

Министерство образования и науки Российской Федерации
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школа электроники и компьютерных наук
Кафедра «Электронные вычислительные машины»

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой
_____ Г.И. Радченко
« ____ » _____ 2018 г.

Разработка приложения под iOS для организации велоквеста

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ-090301.2018.886 ПЗ ВКР

Руководитель работы,
доцент каф. «Электронные
Вычислительные машины»
_____ К.А. Домбровский
« ____ » _____ 2018 г.

Автор работы
студент группы КЭ-484
_____ В.В. Лебедев
« ____ » _____ 2018 г.

Нормоконтролёр, ст. преп. каф.
«Электронные вычислительные
машины»
_____ В.В. Лурье
« ____ » _____ 2018 г.

АННОТАЦИЯ

Автор Лебедев В.В. Разработка приложения под iOS для организации велоквеста. – Челябинск: ФГБОУ ВПО «ЮУрГУ» (НИУ) ВШЭКН; 2018, 50 с., 27 ил., библиогр. список – 22 наим., 6 листов чертежей ф. А4.

Работа посвящена разработке мобильного приложения для организации велоквеста, работающего на платформе iOS.

Данная работа состоит из глоссария, введения, четырёх глав, заключения, библиографического списка.

В первой главе представлен обзор аналогов, общая информация о разрабатываемом программном продукте, выбор средств разработки. Во второй главе – проектирование приложения, описаны сценарии использования. В третьей главе описана реализация программного продукта, инструментарий, использованный при разработке. В четвертой главе – функциональное тестирование и результаты тестов.

					ЮУрГУ-090301.2018.886 ПЗ ВКР						
Изм.	Лист	№ докум.	Лист	Подпись	Дата	Разработка приложения под iOS для организации велоквеста			Лит.	Лист	Листов
Разраб.	В.В. Лебедев				Д					3	50
Пров.	К.А. Домбровский										
Рецензент					ФГБОУ ВПО «ЮУрГУ» (НИУ) Кафедра ЭВМ						
Н. контр.	В.В. Лурье										
Утв.	Г.И. Радченко										

ОГЛАВЛЕНИЕ

ГЛОССАРИЙ	6
ВВЕДЕНИЕ.....	7
1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ РЕШЕНИЙ	9
1.1 Обзор аналогичных проектов	9
1.1.1 Actionbound.....	9
1.1.2 Surprise Me	10
1.1.3 Street adventure.....	11
1.1.4 Art Ovrage	12
1.2 Обзор существующих решений для реализации проекта	13
1.2.1 Xamarin	13
1.2.2 Titanium.....	14
1.2.3 PhoneGap.....	14
1.2.4 Xcode.....	14
2. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ	19
2.1. Требования к приложению	19
2.2. Варианты использования	19
2.3. Архитектура приложения	21
3. РЕАЛИЗАЦИЯ.....	24
3.1. Средства реализации	24
3.2. Реализация контроллеров	24
3.3 Реализация интерфейса приложения.....	31
4. ТЕСТИРОВАНИЕ	37

4.1 Функциональное тестирование приложения	37
ЗАКЛЮЧЕНИЕ.....	47
БИБЛИОГРАФИЧЕСКИЙ СПИСОК.....	48

ГЛОССАРИЙ

1) Квест (городское ориентирование) — интеллектуально-спортивная игра в городе и за его пределами.

2) Скриншот – изображение, показывающее то, что видит пользователь на экране, получаемое устройством.

3) Кроссплатформенное приложение – программное обеспечение, которое способно работать более чем на одной аппаратной платформе и (или) операционной системе.

4) API – набор готовых классов, процедур, функций, структур и констант, предоставляемых приложением (библиотекой, сервисом) или операционной системой для использования во внешних программных продуктах.

5) SDK – набор средств разработки, который позволяет специалистам по программному обеспечению создавать приложения для определённого пакета программ, программного обеспечения базовых средств разработки, аппаратной платформы, компьютерной системы, игровых консолей, операционных систем и прочих платформ.

6) Нативное приложение – это прикладная программа, которая была разработана для использования на специфической платформе или устройстве.

7) Фреймворк – программная платформа, определяющая структуру программной системы; программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

ВВЕДЕНИЕ

В настоящее время довольно сложно представить жизнь без мобильных устройств. В течение последних десятилетий они плотно внедрились в нашу жизнь, став ее неотъемлемой частью. Мобильные устройства стали незаменимыми и полезными помощниками для людей, занятых различными родами деятельности. Сейчас телефоны это не только средство общения, но и многофункциональное устройство.

Изначально мобильные телефоны предназначались для выполнения несложных задач (например, для проверки электронной почты), сейчас же они способны выполнять задачи, которые мы привыкли выполнять с помощью персонального компьютера (например, играть в трехмерные игры, смотреть фильмы, общаться по видеосвязи). Ключевыми преимуществами мобильных устройств перед персональными компьютерами являются компактность и доступность.

На сегодняшний день можно выделить две наиболее популярные мобильные операционные системы: iOS и Android.

Для расширения возможностей мобильного устройства создаются мобильные приложения. Они находят применение в различных сферах жизнедеятельности человека:

- бизнес;
- образование;
- питание;
- здоровье;
- развлечения;
- путешествия.

В настоящее время становятся популярными квесты (реалити-квесты). Квест – это командная игра в реальной жизни, где участники решают различные

головоломки. Существуют различные квесты (например, за час найти выход из запертой комнаты или команда на велосипедах выполняет задания, отмеченные на карте). Некоторые компании проводят такие квесты в качестве рекламных акций, но организация проходит на низком уровне.

В связи с этим, актуальной является задача разработки мобильного приложения для устройств на базе операционной системы iOS, которое упростит организацию велоквеста.

Целью работы является разработка мобильного приложения «ReQuest» для устройств на платформе iOS, позволяющего организовывать спортивные мероприятия, и улучшающего удобство участников. Для работы с данными также необходим сервер, с которым будет работать мобильное приложение.

Для достижения указанной цели необходимо решить следующие задачи:

- 1) Проанализировать существующие аналоги мобильных приложений для iOS.
- 2) Рассмотреть инструменты для реализации проекта.
- 3) Спроектировать мобильное приложение.
- 4) Реализовать мобильное приложение для операционной системы iOS.
- 5) Провести тестирование разработанного приложения.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И СУЩЕСТВУЮЩИХ РЕШЕНИЙ

В данной главе рассматриваются аналогичные проекты, а также различные инструменты для реализации мобильного приложения для операционной системы iOS, существующие на сегодняшний день.

1.1 Обзор аналогичных проектов

1.1.1 Actionbound

Приложение предоставляет возможность выбора квеста, кроме того можно сортировать квесты по категориям, но нет дополнительных возможностей, таких, как построение маршрута до точки с заданием или детализация уровней сложности.

К недостаткам можно отнести:

- отсутствие корректной русскоязычной версии;
- отсутствие пояснений к заданиям;
- отсутствие навигации.



Рисунок 1 – Скриншоты экранов приложения Actionbound

1.1.2 Surprise Me

Является приложением под операционные системы iOS и Android. Приложение предоставляет возможность проходить квесты в любом месте в любое время, так же создавать свои.

К недостаткам можно отнести:

- большая часть квестов платная;
- чтобы получить полный функционал редактора необходимо купить премиум-версию;
- данное приложение больше предназначено для экскурсий.

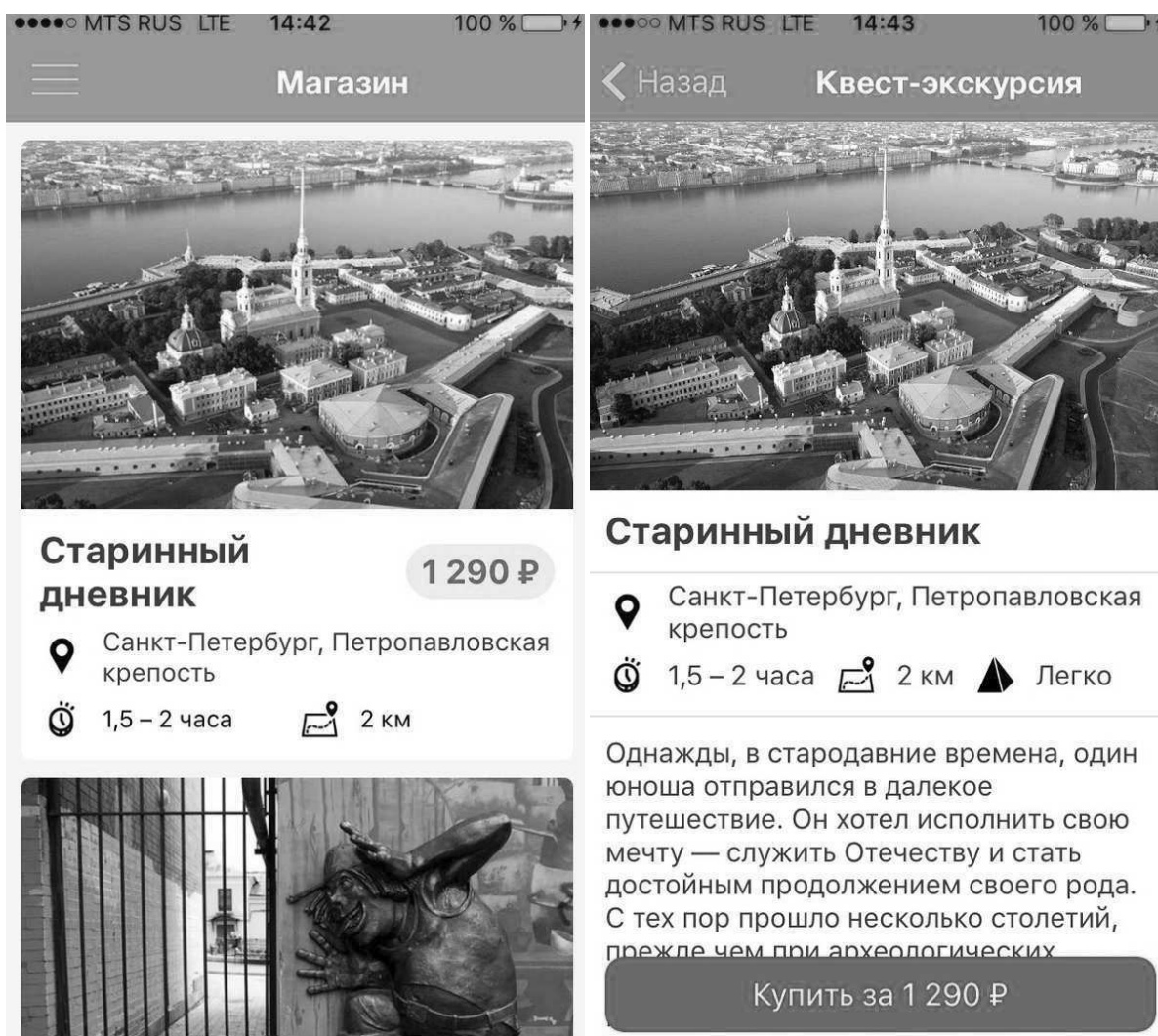


Рисунок 2 – Скриншоты экранов приложения Surprise Me

1.1.3 Street adventure

Приложение предоставляет возможность проходить городские квесты в формате экскурсий.

