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 системного программирования

РАБОТА ПРОВЕРЕНА

Рецензент,
доцент кафедры ИИТиМПИ
ЮУрГГПУ, к.п.н.

_____Л.С. Носова

“___”_____ 2018 г.

ДОПУСТИТЬ К ЗАЩИТЕ

Заведующий кафедрой,
д.ф.-м.н., профессор

_____ Л.Б. Соколинский

“___”_____ 2018 г.

РАЗРАБОТКА МОБИЛЬНОГО ПРИЛОЖЕНИЯ "EVIOT" ДЛЯ УПРАВЛЕНИЯ БЕСПРОВОДНЫМИ УСТРОЙСТВАМИ ИНТЕРНЕТА ВЕЩЕЙ

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ – 02.03.02.2018.115-039.ВКР

Научный руководитель
доцент кафедры СП, к.п.н.

_____ О.Н. Иванова

Автор работы,
студент группы КЭ-402

_____ Е.В. Семенов

Ученый секретарь
(нормоконтролер)

_____ О.Н. Иванова

“___”_____ 2018 г.

Челябинск–2018

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы

Управление устройствами интернета вещей является важной отраслью в становлении нового сегмента технологий умных вещей, помогающих человеку в различных жизненных обстоятельствах управлять устройствами не только находясь вблизи них, но также на больших расстояниях посредством сети Интернет.

В настоящее время IoT развивается главным образом в направлении совершенствования промышленных систем. Но в ближайшее время многие персональные устройства также перейдут в ряд IoT устройств, уже сейчас, например, существуют умные розетки различных компаний, где с помощью мобильного приложения можно выключить и включить, поставить на таймер данное устройство, находясь на работе или в другом месте. Это существенно экономит время на некоторые задачи человека. Также за счет усовершенствованных моделей датчиков современные устройства («вещи») могут считывать, собирать и интерпретировать беспрецедентные объемы данных о потребителях, отправлять их в общую базу государственного реестра, где может производиться расчет различных данных.

В данный момент, отрасль IoT только набирает популярность во всех странах, но многие компании ввиду сложности реализации умного устройства или нехваткой денег на разработку такого устройства не желают реорганизовать свои устройства в умные «вещи», так как нет платформы для создания, тестирования и выпуск в жизнь IoT – устройств.

Цель и задачи

Целью работы является создание мобильного приложения для управления устройствами интернета вещей, взаимодействующего с платформой для разработки умных устройств «Eviot Dev» и веб-сервисом «Eviot Web» с помощью веб-фреймворка Meteor. Для достижения данной цели должны быть решены следующие задачи:

1) произвести обзор и изучить фреймворк Meteor;
2) изучить работу с базой данных NoSql MongoDB;
3) определить требования и спроектировать мобильное приложение;
4) спроектировать базу данных;
5) реализовать и протестировать мобильное приложение.

Структура и объем работы

Работа состоит из введения, четырех глав, заключения, библиографического списка и одного приложения. Объем работы составляет 35 страниц, объем библиографии – 15 источников, объем приложения – 5 страниц.

В главе «Теоретические основы проектирования мобильных приложения для управления устройствами интернета вещей» была произведена постановка задачи, проведен обзор существующих аналогов мобильных приложения для управления умными устройствами, а также рассмотрены современные платформы для разработки мобильных приложений.

Глава «Проектирование мобильного приложения для управления устройствами интернета вещей» посвящена определению требований к разрабатываемому мобильному приложению, описания архитектуры и организации хранения данных на устройстве. В этой же главе описываются диаграмма прецедентов, приведена архитектура мобильного приложения.

В третьей главе, «Реализация мобильного приложения», рассмотрена реализация хранения данных, а также реализация основной функциональности мобильного приложения.

Глава «Тестирование мобильного приложения» посвящена результатам тестирования мобильного приложения. Представлены результаты функционального тестирования, выполненные в работающем приложении, и интеграционного тестирования мобильного приложения и веб-сервиса.

В заключении сделаны выводы о проделанной работе и сформулированы перспективы дальнейшей разработки.

1. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ ПРОЕКТИРОВАНИЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ ДЛЯ УПРАВЛЕНИЯ УСТРОЙСТВАМИ ИНТЕРНЕТА ВЕЩЕЙ

1.1. Постановка задачи

Люди всегда пытались автоматизировать и упростить свою рутинную работу. В современном мире появилась замечательная возможность возложить обязанности на плечи программируемых систем. Умная техника – одна из разновидностей этих систем. В настоящее время существует множество реализаций умной техники от самых разных компаний для различных сфер нашей жизни, например, бытовые электроприборы последних поколений, различные аксессуары, а также одежда со встроенной электроникой.

Для некоторых устройств написаны приложения для мобильных устройств, некоторые управляются непосредственно со встроенной в них панели управления, другие – со специальных пультов управления. В последнее время все популярней становится управление через специализированные приложения и сервисы, к которым можно получить доступ с любого устройства, например, со смартфона или ПК [13].

Основной задачей является разработка мобильного приложения «Eviot App» для управления устройствами интернета вещей на операционной системе Android. Важной особенностью является реализация взаимодействия мобильного приложения с платформой для разработки умных устройств «Eviot Dev» и веб-сервисом «Eviot Web».

Данное приложение поможет пользователям управлять умными устройствами, регистрировать новых пользователей и в целом выполнять роль «Eviot Web» через отдельное мобильное приложение.

Мобильное приложение должно быть предназначено для упрощения управления элементами умного дома и ориентироваться как на тех, кто сталкивается с «умной» техникой впервые, так и на тех, кто ею уже давно пользуется. Мобильное приложение должно иметь возможность добавлять

новые устройства и регистрировать их в базе данных, объединять все устройства в один интерфейс, через который осуществляется управление и просмотр данных.

1.2. Сравнительный анализ аналогов

В настоящий момент существует различные системы управления умным домом, начиная от встроенных систем до облачных сервисов, но мы рассмотрим только те, которые имеют мобильное приложение для управления устройствами интернета вещей.

Ready For Sky [10] – мобильное приложение для управления устройствами умного дома от компании Redmond (рис. 1).

Рис. 1. Мобильное приложение Ready For Sky

Достоинства:

1) приложение существует на мобильных платформах Android и iOS;
2) удобный интерфейс;

3) реализован каталог для покупки умных устройств внутри приложения.

Недостатки:

1) поддержка устройств только от компании;

2) устройства, разрабатываемые компанией Redmond, работают под управлением различных мобильных приложений (чайник, розетка, грили, мультиварки, кофеварки, утюги, - в мобильном приложении Ready For Sky, два других устройства, RSC-11S, RSC-100S, - под управлением приложения R4S Home);

3) управление устройствами осуществляется при помощи Bluetooth.

Broadlink e-Control [2] – мобильное приложение от компании Broadlink, которая позиционирует себя как компания, создающая устройства для облегчения повседневной жизни (рис. 2).

