

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ  
УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра комплексной информационной безопасности электронно-  
вычислительных систем (КИБЭВС)

СБОР МИКРОКЛИМАТИЧЕСКИХ ПАРАМЕТРОВ В УЧЕБНЫХ  
АУДИТОРИЯХ

Отчет по результатам выполнения индивидуального проекта

IT Академии Samsung

Исполнители проекта

\_\_\_\_\_ К.В. Подойницын  
(подпись)

\_\_\_\_\_ Г.А. Астра  
(подпись)

«1» июля 2023 г.

Руководитель проекта

Старший преподаватель каф. КИБЭВС

\_\_\_\_\_ О.В. Пехов  
(подпись)

«1» июля 2023 г.

Томск 2023

Министерство высшего образования и науки Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования

ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра комплексной информационной безопасности электронно-  
вычислительных систем (КИБЭВС)

УТВЕРЖДАЮ

Заведующий кафедрой КИБЭВС доктор  
технических наук, профессор

\_\_\_\_\_ А.А. Шелупанов

«\_\_» \_\_\_\_\_ 2023 г.

ЗАДАНИЕ

на индивидуальный проект IT Академии Samsung  
студентам Подойницыну Кириллу Вадимовичу, Астра Григорию  
Алексеевичу группы 730-1 Факультета Безопасности.

1. Тема индивидуального проекта: «Сбор микроклиматических параметров в учебных аудиториях».
2. Цель проекта: разработать систему сбора климатических параметров в учебных аудиториях на базе технологии LoRa с клиент-серверным приложением.
3. Основные задачи проекта на этапах реализации:
  - Ознакомление с протоколом LoRaWAN;

- Разработка архитектуры и функционала web-приложения;
- Определение программных инструментов для разработки приложения;
- Написание клиент-серверного приложения.

4. Состав и содержание пояснительной записки (перечень, подлежащих проработке вопросов):

- 4.1. Общие разделы исполнить согласно требованиям ОС ТУСУР 01-2021.
- 4.2. Состав рубрик содержания основной части проекта:
  - Задачи, аудитория и актуальность проекта.
  - Архитектура и функционал web-приложения.
  - Рассмотрение существующих аналогов решения.
  - Реализация web-приложения.

Дата выдачи задания: «16» февраля 2023 г.

5. Срок сдачи студентами законченного проекта «1» июля 2023 г.

Руководитель проекта:

Старший преподаватель каф. КИБЭВС \_\_\_\_\_ О.В. Пехов  
(подпись)

Задание приняли к исполнению \_\_\_\_\_ К.В. Подойницын  
(подпись)

\_\_\_\_\_ Г.А. Астра  
(подпись)

## Реферат

Отчет содержит 73 страницы, 55 рисунков, 8 источников.

LORAWAN, LPWAN, IOT, БАЗОВАЯ СТАНЦИЯ, СБОР ПАРАМЕТРОВ В УЧЕБНЫХ АУДИТОРИЯХ, WEB-ПРИЛОЖЕНИЕ, ГРАФИК, УЧЕТНАЯ ЗАПИСЬ, ПОЛЬЗОВАТЕЛЬ, СЕРВЕР, БЕЗОПАСНОСТЬ.

Объекты исследования: системы интернета вещей.

Предмет исследования: способы проектирования IoT-сетей и создание IoT-систем.

Цель проекта: разработать систему сбора климатических параметров в учебном корпусе на базе технологии LoRa.

Пояснительная записка к групповой проектной работе выполнена в текстовом редакторе Microsoft Word 2019.

Оформлено в соответствии с ОС ТУСУР 01 – 2021. [1]

## Содержание

Введение .....	6
1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ .....	7
1.1 API документация .....	7
1.2 Протокол LoRaWAN.....	7
1.3 Архитектура Web-приложения .....	12
1.4 Определение инструментов для реализации приложения .....	13
2 ОБЗОР АНАЛОГОВ.....	17
2.1 Базовые станции .....	17
2.2 Датчики .....	20
3 АРХИТЕКТУРА ПРОЕКТА.....	22
4 ПРАКТИЧЕСКАЯ ЧАСТЬ .....	24
4.1 Обзор базовой станции .....	24
4.2 API-документация Vega server.....	26
4.3 Реализация web-приложения .....	31
5. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ ПО ИСПОЛЬЗОВАНИЮ СИСТЕМЫ. 52	
5.1. Работа с устройством по сбору параметров .....	53
5.2. Настройка базовой станции .....	55
5.3. Настройка сервера.....	56
5.4. Настройка приложения для администрирования .....	58
5.5. Настройка web-приложения.....	60
Заключение.....	63
Список использованных источников .....	64
Приложение А.....	65
Приложение Б .....	66

## Введение

Идея проекта заключается в создании системы по сбору микроклиматических параметров в учебных аудиториях ВУЗа.

Задачи проекта:

- Разработка архитектуры и функционала web-приложения;
- Определение программных инструментов для разработки приложения;
- Написание клиентского приложения.

Аудитория проекта: ВУЗ.

Актуальность проекта: в настоящее время в ТУСУРе контроль микроклиматических параметров в учебных аудиториях проводится в ручном режиме. Существуют нормы СанПиН 2.4.2.2821-10 «Санитарно-эпидемиологические требования к условиям и организации обучения в общеобразовательных учреждениях», согласно которым, например, температура должна составлять от  $+18^{\circ}\text{C}$  до  $+24^{\circ}\text{C}$  и отклонение от этих значений будет препятствием для проведения занятий. Данный проект позволит автоматизировать отслеживание параметров, соответственно, соблюдение данных требований станет намного проще.

# **1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ**

## **1.1 API документация**

API (Application Programming Interface — «программный интерфейс приложения») — описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой. Обычно входит в описание какого-либо интернет-протокола, программного каркаса или стандарта вызовов функций операционной системы. Часто реализуется отдельной программной библиотекой или сервисом операционной системы. Используется программистами при написании всевозможных приложений.

API упрощает процесс программирования при создании приложений, абстрагируя базовую реализацию и предоставляя только объекты или действия, необходимые разработчику.

## **1.2 Протокол LoRaWAN**

Спецификация LoRaWAN представляет собой сетевой протокол с низким энергопотреблением и глобальной сетью (LPWAN), разработанный для беспроводного подключения устройств с батарейным питанием к Интернету, нацеленный на ключевые требования Интернета вещей (IoT). В сети устройства обмениваются сообщениями данных (Data Messages), которые могут содержать в себе как данные приложений, так и MAC-команды, также одно сообщение может содержать оба вида информации [2].

Конструкция сообщения данных представлена на рисунке 1.

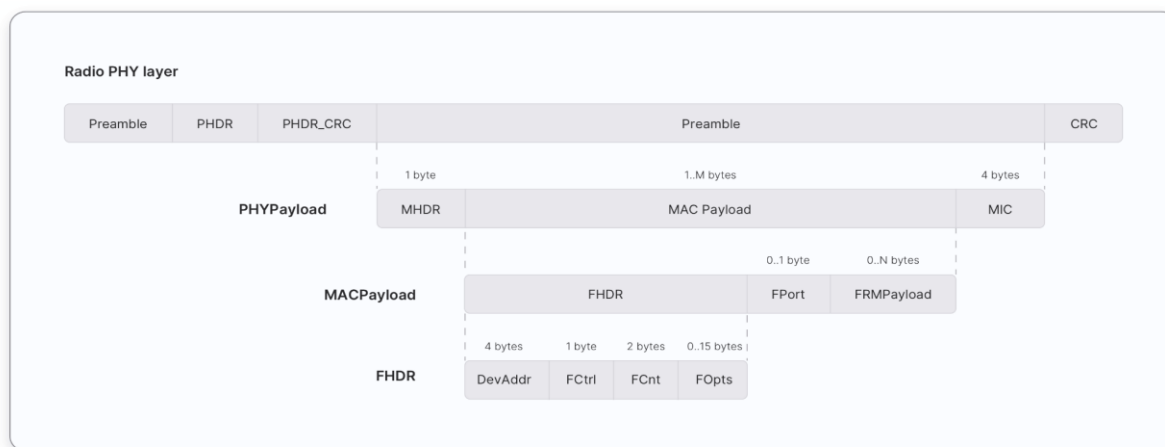


Рисунок 1 – Формат сообщения данных

Полезная нагрузка MAC сообщений данных (MACPayload) состоит из заголовка кадра (FHDR), далее следует необязательное поле порта (FPort) и необязательная полезная нагрузка кадра (FRMPayload).

FHDR состоит из следующих полей:

- DevAddr – локального адреса устройства в сети (4 байта);
- FCtrl – поля управления фреймом (1 байт);
- Fcnt – счетчика кадров (2 байта);
- FOpts – поля параметров кадра (0-15 байтов).

MAC-команды могут отправляться в поле параметров кадра (FOpts), или в поле полезной нагрузки кадра (FRMPayload) сообщения данных, но не в обоих одновременно.

Данные приложения могут быть отправлены в FRMPayload. Но это поле может содержать только один тип данных, либо команды MAC, либо данные приложения.

Если поле FRMPayload содержит команды MAC или данные приложения, оно должно быть зашифровано.



### 1.3 Безопасность протокола

LoRaWAN 1.0 использует ряд ключей безопасности: NwkSKey, AppSKey и AppKey. Все ключи имеют длину 128 бит.

NwkSKey (Network Session Key) используется для взаимодействия между узлом и сетевым сервером. Этот ключ используется для расчета и проверки кода целостности сообщения (Message Integrity Code – MIC). [4]

AppSKey (Application Session Key) используется для шифрования и дешифрования полезной нагрузки. Полезная нагрузка полностью зашифрована между узлом и сервером приложений.

Ключи NwkSKey, AppSKey при ОТАА генерируются при каждой процедуре присоединения. При этом они ни в какой момент времени не передаются, а генерируются отдельно на конечном устройстве и сервере.

Эти ключи являются результатом шифрования AES-128 с ключом AppKey. Идентичность данных ключей обеспечивается тем, что и конечное устройство, и сервер знают все параметры шифрования:

$$\text{NwkSKey} = \text{aes128\_encrypt}(\text{AppKey}, 0x01 \mid \text{AppNonce} \mid \text{NetID} \mid \text{DevNonce})$$

$$\text{AppSKey} = \text{aes128\_encrypt}(\text{AppKey}, 0x02 \mid \text{AppNonce} \mid \text{NetID} \mid \text{DevNonce})$$

0x01, 0x02 – номер ключа (1 – NwkSKey, 2 – AppSKey),

NetID – идентификатор сети. [3]

Случайными числами DevNonce и AppNonce обмениваются конечное устройство и сервер при процедуре присоединения. Они генерируются при каждом запросе присоединения и обеспечивают отличие сессионных ключей для каждого сеанса.

Ключ AppKey (Application Key) – это корневой ключ, специфичный для конечных устройств. Он используется только один раз – при процедуре присоединения. С его помощью подписывается обмен параметрами при

процедуре присоединения, также используется для получения сессионных ключей.

В сети IoT LoRaWAN используется многоуровневая система безопасности передачи данных:

Уровень 1. AES-шифрование на уровне приложения (между конечным устройством и сервером приложений) с помощью 128-битного переменного сессионного ключа AppSKey. Этот ключ хранится на конечном устройстве и на сервере приложений.

Уровень 2. AES-шифрование и проверка целостности сообщений на сетевом уровне (между конечным устройством и сетевым сервером) с помощью 128-битного переменного сессионного ключа NwkSKey. Этот ключ хранится на конечном устройстве и на сетевом сервере и недоступен клиенту.

Уровень 3. Стандартные методы аутентификации и шифрования интернет-протокола (IPsec, TLS и т.п.) при передаче данных по транспортной сети между узлами сети (базовая станция, сетевой сервер, Join-сервер, сервер приложений).

Сетевой сервер LoRaWAN хранит архив DevNonce. Если сервер получает запрос на присоединения с ранее использованным DevNonce (атака повторного воспроизведения), то он отклоняет запрос и не даёт устройству присоединиться к сети.

В сообщениях LoRaWAN используются счётчики (FCntUp и FCntDown) для защиты от отправки повторных данных. При активации устройства счетчики устанавливаются на 0. Каждый раз, когда устройство отправляет сообщение FCntUp увеличивается на единицу, а когда сервер отправляет – то увеличивается FCntDown. Если устройство или сервер получают сообщение со значением счетчика меньше или равной последнему, то данное сообщение игнорируется.

МІС забезпечує цілісність і підли́нність повідомлення. Код цілісності повідомлення вичислюється по всім полям повідомлення, а потім додається к самому повідомленню.

Безпечний обмін повідомленнями представлений на рисунку 2.

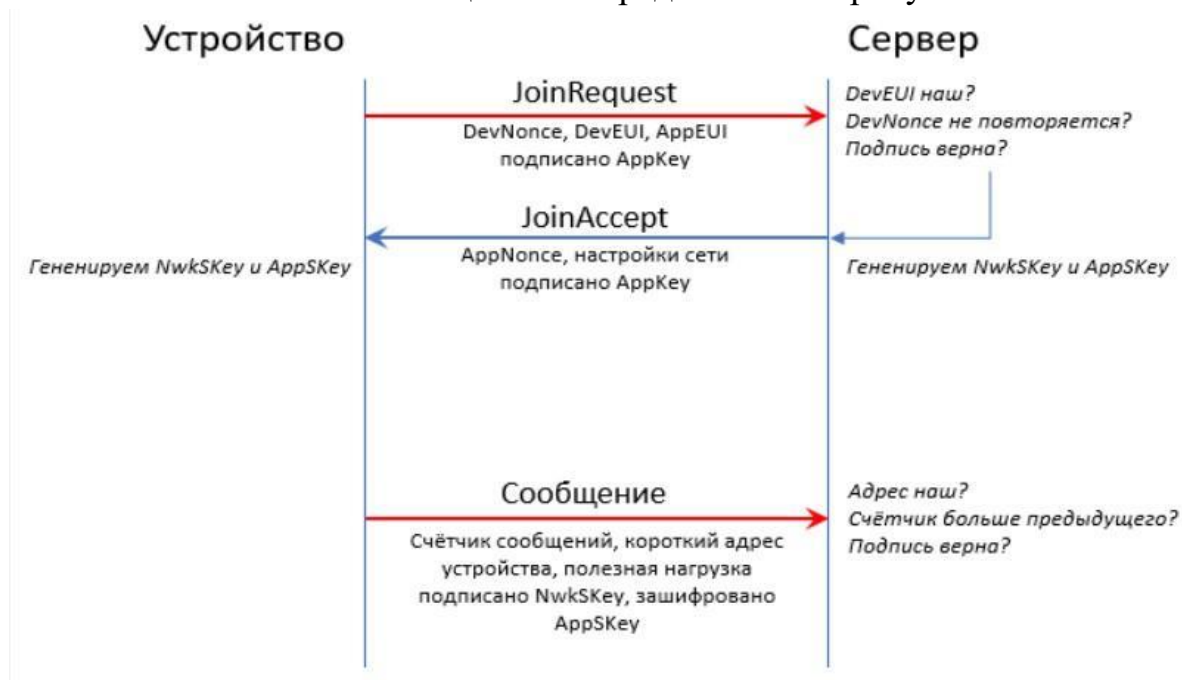


Рисунок 2 – Безпечний обмін повідомленнями в мережі LoRaWAN

### 1.3 Архитектура Web-приложения

Перед начало реализации приложения была разработана архитектура приложения и его функционал. Все это отображено на UML-диаграмме.