К недостаткам можно отнести:

- все квесты платные;
- приложение доступно только в Москве;
- данное приложение не предназначено для проведения мероприятий в форме соревнования.

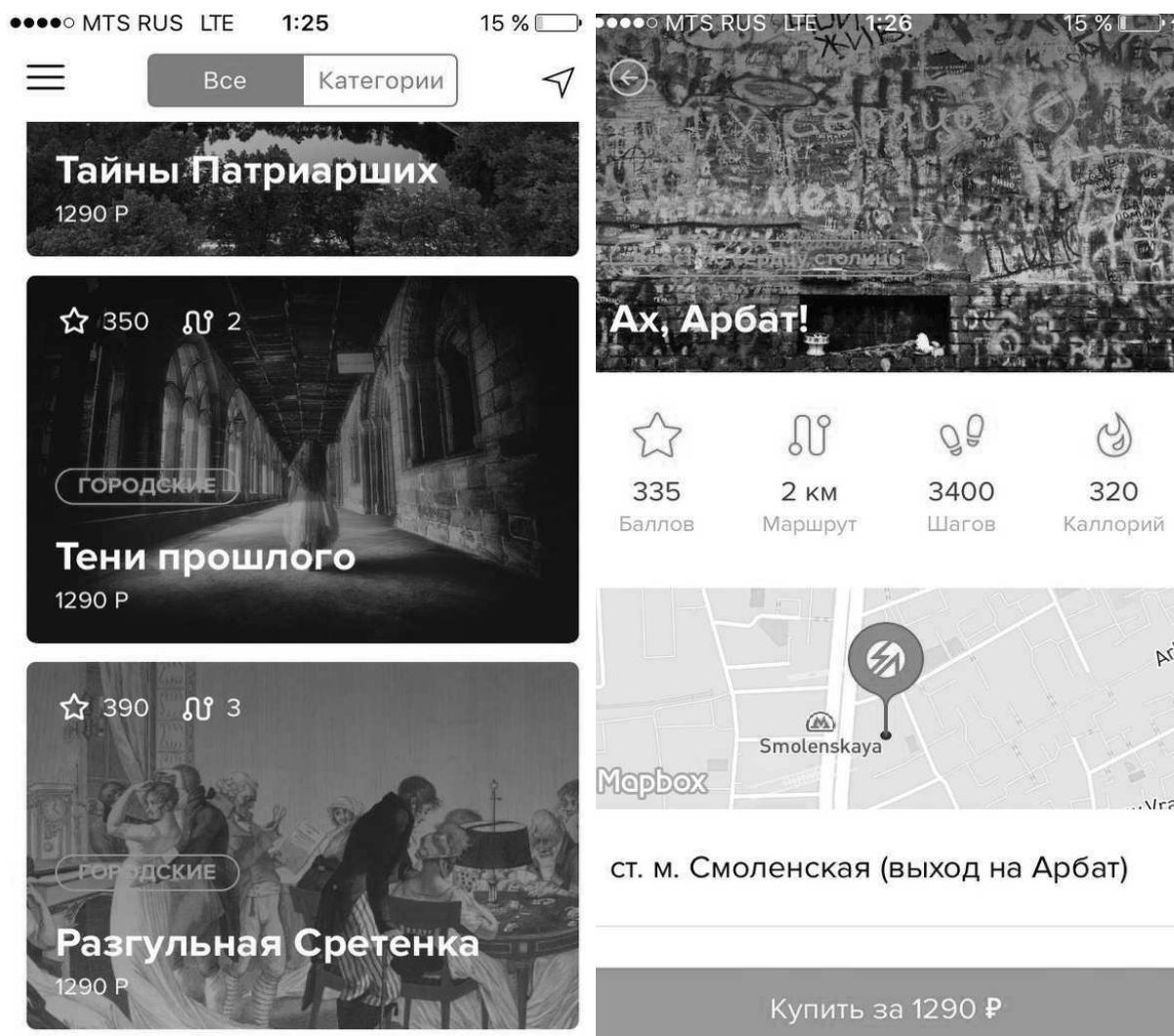


Рисунок 3 – Скриншоты экранов приложения Street Adventure

1.1.4 Art Ovrage

Приложение предоставляет возможность проходить городские квесты в формате экскурсий.

К недостаткам можно отнести:

- отсутствуют задания, данное приложение представляет собой список мероприятий проходящих в городе;
- отсутствуют карты, есть только компас;
- данное приложение не предназначено для проведения мероприятий в форме соревнования.

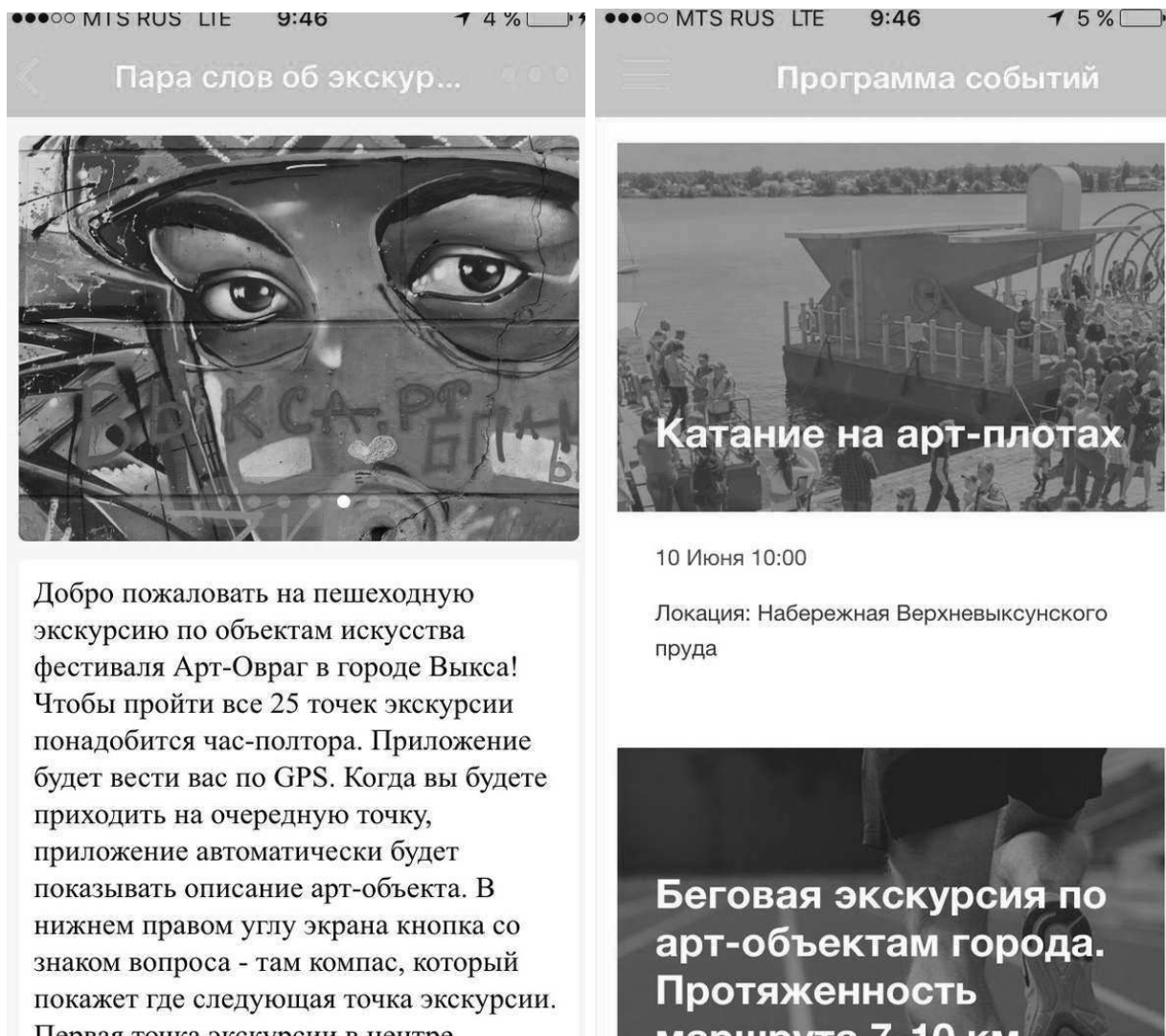


Рисунок 4 – Скриншоты экранов приложения Art Ovrage

1.2 Обзор существующих решений для реализации проекта

Для выполнения поставленной задачи были рассмотрены следующие инструменты для разработки мобильных приложений для платформы iOS:

- Xcode;
- Xamarin;
- Titanium;
- PhoneGap;

1.2.1 Xamarin

Xamarin используется для разработки на объектно-ориентированном языке C#. При использовании инструмента для Android можно получить

кроссплатформенное приложение и покрыть огромный процент мобильного рынка, разработав приложение для двух самых популярных платформ – iOS и Android. Однако разработка приложения планируется только для платформы iOS и Xamarin имеет несколько минусов:

- 1) неполноценный доступ к API.
- 2) некоторые элементы представлены как прокси к нативным, их жизненный цикл слабо связан друг с другом, возможны утечки памяти.
- 3) медленная работа компилятора.

