

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)

ИНТЕРНЕТ ВЕЩЕЙ

Отчет о выполнении промежуточного аттестационного этапа группового
проектного обучения (ГПО)
Проект ГПО – КИБЭВС-1904

Ответственный исполнитель проекта:
Студент гр. 739-1

_____ А.А. Лобанов
«22» июня 2022 г.

Проверил:
Руководитель проекта
Старший преподаватель каф. КИБЭВС

_____ О.В. Пехов
(оценка) (подпись)
«22» июня 2022 г.

Принял:
Ответственный за ГПО на кафедре
Доцент каф. БИС, к.т.н.
_____ И.А. Рахманенко
«22» июня 2022 г.

Исполнители проекта ГПО КИБЭВС-1904:

Студент гр. 739-1 _____ А.А. Лобанов

Студент гр. 739-1 _____ Ц.Б. Цыриторов

Студент гр. 730-2 _____ Г.А. Астра

Студент гр. 730-2 _____ К.В. Подойницын

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

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

Групповое проектное обучение

УТВЕРЖДАЮ

Зав. кафедрой КИБЭВС

Шелупанов Александр Александрович

«___» _____ 20__ г.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

на выполнение проекта № КИБЭВС-1904

1. Основание для выполнения проекта: приказ № 3247ст от 26.06.2019.
2. Наименование проекта: Интернет вещей.
3. Цель проекта:
 - изучение технологий модуляции LoRa и сетевого протокола LoRaWAN;
 - создание IoT-сети, основанной на данных технологиях.
4. Основные задачи проекта на этапах реализации:
 - изучение технологий модуляции LoRa и сетевого протокола LoRaWAN;
 - изучение документации оборудования, которое будет использовано для реализации IoT-сети;
 - произвести настройку данного оборудования;
 - создание работоспособной IoT-сети.

5. Научная новизна проекта: Нет.
6. Планируемый срок реализации: Получение результатов ожидается к декабрю 2022 г.
7. Целевая аудитория (потребители): IoT-разработчики.
8. Заинтересованные стороны: ТУСУР.
9. Источники финансирования и материального обеспечения: Нет
10. Ожидаемый результат (полученный товар, услуга): работоспособная IoT-сеть.
11. Руководитель проекта: Пехов О.В
12. Члены проектной группы:
Лобанов Александр Алексеевич 739-1 (ответственный);
Цыриторов Цырен Биликтуевич 739-1;
Астра Григорий Алексеевич 730-2;
Подойницын Кирилл Вадимович 730-2.
13. Место выполнения проекта: ул. Красноармейская, д. 146, 7 этаж, ауд. 707.
14. Календарный план выполнения проекта:

№ Этапа	Наименование этапа	Содержание работы	Сроки выполнения		Ожидаемые результаты этапа
			Начало	Окончание	
1	Изучение API-функций сервера	Изучение возможностей сервера для создания приложения	01.09.2022	29.09.2022	Систематизированная информация, связанная использованием API-функций в создании приложения
2	Решение вопроса питания конечного устройства	Поиск комплектующих и решения по соединению с микроконтроллером	29.09.2022	20.10.2022	Для конечного устройства решен вопрос, связанный с питанием

№ Этапа	Наименование этапа	Содержание работы	Сроки выполнения		Ожидаемые результаты этапа
			Начало	Окончание	
3	Создание корпуса для конечного устройства	Создание или поиск корпуса для конечного устройства	15.09.2022	20.10.2022	Создан корпус для конечного устройства
4	Разработка структуры и логики работы приложения	Разработка структуры и логики работы приложения	27.10.2022	10.11.2022	Разработанная структура и логика работы приложения
5	Реализация добавления и удаления устройств в приложении	Реализация с использованием API-функций сервера добавления и удаления конечных устройств в приложении	15.10.2022	20.10.2022	Реализованные функции по добавлению и удалению конечных устройств в приложении
6	Реализация авторизации в приложении	Реализация функции авторизации с использованием API-функций сервера	15.10.2022	03.11.2022	Реализованная функция авторизации в приложении
7	Реализация соединения приложения с сервером	Реализация с использованием API-функций сервера соединения приложения с сервером	03.11.2022	17.11.2022	Реализованная возможность соединения приложения с сервером
8	Реализация добавления и удаления пользователей	Реализация с использованием API-функций сервера добавления и удаления пользователей	09.11.2022	24.11.2022	Реализованная возможность добавления и удаления пользователей

№ Этапа	Наименование этапа	Содержание работы	Сроки выполнения		Ожидаемые результаты этапа
			Начало	Окончание	
9	Определение набора инструментов для реализации функций приложения	Определение программных инструментов для создания приложения	10.11.2022	17.11.2022	Определенный набор инструментов для реализации функций приложения
10	Реализация в приложении отображения подключенных устройств и зарегистрированных пользователей	Реализация с использование м API-функций сервера отображения подключенных оконечных устройств и зарегистрированных пользователей	24.11.2022	21.12.2022	Реализованная возможность просмотра подключенных устройств и зарегистрированных пользователей в приложении
11	Изучить возможные атаки на LoRa-сеть	Изучить возможные атаки на LoRa-сеть, произвести попытки реализовать их	24.11.2022	22.12.2022	Изучены типовые атаки на LoRaWAN и LoRa-сеть
12	Настроить и протестировать отправку данных с нескольких конечных устройств	Настроить несколько конечных устройств на одновременную отправку данных на сервер	24.11.2022	22.12.2022	Успешная отправка данных на сервер с нескольких устройств, расположенных на разных расстояниях
13	Настроить и протестировать отправку данных с нескольких конечных устройств	Настроить несколько конечных устройств на одновременную отправку данных на сервер	24.11.2022	21.12.2022	Успешная отправка данных на сервер с нескольких устройств, расположенных на разных расстояниях

№ Этапа	Наименование этапа	Содержание работы	Сроки выполнения		Ожидаемые результаты этапа
			Начало	Окончание	
14	Тестирование web-приложения на этапе разработки	Провести функциональное тестирование web-приложения	24.11.2022	15.12.2022	Выполнено тестирование готового функционала приложения и проверена корректность работы приложения
15	Реализация в приложении отображения показаний датчиков	Реализация в приложении отображения показаний датчиков с использованием API-функций сервера	24.11.2022	21.12.2022	В приложении выводятся показания с датчиков в виде таблицы и в виде графика
16	Отчётность	Написание отчёта по ГПО	15.12.2022	22.12.2022	Отчёт по проделанной работе

«01» сентября 2022 г.

Руководитель проекта:
Старший преподаватель каф. КИБЭВС
(должность)

(подпись) Пехов О.В.
(расшифровка)

Члены проектной группы:

(подпись) Лобанов А.А.
(расшифровка)

(подпись) Цыриторов Ц.Б.
(расшифровка)

(подпись) Астра Г.С.
(расшифровка)

(подпись) Подойницын К.В.
(расшифровка)

Реферат

Отчет содержит 81 страниц, 66 рисунков, 15 источников.

LORA, LORAWAN, БАЗОВАЯ СТАНЦИЯ, МИКРОКОНТРОЛЛЕР, РАДИОМОДУЛЬ, СБОР ПАРАМЕТРОВ, АВТОРИЗАЦИЯ, СОЕДИНЕНИЕ С СЕРВЕРОМ, КЛИЕНТСКОЕ ПРИЛОЖЕНИЕ, АТАКИ НА LORA-СЕТЬ.

Объект исследования: Системы интернет вещей.

Цели работы:

- изучение технологий модуляции LoRa и сетевого протокола LoRaWAN;
- изучение документации оборудования, которое будет использовано для реализации IoT-сети;
- произвести настройку данного оборудования;
- создание работоспособной IoT-сети.

Пояснительная записка к групповой проектной работе выполнена в текстовом редакторе Microsoft Word 2016.

Содержание

Введение.....	10
1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ	11
1.1 Безопасность протокола LoRaWAN.....	11
2 ПРАКТИЧЕСКАЯ ЧАСТЬ	14
2.1 Реализация питания конечного устройства от аккумулятора	14
2.2 Типовые угрозы и атаки на LoRa-сеть.....	18
2.3 Ознакомление с документацией на проект	23
2.4 Разработка структуры и логики работы приложения	30
2.5 Определение инструментов для реализации приложения.....	31
2.6 Реализация WEB-приложения	34
2.7 Тестирование WEB-приложения.....	59
Заключение	67
Список использованных источников	69
Приложение А (обязательное) Файл run.py.....	71
Приложение Б (обязательное) Файл routes.py.....	72
Приложение В (обязательное) Файл forms.py.....	79
Приложение Г (обязательное) Файл init.py	81

Введение

Интернет вещей включает в себя множество технологий, которые позволяют различным устройствам передавать между собой или на центральный сервер различные данные, причем по беспроводной связи. Рынок IoT с каждым днем растет и привлекает все больше компаний для создания собственных конкурирующих продуктов, ведь помимо того, что возможности, которые дает интернет вещей, позволяют уменьшить человеческий контроль над каким-либо процессом, в нем сосредоточены большие средства – к 2029 году ожидается оборот около отметки 2.5 триллиона долларов. [1]

Целью проекта «Исследование и создание IoT-сети, основанной на технологиях модуляции LoRa и сетевого протокола LoRaWAN» является настройка оборудования, способного взаимодействовать между собой при помощи данных технологий, для создания IoT-сети, по которой будет происходить передача данных с конечных устройств на сервер, создание приложения, в котором будет производиться работа с конечными устройствами и полученными данными, а также изучение угроз и способов защиты от атак на сеть.

1 ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Безопасность протокола LoRaWAN

LoRaWAN 1.0 использует ряд ключей безопасности: NwkSKey, AppSKey и AppKey. Все ключи имеют длину 128 бит. [2]

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). [3]

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:

- $NwkSKey = \text{aes128_encrypt}(\text{AppKey}, 0x01 \mid \text{AppNonce} \mid \text{NetID} \mid \text{DevNonce})$
- $\text{AppSKey} = \text{aes128_encrypt}(\text{AppKey}, 0x02 \mid \text{AppNonce} \mid \text{NetID} \mid \text{DevNonce})$
- 0x01, 0x02 – номер ключа (1 – NwkSKey, 2 – AppSKey),
- NetID – идентификатор сети. [4]

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. [5]

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 счётчики 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 счётчика меньше или равной последнему, то данное сообщение игнорируется.

MIC обеспечивает целостность и подлинность сообщения. Код целостности сообщения вычисляется по всем полям сообщения, а затем

добавляется к самому сообщению.

Безопасный обмен сообщениями представлен на рисунке 1.1.

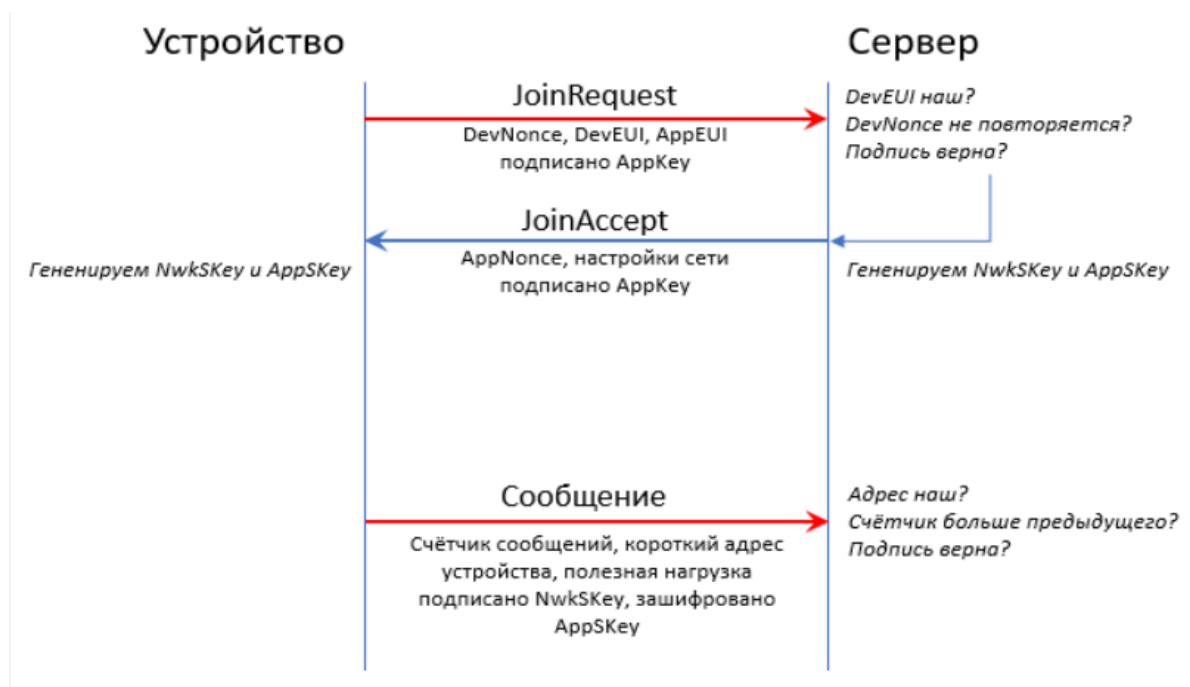


Рисунок 1.1 – Безопасный обмен сообщениями в сети LoRaWAN

2 ПРАКТИЧЕСКАЯ ЧАСТЬ

2.1 Реализация питания конечного устройства от аккумулятора

На данном этапе производился поиск и подключение к конечному устройству аккумулятора. Стояли следующие критерии выбора:

- Возможность подключения к плате.
- Большая продолжительность работы.

