МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации

*Изображение выглядит как текст, Шрифт, снимок экрана, логотип

Автоматически созданное описание*

**РАСЧЕТНО-ГРАФИЧЕСКОЕ ЗАДАНИЕ**

**по дисциплине: «*Технологии и методы программирования*»**

**Тема: «*Аудит конфигурации реле защиты*»**

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБс-324», «АВТФ»  *Петров Максим Игоревич*  «23» мая 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  *Старший преподаватель кафедры ЗИ*  *Медведев Михаил Александрович*  «\_\_\_» \_\_\_\_\_\_ 2025г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Новосибирск 2025

СОДЕРЖАНИЕ

[Введение 3](#_Toc198760586)

[Макет базы данных 4](#_Toc198760587)

[Архитектура веб-сервера 5](#_Toc198760588)

[Создание бэкенда 5](#_Toc198760589)

[Создание фронтенда 8](#_Toc198760590)

[Документация 11](#_Toc198760591)

[Бэкенд 11](#_Toc198760592)

[Безопасность в серверной части 17](#_Toc198760593)

[Фронтенд 19](#_Toc198760594)

[Безопасность в клиентской части 28](#_Toc198760595)

[Тестирование 30](#_Toc198760596)

[Заключение 36](#_Toc198760597)

[Приложение 37](#_Toc198760598)

# Введение

В данной работе мы реализуем проект по предметной области «Безопасность в энергетике». Тема проекта – «Аудит конфигурации реле защиты», который представляет из себя проверку соответствия настроек реле заданным регламентам, отчёт о найденных отклонениях. На наш веб-сервер мы будем загружать конфигурационные файлы, а на выходе получать сгенерированный отчет о нарушениях в нем.

Проект будет полезен для инженеров энергетики, которые обслуживают подстанции и объекты энергетики.

Основные функции системы:

- Добавление конфигураций реле;

- Выполнение аудита конфигурации;

- Генерация отчётов;

- Работа с учётными записями пользователей (аутентификация и авторизация).

# Макет базы данных

Изначально проектируем нашу базу данных. В качестве базы данных было принято решение использовать документную – MongoDB. Так как в нашем проекте загружаются файлы формата JSON и отчет хранится в таком же формате. Теперь проектируем сущности нашего проекта, были взяты следующие сущности:

- Users: хранилище, которое хранит в себе информацию о пользователе, а именно его логин и пароль, а также его фамилию, имя и отчество. Но фамилию, имя и отчество заполнять не обязательно, так как это считается дополнительной информацией, и пользователь сам решает заполнять их или нет.

- Relays: хранилище, которое хранит в себе конфигурационный файл. При загрузке файла он изначально будет передан в хранилище, а после будет создан отчет, основываясь на этих данных.

Теперь построим ER-диаграмму

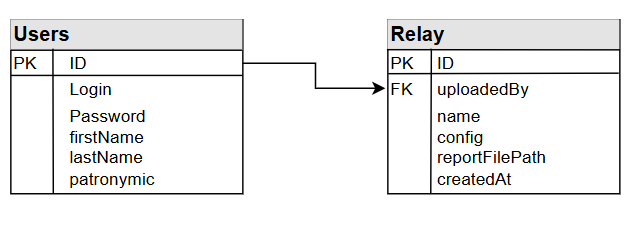


Рисунок 1 - ER-диаграмма базы данных

# Архитектура веб-сервера

# Создание бэкенда

Приступим к созданию серверной части нашего сервиса. Серверная часть будет написана с использованием Node.js Express. Сервер будет принимать запросы от фронтенда и отвечать на них. Создаем директорию relay\_audit\_server, в которой у нас будет хранится основный файл сервера server.js, который будет устанавливать подключение с базой данных и иметь жоступ к HTTP запросам от фронтенда:

1. routes – данная директория хранит файлы, в которых описана логика выполнения HTTP запросов и взаимодействия с базой данных.

2. mongo – хранит файлы, в которых описана структура документов коллекции в базе данных.

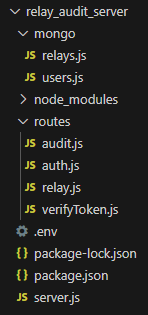


Рисунок 2 – Содержимое директории relay\_audit\_server

В директории routes мы создали следующие файлы:

1. audit – модуль для аудита конфигурации реле. В данном файле мы можем проверять конфигурационные параметры, формировать отчет

2. auth – реализация регистрации, входа, обновления токенов, обновления профиля и смены пароля

3. relay – работа с объектами реле: создание, чтение, обновление, удаление

4. verifyToken – проверка JWT токена



Рисунок 3 – Настройка файла server.js

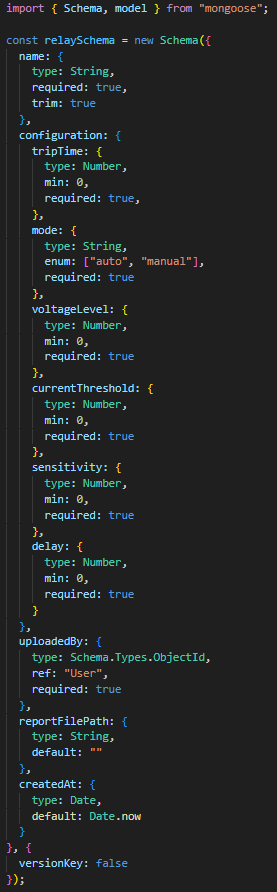


Рисунок 4 – Настройка файла relays.js

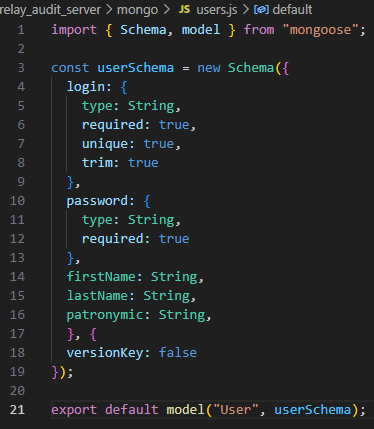


Рисунок 5 – Настройка файла users.js

И теперь проверим запуск нашего сервера.

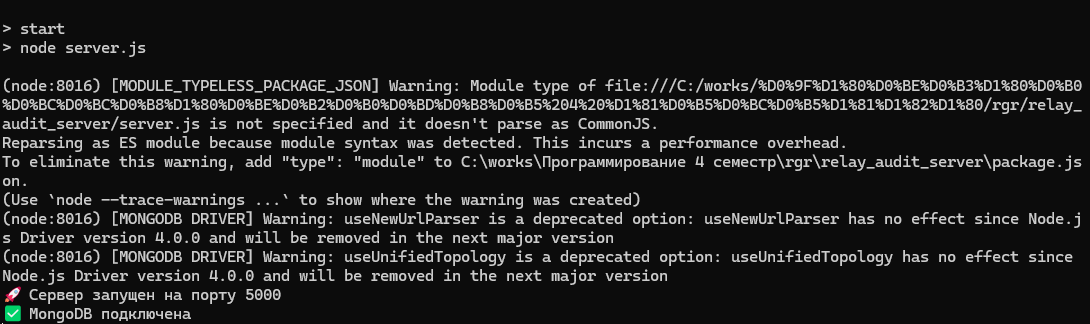


Рисунок 6 – Запуск сервера

# Создание фронтенда

После чего создаем веб-приложение, которое будет представлять SPA структуру и будет выбран React. Для начала при помощи команды: npx create-react-app relay\_audit создаем наш фронтенд.