1.2.2 Titanium

Минусы:

- 1) Высокая стоимость.
- 2) Есть задержки при запуске приложения из-за загрузки библиотеки.
- 3) Трудно создавать сложные приложения, так как использование JavaScript отрицательно сказывается на производительности приложений.

1.2.3 PhoneGap

PhoneGap использует Javascript и HTML5. Но в разрабатываемом мобильном приложении использование Flash, HTML и Javascript не является целесообразным, так как производительность приложения на PhoneGap может быть намного ниже, чем у нативного.

1.2.4 Xcode

Для реализации приложения была выбрана интегрированная среда разработки Xcode, которая является самой популярной средой для разработки приложений для мобильной платформы iOS.

Основные преимущества Xcode:

- 1) Simulator (рисунок 5) - с помощью iOS SDK XCode создавать,

устанавливать, запускать и отлаживать написанные приложения в симуляторе на базе Mac, что оптимизирует рабочий процесс разработки.

2) Source Editor (рисунок 6) – текстовый редактор, инструмент позволяющий расширение кода, свертывание кода, осуществляет подсветку синтаксиса, отображает предупреждения, ошибки и другую контекстно-зависимую информацию, встроенную в ваш код.

3) Asset Catalog (рисунок 7) - инструмент позволяющий управлять изображениями вашего приложениями, группируя разрешения одного и того же ресурса.

4) Interface Builder (рисунок 8) - инструмент позволяющий создавать и тестировать интерфейс без написания строки кода.



Рисунок 5 – Интерфейс симулятора iPhone 8 plus в XCode

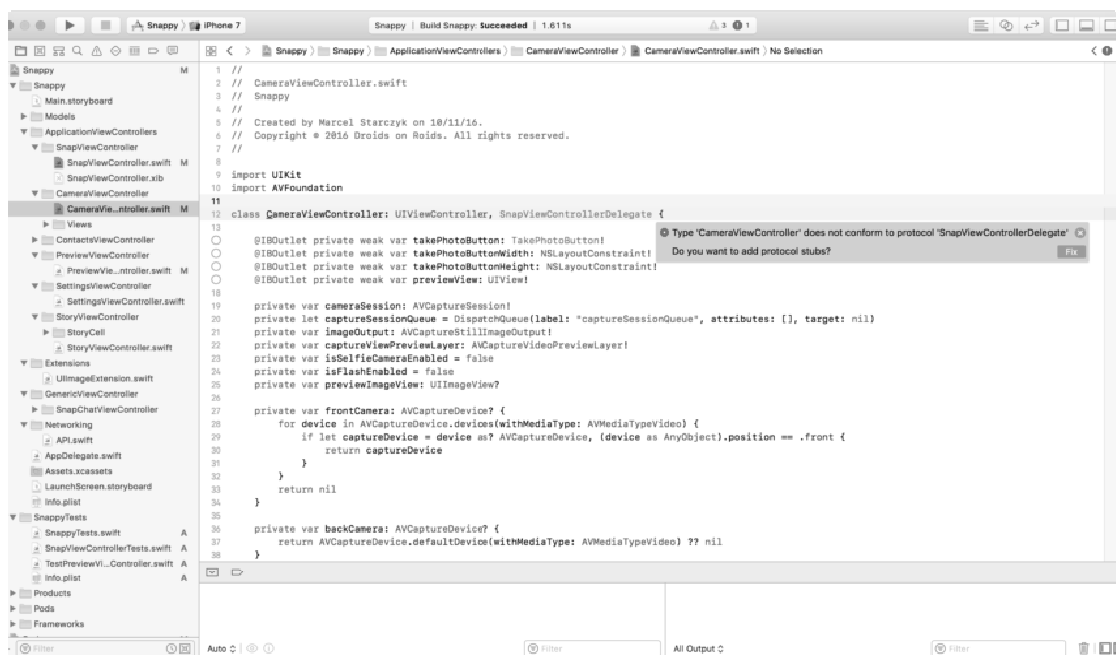


Рисунок 6 – Интерфейс Source Editor в XCode

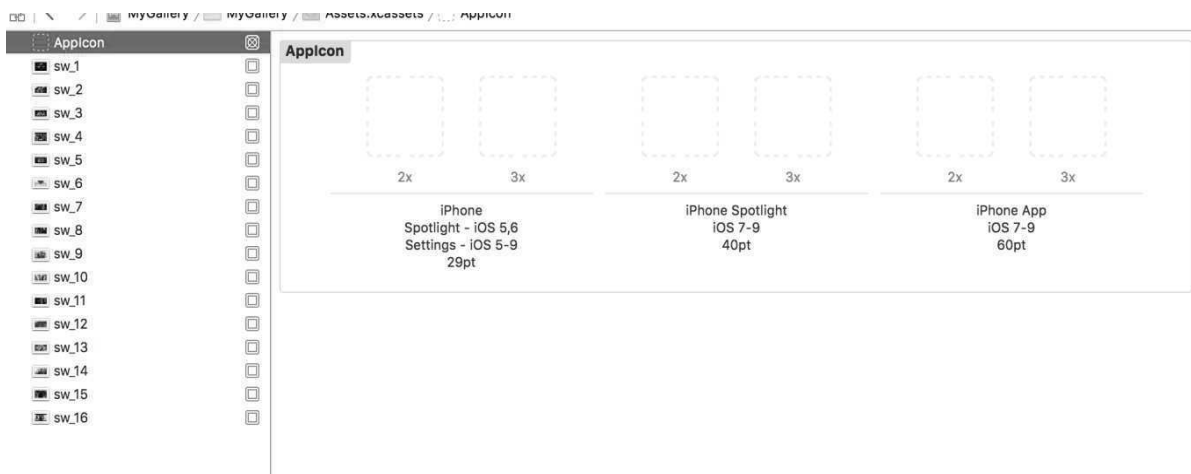


Рисунок 7 – Интерфейс Assets Catalog в XCode

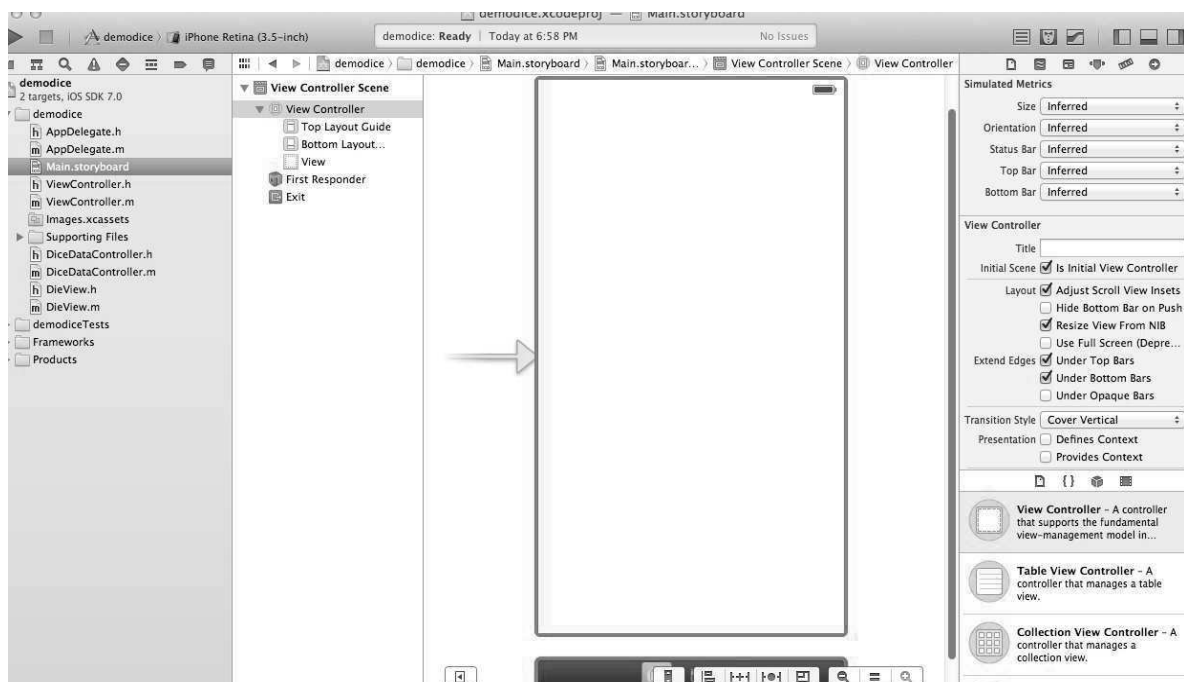


Рисунок 8 – Интерфейс Interface Builder в XCode

Основной язык программирования для разработки приложения выбран Swift версии 3.1, компилируемый объектно-ориентированный язык программирования, используемый корпорацией Apple.

Основные преимущества Swift перед Objective-C:

- 1) Apple и IBM поддерживают разработку Swift.
- 2) Swift — более компактный язык программирования, чем Objective-C.
- 3) Синтаксис и строение языка исключают несколько типов ошибок, которые потенциально возможны в Objective-C. Эта особенность языка позволяет избежать нежелательных погрешностей.

Для разработки мобильных приложений для платформы iOS используется фреймворк Cocos2d. Cocos2d имеет более 48 тысяч библиотек и используется в более чем 3 миллионах приложений. В качестве хранилища данных для мобильного приложения был выбран внешний сайт, использующий формат представления географических структур - GeoJSON. Он имеет полный функционал: добавление, удаление, редактирование.

Вывод

В результате анализа предметной области и существующих средств разработки, нами была сформулирована постановка задачи, рассмотрены аналогичные проекты, имеющиеся на рынке, а также рассмотрены инструменты для разработки мобильного приложения.