Для устройства требуется источник рабочего напряжения от 2,0 до 3,6 В (VDD). Встроенный регулятор используется для подачи внутреннего цифрового питания напряжением 1,8 В. Часы реального времени (RTC) и резервные регистры могут питаться от напряжения VBAT, когда основной источник питания VDD отключен. [6]

На рисунке 2.1 показана схема источника питания, подключаемого к плате.

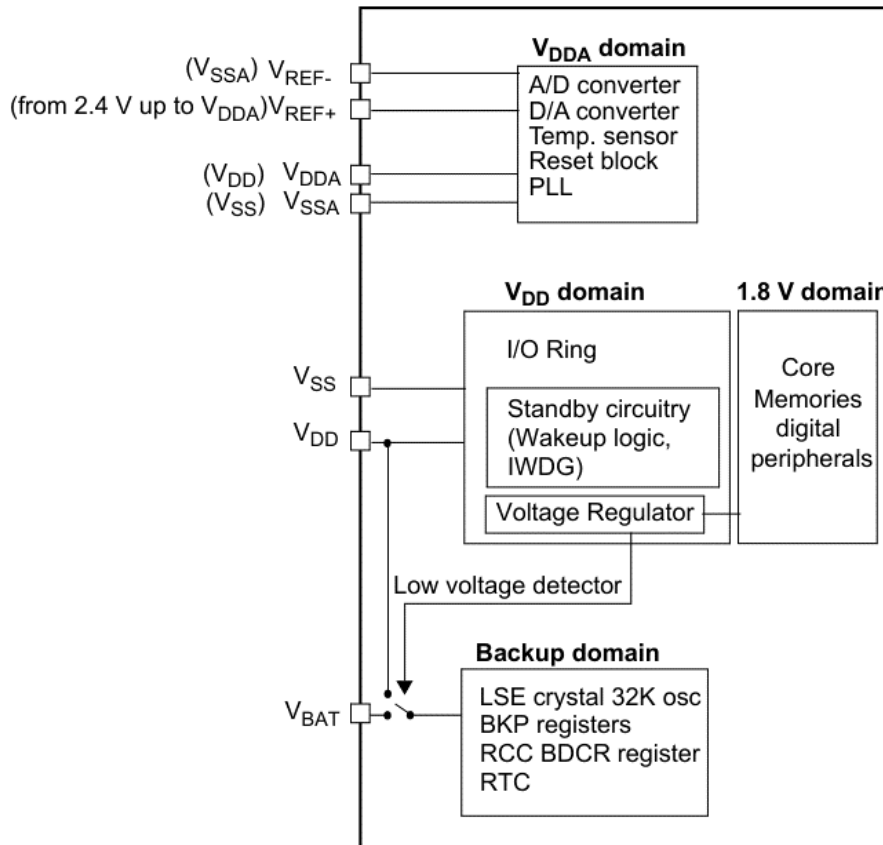


Рисунок 2.1 - Обзор подключения источника питания

“Плюс”: +5V или +3.3V. Может быть подписан на плате как 5V, 3.3V, 3V3, VCC. “Минус”: общий провод, 0V. Может быть подписан на плате как GND. [7]

Напряжения, которые показаны на схеме, нельзя превышать. То есть указанный вольтаж от 2,4, до 3,6 или 5 В нельзя превышать подключаемым аккумулятором. При нарушении такого правила последует 100% повреждение проводов и самой платы, а также других подключаемых к ней модулей.

В большинстве своем с таким напряжением батареи имеют неподходящий форм-фактор: аккумуляторы типа AA или AAA, CR и другие, которыми питать систему будет проблематично.

Варианты для питания в пин 5V:

- Для стабильных 5V на выходе – литиевый аккумулятор и повышающий до 5V модуль (дополнительный). У таких модулей запас по току 2А, плохое энергосбережение, что для разрабатываемой системы не подойдет.

- Литиевый аккумулятор – напряжение на пине 5V и GPIO будет 4.2-3.5V. Работа МК от напряжения ниже 4V также возможна, но очень нестабильна.

- Пальчиковые батарейки (AAA или AA) – 3 штуки реализуют 4.5-3V, что граничит с риском нестабильной работы. Если соединять 4 единицы – более стабильно. Но небольшая емкость, а также отсутствие возможности заряжать снова.

- Платы с кварцем (тактовым генератором) на 8 МГц позволяют питать схему от низкого напряжения (2.5V), но стабильная работа не гарантируется.

Также, изучая документацию, был найден способ подключить источник питания большего напряжения. По контакту VIN на плате можно соединять источники питания с 7-12 В. Как правило, лучше подносить питание через пин Vin от блока питания, так как линейный стабилизатор даёт очень качественное напряжение и вероятность нестабильной работы уменьшается.

Питание через контакт VIN:

- Блок питания с напряжением 7...12 Вольт. Вариант
- 9V батарейка “Крона” – плохой вариант, так как емкость кроны крайне мала.
- Сборка из двух-трёх литиевых аккумуляторов: напряжение 12.6.. 9V в процессе разряда.
- Сборка из одного литиевого аккумуляторов: напряжение 8.4.. 6V в процессе разряда. У этого и предыдущего вариантов емкость достаточно большая, чтобы система наподобие созданной могла работать достаточно долго. Еще одним преимуществом является возможность заново заряжать аккумулятор. Что касается энергосбережения, то стоит сказать, что используемый стабилизатор при питании через VIN потребляет ток, поэтому часть заряда будет уходить ему. Но ввиду большой емкости потерями на стабилизатор можно пренебречь.

На рисунке 2.2 представлена структурная схема соединения батареи с платой. Как видно, соединение идет по двум контактам – VIN и GND.

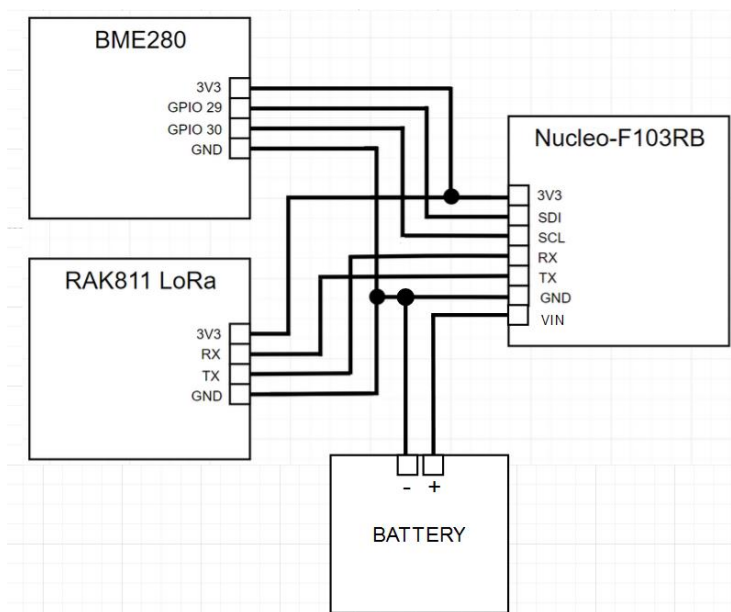


Рисунок 2.2 - Структурная схема устройства

На рисунке 2.3 представлено устройство вне корпуса.

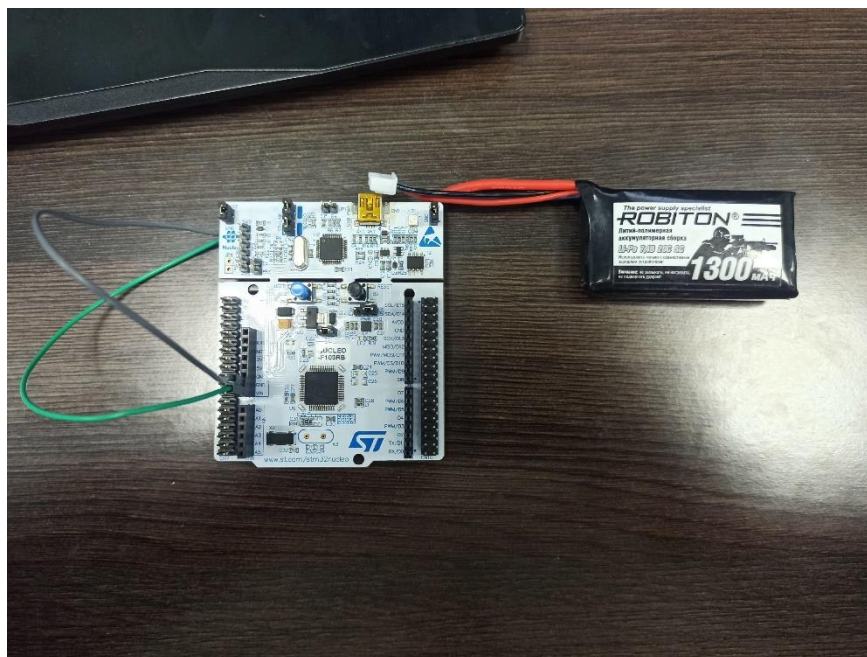


Рисунок 2.3 - Устройство с подключенной батареей

В качестве корпуса выступила распаячная коробка размерами 150x110x70 мм. Размер был выбран так, чтобы никакие контакты не задевали друг друга (те, что нет возможности изолировать полностью), а также, чтобы все компоненты конечного устройства могли поместиться. У корпуса имеется степень защиты IP 44, что позволяет при необходимости располагать устройство на улице, а также в помещении с повышенной влажностью.

Еще одним преимуществом является наличие отверстий, которые можно закрыть резиновыми прокладками. Их удобство заключается в том, что у используемого радиомодуля присутствует антенна, которую закрывать в корпусе не желательно, так как это будет препятствием по передаче данных на сервер и которую как раз через данные вырезы в корпусе можно вынести наружу. Аналогично – с кабелем для отладки работы конечного устройства. При возникновении сбоя в работе будет иметься быстрый доступ к микроконтроллеру и его подключению к компьютеру без извлечения из корпуса.

На рисунке 2.4 представлено конечное устройство в корпусе.



Рисунок 2.4 - Устройство в корпусе

2.2 Атаки на LoRa-сеть

В корпусе ВУЗа устанавливаются системы контроля за параметрами температуры, влажности и др. Датчик температуры подключается с помощью радиомодуля по беспроводному протоколу LoRaWAN. За физическую безопасность систем отвечают системы видеонаблюдения в аудиториях, замки, датчики движения и открытия дверей, а за информационную безопасность – встроенные средства протокола, отвечающие за шифрование, добавление дополнительных вставок для контроля целостности и др. Также дополнительные решения могут усилить безопасность передачи данных и их обработки.

Конечные устройства и сервер «общаются» между собой, используя на канальном уровне открытый протокол LoRaWAN, используя модуляцию LoRa на физическом. В основе нее лежит техника расширения спектра – линейная частотная модуляция. Она позволяет обеспечить передачу данных на дальние расстояния и низкое энергопотребление. [8]

Одной из задач корректной настройки сети – выбор так называемого коэффициента расширения спектра SF: он определяет разрядность символа данных (в битах), передаваемого через радиointерфейс за время T_{sym} . Соответственно, скорости передачи данных составит с учетом используемой частоты, разделенной на 3 канала (863 – 870 МГц), составят от 0,3 до 22 кбит/с. Сложность выбора заключается в том, что сообщения с одинаковым спектром сигнала могут сталкиваться друг с другом, тем самым будет нарушаться целостность или вовсе потеряют структуру кадра LoRaWAN. [9]

Для наглядности ниже приведена теоретическая отправка сообщений в случайное время с использованием различных SF на рисунке 2.5.

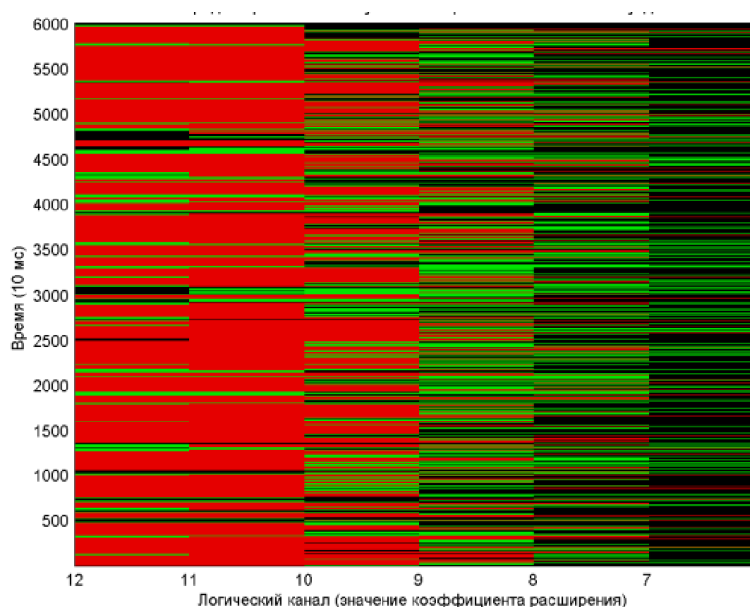


Рисунок 2.5 - Моделирование передачи сообщений LoRa

Как видно из графика, при наибольшем значении коэффициента расширения практически все кадры перекрываются друг другом.

