

Министерство науки и высшего образования Российской Федерации
Федерального государственного бюджетного образовательного учреждения
высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ
И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра комплексной информационной безопасности электронно-
вычислительных систем (КИБЭВС)

ПРИЛОЖЕНИЕ, РЕАЛИЗУЮЩЕЕ ДИСКРЕЦИОННУЮ МОДЕЛЬ
ДОСТУПА К ДАННЫМ

Курсовая работа по дисциплине «Программно-аппаратные средства
обеспечения информационной безопасности»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Студенты группы 729-1

_____ Э. Э. Белозерцев

Доцент кафедры БИС

_____ И. А. Рахманенко

«__» _____ 2023 г.

Реферат

Курсовая работа содержит 30 страницы пояснительной записки с учетом приложений (1 экземпляр на 6 листах), 23 рисунков.

Ключевые слова: дискреционная модель доступа, права доступа, Express.js, JavaScript, web-приложение.

Название программы: «Приложение, реализующее дискреционную модель доступа к данным».

Цель работы: изучение предметной области, проектирование, разработка и тестирование системы, позволяющей разграничить доступ субъектов к объектам на основе дискреционной модели.

Приложение представляет собой клиент-серверное приложение и состоит из трех основных подсистем:

- Серверное приложение (NodeJS с использованием Express.js).
- База данных (SQLite).
- Клиентское приложение (React).

Пояснительная записка выполнена в «Microsoft Office Word TLSC профессиональный плюс 2021», оформлена согласно ОС ТУСУР 01-2021.

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)

УТВЕРЖДАЮ

Заведующий кафедрой КИБЭВС

д-р техн. Наук, профессор

_____ А. А. Шелупанов

«___» _____.2022

ЗАДАНИЕ

На курсовую работу по дисциплине «Программно-аппаратные средства обеспечения информационной безопасности» студенту Белозерцеву Эдуарду Эдуардовичу группы 729-1 факультета Безопасности

1. Тема работы:

Приложение, реализующее дискреционную модель доступа к данным.

2. Срок сдачи студентом законченной работы:

«12» июня 2023 года.

3. Исходные данные к работе:

- Express.js (NodeJs) – серверная часть;
- HTML, CSS, JavaScript – клиентская часть;
- SQLite – база данных.

4. Содержание пояснительной записки:

- титульный лист;
- реферат;
- техническое задание;
- введение;
- обзор предметной области;
- описание архитектуры приложения;
- разработка приложения;
- тестирование;
- заключение;
- список литературы;
- приложение (Руководство программиста).

5. Перечень подлежащих разработке вопросов:

5.1 Веб-приложение с возможностью создания (через веб-интерфейс) новых пользователей и определения их прав доступа.

5.2 Механизм дискреционного разграничения доступа к данным: право на чтение (GET-запрос в HTTP), право на запись (POST-запрос), право на изменение (PUT-запрос) и удаление (DELETE-запрос).

5.3 Модель базы данных для предметной области (пользователи и их права доступа).

5.4 Авторизация пользователей и получение пользователями доступа к данным в соответствии с их правами.

6. Задание выдано:

Руководитель:

к.т.н., доцент кафедры БИС

И. А. Рахманенко

(подпись)

«__» _____ 2023 г.

7. Задание принято к исполнению:

Студент группы 729-1

Э. Э. Белозерцев

(подпись)

«__» _____ 2023 г.

Введение

В дискреционной модели разграничения доступа (DAC) принцип контроля доступа основан на правах, назначенных самими владельцами данных или ресурсов. В этой модели владельцы имеют полный контроль над доступом и могут устанавливать права доступа для других пользователей.

Цель работы – изучение предметной области, проектирование, разработка и тестирование приложения, реализующего дискреционную модель доступа к данным.

Для разработки выбраны следующие технологии:

- База данных – SQLite (пакеты `sqlite` и `sqlite3` из NPM);
- Серверная часть – фреймворк `Express.js`;
- Клиентская часть – библиотека `React.js` (с использованием `JSX` и `CSS`).

Таким образом во всех подсистемах приложения для разработки применяется язык программирования JavaScript.

1 Обзор предметной области

Вследствие наличия многих возможных источников нарушения безопасности информации, обрабатываемой с использованием средств вычислительной техники, возникает задача обеспечения защищенности этой информации без существенного ухудшения показателей качества реализации процесса её обработки. Одним из методов защиты, используемых для решения этой задачи, является разграничение доступа пользователей компьютерной системы к имеющимся в системе информационным ресурсам [1].

Основные понятия дискреционной модели – субъекты (пользователи), объекты (файлы, документы и другие) и разрешения (операции, направленные на обработку или взаимодействие с данными). Для каждой пары субъект-объект явно перечисляются доступные привилегии, определяющие санкционированные действия данного субъекта по отношению к данному объекту [2].

В зависимости от реализации права доступа могут задаваться либо администратором, либо пользователем, который создал объект (владельцем). Для работы выбран подход определения прав доступа владельцами.

2 Описание архитектуры приложения

Приложение должно быть реализовано в виде клиент-серверного приложения и представлять собой простой блог – у каждого зарегистрированного пользователя есть личная страница, где он может оставлять записи и настраивать права доступа других пользователей к своему блогу.

В приложении должны быть реализованы следующие функции:

- Регистрация и авторизация пользователей.
- Дискреционное разграничение доступа к личной странице каждого пользователя: право на чтение (GET-запрос в HTTP), право на запись (POST-запрос), право на изменение (PUT-запрос) и удаление (DELETE-запрос).
- Просмотр, создание, изменение и удаление постов на страницах пользователей

Доступ к клиентской части приложения должен осуществляться с помощью любого современного интернет-браузера. Для полноценного функционирования клиентской части приложения в браузере должно быть разрешено выполнение JavaScript.

Основные страницы, которые должны быть представлены в клиентском приложении:

- Страница регистрации и авторизации.
- Личная страница пользователя с постами.
- Страница настроек прав доступа.

Серверная часть должна представляться собой веб-сервер, в непрерывном режиме ожидающий входящих http-запросов и набора скриптов, обрабатывающих эти запросы. Метод дискреционного разграничения доступа реализуется именно на уровне серверной части приложения с использованием промежуточных обработчиков запросов (в концепции Express – «middleware») и матрицы доступа.