2. ПРОЕКТИРОВАНИЕ ПРИЛОЖЕНИЯ

В данном разделе описаны требования к проектируемому приложению, рассмотренные варианты использования и основные классы.

2.1. Требования к приложению

Разрабатываемое мобильное приложение должно позволять:

- авторизацию с помощью приложения VK;
- отслеживание геолокации пользователей;
- получать детали задания;
- навигацию пользователей;
- передачу ответов на задание организаторам;
- просмотр раздела контакты.

Помимо функциональных требований, можно выделить следующие нефункциональные требования:

- приложение должно иметь интуитивно понятный интерфейс;
- приложение должно работать на устройствах начиная с iPhone 5s;
- мобильное приложение должно быть реализовано на языке Swift;

2.2. Варианты использования

Диаграмма прецедентов (диаграмма вариантов использования) – это диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Рассмотрим варианты использования разрабатываемого мобильного приложения. Диаграмма вариантов использования представлена на рисунке 9.

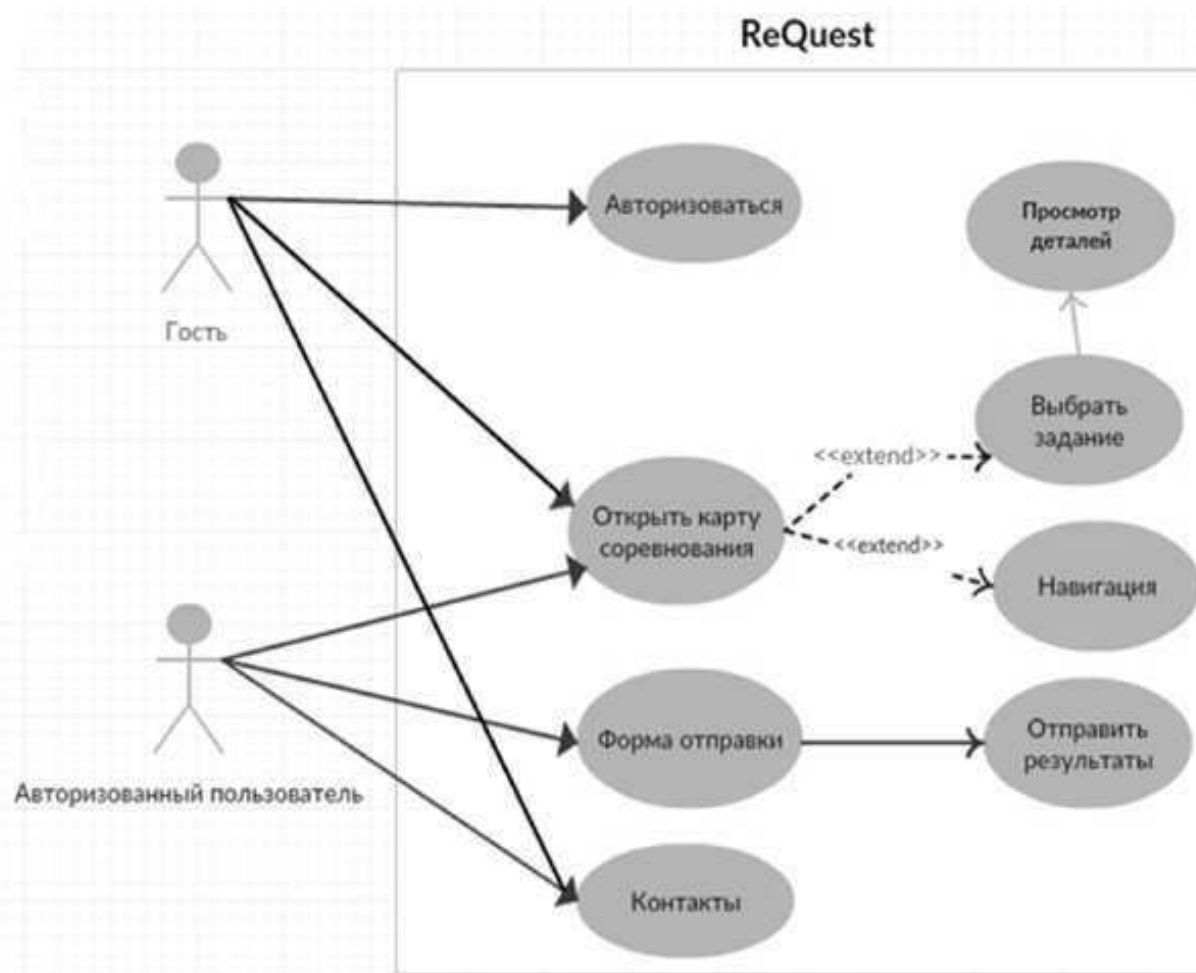


Рисунок 9 – Диаграмма вариантов использования

Пользователь – человек, который взаимодействует с приложением.

Краткое описание вариантов использования:

1. Авторизация. Пользователь авторизуется через главное меню.
2. Просмотр карты соревнований. Пользователь просматривает экран соревнования.
3. Просмотр деталей задания. Пользователь на экране карты соревнований, выбирает иконку задания, снизу открываются подробности.
4. Отправка результатов. Пользователь заходит в форму отправки через главное меню, вводит данные, которые отправляются с помощью SDK VK.

5. Просмотр контактов. Пользователь просматривает экран контактной информации, содержащий краткую информацию о приложении и контакты разработчика.

2.3. Архитектура приложения

Архитектура данного приложения использует паттерн проектирования MVC (модель-представление-контроллер). Назначением этого паттерна является разбиение приложения на блоки, имеющие между собой слабые связи. Поэтому, редактируя один блок, мы оказываем минимальное влияние на остальные.



Рисунок 10 – Архитектура приложения

Контроллер – связующее звено этой системы. Обрабатывает события, создаваемые пользователем (нажатия кнопок, поиск, переходы на другие экраны), наполняет представление данными из модели.

1. MainViewController – контроллер, реализующий логику перемещения между рубриками на главном экране.
2. AuthVKViewController – контроллер, реализующий авторизацию через приложение VK.
3. QuestMapViewController – контроллер, реализующий отображение карт, пользовательской навигации, просмотр деталей заданий.
4. SendAViewController – контроллер, реализующий отправку ответа

используя SDK VK.

5. `ContactViewController` – Контроллер, реализующий просмотр контактной информации.

На рис. 5 представлена диаграмма классов основных экранов мобильного приложения «ReQuest».

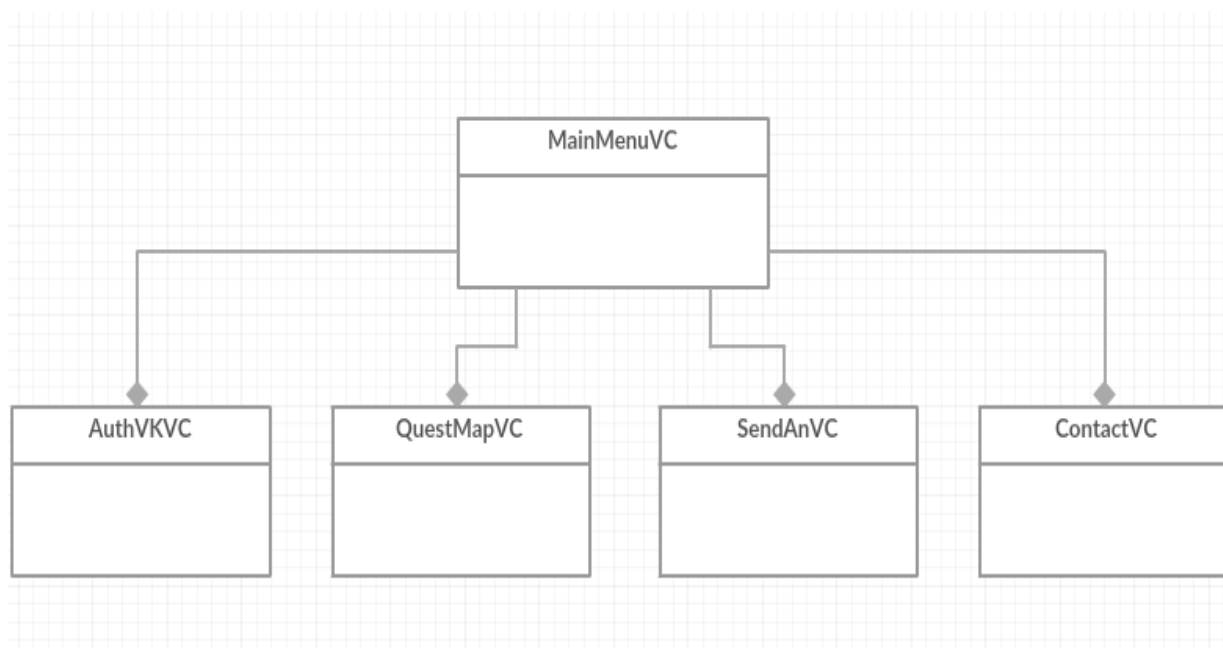


Рисунок 11 – Диаграмма классов

Представление – компонент, отвечающий за отображение информации.

В разрабатываемом приложении данный компонент включает в себя экраны:

1. Menu – экран главного меню, является стартовым экраном приложения.
2. Autoriz – экран авторизации в приложении, авторизация происходит с помощью SDK от Вконтакте.
3. Quest – экран соревнования, навигации и просмотра заданий, является основным экраном в приложении, так как содержит большую часть реализованных функций.
4. Answer – экран ввода ответов, ответы отправляются с помощью SDK от Вконтакте.

5. Contact – экран с контактной информацией, содержит основную информацию о приложении и контакты разработчика.

Модель предоставляет данные, а представление их отображает пользователю.

3. РЕАЛИЗАЦИЯ

3.1. Средства реализации

Для подключения сторонних библиотек в проект интегрирован CocoaPods – средство для управления зависимостями Cocoa-библиотек.