LoRaWAN является относительно новым протоколом, соответственно, уровень его безопасности развит недостаточно хорошо. Несмотря на то, что технология LoRa предусматривает некоторые механизмы безопасности, такие как шифрование и цифровая подпись, уровень безопасности проработан недостаточно.

В IoT одним из самых известных уязвимых мест является этап добавления нового устройства в сеть. Решением выступает активация

устройства посредством процедуры присоединения. Он необходим, чтобы контролировать доступ устройств в сеть и предотвращать их взаимодействие с другими объектами сети. В проекте используется аутентификация ОТАА, в котором используются динамические ключи шифрования. Вторым способом является активация АВР, который имеет статические ключи, т.е. после перезагрузки конечного узла они не будут изменены. Также, сама по себе процедура присоединения здесь отсутствует.

В протоколе существуют так называемые счетчики кадров, которые находятся как на конечном устройстве, так и на сервере. У них идет счет, соответственно, пришедших и ушедших кадров (на узле и на сервере разные счетчики). И данный подход небезопасен, так как после перезагрузки активированного устройства или обмена сообщениями при присоединении счетчик сбрасывается на 0. Соответственно, используя те же ключи, счетчик будет заново вести счет кадров. В таком случае злоумышленник может перехватить сообщения на последней сессии с большими значениями счетчика и использовать его в текущей сессии. К слову, такому подвержена и активация ОТАА.

Реализовать данную угрозу можно с использованием анализатора трафика (к примеру, WireShark) и передатчик LoRa для повторного воспроизведения сообщений. В случае данного проекта, где участвовать будет не так много устройств на первых этапах внедрения в учебный корпус, для реализации потребуется достаточно большое время. Но здесь появляется другая проблема – так как узлы находятся в аудиториях, доступ к ним будет у всех, кто имеет доступ в аудитории. И при должных навыках после перезагрузки устройства уже не нужно будет ждать переполнения счетчика.

Если говорить про атаки на шлюз, а в данном случае им является базовая станция, то здесь возможна ситуация создания вредоносного шлюза, который может быть добавлен в сеть через UDP-spoofing. То есть, такая уязвимость заключается в том, что сообщение от сервера АСК при установлении связи не содержит информации о том, какое сообщение оно действительно

подтверждает, оно лишь только подтверждает последнее полученное сообщение. Поэтому возможно, что взломанный вредоносный шлюз может сохранить подтверждение и поддерживать его для будущих сообщений, поступивших от конечных устройств. Чтобы это реализовать необходим доступ к шлюзу, навыки получения и распознавания АСК сообщений (опять же делается через сниффер) и возможности по отправке АСК-сообщений конечному устройству. Таким образом, будет нанесен большой ущерб всей сети, так как злоумышленник будет производить полный контроль над отправляемыми сообщениями от сервера.

Также есть еще одна угроза, связанная с изменением сообщения, которое передается от сетевого сервера к серверу приложений. В модели предполагается, что канал между двумя серверами не защищен со стороны пользователя. Соответственно, источники могут быть как внутренние, так и внешние. Объектом защиты будет передаваемая информация. В результате реализации атаки будет изменение данных, то есть нарушена целостность объекта защиты. Схематично модель представлена на рисунке 2.6. [10,11]

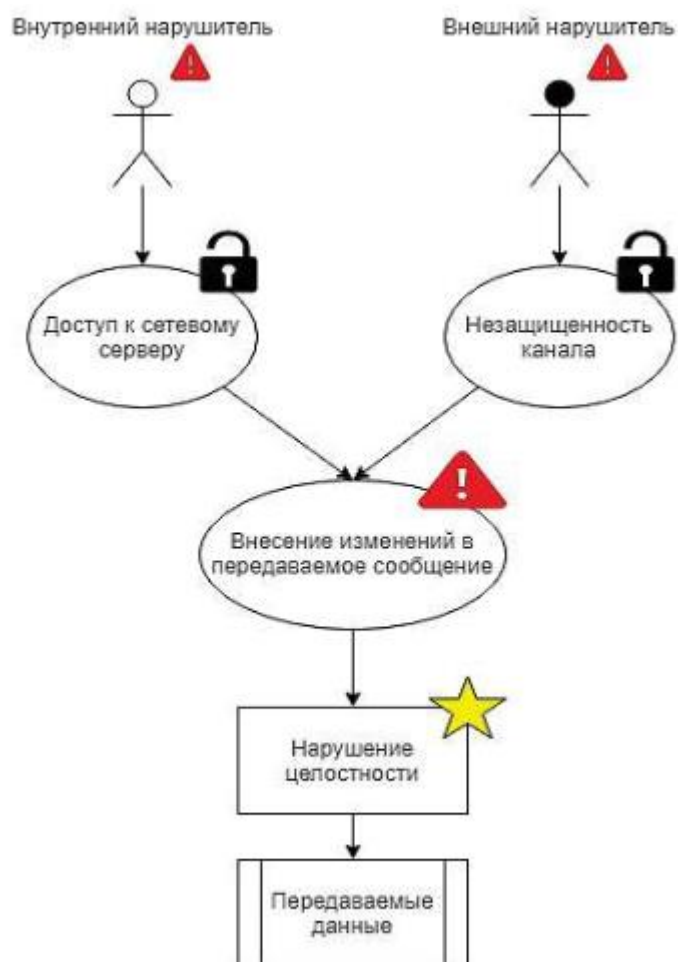


Рисунок 2.6 - Структурная схема угрозы

Внутренний нарушитель может реализовать угрозу через доступ к сетевому серверу, через который и проходят сообщения. То есть их изменение произойдет до отправки.

Внешний нарушитель будет использовать уязвимости канала передачи. Использоваться может MITM – атака «человек посередине». Злоумышленник ретранслирует связь в канале, то есть передаваемые между сетевым сервером и сервером приложений сообщения будут переходить через нарушителя.

Данные шифруются ключом шифрования сессии – AppSKey. Он известен только конечному устройству и серверу приложений. Используя счетчик, шифрование производится алгоритмом AES. Производится операция XOR исходного текста данных и зашифрованного AES значения счетчика в данный момент времени. Поэтому для реализации атаки MITM злоумышленнику необходимо знать и структуру данных, что уже практически

невозможно. Но, допуская такое, изменить данные можно с помощью атаки манипуляции битами. Изменения происходят таким образом, чтобы контрольная сумма осталась прежней. Эта атака обычно происходит в тех случаях, когда функция шифрования принимает некоторые входные данные, добавляет случайную строку и добавляет к ней другую строку перед ее шифрованием. С предыдущим блоком шифртекста и последующим блоком производится операция XOR, для того, чтобы снять наложение текста, в итоге – расшифрованный текст. Таким образом, если будет изменен, например, один байт шифртекста в предыдущем блоке, то изменится один байт открытого текста в следующем блоке.

Средством защиты от подобной угрозы может служить добавление к сообщению кода аутентификации, который основан на сессионном ключе приложения. Размер будет увеличен на 4 байта. С помощью него может быть проверена целостность, так как при изменении любого бита значение кода будет другим. Соответственно, такое сообщение будет отброшено сервером.

2.3 Ознакомление с документацией на проект

Перед началом работы над проектом необходимо было изучить документацию на него, в особенности API-документацию сервера ВЕГА для определения функций, необходимых для дальнейшей разработки функционала и логики приложения. К данным функциям относятся: авторизация пользователя (Рисунок 2.7), получение списка зарегистрированных пользователей (Рисунок 2.8), получение списка подключенных устройств (Рисунок 2.9), возврат сохранённых данных, полученных с устройств (Рисунок 2.10), добавление учетной записи пользователя (Рисунок 2.11), удаление учетной записи пользователя (Рисунок 2.12) [12].

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 2.7 – Функция для авторизации пользователя

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 2.8 – Функция получения списка зарегистрированных пользователей

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 2.9 – Функция получения списка подключенных устройств

Return saved data from device

Request message:

```
{
  "cmd": "get_data_req",
  "devEui": string,
  "select"? : {
    "date_from"? : integer, //[[optional] Extra optional for searching
                        //[[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 device4
      "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 2.10 – Функция возврата сохраненных данных, полученных с устройства

Add new user or modify settings of existed ones (manage_users_req, manage_users_resp)

Request message:

```
{
  "cmd": "manage_users_req",
  "user_list":
  [
    {
      "login": string,           // User login as string
      "password?": string,      // [optional] – password string. Should be exist when new user is added
      "device_access": string,   // [optional] – access level to devices (see below possible values). If
                                // "FULL", "devEui_list" would be ignored ("SELECTED" - default)
      "consoleEnable": bool,     // [optional] – enable to connect to console subchart with debug
                                // information (false – is default)
      "devEui_list?":           // [optional] – list of DevEui that may be accessible by user
      [                         // By default is empty
        "devEui_1",
        ...,
        "devEui_n"
      ],
      "command_list?":          // [optional] – list of commands groups that may be accessible by user
      [                         // By default is empty
        "command_1",
        ...,
        "command_n"
      ],
      "rx_settings?":           // [optional] – setting of online receiving messages
      {
        "unsolicited?": boolean, // [optional] – online message is sending [true] or not sending [false -
                                // default]
        "direction?": string,    // [optional] – direction of possible online messages (see below)
        "withMacCommands?": boolean // [optional] – online message contains MAC commands
      }
    }, ...
  ]
}
```

Possible values for "token":

- "FULL" – user have access to all devices;
- "SELECTED" – user have access to device that was selected by administrator.

Possible string values of "direction":

- "UPLINK" – data from device to server;
- "DOWNLINK" – data from server to device;
- "ALL" – all data from and to device.

Response message:

```
{
  "cmd": "manage_users_resp",
  "status": boolean,
  "err_string?": string // [optional exist if "status" is false] – string code of error
  "add_user_list":
  [
    {
      "login": string,
      "status": boolean // User's adding or updating status
    }, ...
  ]
}
```

Possible values for "err_string":

- "inaccessible_command" – returns if current user don't have access for this command
- "invalid_cmd" – returns if command don't contain "user_list";

Рисунок 2.11 – Функция для добавления учетной записи пользователя

Delete registered users (delete_users_req, delete_users_resp)

Request message:

```
{
  "cmd": "delete_users_req",
  "user_list":
  [
    "login_1",
    ...,
    "login_n"
  ]
}
```

Response message:

```
{
  "cmd": "delete_users_resp",
  "status": boolean,
  "err_string"? : string,      //[optional exist if "status" is false] – string code of error
  "delete_user_list":
  [
    {
      "login": string,
      "status": boolean,
    }, ...
  ]
}
```

Possible values for "err_string":

- "inaccessible_command" – returns if current user don't have access for this command;
- "invalid_cmd" – returns if command don't contain "delete_user_list";
- "empty_input_parameter_list" – returns if "delete_user_list" is empty

Example request message:

```
{
  "cmd": "delete_users_req",
  "user_list":
  [
    "user1",
    "user2"
  ]
}
```

Example response message:

```
{
  "cmd": "delete_users_resp",
  "status": true,
  "delete_user_list":
  [
    {
      "login": "user1",
      "status": true
    },
    {
      "login": "user2",
      "status": false
    }
  ]
}
```

Рисунок 2.12 – Функция для удаления учетной записи пользователя

Выделенные функции из документации сервера ВЕГА были использованы для дальнейшего выполнения поставленной задачи. а именно, разработка логики и структуры приложения. Поскольку, зная функционал сервера, можно разработать приложение с учётом имеющихся функций и критериев, установленных функционалом сервера.

2.4 Разработка структуры и логики работы приложения

После ознакомления с документацией на проект необходимо было разработать UML диаграмму классов для разработки WEB-приложения.

На рисунке 2.13 представлена UML диаграмма классов.

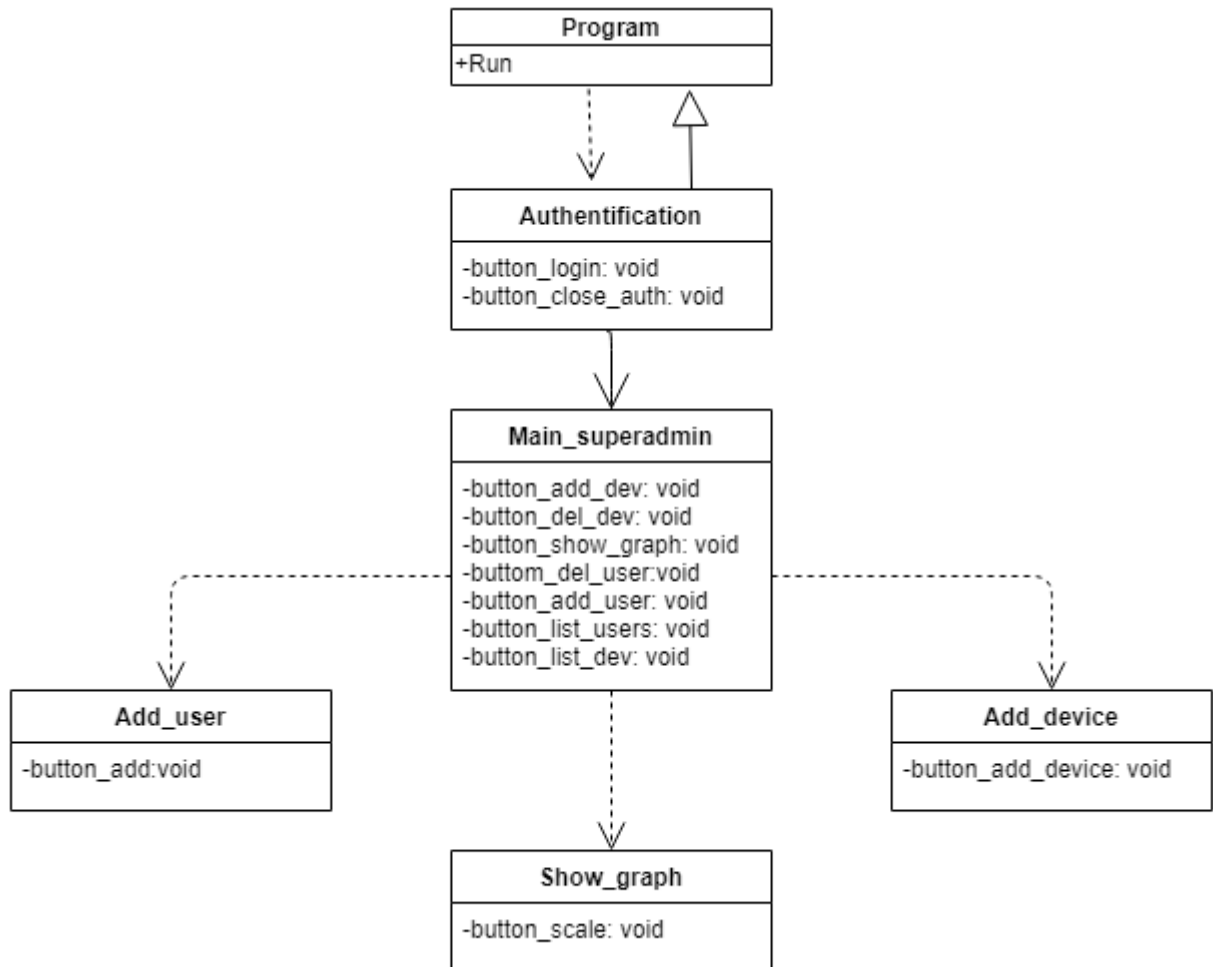


Рисунок 2.13 – UML диаграмма

После создания диаграммы классов необходимо расписать функции кнопок, которые будут выполняться при их активации.

Функции кнопок:

- button_login – проверка совпадения введенных логина и пароля с базы данных (Authentification);
- button_close_auth – выход из окна авторизации и закрытие программы (Authentification);

- button_add_dev – перенаправляет на страницу добавления устройства (Main_superuser);
- button_add_device – добавление нового устройства (Add_device);
- button_del_dev – удаление устройства (Main_superuser);
- button_show_graph – отображение графика (Main_superuser);
- button_add_user – перенаправление на окно добавления нового пользователя (Main_superuser);
- button_del_user – удаление пользователя;
- button_add – добавление нового пользователя, путём занесения логина и пароля в базу данных (Add_user);
- button_scale – изменение количества точек для построения отображаемого графика.

2.5 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 заниматься написанием кода.

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 изначальная 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 [13].

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 2.1.

Таблица 2.1 - Сравнение фреймворков.

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

По итогу сравнения по установленным критериям был выбран фреймворк Flask [14].

2.6 Реализация WEB-приложения

На этот семестр была поставлена задача написать web-приложение для взаимодействия с сервером IoT Vega Server. Web-приложение было написано на языке программирования Python с использованием фреймворка Flask. В этом приложении реализованы такие функции как: авторизация пользователей, подключение к серверу Vega, получение данных с устройства, подключение и удаление новых устройств и получение списка подключенных устройств.

В процессе написания web-приложения было необходимо взаимодействовать с сервером Vega. Сетевой сервер IOT Vega Server – это инструмент для организации сетей стандарта LoRaWAN любого масштаба. Предназначен для управления опорной 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. WebSocket – протокол связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени.

На рисунке 2.14 представлена блок-схема функции подключения к серверу Вега и отправки методов на него.



Рисунок 2.14 – Блок-схема функции подключения к серверу Вега

На рисунке 2.15 представлена функция, которая отвечает за подключение к серверу Вега и отправку 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 2.15 – Подключение к серверу

Блок-схема функции, которая отвечает за авторизацию на сервере Вега представлена на рисунке 2.16.



Рисунок 2.16 – Блок-схема функции авторизации на сервере Вега

На рисунке 2.17 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 2.18).

```

D:\Учеба\ГПО\IoT Vega Server (win) v1.2.1\iot-vega-server.exe
[CWebSocketServer] New connection
[CWebSocket::closeConnectionSlot]
  
```

Рисунок 2.17 – Событие о новом подключении

```

@app.route('/login/', methods=['post', 'get'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        login = form.login.data
        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("Неверный логин/пароль", 'error')
    return render_template('login.html', form=form)

```

Рисунок 2.18 – Авторизация на сервере Vega

Страница авторизации представлена на рисунке 2.19.

ТУСУР ГПО КИБЭВС-1904

Вход в учетную запись

Логин:

root

Пароль:

.....

Войти

Рисунок 2.19 – Страница авторизации

На рисунке 2.20 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 2.20 – Выход из приложения

На рисунке 2.21 представлена блок-схема функции, которая отвечает за отображение главной страницы.

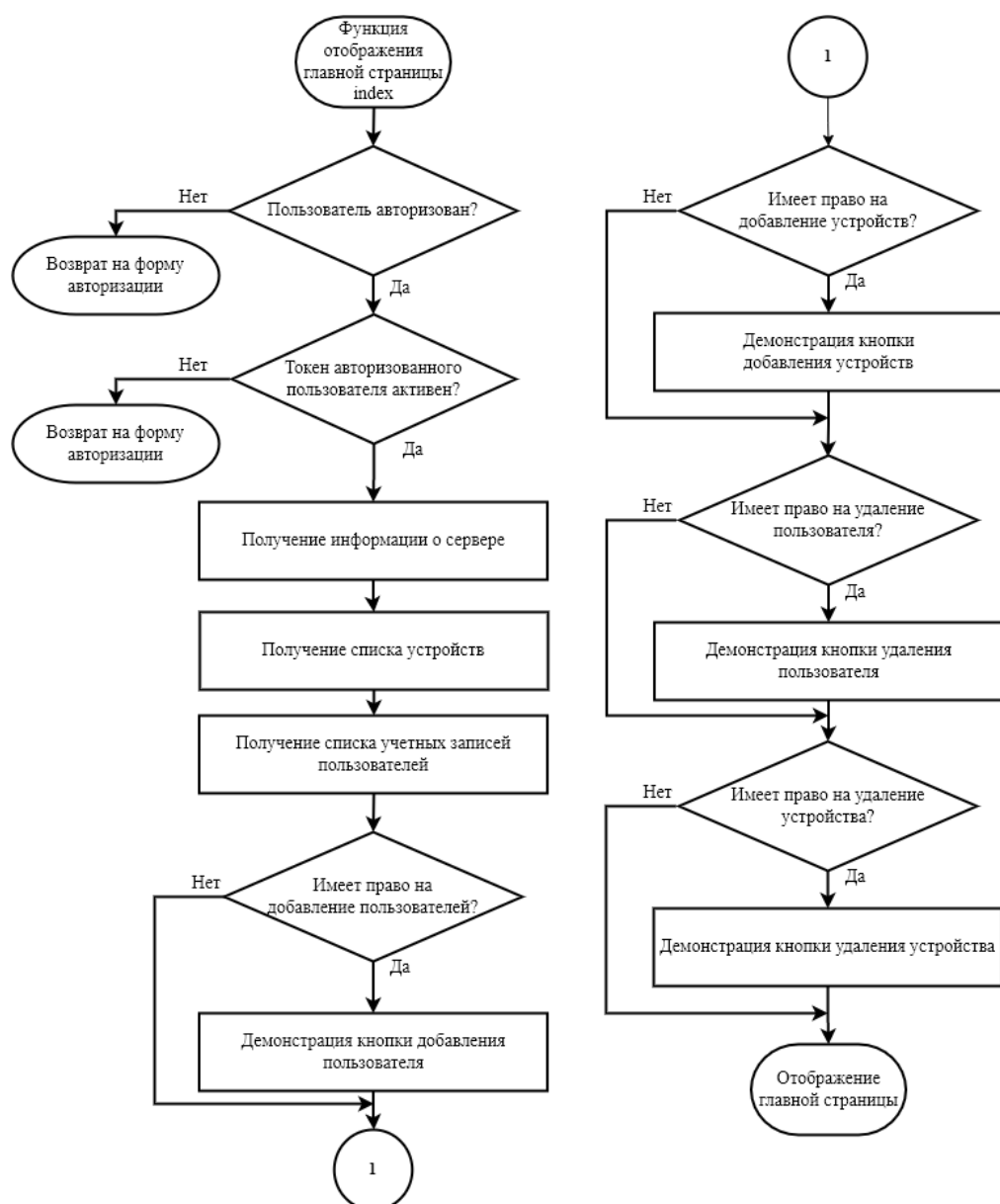


Рисунок 2.21 – Блок-схема функции главной страницы

На рисунке 2.22 представлена функция, которая отвечает за отображение подключенных устройств и зарегистрированных пользователей. 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 2.22 – Отображение подключенных устройств

Вид данной страницы представлен на рисунке 2.23.

ТУСУР ГПО КИБЭВС-1904

Подключенные устройства Список пользователей

Создать учетную запись

Добавить устройство

Выход

devName	AppEui	devEui	
dev2_2909	AC1F09FFF8680811	AC1F09FFFE015302	Удалить
device_number_one	AC1F09FFF8680811	AC1F09FFFE015304	Удалить

Время: 2022-12-20 19:50:29

Город: Томск (зима)

Рисунок 2.23 – Подключенные устройства

Здесь же можно перейти на вкладку списка пользователей (рисунок 2.24).

ТУСУР ГПО КИБЭВС-1904

Подключенные устройства **Список пользователей**

Login	
admin	Удалить
temperature_admin	Удалить
pressure_admin	Удалить
proverka	Удалить

Рисунок 2.24 – Список пользователей

Блок-схема функции, которая отвечает за добавление устройства представлена на рисунке 2.25.

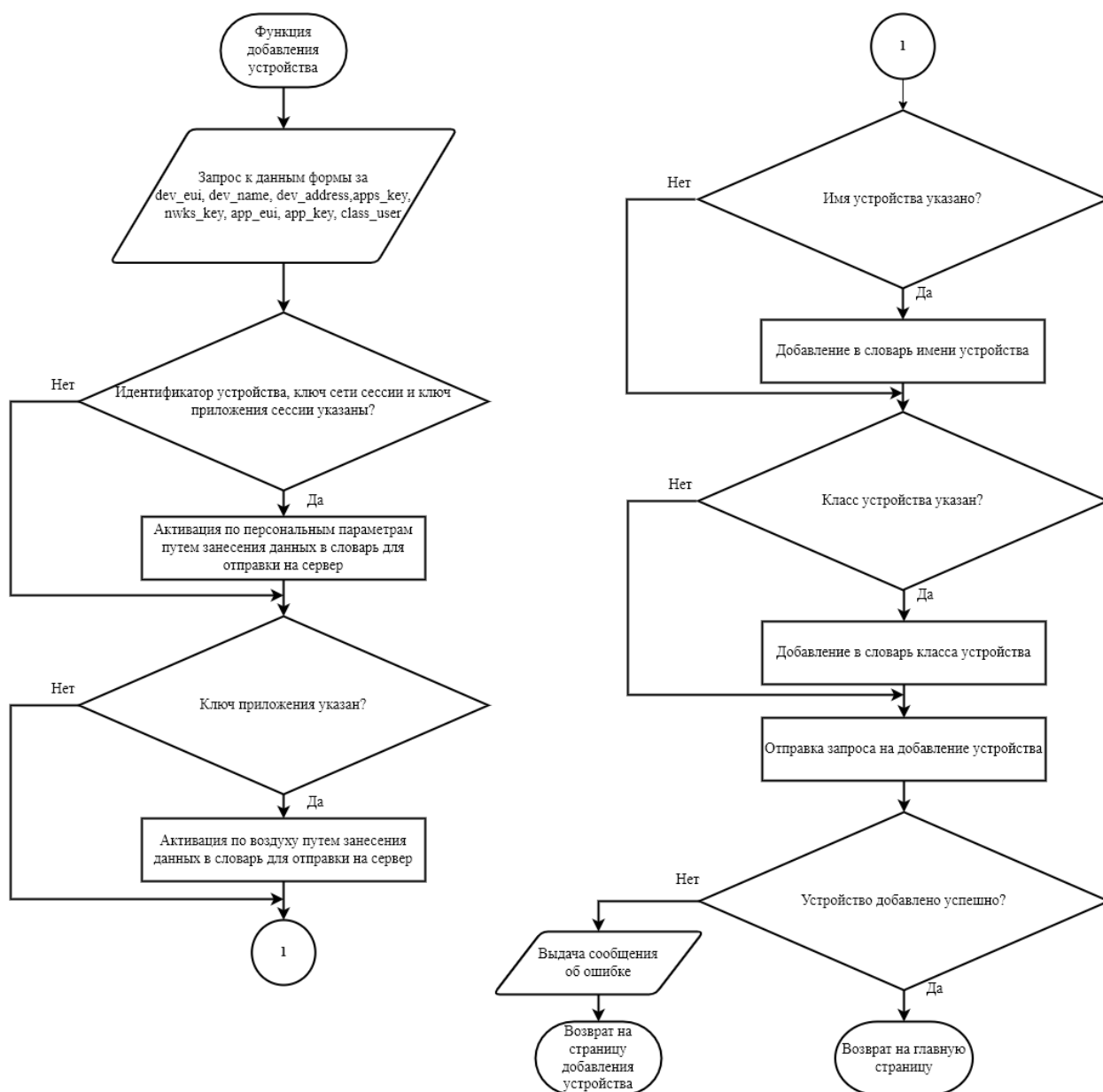


Рисунок 2.25 – Блок-схема функции добавления устройства

Функция, которая отвечает за добавление устройства, представлена на рисунках 2.26– 2.27. В начале производится проверка какой метод пришел на веб-приложение. Затем происходит проверка валидации формы при отправке post метода и получение данных в переменные. После чего происходит установка значений для запроса, который впоследствии отправляется и проверяется.