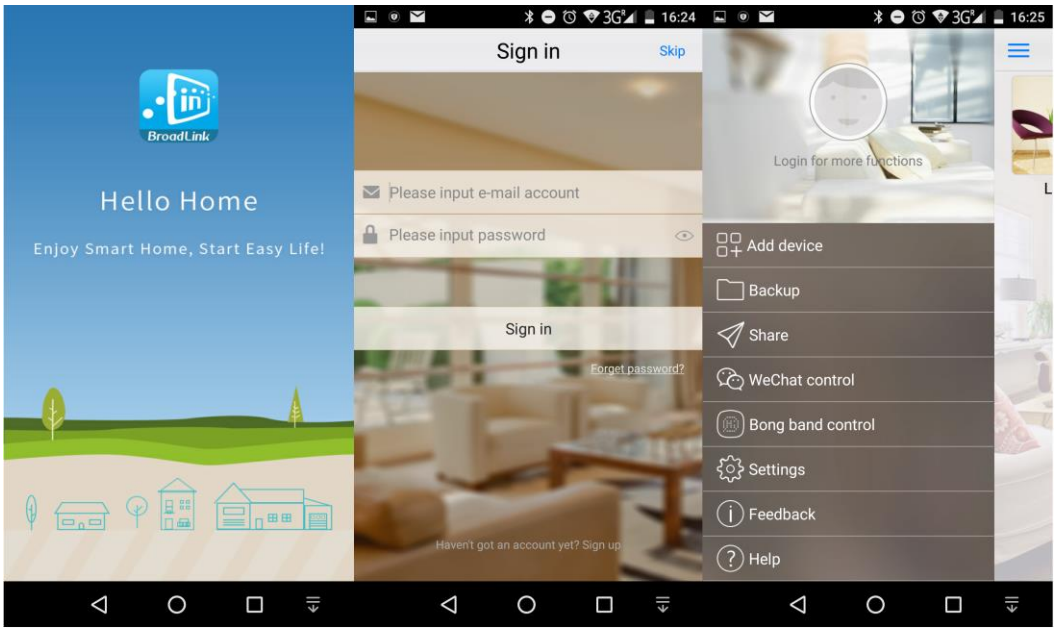

Рис. 2. Мобильное приложение Broadlink e-Control

Достоинства:

1) приложение существует на мобильных платформах Android и iOS;

2) управление устройствами при помощи Wi-Fi.

Недостатки:

1) поддержка устройств только от компании.

eWeLink [4] – мобильное приложение для управления устройствами интернета вещей от компании Sonoff (рис. 3)

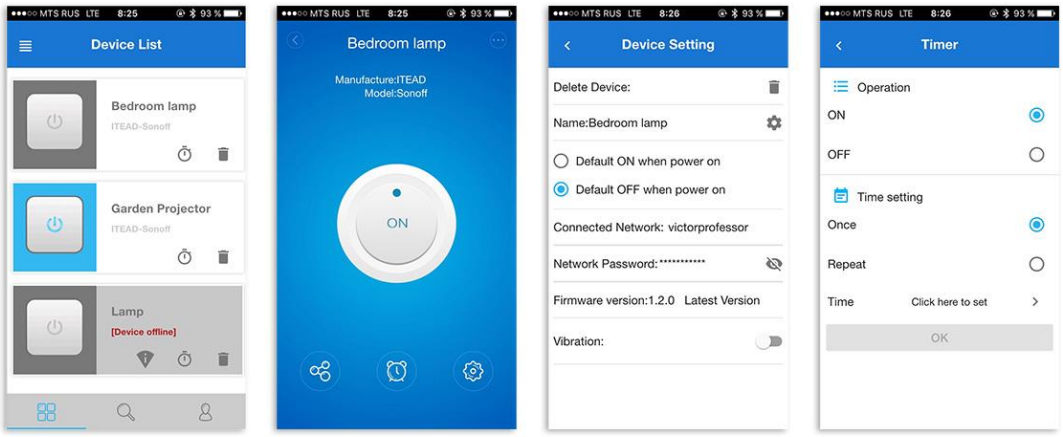

Рис. 3. Мобильное приложение eWeLink

Достоинства:

1) приложение существует на мобильных платформах Android и iOS;

2) управление устройствами при помощи Wi-Fi;

3) мультиязычный интерфейс.

Недостатки:

1) поддержка устройств только от компании и от некоторых известных брендов других компаний умных устройств (только nest, Google Home, amazon alexa).

Представленные на рынке мобильные приложения имеют как достоинства, так и недостатки. Среди основных недостатков, присущих почти всем имеющимся на рынке устройствам «умного дома», можно назвать отсутствие поддержки широкого спектра устройств других производителей. Это говорит о том, что можно улучшить функциональные особенности моего приложения, по сравнению с существующими решениями.

1.3. Современные платформы разработки мобильных приложений

Наряду с быстрыми темпами развития смартфонов и других персональных устройств также быстро развиваются различные платформы для

разработки мобильных приложений. В настоящее время их количество увеличивается, с изменениями функциональных требований смартфонов изменяются существующие платформы, посредством перестройки и обновления решений.

Сегодня существуют два лидера операционных систем для смартфонов:

1) Android - 73.52% (весь мир);
2) iOS - 19.37% (весь мир) [14].

Данные операционные системы используются повсеместно во многих странах и составляют огромную долю использования по сравнению с другими операционными системами для смартфонов.

В рамках задачи, я разрабатываю мобильное приложение для ОС Android, в связи с этим ниже представлены описания нескольких платформ для разработки мобильных приложений под данную операционную систему [15].

1. **Appery.io** [3] – облачный сервис для разработки мобильных приложения для ОС Android. Имеет open-source фреймворк Apache Cordova (Phone Gap) и встроенные компоненты. Может быть подключен к разнообразному REST API, обеспечивает добавление облачной базы данных и связь ее с приложением, одновременно сохраняя показатели.

Важно, что Appery подразумевает создание собственных плагинов, можно наладить сотрудничество в реальном времени с командой разработчиков, бизнес-пользователями и клиентами.

2. **Android Studio** [1] – официальная среда разработки под Android. В целом очень хорошая среда разработки где встроены все необходимые компоненты, такие как intellij idea, поддержка gradle, встроенный эмулятор для тестирования приложения на различных устройствах.

3. **Фреймворк «Meteor»** [6] – веб-платформа на языке JavaScript [7], предназначенная для разработки Web-приложений реального времени, а также преобразованием их в мобильное приложение при помощи внут-

реннего модуля. Для связи с современными браузерами используется протокол Distributed Data Protocol, поддерживаемый с помощью WebSocket'ов, либо, если поддержки веб-сокетов и DDP нет - AJAX.

Код Meteor работает поверх node.js. Ядром Meteor является протокол DDP. Он предназначен для работы с коллекциями JSON-документов, позволяя легко создавать, обновлять, удалять, запрашивать и просматривать их. По умолчанию в качестве хранилища таких документов используется MongoDB.

Одна из важнейших особенностей платформы состоит в том, что она позволяет использовать один и тот же код как на стороне сервера, так и на стороне клиента. Между сервером и клиентом, как правило, передаются данные, а не HTML-код.