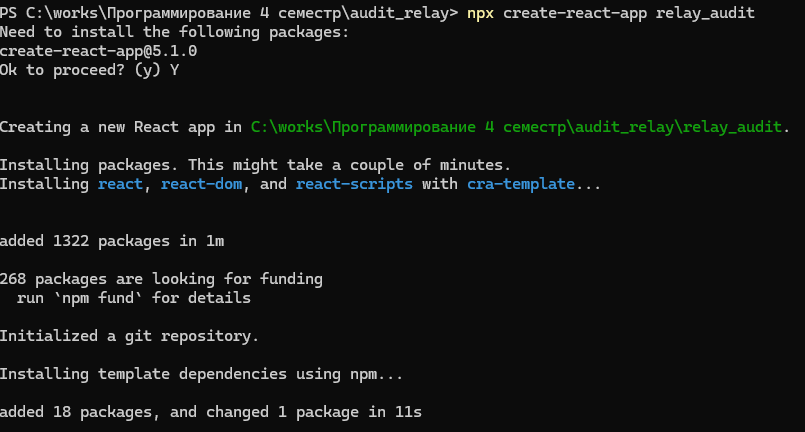


Рисунок 7 – Создание директории проекта

Также заранее установим, при помощи команды, утилиту, которая позволит нам работать с HTTP-запросами: npm install react-router-dom axios

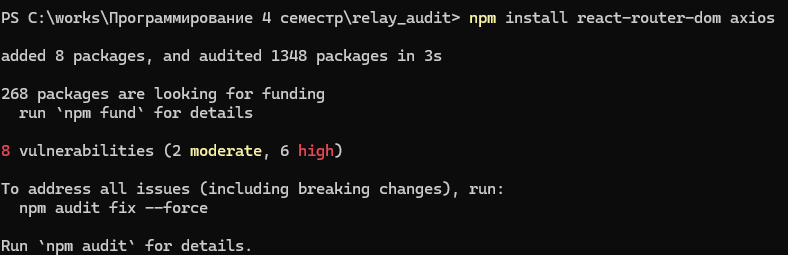


Рисунок 8 – Установка утилиты axios

Теперь создаем в директории src папку pages для страниц в нашем сервисе.

1. Register – страница регистрации.

2. Login – страница авторизации.

3. Home – главная страница.

4. Profile – страница профиля пользователя

5. Upload –страница для загрузки конфигураций.

6. Report – страница списков отчетов.

7. Details – страница с информацией о конфигурациях.

После этого удаляем все файлы кроме App.js, index.js. В App.js нужно изменить содержимое, а именно необходимо создать защищенный роутинг, чтобы нельзя было получить доступ к защищенным страницами, изменяя URL-адрес в поисковой строке браузера.

Далее нам потребуется создать дополнительный файл PrivateRoute.js, в котором мы настроим переадресацию на страницу авторизации, если пользователь не авторизован и пытается получить доступ к защищенной странице.

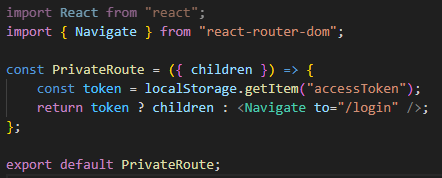


Рисунок 9 – Настройка файла PrivateRoute.js



Рисунок 10 – Настройка файла App.js

Структура проекта выстроена, теперь можно приступить к реализации страниц и логики на нашем сервере и клиенте.

# Документация

# Бэкенд

Регистрация:

Серверная часть принимает HTTP запрос, который отправляет клиентская часть и направляет его в базу данных. За это отвечает файл auth.js.

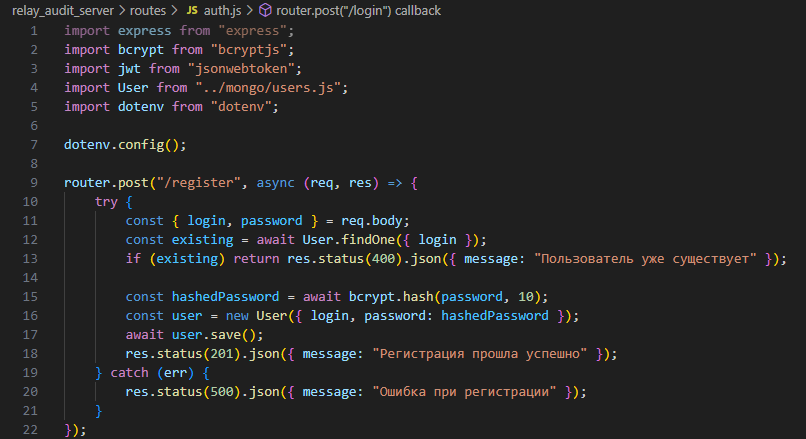


Рисунок 11 – Код серверной части auth.js

Авторизация:

Для авторизации происходит обработка HHTP запроса в файле auth.js. Для этого нужно дополнить ранее написанный маршрут для регистрации. Новый маршрут, также будет направлен в базу данных для проверки, что пользователь действительно существует.

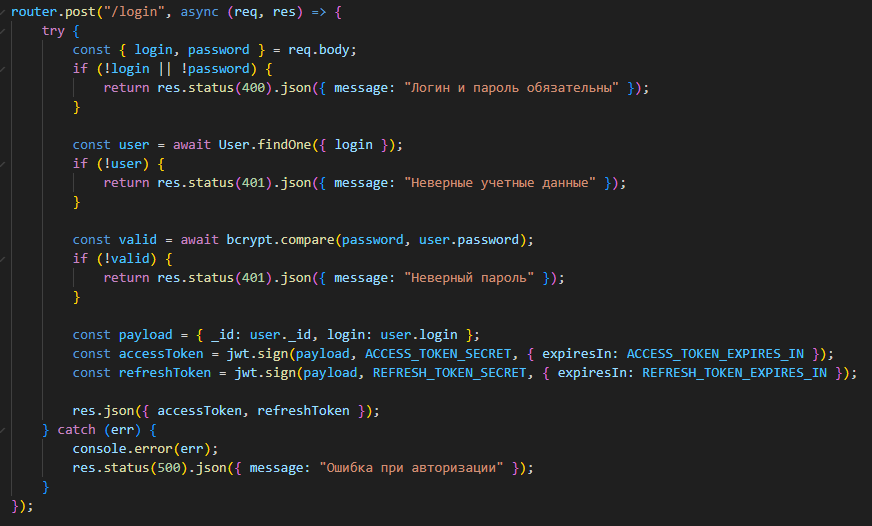


Рисунок 12 – Добавление маршрута для авторизации в auth.js

Главная страница:

Данная странице не обрабатывает никакие HTTP запросы. Главная страница нам нужна для перехода в профиль и загрузки конфигурационных файлов.

Профиль:

В серверной части происходит обработка HTTP запроса на смену пароля и добавления дополнительной информации. Данные маршруты содержатся в файле auth.js:

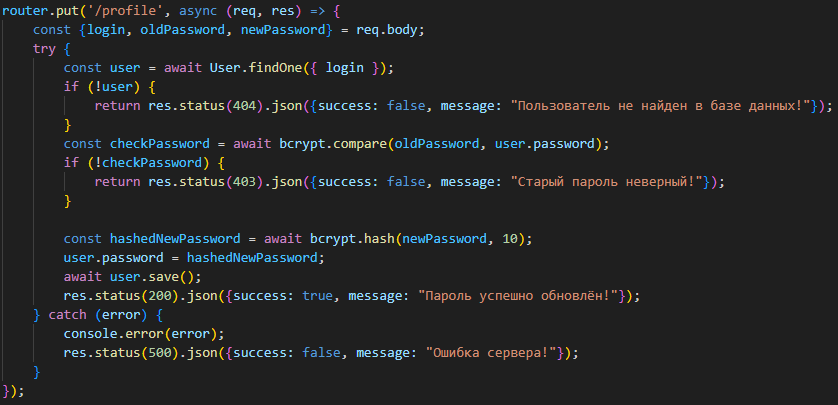


Рисунок 13 - Добавление маршрута для изменения пароля в auth.js

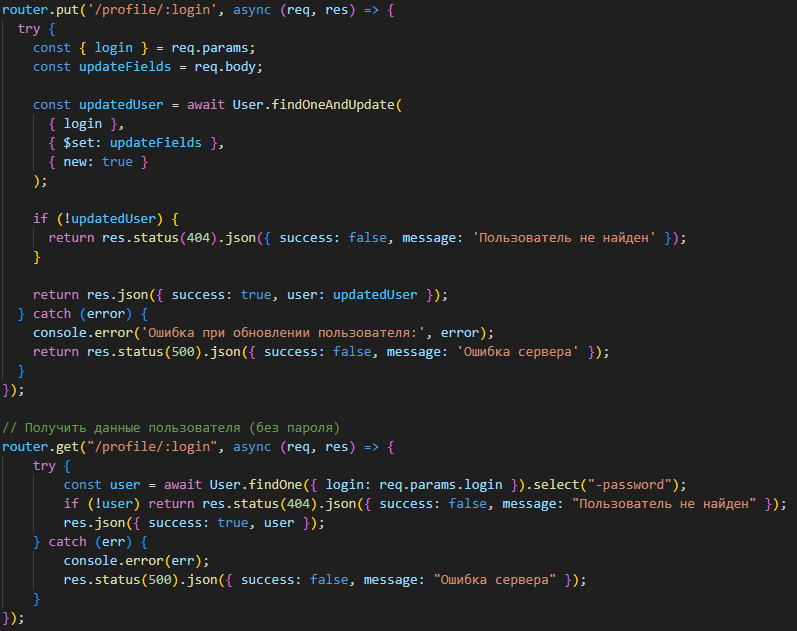


Рисунок 14 – Добавление маршрутов для получения и обновления данных у пользователя

Страница загрузки конфигурационных файлов:

За загрузку файла на сервере отвечает файл relay.js. Данный файл принимает HTTP запрос и отправляет его в базу данных, чтобы внести данный файл в список. После чего создается директория reports, которая хранит уже готовые отчеты о нарушениях. Сам отчет создается при помощи файла audit.js. Файл нужно загружать в формате JSON, после чего отчет формируется и сохраняется тоже в формате JSON.



Рисунок 15 – Маршрут для создания реле

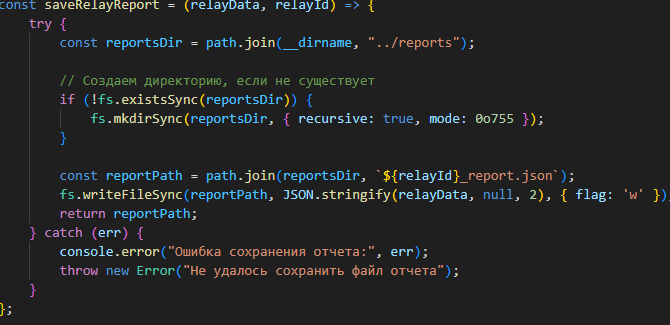


Рисунок 16 – Создание директории reports

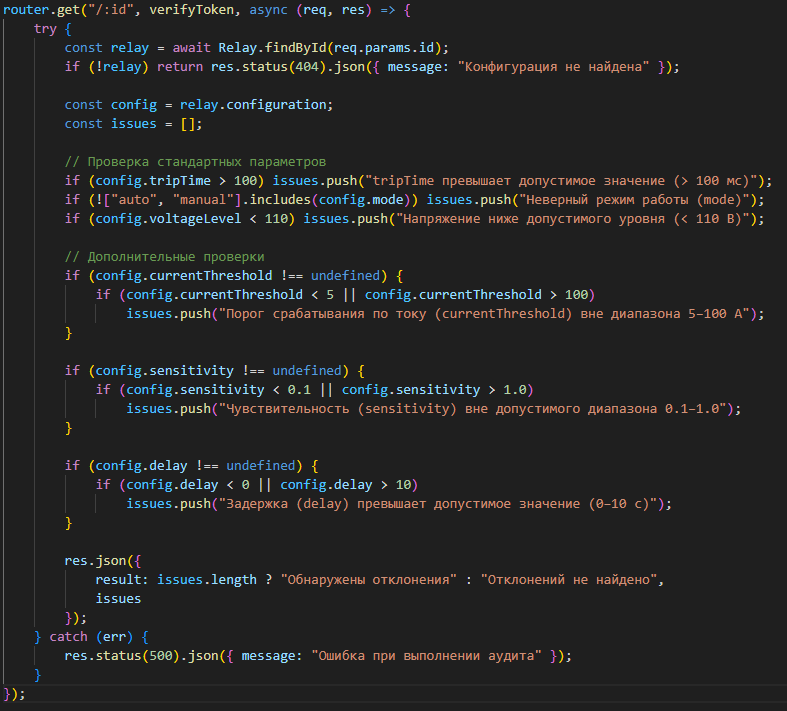


Рисунок 17 – Создание отчета о нарушениях

Страница списков отчета:

Серверная часть принимает HTTP запросы на удаление реле. Также она хранит списки действующих отчетов, все эти маршруты содержатся в файле relay.js.

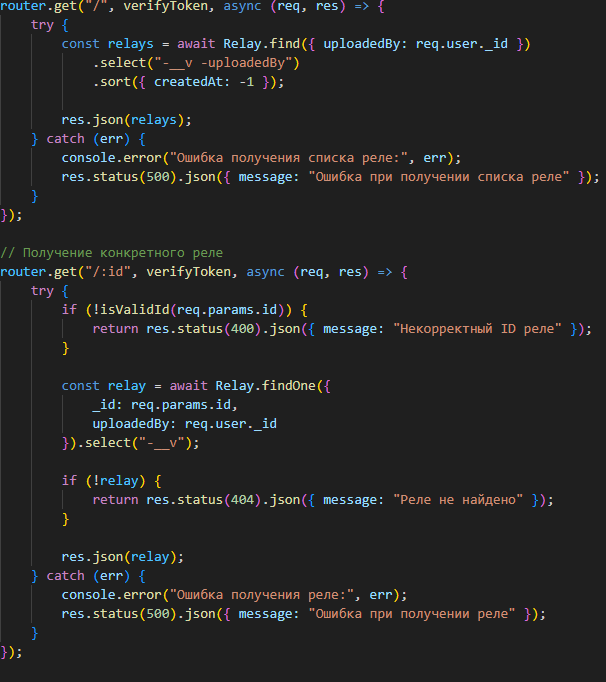


Рисунок 18 – Маршруты для получения списка отчетов и реле

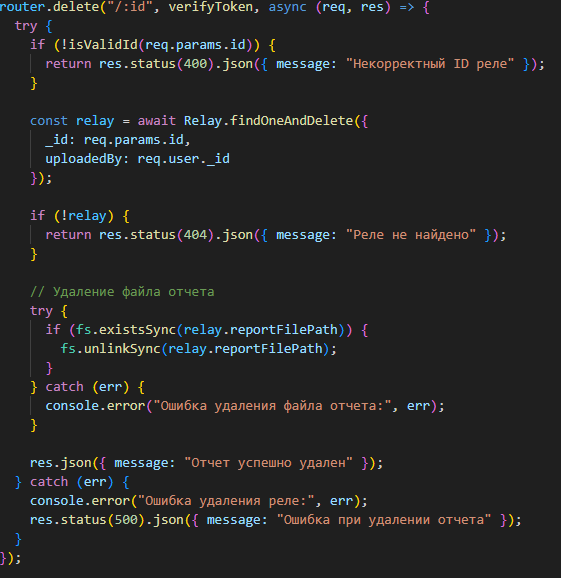


Рисунок 19 – Маршрут для удаления реле

Страница с информацией о файлах

Здесь серверная часть не принимает никаких HTTP запросов и не содержит никаких маршрутов.