В проект интегрированы следующие библиотеки:

- 1) MapboxSDK – SDK компании Mapbox;
- 2) VK-ios-SDK – SDK компании Вконтакте.

3.2. Реализация контроллеров

Контроллер представляет собой набор классов, которые выражают реакцию приложения на действия пользователя: сохраняют данные в локальное хранилище, формируют представления на основе данных из моделей, обрабатывают нажатия, жесты и другие действия.

Контроллер MainMenuViewController (рисунок 12) отвечает за обработку экрана, отображающего список функций приложения. Данный контроллер принимает делегат UIMainMenuViewControllerDelegate для реализации списка экранов, каждый из которых содержит данные для соответствующих категорий из меню.

Контроллер AuthVKViewController содержит логику обработки функции «Авторизация». При нажатии на эту иконку, вызывается соответствующий метод для работы с SDK социальных сетей.

Контроллер QuestMapViewController полностью отвечает за навигацию в приложении, получение заданий и их просмотр.

Получение данных для контроллера QuestMapViewController организовано с помощью GeoJSON. Файл с данными в формате GeoJSON (рисунок 13) считывается с внешнего источника и данные отображаются на карте в контроллере. Для организации и редактирования такого файла есть множество

удобных сервисов (рисунки 14 и 15).

Также были реализованы контроллеры:

1) SendAController – обрабатывает введенную информацию и отправляет с помощью инструментов SDK VK.

2) ContactViewController – экран контактной информации.

```
class MainMenuViewController: UITableViewController {  
    override func viewDidLoad() {  
        super.viewDidLoad()  
        tableView.contentInset = UIEdgeInsets(top: 10, left: 0, bottom: 0, right: 0)  
        tableView.tableFooterView = UIView() } }  
    override func tableView(_ tableView: UITableView,  
        cellForRowAt indexPath: IndexPath) -> UITableViewCell {  
        let cell = tableView.dequeueReusableCell(withIdentifier: "example", for: indexPath) as!  
        MapTableViewCell  
        let center = CLLocationCoordinate2D(latitude: 61.370002, longitude: 55.160332)  
        let camera = MGLMapCamera(lookingAtCenter: center, fromDistance: 0, pitch: 0,  
        heading: 0)  
        let snapshotOptions = MGLMapSnapshotOptions(styleURL: map.styleURL, camera:  
        camera, size: cell.MapImageView.bounds.size)  
        snapshotOptions.zoomLevel = 12.5  
        let snapshotter = MGLMapSnapshotter(options: snapshotOptions)  
        snapshotter.start { (image, error) in  
            guard let image = image else { return }  
            cell.themeImageView.image = image.image }  
        cell.itemMarkerImageView.image = item.defaultMarker  
        return cell  
    }  
}
```

Рисунок 12 – Листинг реализации контроллера MainMenuViewController


```

{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "properties": {
        "name": "Задание 1",
        "difficult": "Средняя",
        "category": "Наблюдательность",
        "description": "Сосчитать количество берез в радиусе 10м"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [
          61.33529663085937,
          55.15406051701917
        ]
      }
    },
    {
      "type": "Feature",
      "properties": {
        "marker-color": "#7e7e7e",
        "marker-size": "medium",
        "marker-symbol": "",
        "name": "Задание 2",
        "difficult": "Средняя",
        "category": "Наблюдательность",
        "description": "Найти название животного"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [
          61.33529663085937,
          55.15406051701917
        ]
      }
    }
  ]
}

```


```

    "geometry": {
      "type": "Point",
      "coordinates": [
        61.35503768920898,
        55.16558354205109
      ]
    } },
    {
      "type": "Feature",
      "properties": {
        "name": "Задание 3",
        "difficult": "Средняя",
        "category": "Сила",
        "description": "Присутствует организатор"
      },
      "geometry": {
        "type": "Point",
        "coordinates": [
          61.34010314941406,
          55.159160479203756
        ]
      } },
    {
      "type": "Feature",
      "properties": {
        "name": "Задание 4",
        "difficult": "Средняя",
        "category": "Смекалка",
        "description": "Присутствует организатор"
      },
    },

```

```

    "geometry": {
      "type": "Point",
      "coordinates": [
        61.35787010192871,
        55.14842037652591
      ] } },
  {
    "type": "Feature",
    "properties": {
      "marker-color": "#7e7e7e",
      "marker-size": "medium",
      "marker-symbol": "",
      "name": "Задание 5",
      "difficult": "Средняя",
      "category": "Ловкость",
      "description": "На дереве спрятан код"
    },
    "geometry": {
      "type": "Point",
      "coordinates": [
        61.36362075805663,
        55.1561201956302
      ]
    }
  }
  {
    "type": "Feature",
    "properties": {
      "name": "Задание 6",
      "difficult": "Высокая",

```

```

    "category": "Смекалка",
    "description": "Присутствует организатор"
  },
  "geometry": {
    "type": "Point",
    "coordinates": [
      61.39088810192871,
      55.16440337652591
    ] }},
{
  "type": "Feature",
  "properties": {
    "name": "Задание ",
    "difficult": "Высокая",
    "category": "Наблюдательность",
    "description": "Красный объект"
  },
  "geometry": {
    "type": "Point",
    "coordinates": [
      61.40430450439453,
      55.17651507880266
    ]   }}

```

Рисунок 13 – Листинг GeoJSON файла

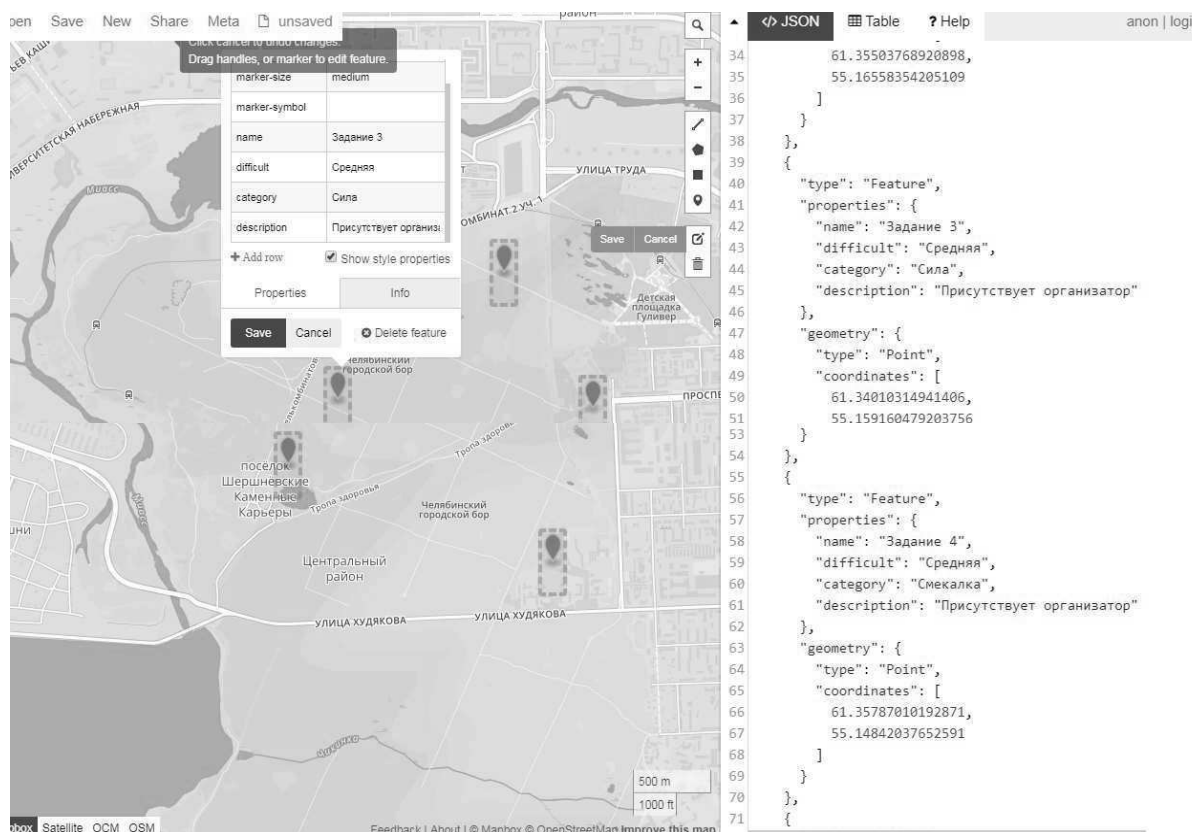


Рисунок 14 – Скриншот GeoJSON редактора

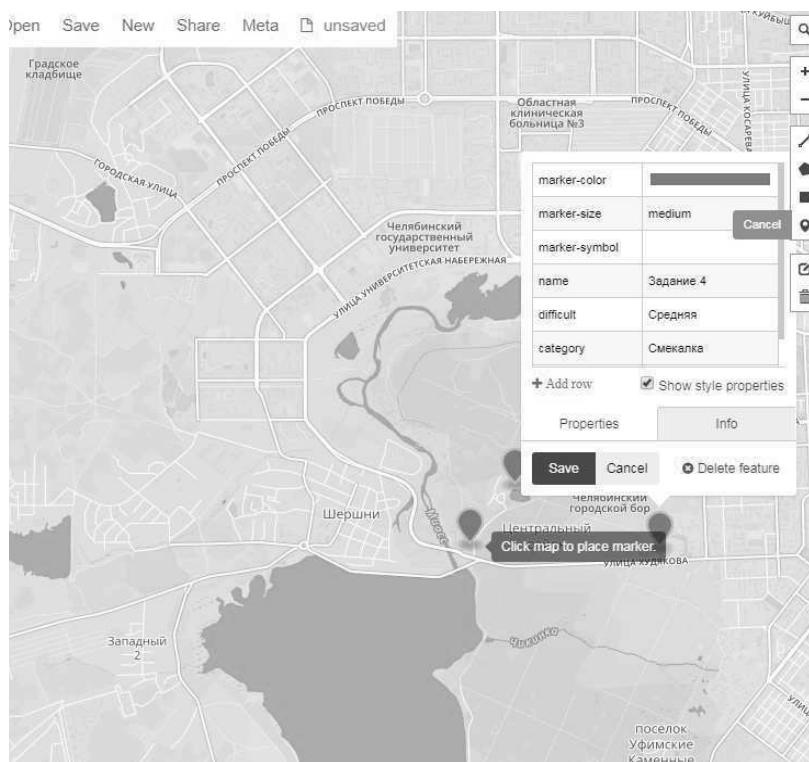


Рисунок 15 – Скриншот добавления новой точки в GeoJSON редакторе

3.3 Реализация интерфейса приложения

Стандартный инструмент создания интерфейса – Interface Builder (рисунок 16)

са для разработчиков. Эти объекты содержат такие элементы, как текстовые поля, таблицы данных, форма для ввода даты, кнопки. Для создания интерфейса данные объекты перетаскиваются из существующей палитры объектов на экран. Auto Layout занимается динамическим вычислением позиции и размера всех объектов иерархии объектов на основе правил, заданных для того или иного объекта. Эти правила называются layout constraints и задаются вручную. Для создания связи с кодом используются поля класса Outlet, ссылающиеся на конкретные объекты в Interface Builder. Все экраны и их связи друг с другом хранятся в файле storyboard (рисунок 17). Этот механизм позволяет автоматизировать систему перехода между экранами без добавления кода в программу, а также возможность добавления и настройки элементов графического интерфейса.