UML-диаграмма представлена на рисунке 3.

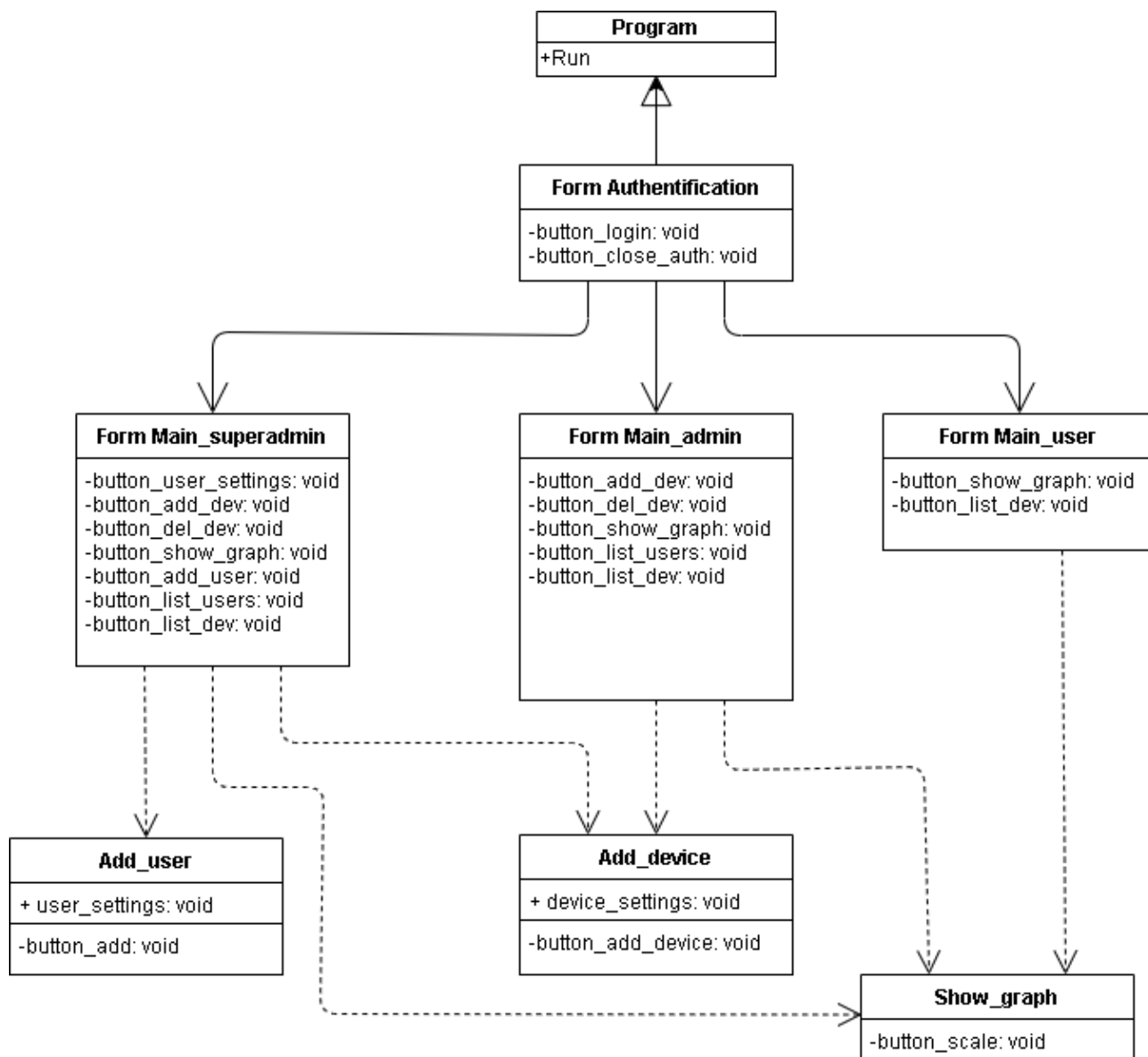


Рисунок 3 - UML-диаграмма приложения

Из диаграммы видно, что в зависимости от того, какой пользователь войдет в систему, ему будут доступны разные функции приложения. Так, обычному пользователю (Main\_user) из функций будет доступен просмотр списка подключенных устройств и возможность просмотра графика температур.

Для администратора (Main\_admin) из функций будет возможность просмотреть список пользователей, устройств и график температур, а также администратор будет иметь возможность добавлять или удалять устройства.

Полными правами будет обладать пользователь суперадмин (Main\_superuser), функционал у данного пользователя позволит удалять и добавлять как устройства, так и пользователей, также он сможет просматривать всё тоже самое, что и два предыдущих пользователя.

В соответствии с UML-диаграммой в web-приложении планируется создать следующие учетные записи:

- Обычный пользователь;
- Админ;
- Суперадмин.

Весь функционал, доступный пользователям данных учетных записей указан в диаграмме и описан ранее.

## **1.4 Определение инструментов для реализации приложения**

Для реализации приложения в рамках проекта необходимо было выбрать язык программирования для написания приложения, среду разработки, систему управления базами данных и фреймворк.

В качестве языка программирования был выбран Python из-за наличия возможности подключения библиотек, необходимых для работы с IoT Vega Server посредством использования API-функций.

При выборе среды разработки (IDE) выбор оказался между мультязычными средами с поддержкой языка программирования Python и средами, специально разработанными для работы с данным языком программирования. Однако, к наиболее известным мультязычным IDE с поддержкой Python относится Visual Studio, но в данный момент поддержка языка программирования доступна далеко не на всех операционных системах.

Помимо Visual Studio, есть Visual Studio Code, который поддерживает Python на всех операционных системах, но при этом является не средой разработки, а редактором кода в котором можно писать код.

Но по сравнению с IDE, редакторы кода имеют свои недостатки при работе с кодом:

- Отсутствие синтаксического анализатора языка программирования, упрощающего задачу написания кода с помощью указания ссылки на используемый метод или класс.
- Отсутствие подсказок с возможным продолжением кода, что заметно упрощает написание кода и освобождает память от запоминания всех имеющихся функций.

Однако, у них есть и плюсы, заключающиеся в том, что:

- Они менее требовательны к ресурсам системы.
- Возможность расширения функционала за счет плагинов.
- Поддержка многих языков программирования, даже тех, для которых нет удобной среды разработки.

Но опираясь на удобство написания кода при выборе среды разработки, было принято решение рассмотреть специально разработанные среды, к которым относятся PyCharm и Spyder.

Основной целью для выбора подходящей среды разработки является реализация web-приложения.

Обе среды предназначены для разработки на языке программирования python, при этом Spyder имеет полностью бесплатную основу распространения из-за открытого исходного кода, в то время как «PyCharm» имеет как бесплатную версию среды под названием «The Community Edition», так и платную версию «The Professional Edition», что нельзя отнести к какому-то плюсу одной из двух сред.

Также стоит отметить, что на обоих IDE есть возможность написания web-приложения.

Однако, если сравнивать по изначальному инструментарию после установки программ, то в PyCharm нет ненужных функций для реализации web-приложения, в то время как «Spyder» больше направлен на разработку приложений для анализа, обработки и представления данных в цифровой среде из-за встроенного пакетного менеджера «Anaconda».

Данный менеджер используется в анализе данных и машинном обучении. И при этом довольно быстро работает с библиотеками для математики, что не подходит под цели нашего приложения, и такой функционал будет попросту лишним. При этом данный менеджер со всем функционалом устанавливается вместе со средой разработки, что может повлиять на конечную скорость работы среды разработки. В то время как начальная версия «PyCharm The Community Edition» имеет только базовые инструменты, необходимые для разработки, однако тоже имеет поддержку менеджера «Anaconda», но его надо загружать отдельно. Поэтому отсутствие данного менеджера в основных функциях «PyCharm» облегчает навигацию по функционалу среды разработки и делает интерфейс программы более простым для освоения

На основе этого сравнения был сделан вывод, что среда разработки «PyCharm» больше подходит для разработки web-приложения, чем «Spyder», поскольку базовая версия является более простой и понятной по навигации в интерфейсе, не содержит функций, которые никак не будут относиться к разработке web-приложения, а будут только мешать и усложнять навигацию разработчику [4].

В качестве СУБД используется SQLite поскольку она является встроенной в ПО IOT Vega Server и для работы с ней не требуются настройки.

В качестве фреймворка, необходимо было подобрать популярный фреймворк из-за большого наличия материала по работе с ним, фреймворк, который изначально имеет встроенный отладчик, для удобного выявления ошибок, возможность проведения модульного тестирования и шаблонизатор, чтобы облегчить создания интерфейса сайта.

На основе этих параметров были подобраны фреймворки Flask, Web.py и Django.

Результат сравнения фреймворков представлен в таблице 1.

Таблица 1 - Сравнение фреймворков.

Фреймворк Критерий	Flask	Web.py	Django
Шаблонизатор	+	+	+
Встроенный отладчик	+	+	-
Модульное тестирование	+	-	-

По итогу сравнения по установленным критериям был выбран фреймворк Flask.



## 2 ОБЗОР АНАЛОГОВ

В разрабатываемой IoT-системе используются как базовая станция, так и конечное устройство, которое собирает и передает на сервер данные, а также пользовательское приложение, которое отображает полученные данные с сервера. Поэтому целесообразно было исследовать рынок на наличие аналогов системы в целом.

### 2.1 Базовые станции

#### 1. Robustel R3000 LG4LA [5]

На рисунке 4 приведено фото данной станции.



Рисунок 4 - R3000 LG от компании Robustel

Ниже приведены основные характеристики данной станции:

- Поддержка международных диапазонов частот LoRaWAN (433-434 МГц и 863-870 МГц).
- Наличие GSM модема 2G/3G/4G.
- Напряжение питания: 9...60 В постоянного тока.
- Наличие PoE.
- Габариты, мм 125 x 104 x 43,5.
- Вес, гр. 570.
- Мощность передачи сигнала – (+24,5) дБм.
- Рабочие температуры: -40...+75°C).
- Дальность связи – от 5 до 15 км.
- ГЛОНАСС/GPS – есть.
- Наличие следующих интерфейсов: ETHERNET: 2x 10/100 Мбит/с, 2x LAN или 1x LAN + 1x WAN, USB 2.0,. Есть возможность подключения антенн LoRa, GSM, ГЛОНАСС/GPS.
- Исполнение: IP30, металлический корпус.

Стоимость данной БС – 187245 руб., что в несколько раз превосходит используемую в проекте.

## 2. Вега БС-1.2 [6]

На рисунке 5 представлено фото данной БС.



Рисунок 5 - Вега БС-1.2

Ниже приведены основные характеристики данной станции:

- Напряжение питания –Passive PoE 4.5(+) 7.8(-) 15 Вт.
- Габариты, мм 192 x 183 x 75.
- Вес, гр. 1230.
- Рабочие температуры: -40...+70°C).
- Наличие Ethernet, USB-порта.
- Исполнение: IP67.

Стоимость данной БС – 36660 руб.

## 2.2 Датчики

В качестве готовых решений конечных устройств, которые бы передавали данные на сервер, были рассмотрены существующие датчики по снятию температуры, отправляющие показания по протоколу LoRaWAN.

### 1. Saures Wi-Fi [7]

На рисунке 6 представлено фото данного устройства.



Рисунок 6 - Saures Wi-Fi

Основные характеристики приведены ниже:

- Отправка данных по Wi-Fi;
- Частота радиоканала 2.4 ГГц;
- Просмотреть данные можно в облаке Saures;
- Габариты, мм 55 x 155 x 170;
- Вес, г. 230;
- Наличие батареи: AA - 1.5В;

– Крепится на стену.

Стоимость датчика – 5500 руб.

Удобство заключается в компактности и удобстве эксплуатации, все данные можно просмотреть в облаке компании. Из недостатков – привязка к облаку производителя; при отсутствии батареек нет возможности зарядить как аккумулятор.

### 3 АРХИТЕКТУРА ПРОЕКТА

Проект состоит из следующих элементов:

1. Устройство по сбору микроклиматических параметров – конечное устройство, отправляющее сообщения по беспроводной сети на шлюз и/или получающее сообщения от него. Он состоит из следующих элементов:

- Датчик – измерительное устройство.
- Модуль LoRa – приемопередатчик с технологией LoRa, предназначенный для беспроводной передачи данных.
- Микроконтроллер – главный элемент конечного устройства, управляющий датчиком и модулем LoRa.

2. Базовая станция – центральный элемент построения сети на основе технологии LoRaWAN. Работает по принципу прозрачного шлюза между оконечными устройствами и сервером .

3. Сервер – элемент сети, отвечающий за управление и обслуживание всей сетью LoRa.

4. Клиент – клиентское приложение, с помощью которого пользователь имеет возможность взаимодействовать с сетью LoRa.

Схема архитектуры представлено на рисунок 7.

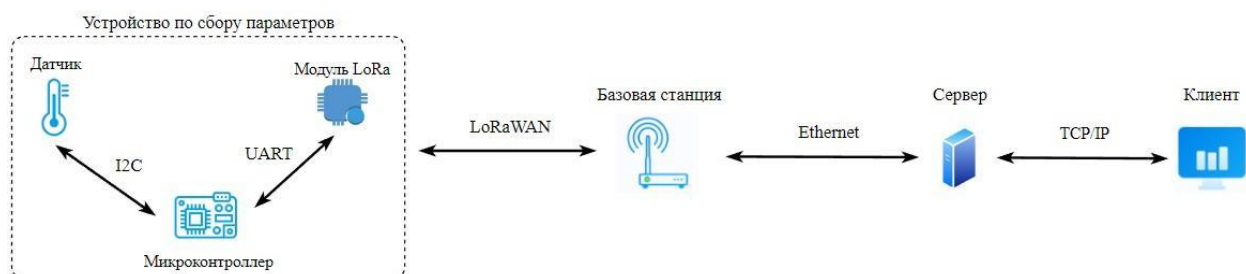


Рисунок 7 – Архитектура проекта

Для написания программы взаимодействующей с сервером Вега использовался язык программирования Python с подключенной библиотекой “websocket”. Программный код был написан с помощью IDE PyCharm.

Для создания прошивки микроконтроллера Nucleo F103RB использовался язык программирования C++ с подключенными библиотеками “mbed” и “BME280”. Программный код был написан в программной платформе для разработки IoT-устройств Mbed Studio.