```

@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 2.26 – 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 2.27 – Добавление устройства. Часть 2

На рисунке 2.28 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 2.28 – Добавление устройства

Для проверки того, что добавление работает верно можно снова обратиться к приложению администрирования от компании Vega Iot Vega Admin Tool (рисунок 2.29).

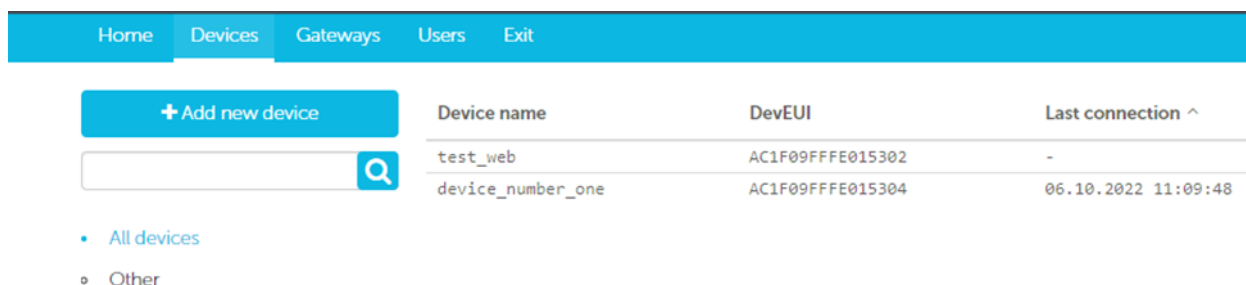


Рисунок 2.29 – Добавленное устройство в Iot Vega Admin Tool

Блок-схема функции, которая отвечает за удаление устройства представлена на рисунке 2.30.

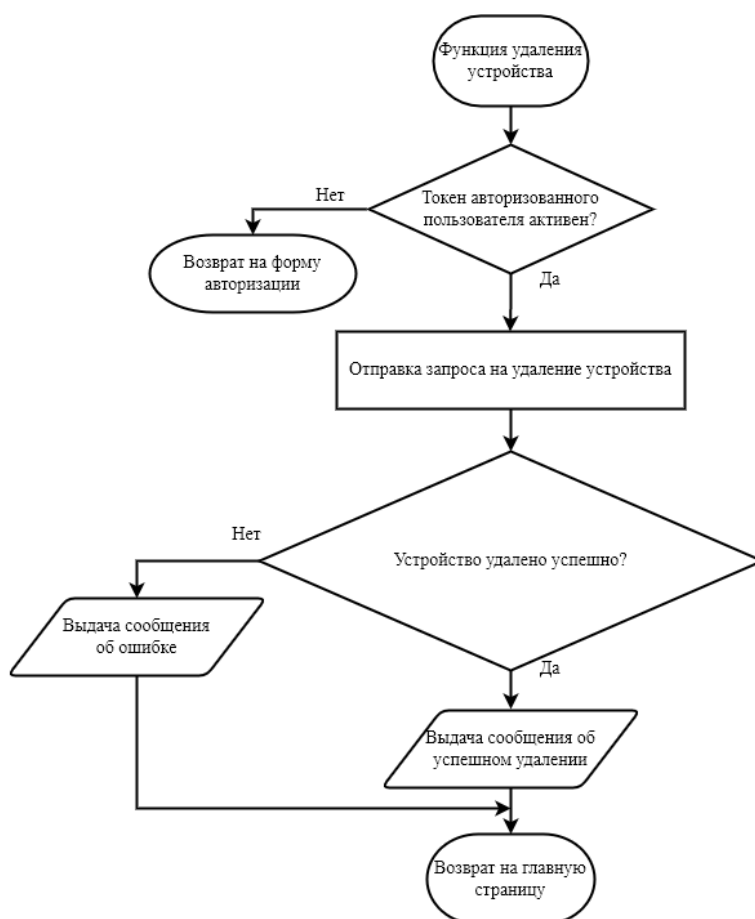


Рисунок 2.30 – Блок-схема удаления устройства

Функция, который отвечает за удаление устройства представлен на рисунке 2.31. 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 2.31 – Удаление устройства

Для того, чтобы удалить устройство в списке подключенных устройств нужно нажать на кнопку “Удалить”, расположенную рядом с идентификатором устройства (рисунок 2.32).

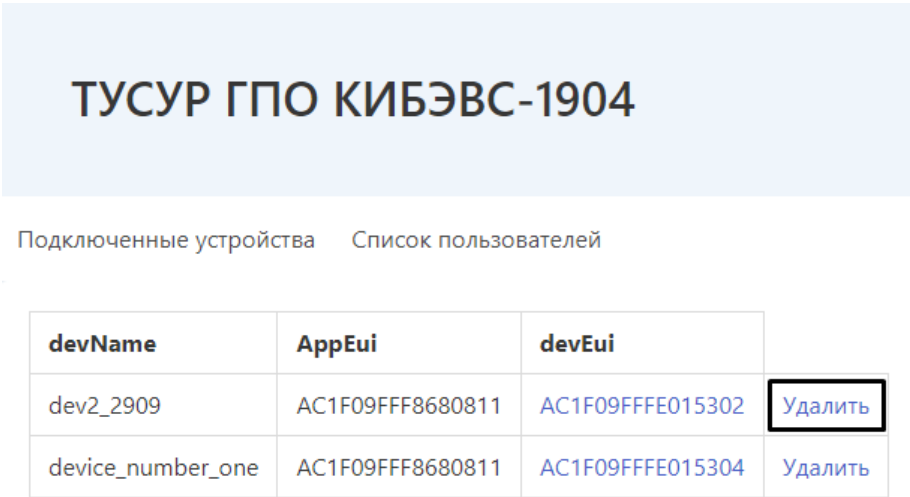


Рисунок 2.32 – Кнопка “Удалить”

После чего будет выведено сообщение об успешном удалении (рисунок 2.33).

ТУСУР ГПО КИБЭВС-1904

Подключенные устройства Список пользователей

Устройство было успешно удалено

devName	AppEui	devEui	
device_number_one	AC1F09FFF8680811	AC1F09FFFE015304	Удалить

Рисунок 2.33 – Сообщение об успешном удалении

Для того, чтобы проверить, что удаление сработало корректно можно посмотреть список устройств в IoT Vega Admin Tool (рисунок 2.34).

Home

Devices

Gateways

Users

Exit

+ Add new device

Q

All devices

Other

Device name	DevEUI	Last connection ^
device_number_one	AC1F09FFFE015304	06.10.2022 11:09:48

Рисунок 2.34 – Список устройств после удаления в Iot Vega Admin Tool

Как видно удаление устройства работает корректно.

Блок-схема функции, которая отвечает за добавление пользователей представлена на рисунке 2.35.

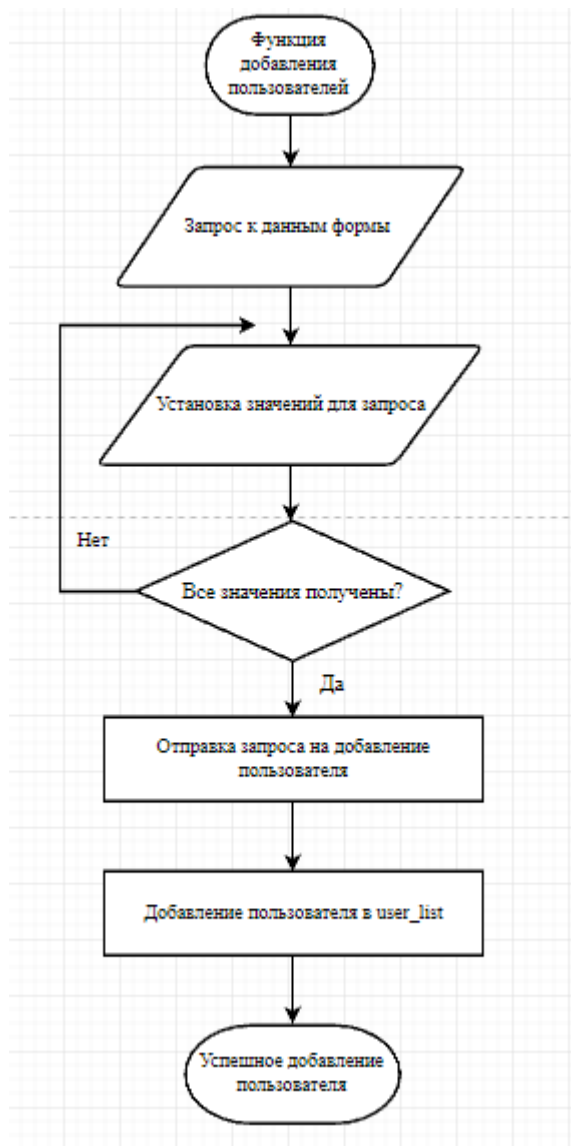


Рисунок 2.35 – Блок-схема функции добавления пользователей

На рисунках 2.36 – 2.37 представлена реализованная функция, которая отвечает за добавление учетной записи пользователя. 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 2.36 – 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 2.37 – Добавление учетной записи пользователя. Часть 2

Страница добавления новой учетной записи пользователя представлена на рисунке 2.38.

Логин:

Пароль:

Доступ к устройству:
Установка доступа к устройству
 ▼

Работа с консолью
Установка возможности подключения к консоли для получения информации отладки
☒

DevEui список
Список devEui которые будут доступны для учетной записи

AC1F09FFFE015304

AC1F09FFFE015302

Список команд
Список команд которые будут доступны для учетной записи

get_users

manage_users

delete_users

get_device_appdata

Unsolicited
Возможность отправки сообщений онлайн
☒

Направление
Возможное направление онлайн сообщений
 ▼

С MAC командами
Онлайн сообщения, которые содержат MAC команды
☒

Рисунок 2.38 – 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 пользователь был добавлен (рисунок 2.39). Для проверки корректности работы нужно обратиться к IoT Vega Admin Tool (рисунок 2.40).

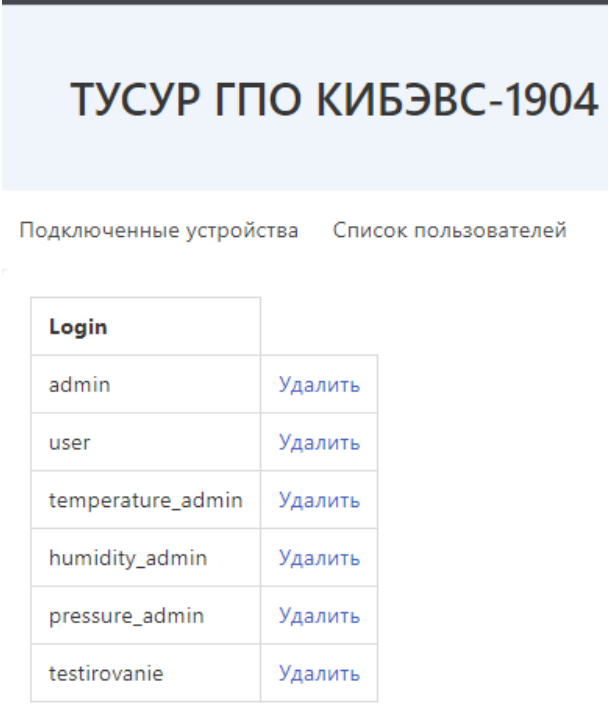


Рисунок 2.39 – Добавление учетной записи пользователя

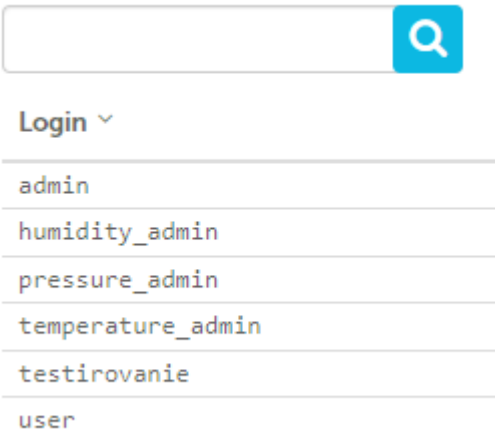


Рисунок 2.40 – Список учетных записей в IoT Vega Admin Tool

Блок-схема функции, которая отвечает за удаление пользователей представлена на рисунке 2.41.