# Безопасность в серверной части

В качестве безопасности было принято решение использовать JWT в наш веб-сервис. JWT – это способ передачи информации между двумя сторонами как подписанный JSON-объект. В нашем сервисе он будет использоваться для защиты защищенных маршрутов. Также добавим еще продление сессии с помощью localstorag.

Для начала в серверной части в ранее созданном файле .env указываем переменные окружения, которые представляют собой: ключи для access и refresh токенов, сроки их действия, а также настроим дополнительные парметры.

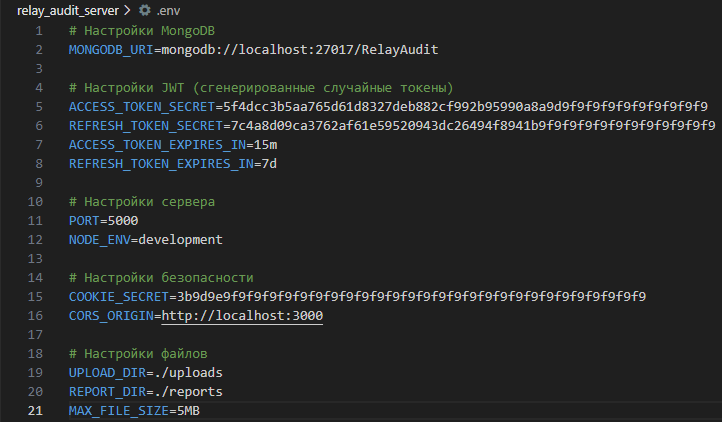


Рисунок 20 –Содержимое файла .env

Далее нам необходимо обновить файл auth.js, а именно маршрут /login.

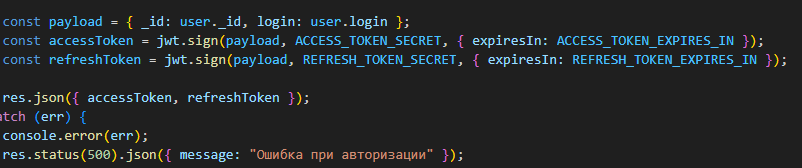


Рисунок 21 – Обновление маршрута /login

Создадим новый маршрут /refresh-token, который будет обновлять access токен. Данный маршрут мы создаем также в файле aunt.js.

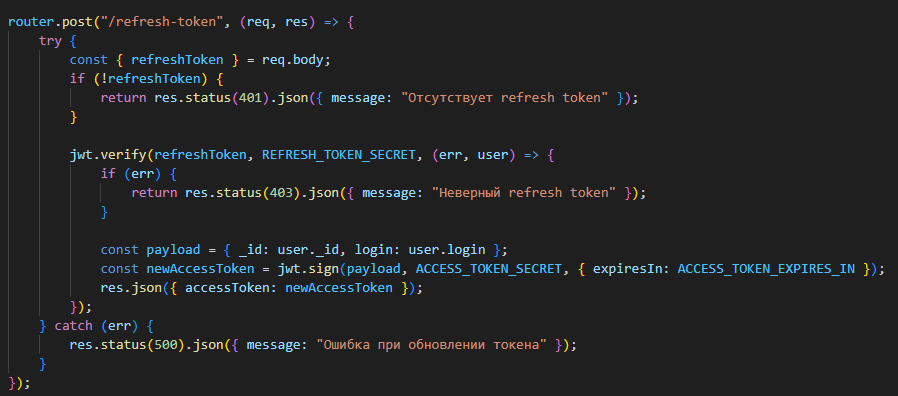


Рисунок 22 – Создание нового маршрута /refresh-token

Также на сервере нам необходимо в каждом защищенном запросе выполнять проверку на наличие заголовка с токеном. В файл verifyToken мы будем встраивать в защищенные запросы.

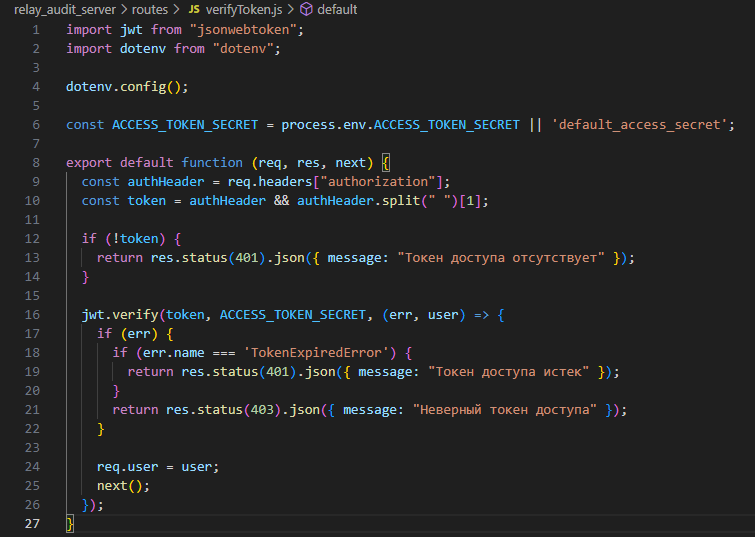


Рисунок 23 –Содержимое файла verifyToken.js

Также нам нужно обеспечить защиту от XSS, потому что мы будем использовать localstorage и sessionstorage для хранения токенов. Для этого необходимо включить Content Security Policy, то есть запретить выполнение инлайновых скриптов.

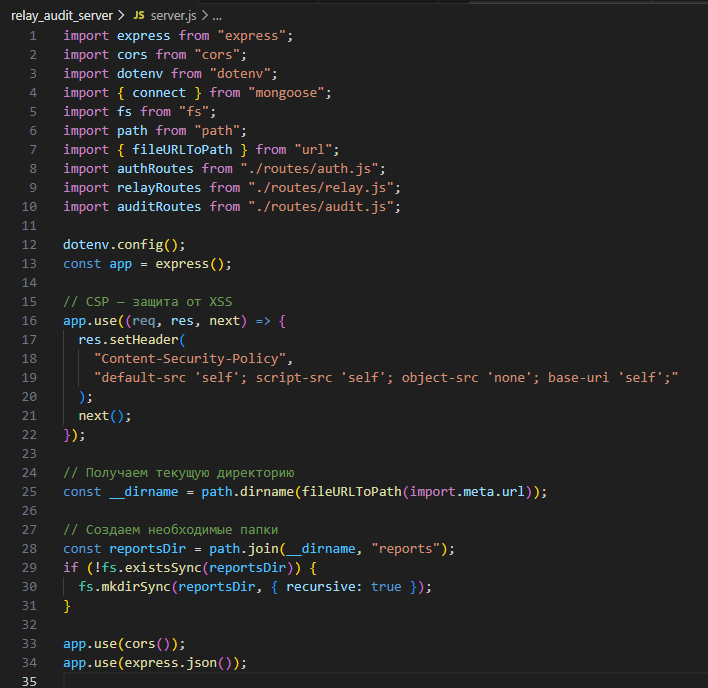


Рисунок 24 – Включение Content Security Policy

# Фронтенд

Регистрация:

Реализуем регистрацию с полями для логина и пароля, а также кнопкой зарегистрироваться. На странице пользователь также сможет перейти на страницу авторизации, если он уже имеет аккаунт. Для этого мы будем использовать файл Register.js, который у нас находится в фронтенде.