В устройстве по сбору параметров используются следующие технологии: шина I2C для соединения микроконтроллера и сенсора и протокол UART для связи микроконтроллера с модулем LoRa. Само устройство для связи с базовой станцией использует протокол LoRaWAN. Базовая станция в свою очередь подключается к серверу по Ethernet. А клиент взаимодействует с сервером по протоколу TCP/IP.

## **4 ПРАКТИЧЕСКАЯ ЧАСТЬ**

### **4.1 Обзор базовой станции**

Базовая станция Вега БС-2.2 от компании «ВЕГА АБСОЛЮТ» (рисунок 8) предназначена для развёртывания сети LoRaWAN на частотах диапазона 863-870 МГц. Базовая станция – это центральный элемент построения сети на основе технологии LoRaWAN.

Питание данной базовой станции и сообщение с сервером осуществляется через канал Ethernet.

Одной из особенностей данной станции является наличие GPS/ГЛОНАСС модуля, остальные характеристики базовой станции Вега БС-2.2 представлены на рисунке 8.





#### Характеристики

GPS приемник	да, со встроенной антенной
3G модем	да
Операционная система	Linux
Канал связи с сервером	Ethernet, GSM 3G
USB-порт	да
Диапазон рабочих температур, °C	-40...+70
Количество каналов LoRaWAN®	8
Частотный диапазон	863-870 МГц
Мощность передатчика	до 500 мВт (27 dBm)
Антенный разъём	N-Type female
Дальность радиосвязи в сельской местности	до 15 км
Дальность радиосвязи в плотной городской застройке	до 5 км
Потребляемая мощность	до 10 Вт
Тип питания	Passive POE 4,5(+) 7,8(-) 15Вт
Напряжение питания	12...48 В
Размеры корпуса, не более, мм	192 x 183 x 75
Степень защиты корпуса	IP67
Крепление	на балки/мачты
Габариты упаковки, мм	250 x 220 x 85
Вес комплекта в упаковке, кг	1,250

Рисунок 8 – Вид и характеристики базовой станции Вега БС-2.2

## 4.2 API-документация Vega server

У Vega server имеются различные API-функции, необходимые для взаимодействия с клиентским приложением.

При изучении API-документации сервера были выделены те функции, которые необходимо реализовать для дальнейшего написания клиентского приложения. К ним относятся: авторизация пользователя (рисунок 9), получение информации о сервере (рисунок 10), получение списка устройств (рисунок 11), получение списка зарегистрированных пользователей (рисунок 12), возврат сохраненных данных с устройств (рисунок 13). [8]

### User authorization

---

#### Request message:

```
{
  "cmd": "auth_req",
  "login": string,           // case insensitive string
  "password": string        // original password string without any encoding
}
```

#### Response message:

```
{
  "cmd": "auth_resp",
  "status": boolean,
  "err_string"? : string    //[[optional exist if "status" is false] – string code of error
  "token"? : string,        //[[optional exist if "status" is true] – session string token (32 HEX symbols)
  "device_access"? : string, //[[optional exist if "status" is true] – access level to devices (see below possible values)
  "consoleEnable": bool,    //[[optional exist if "status" is true] – enable to connect to console subchart with debug information
  "command_list"? :         //[[optional exist if "status" is true] – accessible commands1 (see below possible values)
  [
    "command_1",
    ...,
    "command_n"
  ],
  "rx_settings"? :          //[[ optional exist if "status" is true] – setting of online receiving messages
  {
    "unsolicited": boolean, //online message is sending [true] or not sending [false - default]
    "direction": string,    //[[optional absent if "unsolicited" is false] – direction of possible online messages (see below)
    "withMacCommands": boolean //[[optional] – online message contains MAC commands
  }
}
```

Рисунок 9 – Функция для авторизации пользователя

## Server information (server\_info\_req, server\_info\_resp)

---

### Request message:

```
{
  "cmd": "server_info_req"
}
```

### Response message:

```
{
  "cmd": "server_info_resp",
  "status": boolean,           // status of command execution
  "err_string?": string,       // [optional – exist if "status" is false] error string code (see below
                                // description)
  "version?": string,          // [optional – exist if "status" is true] Version of server (e.g. "1.1.4")
  "time?":                     // [optional – exist if "status" is true] Time info
  {
    "utc": number,             // Server UTC timestamp (milliseconds from Linux epoch)
    "time_zone": string,       // Time zone (e.g. "UTC+01:00")
    "time_zone_utc_offset": number // Time offset from UTC in seconds (e.g. 3600)
  },
  "device_count"?:8           // [optional – exist if "status" is true] Device count info
  {
    "device_count_vega": number, // Count of registered devices of "Vega Absolute" company
    "device_count_other": number, // Count of registered devices of others manufacturers
    "available_device_count": number // Count of free remain devices
  }
}
```

### Example response message:

```
{
  "cmd": "server_info_resp",
  "status": true,
  "version": "1.2.0",
  "time":
  {
    "utc": 1513584871084,
    "time_zone": "UTC+07:00",
    "time_zone_utc_offset": 25200
  },
  "device_count":
  {
    "device_count_vega": 24,
    "device_count_other": 75,
    "available_device_count": 925
  }
}
```

Рисунок 10 – Функция для получения информации о сервере

## Get list of devices with attribute set

---

Request message:

```
{
  "cmd": "get_device_appdata_req",
  "keyword"?:           //[optional] See possible values
  [
    string,...
  ]
  "select"?:"           //[optional] Filter object
  {
    "appEui_list"?:      //[optional] List of corresponding AppEUI for request
    [
      "appEui_1",
      ...,
      "appEui_n"
    ]
  }
}
```

Рисунок 11 – Функция получения списка устройств

## Get list of registered users

---

### Request message:

```
{
  "cmd": "get_users_req",
  "keyword"?:           //[optional] See below description
  [
    string, ...
  ]
}
```

### Possible string values of "keyword":

- "no\_command\_and\_devEui" – return list of user without "devEui\_list" and "command\_list"

### Response message:

```
{
  "cmd": "get_users_resp",
  "status": boolean,           // Main status of execution
  "err_string"?: string,       //[optional exist if "status" is false] – string code of error
  "user_list":
  [
    {
      "login": string,
      "device_access": string, //Access level to devices (see below possible values). If "FULL",
                                // "devEui_list" would be ignored
      "consoleEnable": bool,   //Enable to connect to console subchart with debug information
      "devEui_list"?:          //[optional absent if "no_command_and_devEui" is exist] List of
                                // DevEUI that is accessible for user
      [
        "devEui_1",
        ...,
        "devEui_n"
      ],
      "command_list"?:         //[optional absent if "no_command_and_devEui" is exist] List of
                                // commands that is accessible for user
      [
        "command_1",
        ...,
        "command_n"
      ],
      "rx_settings"?:         //[optional absent if "no_command_and_devEui" is exist] – setting of
                                // online receiving messages
      {
        "unsolicited": boolean, //online message is sending [true] or not sending [false - default]
        "direction"?: string,   //[optional absent if "unsolicited" is false] – direction of possible
                                // online messages (see below)
        "withMacCommands"?:boolean // [optional] – online message contains MAC commands
      }
    }, ...
  ]
}
```

Рисунок 12 – Функция получения списка зарегистрированных пользователей



## Return saved data from device

### Request message:

```
{
  "cmd": "get_data_req",
  "devEui": string,
  "select"? : //[[optional] Extra optional for searching
  {
    "date_from"? : integer, //[[optional] server UTC timestamp as number (milliseconds from Linux epoch)
    "date_to"? : integer, //[[optional] server UTC timestamp as number (milliseconds from Linux epoch)
    "begin_index"? : integer, //[[optional] begin index of data list [default = 0]
    "limit"? : integer //[[optional] limit of response data list [default =1000]
    "direction"? : string, //[[optional] direction of message transition (see below description)
    "withMacCommands"? : boolean //[[optional] add MAC commands to response
  }
}
```

### Response message:

```
{
  "cmd": "get_data_resp",
  "status": boolean, // Status of execution of command (global status)
  "err_string"? : string, //[[optional] If "status" = false, contains error description (see below description)

  "devEui": string,
  "direction"? : string, //[[optional – exist if "status" = true]
  "totalNum"? : integer, //[[optional – exist if "status" = true] Total existing number of data corresponding type
  "data_list"? : //[[optional – exist if "status" = true] Data, transmitted by device
  [
    {
      "ts " : integer, // Server UTC receiving timestamp (milliseconds from Linux epoch)
      "gatewayId": string, // Gateway IDs that receive data from device
      "ack": boolean, // Acknowledgement flag as set by device
      "fcnt": integer, // Frame counter, a 32-bit number (uplink or downlink based on "direction" value)
      "port": integer, // Port (if = 0, use JOIN operations or MAC-commands only)
      "data" : string, // Decrypted data payload
      "macData"? : string, //[[optional– exist if "withMacCommands" true and MAC command is present] MAC command data from device
      "freq": integer, // Radio frequency at which the frame was received/transmitted, in Hz
      "dr": string, // Spreading factor, bandwidth and coding rate "SF12 BW125 4/5"
      "rssi": integer, //[[optional – exist if packet direction "UPLOAD"] Frame rssi, in dBm, as integer number
      "snr": float, //[[optional – exist if packet direction "UPLOAD"] Frame snr, in dB
      "type": string, // Type of packet (see below description). May contains several types joined via "+"
      "packetStatus"? : string //[[optional – exist if packet direction "UPLOAD"] Status of downlink message only (see below description)
    }, ...
  ]
}
```

Рисунок 13 – Функция возврата сохраненных данных с устройства

Все представленные выше функции необходимы для реализации web-приложения.

### 4.3 Реализация web-приложения

Web-приложение было написано на языке программирования Python с использованием фреймворка Flask. В этом приложении реализованы такие функции как: авторизация пользователей, добавление и удаление пользователей, подключение к серверу Vega, получение данных с устройства, добавление и удаление новых устройств и получение списка подключенных устройств и пользователей.

В процессе написания web-приложения было необходимо взаимодействовать с сервером Vega. Сетевой сервер IOT Vega Server – это инструмент предназначенный для управления сетью базовых станций, приема данных с конечных устройств и передачи их внешним приложениям, а также передачи данных от внешних приложений на LoRaWAN устройства. Открытый API, основанный на технологии Web Socket позволяет подключать к IOT Vega Server внешние приложения и использовать возможности LoRaWAN сетей в проекте.

Web Socket – протокол связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени.

На рисунке 14 представлен код самой функции подключения к серверу Вега и отправки методов на него.

```

def send_req(req: dict) -> dict:
    ws = create_connection(url=URL_WS)
    if req.get("cmd") != "auth_req":
        recovery_session = {"cmd": "token_auth_req", 'token': session.get('token')}
        ws.send(json.dumps(recovery_session))
        resp_json = json.loads(ws.recv())
        if resp_json.get('cmd') == "console":
            ws.recv()
        if resp_json.get("status"):
            session['token'] = resp_json.get('token')
        print("=====")
        print(f"|command - {req.get('cmd')}|")
        print(resp_json)
        print("=====")
    print("sended")
    ws.send(json.dumps(req)) # Get command
    print("recieved")
    resp_json = json.loads(ws.recv())
    if resp_json.get('cmd') == "console":
        ws.recv()
        return send_req(req)
    print("=====")
    print(f"|command - {req.get('cmd')}|")
    print(resp_json)
    print("=====")
    ws.close()
    return resp_json

```

Рисунок 14 – Подключение к серверу

На рисунке 15 представлен фрагмент кода, с помощью которого выполняется авторизация. Если логин и пароль вводятся корректно, то авторизация проходит успешно, иначе происходит возврат на форму авторизации. О том, что авторизация выполнена успешно можно судить по консоли IoT Vega Server, в котором появляется событие об успешном подключении (рисунок 16).




```

@app.route('/login/', methods=['post', 'get'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        login = form.login.data
        password = form.password.data

        # Autorization on VEGA server
        autreq = {"cmd": "auth_req", # Don't change!
                  "login": str(login), # Login name
                  "password": str(password) # password
                 }
        autresp = send_req(autreq)
        if autresp.get("err_string") is None:
            session["command_list"] = autresp.get("command_list")
            session['token'] = autresp.get("token")
            return redirect(url_for('index'))
        else:
            flash("Неверный логин/пароль", 'error')
    return render_template('login.html', form=form)

```

Рисунок 15 – Авторизация на сервере Vega



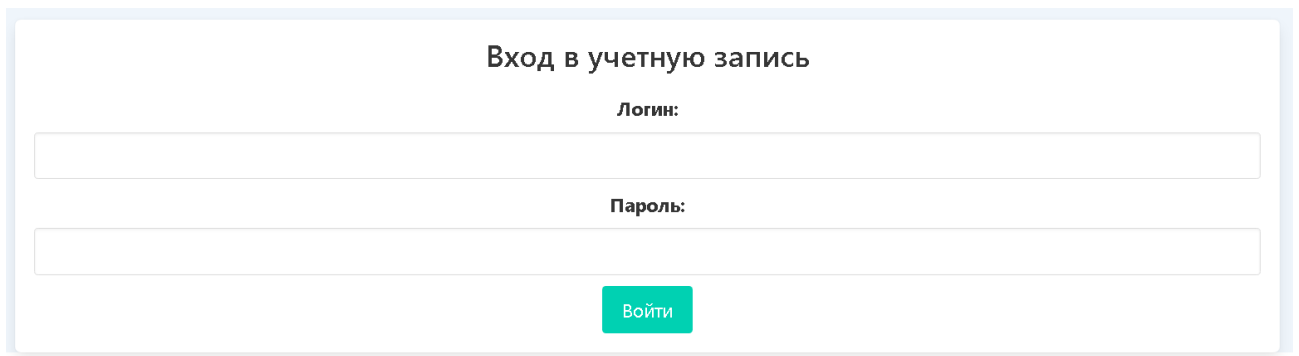
```

[CWebsocketServer] New connection
[CWebSocket::closeConnectionSlot]

```

Рисунок 16 – Событие о новом подключении

Страница авторизации представлена на рисунке 17.



Вход в учетную запись

Логин:

Пароль:

Войти

Рисунок 17 – Страница авторизации

На рисунке 18 представлена функция, которая отвечает за выход из приложения. Отправляется запрос на выход из приложения, в результате которого происходит перенаправление на окно авторизации.

```
@app.route('/logout/')
def logout():
    if 'token' in session:
        log_out = {"cmd": "close_auth_req", # Don't change!
                    "token": session.get("token")}
        out = send_req(log_out)
        if out.get("err_string") is None:
            flash("You have been logged out.")
            session.pop('token', None)
            return redirect(url_for('login'))
    return redirect(url_for('login'))
```

Рисунок 18 – Выход из приложения

На рисунках 19-20 представлена непосредственно сама функция, которая отвечает за отображение главной страницы и её UML-диаграмма. Здесь на сервер Веги отправляются соответствующие команды для получения списков подключенных устройств и зарегистрированных пользователей, а также производится проверка команд, доступных пользователю, для того, чтобы понять, какие кнопки для него будут активными.