Рисунок 2.41 – Блок-схема функции удаления пользователей

На рисунке 2.42 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 2.42 – Удаление учетной записи пользователя

Для того, чтобы удалить учетную запись пользователя нужно также нажать на кнопку “Удалить” в списке пользователей (рисунок 2.43).

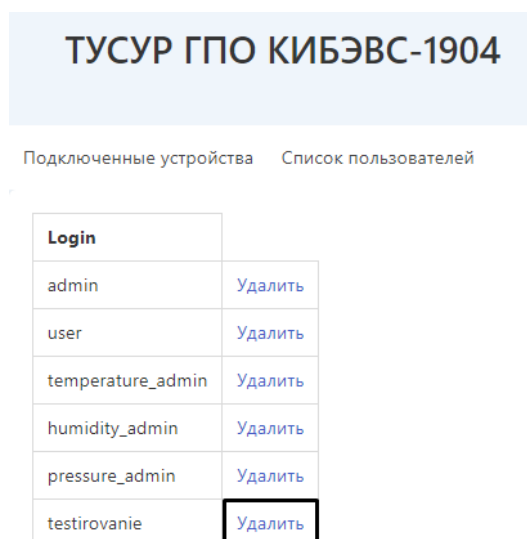


Рисунок 2.43 – Кнопка “Удалить”

Для того, чтобы проверить, что удаление сработало корректно нужно обратиться к IoT Vega Admin Tool (рисунок 2.44).

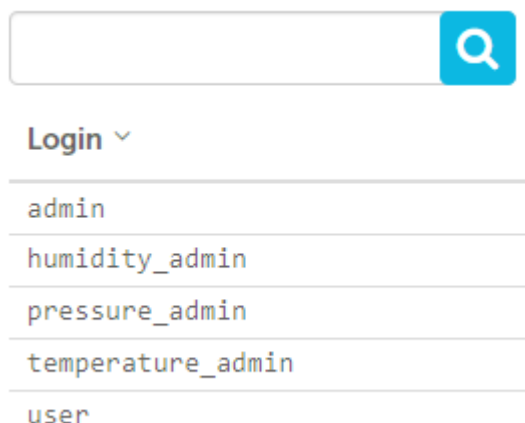


Рисунок 2.44 – Список пользователей после удаления в Iot Vega Admin Tool

Удаление пользователей работает тоже корректно.

Блок-схема функции, которая отвечает за построение графика по полученным значениям, представлена на рисунке 2.45.

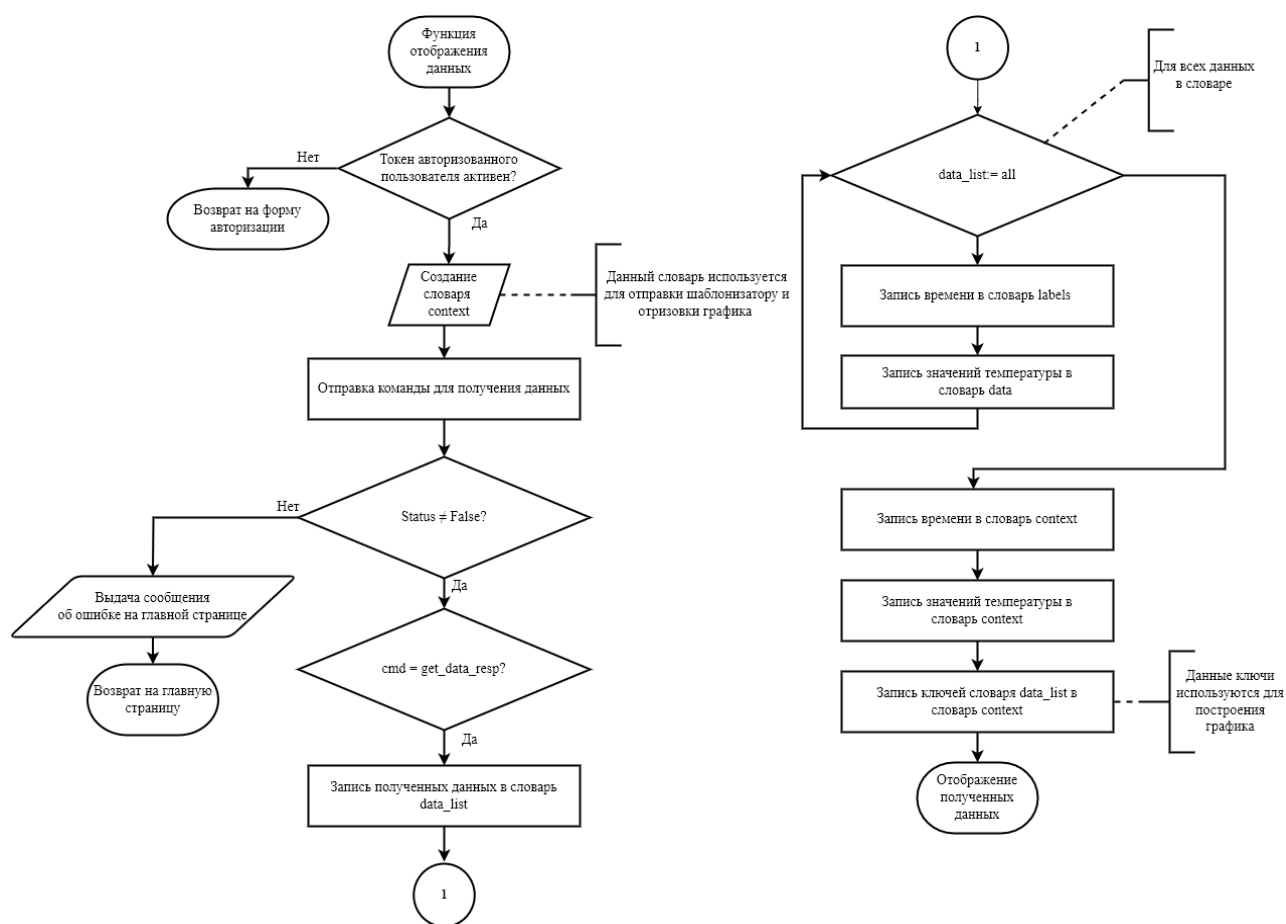


Рисунок 2.45 – Блок-схема функции отображения данных

Функция, которая отвечает за построение графика по полученным значениям, представлена на рисунке 2.46. 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 2.46 – Построение графика

Для того, чтобы перейти на страницу с графиком необходимо нажать на идентификатор устройства в списке устройств, график представлен на рисунке 2.47. Для отрисовки графиков была использована JavaScript библиотека визуализации данных ChartJS. [15]

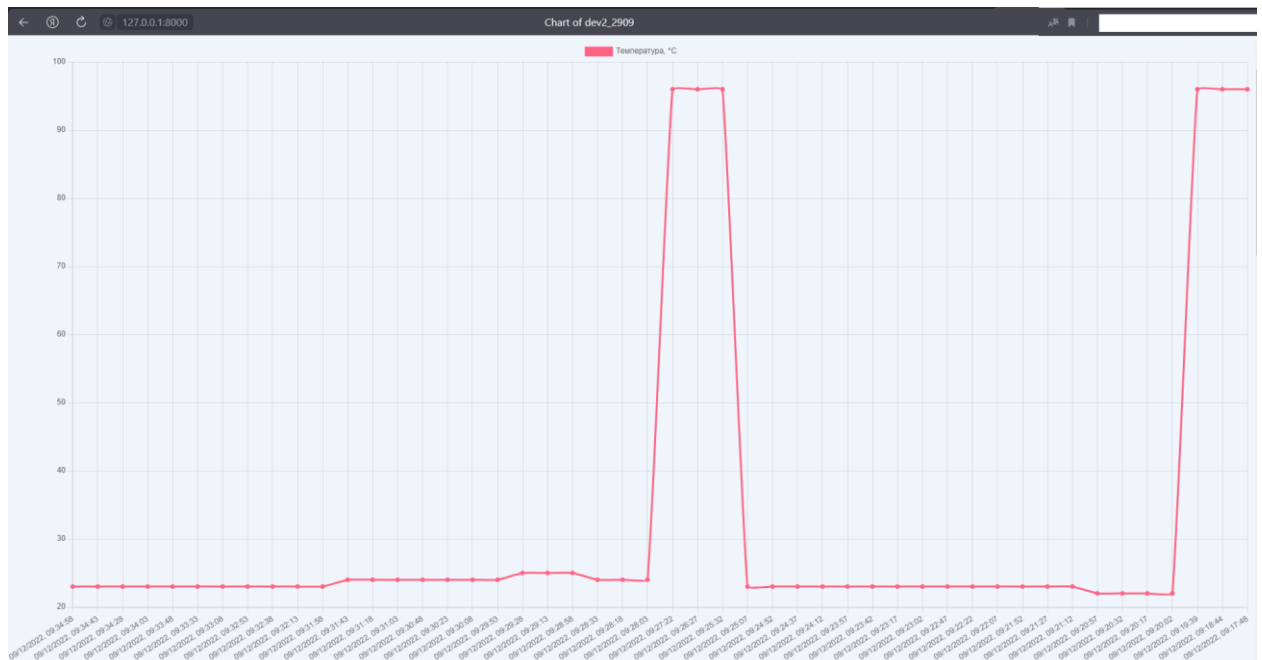


Рисунок 2.47 – График полученных данных

По вертикали на представленном графике температура в °C, по горизонтали время получения данных. Под графиком на данной странице представлен список полученных пакетов (рисунок 2.48).

ack	data	dlr	fcnt	freq	gatewayid	port	rssi	snr	ts	type
0	23	SF7 BW125 4/5	2	867300000	0000000000000001	1	-73	9.5	09/12/2022, 09:34:58	UNCONF_UP
0	23	SF7 BW125 4/5	1	868500000	0000000000000001	1	-77	9	09/12/2022, 09:34:43	UNCONF_UP
0	23	SF7 BW125 4/5	0	867500000	0000000000000001	1	-77	7.5	09/12/2022, 09:34:28	UNCONF_UP
0	23	SF7 BW125 4/5	2	868100000	0000000000000001	1	-76	9.2	09/12/2022, 09:34:03	UNCONF_UP
0	23	SF7 BW125 4/5	1	867700000	0000000000000001	1	-75	8.5	09/12/2022, 09:33:48	UNCONF_UP
0	23	SF7 BW125 4/5	0	868500000	0000000000000001	1	-75	7.8	09/12/2022, 09:33:33	UNCONF_UP
0	23	SF7 BW125 4/5	2	867100000	0000000000000001	1	-72	8.5	09/12/2022, 09:33:08	UNCONF_UP
0	23	SF7 BW125 4/5	1	868500000	0000000000000001	1	-69	9.8	09/12/2022, 09:32:53	UNCONF_UP
0	23	SF7 BW125 4/5	0	867500000	0000000000000001	1	-70	7.5	09/12/2022, 09:32:38	UNCONF_UP
0	23	SF7 BW125 4/5	2	868100000	0000000000000001	1	-70	9.2	09/12/2022, 09:32:13	UNCONF_UP
0	23	SF7 BW125 4/5	1	867900000	0000000000000001	1	-75	10	09/12/2022, 09:31:58	UNCONF_UP
0	24	SF7 BW125 4/5	0	868300000	0000000000000001	1	-79	7	09/12/2022, 09:31:43	UNCONF_UP
0	24	SF7 BW125 4/5	2	867100000	0000000000000001	1	-73	8.5	09/12/2022, 09:31:18	UNCONF_UP
0	24	SF7 BW125 4/5	1	867100000	0000000000000001	1	-75	9.2	09/12/2022, 09:31:03	UNCONF_UP
0	24	SF7 BW125 4/5	0	867500000	0000000000000001	1	-71	7.2	09/12/2022, 09:30:48	UNCONF_UP
0	24	SF7 BW125 4/5	2	868100000	0000000000000001	1	-75	8.8	09/12/2022, 09:30:23	UNCONF_UP
0	24	SF7 BW125 4/5	1	867900000	0000000000000001	1	-76	9	09/12/2022, 09:30:08	UNCONF_UP
0	24	SF7 BW125 4/5	0	868500000	0000000000000001	1	-76	8.5	09/12/2022, 09:29:53	UNCONF_UP
0	25	SF7 BW125 4/5	2	867300000	0000000000000001	1	-75	9.8	09/12/2022, 09:29:28	UNCONF_UP
0	25	SF7 BW125 4/5	1	868500000	0000000000000001	1	-77	8.5	09/12/2022, 09:29:13	UNCONF_UP
0	25	SF7 BW125 4/5	0	867500000	0000000000000001	1	-81	7.8	09/12/2022, 09:28:58	UNCONF_UP
0	24	SF7 BW125 4/5	3	867900000	0000000000000001	1	-81	9.8	09/12/2022, 09:28:33	UNCONF_UP

Рисунок 2.48 – Список полученных пакетов

В адресной строке (рисунок 2.49) можно регулировать по скольким значениям будет построен график (тот самый лимит).

