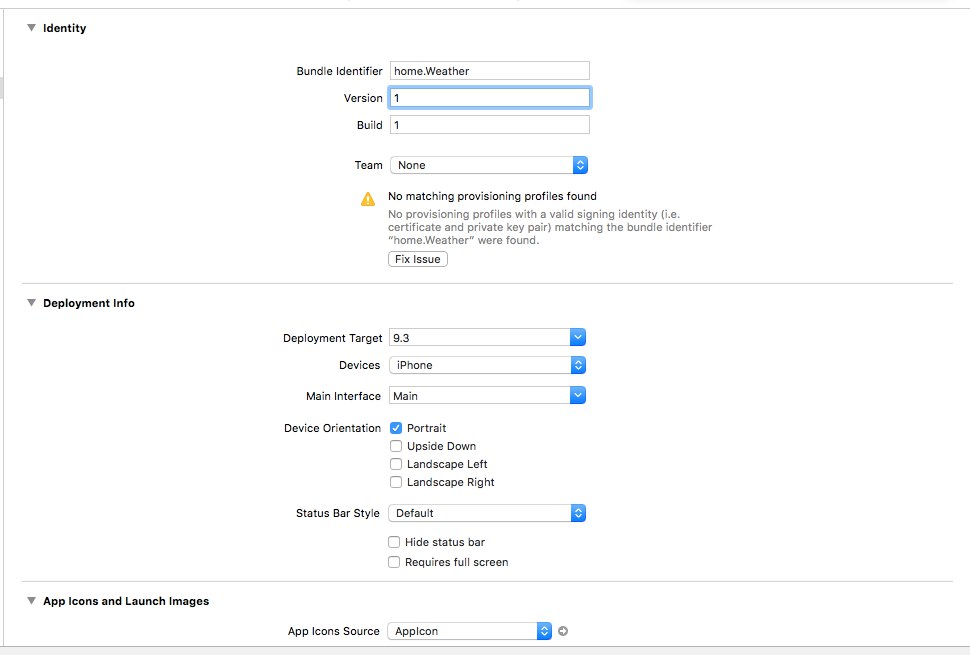


Рис 4.3 Настройка проекта

## 4.3. Разработка интерфейса

Необходимо создать ViewController с реализацией AutoLayout. Внешний вид интерфейса изображен на рис 4.4.

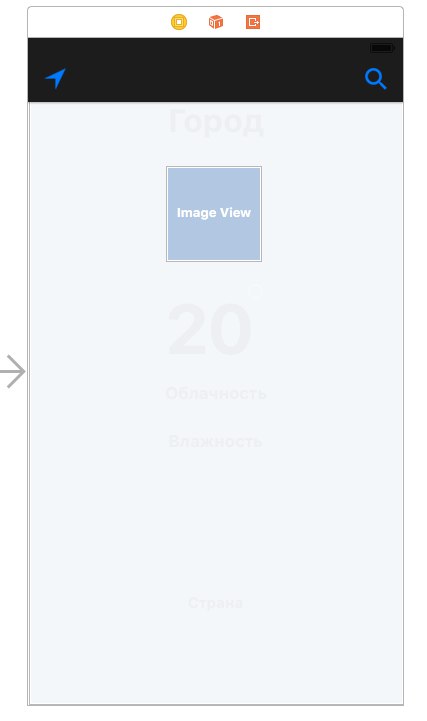


Рис 4.4 Внешний вид интерфейса

После создания контроллеров необходимо реализовать autolayout для правильного отображения элементов на устройствах разных диагоналей экрана.

Настройка constraints (ограничения) изображена на рис4.5. Аналогично настраивается для всех контроллеров.