---

```

@app.route('/')
def index():
    context = dict()
    if 'token' not in session:
        return redirect('login')
    # Get information from connected server
    srvinfo = {"cmd": "server_info_req"} # Don't change!

    # Get device list w/attributes
    devalist = {"cmd": "get_device_appdata_req"} # Don't change!

    # Get reg users
    reguser = {"cmd": "get_users_req"} # Don't change!
    infresp_dict = send_req(srvinfo)
    if infresp_dict.get("err_string") == "unknown_auth":
        return redirect(url_for('login'))
    if "manage_users" in session.get("command_list"):
        context['is_can_create_user'] = True
    if "manage_devices" in session.get("command_list"):
        context['is_can_add_device'] = True
    if "delete_users" in session.get("command_list"):
        context['is_can_delete_user'] = True
    if "delete_devices" in session.get("command_list"):
        context['is_can_delete_device'] = True
    time_serv_now = infresp_dict.get("time").get("utc") / 1000
    local_timezone = tzlocal.get_localzone()
    serv_time = datetime.fromtimestamp(time_serv_now, local_timezone)
    context['time'] = serv_time.strftime("%Y-%m-%d %H:%M:%S")
    context['city'] = infresp_dict.get("time").get("time_zone", 'None')
    # Get dev list w/attributes
    devalistresp = send_req(devalist)
    context["devices_list"] = devalistresp.get("devices_list")

    # Get registered users
    reguserresponse = send_req(reguser)
    context["user_list"] = reguserresponse.get("user_list")
    return render_template('index.html', context=context)

```

Рисунок 19 – Отображение подключенных устройств и пользователей

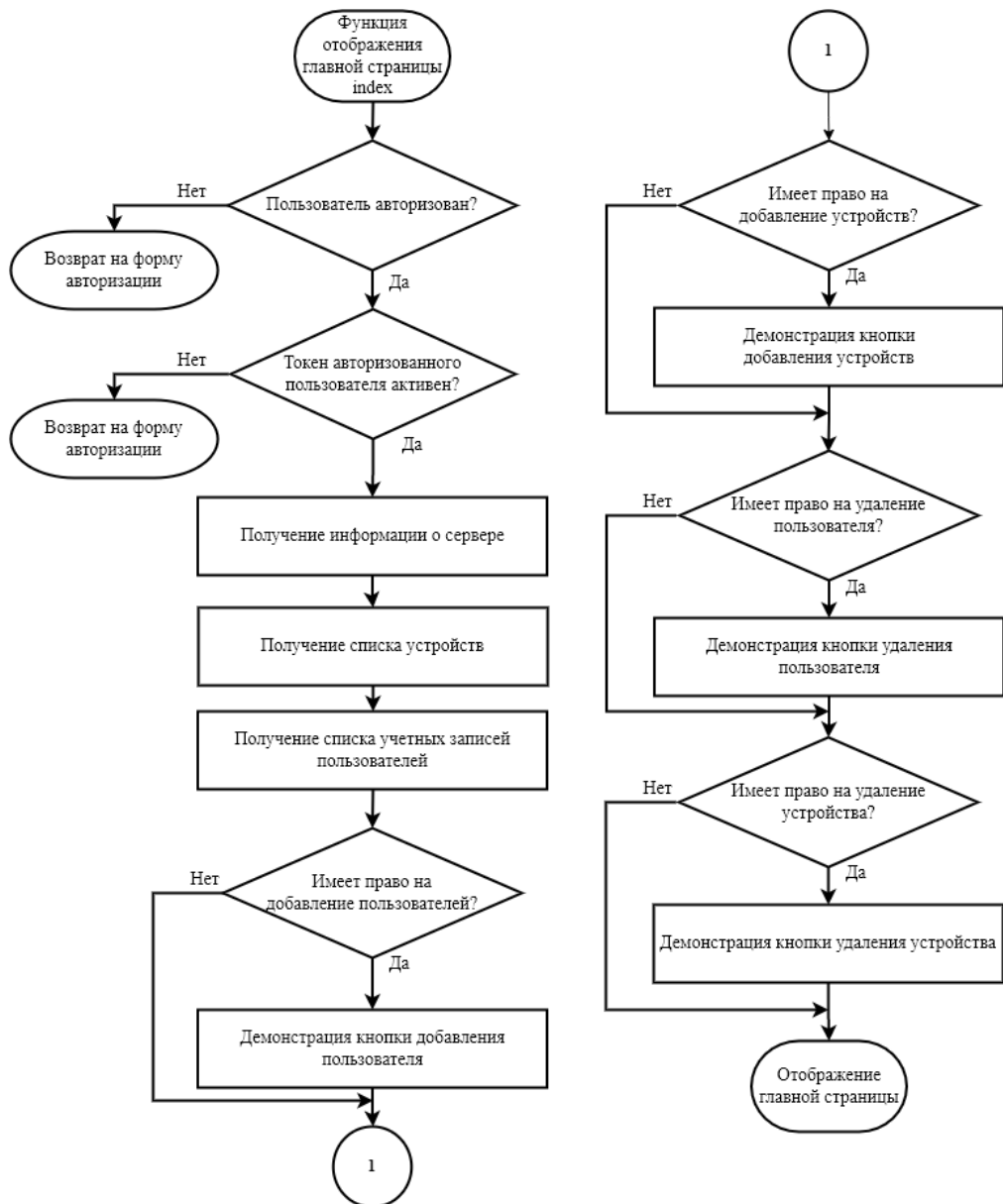


Рисунок 20 – UML-диаграмма главной страницы

Вид данной страницы представлен на рисунке 21.

Подключенные устройства
Список пользователей

Создать учетную запись
Добавить устройство
Выход

devName	AppEui	devEui	
dev2_2909	AC1F09FFF8680811	AC1F09FFFE015302	Удалить
device_number_one	AC1F09FFF8680811	AC1F09FFFE015304	Удалить

29.06.2023 17:02:20  
Город: Томск

Рисунок 21 – Подключенные устройства

Функция, которая отвечает за добавление устройства, представлена на рисунках 22–23. В начале производится проверка какой метод пришел на web-приложение. После чего происходит запрос к данным формы, а затем проверка того, что данные заполнены верно, если все данные записаны верно, то все эти переменные записываются в словарь для отправки запроса на сервер. Полученный ответ от сервера впоследствии проходит проверку на то, добавлено устройство или нет.

```

@app.route('/add_device/', methods=['post', 'get'])
def add_device():
    context = dict()
    form = AddDeviceForm()
    if request.method == "POST":
        if form.is_submitted():
            dev_eui = form.dev_eui.data # запрос к данным формы
            dev_name = form.dev_name.data
            dev_address = form.dev_address.data
            apps_key = form.apps_key.data
            nwks_key = form.nwks_key.data
            app_eui = form.app_eui.data
            app_key = form.app_key.data
            class_user = form.class_user.data
            set_data = {
                "devEui": dev_eui,
            }
            if dev_address is not None and apps_key != "" and nwks_key is not None:
                abp = {
                    "devAddress": dev_address,
                    "appsKey": apps_key,
                    "nwksKey": nwks_key
                }
                set_data["ABP"] = abp
            if app_key != "":
                otaa = {
                    "appKey": app_key,
                }
                if app_eui != "":
                    otaa["appEui"] = app_eui
                set_data["OTAA"] = otaa

            if dev_name is not None:
                set_data["devName"] = dev_name
            if class_user is not None:
                set_data["class"] = class_user
            set_data["frequencyPlan"] = {
                "freq4": 867100000,
                "freq5": 867300000,
            }

```

Рисунок 22 – Добавление устройства. Часть 1

```

set_data["frequencyPlan"] = {
    "freq4": 867100000,
    "freq5": 867300000,
    "freq6": 867500000,
    "freq7": 867700000,
    "freq8": 867900000
}
device_list = list()
device_list.append(set_data)
query = {
    "cmd": "manage_devices_req",
    "devices_list": device_list
}
print(f'query add dev - {query}')
resp = send_req(query)
if resp.get("err_string") is None and resp.get('device_add_status')[0].get("status") in success_result_add_device_list:
    return redirect(url_for('index'))
else:
    flash(resp.get('device_add_status')[0].get("status"), 'error')
    return render_template('add_device.html', form=form, context=context)
return render_template('add_device.html', form=form, context=context)

```

Рисунок 23 – Добавление устройства. Часть 2

На рисунке 24 представлена страница добавления устройства.

**EUI устройства\*:**  
EUI устройства, должно быть 16 символов в HEX  
AC1F09FFFE015302

**Имя устройства:**  
Название устройства  
test\_web

**Класс пользователя**  
Класс для данного устройства  
CLASS\_C

ABP (or OTAA)  
Активация по персональным параметрам

**Адрес устройства:**  
32 битный адрес устройства, должен быть в диапазоне от 0x00000001...0xFFFFFFFF

**Application session key:**  
Ключ приложения сессии состоящая из 32 символов HEX

**Network session key:**  
Ключ сети сессии состоящая из 32 символов HEX

OTAA (or ABP)  
Активация параметров по воздуху

**Application EUI:**  
EUI приложения состоящий из 16 символов HEX  
AC1F09FFF8680811

**Application key:**  
Ключ приложения состоящий из 32 символов HEX  
AC1F09FFFE015302AC1F09FFF8680811

Добавить

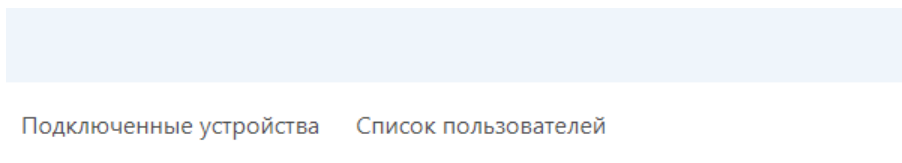
Рисунок 24 – Добавление устройства

Функция, которая отвечает за удаление устройства представлена на рисунке 25. Здесь в начале производится проверка на то, что пользователь авторизован, после чего создается список, в который заносится номер устройства, и отправляется команда для удаления, затем выполняется проверка на то, что устройство удалено.

```
@app.route("/delete_device/<string:dev_eui>", methods=["GET"])
def delete_device(dev_eui):
    if session.get('token') is None:
        redirect(url_for('login'))
    device_list = list()
    device_list.append(dev_eui)
    query = {
        "cmd": "delete_devices_req",
        "devices_list": device_list
    }
    print(f"query - {query}")
    resp = send_req(query)
    if resp.get("status") and resp.get("device_delete_status")[0].get('status') == 'deleted':
        flash("Устройство было успешно удалено", "info")
    else:
        flash(f"Устройство не было удалено. потому что '{resp.get('err_string')}'", "error")
    return redirect(url_for('index'))
```

Рисунок 25 – Удаление устройства

Для того, чтобы удалить устройство в списке подключенных устройств нужно нажать на кнопку “Удалить”, расположенную рядом с идентификатором устройства (рисунок 26).



Подключенные устройства    Список пользователей

devName	AppEui	devEui	
dev2_2909	AC1F09FFF8680811	AC1F09FFFE015302	Удалить
device_number_one	AC1F09FFF8680811	AC1F09FFFE015304	Удалить

Рисунок 26 – Кнопка “Удалить”



После чего будет выведено сообщение об успешном удалении (рисунок 27).

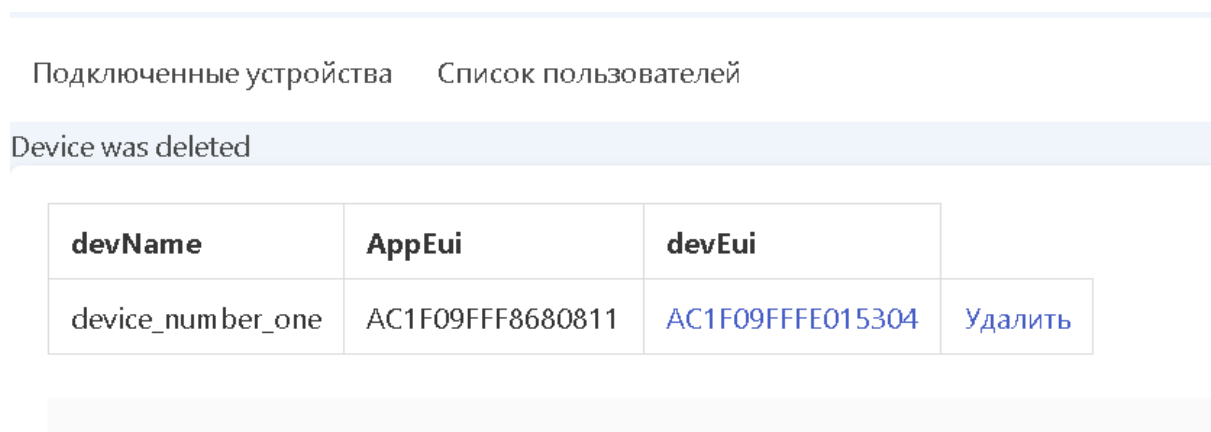


Рисунок 27 – Сообщение об успешном удалении

Далее в соответствии с UML-диаграммой в IoT Vega Admin Tool были созданы следующие учетные записи пользователей:

- Обычный пользователь;
- Администратор;
- Суперадмин.

Данные записи отображаются в списке пользователей на главной странице (рисунок 28).

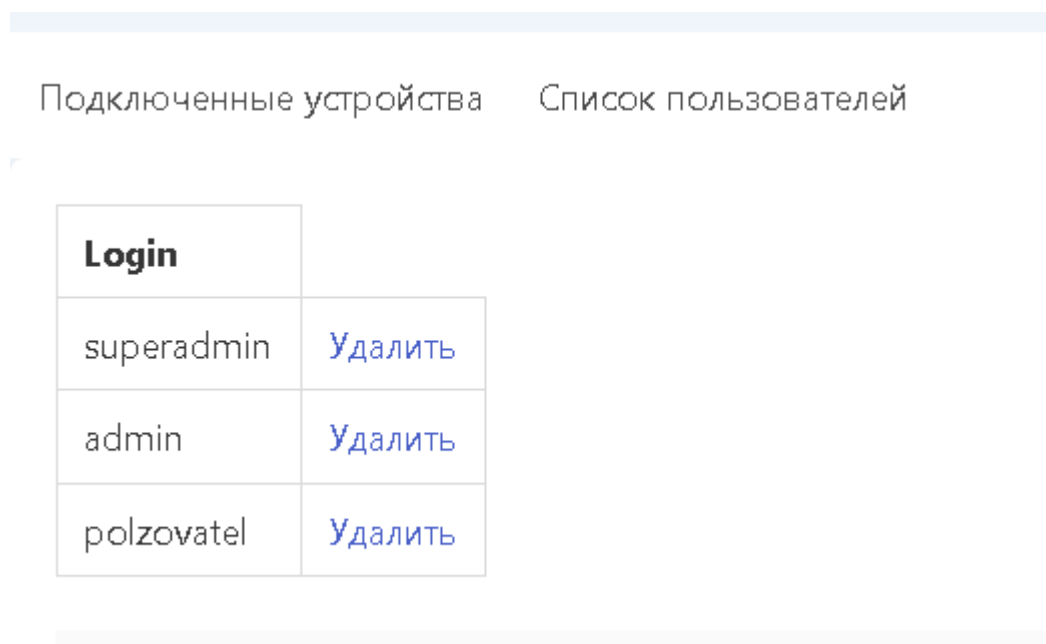


Рисунок 28 – Список пользователей

Для проверки на соответствие выданных прав созданным пользователям был выполнен вход в каждую из учетных записей.