Веб-сервер должен обеспечивать взаимодействие при помощи вызова соответствующих API-функций. Список конечных точек («endpoints»), которые обязательно должны быть реализованы:

- «/account/register» и «/account/login» для POST-запросов для регистрации и аутентификации соответственно;
- «/api/posts» для выполнения основных четырех типов запросов (GET, POST, PUT, DELETE) на работу с постами пользователя;
- «/api/settings» для выполнения запросов на работу с матрицей доступа.

База данных должна состоять из трех связанных таблиц – «Посты», «Пользователи» и «Права» (Рисунок 2.1).



Рисунок 2.1 – Диаграмма базы данных

На все поля таблиц БД должно быть наложено ограничение «NOT NULL», запрещающее создание записей с пустыми полями.

В таблице «Rights» предполагается использование типа BOOL для полей прав доступа, но в виду отсутствия в SQLite логического типа использоваться должен тип INTEGER с приведением к типу BOOL внутри программы.

3 Разработка приложения

Проект состоит из двух папок: «backend» и «frontend» (Рисунок 3.1).



Рисунок 3.1 – Структура проекта

3.1 Серверная часть приложения

Главным файлом проекта является «index.js» – в нем подключаются основные зависимости, создается и конфигурируется главный объект сервера и запускается главный цикл ожидания входящих запросов.

Важные скрипты, в которых задается маршрутизация запросов и вызов функций-обработчиков (из файла «functions.js») – «accountAcontroller.js» и «apiController.js», находящиеся в папке «controllers» (Рисунок 3.1).

Маршрутизация запроса с использованием фреймворка Express выглядит следующим образом: для объекта «router» настраивается тип запроса и URL-путь, к которому прикрепляется функция обработчик. В приведенном в пример случае обработчик – анонимная функция (Рисунок 3.2). Внутри функции выполняется деструктуризация полей тела запроса, затем проверка корректности значений этих полей и непосредственная регистрация (вызов функции для создания нового пользователя в базе данных) – окончанием обработки запроса является отправка ответа в формате JSON.

```

router.post('/register', async (req, res) => {
  const { username, password } = req.body

  if (areCredentialsCorrect(username, password)) {
    let result = await register(username, password)
    if (result.status) return res.json({...result, ...generateToken(username, password)})
    return res.json(result)
  }

  return res.json({status: false, message: 'Your credentials dont meet the mininum security requirements'})
})

```

Рисунок 3.2 – Настройка обработки POST-запроса на адрес «/account/register»

Проверка прав доступа реализована в скрипте «authMiddleware.js». Основной смысл использования промежуточных обработчиков – направить запрос по разным маршрутам для его обработки в зависимости от условия – корректности токена пользователя. Если предоставленный пользователем токен удовлетворяет требованиям, то обработка запроса передается следующему в цепочке обработки скрипту, в обратном случае обработка прекращается с возвратом сообщения об ошибке пользователю (Рисунок 3.3)

```

const isTokenValid = checkToken(token, username, candidate.password)
if (isTokenValid) return next()

return res.json({status: false, message: "Not authorized - token is invalid"})

```

Рисунок 3.3 – Смысловая часть скрипта «authMiddleware.js»

Краткое описание не упомянутых файлов и папок:

- «node_modules/» – хранит файлы подключенных зависимостей
- «database.sqlite» – файл базы данных
- «package.json» и «package-lock.json» – хранят информацию о проекте (версия, автор, зависимости).

Список конечных точек API серверной части показан в таблице 3.1

Таблица 3.1 – Конечные точки серверной части приложения (API-endpoints)

Тип HTTP-запроса	Адрес	Назначение
GET	«/api /posts /:username»	Получение постов пользователя
POST	«/api /posts /:username»	Создание нового поста
PUT	«/api /posts /:id»	Изменение конкретного поста
DELETE	«/api /posts /:id»	Удаление конкретного поста
GET	«/api /settings»	Получение настроек доступа пользователя (матрицы доступа)
POST	«/api /settings»	Изменение матрицы доступа
POST	«/api /settings /check»	Получение информации о правах доступа из матрицы доступа
POST	«/api /settings /add»	Добавление нового правила (нового субъекта) в матрицу доступа
GET	«/api /friends»	Получение списка всех пользователей, разрешивших доступ на просмотр их страницы

3.2 Клиентская часть приложения

Клиентское приложение разрабатывается в формате SPA («single-page application») с использованием библиотеки React, поэтому файл «index.html» изначально не содержит никакой функциональной части – только корневой элемент (единственный тег «div» с идентификатором «root»). JS-код, который загружается в браузер пользователя при первом обращении к сайту, будет использовать этот корневой элемент в качестве начальной точки для построения DOM-дерева и отрисовки элементов на странице.

Главный файл клиентской части приложения – «index.js». Именно в нем подключаются основные зависимости верхнего уровня, происходит поиск корневого элемента и определяется порядок отрисовки приложения внутри корневого элемента.

В файле «App.js» настраивается маршрутизация пользователя внутри клиентской части приложения. Важно упомянуть, что благодаря использования React это маршрутизация не имеет ничего общего с URL-адресами для запросов к серверу. Пользователь может переключаться между страницами без перезагрузки, поэтому никаких обращений к серверу на получение дополнительных страниц не происходит. Для неавторизованного пользователя определены только маршруты для авторизации и регистрации (Рисунок 3.4)

```
frontend > src > App.js ...
17     return (
18       <authContext.Provider value={{ username, setUsername, token, setToken }}>
19         {
20           username && token ?
21           // Routes for registered user
22 >       <Routes>---
27         </Routes>
28       :
29       // Routes for guest
30       <Routes>
31         <Route path='/login' element={< LoginPage />} />
32         <Route path='/register' element={< RegisterPage />} />
33         <Route path="/" element={<Navigate to="/login" replace />} />
34       </Routes>
35     }
36   </ authContext.Provider>
37 )
```

Рисунок 3.4 – Маршрутизация в клиентской части (JSX-разметка)

Элементы, которые прикрепляются к адресам маршрутов, хранятся в папке «pages». В приложении предусмотрены четыре основных страницы, а верстка и функционал отдельных компонентов вынесена в файлы из папки «components» (Рисунок 3.5).