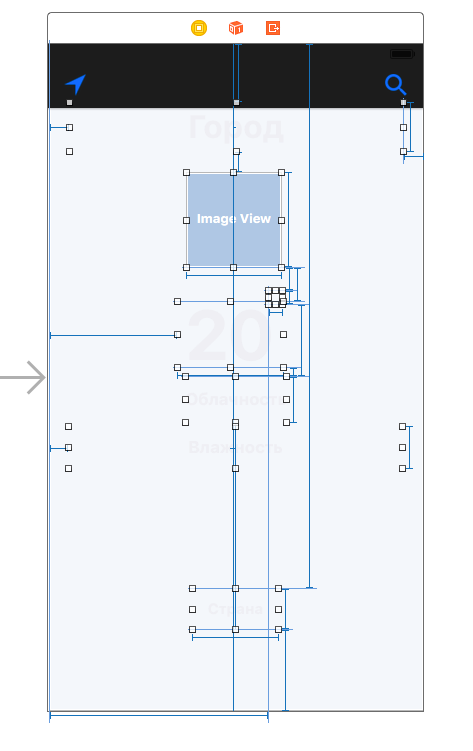


Рис 4.5 AutoLayout

## 4.4. Работа с кодом

Для начала нам необходимо создать приватную константу, выполняющую основной запрос к API.

private let openWeatherMapBaseURL = "http://api.openweathermap.org/data/2.5/weather"

Создаем приватную константу, хранящую API ключ приложения.

private let openWeatherMapAPIKey = "fa3d632fc21c3dc944b6e5aabce59b72"

Создаем четыре переменные: icon для хранения изображения о состоянии погоды, backgroundColor для хранения изображений background'a, меняющиеся в соответствии о состоянии погоды, refreshControl для обновления состояния погоды, locationManager для использования геолокации.

var icon: UIImage?

var backgroundColor: String?

var refreshControl = UIRefreshControl()

var locationManager: CLLocationManager = CLLocationManager()

Объявляем функцию, которая будет считывать геолокацию один раз, после первого её получения останавливаться. Из данных геолокации используем только географические координаты широты и долготы.

func locationManager(manager: CLLocationManager, didUpdateLocations locations: [CLLocation]) {

let currentLocation = locations.last! as CLLocation

if (currentLocation.horizontalAccuracy > 0) {

locationManager.stopUpdatingLocation()

let coords = CLLocationCoordinate2DMake(currentLocation.coordinate.latitude, currentLocation.coordinate.longitude)

print(coords)

self.weatherFor(coords)

}

}

При ошибке геолокации или ее отсутствии или при запрете на использование пользователем выводится стандартная информация о погоде по городу Москва

func locationManager(manager: CLLocationManager, didFailWithError error: NSError) {

print(error.localizedDescription)

getWeather("Moscow")

}

Функция, которая получает данные с openWeatherMap по названию города.

func getWeather(city: String) {

let session = NSURLSession.sharedSession()

let weatherRequestURL = NSURL(string: "\(openWeatherMapBaseURL)?APPID=\(openWeatherMapAPIKey)&q=\(city)")!

let dataTask = session.dataTaskWithURL(weatherRequestURL) {

(data: NSData?, response: NSURLResponse?, error: NSError?) in

if let error = error {

print("Error:\n\(error)")

}

else {

do {

let weather = try NSJSONSerialization.JSONObjectWithData(

data!,

options: .MutableContainers) as! [String: AnyObject]

catch let jsonError as NSError {

print("JSON error \(jsonError.description)")

}

}

}

dataTask.resume()

}

Реализуем функцию обновления. Обновляется по cityLabel - название города в интерфейсе.

func refresh(sender:AnyObject) {

self.getWeather(cityLabel.text!.stringByReplacingOccurrencesOfString(" ", withString: ""))

self.refreshControl.endRefreshing()

}

Реализуем функцию, которая получает данные с openWeatherMap по геолокации.

func weatherFor(geo: CLLocationCoordinate2D) {

let session = NSURLSession.sharedSession()

let weatherRequestURL = NSURL(string: "http://api.openweathermap.org/data/2.5/weather?lat=\(geo.latitude)&lon=\(geo.longitude)&appid=fa3d632fc21c3dc944b6e5aabce59b72")

let dataTask = session.dataTaskWithURL(weatherRequestURL!) {

(data: NSData?, response: NSURLResponse?, error: NSError?) in

if let error = error {

print("Error:\n\(error)")