На рисунке 29 изображен интерфейс учетной записи обычного пользователя.

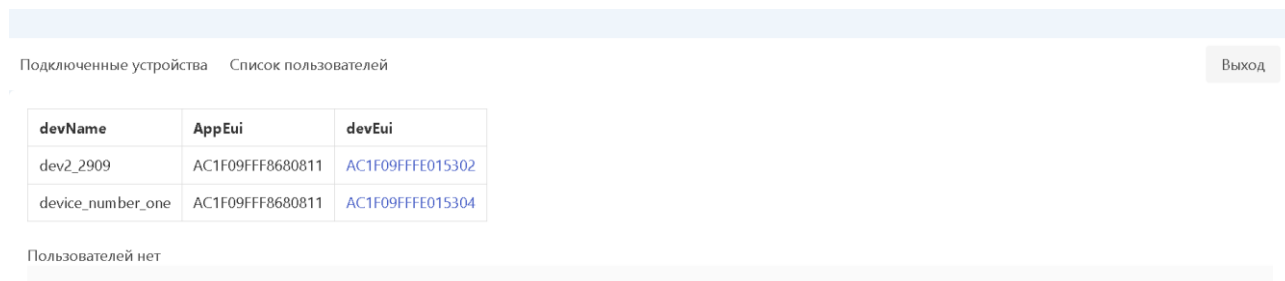


Рисунок 29 – Интерфейс обычного пользователя

Скриншот интерфейса пользователя с ролью администратора представлен на рисунке 30.

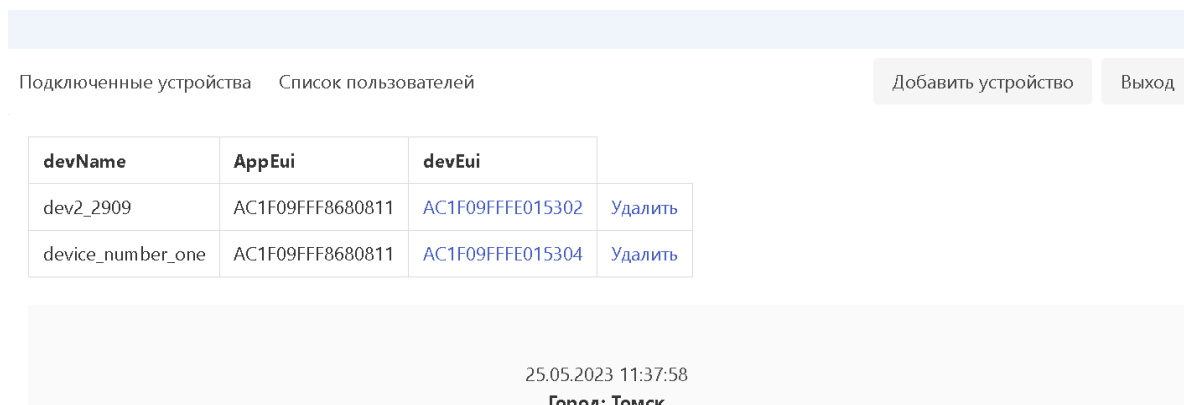


Рисунок 30 - Интерфейс администратора

Интерфейс пользователя учетной записи с правами superadmin представлен на рисунке 31.

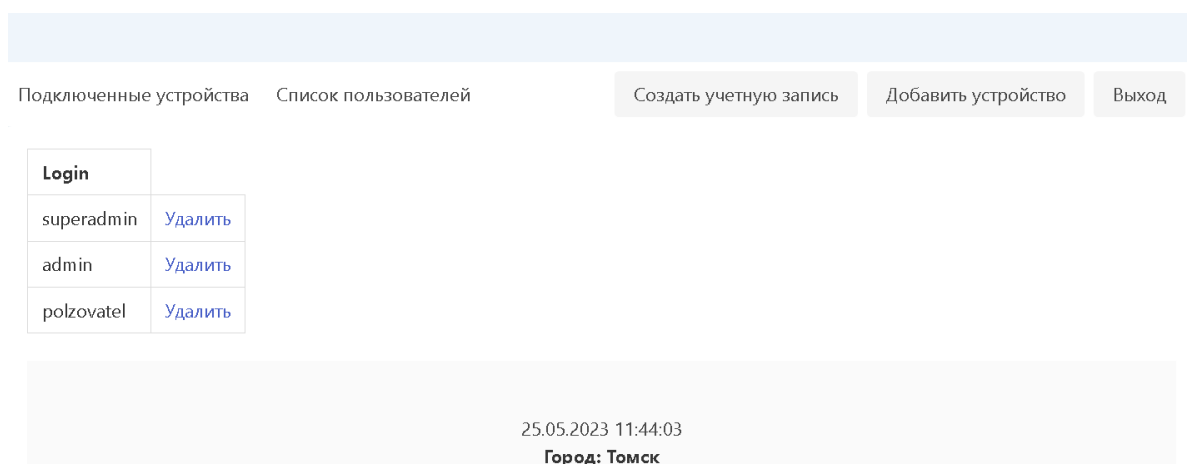


Рисунок 31 - Интерфейс пользователя с ролью superadmin

На рисунках 32–33 представлена функция, которая отвечает за добавление учетной записи пользователя через web-приложение. Здесь принцип работы такой же как и в добавлении устройства.

```

@app.route('/create_user/', methods=['post', 'get'])
def create_user():
    context = dict()
    form = CreateUserForm()
    if request.method == "POST":
        form.devEui_list.choices = form.devEui_list.data
    if form.validate_on_submit():
        login = form.login.data # запрос к данным формы
        password = form.password.data
        device_access = form.device_access.data
        console_enable = form.console_enable.data
        devEui_list = form.devEui_list.data
        command_list = form.command_list.data
        unsolicited = form.unsolicited.data
        direction = form.direction.data
        with_MAC_Commands = form.with_MAC_Commands.data
        # установка значений для запроса
        set_data = {
            "login": login,
            "password": password,
            "device_access": device_access,
            "consoleEnable": console_enable,
            "devEui_list": devEui_list,
            "command_list": command_list,
            "rx_settings": {
                "unsolicited": unsolicited,
                "direction": direction,
                "withMacCommands": with_MAC_Commands
            }
        }
        user_list = list()
        user_list.append(set_data)
        query = {
            "cmd": "manage_users_req",
            "user_list": user_list
        }
        resp = send_req(query)
        if resp.get("err_string") is None:
            return redirect(url_for('index'))

```

Рисунок 32 – Добавление учетной записи пользователя. Часть 1

```

        if resp.get("err_string") is None:
            return redirect(url_for('index'))
        else:
            flash(resp.get("err_string"), 'error')
    }
    return render_template('create_user.html', form=form, context=context)
}

query = {
    "cmd": "get_devices_req"
}

resp = send_req(query)
devices_list = resp.get("devices_list")
form.devEui_list.data = [dev.get("devName") for dev in devices_list]
dev_Euis = [dev.get("devEui") for dev in devices_list]
dev_names = [dev.get("devName") for dev in devices_list]
form.command_list.choices = session.get('command_list')
form.devEui_list.choices = dev_Euis
}
return render_template('create_user.html', form=form, context=context)

```

Рисунок 33 – Добавление учетной записи пользователя. Часть 2

Страница добавления новой учетной записи пользователя представлена на рисунке 34.

**Логин:**

**Пароль:**

**Доступ к устройству:**

SELECTED ▼

**Работа с консолью**

☐

**DevEui список**

AC1F09FFFE015304

**Список команд**

get\_users

manage\_users

delete\_users

get\_device\_appdata

**Unsolicited**

☐

**Направление**

Рисунок 34 – Добавление учетной записи пользователя

Здесь необходимо выбрать логин и пароль для учетной записи пользователя, после чего выбрать к каким устройствам имеет доступ пользователь, ко всем или только к определенным, если же к определенным, то необходимо выбрать из списка. Также необходимо выбрать список команд, которые будут доступны пользователю и другие настройки, такие как

возможность работы с консолью, возможности отправки сообщений, отправка сообщений с MAC-командами и направление сообщений (с сервера или на сервер).

Данный тестовый пользователь был добавлен (рисунок 35). Для проверки корректности работы нужно обратиться к IoT Vega Admin Tool (рисунок 36).

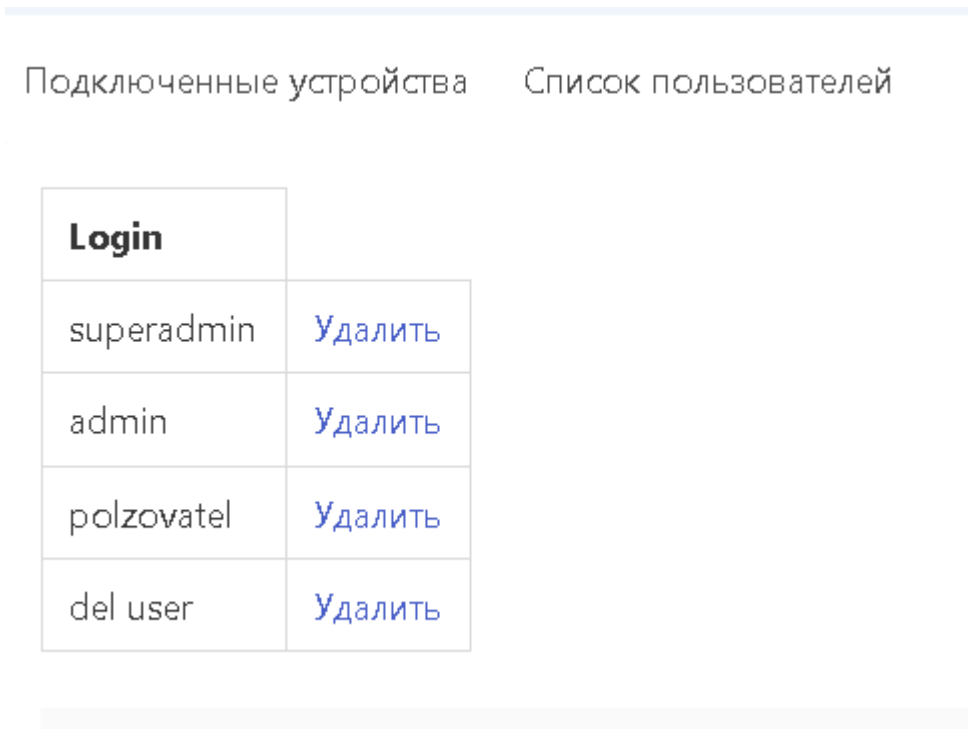


Рисунок 35 – Добавление учетной записи пользователя

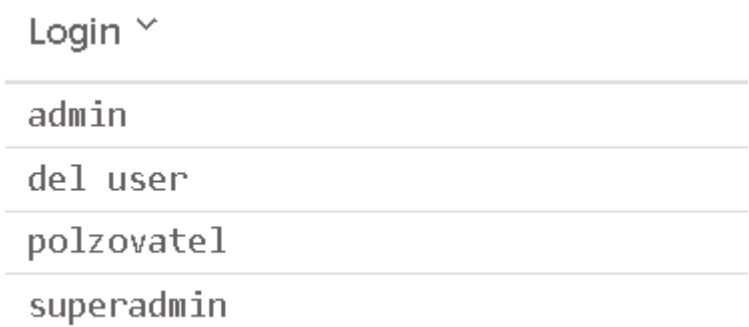


Рисунок 36 – Список учетных записей в IoT Vega Admin Tool

На рисунке 37 представлена функция удаления учетной записи пользователя, который работает по тому же принципу, что и удаление устройства.

```
@app.route("/delete_user/<string:login>", methods=["GET"])
def delete_user(login):
    if session.get('token') is None:
        redirect(url_for('login'))
    user_list = list()
    user_list.append(login)
    query = {
        "cmd": "delete_users_req",
        "user_list": user_list
    }
    print(f"query - {query}")
    resp = send_req(query)
    if resp.get("status") and resp.get("delete_user_list")[0].get('status') and resp.get("err_string") is None:
        flash("Учетная запись была удалена", "info")
    else:
        flash(f"Учетная запись не была удалена, потому что '{resp.get('err_string')}'", "error")
    return redirect(url_for('index'))
```

Рисунок 37 – Удаление учетной записи пользователя

Для того, чтобы удалить учетную запись пользователя нужно также нажать на кнопку “Удалить” в списке пользователей (рисунок 38).

Подключенные устройства      Список пользователей

Список пользователей	
Login	
superadmin	Удалить
admin	Удалить
polzovatel	Удалить
del user	Удалить

Рисунок 38 – Кнопка “Удалить”



После удаления пользователя появляется уведомление об удалении пользователя (Рисунок 39).

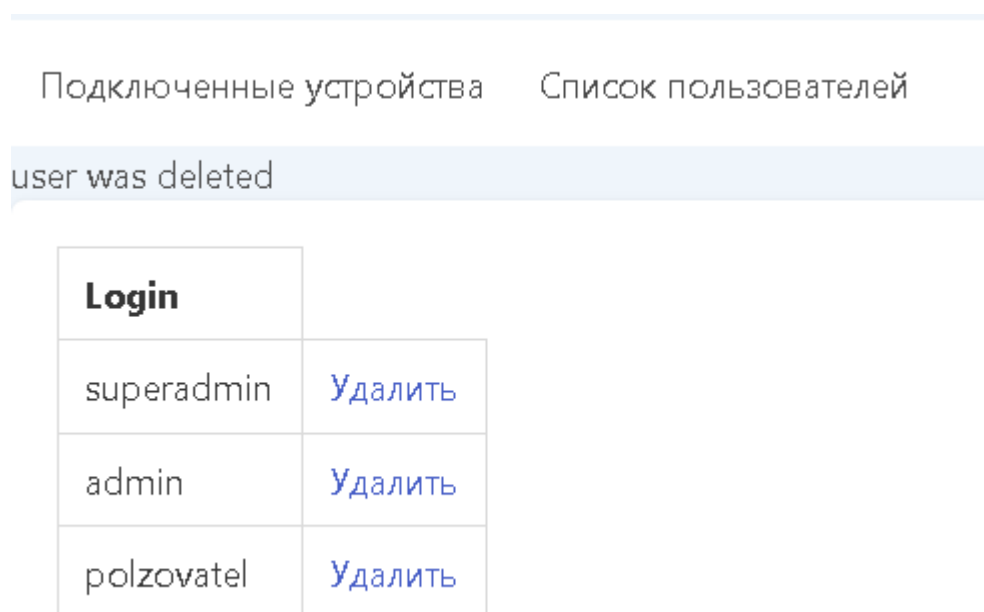


Рисунок 39 – Уведомление об удалении

Для того, чтобы проверить, что удаление сработало корректно нужно обратиться к IoT Vega Admin Tool (рисунок 40).