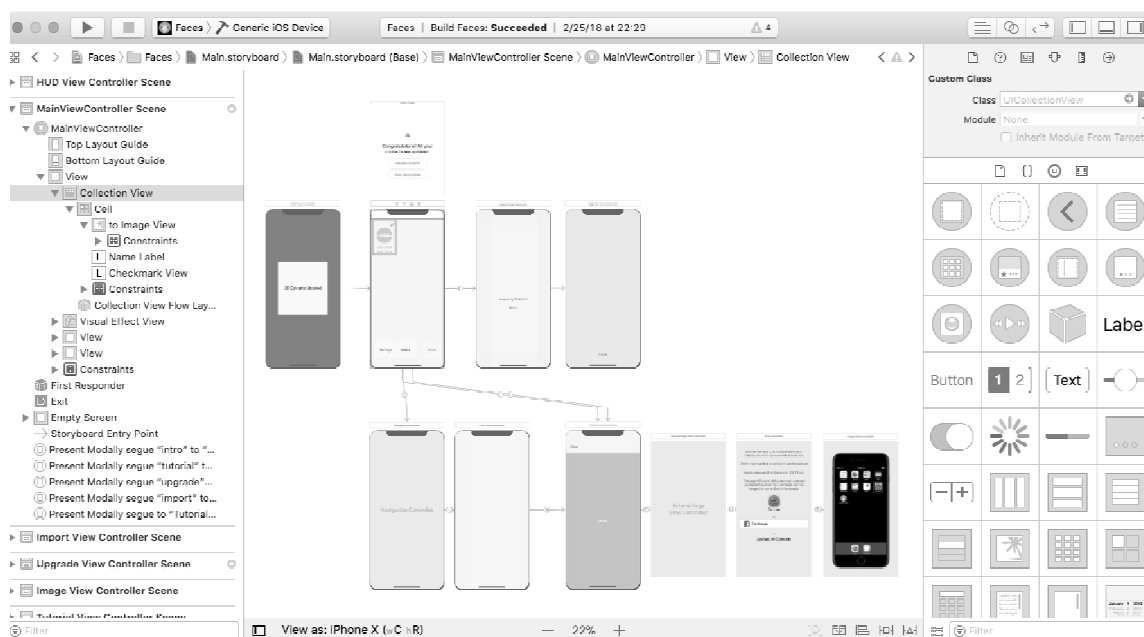


Рисунок 16 – Интерфейс Interface Builder в XCode



Рисунок 17 – Интерфейс Storyboard в XCode

Входным экраном приложения является экран класса MainMenuController (рисунок 18). На данном экране пользователь видит меню приложения. Так же он может перейти в любой из разделов меню.



Рисунок 18 – Главный экран приложения

Главным рабочим экраном данного приложения является QuestMapViewController (рисунок 19). Он отвечает за отображение ключевой информации данного приложения.

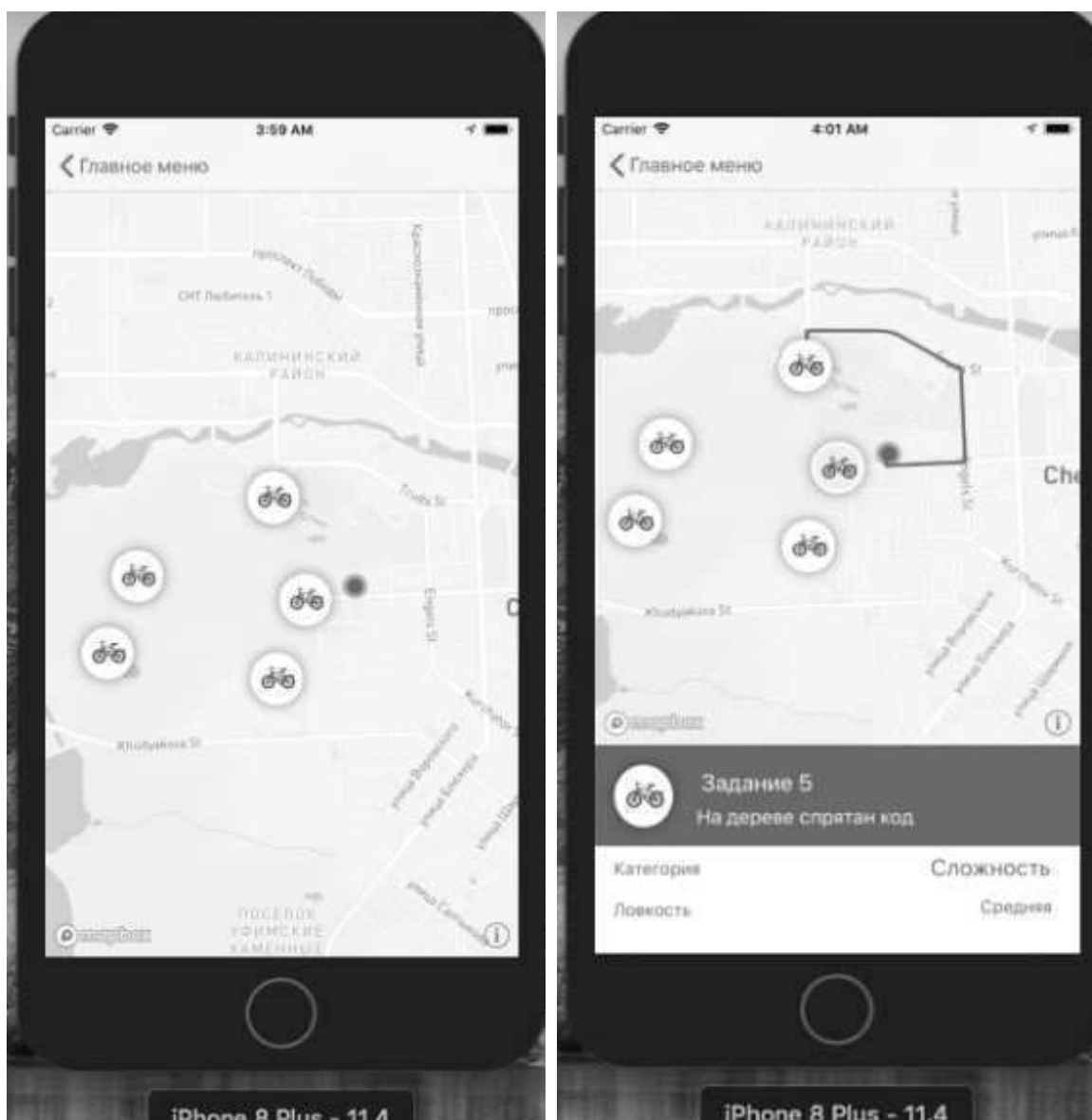


Рисунок 19 – Снимки экрана интерактивной карты

На данном экране пользователь видит свое текущее местоположение (рисунок 20). И контрольные точки с заданиями, при нажатии на которые появляется описание задания, его категория и уровень сложности. Поддерживается смена описания перелистываниями вправо и влево. Так же при выборе контрольной точки до нее автоматически прокладывается маршрут от текущего местоположения.

```

func addUserLocationDot(to style: MGLStyle) {

    if !CLLocationCoordinate2DIsValid(userLocationFeature.coordinate) {

        userLocationFeature.coordinate = centerCoordinate

    }

    userLocationSource = MGLShapeSource(identifier: "user-location", features:
[userLocationFeature], options: nil)

    let userLocationStyle = MGLCircleStyleLayer(identifier: "user-location-style", source:
userLocationSource!)

    userLocationStyle.circleColor = NSExpression(forConstantValue: mapViewController)

    userLocationStyle.circleRadius = NSExpression(forConstantValue: 7)

    userLocationStyle.circleStrokeColor = NSExpression(forConstantValue: mapViewController)

    userLocationStyle.circleStrokeWidth = NSExpression(forConstantValue: 4)

    userLocationStyle.circleStrokeOpacity = NSExpression(forConstantValue: 0.5)

    style.addSource(userLocationSource!)

    style.addLayer(userLocationStyle)

}