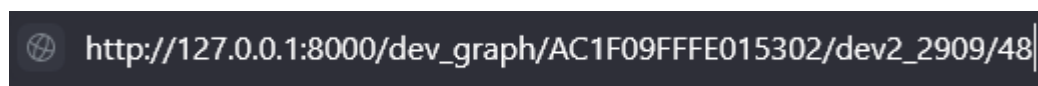


Рисунок 2.49 – Адресная строка

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.

Для проверки верности отображенных полученных данных можно обратиться к приложению администрирования от компании Vega Iot Vega Admin Tool (рисунок 2.50).





				
Date	Type	Data		
09.12.2022 16:34:58	UNCONF_UP	23		
09.12.2022 16:34:43	UNCONF_UP	23		
09.12.2022 16:34:28	UNCONF_UP	23		
09.12.2022 16:34:18	JOIN_ACC	2019a0aa3d9c833438b025fdf57395687e016431917c002a33e71b45690b1b94d7		
09.12.2022 16:34:18	JOIN_REQ			
09.12.2022 16:34:3	UNCONF_UP	23		
09.12.2022 16:33:48	UNCONF_UP	23		
09.12.2022 16:33:33	UNCONF_UP	23		
09.12.2022 16:33:23	JOIN_ACC	20979ef4815c916ef3c33c9a0f4c00906f730219aa159d236118c6d65bc29d2dcc		
09.12.2022 16:33:23	JOIN_REQ			
09.12.2022 16:33:8	UNCONF_UP	23		
09.12.2022 16:32:53	UNCONF_UP	23		
09.12.2022 16:32:38	UNCONF_UP	23		
09.12.2022 16:32:28	JOIN_ACC	2083e23190b3fbd7eb9e20c2657cd100cfe625d26dfd5749a54bb41783fd1325		
09.12.2022 16:32:28	JOIN_REQ			
09.12.2022 16:32:13	UNCONF_UP	23		
09.12.2022 16:31:58	UNCONF_UP	23		
09.12.2022 16:31:43	UNCONF_UP	24		
09.12.2022 16:31:33	JOIN_ACC	203313e2fe0acd913e19bf218f6c8c2f81d1f0d3ac063cf0b091409b830a4d74ff		
09.12.2022 16:31:33	JOIN_REQ			
09.12.2022 16:31:18	UNCONF_UP	24		
09.12.2022 16:31:3	UNCONF_UP	24		
09.12.2022 16:30:48	UNCONF_UP	24		
09.12.2022 16:30:38	JOIN_ACC	20a50abbd78afcef66e47212f6ad8aa5c556da11ac497bb1b2d583f47cfe0fbcd		
09.12.2022 16:30:38	JOIN_REQ			
09.12.2022 16:30:23	UNCONF_UP	24		
09.12.2022 16:30:8	UNCONF_UP	24		
09.12.2022 16:29:53	UNCONF_UP	24		
09.12.2022 16:29:43	JOIN_ACC	205751b541bf36e2e8fd44f4c2478e8cb2ac6c10254cf19c96b924e07fee0687b3		
09.12.2022 16:29:43	JOIN_REQ			

Рисунок 2.50 – Полученные пакеты в IoT Vega Admin Tool

Если судить по последним пакетам можно сделать вывод о том, что все работает корректно.

Полный листинг кода веб-приложения представлен в приложениях А – Г.

2.7 Тестирование WEB-приложения

Тестирование приложения на данном этапе разработки проводилось по двум параметрам: функциональному и на удобство использования.

Целью функционального тестирования является проверка работоспособности и корректности работы реализованных функций.

Функциональное тестирование выполнялось по следующим пунктам:

- Пользователь существует в системе с введенным логином и паролем.
- Пользователь с введенным логином не существует в системе;
- Пользователь с введенным логином существует в системе, но пароль неверный.

- Возможность добавления устройств.
- Возможность удаления устройств.
- Возможность добавления пользователей.
- Возможность удаления пользователей.

При осуществлении входа в учётную запись существующего пользователя, выдавало ошибку, представленную на рисунке 2.51.

TypeError

TypeError: string indices must be integers

```
Traceback (most recent call last)
File "C:\GPO\web\venv\lib\site-packages\flask\app.py", line 2548, in __call__
    return self.wsgi_app(environ, start_response)
File "C:\GPO\web\venv\lib\site-packages\flask\app.py", line 2628, in wsgi_app
    response = self.handle_exception(e)
File "C:\GPO\web\venv\lib\site-packages\flask\app.py", line 2625, in wsgi_app
    response = self.full_dispatch_request()
File "C:\GPO\web\venv\lib\site-packages\flask\app.py", line 1822, in full_dispatch_request
    rv = self.handle_user_exception(e)
File "C:\GPO\web\venv\lib\site-packages\flask\app.py", line 1820, in full_dispatch_request
    rv = self.dispatch_request()
File "C:\GPO\web\venv\lib\site-packages\flask\app.py", line 1796, in dispatch_request
    return self.ensure_sync(self.view_functions[rule.endpoint])(**view_args)
File "C:\GPO\web\venv\lib\site-packages\flask\app.py", line 110, in index
    context["devices_list"] = devalistresp["devices_list"]
TypeError: string indices must be integers
```

Рисунок 2.51 – Ошибка авторизации

Данная ошибка была исправлена, и авторизация выполняется корректно.

Результат выполнения авторизации представлен на рисунках 2.52-2.53.

ТУСУР ГПО КИБЭВС-1904

Вход в учетную запись

Логин:

root

Пароль:

.....

Войти

Рисунок 2.52 – Окно авторизации

ТУСУР ГПО КИБЭВС-1904

Подключенные устройства

Список пользователей

Создать учетную запись

Добавить устройство

Выход

devName	AppEui	devEui	
device_number_one	AC1F09FFF8680811	AC1F09FFFE015304	Удалить

Рисунок 2.3 – Успешный вход

При успешном вводе данных существующего пользователя авторизация выполняется корректно.

Далее в ходе выполнения функционального тестирования были введены данные несуществующего пользователя. Результатом попытки авторизации является вывод строки-сообщения о не корректности введенных данных (Рисунок 2.54).

ТУСУР ГПО КИБЭВС-1904

Неверный логин/пароль

Логин:

user5

Пароль:

Войти

Рисунок 2.54 – Уведомление о некорректных данных

Затем была выполнена проверка авторизации при вводе логина существующего пользователя и неправильного пароля, результатом которой является аналогичное уведомление о некорректности введенных данных.

Также в ходе тестирования была обнаружена ошибка, заключающаяся в том, что при обновлении страницы, сообщение о неправильных введенных данных или о том, что был произведен выход из учётной записи, не пропадало. Однако эта ошибка была исправлена и при обновлении страницы уведомление работает корректно (Рисунок 2.55-2.56).

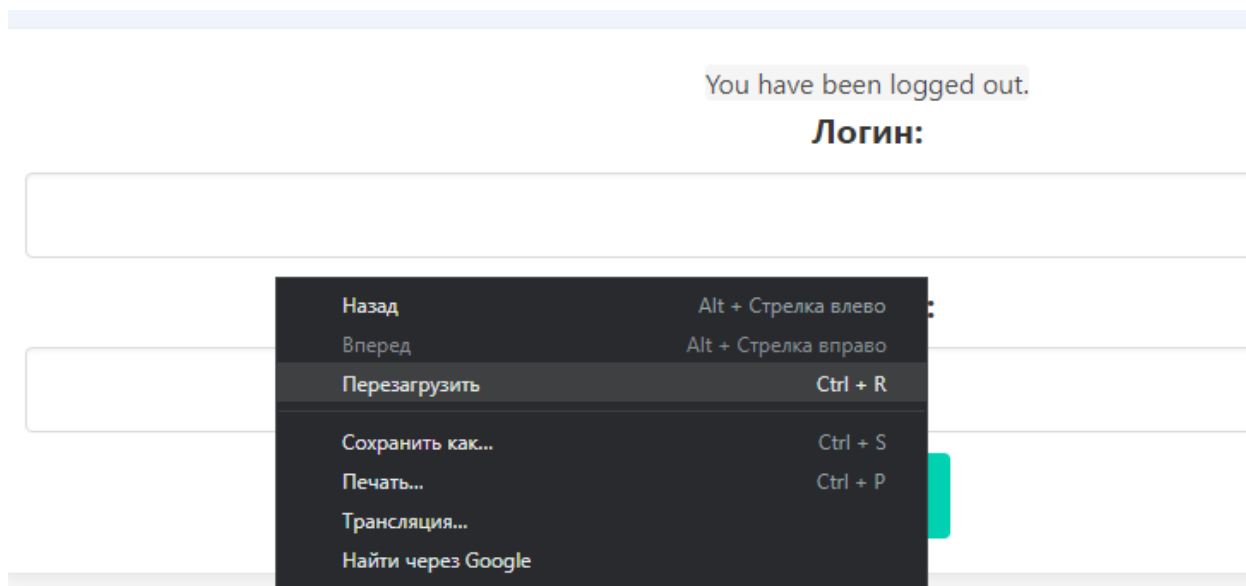


Рисунок 2.55 – Обновление страницы с уведомлением

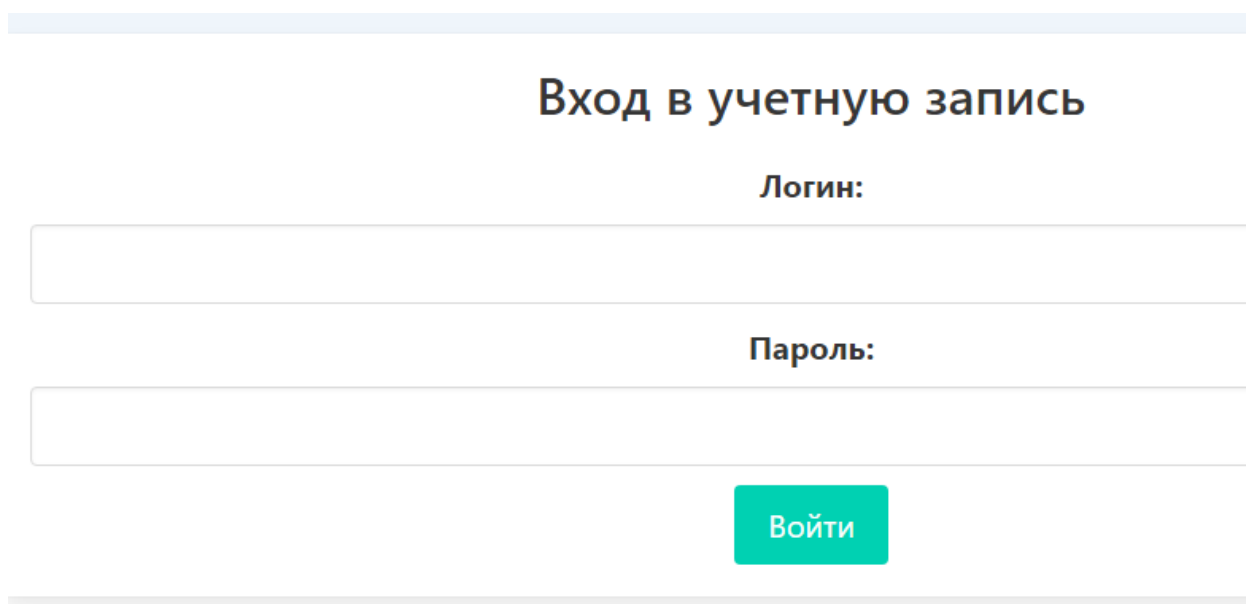


Рисунок 2.56 – Отсутствие уведомления после обновления страницы

Затем в ходе тестирования были выполнены проверки на добавление и удаление устройств. Результаты тестирования представлены на рисунках 2.57 – 2.59.