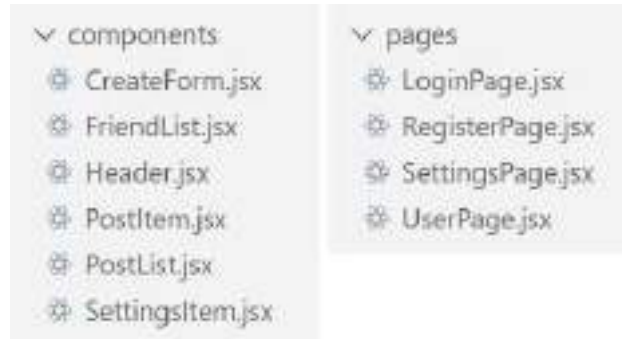


Рисунок 3.5 – Содержимое папок «pages» и «components»

Такой подход позволяет повысить модульность кода и его читаемость. Благодаря вынесению компонентов в отдельные модули возвращаемая разметка основной страницы становится минималистичной и доступной для понимания – страница «UserPage» возвращает элементы «Header», «CreateForm», «PostList» и «FriendList» по порядку (Рисунок 3.6). При загрузке страницы выполняется API-запрос к серверу для получения информации, которой будет заполнен JSX-шаблон страницы. Полученный JSON-ответ разбирается и сохраняется в переменную состояния – на рисунке показан код создания переменной состояния с помощью функции «useState», предоставляемой React, и функции выполнения запроса к серверу для получения информации и обновления созданного состояния – списка «друзей» (Рисунок 3.7).

```
return (  
  <div className='userPage container'>  
    <Header />  
    <CreateForm />  
  
    <Postlist className='postlist' username={targetUser}>...  
  </PostList>  
  
    {  
      targetUser == username &&  
      <Friendlist className='postlist' username={targetUser}>...  
    </FriendList>  
    }  
  </div>  
)
```

Рисунок 3.6 – Верстка страницы «UserPage»

```

const [friends, setFriends] = useState([])

const fetchFriends = async () => {
  const response = await (await fetch('/api/friends', { ...
  })).json()

  if (response.status) setFriends(response.data)
}

```

Рисунок 3.7 – Часть кода, обеспечивающего смысловое наполнение страницы

Компоненты могут либо использовать друг друга, либо быть независимыми элементами, написанными с помощью базовых HTML-тегов. Для работы выбран второй вариант, поэтому возвращаемая верстка представляет собой обычную HTML-разметку (Рисунок 3.8).

```

return (
  <form className='createForm_createForm'>
    <h1 className='createForm_header'>
      New post:
    </h1>
    <input
      className='createForm_input'
      type='text'
      name='header'
      id='header'
      placeholder='Header' />
    <input
      className='createForm_input'
      type='text'
      name='text'
      id='text'
      placeholder='Text' />
    <button id='createButton' onClick={createButtonHandler} className='createForm_button'>Create</button>
  </form>
)

```

Рисунок 3.8 – Верстка компонента «createForm» в виде HTML-разметки

Список не упомянутых файлов и папок:

- «public/» – хранит файлы (в том числе index.html), которые должны быть доступны по прямому обращению (sitename.com/image.png);
- «src/contexts» – папка для хранения некоторых значений в глобальном контексте выполнения приложения;
- «App.css» – CSS-файл с параметрами стилизации всех элементов приложения;
- «build/» – папка для хранения итоговой версии клиентского приложения (версии не для разработки), которая создается после финальной сборки – в ней содержится минимизированный JS-код (Рисунок 3.9).

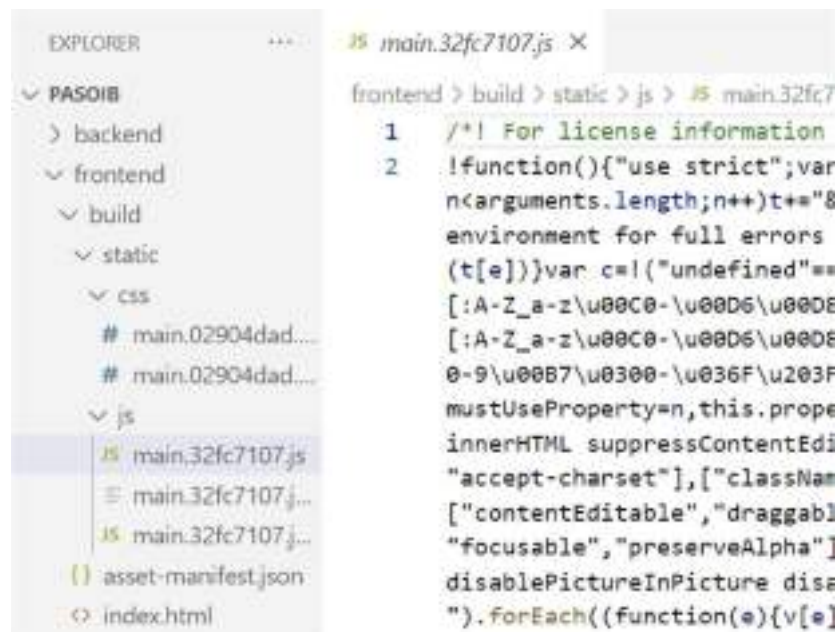


Рисунок 3.9 – Итоговая (минимизированная) версия клиентского приложения

База данных создана в соответствии со схемой базы данных, разработанной на этапе проектирования. Для создания таблиц базы данных использовались SQL-запросы (Рисунок 3.10)

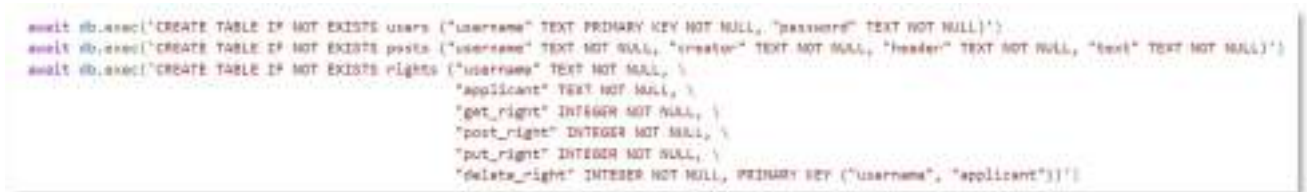


Рисунок 3.10 – SQL-запросы для создания таблиц базы данных

4 Тестирование

Для тестирования определены основные разделы и функции:

- регистрация и авторизация;
- CRUD операции (создание, чтение, изменение и удаление) над постами владельцем страницы;
- изменение настроек доступа;
- CRUD операции (создание, чтение, изменение и удаление) над постами авторизованным пользователем, имеющим доступ;
- попытка реализации CRUD операций пользователем, не имеющим доступа.

