МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ

Государственное бюджетное профессиональное образовательное учреждение Московской области «Ногинский колледж»

Курсовой проект

по МДК.09.01 Проектирование и разработка веб-приложений

ПМ.09 Проектирование, разработка и оптимизация веб-приложений

Тема:

Использование фреймворка NodeJS в разработке веб-приложения «БЛОГ-ПЛАТФОРМА»

Разработчик:

студент группы 3ИСПР

Алексеев А.А.

(подпись)

Оценка защиты курсового проекта

Руководитель проекта:

преподаватель

Степанов С.О.

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

(подпись)

« »

Дата защиты

« » 2025г.

Ногинск, 2025

Содержание

[Введение 3](#_Toc199534403)

[1. Описание предметной области 5](#_Toc199534404)

[1.1 Краткая характеристика веб-приложения 5](#_Toc199534405)

[1.2 Описание программных инструментов и средств разработки 6](#_Toc199534406)

[1.2.1 Язык программирования «javascript» 6](#_Toc199534407)

[1.2.2 Фреймворк «express.js» 7](#_Toc199534408)

[1.2.3 Серверная часть 8](#_Toc199534409)

[1.2.4 Среда разработки 9](#_Toc199534410)

[2. Практическая часть 11](#_Toc199534411)

[2.1 Проектирование веб-приложения 11](#_Toc199534412)

[2.2 Подготовка к работе 13](#_Toc199534413)

[2.3 Разработка веб-приложения 15](#_Toc199534414)

[2.3.1 Моделирование данных 15](#_Toc199534415)

[2.3.2 Создание маршрутов и контроллеров 16](#_Toc199534416)

[2.3.3 Шаблоны ejs и вёрстка страниц 18](#_Toc199534417)

[2.3.4 Обработка форм и загрузка изображений 19](#_Toc199534418)

[2.3.5 Интеграция wysiwyg-редактора 21](#_Toc199534419)

[2.3.6 Тема и персонализация 21](#_Toc199534420)

[2.3.7 Аутентификация и авторизация 21](#_Toc199534421)

[2.3.8 Профиль и настройки 22](#_Toc199534422)

[2.3.9 Seed-данные 22](#_Toc199534423)

[2.4 Размещение веб-приложения на хостинге 24](#_Toc199534424)

[Заключение 26](#_Toc199534425)

[Список использованной литературы 28](#_Toc199534426)

# ВВЕДЕНИЕ

Бурное развитие информационных технологий и широкое распространение интернета способствуют увеличению популярности различных веб-сервисов, особенно блог-платформ. Подобные ресурсы дают возможность пользователям публиковать авторские материалы, высказывать собственные взгляды и обмениваться информацией с широкой аудиторией.

Для создания современных, стабильных и масштабируемых веб-приложений всё чаще применяются мощные инструменты разработки, такие как язык программирования JavaScript и фреймворк Express.js.

**Цель курсового проекта** заключается в проектировании и разработке веб-приложения — блог-платформы с применением языка JavaScript и фреймворка Express.js.

Для реализации поставленной цели требуется выполнить следующие задачи:

- изучить принципы работы с фреймворком Express.js;

- разработать архитектуру веб-приложения;

- создать пользовательский интерфейс и серверную часть системы;

- реализовать ключевые функции блог-платформы: регистрацию, создание и редактирование записей, комментирование публикаций;

- провести тестирование и проанализировать работоспособность готового приложения.

Объектом курсовой работы выступает процесс разработки веб-приложений с использованием фреймворка Express.js.

Предметом курсовой работы является разработка и реализация функциональной блог-платформы, использующей возможности фреймворка Express.js — минималистичного и гибкого веб-фреймворка для Node.js. Основное внимание в проекте уделяется созданию серверной части приложения, обеспечивающей маршрутизацию, обработку HTTP-запросов, взаимодействие с базой данных, а также реализацию ключевых функций блог-платформы: регистрация и аутентификация пользователей, создание, редактирование и удаление постов, отображение ленты публикаций, а также комментирование и взаимодействие с контентом. Выбор Express.js обусловлен его простотой, расширяемостью и активной экосистемой, что делает его эффективным инструментом для построения полноценных веб-приложений.

Актуальность темыисследования определяется высоким спросом на удобные и функциональные веб-приложения для информационного взаимодействия в интернете. Express.js, как один из самых востребованных фреймворков для Python, обеспечивает быстрое создание, высокую безопасность и масштабируемость решений, что делает его эффективным инструментом для реализации современных блог-сервисов.