В связи с гибкими функциональными особенностями фреймворка Meteor, был выбран именно этот инструмент для разработки мобильного приложения. Он полностью удовлетворяет требования для взаимодействия с платформой для разработчика «Eviot Dev» и веб-сервисом «Eviot Web», так как данные инструменты также реализованы при помощи Meteor.

2. ПРОЕКТИРОВАНИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ УПРАВЛЕНИЯ БЕСПРОВОДНЫМИ УСТРОЙСТВАМИ ИНТЕРНЕТА ВЕЩЕЙ

2.1. Анализ требований

В ходе проектирования приложения были определены функциональные возможности, предоставляемые пользователю.

Определим основных актеров, взаимодействующих с системой.

Авторизированный пользователь – это пользователь, использующий мобильное приложение для управления купленным устройством интернета вещей.

Неавторизированный пользователь – пользователь мобильного приложения, у которого нет доступа к функционалу системы, кроме регистрации.

Диаграмма вариантов использования мобильного приложения приведена на рис. 4. С мобильным приложением взаимодействуют 2 актера: авторизированный пользователь и неавторизированный пользователь.

Представленные актеры могут совершать следующие действия.

Неавторизированный пользователь:

- регистрация пользователя в системе – происходит регистрация нового пользователя в системе, используя логин, email и пароль, и пользователь переходит в режим авторизованного пользователя.

Авторизированный пользователь:

- редактирование профиля пользователя – действие происходит на странице профиля, где имеется возможность сменить имя и email пользователя;
- просмотр списка устройств – пользователь может просмотреть список установленных устройств
- просмотр детальной страницы устройства – пользователь может просмотреть детальную информацию устройства, где реализован функционал управления между умным устройством и смартфоном;

- управление устройством – пользователь может управлять устройством на детальной странице устройства;
- настройка Wi-Fi данных – для соединения с умным устройством используется Wi-Fi соединение, поэтому пользователь, прежде чем добавить первое устройство, должен заполнить данные SSID и пароль от Wi-Fi;
- добавление нового устройства – пользователь может добавить новое устройство в список всех доступных устройств;
- просмотр справочной информации – пользователь может просмотреть справочную информацию о приложении и инструкции его использования.

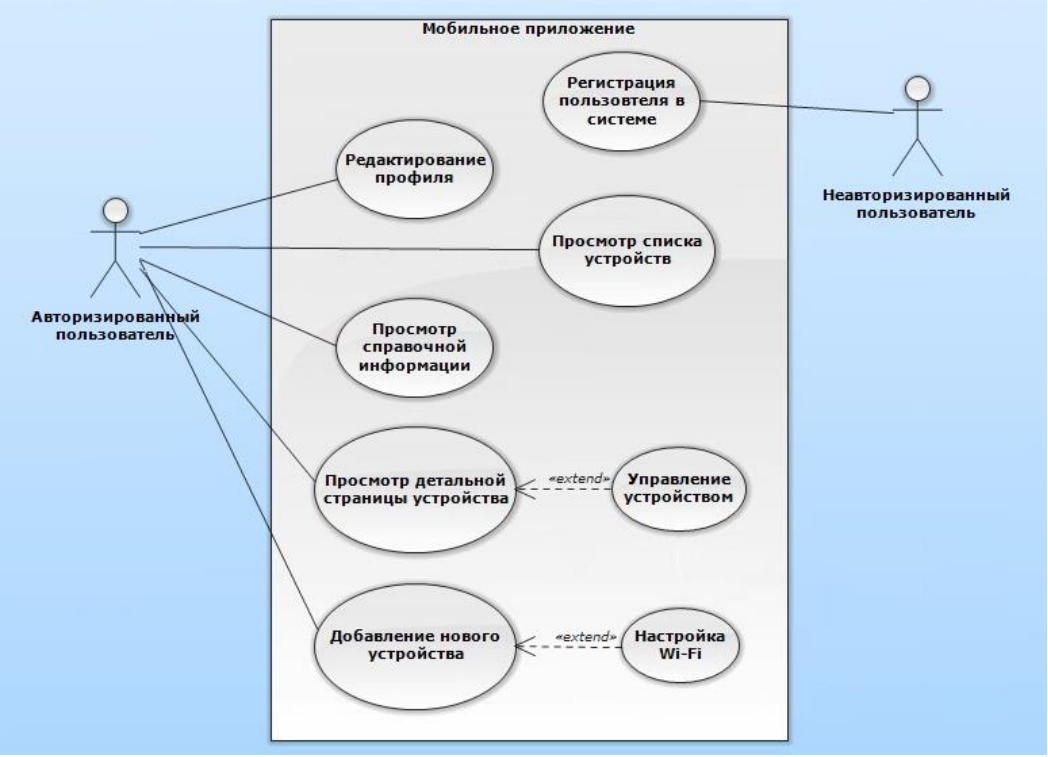

Рис. 4. Диаграмма вариантов использования мобильного приложения

2.2. Архитектура мобильного приложения

Архитектура мобильного приложения – общая схема работы мобильного приложения, состоящая из различных архитектурных слоев [12].

Концепция слоев - модель, используемая разработчиками программного обеспечения для разделения сложных систем на более простые части.

На рис. 5 представлена архитектура мобильного приложения.

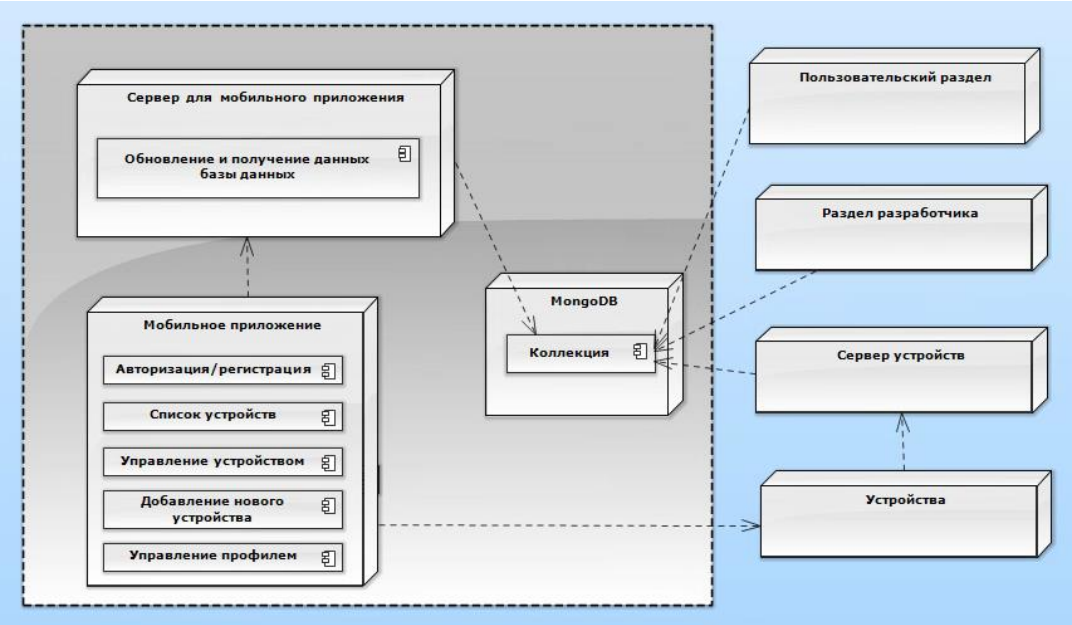

Рис. 5. Архитектура мобильного приложения

Узел «Мобильное приложение» представляет собой мобильное приложение, в котором доступны следующие возможности для пользователя:

1) авторизация или регистрация – пользователь может войти под своим логином и паролем в личный профиль, иначе зарегистрироваться;

2) список устройств – пользователь может добавить в список новое устройство либо просмотреть текущий список доступных устройств;

3) управление устройством – пользователь может зайти на внутреннюю страницу устройства и управлять им;

4) добавление нового устройства – пользователь может добавить новое устройство в список доступных устройств;

5) управление профилем – пользователь может задать себе имя, телефон, данные от Wi-Fi.