} else {

do {

let weather = try NSJSONSerialization.JSONObjectWithData(

data!,

options: .MutableContainers) as! [String: AnyObject]

Выводим полученную информацию на экран.

self.cityLabel.text = String(weather["name"]!)

let temp = weather["main"]!["temp"]!! as! Int

self.temp.text = String(temp - 273)

self.cloudLabel.text = String("\(weather["weather"]![0]!["description"]!!)")

self.countryLabel.text = String(weather["sys"]!["country"]!!)

self.humidity.text = String("Влажность: \(weather["main"]!["humidity"]!!)%")

let srtIcon = weather["weather"]![0]!["icon"]!!

self.icon = self.weatherIcon(srtIcon as! String)

self.imageView.image = self.icon

let strBackground = weather["weather"]![0]!["icon"]!!

self.backgroundColor = String(self.backgroundImage(strBackground as! String))

self.view.backgroundColor = UIColor(patternImage: UIImage(named: self.backgroundColor!)!)

Реализуем поиск по городу при помощи barButtonItem

@IBAction func searchButton(sender: AnyObject) {

let alert = SCLAlertView()

let txt = alert.addTextField("Введите город")

txt.textAlignment = .Center

txt.keyboardType = .ASCIICapable

alert.addButton("OK") {

if txt.text != "" {

self.getWeather(txt.text!.stringByReplacingOccurrencesOfString(" ", withString: ""))

}

}

alert.showTitle("Введите город", subTitle: "", style: .Success, closeButtonTitle: "Отмена", duration: 0, colorStyle: 202020, colorTextButton: 0xFFFFFF)

Реализуем кнопку, которая обновляет информацию о геолокации.

@IBAction func checkGeo(sender: AnyObject) {

locationManager.desiredAccuracy = kCLLocationAccuracyBest

locationManager.delegate = self

locationManager.requestWhenInUseAuthorization()

locationManager.startUpdatingLocation()

}

Создаем класс Reachability.swift и выполняем проверку на наличие интернет - соединения.

public class Reachability {

class func isConnectedToNetwork()->Bool {

var zeroAddress = sockaddr\_in()

zeroAddress.sin\_len = UInt8(sizeofValue(zeroAddress))

zeroAddress.sin\_family = sa\_family\_t(AF\_INET)

let defaultRouteReachability = withUnsafePointer(&zeroAddress) {

SCNetworkReachabilityCreateWithAddress(nil, UnsafePointer($0))

}

var flags = SCNetworkReachabilityFlags()

if !SCNetworkReachabilityGetFlags(defaultRouteReachability!, &flags) {

return false

}

let isReachable = (flags.rawValue & UInt32(kSCNetworkFlagsReachable)) != 0

let needsConnection = (flags.rawValue & UInt32(kSCNetworkFlagsConnectionRequired)) != 0

return (isReachable && !needsConnection)

}

}

## 4.5 Тестирование приложения на реальном устройстве

Скомпилируем приложение на реальное устройство. Для тестирования используется iPhone 5S с установленной iOS 9.3.2

Внешний вид контроллера приложения изображен на рис. 4.6.