# 1. Описание предметной области

**1.1 КРАТКАЯ ХАРАКТЕРИСТИКА ВЕБ-ПРИЛОЖЕНИЯ**

Веб-приложение — это компьютерная программа, которая запускается прямо в веб-браузере. Для работы ей не требуется установка на компьютер, поэтому доступ к веб-приложению можно получить с любого устройства, имеющего веб-браузер, при условии, что оно подключено к сети Интернет.

Основная цель веб-приложения — предоставление пользователю определённой функциональности или сервиса, который позволяет решать определённые проблемы или упрощает выполнение задачи.

Основные характеристики веб-приложения:

- Интерактивность. Возможность взаимодействия пользователя с приложением.

- Доступность. Веб-приложение может быть запущено на любом устройстве с доступом к интернету.

- Масштабируемость. Способность приложения эффективно работать при большом числе пользователей.

- Обновляемость. Возможность внесения изменений и обновления приложения без необходимости переустановки на устройствах пользователей.

- Безопасность. Защита приложения от несанкционированного доступа и атак.

Веб-приложения широко используются в различных сферах, включая медицину, банковское дело, электронную коммерцию и многие другие.

Блог-платформа — это специальная платформа или система, предназначенная для ведения блогов. Она предоставляет пользователям инструменты для создания, публикации и управления контентом.

Блог-платформы могут быть различными по функционалу и сложности, от простых платформ для личных блогов до сложных систем для ведения корпоративных блогов с множеством авторов и аудитории.

Некоторые основные функции блог-платформ:

- Создание и редактирование контента. Удобные редакторы позволяют писать статьи, добавлять изображения, видео и другие мультимедийные элементы.

- Управление публикациями. Возможность планировать публикации, управлять черновиками и архивом статей.

- Интерактивные элементы. Комментарии, лайки и другие элементы взаимодействия с аудиторией.

- Аналитика. Инструменты для отслеживания посещаемости, вовлечённости и других метрик.

- SEO-оптимизация. Возможности для улучшения видимости контента в поисковых системах.

**1.2 Описание программных инструментов и средств разработки**

### 1.2.1 Язык программирования «JavaScript»

**JavaScript** — это **высокоуровневый, интерпретируемый язык программирования**, изначально разработанный для добавления интерактивности на веб-страницы. Со временем он превратился в полноценный универсальный язык, применяемый как на стороне клиента (в браузере), так и на стороне сервера (с помощью Node.js).

Основные особенности JavaScript:

* Динамическая типизация: переменные не требуют указания типа.
* Объектно-ориентированность: поддержка объектов, прототипного наследования.
* Функциональное программирование: функции являются объектами первого класса.
* Асинхронность: поддержка колбэков, промисов и async/await.
* Интерпретируемость: код выполняется сразу, без необходимости компиляции.
* Кросс-платформенность: работает во всех современных браузерах и на серверах через Node.js.

Где используется JavaScript:

* Веб-разработка (frontend): для создания интерактивных пользовательских интерфейсов.
* Серверная разработка (backend): с использованием среды Node.js.
* Мобильные приложения: через фреймворки React Native, Ionic и другие.
* Разработка настольных приложений: с помощью Electron.
* Игры и графика: WebGL, Three.js.

### 1.2.2 Фреймворк «Express.js»

Фреймворк (от англ. framework — «каркас», «основа») — это набор готовых инструментов, библиотек и правил, которые помогают разработчику быстрее и проще создавать программы или веб-приложения.

Он обеспечивает структуру и шаблон для проекта, избавляя программиста от необходимости писать рутинный код с нуля.

Express.js — это минималистичный и гибкий веб-фреймворк для Node.js, предназначенный для создания серверных веб-приложений и API.

Благодаря своей простоте, Express.js позволяет разработчику сосредоточиться на бизнес-логике приложения, не тратя время на написание низкоуровневого серверного кода. Он легко расширяется с помощью внешних модулей и обладает отличной совместимостью с другими библиотеками и инструментами в экосистеме Node.js.

Это делает Express.js одним из самых популярных выборов для создания:

* Одностраничных приложений (SPA).
* RESTful API.
* Серверной части для мобильных и веб-приложений.

Он предоставляет набор удобных инструментов для обработки HTTP-запросов, маршрутизации, работы с шаблонами, подключаемыми модулями (middleware) и другими функциями веб-сервера.

Основные особенности:

* Простота в использовании и изучении.
* Гибкая система маршрутов (GET, POST, PUT, DELETE и др.).
* Поддержка middleware — промежуточных функций между запросом и ответом.
* Легкая интеграция с базами данных и другими библиотеками.
* Часто используется для создания REST API и backend'а в MERN/MEAN стеке.