Данный узел взаимодействует с «Сервером для мобильного приложения».

Узел «Сервер для мобильного приложения» взаимодействует с базой данных MongoDB для обновления и получения новых данных из базы данных.

Мобильное приложение взаимодействует с устройствами.

База данных MongoDB осуществляет механизм хранения и доступа к коллекциям мобильного приложения.

2.3. Организация системы хранения данных на устройстве

Фреймворк Meteor использует не реляционную базу данных MongoDB, которая использует в себе хранение данных с помощью коллекций в формате JSON.

Хранение текущей сессии пользователя на мобильном устройстве реализовано с помощью MongoDB на базе технологии localStorage.

Технология известна как веб-хранилище, простое хранилище, или его альтернативный интерфейс хранилище сессии, этот интерфейс API обеспечивает хранение пары ключ/значение и доступны в реализациях WebView [11].

Согласно [8], каждый объект хранения связан со списком пар «ключ-значение» при его создании. Несколько отдельных объектов, реализующих Интерфейс хранения, могут быть одновременно связаны с одним и тем же списком пар «ключ-значение».

Атрибут length должен возвращать количество пар «ключ-значение», присутствующих в списке, связанном с объектом.

Метод key (n) должен возвращать имя N-го ключа в списке. Порядок ключей определяется агентом пользователя, но должен быть согласован внутри объекта, если количество ключей не изменяется. Таким образом, добавление или удаление ключа может изменить порядок ключей, но простое изменение значения существующего ключа не должно происходить. Если N больше или равно числу пар ключ-значение в объекте, то этот метод должен возвращать NULL.

Метод getItem(key) должен возвращать текущее значение, связанное с данным ключом. Если данный ключ не существует в списке, связанном с объектом, то этот метод должен возвратить null.

Метод setItem(key, value) должен сначала проверить, существует ли

пара ключ/значение с данным ключом в списке, связанном с объектом.

Если это не так, то новая пара ключ/значение должна быть добавлена в список, с заданным ключом и с его значением, установленным в value.

Если данный ключ существует в списке и его значение не равно value, то его значение должно быть обновлено до value. Если его предыдущее значение равно value, то метод не должен ничего делать.

Если не удалось задать новое значение, метод должен выдать исключение QuotaExceededError. Настройка может завершиться ошибкой, если, например, пользователь отключил хранилище для веб-приложений или если квота была превышена.

Можно сделать вывод, что данная технология является стандартизированной для локального хранения данных на мобильном устройстве.

3. РЕАЛИЗАЦИЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

3.1. Реализация хранения данных

MongoDB – является документно-ориентированной базой данных [9]. В ней хранятся коллекции мобильного приложения, описание которых приводится ниже.

В коллекции ***users*** хранится информация о пользователях. Коллекция состоит из следующих полей:

1) _id – тип String, идентификатор пользователя;

2) createdAt – тип Date, дата когда пользователь зарегистрировался;

3) username – тип String, логин пользователя;

4) name – тип String, имя пользователя;

5) roles – тип String[], список ролей, в которых состоит пользователь;

6) services – тип Object, используется системой авторизации пользователя, в данный момент доступна авторизация по логину и паролю.

В коллекции ***device*** хранится информация о подключенных к мобильному приложению устройств. Коллекция состоит из следующих полей:

1) _id – тип String, идентификатор устройства;

2) name – тип String, отображаемое имя устройства;

3) deviceTypeID – тип String, идентификатор типа устройства;

4) owner – тип String, идентификатор пользователя, который владеет устройством;

5) versionID – тип String, идентификатор версии типа устройства;

6) online – тип Boolean, если устройство подсоединено, то true;

7) createdAt – тип Date, когда устройство было подключено.

В коллекции ***deviceType*** хранится информация о типах устройств. Коллекция состоит из следующих полей:

1) _id – тип String, идентификатор;

2) name – тип String, отображаемое имя;

3) owner – тип String, идентификатор пользователя;

4) createdAt – тип Date, дата создания.

В коллекции ***deviceTypeVersion*** хранится информация о версии типа устройства. Коллекция состоит из следующих полей:

1) _id – тип String, идентификатор;

2) deviceTypeID – тип String, идентификатор типа устройства;

3) version – тип String, отображаемое название версии;

4) interfaces – тип Object, ассоциативный массив из объектов, в которых содержатся следующие поля:

a) typeID – тип String, идентификатор типа интерфейса;

b) label – тип String, отображаемое название;

c) options – тип Object, значение дополнительных опций в виде ключ => значение;

d) x и y – тип Integer, координаты отображение элемента на панели редактирования;

e) binds – тип Object, список привязки к другим элементам интерфейса.

5) page – тип Object, настройки отображения детальной страницы устройства. Состоит из следующих полей:

a) html – тип String, текст в виде HTML разметки, с использованием препроцессора Blaze, для возможности встраивания UI элементов;

b) js – тип String, JavaScript код, который будет выполняться при переходе на детальную страницу устройства;

c) less – тип String, CSS стили детальной страницы, с использованием препроцессора LESS.

6) createdAt – тип Date, дата создания.

В коллекции ***deviceInterfaceType*** хранится информация о типе интерфейсе взаимодействия с устройством. Коллекция состоит из следующих полей:

1) _id – тип String, идентификатор;

2) label – тип String, отображаемое название;

3) description – тип String, краткое описание;

4) author – тип String, идентификатор пользователя;

5) inputs – тип Array, список входных переменных;

6) outputs – тип Array, список выходных переменных;

7) options – тип Object, список параметров, которые требуются для работы типа интерфейса.

8) code – тип String, код на языке JavaScript, который будет выполняться на сервере устройств;

9) createdAt – тип Date, дата создания.

В коллекции ***deviceUITemplate*** хранится информация о UI элементах. Коллекция состоит из следующих полей:

1) _id – тип String, идентификатор;

2) name – тип String, отображаемое название;

3) author – тип String, идентификатор пользователя;

4) js – тип String, JavaScript код шаблона фукнции, который должен вернуть строку в HTML;

5) jsGlobal – тип String, JavaScript код, который дополнительно выполняется на странице, где находится данный UI элемент;

6) less – тип String, CSS стили для UI элемента с препроцессором LESS;

7) createdAt – тип Date, дата создания.