```

Рисунок 20 – Листинг добавления местоположения пользователя на карту

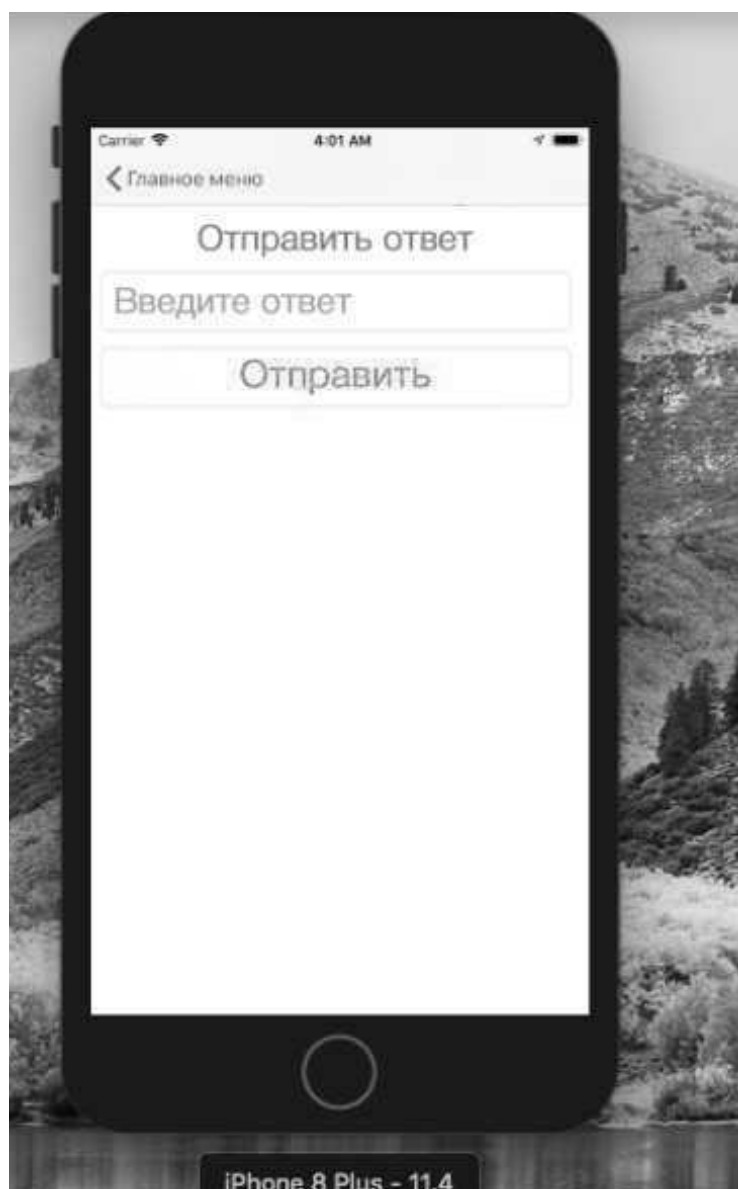


Рисунок 21 – Снимок экрана отправки ответов

4. ТЕСТИРОВАНИЕ

Выделяют несколько основных видов тестирования мобильных приложений:

- 1) Функциональное тестирование – тестирование на соответствие начальным требованиям.
- 2) Лабораторное тестирование – тестирование влияния внешних факторов, таких как качество соединения с Интернет.
- 3) Тестирование производительности.
- 4) Тестирование утечек памяти.
- 5) Юзабилити-тестирование – тестирование на удобство интерфейса.
- 6) Тестирование на установку.
- 7) Тестирование на соответствие стандартам – тестирование на соответствие общепринятым стандартам разрабатываемой платформы.

Для тестирования реализованного приложения было выбрано функциональное тестирование.

4.1 Функциональное тестирование приложения

Функциональное тестирование – это тестирование разработанного программного обеспечения в целях проверки реализованности функциональных требований, то есть способности программного обеспечения в определенных условиях решать задачи, нужные пользователям [8]. Функциональные требования определяют, что именно делает программное обеспечение, какие задачи оно решает.

Работа приложения была протестирована в среде разработки xCode на эмуляторе iPhone SE и iPhone 8 plus с версией iOS 11.4. (рисунок 22)



Рисунок 22 – Тестирование на iPhone SE и 8 plus

Тест № 1

Цель: протестировать функцию просмотра соревнования.

Ожидаемый результат: приложение переходит на экран интерактивной карты с метками местоположения

Входные данные: пользователь находится на главном экране приложения.

Процедура тестирования: пользователь нажимает на кнопку интерактивной карты.

Полученный результат: совпадает с ожидаемым (рисунок 23).

Вывод: тест пройден.

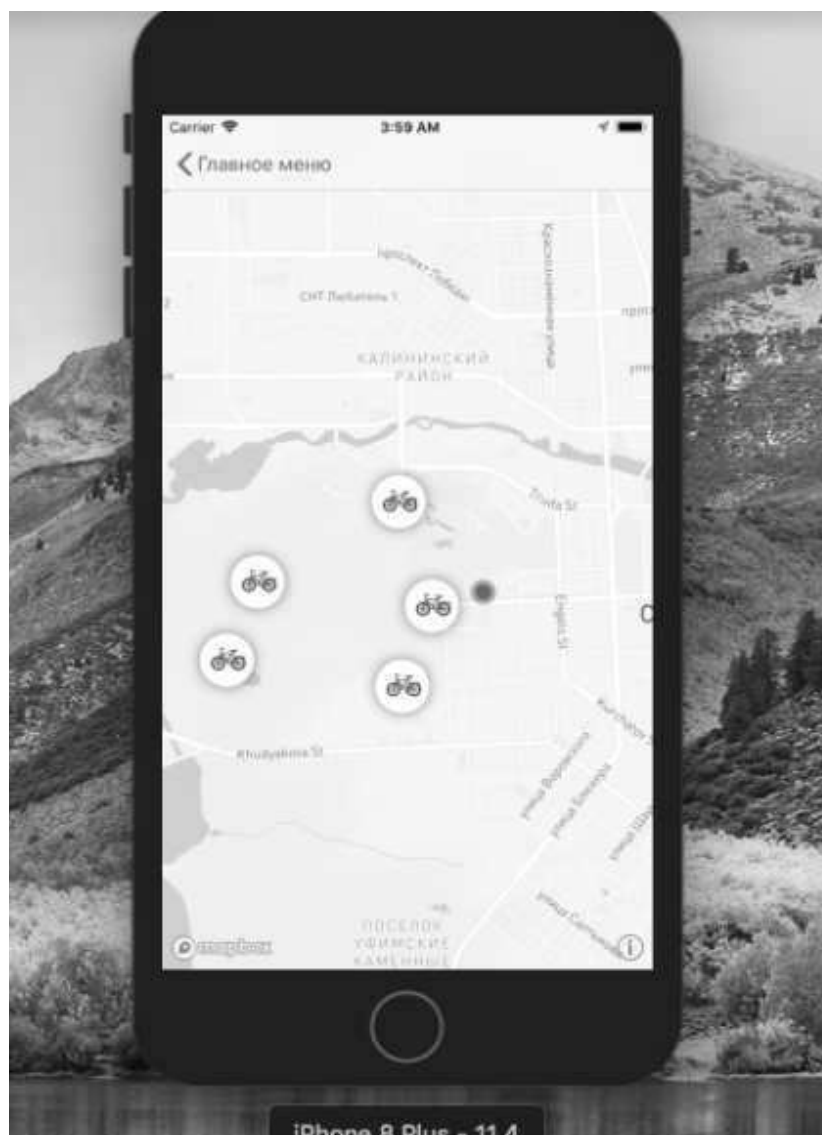


Рисунок 23 – Снимок экрана интерактивной карты

Тест № 2

Цель: протестировать функцию просмотра деталей точки задания.

Ожидаемый результат: приложение переходит на экран интерактивной карты с метками местоположения, выбранная метка подсвечивается и в нижней части экрана появляется описание задания, уровень сложности и категория задания.

Входные данные: пользователь находится на главном экране приложения.

Процедура тестирования: пользователь нажимает на кнопку интерактивной карты, приложение переходит на экран интерактивной карты, на нем пользователь нажимает на маркер задания (значок велосипеда).

Полученный результат: совпадает с ожидаемым (рисунок 24).

Вывод: тест пройден.

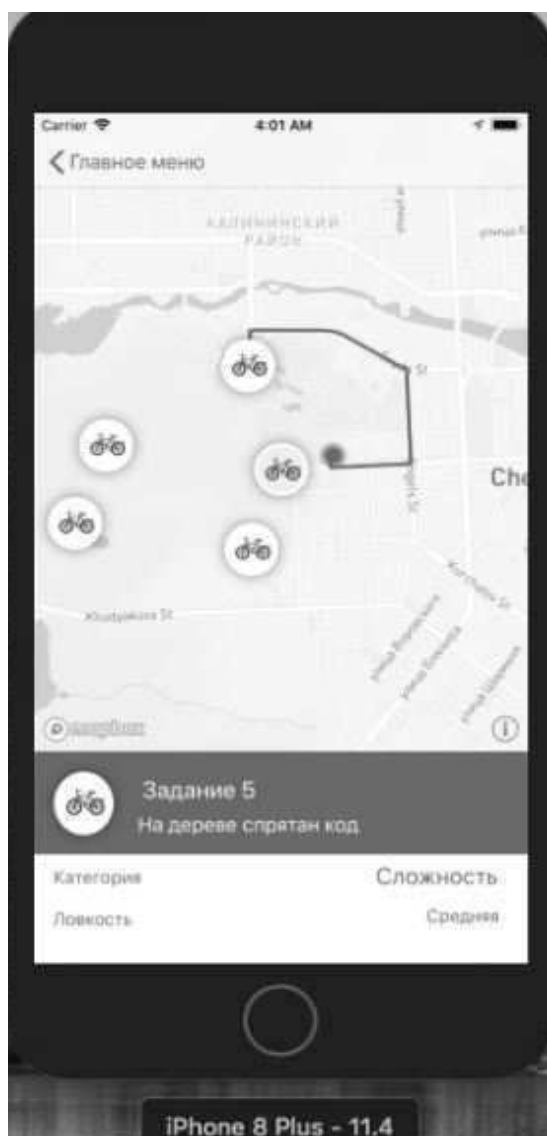


Рисунок 24 – Снимок экрана интерактивной карты с выбранным заданием

Тест № 3

Цель: протестировать функцию навигации.

Ожидаемый результат: построенный маршрут от текущего местоположения устройства до заданной координаты совпадает с маршрутом построенным Google Maps.