### 1.2.3 Серверная часть

Node.js — это платформа с открытым исходным кодом для работы с языком JavaScript. Она позволяет писать серверный код для веб-приложений и динамических веб-страниц, а также программ командной строки.

Node.js нужна для решения следующих задач:

- Разработка серверных приложений. Среда позволяет разрабатывать веб-серверы, способные работать с миллионами одновременных соединений.

- Асинхронное программирование. Код выполняется асинхронно, благодаря чему ресурсы сервера расходуются грамотнее, а приложение может обрабатывать множество запросов одновременно.

- Работа с большими объёмами данных. За счёт того, что Node.js хорошо справляется с высокой нагрузкой, её часто применяют для создания приложений, где нужна обработка больших объёмов информации.

- Создание микросервисов. Архитектура среды позволяет разрабатывать небольшие изолированные сервисы, которые легко масштабировать и изменять.

- Разработка сетевых приложений. К примеру, Node.js подходит для разработки чатов, игровых серверов и RESTful API, в которых нужна высокая скорость обработки запросов.

- Интеграция со сторонними инструментами через API. С помощью Node.js настраивают взаимодействие сайта с базами данных, фреймворками и библиотеками.

MySQL — это реляционная система управления базами данных (СУБД), которая распространяется как свободное программное обеспечение. Она позволяет хранить, организовывать большие объёмы данных и манипулировать ими.

MySQL нужна для решения следующих задач:

- Хранение данных. Позволяет сохранять данные в структурированном виде и даёт возможность быстро получать к ним доступ.

- Управление данными. MySQL даёт возможности для управления данными, например, создание, изменение и удаление таблиц, индексов и отношений между ними.

- Обеспечение безопасности данных. У системы продвинутая система контроля прав для разных пользователей, есть шифрование и верификация.

- Возможность контролировать версии базы данных. Можно делать бэкапы, обновлять базу или откатывать назад.

Command Prompt — это текстовый интерфейс для взаимодействия с операционной системой, который позволяет запускать различные команды и выполнять задачи напрямую через текстовый интерфейс, минуя графический пользовательский интерфейс (GUI).

### 1.2.4 Среда разработки

Visual Studio Code — это бесплатный, кроссплатформенный редактор исходного кода, разработанный компанией Microsoft. Он поддерживает работу на операционных системах Windows, Linux и macOS, сочетая в себе легковесность, гибкость и широкие функциональные возможности, которые делают его одним из самых популярных инструментов среди разработчиков по всему миру.

Основные особенности и преимущества VS Code:

- Поддержка множества языков программирования

Включает в себя поддержку таких языков, как JavaScript, TypeScript, Python, Java, C++, PHP, HTML, CSS, Go, C#, и многих других. Благодаря расширениям функциональность можно легко расширить.

- Интеллектуальные подсказки и автодополнение (IntelliSense)

VS Code автоматически предлагает завершение кода, информацию о функциях, переменных и типах, что значительно ускоряет написание кода и уменьшает количество ошибок.

- Встроенный терминал

Позволяет запускать команды прямо из редактора, без переключения на внешний терминал. Это удобно для запуска скриптов, команд Git или установки пакетов.

- Интеграция с системой контроля версий Git и GitHub

- Отладка приложений

Редактор поддерживает встроенную отладку, включая установку точек останова, пошаговое выполнение кода, просмотр переменных и стеков вызовов. Поддерживается множество языков и фреймворков через расширения.

- Live Share — совместная работа в реальном времени

Разработчики могут делиться своим рабочим пространством с коллегами, редактировать и отлаживать код совместно, как будто работают за одним компьютером.

- Гибкая настройка интерфейса

- Работа с удалёнными серверами и контейнерами

# 2. Практическая часть

## 2.1 Проектирование веб-приложения

Процесс создания веб-приложения не начинается с написания кода, а с детального анализа требований и проектирования системы. Это фундамент, на котором будет строиться всё приложение, и от которого напрямую зависит удобство, надёжность и гибкость конечного продукта. Проектирование позволяет заранее продумать логику, структуру данных, взаимодействие компонентов и пользовательский интерфейс, избежать ошибок на поздних этапах и упростить дальнейшее развитие.

Постановка цели и задачи

Цель проекта — разработать современную, функциональную и визуально привлекательную блог-платформу, где каждый пользователь может:

* Регистрация/вход — создать аккаунт и аутентифицироваться;
* Личный кабинет — увидеть список своих записей, настроить тему и цвета оформления;
* CRUD-посты — создавать, редактировать, удалять записи с помощью WYSIWYG-редактора;
* Загрузка медиа — прикреплять главное изображение и файлы к постам;
* Просмотр — открывать отдельную страницу записи и видеть текст, изображения и вложения.