Итого в базе данных хранится 6 коллекций мобильного приложения.

3.2. Реализация основной функциональности приложения

Для разработки мобильного приложения потребовалось установить несколько дополнительных пакетов, которые реализуют важные функциональные особенности, а именно:

1) accounts-base и accounts-password – официальные пакеты Meteor для работы с авторизацией/регистрацией пользователей и добавлением данных в базу данных;

2) chrismbeckett:toastr – плагин предназначен для отображения уведомлений информации, успешного или не успешного действий;

3) raix:handlebar-helpers – этот пакет обеспечивает простой способ использования сеансов и коллекций в среде шаблонов Meteor Spacebars;

4) JQuery – библиотека JavaScript, фокусирующаяся на взаимодействии JavaScript и HTML;

5) alexwine:bootstrap-4 – инструмент для разработки с гибкой сетчатой системой [5];

6) fortawesome:fontawesome – иконочные шрифты для внешнего вида мобильного приложения;

7) iron:router – гибкая маршрутизация между страницами.

Meteor фреймворк основан на шаблонной среде разработки, исходя из этого для мобильного приложения можно описать шаблоны, используемые в приложении, ниже:

1) login.html – страница для авторизации пользователя;

2) register.html – страница для регистрации нового пользователя;

3) eviotlayout.html – коренной шаблон, под которую реализована маршрутизация;

4) header.html – статичная шапка мобильного приложения;

5) footer.html – статичное меню прикрепленное к нижней части экрана мобильного приложения;

6) account.html – страница профиля пользователя;

7) checkdevice.html – страница добавления нового умного устройства;

8) detail.html – детальная страница устройства, парсится из базы данных под определенный id устройства;

9) information.html – информационная справка о приложении;

10) listofdevice.html – список доступных устройств для пользователя;

11) notloggingin.html – страница для отображения страницы не вошедшему в аккаунт пользователю;

12) nulldevices.html – страница для отображения с приветствием и если доступных устройств в списке равно нулю;

13) settings.html – страница с настройками мобильного приложения;

14) setfiwifi.html – страница для установления настроек Wi-Fi роутера.

Для реализации функциональных возможностей был использован язык программирования JavaScript, наиболее важные функции приложения и их реализация приведены ниже.

1. Авторизация пользователя. Пользователь вводит данные в поля логин и пароль, далее обработчик событий отправляет запрос в базу данных о существовании пользователя, если успешно – пользователь входит в систему.

Листинг 1. Авторизация

```
//login.js
Template.login.onCreated (function () {
  if (Meteor.loggingIn() || Meteor.userId()) {
    Router.go('/listofdevice');
  }
})

Template.login.events({
    'click #login-button': function(e){
        let username = $('#login-username').val();
        let password = $('#login-password').val();

        Meteor.loginWithPassword (username,password,function(e){
          if (e) {
            toastr["error"](e.reason, "Авторизация")
          } else {
            Router.go('/listofdevice');
      }
        });
    }
});
```

2. Регистрация пользователя. При регистрации пользователя данные собираются с полей ввода e-mail, логина и пароля, далее обработчик событий регистрирует нового пользователя в системе, добавляя его в базу данных в коллекцию users.

Листинг 2. Регистрация

```
//register.js
Template.register.events({
```

```
'click #register-btn': function(e){
    let email = $('#email').val();
    let username = $('#username').val();
    let password = $('#password').val();

    console.log(email,username,password);
    Accounts.createUser({email: email, username: username, password: pass-
word}, function(e) {
      if (e) {
        // console.log(e);
        toastr["error"](e.reason, "Регистрация")
      } else {
        Router.go('/');
      }
    });}
});
```

3. Маршрутизация между страницами. Для использования маршрутизации между страницами используется дополнительный плагин router, в котором описан метод route для отображения страницы без ее перезагрузки.

Листинг 3. Маршрутизация между страницами

```
//router.js
Router.configure({
  layoutTemplate: 'eviotlayout',
  waitOn: function() {
    return [Meteor.subscribe('deviceUITemplates')]
  }
});
Router.route('/', {name: 'login', layoutTemplate: ''});
Router.route('/register', {name: 'register', layoutTemplate: ''});
Router.route('/listofdevice', {
  name: 'listofdevice',
  waitOn: function() {
    return [Meteor.subscribe('devices')];
  },
  data: function() {
    var devices = DeviceCollection.find({owner: Meteor.userId()});
    return {
      devices: devices,
    };
```

```
});
Router.route('/searchdevice', {name: 'searchdevice'});
Router.route('/account', {name: 'account'});
Router.route('/settings', {name: 'settings'});
Router.route('/setwifi', {name: 'setwifi'});
Router.route('/information', {name: 'information'});
Router.route('/nulldevices', {name: 'nulldevices'});
Router.route('/notloggingin', {name: 'notloggingin'});
Router.route('/testpage', {name: 'testpage'});
Router.route('/checkdevice', {name: 'checkdevice'});
Router.route('/devicePage', {name: 'devicePage'});
Router.route('/detail:_id', {
    name: 'detail',
    waitOn: function() {
        return [
            Meteor.subscribe('device', this.params._id),
            Meteor.subscribe('deviceTypeFromDeviceID', this.params._id),
            Meteor.subscribe('deviceTypeVersionFromDeviceID',
this.params._id),
        ];
    },
    data: function() {
        var result = DeviceCollection.findOne(this.params._id);
        if(result) {
            result['deviceType'] = DeviceTypeCollec-
tion.findOne(result.deviceTypeID);
            result['version'] = DeviceTypeVersionCollec-
tion.findOne(result.versionID);
        }
        return result;
    }
});
```

4. Добавление нового устройства с помощью отправки запросов на сервер устройства. Для добавления нового устройства мы отправляем запрос на устройство и проверяем, является ли данное устройство поддерживаемым на платформе, далее устанавливаем для устройства данные сети Wi-Fi, затем оно регистрируется в базе данных, где устройству присваивается UserId, в дальнейшем оно автоматически перезагружается и тем самым становится доступным в списке устройств.