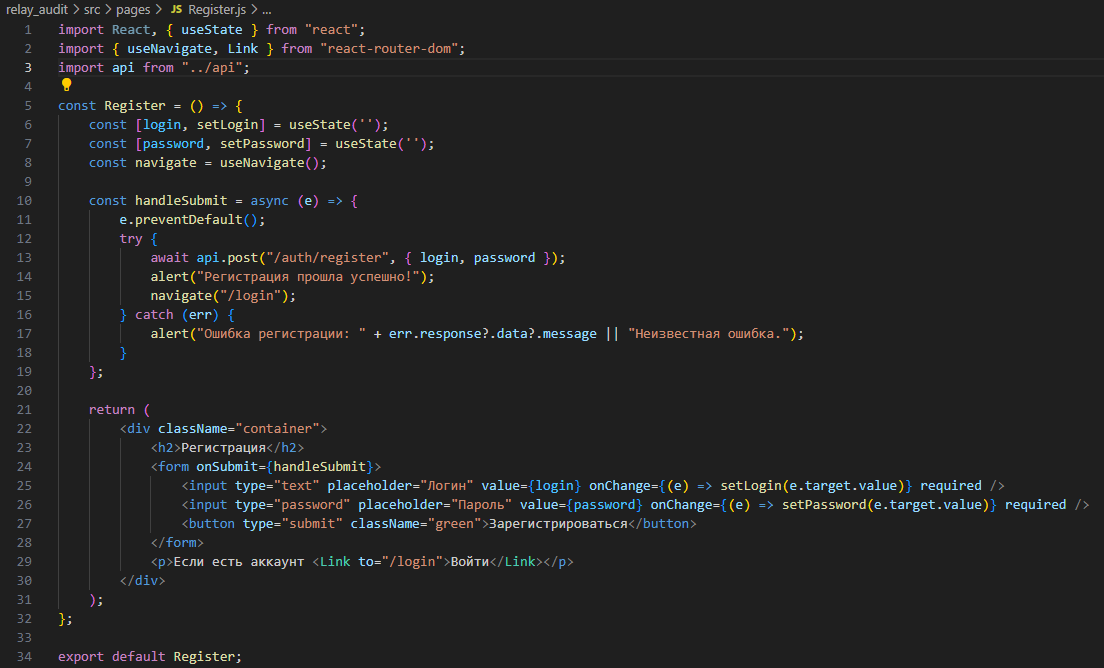


Рисунок 25 – Код страницы Register.js

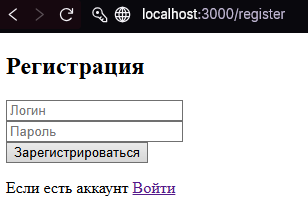


Рисунок 26 – Вид страницы регистрации

Авторизация:

Для авторизации также сделаем поля для ввода и кнопку войти. Также на странице будет присутствовать ссылка на страницу регистрации, если пользователь не имеет аккаунта.

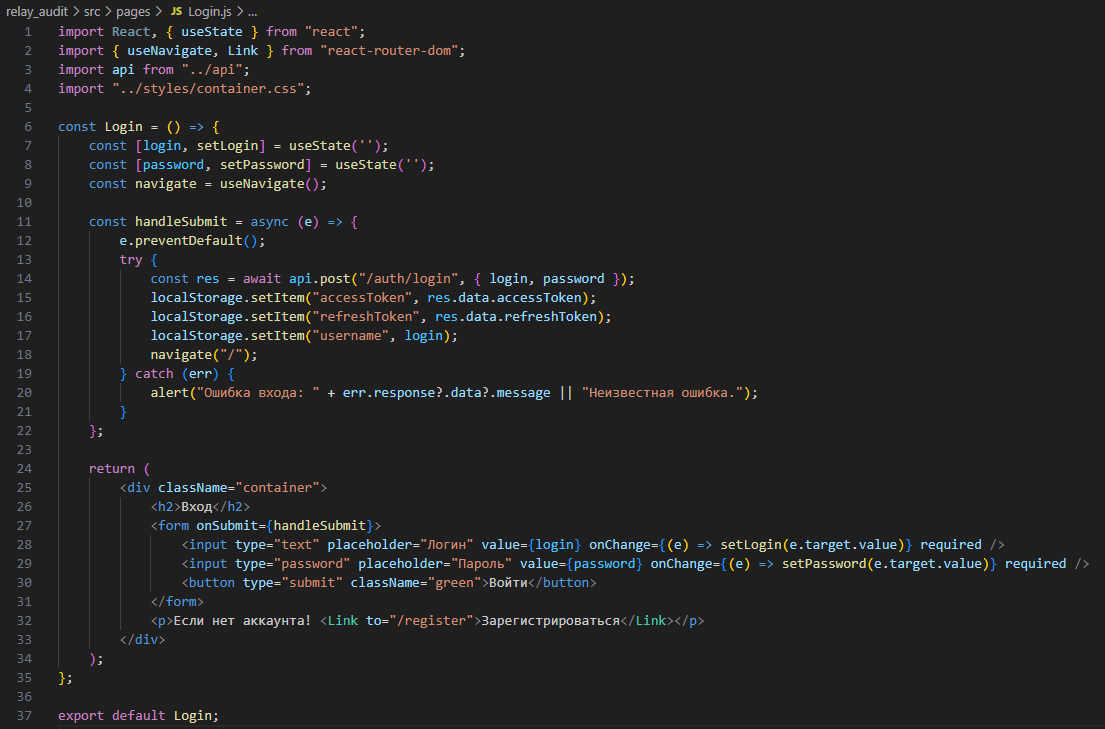


Рисунок 27 – Код страницы авторизации

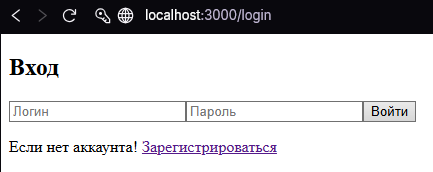


Рисунок 28 – Вид страницы авторизации

Главная страница:

На главной странице пользователь может перейти на страницу загрузки нового файла. Также он сможет выйти из системы с помощью кнопки и перейти в профиль своей учетной записи. За главную страницу на фронтенде отвечает файл Home.js.

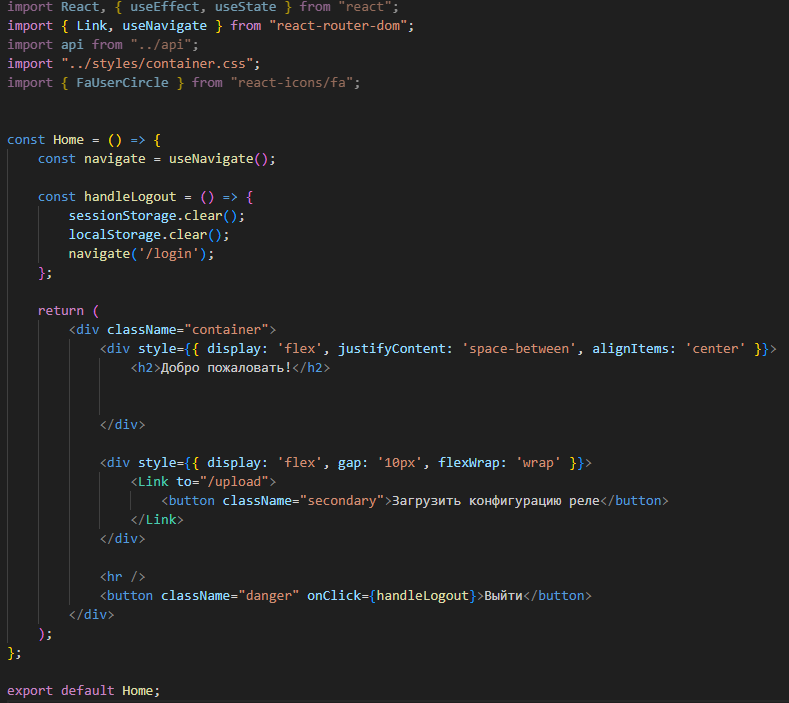


Рисунок 29 – Код главной страницы

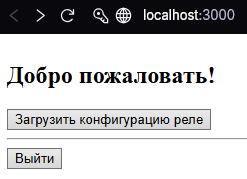


Рисунок 30 – Вид главной страницы

Страница профиля

На странице профиля нам будет предложено ввести дополнительную информацию о пользователе, а также изменить пароль. Для изменения пароля нужно нажать на кнопку показать пароль. За это у нас будет отвечать Profile.js.