Задачи:

* Реализовать безопасную аутентификацию (Passport.js + bcrypt).
* Настроить гибкую работу с файлами (Multer + загрузка на сервер).
* Обеспечить удобный интерфейс редактирования через Quill.js.
* Позволить пользователю выбирать светлую/тёмную тему и цвета фон/текст постов.
* Спланировать маршруты и шаблоны для «Моих постов» и «Всех постов».

Выбор архитектуры

Для разделения ответственности и упрощения поддержки выбрана трёхуровневая архитектура:

1. Уровень представления

* EJS (+ express-ejs-layouts) генерирует HTML на сервере;
* SCSS + Bootstrap отвечает за стили и адаптивность.

1. Уровень бизнес-логики

* Express.js обрабатывает маршруты, middleware и логику контроллеров;
* Контроллеры authController, postsController, profileController.

1. Уровень данных

* MySQL хранит пользователей, посты и файлы;
* Sequelize ORM описывает модели (User, Post, File) и их связи.

Планирование страниц и интерфейсов

При проектировании определены ключевые представления:

* Главная страница:

если не залогинен — перенаправление на форму входа;

при входе — «Мои посты» (список заголовков с кнопками редактирования/удаления).

* Все посты (/posts/all):

общий список без кнопок управления, только просмотр.

* Создание/редактирование поста (/posts/new, /posts/:id/edit):

поля «Заголовок», Quill-редактор для текста, выбор главного изображения и вложений.

* Просмотр поста (/posts/:id):

главное изображение в карточке, отрендеренный HTML-контент, список вложений-ссылок.

* Профиль (/profile):

имя и email пользователя, общее число записей, ссылка на «Настройки».

* Настройки (/profile/settings):

выбор theme (light/dark), postBg и postTextColor (инпуты color).

Каждый экран заполняется данными динамически, что обеспечивает гибкость и простоту поддержки.

## 2.2 Подготовка к работе

Перед тем как приступить к написанию кода нашей блог-платформы, необходимо подготовить и настроить среду разработки. Сначала устанавливаем серверную платформу Node.js вместе с менеджером пакетов npm — они будут основой для запуска приложения и управления зависимостями. После установки важно убедиться, что в терминале корректно отображаются версии команд node -v и npm -v. Следующим шагом позаботимся о базе данных: устанавливаем MySQL-сервер, создаём новую базу (например, blog\_platform) и назначаем для неё пользователя с паролем. Это позволит нам хранить пользователей, записи и файлы нашего блога.

Далее клонируем проект на локальную машину через Git и переходим в его папку. Команда npm install подхватывает все необходимые библиотеки — Express для веб-сервера, Sequelize для работы с БД, Passport и bcrypt для аутентификации, Multer для загрузки файлов, Quill для редактора и прочее. Когда зависимости установлены, создаём файл окружения: копируем .env.example в новый .env и вносим туда параметры подключения к базе, секретную фразу для сессий и номер порта. Это позволяет конфиденциально хранить пароли и настройки, не внося их в само приложение.

Перед первым запуском собираем стили: запускаем скрипт npm run build:scss, который преобразует SCSS-файлы в единый CSS. Затем наполняем базу тестовыми данными — для этого предусмотрены сидеры. Достаточно выполнить npx sequelize db:seed:all, и в таблицы Users и Posts загрузятся несколько примеров пользователей и записей, что сразу даст понимание, как выглядит фронтенд при реальной нагрузке.

И, наконец, запускаем сервер. В режиме разработки используем npm run dev: благодаря nodemon при каждом изменении кода сервер перезапустится автоматически. По умолчанию приложение стартует по адресу [https://localhost:3000](https://localhost:3000/) (например, придётся принять самоподписанный сертификат для HTTPS). Если же нужен статичный запуск без перезагрузок, достаточно команды npm start, которая просто выполнит node src/app.js.

Теперь всё готово: редактор Quill, система аутентификации, загрузка файлов и стили на месте, а база с тестовыми записями позволяет сразу проверять функционал. Можно приступать к реализации и отладке бизнес-логики блог-платформы.