Листинг 4. Добавление нового устройства

```
//checkdevice.js
Template.checkdevice.events({
    'click #check'(event, template) {
        var server = 'http://192.168.0.1'; // ip адрес устройства
        wait();
        $.get(server + '/is-eviot-device', function(data) { // является ли
это поддерживаемым устройством
            if(data != 'ok') {
                return fail('is-eviot-device not ok: ' + data);
            }
            $.get(server + '/set-wifi?ssid=' + Meteor.user().ssid_wifi +
'&pass=' + Meteor.user().password_wifi, function(data) { // устанавливаем
данные нашей wifi сети
                if(data != 'ok') {
                    return fail('set-wifi not ok: ' + data);
                }
                $.get(server + '/set-user-id?id=' + Meteor.userId(), func-
tion(data) { // передаем id текущего пользователя
                    if(data != 'ok') {
                        return fail('set-user-id not ok: ' + data);
                    }
                    $.get(server + '/do-registration', function(data) { //
говорим устройству, чтобы оно зарегистрировалось в платформе
                        if(data != 'ok') {
                            return fail('do-registration not ok: ' + data);
                        }
                        setTimeout(function() { // ждем 10 секунд
                            $.get(server + '/is-registration', func-
tion(data) { // проверяем результат регистрации
                                if(data != 'ok') {
                                    return fail('is-registration not ok: '
+ data);
                                }
                                $.get(server + '/restart', function(data) {
// перезагружаем устройство
                                    success();
                                })
                            });
                        }, 20000);
                    }).fail(function(err) {
```

```
                        fail('do-registration failed ' +
JSON.stringify(err, null, 4));
                    })
                })
            })
        }).fail(function(err) {
            fail('is-eviot-device fail ' + JSON.stringify(err, null, 4));
        })
    }
});
```

5. Отображение детальной страницы устройства. Для рендеринга детальной страницы устройства мы регистрируем «хелпер» compileView в качестве функции, которая принимает на вход данные об устройстве. Внутри нее происходит компиляция HTML кода детальной страницы устройства. Компилируются и применяются CSS стили на страницу в мобильном приложении, также возвращается из функции HTML разметка.

Листинг 5. Отображение детальной страницы устройства

```
//detail.js
Template.detail.helpers({
  getTitle: function () {
    var CurrentDevice = DeviceCollec-
tion.findOne(Router.current().params._id);
    if (CurrentDevice){
      return CurrentDevice.name;
    };
  },
  compileView(data) {
      var html = data.version.page.html;
      var deviceView = new Blaze.View(
          'deviceView',
          function() {
              var view = this;
              return (eval(SpacebarsCompiler.compile(html)))();
          }
      );
      var renderDiv = $('<div></div>')[0];
      Blaze.renderWithData(deviceView, data, renderDiv);
      var _less = data.version.page.less;
      if(_less) {
```

```
            less.render(_less).done(function(result) {
                var style = '<style id="devicePageCss">'
                        + result.css
                    + '</style>';
                $('style#devicePageCss').remove();
                $('body').append($(style))
            });
        }
        return renderDiv.innerHTML;
    },

    echo(val) {
        return val;
    }
});
```

Таким образом, в мобильном приложении получилось 14 различных шаблонов и 11 экранов, для использования JavaScript была задействована библиотека JQuery, а также дополнительные плагины, входящие в состав Meteor.

4. ТЕСТИРОВАНИЕ МОБИЛЬНОГО ПРИЛОЖЕНИЯ

4.1. Функциональное тестирование мобильного приложения

Для функционального тестирования мобильного приложения были выбраны все функции, определенные в функциональных требованиях при проектировании приложения. Результаты тестирования представлены в табл. 1. Тестирование производилось на двух устройствах собственной сборки с поддержкой wi-fi: лампы с диммером и электрической розетки.

В устройствах встроен микроконтроллер ESP8266, в который встроен собственный Wi-Fi модуль.

Табл. 1. Сведения о проведенном функциональном тестировании мобильного приложения

№.	Функция	Ожидаемый результат	Полученный результат	Заключение
1.	Регистрация нового пользователя с помощью e-mail, логина, пароля и добавление данных в коллекцию users	Новый пользователь зарегистрирован	Новый пользователь зарегистрирован	Функция работает
2.	Авторизация пользователя с помощью логина и пароля	Пользователь вошел в аккаунт	Пользователь вошел в аккаунт	Функция работает
3.	Добавление нового устройства, при условии, что пользователь присоединился к wi-fi модулю устройства	Новое устройство добавлено в список доступных устройств	Новое устройство добавлено в список доступных устройств	Функция работает
4.	Отображение детальной информации устройства для управления, которое парсится из базы данных под определенным id устройства	Детальная страница отображена для определенного устройства	Детальная страница отображена для определенного устройства	Функция работает
5.	Работоспособность кнопок включения/выключения устройства, ползунков для изменения яркости и таймера	Кнопки и ползунки изменяют параметры устройства	Кнопки и ползунки успешно изменяют параметры устройства	Функция работает

Продолжение табл. 1

№.	Функция	Ожидаемый результат	Полученный результат	Заключение
6.	Установка настроек wi-fi для поиска устройства и добавления	Данные wi-fi добавляются в базу данных к пользователю	Данные wi-fi добавляются в базу данных к пользователю	Функция работает
7.	Редактирование профиля устройства при помощи изменения имени, телефона	Данные обновляются в базе данных	Данные обновляются в базе данных	Функция работает
8.	Выход из профиля при нажатии на кнопку «Выйти» в профиле пользователя	Пользователь вышел из профиля на начальную страницу приложения	Пользователь вышел из профиля на начальную страницу приложения	Функция работает
9.	Открытие страницы информационной справки о приложении	Отображена справочная информация	Отображена справочная информация	Функция работает
10.	Отображение страницы приветствия и при условии, что доступных устройств нуль	Отображена приветственная страница	Отображена приветственная страница	Функция работает
11.	Отображение всплывающих подсказок, если пользователь вводит неверные данные от аккаунта или при регистрации	Всплывающие подсказки отображаются	Всплывающие подсказки отображаются	Функция работает
12.	Отображение пользователю страницы о невыполненном входе в аккаунт, если он принудительно попал на страницу для отображения только вошедшим в аккаунт пользователям	Страница о невыполненном входе отображена	Страница о невыполненном входе отображена	Функция работает
13.	Удаление доступного устройства из списка устройств и базы данных	Устройство удалено из списка доступных устройств и базы данных	Устройство удалено из списка доступных устройств и базы данных	Функция работает

Окончание табл. 1

№.	Функция	Ожидаемый результат	Полученный результат	Заключение
14.	Отображение страницы настроек Wi-Fi, при попытке добавить новое устройство и данные не заполнены	Страница с настройками Wi-Fi отображена	Страница с настройками Wi-Fi отображена	Функция работает

Таким образом, функциональное тестирование мобильного приложения подтвердило, что все функции успешно работают.

4.2. Интеграционное тестирование мобильного приложения и веб-сервиса

При проведении интеграционного тестирования мобильного приложения и веб-сервиса были выбраны общие функциональные особенности, которые должны изменяться как на стороне мобильного приложения, так и на стороне веб-сервиса в автоматическом режиме:

1) при добавлении нового устройства в список доступных устройств, умное устройство успешно регистрируется в базе данных, перезагружается и добавляется в список доступных устройств, как в мобильном приложении, так и в пользовательском разделе веб-сервиса;

2) при изменении внутренних функций на странице детальной информации устройства, т.к. изменение состояния работоспособности устройства (вкл/выкл), параметры ползунка и др. особенностей, данные успешно изменяются как на стороне мобильного приложения, так и на стороне пользовательского раздела веб-сервиса;

3) при регистрации нового пользователя внутри мобильного приложения, пользователь успешно добавляется в общую базу данных MongoDB в коллекцию users, следовательно, он может зайти в свой профиль как с помощью мобильного приложения, так и с помощью пользовательского раздела веб-сервиса;

4) при изменении данных профиля и настроек Wi-Fi, данные

успешно обновляются в общей базе данных MongoDB в коллекции users.