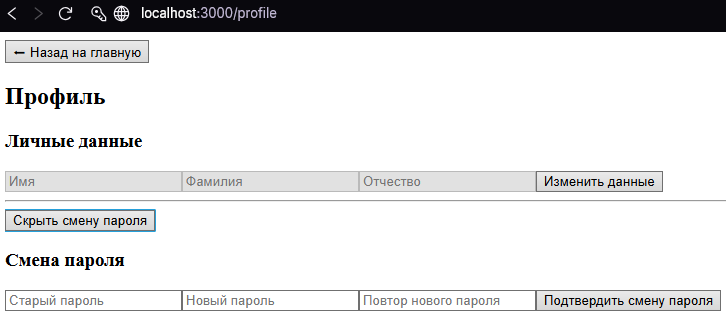


Рисунок 31 – Вид страницы профиля

Страница загрузки файла:

Данная страница содержит в себе кнопку загрузки файла (файл принимается только в формате JSON). После чего можно нажать на кнопку загрузить и у нас будет составляться отчет. За эту страницу отвечает файл Upload.js.



Рисунок 32 – Код страницы загрузки файла

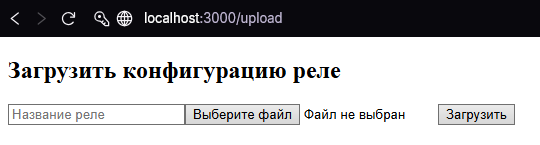


Рисунок 33 – Вид страницы загрузки файла

Страница отображения информации об отчетах конфигурации реле:

Страница отвечает за отображение отчетов всех конфигураций, которые мы загрузили, а также есть возможность удалить отчеты. Когда пользователь загрузит конфигурационный файл его перебросит на данную страницу. За эту страницу на фронтенде отвечает Report.js.



Рисунок 34 – Код страницы отображения отчетов

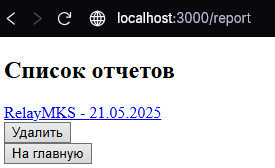


Рисунок 35 – Страница списка отчетов

Страница отображения отчетов об нарушениях:

Данная страница будет отображать информацию с конфигурационных файлов реле. После нажатия на название отчета нас перенаправит на эту страницу. За эту страницу на фронтенде отвечает файл Details.js. В серверной части файл опять будет отвечать relay.js, который будет получать отчет из директории reports, которая создается здесь же и после чего отправит его на фронтенд и базу данных.



Рисунок 36 – Код страницы отображения данных нарушений

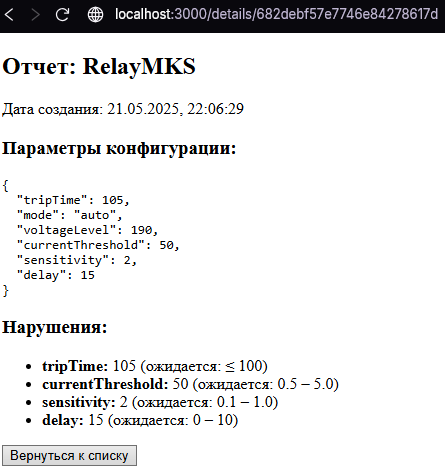


Рисунок 37 – Вид страницы отображения информации с конфигурационного файла

# Безопасность в клиентской части

Для фронтенда в качестве безопасности необходимо реализовать добавление токена к каждому запросу и автоматическое обновление токена при окончании его действия. Для этого нам потребуется сторонний файл api.js и используем его в нужных нам запросах.



Рисунок 34 – Настройка файла api.js

# Тестирование

Перед тестированием для удобства добавим css-стили, чтобы наш сервис имел красивый и понятный интерфейс. Стили взяты с открытых источников.

Для начала перейдем на страницу регистрации и создадим нового пользователя. Вводим логин, пароль. После чего нажимаем кнопку регистрации. После успешной регистрации мы будем перенаправлены на страницу авторизации, где также вводим данные пользователя и нажимаем кнопку войти. После чего нас перебросит на главную страницу сервиса.

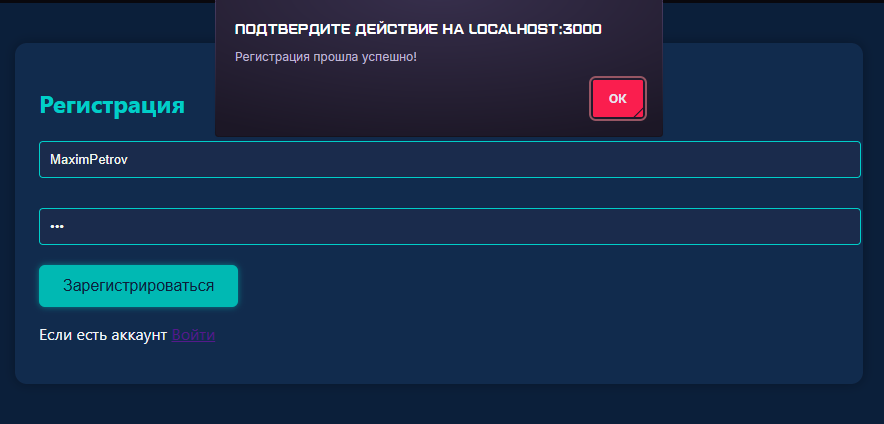


Рисунок 36 – Регистрация пользователя

После успешной регистрации нас перенаправит на страницу авторизации, где также вводим данные пользователя и нажимаем кнопку войти.

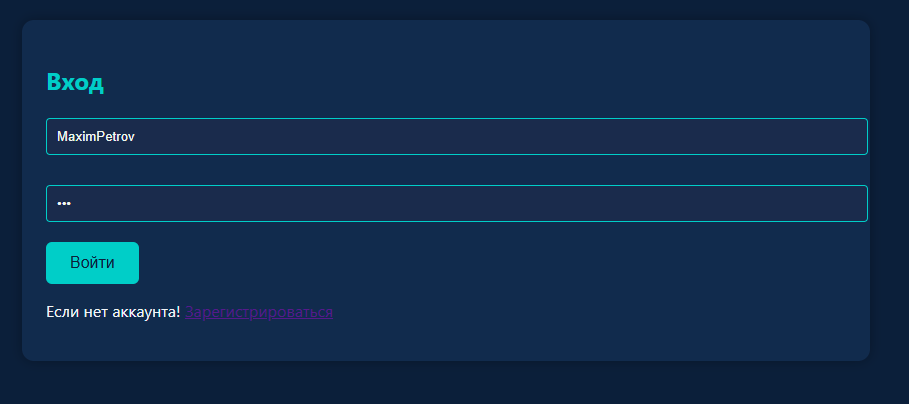


Рисунок 37 – Авторизация пользователя

Теперь мы оказываемся на главной странице сервиса.