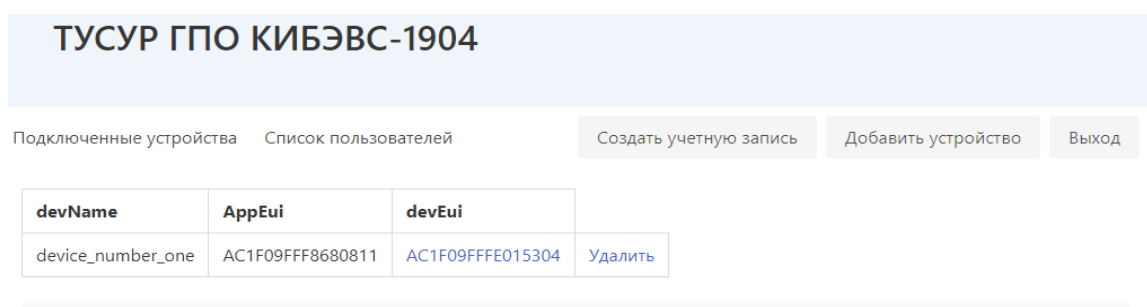


Рисунок 2.57 – Изначальный список устройств

ТУСУР ГПО КИБЭВС-1904

Подключенные устройства Список пользователей

Создать учетную запись

Добавить устройство

Выход

devName	AppEui	devEui	
Test add device	AC1F09FFF8501043	AC1F09FFFE015302	Удалить
device_number_one	AC1F09FFF8680811	AC1F09FFFE015304	Удалить
Test add device2	AC1F09FFF8501043	AC1F09FFFE015306	Удалить

Рисунок 2.58 – Список с добавленными устройствами

ТУСУР ГПО КИБЭВС-1904

Подключенные устройства Список пользователей

Создать учетную запись

Добавить устройство

Выход

Устройство было успешно удалено

devName	AppEui	devEui	
device_number_one	AC1F09FFF8680811	AC1F09FFFE015304	Удалить
Test add device2	AC1F09FFF8501043	AC1F09FFFE015306	Удалить

Рисунок 2.59 – Удаление устройства

Также помимо удаления и добавления новых устройств была протестирована функция добавления и удаления учётных записей пользователей. Результаты тестирования представлены на рисунках 2.60 – 2.63.

Подключенные устройства Список пользователей

Создать учетную запись

Добавить устройство

Выход

Login	
admin	Удалить
user	Удалить
temperature_admin	Удалить
pressure_admin	Удалить

Рисунок 2.60 – Список пользователей перед добавлением нового

ТУСУР ГПО КИБЭВС-1904

Логин:

user_test

Пароль:

.....

Доступ к устройству:

Установка доступа к устройству

FULL



Рисунок 2.61 – Добавление нового пользователя

Подключенные устройства

Список пользователей

Login	
admin	Удалить
user	Удалить
temperature_admin	Удалить
pressure_admin	Удалить
user_test	Удалить

Рисунок 2.62 – Список пользователей после добавления нового

Login	
admin	Удалить
temperature_admin	Удалить
pressure_admin	Удалить
user_test	Удалить

Рисунок 2.63 – Список пользователей после удаления

Тестирование на удобство использования предполагает проверку навигации и контента, с которым будет взаимодействовать пользователь во время работы с web-приложением.

В рамках тестирования удобства использования было проведено тестирование по следующим пунктам.

- Проверка наличия подсказок в полях логина и пароля.
- Отсутствие орфографических и грамматических ошибок.
- Корректность переходов и навигации между страницами.

В процессе тестирования на отображение подсказок в полях никаких ошибок выявлено не было.

Результаты тестирования на отображение подсказок представлены на рисунках 2.64-2.65.

Вход в учетную запись

Логин:

Заполните это поле.

Войти

Рисунок 2.64 – Подсказка о необходимости ввода логина

Вход в учетную запись

Логин:

Пароль:

Заполните это поле.

Рисунок 2.65 – Подсказка о необходимости ввода пароля

В результате проведенного тестирования удобства использования был выявлен некорректный выход из профиля пользователя, приводящий к ошибке, которая впоследствии была исправлена, другие переходы между вкладками на страницах работают и отображаются корректно.

Заключение

В ходе выполнения проекта «Исследование и создание IoT-сети, основанной на технологиях модуляции LoRa и сетевого протокола LoRaWAN»:

- Было реализовано питание конечного устройства от внешнего аккумулятора и был решен вопрос с приобретением подходящего корпуса. В совокупности описанные решения позволят в дальнейшем разместить устройства в учебных аудиториях.
- Были изучены типовые угрозы на сеть LoRaWAN, обеспечение безопасности в самом протоколе, а также рассмотрены некоторые атаки на шлюз, конечные устройства и канал передачи данных.
- Была изучена документация на проект;
- Была разработана UML-диаграмма, демонстрирующая структуру и логику работы веб-приложения;
- Были определены язык программирования и инструменты, необходимы для реализации веб-приложения;
- Было реализовано соединение приложения с сервером при использовании API-функций сервера Вега;
- Была реализована авторизация в приложении при использовании API-функций сервера Вега;
- Было реализовано отображение подключенных устройств и зарегистрированных пользователей в приложении;
- Было реализовано добавление устройств в приложении при использовании API-функций сервера Вега;
- Было реализовано удаление устройств в приложении при использовании API-функций сервера Вега;
- Было реализовано добавления учетных записей пользователей при использовании API-функций сервера Вега;

- Было реализовано удаление учетных записей пользователей при использовании API-функций сервера Вега;
- Было реализовано отображения показаний датчиков в приложении в виде графика и таблицы;
- Было проведено функциональное тестирование, а также тестирование на удобство использования web-приложения, в ходе которых была проверена работоспособность заявленных функций, а также корректность работы и удобство использования приложения.

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

Список использованных источников

1. Internet of Things [IoT] Market Size, Share & Trends, 2029. [Электронный ресурс]: Интернет вещей. URL: [https://www-fortunebusinessinsights-com.translate.goog/industry-reports/internet-of-things-iot-market-100307?_x_tr_sl=auto&_x_tr_tl=ru&_x_tr_hl=ru](https://www.fortunebusinessinsights-com.translate.goog/industry-reports/internet-of-things-iot-market-100307?_x_tr_sl=auto&_x_tr_tl=ru&_x_tr_hl=ru) (дата обращения 15.12.2022).
2. LoRaWAN Message Types. [Электронный ресурс]: Message Types. URL: <https://www.thethingsnetwork.org/docs/lorawan/message-types> (дата обращения 10.12.2022).
3. LoRaWAN Security. [Электронный ресурс]: Security Keys. URL: <https://www.thethingsnetwork.org/docs/lorawan/security> (дата обращения 10.12.2022).
4. Обеспечение безопасности в беспроводных протоколах на примере LoRaWAN [Электронный ресурс]: Обеспечение безопасности. URL: <https://habr.com/ru/post/458394> (дата обращения 10.12.2022).
5. Сеть LoRaWAN: безопасность обеспечивается [Электронный ресурс]: Сеть LoRaWAN. URL: <https://www.iksmedia.ru/articles/5573226-Set-LoRaWAN-bezopasnost-obespechiva.html> (дата обращения 10.12.2022).
6. RM0008 Reference manual. RM0008 STM32F101xx, STM32F102xx, STM32F103xx, STM32F105xx and STM32F107xx advanced Arm®-based 32-bit MCUs [Электронный ресурс]: st.com URL: <https://www.st.com/en/microcontrollersmicroprocessors/stm32f103rb.html#documentation> (дата обращения 10.09.2022).
7. Питание схемы [Электронный ресурс]: alexgyver.ru. URL: <https://alexgyver.ru/lessons/arduino-power/> (дата обращения 20.11.2022).
8. Анализ уязвимостей в энергоэффективных сетях дальнего радиуса действия на примере LoRaWAN [Электронный ресурс]: <https://earchive.tpu.ru>. URL: https://earchive.tpu.ru/bitstream/11683/46208/1/conference_tpu-2017-C24_V1_p122-126.pdf (дата обращения 21.11.2022).

9. Экспериментальное моделирование передачи данных в беспроводных сетях интернета вещей. Протокол LoRa [Электронный ресурс]: <https://www.elibrary.ru>. URL: <https://elibrary.ru/item.asp?id=45772874> (дата обращения 21.11.2022).

10. Анализ энергозатрат при реализации защиты от атаки "незаконного изменения" в канале между сетевым сервером и сервером приложения в сети LoRaWAN [Электронный ресурс]: <https://www.elibrary.ru>. URL: <https://elibrary.ru/item.asp?id=41146794>

11. Разработка защищенного узла сенсорной сети на базе технологии LoRa [Электронный ресурс]: <https://www.elibrary.ru>. URL: <https://elibrary.ru/item.asp?id=46322726>

12. API VEGA-lora rev21 [Электронный ресурс]: API VEGA. URL: <https://iotvega.com/content/ru/soft/server/API%20VEGA-lora%20rev21.pdf> (дата обращения 22.09.2022).

13. Среда разработки PyCharm [Электронный ресурс]: JetBrains.com. URL: <https://www.jetbrains.com/ru-ru/pycharm> (дата обращения 20.10.2022).

14. Фреймворк Flask [Электронный ресурс]: Flask. URL: <https://flask.palletsprojects.com/en/2.2.x> (дата обращения 20.10.2022).

15. ChartJS – JavaScript-библиотека визуализации данных [Электронный ресурс]: ChartJS. URL: <https://habr.com/ru/company/developersoft/blog/185210/> (дата обращения 25.11.2022).

Приложение А

(Обязательное)

Файл run.py

```
from vega import app
from vega import routes

if __name__ == '__main__':
    app.run(host='127.0.0.1', port=8000, debug=True)
```

Приложение Б

(Обязательное)

Файл routes.py

```
from flask import render_template, request, redirect, url_for,
flash, session, make_response
import json
import websocket
from websocket import create_connection
from vega import app
from datetime import datetime, timezone
import tzlocal
from .forms import LoginForm, CreateUserForm, AddDeviceForm

URL_WS = "ws://localhost:8002/"

success_result_add_device_list = [
    "added",
    "updated",
    "nothingToUpdate",
    "updateViaMacBuffer"
]

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

@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'))

@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'))

@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)

@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'))
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)

@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,
        "mwksKey": nwks_key
    }
    set_data["ABP"] = abp
if app_key != "":
    ottaa = {
        "appKey": app_key,
    }
    if app_eui != "":
        ottaa["appEui"] = app_eui
    set_data["OTAA"] = ottaa

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,
    "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)

@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)

@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'))

@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)

```

Приложение В

(Обязательное)

Файл forms.py

```
import wtforms
from flask_wtf import FlaskForm
from wtforms.validators import Length, InputRequired, NumberRange,
ValidationError

all_command_list = [
    "get_users",
    "manage_users",
    "delete_users",
    "get_device_appdata",
    "get_data",
    "send_data",
    "manage_device_appdata",
    "delete_device_appdata",
    "get_gateways",
    "manage_gateways",
    "delete_gateways",
    "get_devices",
    "manage_devices",
    "delete_devices",
    "get_coverage_map",
    "get_device_downlink_queue",
    "manage_device_downlink_queue",
    "server_info",
    "send_email",
    "tx"
]

class LoginForm(FlaskForm):
    login = wtforms.StringField("Логин: ",
    validators=[InputRequired()])
    password = wtforms.PasswordField("Пароль: ",
    validators=[InputRequired(), Length(min=1, max=100)])
    # remember = BooleanField("Запомнить", default=False)
    submit = wtforms.SubmitField("Войти")

class CreateUserForm(FlaskForm):
    login = wtforms.StringField("Логин: ",
    validators=[InputRequired()])
    password = wtforms.PasswordField("Пароль: ", validators=[])
    device_access = wtforms.SelectField("Доступ к устройству: ",
    choices=["FULL", "SELECTED"], default="SELECTED")
    console_enable = wtforms.BooleanField("Работа с консолью")
    devEui_list = wtforms.SelectMultipleField("DevEui список")
    command_list = wtforms.SelectMultipleField("Список команд",
    choices=all_command_list)
```

```

    unsolicited = wtforms.BooleanField("Unsolicited")
    direction = wtforms.SelectField("Направление",
    choices=["UPLINK", "DOWNLINK", "ALL"])
    with_MAC_Commands = wtforms.BooleanField("С MAC командами")
    submit = wtforms.SubmitField("Внести")

def is_string_is_hex(form, field):
    for ch in field.data:
        if ((ch < '0' or ch > '9') and
            (ch < 'A' or ch > 'F')):
            raise ValidationError('Field must be in HEX')

class AddDeviceForm(FlaskForm):
    dev_eui = wtforms.StringField("EUI устройства:",
                                validators=[InputRequired(),
                                Length(16, 16,
                                message="Must be 16 length")])
    dev_name = wtforms.StringField("Имя устройства: ")
    dev_address = wtforms.IntegerField(
        "Адрес устройства: ",
        validators=[NumberRange(min=0x00000001, max=0xFFFFFFFF,
                                message="0x00000001 and 0xFFFFFFFF desired")])
    apps_key = wtforms.StringField("Application session key: ",
                                validators=[Length(32, 32,
                                message="Must be 32 length"), is_string_is_hex], default="")
    nwks_key = wtforms.StringField("Network session key: ",
                                validators=[Length(32, 32,
                                message="Must be 32 length"), is_string_is_hex],
                                default="")
    app_eui = wtforms.StringField("Application EUI: ",
                                validators=[Length(16, 16,
                                message="Must be 16 length"), is_string_is_hex],
                                default="")
    app_key = wtforms.StringField("Application key: ",
                                validators=[Length(32, 32,
                                message="Must be 32 length"), is_string_is_hex],
                                default="")
    class_user = wtforms.SelectField("Класс пользователя",
    choices=['CLASS_A', 'CLASS_C'])
    submit = wtforms.SubmitField("Добавить")

```


Приложение Г
(Обязательное)
Файл init.py

```
from flask import Flask  
import os
```

```
app = Flask(__name__, static_folder="static")  
basedir = os.path.abspath(os.path.dirname(__file__))  
app.config['SECRET_KEY'] = 'hard to guess'
```