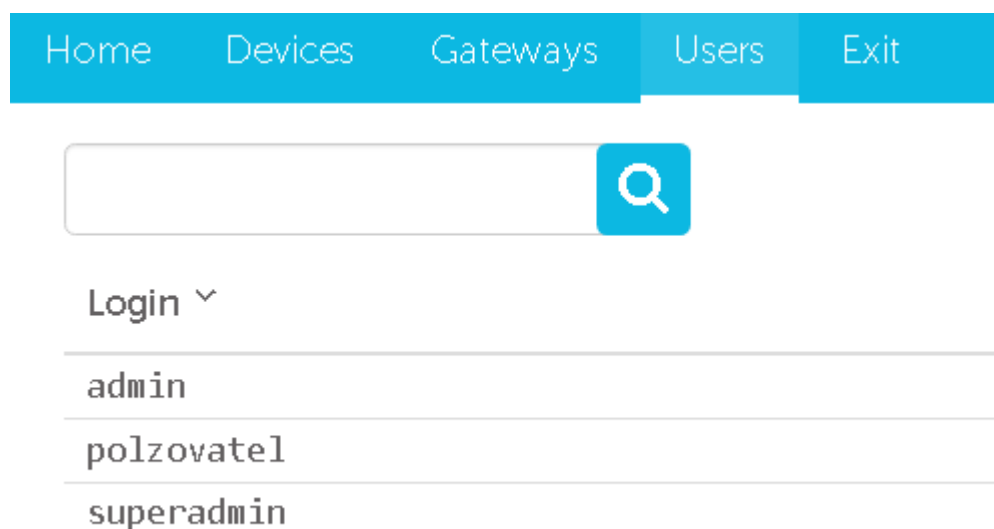


Рисунок 40 – Список пользователей после удаления в Iot Vega Admin Tool

Функция, которая отвечает за построение графика по полученным значениям, представлена на рисунке 41. Здесь отправляется команда для

получения последних данных с устройства по указанному пользователем лимитом. После чего выполняется проверка полученных данных и если получены правильные данные, то пакет обрабатывается и берутся только нужные для построения графика значения, затем обрабатывается словарь и берутся ключи из него для построения таблицы.

```
@app.route('/dev_graph/<string:dev_eui>/<string:devName>/<int:limit>', methods=["GET"])
def dev_chart(dev_eui, devName, limit):
    if session.get('token') is None:
        redirect(url_for('login'))
    context = dict()
    title = f"Chart of {devName}"
    context["legend"] = devName
    query_req = {
        "cmd": "get_data_req",
        "devEui": dev_eui,
        "select":
        {
            "limit": limit
        }
    }
    resp = send_req(query_req)
    if not resp.get("status"):
        flash(resp.get("err_string"), 'error')
        return render_template("index.html")
    if resp.get("cmd") == "get_data_resp":
        data_list = resp.get("data_list")
        labels = list()
        data = list()
        for every_data in data_list:
            every_data['ts'] = str(datetime.fromtimestamp(every_data.get('ts')/1000, timezone.utc).strftime("%d/%m/%Y, %H:%M:%S"))
            labels.append(every_data.get('ts'))
            data.append(every_data.get('data'))
        context["data"] = data
        context["labels"] = labels
        context["raw_data_list"] = data_list
        try:
            context["raw_data_list_keys"] = data_list[0].keys()
        except IndexError:
            flash("Ошибка при построении графика", 'error')
            return redirect(url_for('index'))
    return render_template("chart.html", context=context, title=title)
```

Рисунок 41 – Построение графика

Для того, чтобы перейти на страницу с графиком необходимо нажать на идентификатор устройства в списке устройств, график представлен на рисунке 42. Для отрисовки графиков была использована JavaScript библиотека визуализации данных ChartJS. [7]



Рисунок 42 – График полученных данных

Стандартно график строится по последним 48 значениям поскольку с оконечных устройств данные будут отправляться на сервер через каждые полчаса, что равняется 48 пакетам в сутки. Было реализовано динамическое обновление графика раз в 30 минут. На рисунке 43 представлен фрагмент кода отвечающий за обновление графика.

```

43 $(document).ready(function(){
44   var ctx = document.getElementById('myChart').getContext('2d');
45   var chart = new Chart(ctx, {
46     type: 'line',
47     data: {
48       labels: {{ context.labels|tojson }},
49       datasets: [{
50         label: 'Температура, *C',
51         backgroundColor: 'rgb(255, 99, 132)',
52         borderColor: 'rgb(255, 99, 132)',
53         data: {{ context.data|tojson }},
54         tension: 0.1,
55       }]
56     });
57   setInterval(function() {
58     var ctx = document.getElementById('myChart').getContext('2d');
59     var chart = new Chart(ctx, {
60       type: 'line',
61       data: {
62         labels: {{ context.labels|tojson }},
63         datasets: [{
64           label: 'Температура, *C',
65           backgroundColor: 'rgb(255, 99, 132)',
66           borderColor: 'rgb(255, 99, 132)',
67           data: {{ context.data|tojson }},
68           tension: 0.1,
69         }]
70       });
71     },60000);

```

Рисунок 43 - Обновление графика

## 5. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ ПО ИСПОЛЬЗОВАНИЮ СИСТЕМЫ

Для того, чтобы запустить web-приложение с полным функционалом, необходимо выполнить следующие пункты:

- Подключить устройство по сбору параметров;
- Настроить базовую станцию;
- Настроить сервер;
- Настроить приложение для администрирования;
- Настроить web-приложение.

### 5.1. Работа с устройством по сбору параметров

Первым делом необходимо собрать в одну цепь все модули устройства. Сделать это необходимо согласно схеме, показанной на рисунке 44.

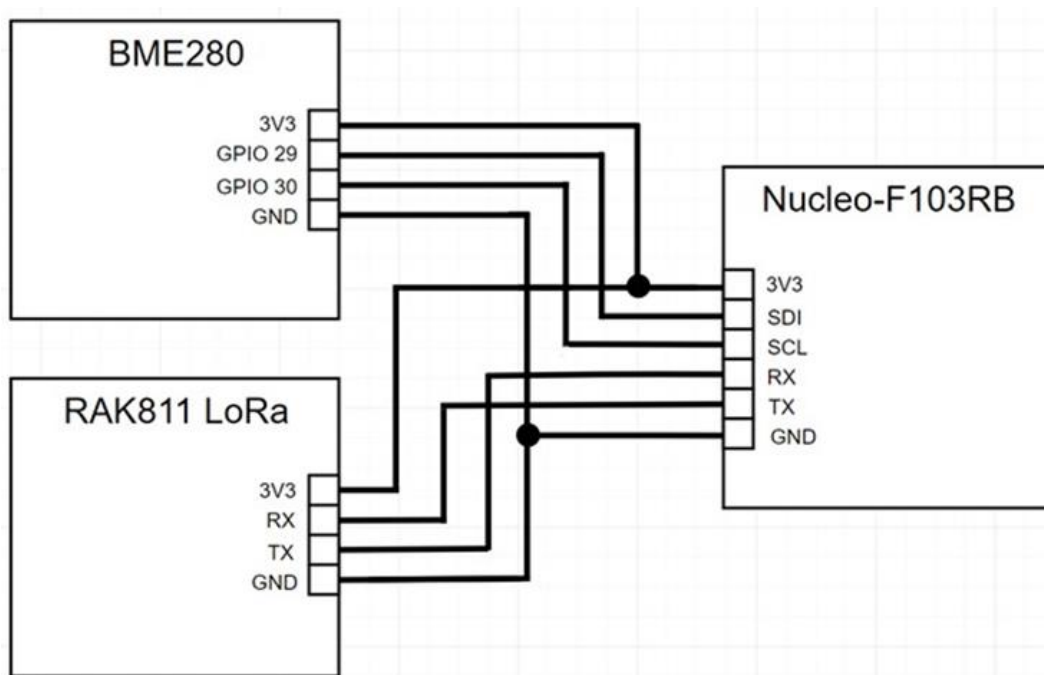


Рисунок 44 – Схема подключения

Все устройства питаются от микроконтроллера, который сам подключен через USB к компьютеру, от которого одновременно загружается необходимая

прошивка, или от аккумулятора. Питание осуществляется выводом на 3.3 В. Заземление – GND. Выходы сенсора «питание» и «заземление» – к выводам питания 3V3 и GND на микроконтроллер, выводы для снятия показаний с сенсоров: DIO29 - к D15 на плате микроконтроллера (SCL), DIO30 – к D14 (SDA). Конечное устройство RAK 811 LoRa подключено при помощи XBee USB Adapter к D2 и D8 (RX и TX).

После этого на микроконтроллер необходимо загрузить прошивку. Если использовать Mbed OS, то сделать нужно следующее: необходимо из репозитория по ссылке <https://github.com/tsr-bloody/GPO-storage> (или отсканировать QR-код, показанный на рисунке 45) скомпилировать и собрать проект (цифра 1), после чего нажать на кнопку (цифра 2, рисунок 46).



Рисунок 45 – QR-код на репозиторий с проектом в GitHub

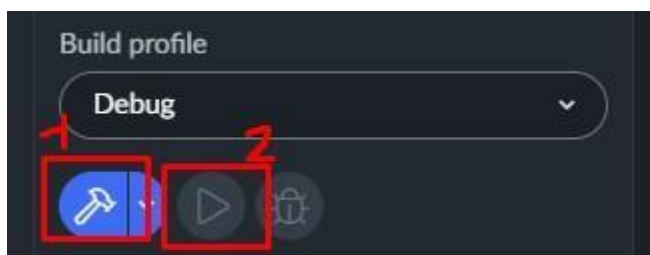


Рисунок 46 – Сборка и загрузка прошивки

После успешной загрузки начнет мигать светодиод на микроконтроллере.

## 5.2. Настройка базовой станции

После прошивки микроконтроллера необходимо подключить базовую станцию к компьютеру с помощью провода USB-mini USB, после чего произвести инициализацию базовой станции на персональном компьютере посредством СОМ порта, используя терминальную программу PuTTY. Далее в той же программы нужно получить необходимую информацию - IP-адрес станции, который необходим для ее настройки через веб-интерфейс. (рисунок 47).

```
am335x-evm login: root
Password:
root@am335x-evm:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 6C:C3:74:3E:E7:C7
          inet addr:192.168.17.207  Bcast:192.168.17.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:10617 errors:0 dropped:19 overruns:0 frame:0
          TX packets:694 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:710391 (693.7 KiB)  TX bytes:94097 (91.8 KiB)
          Interrupt:56

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:419 errors:0 dropped:0 overruns:0 frame:0
          TX packets:419 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:76339 (74.5 KiB)  TX bytes:76339 (74.5 KiB)

root@am335x-evm:~# █
```

Рисунок 47 – Инициализация БС на ПК

Затем необходимо выполнить настройку основных параметров базовой станции через веб-интерфейс (рисунок 48), описанных ниже, все остальные же параметры оставить по умолчанию:

- - настройки подключения к серверу LoRaWAN:
- - логи LoRa.

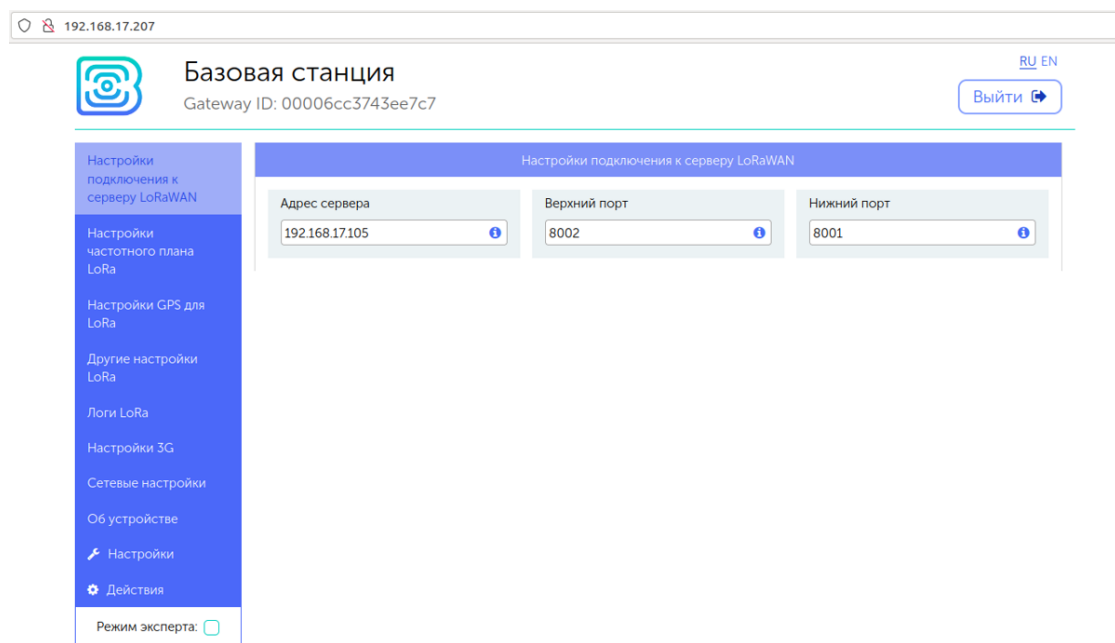


Рисунок 48 – Настройки подключения к серверу LoRaWAN

### 5.3. Настройка сервера

После настройки базовой станции необходимо настроить сервер, в качестве которого используется IOT Vega Server. Данный сервер необходим для управления сетью базовой станции, приема, передачи данных с устройств внешним приложениям и наоборот.

Перед запуском сервера необходимо изменить файл конфигурации “settings.conf” (рисунок 49).