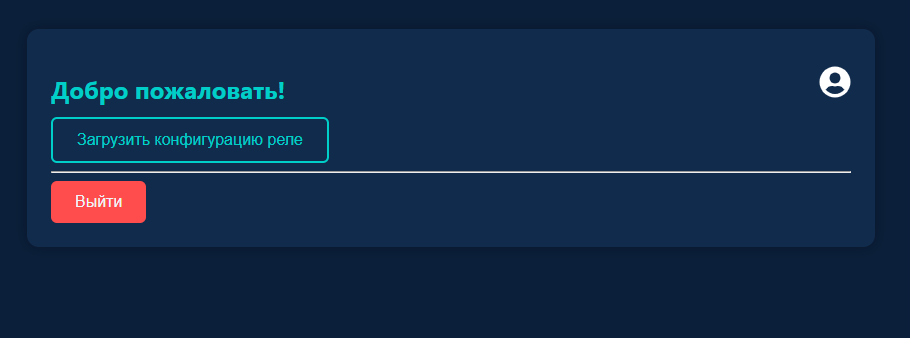


Рисунок 38 – Главная страница сервиса

Далее на главной странице мы нажмем на иконку в правом верхнем углу и перейдем в профиль.

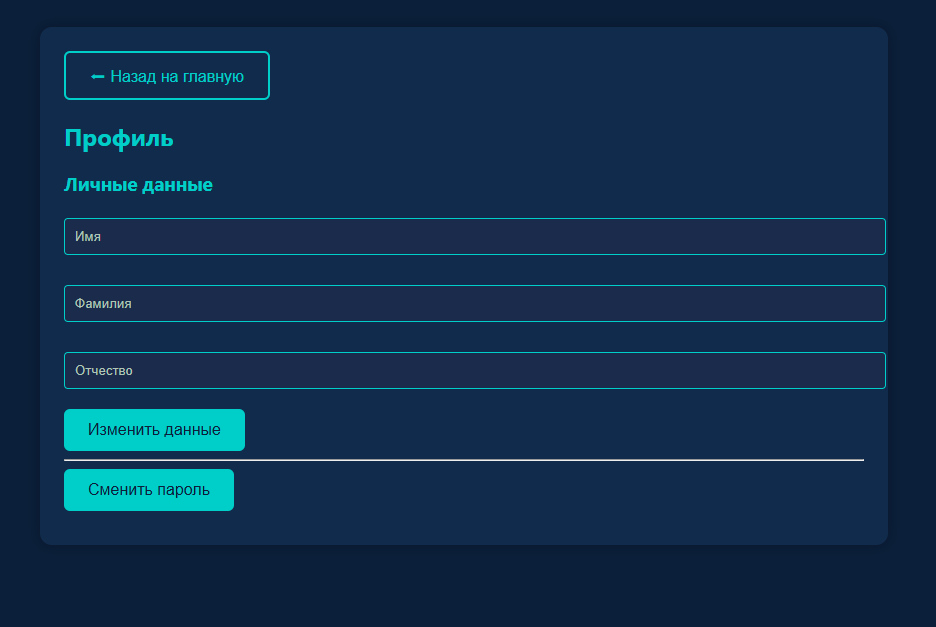


Рисунок 39 – Страница профиля

Теперь добавим дополнительную информацию о нашем пользователе:

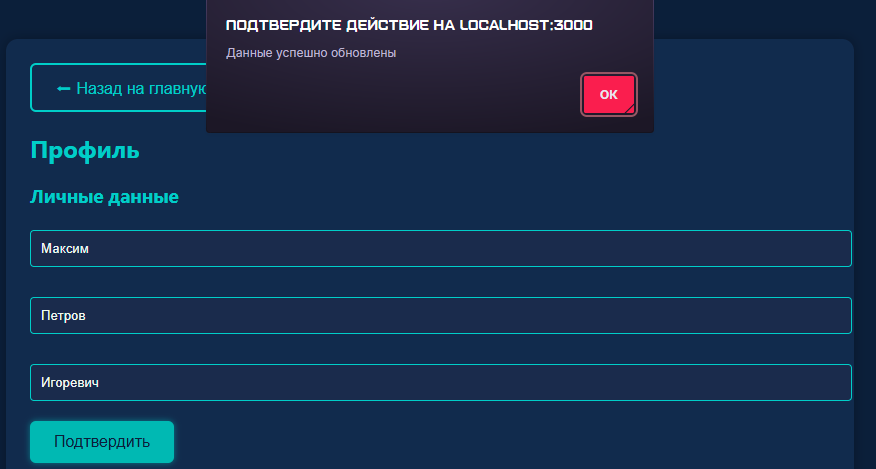


Рисунок 40 – Ввод дополнительной информации

После попробуем изменить наш пароль. Для этого нажимаем на кнопку показать пароль и меняем его.

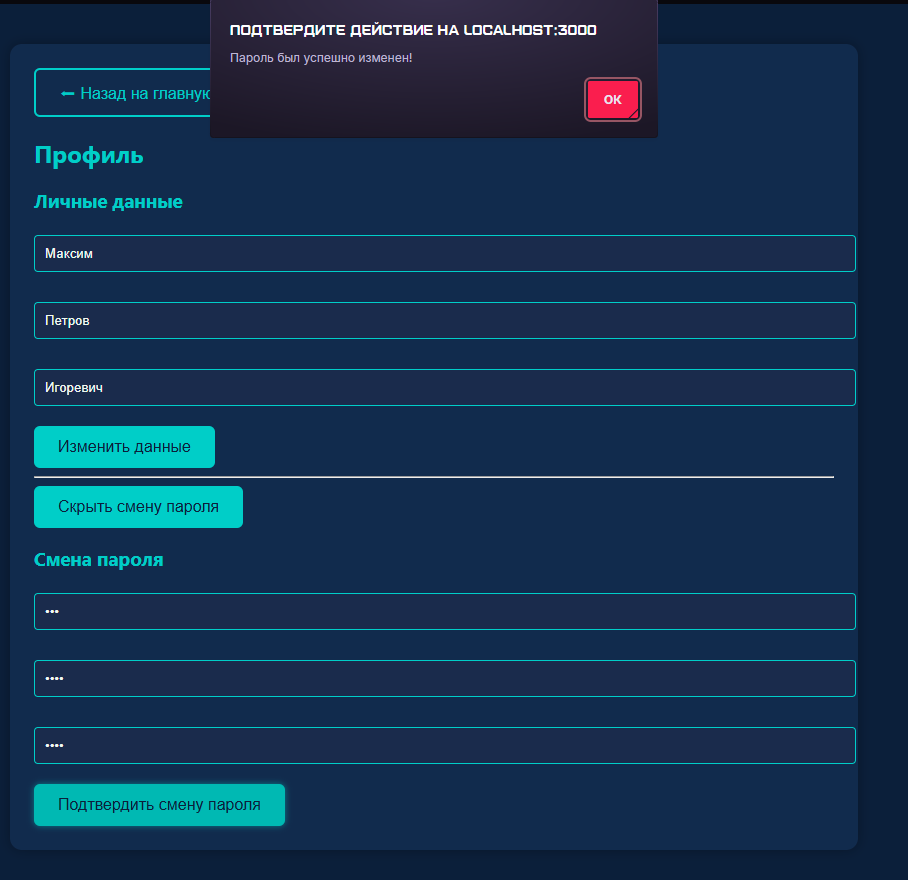


Рисунок 41 – Изменение пароля пользователя

Загружаем конфигурацию реле. Для этого нажимаем на кнопку загрузить реле и нас перемещает на страницу загрузки.

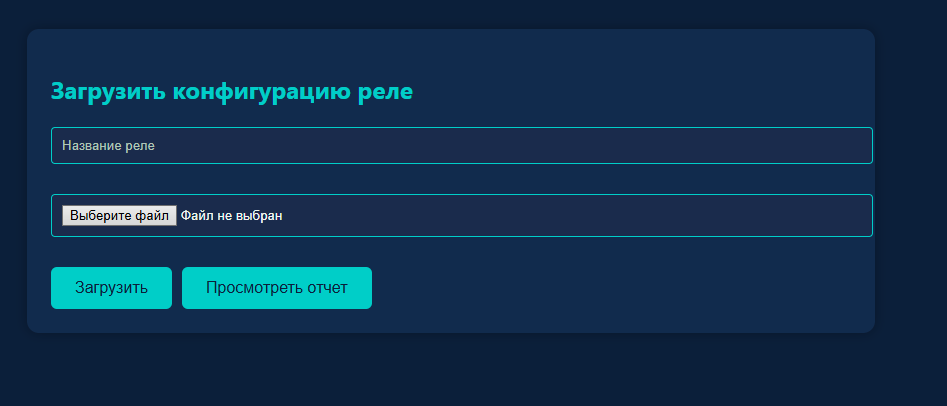


Рисунок 42 – Страница загрузки реле

Изначально нужно ввести имя нашего реле, а затем выбрать файл. После этого нас перебросит на страницу отчетов, и мы можем выбирать все отчеты, которые нам доступны.