Создан пользователь «alice:12345» через веб-интерфейс приложения (Рисунки 4.1 и 4.2). После первого входа на личной странице пользователя списки постов и друзей пусты. Попытка входа с неправильным паролем невозможна (Рисунок 4.3).



Рисунок 4.1 – Регистрация нового пользователя

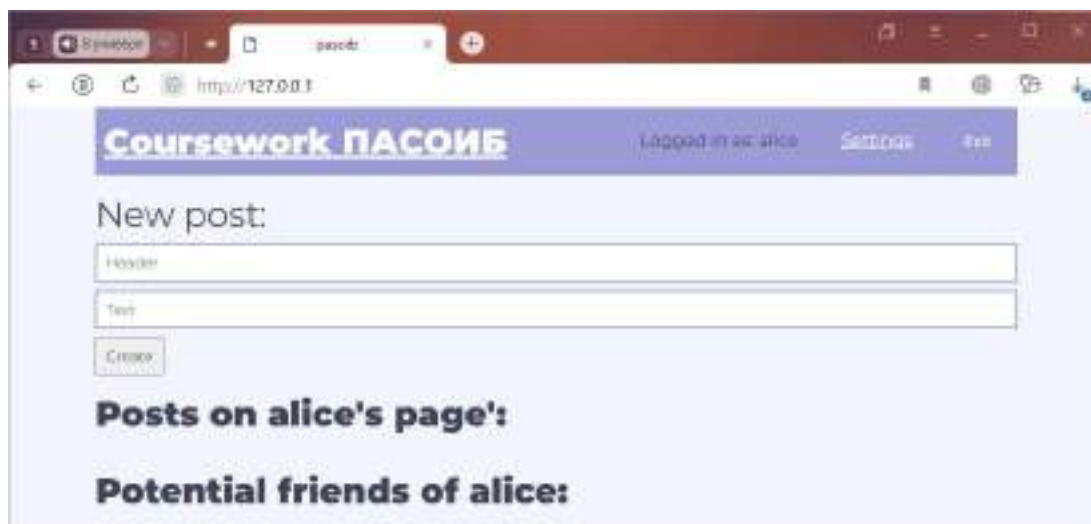


Рисунок 4.2 – Пустая страница вновь созданного пользователя

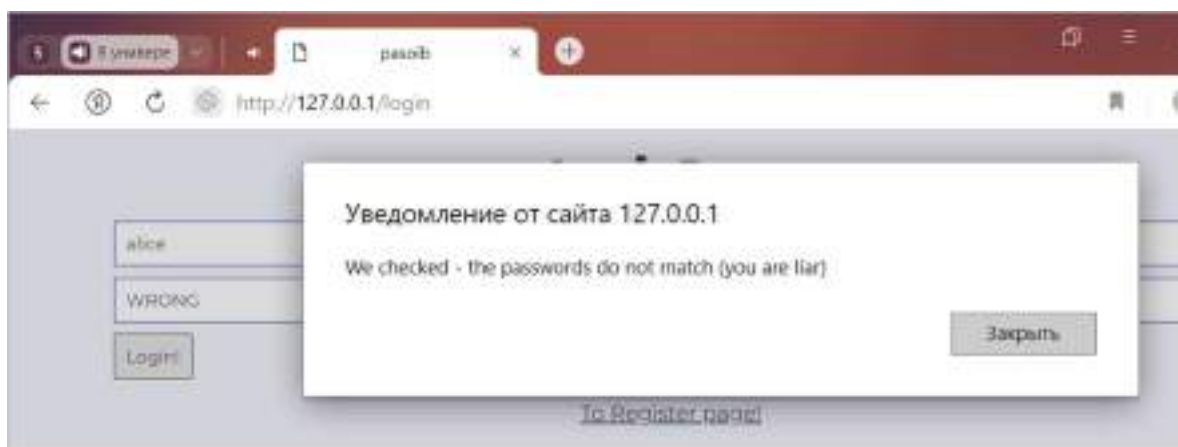


Рисунок 4.3 – Уведомление о неправильности введенного пароля

Проверена возможность создания, удаления и редактирования постов владельцем страницы. Владельцу по умолчанию доступно создание и удаление постов на своей странице: созданы несколько постов, удален пост с заголовком «Post to be deleted», после чего пост с идентификатором 3 отредактирован «4th POST», а пост с идентификатором 2 удален (Рисунок 4.4).