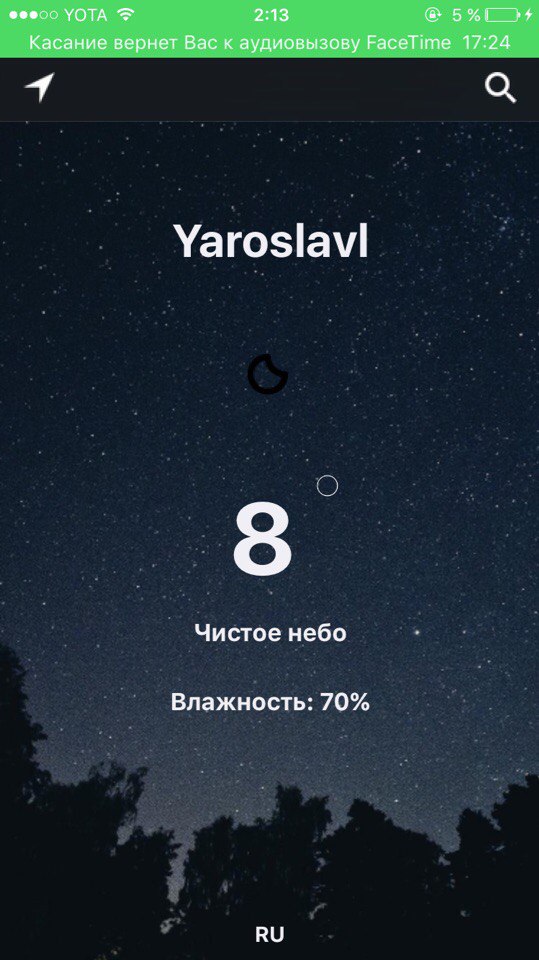


Рис 4.6 Внешний вид приложения

Тестирование возможности поиска по городу на рис 4.7.

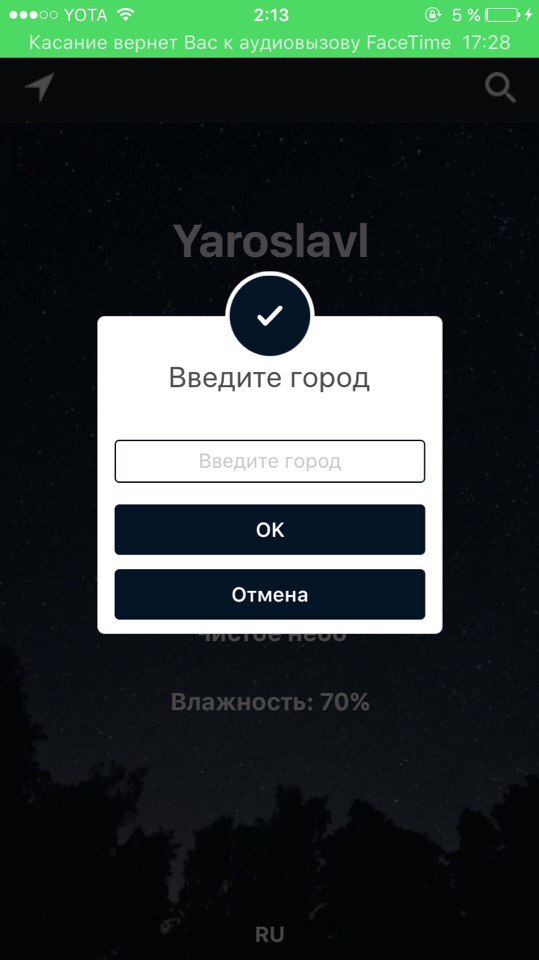


Рис 4.7 Поиск по городу

ЗАКЛЮЧЕНИЕ

Завершая работу, следует отметить, что все задания работы были успешно выполнены.

Так в ходе работы было выполнено следующее:

1. Выполнен обзор возможностей Xcode 7.3.1, использованных при разработке мобильного iOS приложения.
2. Разработано и реализовано мобильное iOS приложение “Weather”.
3. Созданное при помощи Xcode 7.3.1 мобильное приложение позволяет узнать всю необходимую информацию о состоянии погоды.
4. Ссылка на Github: https://github.com/ISUCT/OOPCoursework\_2016/tree/Antsiferov\_E\_A

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. SwiftBook [Электронный ресурс] - http://swiftbook.ru/doc/about-swift
2. Википедия [Электронный ресурс] - https://ru.wikipedia.org/wiki/Xcode