Входные данные: пользователь находится на главном экране приложения.

Процедура тестирования: пользователь нажимает на кнопку интерактивной карты, приложение переходит на экран интерактивной карты, на нем пользователь нажимает на маркер задания (значок велосипеда).

Полученный результат: совпадает с ожидаемым (рисунок 25).

Вывод: тест пройден.

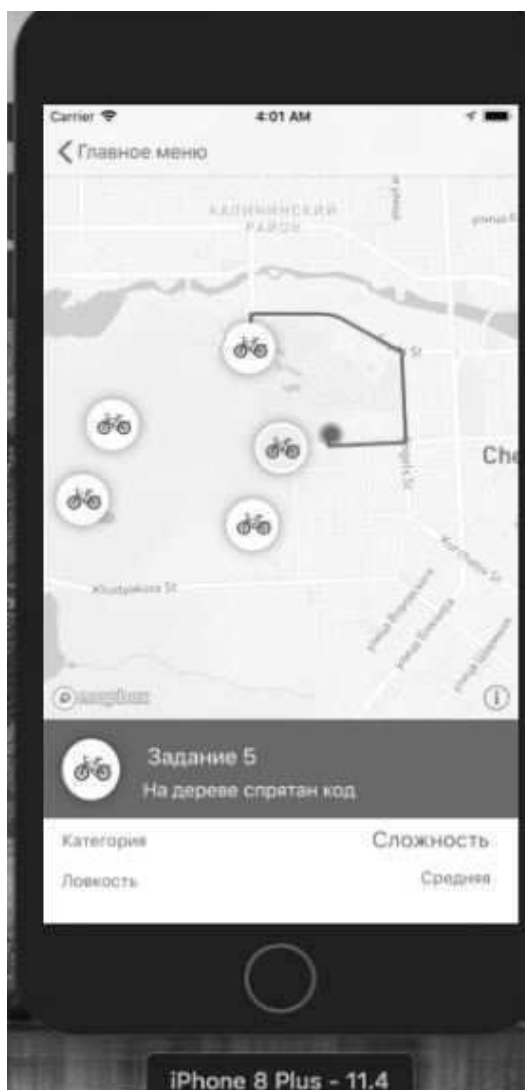


Рисунок 25 – Снимок экрана интерактивной карты с проложенным маршрутом

Тест № 4

Цель: протестировать точность определения геопозиции.

Ожидаемый результат: определенные координаты совпадут с координатами на другом устройстве.

Входные данные: пользователь находится на главном экране приложения.

Процедура тестирования: пользователь нажимает на кнопку интерактивной карты, приложение переходит на экран интерактивной карты, на нем пользователь нажимает на маркер задания (значок велосипеда).

Полученный результат: совпадает с ожидаемым.

Вывод: тест пройден.

Тест №5

Цель: проверка работы выбора заданий.

Ожидаемый результат: Выделится следующее задание и появится его описание, смена происходит в порядке убывания или возрастания, в зависимости от стороны перелистывания.

Входные данные: пользователь находится на главном экране приложения.

Процедура тестирования: перелистывать влево вправо по этикетке с деталями задания.

Полученный результат: совпадает с ожидаемым.

Вывод: тест пройден.

Тест №6

Цель: проверка работы информационного экрана.

Ожидаемый результат: отображается экран с информацией о приложении и контактами разработчика.

Входные данные: главный экран приложения.

Процедура тестирования: пользователь нажимает на кнопку контакты.

Полученный результат: совпадает с ожидаемым (рисунок 26).

Вывод: тест пройден.



Рисунок 26 – Снимок экрана контактной информации

Тест №7

Цель: проверка корректности работы на различных разрешениях экрана.

Ожидаемый результат: информация корректно отображается на всех разрешениях экранов.

Входные данные: экран, на котором открыт просмотр деталей задания.

Процедура тестирования: Запуск приложения на iPhone 8 plus и iPhone SE. Экран iPhone 8 plus имеет разрешение 1920*1080, а iPhone SE 1136*640, были выбраны именно эти модели, так как iPhone 8 plus имеет самое высокое разрешения, а iPhone SE самое низкое (рисунок 27).

Полученный результат: совпадает с ожидаемым.

Вывод: тест пройден.



Рисунок 27 – Снимки экранов iPhone 8 plus и iPhone SE

Тест №8

Цель: проверка динамического перестроения кратчайшего маршрута при движении.

Ожидаемый результат: при передвижении, маршрут перестраивается по кратчайшему пути.

Входные данные: экран интерактивной карты.

Процедура тестирования: смена местоположения, наблюдение за перестроением маршрута.

Полученный результат: совпадает с ожидаемым.

Вывод: тест пройден.

Тест №8

Цель: проверка работы авторизации

Ожидаемый результат: при нажатии на кнопку «Авторизация» в главном меню произойдет переход на форму авторизации через приложение Вконтакте.

Входные данные: главное меню приложения.

Процедура тестирования: нажать на кнопку «Авторизация», при переходе на форму Вконтакте, разрешить приложению доступ к информации.

Полученный результат: совпадает с ожидаемым.

Вывод: тест пройден.

Тест №10

Цель: проверка работы формы отправки ответов.

Ожидаемый результат: после отправки ответа, через приложение Вк придет ответ с подтверждением принятия ответа.

Входные данные: главное меню приложения.

Процедура тестирования: нажать на кнопку «Отправить ответ», при переходе на форму отправки, ввести ответ и нажать кнопку отправить.

Полученный результат: совпадает с ожидаемым.

Вывод: тест пройден.

Тест №11

Цель: проверка работы формы отправки ответов, при неправильном ответе.

Ожидаемый результат: после отправки неправильного ответа, через приложение Вконтакте придет ответ с подтверждением отклонения ответа.

Входные данные: главное меню приложения.

Процедура тестирования: нажать на кнопку «Отправить ответ», при переходе на форму отправки, ввести неправильный ответ и нажать кнопку отправить.

Полученный результат: совпадает с ожидаемым.

Вывод: тест пройден.

Вывод: для проведения функционального тестирования разработанного приложения, было разработано 11 функциональных тестов, все пройдены успешно.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы были разработано мобильное приложение для iOS. Для достижения этой цели были выполнены следующие задачи:

- 1) проведен обзор аналогичных мобильных приложений для платформы iOS.
- 2) проведен обзор инструментов для реализации проекта.
- 3) спроектировано мобильное приложение с использованием диаграммы вариантов использования и диаграммы классов.
- 4) реализовано мобильное приложение для операционной системы iOS.
- 5) проведено тестирование разработанного приложения.

Все поставленные задачи были решены, цель достигнута.

Разработанное приложение имеет перспективы дальнейшего развития. С учетом усложнения бизнес-процесса и ростом требований заказчика, возникает потребность в расширении функционала системы.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Документация Swift. [Электронный ресурс] URL: <https://developer.apple.com/documentation/swift> (дата обращения: 03.05.2018).
2. Введение в JSON. [Электронный ресурс] URL: <http://www.json.org/json-ru.html> (дата обращения: 09.05.2018).
3. Грей Э. Swift. Карманный справочник. Программирование в среде iOS и OS X. – М.: Вильямс, 2016. – 288 с.
4. Марк Д. Swift. Разработка приложений в среде Xcode для iPhone и iPad с использованием iOS SDK. – М.: Вильямс, 2016. – 816 с.
5. Харазян А. Язык Swift. Самоучитель. – СПб.: БХВ-Петербург, 2016. – 176 с.
6. Документация VK-ios-SDK. [Электронный ресурс] URL: https://vk.com/dev/ios_sdk (дата обращения: 09.05.2018).
7. Усов В. Swift. Основы разработки приложений под iOS и macOS. – СПб.: Питер, 2017. – 368 с.
8. Документация PhoneGap. [Электронный ресурс] URL: <https://phonegap.com/> (дата обращения: 03.05.2018).
9. Документация Mapbox SDK. [Электронный ресурс] URL: <https://www.mapbox.com/ios-sdk/api/> (дата обращения: 09.05.2018).
10. Документация CocosPods. [Электронный ресурс] URL: <https://guides.cocoapods.org/> (дата обращения: 09.05.2018).
11. Документация Xamarin. [Электронный ресурс] URL: <https://developer.xamarin.com/> (дата обращения: 03.05.2018).
12. Документация GeoJSON. [Электронный ресурс] URL: <https://tools.ietf.org/html/rfc7946/> (дата обращения: 09.05.2018).

13. Документация Titanium. [Электронный ресурс] URL: <https://www.appcelerator.com/mobile-app-development-products/> (дата обращения 03.05.2018).
14. Документация Objective-C. [Электронный ресурс] URL: https://developer.apple.com/documentation/objectivec/objective_c_runtime?changes=_3/ (дата обращения: 03.05.2018).
15. Ларман, Крэг Применение UML 2.0 и шаблонов проектирования. Введение в объектно-ориентированный анализ, проектирование и итеративную разработку / Крэг Ларман. - Москва: Гостехиздат, 2017. - 736 с.
16. Interface Builder документация. [Электронный ресурс] URL: <https://developer.apple.com/xcode/interface-builder/> (дата обращения: 03.05.2018).
17. Storyboard документация. [Электронный ресурс] URL: https://developer.apple.com/documentation/uikit/uistoryboard?changes=_7/ (дата обращения: 09.05.2018).
18. View controllers документация. [Электронный ресурс] URL: https://developer.apple.com/documentation/uikit/view_controllers/ (дата обращения: 09.05.2018).
19. View layout документация. [Электронный ресурс] URL: https://developer.apple.com/documentation/uikit/view_layout/ (дата обращения: 09.05.2018).
20. Touches presses and gestures. [Электронный ресурс] URL: https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/ (дата обращения: 09.05.2018).
21. Image I/O Framework [Электронный ресурс] URL:

<https://developer.apple.com/documentation/imageio/> (дата обращения: 09.05.2018).

22. Keyboard and menus [Электронный ресурс] URL: https://developer.apple.com/documentation/uikit/keyboard_and_menus/ (дата обращения: 09.05.2018).