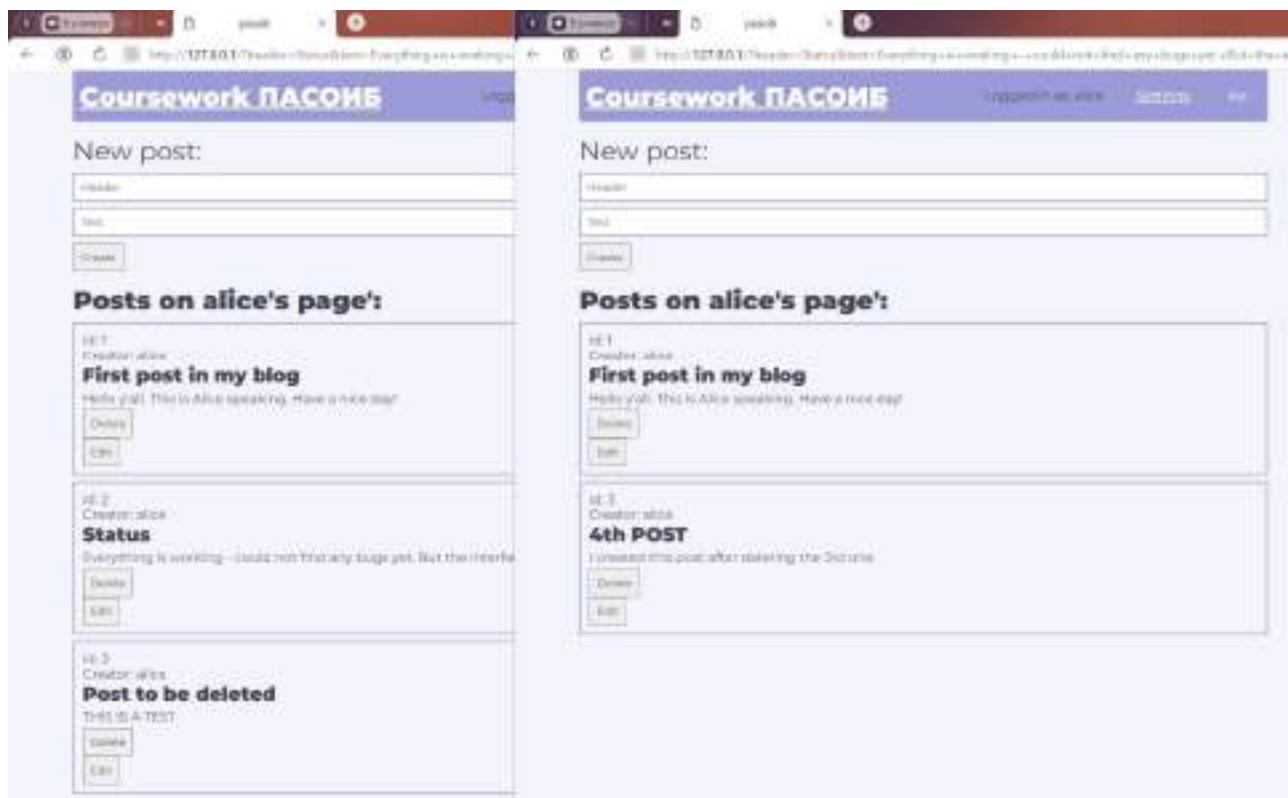


Рисунок 4.4 – Созданные посты (слева) и новое состояние списка постов после удаления поста и редактирования поста

Из предыдущих проверок уже стало понятно, что владелец успешно реализует свое право на получение постов. Еще раз убедиться в этом можно, посмотрев статистику отправки запросов при перезагрузке страницы – отправлены два запроса на получение списка постов и списка друзей (просмотр через вкладку «Network» инструментов разработчика в браузере) (Рисунок 4.5)

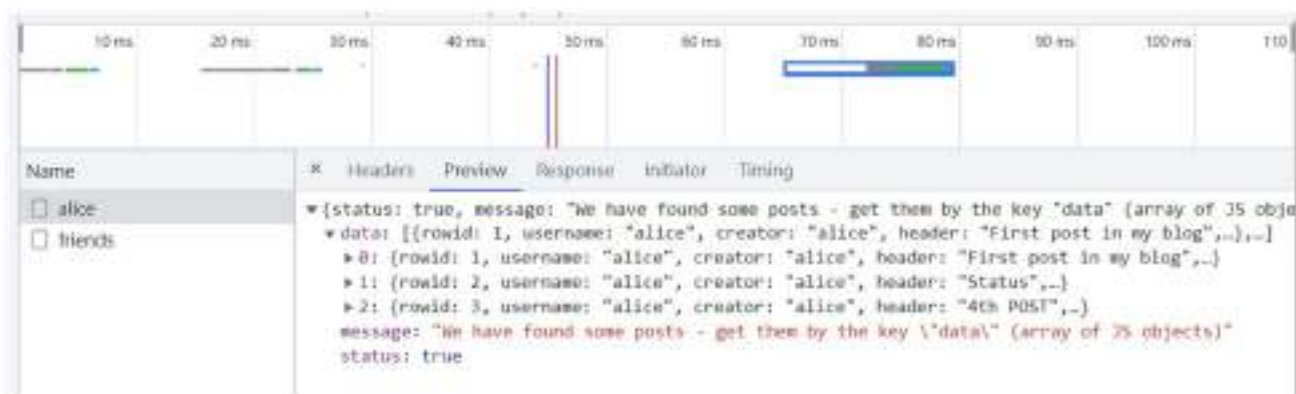


Рисунок 4.5 – Просмотр вкладки «Network» инструментов разработчика после перезагрузки страницы

Созданы правила доступа к странице пользователя «alice» (далее – Алиса) для пользователей (Рисунок 4.6):

- «bob» (далее – Боб) – разрешены все операции
- «jack» (далее – Джэк) – разрешен доступ только на чтение
- «mellory» (далее – Мэллори) – запрещено всё (равнозначно тому, как если бы правило не было создано вообще)

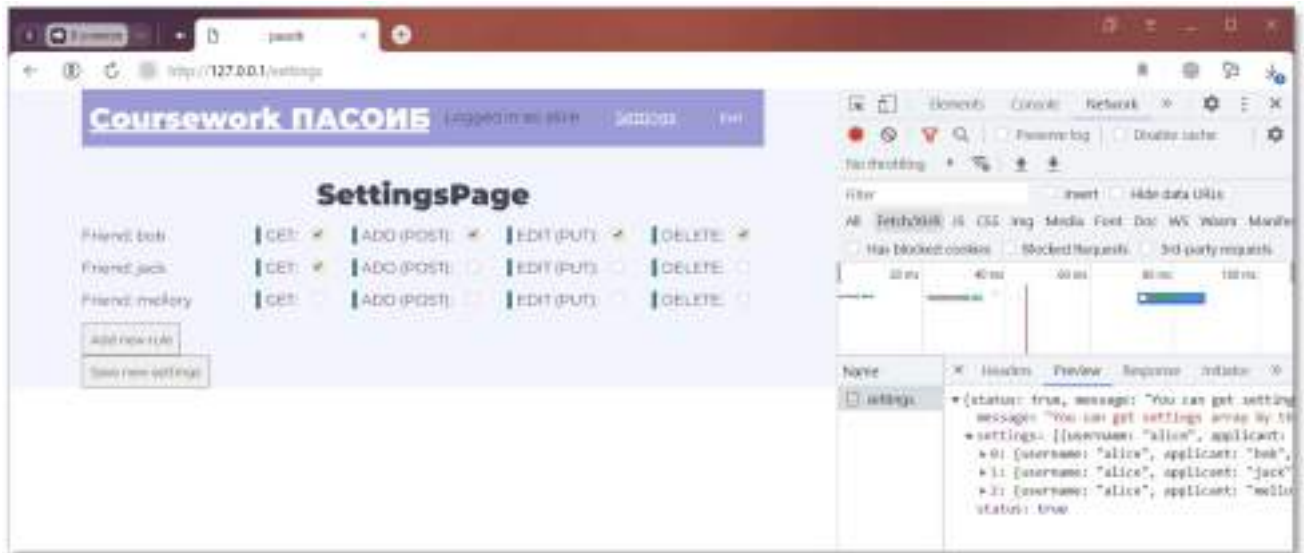


Рисунок 4.6 – Страница настроек пользователя «alice»