Можно сделать вывод, что интеграционное тестирование проведено успешно и мобильное приложение работает в связке со всем веб-сервисом, что дает хорошую гибкость всей системе для использования и управления устройствами интернета вещей.

Проверка возможностей управления беспроводными устройствами интернета вещей сторонних производителей не осуществлялась, планируется в дальнейшем.

Скриншоты работающего мобильного приложения представлены на рис. 1–7 приложения.

ЗАКЛЮЧЕНИЕ

В настоящее время, рынок IoT устройств только набирает популярность. Каждая новая компания имеет свою идеологию и работает в основном на собственной закрытой платформе для разработки интернета вещей, не давая другим компаниям на их базе усовершенствовать свои устройства в «умные» устройства. Но данный метод и технология может позволить существенно модернизировать и ускорить темпы роста интернета вещей.

Управление устройствами интернета вещей является важной отраслью в становлении нового сегмента технологий умных вещей, помогающих человеку в различных жизненных обстоятельствах управлять устройствами не только находясь вблизи них, но также на больших расстояниях.

Целью работы является создание мобильного приложения для управления устройствами интернета вещей, взаимодействующего с платформой для разработки умных устройств «Eviot Dev» и веб-сервисом «Eviot Web» с помощью веб-фреймворка Meteor. Для достижения данной цели были решены следующие задачи:

1) произведен обзор и изучен фреймворк Meteor;
2) изучена работа и особенности базы данных NoSql MongoDB;
3) определены требования и спроектировано мобильное приложение;
4) спроектирована база данных;
5) реализовано и протестировано мобильное приложение.

Все поставленные задачи были решены, цель достигнута.

Разработанное мобильное приложение имеет перспективы дальнейшего развития. С учетом роста конкуренции и правом быть самой передовой компанией, возникает потребность в расширении функционала системы.

В перспективе планируется реализовать следующие возможности:

- интеграция с действующими системами умного дома, такими как Apple HomeKit, Google Home, Amazon Alexa, Redmond и др.;

- аналитика действий пользователя и подбор оптимальных параметров для пользователя;
- создание push-уведомлений;
- создание общих сценариев управления устройствами в экосистеме умного дома пользователя;
- внедрение голосового ассистента.

СПИСОК ЛИТЕРАТУРЫ

1. Android Developers | SDK Tools. [Электронный ресурс] URL: https://developer.android.com/studio/ (дата обращения: 01.05.2018).
2. e-Control – Приложение в Google Play. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.readyforsky&hl=ru (дата обращения: 01.05.2018).
3. Enterprise Mobile App Builder. [Электронный ресурс] URL: https://appery.io/ (дата обращения: 01.05.2018).
4. eWeLink – Приложение в Google Play. [Электронный ресурс] URL: https://play.google.com/store/apps/details?id=com.readyforsky&hl=ru (дата обращения: 01.05.2018).
5. Introduction | Bootstrap. [Электронный ресурс] URL: http://getbootstrap.com/docs/4.1/getting-started/introduction/ (дата обращения: 3.03.2018).
6. Introduction | Meteor Guide. [Электронный ресурс] URL: https://guide.meteor.com/ (дата обращения: 01.03.2018).
7. JavaScript. [Электронный ресурс] URL: https://www.javascript.com/learn/javascript/ (дата обращения: 03.03.2018).
8. LocalStorage на пальцах. [Электронный ресурс] URL: https://tproger.ru/articles/localstorage/ (дата обращения: 30.04.2018).
9. MongoDB Documentation. [Электронный ресурс] URL: https://docs.mongodb.com/ (дата обращения: 10.03.2018).
10. Ready for Sky – Приложение в Google Play. [Электронный ресурс] URL: https://goo.gl/RQuY93 (дата обращения: 01.05.2018).
11. Web Storage (Second Edition). [Электронный ресурс] URL: https://www.w3.org/TR/webstorage/#terminology (дата обращения: 30.04.2018).
12. Архитектура мобильного клиент-серверного приложения. [Электронный ресурс] URL: https://habr.com/post/246877/ (дата обращения: 07.05.2018).

13. Интернет Вещей. Безграничные возможности взаимодействия человека и машины. [Электронный ресурс] URL: http://www.ey.com/Publication/vwLUAssets/EY-mne-internet-of-things-rus/$File/EY-mne-internet-of-things-rus.pdf (дата обращения: 31.03.2018).

14. Рынок мобильных ОС: статистика за сентябрь 2017 г. [Электронный ресурс] URL: http://w7phone.ru/rynok-mobilnyx-os-statistika-za-sentyabr-2017-141927/ (дата обращения: 25.04.2018).

15. Топ-10 платформ для разработки мобильных приложений. [Электронный ресурс]. URL: https://robo-hunter.com/news/top-10-platform-dlya-razrabotki-mobilnih-prilojenii (дата обращения: 03.04.2018).