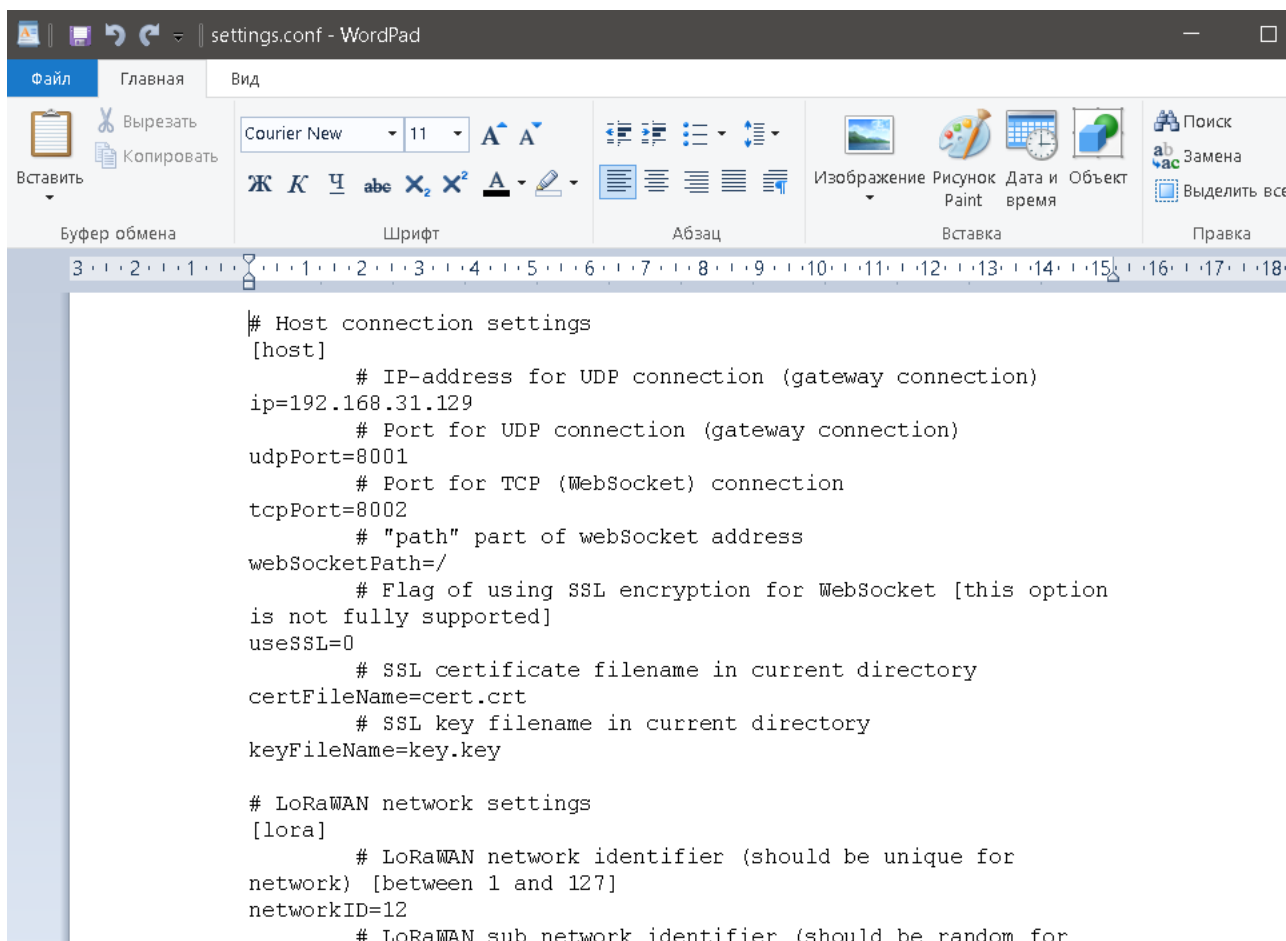
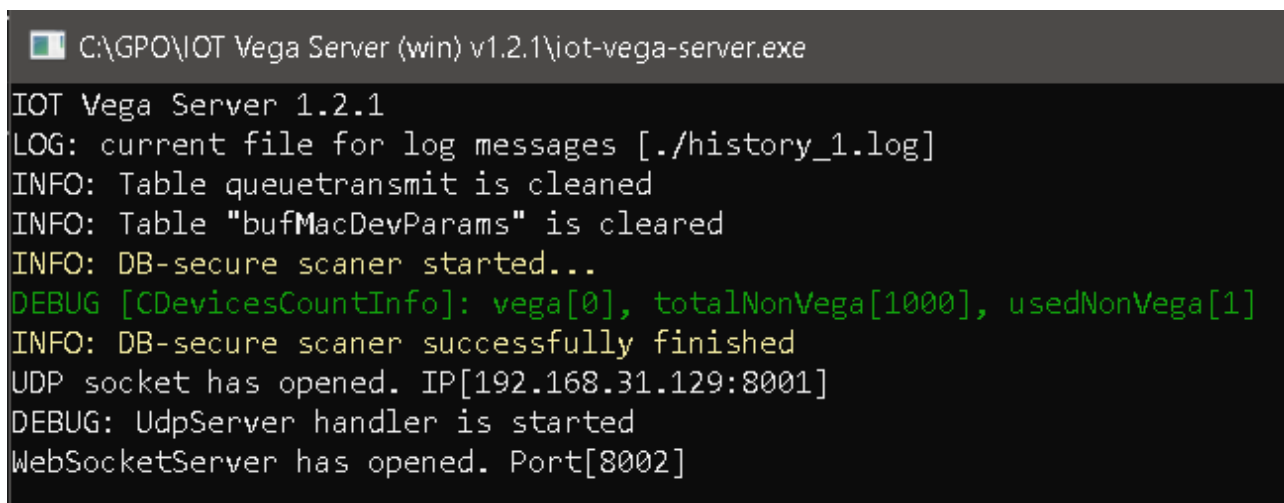


Рисунок 49 – Изменение файла конфигурации

В данном файле необходимо изменить строки “ip”, которая отвечает за IP-адрес компьютера, на котором будет расположен сервер, а также строку “password”, которая отвечает за пароль суперпользователя - он понадобится в дальнейшем при обращении к серверу через приложение IOT Vega AdminTool.

После первичной настройки нужно запустить непосредственно сам сервер. (рисунок 50)



```
C:\GPO\IOT Vega Server (win) v1.2.1\iot-vega-server.exe
IOT Vega Server 1.2.1
LOG: current file for log messages [./history_1.log]
INFO: Table queuetransmit is cleaned
INFO: Table "bufMacDevParams" is cleared
INFO: DB-secure scanner started...
DEBUG [CDevicesCountInfo]: vega[0], totalNonVega[1000], usedNonVega[1]
INFO: DB-secure scanner successfully finished
UDP socket has opened. IP[192.168.31.129:8001]
DEBUG: UdpServer handler is started
WebSocketServer has opened. Port[8002]
```

Рисунок 50 – Запуск сервера

После корректного запуска сервера будут выведены строки “UDP socket has opened” и “WebSocketServer has opened”.

#### 5.4. Настройка приложения для администрирования

Для администрирования сервера нужно использовать приложение IOT Vega Admin Tool.

Перед тем, как начать работать с приложением необходимо настроить файл конфигурации config.js (рисунок 51), а именно изменить постоянный IP-адрес сервера и номер порта в соответствии с ранее измененным файлом конфигурации. Поскольку web-приложение запускается на локальном сервере, то указывается локальный IP компьютера, на котором будет запущен сервер и web-приложение.

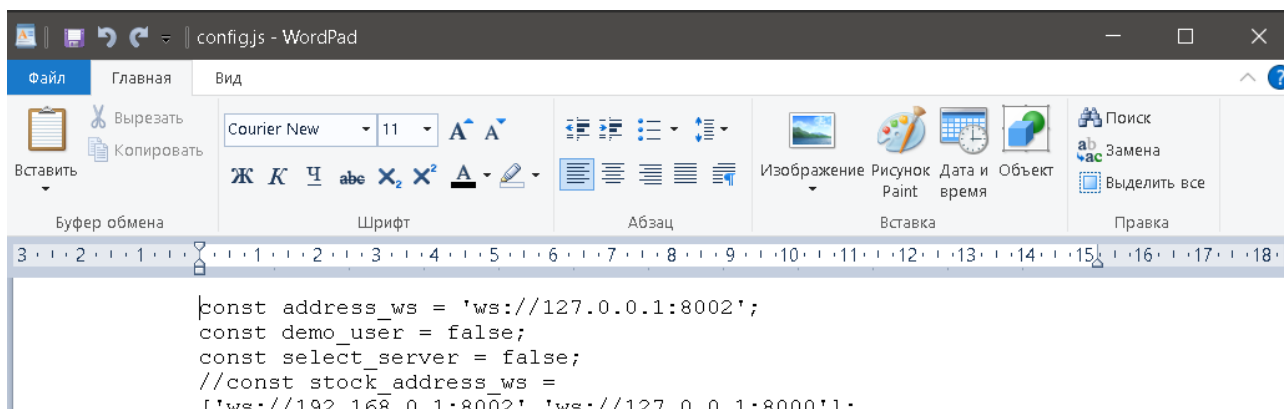


Рисунок 51 – Настройка файла конфигурации веб-приложения Admin Tool

Далее нужно произвести вход в веб-приложение, где используются логин и пароль суперпользователя, прописанные в файле конфигурации сервера. (рисунок 52)

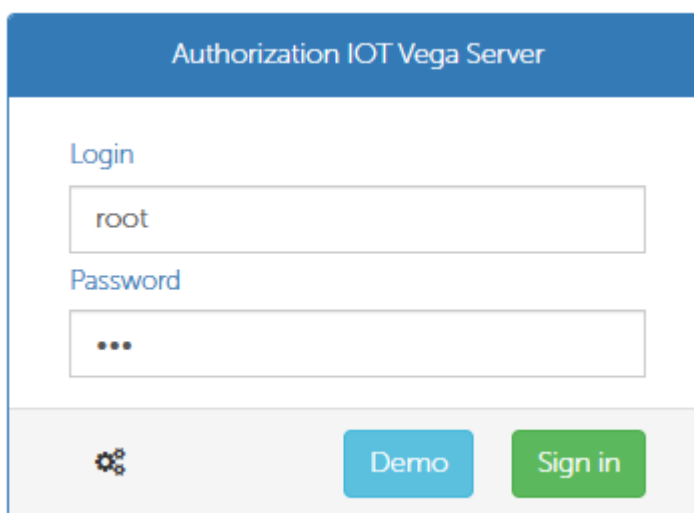


Рисунок 52 – Окно авторизации в приложение IOT Vega Admin Tool

После авторизации в Admin Tool открывается главная страница. На ней можно увидеть информацию о времени, часовом поясе и версии сервера, количество БС, количество подключенных оконечных устройств и количество пользователей. Поскольку используемая базовая станция имеет GPS, то на карте отображается ее примерное местоположение (рисунок 53).

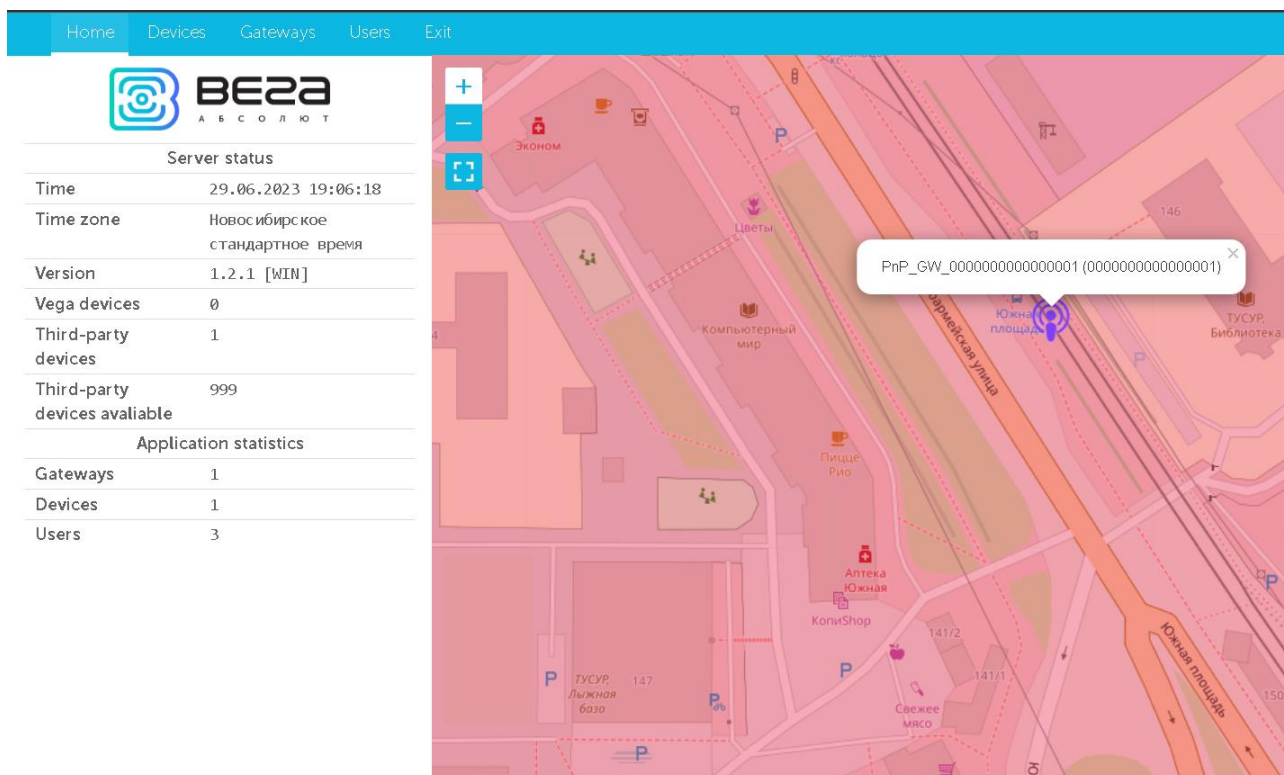
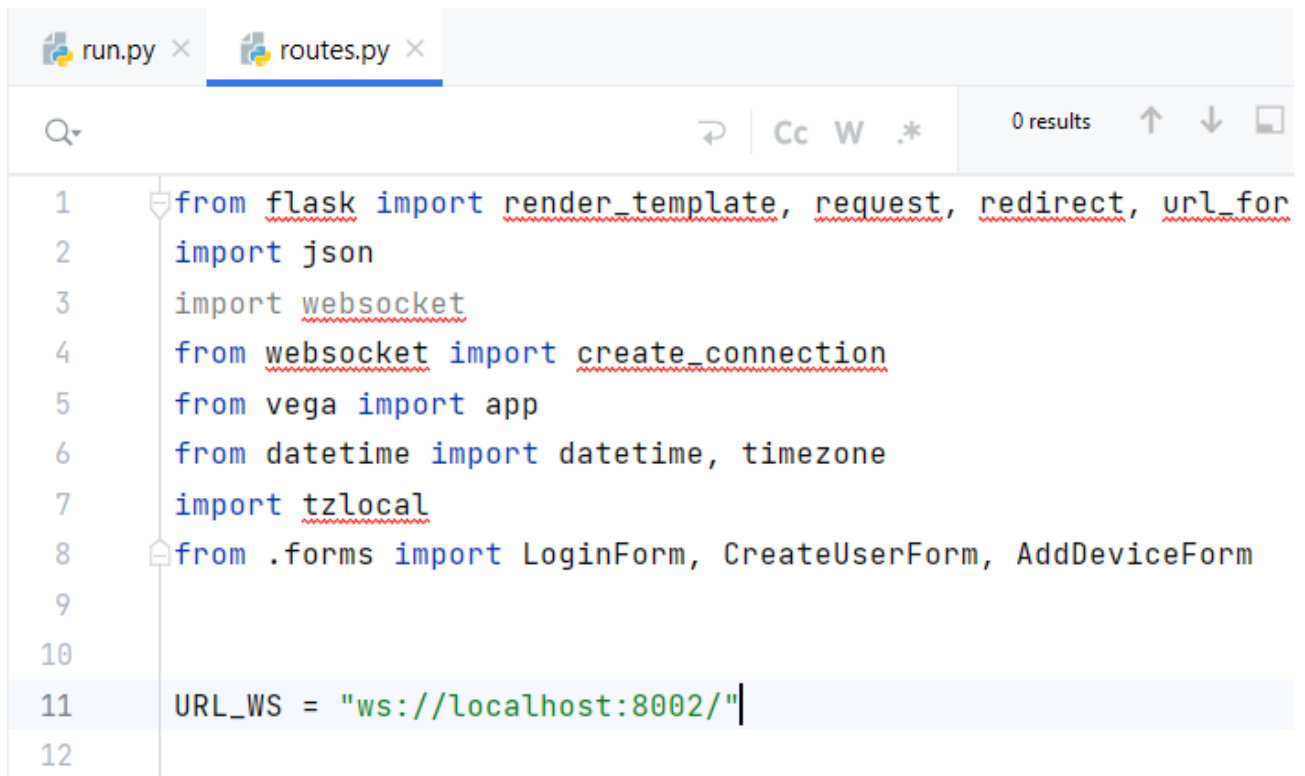


Рисунок 53 – Главная страница IOT Vega Admin Tool

## 5.5. Настройка web-приложения

Перед запуском web-приложения необходимо изменить в коде строки, которые выделены на рисунке 54, если приложение запускается не на локальном сервер. Иначе ничего менять не надо..