Осуществлен вход под учетными записями Боба, Джэка и Мэллори. Боб может создавать, удалять и изменять посты на странице Алисы (на его странице дополнительно есть ссылка для перехода на ее страницу):

- изменен пост с идентификатором 3;
- удален пост с идентификатором 1;
- создан пост с идентификатором 5;

Джеку доступен только просмотр постов Алисы, а Мэллори при попытке перехода на страницу Алисы принудительно перенаправляется на личную страницу (Рисунки 4.7 – 4.12).

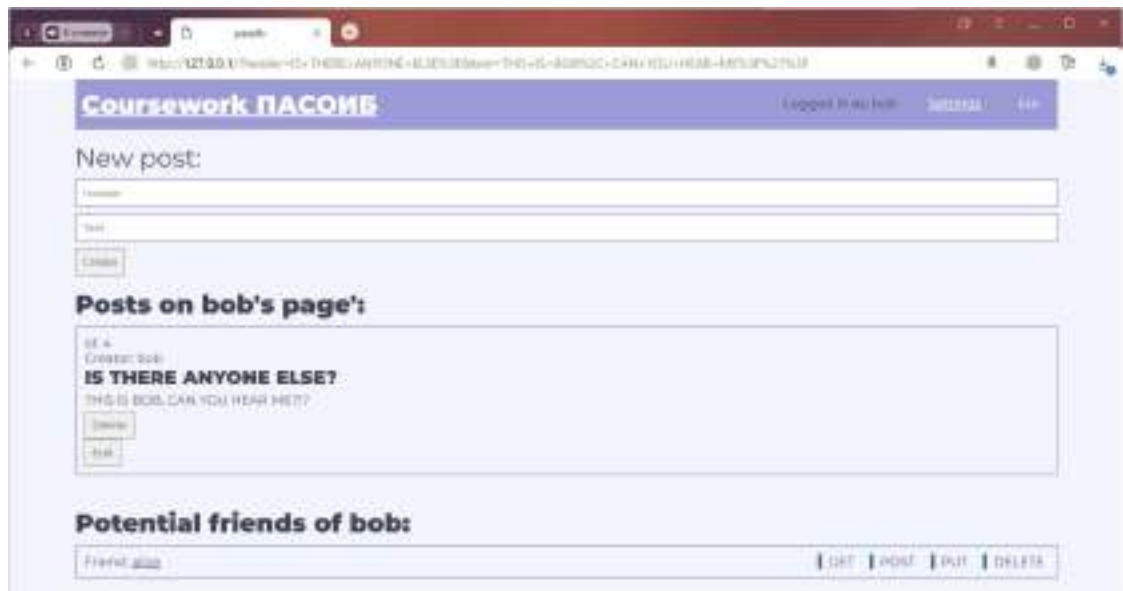


Рисунок 4.7 – Личная страница Боба

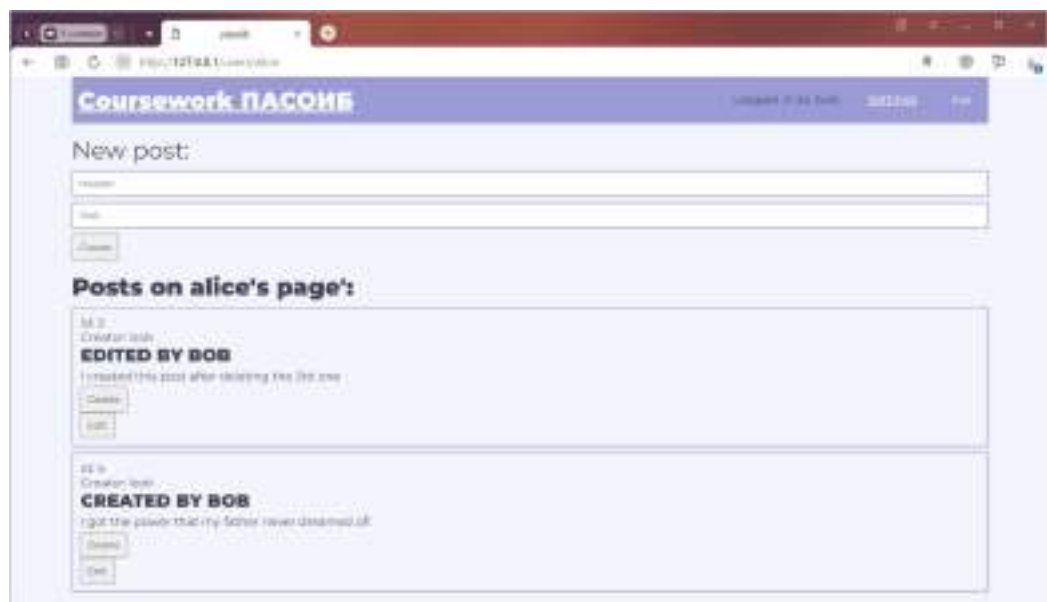


Рисунок 4.8 – Страница Алисы после внесения изменений Бобом

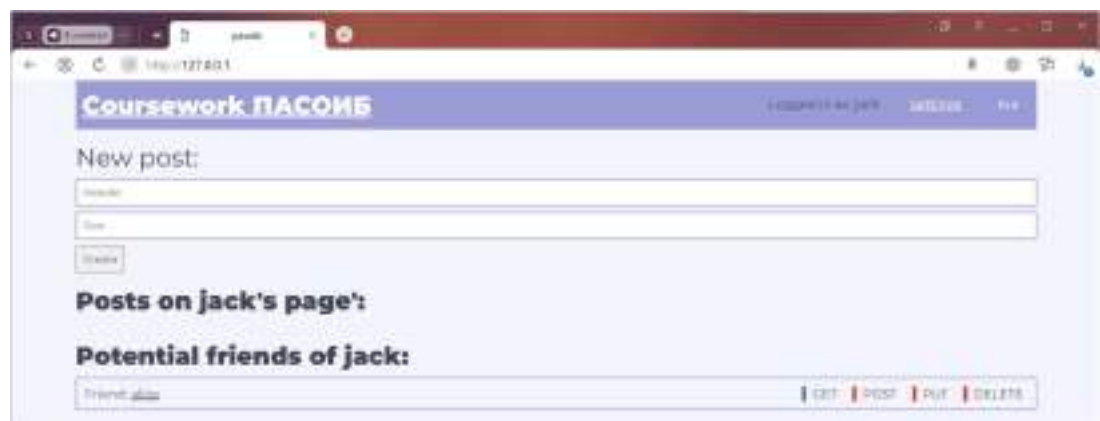


Рисунок 4.9 – Личная страница Джека

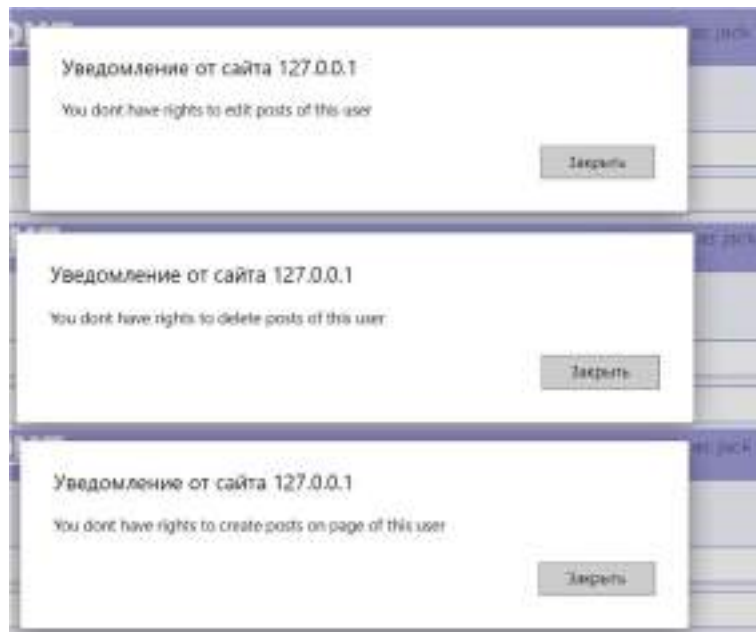


Рисунок 4.10 – Уведомления о недостаточности прав для редактирования, удаления и создания постов на странице Алисы



Рисунок 4.11 – Уведомление о невозможности просмотра страницы Алисы пользователем Мэллори

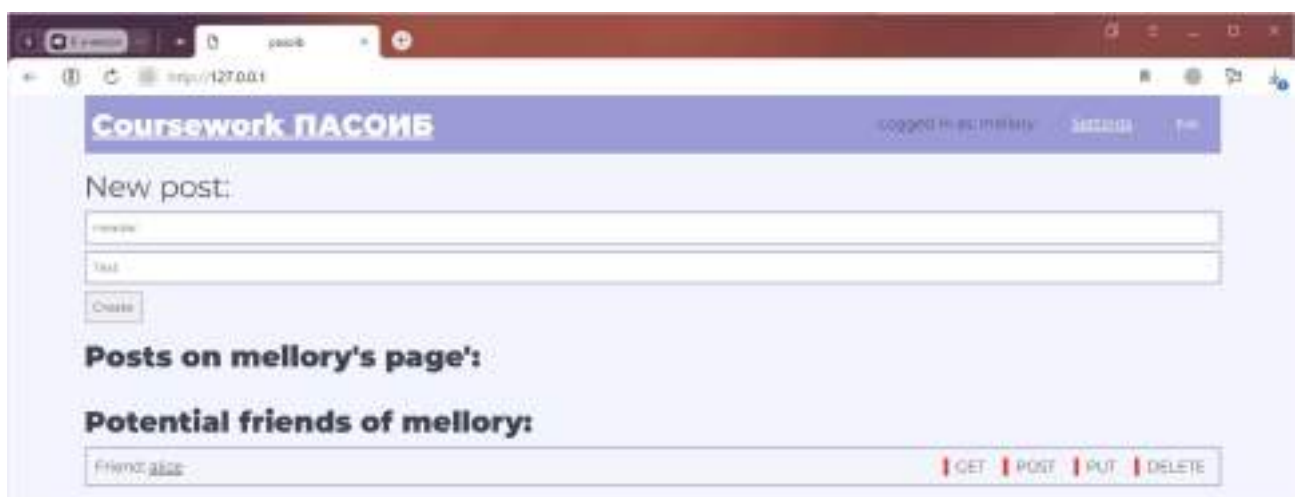


Рисунок 4.12 – Личная страница Мэллори

Заключение

В результате выполнения курсовой работы создано клиент-серверное приложение в тематике личного блога, позволяющее регистрировать новых пользователей, выполнять основные операции для работы с постами пользователей (CRUD операции) и ограничивать доступ к страницам пользователей с использованием дискреционного подхода.

В ходе работы:

- исследована предметная область;
- спроектирована структура приложения;
- составлена модель базы данных для предметной области;
- разработано приложение в соответствии с требованиями технического задания;
- разработано «Руководство администратора».

Список литературы

1. Информационная безопасность: концептуальные и методологические основы защиты информации: учебное пособие / А. А. Малюк. – М. : Горячая линия-Телеком, 2004. – 280 с.
2. Особенности реализации дискреционной модели управления доступом средствами ролевой модели / М.А. Бирюков, Е. В. Волкова, И. Б. Саенко – Санкт-Петербург, 2017.
3. Образовательный стандарт вуза ОС ТУСУР 01-2021. Работы студенческие по направлениям подготовки и специальностям технического профиля. Общие требования и правила оформления от 25.11.2021.
4. ГОСТ 19.504-79 «РУКОВОДСТВО ПРОГРАММИСТА. Требования к содержанию и оформлению».

Приложение А
(обязательное)
Руководство программиста

1 Назначение и условия применения программы

Основное назначение системы состоит в обеспечении доступа пользователям к блогу с использованием дискреционного механизма разграничения доступа. Она должна предоставлять пользователю возможность создавать (CREATE), читать (READ), изменять (UPDATE) и удалять (DELETE) посты на своей странице и на страницах других пользователей, если последние дали установили соответствующие разрешения. Так же пользователь имеет возможность редактирования матрицы доступа к своему блогу.