ПРИЛОЖЕНИЕ

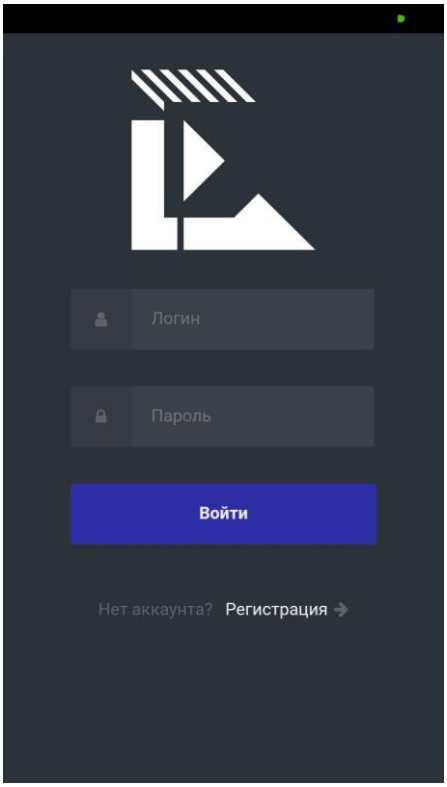

Рис. 1. Окно входа

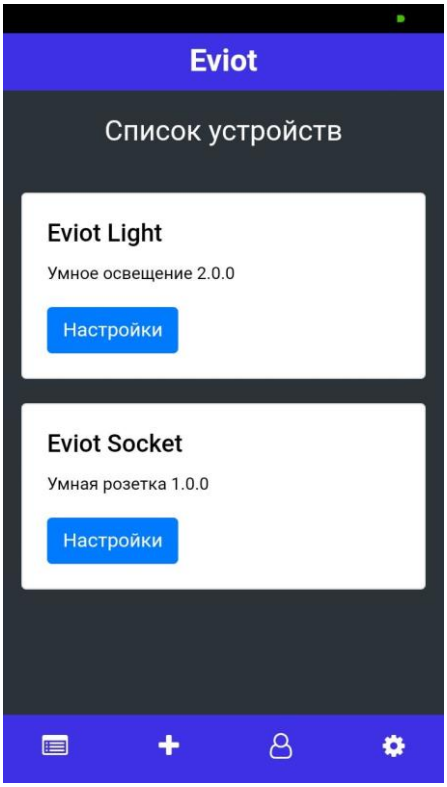

Рис. 2. Список доступных устройств

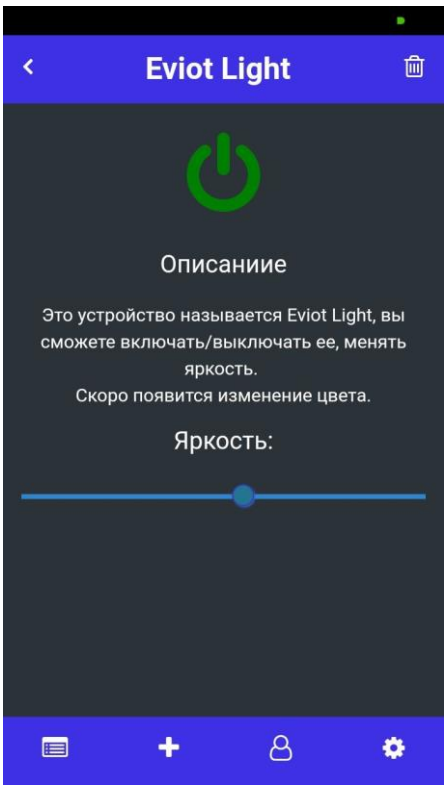

Рис. 3. Детальная страница устройства

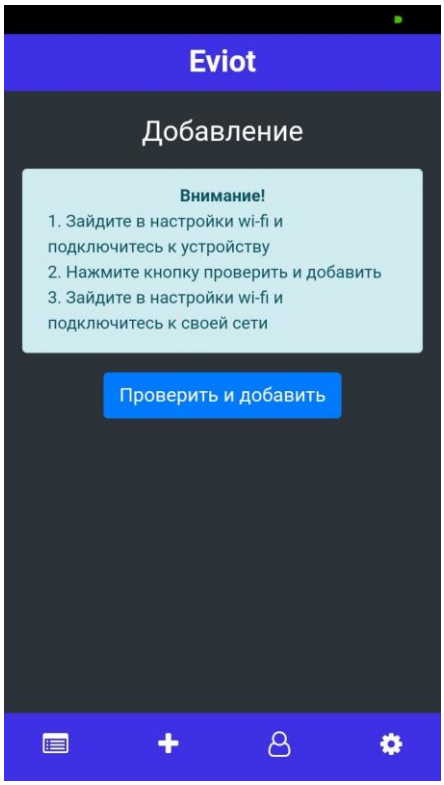

Рис. 4. Добавление нового устройства

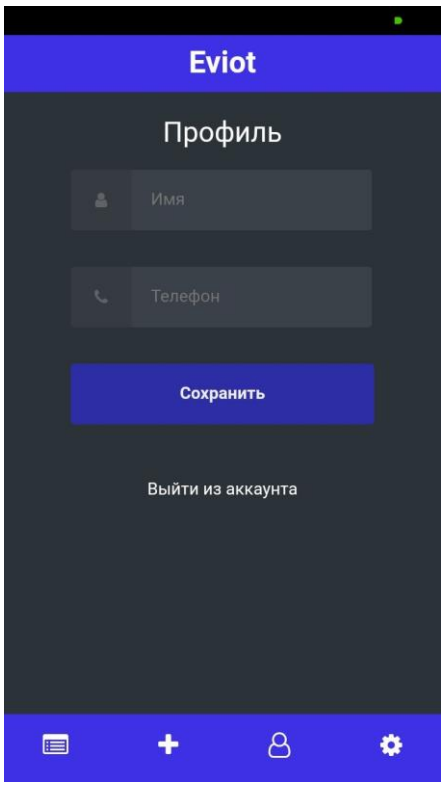

Рис. 5. Редактирование профиля пользователя

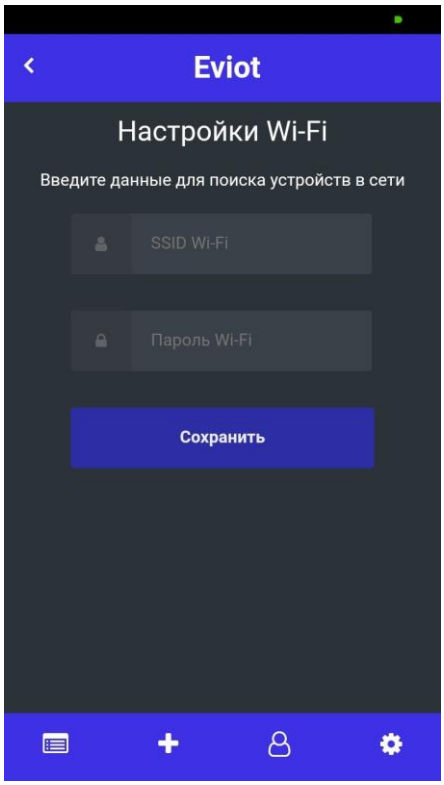

Рис. 6. Настройки Wi-Fi

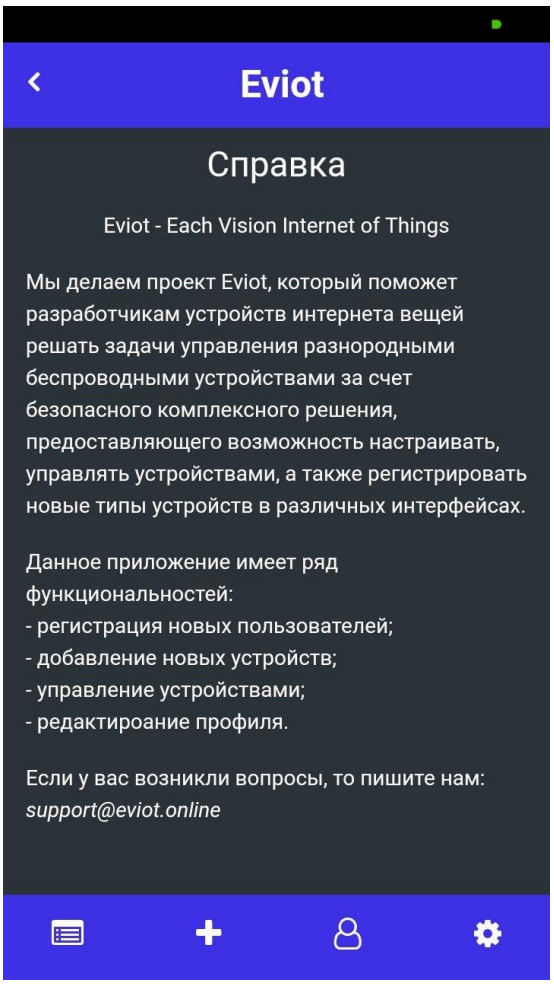

Рис. 7. Справочная информация