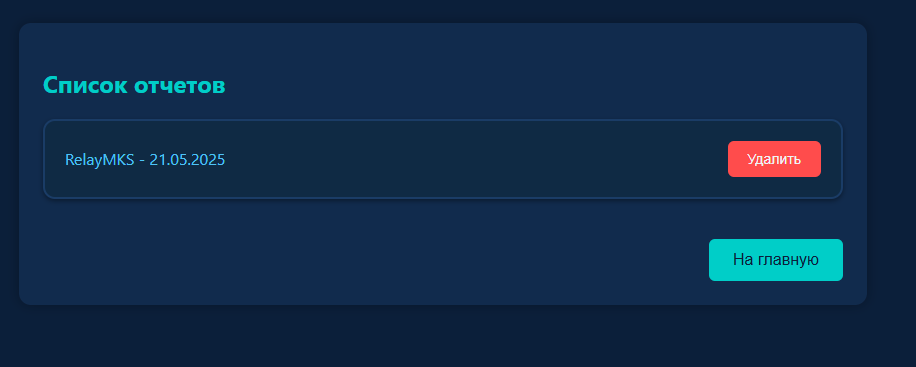


Рисунок 43 – Страница списка отчетов

Для перехода к отчету нажимаем на название нашего конфигурационного файла:

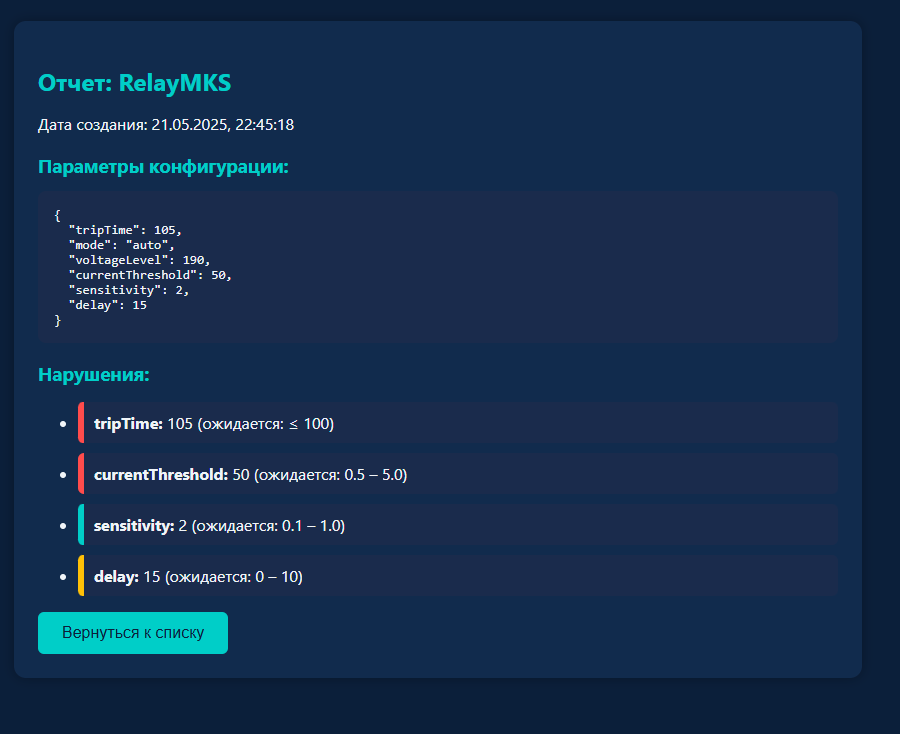


Рисунок 44 – Страница отчета реле

Далее можем вернуться назад к отчетам и удалить нашу конфигурацию.

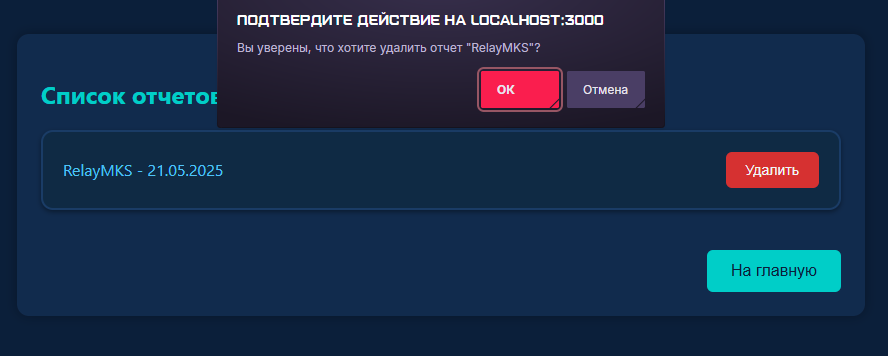


Рисунок 45 –Удаление отчета

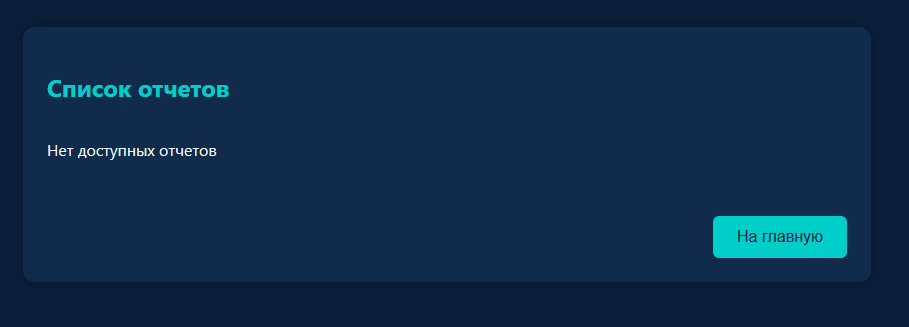


Рисунок 46 – Страница после удаления отчета

# Заключение

В ходе выполнения расчетно-графического задания по дисциплине «Технологии и методы программирования» была разработана полноценная информационная система для аудита конфигураций реле защиты. Основной целью проекта являлось создание веб-сервиса, позволяющего автоматизировать процесс проверки конфигурационных файлов на соответствие регламентам, а также генерацию отчетов о выявленных нарушениях.

Разработка охватила как серверную, так и клиентскую части. Для реализации серверной части использовался стек технологий Node.js и Express, а в качестве базы данных была выбрана MongoDB, что позволило эффективно обрабатывать и хранить данные в формате JSON. Клиентская часть была реализована с использованием React, что обеспечило создание современного одностраничного веб-приложения (SPA) с удобным пользовательским интерфейсом.

Особое внимание было уделено вопросам безопасности: реализована система аутентификации и авторизации пользователей на основе JWT, защита маршрутов, а также меры по предотвращению XSS-атак. В процессе тестирования были проверены основные сценарии взаимодействия пользователя с системой, включая регистрацию, авторизацию, загрузку конфигураций, генерацию и просмотр отчетов.

Таким образом, поставленные цели и задачи работы были полностью достигнуты, а разработанный сервис продемонстрировал свою работоспособность и прикладную значимость.

# Приложение

Ссылка на репозиторий: <https://github.com/Mixassss/RGR_4.git>