1.1 Требования к техническим средствам

Минимальные требования к характеристикам компонентов технического обеспечения, при которых значения временных параметров Системы должны соответствовать предъявленным требованиям, для сервера с БД:

- процессор – Ryzen 7 5800H;
- объем оперативной памяти – 32 Гб;
- дисковая подсистема – 1 Тб;
- сетевой адаптер – 100 Мбит/с.

1.2 Требования к общему программному обеспечению

Для функционирования клиентского программного обеспечения как части системы устройство должно удовлетворять системным требованиям современных браузеров и позволять выполнение JavaScript в браузере, а также иметь стабильный доступ в Интернет (после полноценного запуска системы) или в локальную сеть (на этапе разработки и тестирования).

1.2.1 Требованиям к периферийным устройствам

Устройство клиента должно, как минимум иметь:

- монитор с разрешением FullHD;
- компьютерную мышь;
- клавиатуру.

2 Характеристика программы

Система рассчитана на круглосуточный бесперебойный режим работы.

Приложение реализовано в виде клиент-серверного приложения и представляет собой простой блог – у каждого зарегистрированного пользователя есть личная страница, где он может оставлять записи и настраивать права доступа других пользователей к своему блогу.

В приложении реализованы следующие функции:

- Регистрация и авторизация пользователей.
- Дискреционное разграничение доступа к личной странице каждого пользователя: право на чтение (GET-запрос в HTTP), право на запись (POST-запрос), право на изменение (PUT-запрос) и удаление (DELETE-запрос).
- Просмотр, создание, изменение и удаление постов на страницах пользователей
- Основные страницы, которые представлены в клиентском приложении:
- Страница регистрации и авторизации.
- Личная страница пользователя с постами.
- Страница настроек прав доступа.

Веб-сервер обеспечивает взаимодействие при помощи вызова соответствующих API-функций.

Для разработки использованы следующие технологии:

- База данных – SQLite (пакеты `sqlite` и `sqlite3` из менеджера пакетов Node.js);
- Серверная часть – фреймворк `Express.js`;
- Клиентская часть – библиотека `React.js` (с использованием `JSX` и `CSS`).

Таким образом во всех подсистемах приложения для разработки применяется язык программирования JavaScript.

С подробным описанием структуры проекта можно ознакомиться с помощью Пояснительной записки.

3 Обращение к программе

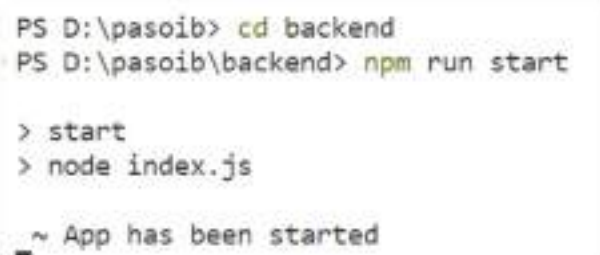
3.1 Загрузка и запуск программы

Следует убедиться, что Node.js и NPM успешно установлены и работают – сделать это можно командой для проверки версий («`npm --version`» и «`node --version`»).

Для запуска серверной части программы необходимо загрузить файлы проекта в директорию, откуда будет осуществляться запуск.

Перейти в папку «`backend`» и установить (при необходимости) недостающие пакеты («`npm install`») – список зависимостей содержится в файле «`package.json`».

Запустить сервер командой «`npm run start`» (Рисунок 3.1) для режима «`production`» или «`npm run dev`» для режима разработки.



```
PS D:\pasoib> cd backend
PS D:\pasoib\backend> npm run start

> start
> node index.js

~ App has been started
```

Рисунок 3.1 – Запуск серверной части приложения

Для запуска клиентской части приложения нужно открыть браузер и ввести в строке поиска «site-name.com/», где site-name.com – либо доменное имя сайта, либо IP-адрес. Для разработки используется 3000 порт, для production-сборки используется порт по умолчанию для протокола http – 80.

3.2 Выполнение и завершение программы

Для остановки работы системы (сервера) можно воспользоваться несколькими способами:

- ввести комбинацию клавиш «Ctrl+C» и подтвердить остановку пакета через терминал;
- вручную остановить процесс.

Для остановки клиентской части приложения достаточно закрыть вкладку браузера. Для удаления всей информации о работе с сайтом из браузера пользователя следует очистить файлы cookie и хранилище localStorage.

4 Входные и выходные данные

Входные данные, которые должны подаваться на вход **сервера** системы, показаны в таблице 4.1.

Таблица 4.1 – Конечные точки серверной части приложения (API-endpoints)

Тип HTTP-запроса	Адрес	Назначение
GET	«/api /posts /:username»	Получение постов пользователя
POST	«/api /posts /:username»	Создание нового поста
PUT	«/api /posts /:id»	Изменение конкретного поста
DELETE	«/api /posts /:id»	Удаление конкретного поста
GET	«/api /settings»	Получение настроек доступа пользователя (матрицы доступа)
POST	«/api /settings»	Изменение матрицы доступа
POST	«/api /settings /check»	Получение информации о правах доступа из матрицы доступа
POST	«/api /settings /add»	Добавление нового правила (нового субъекта) в матрицу доступа
GET	«/api /friends»	Получение списка всех пользователей, разрешивших доступ на просмотр их страницы

Входные данные, которые могут подаваться на вход клиентской части

Системы:

- полный bundle скриптов на языке JS вместе с CSS (в минимизированном виде);
- ответы сервера на API-запросы в формате JSON.

5 Сообщения

В случае возникновения любых сообщений об ошибке на сервере следует убедиться в наличии требуемых установленных пакетов, в целостности базы данных и перезапустить сервер.

Сообщения на клиентском устройстве:

- «Создан новый пользователь»;
- «Пользователь авторизован»;
- «Новый пост создан»;
- «Пост безвозвратно удален»;
- «Содержимое поста изменено»;
- «Нет доступа на просмотр – перенаправление на личную страницу...»;
- «Нет прав для удаления постов этого пользователя»;
- «Нет прав для создания постов на странице этого пользователя»;
- «Нет прав для изменения постов на странице этого пользователя»;
- «Вы вышли из учетной записи»

Действия на клиентской части приложения при получении сообщений не требуются.