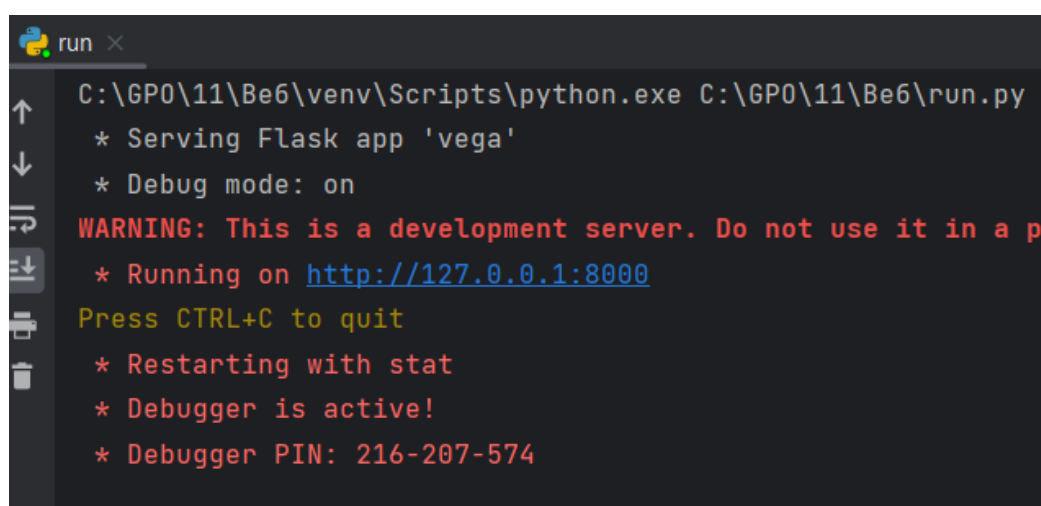


```
1 from flask import render_template, request, redirect, url_for
2 import json
3 import websocket
4 from websocket import create_connection
5 from vega import app
6 from datetime import datetime, timezone
7 import tzlocal
8 from .forms import LoginForm, CreateUserForm, AddDeviceForm
9
10
11 URL_WS = "ws://localhost:8002/"
12
```

Рисунок 54 – Строки подлежащие редактированию

После того, как подключили устройство для сбора параметров и станцию, запустили Vega Server и настроили приложение. Необходимо запустить файл run.py, который после запуска выводит ссылку для перехода в web-приложение.

Запущенное web-приложение представлено на рисунке 55.



```
run x
C:\GP0\11\Be6\venv\Scripts\python.exe C:\GP0\11\Be6\run.py
* Serving Flask app 'vega'
* Debug mode: on
WARNING: This is a development server. Do not use it in a p
* Running on http://127.0.0.1:8000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 216-207-574
```

Рисунок 55 – Ссылка для входа в web-приложение

После авторизации в приложении на графике будет отображаться текущая температура, передаваемая с устройства.

## **Заключение**

В ходе выполнения проекта «Сбор микроклиматических параметров в учебных аудиториях»:

- Была изучена документация API-функций сервера Вега и выбраны те функции, которые необходимы для web-приложения.
- Были изучены принципы работы и обеспечения безопасности протокола LoRaWAN.
- Была разработана архитектура и логика web-приложения.
- Было реализовано web-приложение в соответствии с составленной архитектурой.
- Были созданы и настроены учетные записи пользователей web-приложения.
- Была протестирована вся система на корректность работы, начиная с подключения к серверу, заканчивая выводом графика температур и функциями отдельных учетных записей.

Пояснительная записка была написана в соответствии с требованиями ОС ТУСУР 01-2021.

## Список использованных источников

1. Образовательный стандарт ВУЗа ОС ТУСУР 01-2021. [Электронный ресурс]. – Режим доступа: [https://storage.tusur.ru/files/40668/rules\\_tech\\_01-2021.pdf](https://storage.tusur.ru/files/40668/rules_tech_01-2021.pdf) (дата обращения 26.06.2023).
2. LoRaWAN Message Types. [Электронный ресурс]. – Режим доступа: <https://www.thethingsnetwork.org/docs/lorawan/message-types> (дата обращения 09.02.2023).
3. LoRaWAN Security. [Электронный ресурс]. – Режим доступа: <https://www.thethingsnetwork.org/docs/lorawan/security> (дата обращения 09.02.2023).
4. Среда разработки PyCharm [Электронный ресурс]. – Режим доступа: <https://www.jetbrains.com/ru-ru/pycharm/> (дата обращения 16.03.2023).
5. Robustel R3000 LG4LA [Электронный ресурс]. – Режим доступа: <https://www.robustel.com/en/product/r3000-lg-industrial-lorawan-gateway/> (дата обращения 16.02.2023).
6. Вега БС-1.2. [Электронный ресурс]. – Режим доступа: <https://iotvega.com/product/bs01-2> (дата обращения 16.02.2023).
7. Датчик температуры SAURES Wi-Fi [Электронный ресурс]. – Режим доступа: <https://www.saures.ru/katalog/umnye-datchiki/komplekt-datchik-temperatury-wi-fi/> (дата обращения 16.02.2023).
8. API документация Vega server. [Электронный ресурс]. – Режим доступа: <https://iotvega.com/product/server> (дата обращения 13.04.2023).



## Приложение А

(Обязательное)

Код для взаимодействия с сервером при помощи API

```
@app.route('/login/', methods=['post', 'get'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        login = form.login.data # запрос к данным формы
        password = form.password.data

        # Authorization on VEGA server
        autreq = {"cmd": "auth_req", # Don't change!
                  "login": str(login), # Login name
                  "password": str(password) # password
                 }
        autresp = send_req(autreq)
        if autresp.get("err_string") is None:
            session["command_list"] = autresp.get("command_list")
            session['token'] = autresp.get("token")
            return redirect(url_for('index'))
        else:
            flash("Invalid login/password", 'error')
    return render_template('login.html', form=form)
```

## Приложение Б

(Обязательное)

Код прошивки микроконтроллера

```
#include "mbed.h"
#include "BME280.h"
#include <arm_acle.h>
#include <cstring>
#include <string>
#include <iostream>
#include <cstdlib>
#include <charconv>

#define MAX_DIGITS 10
RawSerial pc(USBTX, USBRX);
RawSerial dev(D8, D2);
BME280 sensor(I2C_SDA, I2C_SCL); void
dev_recv()
{ while(dev.readable())
{ pc.putc(dev.getc());
} } void
pc_recv()
{ while(pc.readable())
{ dev.putc(pc.getc());
}
}

//void gotoSleep(void)
//{

// //включение часов управления PWR
```

```

// RCC->APB1ENR |= (RCC_APB1ENR_PWREN);

// //установка бита SLEEPDEEP в регистре управления системой Cortex
// SCB->SCR |= SCB_SCR_SLEEPDEEP_Msk; // регистр управления системой
SCB

// выбирается обычный сон или глубокий сон (как выбрано в данном случае)

// //выбор режима ожидания

// PWR->CR |= PWR_CR_PDDS; // переход к регистру питания
периферийного устройства или переход к регистру управления и установка 1
в бит PWR_CR_PDDS

// //снятие флага пробуждения, очищение

// PWR->CR |= PWR_CR_CWUF; // если флаг WUF будет иметь значение 1, то
в режим сна устройство не перейдет. Сам по себе WUF в режиме чтения.
Поэтому используем CWUF и через него меняем

// //включение запроса пина для пробуждения Ожидание Прерывания // PWR-
>CSR |= (PWR_CSR_EWUP); // устанавливаем в 1 бит сброса флага
пробуждения.

// в даташите нет этого этапа, но он отчасти нужен, так как используем
пробуждение - переходим в регистр состояния управления питанием и
включаем пробуждение

// //ожидание Прерывания

// __wfi();

//}

void print_f(char temperature[], char pressure[], char humidity[]) {
pc.printf("%i\n", sensor.getTemperature());
pc.printf("-----\n"); for (int i = 0; i <
2; i++)

```

```

{
    pc.printf("%c",
temperature[i]);
} pc.printf("\n");
pc.printf("-----\n");
pc.printf("-----\n");

pc.printf("%i\n", sensor.getPressure());
pc.printf("-----\n"); for (int i =
0; i < 4; i++)
{
    pc.printf("%c",
pressure[i]);
} pc.printf("\n");
pc.printf("-----\n");
pc.printf("-----\n");

pc.printf("%i\n", sensor.getHumidity());
pc.printf("-----\n"); for (int i = 0;
i < 2; i++)
{
    pc.printf("%c",
humidity[i]);
}

pc.printf("\n");
pc.printf("-----\n");
pc.printf("-----\n");
}

//char inttochar(int param)
//{

```

```
//}
```

```
int main() { int q=0; while(1) { char
command_WAKE_UP[2] = {'a', 't'}; for(int i = 0; i
< strlen(command_WAKE_UP); i++)
{
dev.putc(command_WAKE_UP[i]); pc.printf("%c",
command_WAKE_UP[i]);
} dev.putc('\n');
dev.putc('\r');
wait(3);
pc.baud(115200);
dev.baud(115200);
char command_JOIN[9] = {'a', 't', '+', 'j', 'o', 'i', 'n', '\n', '\r'};
if(q==3) // 10 { for(int i = 0; i < 9; i++)
{ dev.putc(command_JOIN[i]);
pc.printf("%c", command_JOIN[i]);
} q=0;
wait(1
0); }
q++;
//printf(data.Temperature, data.Pressure, data.Humidity);
char command_SEND[23] = {'a', 't', '+', 's', 'e', 'n', 'd', '=', 'l', 'o', 'r', 'a', ':', 'l', ':'};
// типа метод по преобразованию инта температуры в символы
char* temp; int k = sensor.getTemperature(); pc.printf("\n%d\n",
sensor.getTemperature()); temp = (char *)malloc(10 *
sizeof(char)); int v = 0; //количество цифр в числе n
//разбиваем на отдельные символы число n while
(k > 9)
```

```

{ temp[v++] = (k % 10) +
'0'; k = k / 10; } temp[v++]
= k + '0'; temp[v] = '\0';
char t;

//инвертируем массив символов
for (int i = 0; i < v / 2; i++)
{ t = temp[i]; temp[i] =
temp[v - 1 - i]; temp[v -
1 - i] = t;
} v = 0; for(int i = 0; i <
strlen(temp);i++)
{
command_SEND[i+15] += temp[i];
} free(temp);

// типа метод по преобразованию инта давления в символы
//char* pres;
//int k2 = sensor.getPressure();
//pres = (char *)malloc(10 * sizeof(char));
//int v2 = 0; //количество цифр в числе n
//разбиваем на отдельные символы число n
//while (k2 > 9)
//{
// pres[v2++] = (k2 % 10) + '0';
// k2 = k2 / 10;
//}
//pres[v2++] = k2 + '0';
//pres[v2] = '\0';
//char t2;

```

```

//инвертируем массив символов
//for (int i = 0; i < v2 / 2; i++)
//{
// t2 = pres[i];
// pres[i] = pres[v2 - 1 - i];
// pres[v2 - 1 - i] = t2;
//}
//v2 = 0;
//for(int i = 0; i < strlen(pres);i++)
//{
// command_SEND[i+15+strlen(temp)] += pres[i];
//}
//free(pres);

// типа метод по преобразованию инта влажности в символы
//char* hum;
//int k3 = sensor.getHumidity();
//hum = (char *)malloc(10 * sizeof(char));
//int v3 = 0; //количество цифр в числе n
//разбиваем на отдельные символы число n
//while (k3 > 9)
//{
// hum[v3++] = (k3 % 10) + '0';
// k3 = k3 / 10;
//}
//hum[v3++] = k3 + '0';
//hum[v3] = '\0';
//char t3;
//инвертируем массив символов

```

```

//for (int i = 0; i < v3 / 2; i++)
//{
// t2 = hum[i];
// hum[i] = hum[v3 - 1 - i];
// hum[v3 - 1 - i] = t2;
//}
//v3 = 0;
//for(int i = 0; i < strlen(hum);i++)
//{
// command_SEND[i+15+strlen(temp)+strlen(pres)] += hum[i];
//}
//free(hum);

pc.baud(115200); dev.baud(115200); for(int i
= 0; i < strlen(command_SEND); i++)
{
dev.putc(command_SEND[i]); pc.printf("%c",
command_SEND[i]);
} dev.putc('\n');
dev.putc('\r');
wait(7);

pc.attach(&pc_recv, Serial::RxIrq); dev.attach(&dev_recv,
Serial::RxIrq);

char command_SLEEP[28] = {'a', 't', '+', 's', 'e', 't', '_', 'c', 'o', 'n', 'f', 'i', 'g',
'=', 'd', 'e', 'v', 'i', 'c', 'e', ':', 's', 'l', 'e', 'e', 'p', ':', 'l'};
for(int i = 0; i < strlen(command_SLEEP); i++)
{

```



```
dev.putc(command_SLEEP[i]); pc.printf("%c",
command_SLEEP[i]);
}

dev.putc('\n');
dev.putc('\r');
//data.Warning = '1';
ThisThread::sleep_for(5000); // или 10000 для демонстрации (а если для
реальной работы, то 2700000, то есть сон 45 минут)
}
}
```