## 2.3 Разработка веб-приложения

Разработка нашей блог-платформы представляет собой последовательное построение всего веб-приложения от основы до деталей. Сначала мы формируем модели данных, которые задают, какие сущности и в каком виде будут храниться в базе. Затем проектируем маршруты и контроллеры — точки входа для клиентских запросов и места, где реализуется основная логика работы: создание, чтение, обновление и удаление записей. После этого создаём шаблоны для рендеринга страниц и настраиваем стили, обеспечивая адаптивность и единый визуальный язык. Параллельно добавляем поддержку загрузки изображений и других файлов, интегрируем WYSIWYG-редактор для удобного ввода контента. Не забываем про безопасность: настраиваем аутентификацию, защищаем маршруты, следим за корректным выводом HTML и надёжным хешированием паролей. В результате получается законченный продукт, где пользователь может зарегистрироваться, создать запись, добавить к ней картинку и файлы, посмотреть свои и чужие посты, настроить тему оформления и цвет текста — и всё это работает как единое целое.

2.3.1 Моделирование данных

Начинаем с проектирования структуры базы, ведь именно на ней держится вся наша логика. В папке models/ каждая сущность соответствует отдельному файлу-модели. Модель User описывает, как и где хранить информацию о пользователях — логин, пароль, электронную почту и настройки темы. Модель Post содержит заголовок и HTML-содержимое, связанное с пользователем. Наконец, модель File отвечает за описание загруженных файлов: путь на сервере, MIME-тип и флаг, указывающий, является ли файл главным изображением поста. Благодаря Sequelize ORM мы задаём эти параметры декларативно: достаточно описать поля, их типы и связи, и фреймворк автоматически создаст или синхронизирует таблицы в MySQL, обеспечив корректную работу ссылок и целостность данных.:

- User: username, email, password, theme, postBg, postTextColor

- Post: title, content, user\_id → User

- File: filename, path, mime\_type, post\_id → Post, user\_id → User, isMain

2.3.2 Создание маршрутов и контроллеров

После того как модель готова, мы переходим к описанию URL-маршрутов в папке routes/ (Рис. 1). Каждый маршрут сочетает в себе путь и метод HTTP, а также middleware для проверки авторизации. Например, маршрут GET /posts выводит список собственных записей, а GET /posts/all — общий список всех постов. Для обработки запросов создаём контроллеры в controllers/: в них сосредоточена бизнес-логика — чтение из базы, валидация, перелинковка данных из нескольких таблиц и подготовка объекта для шаблона (Рис. 2). Такой подход позволяет чётко отделить обработку HTTP-запросов от их визуального представления и работы с данными.

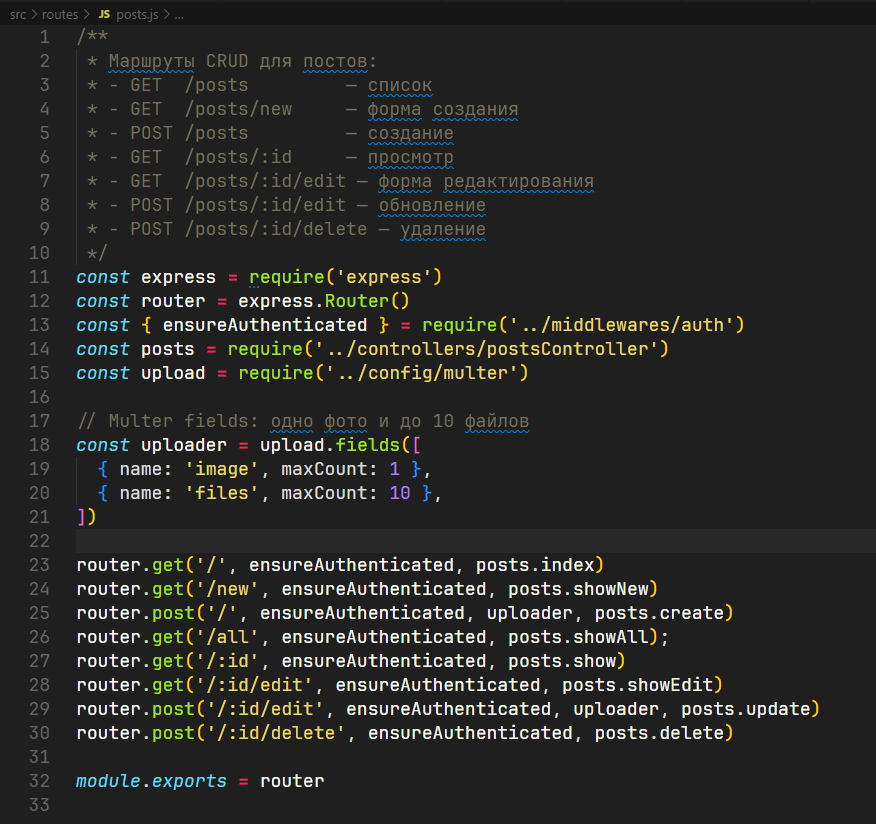


Рисунок 1. Пример маршрута posts.js.

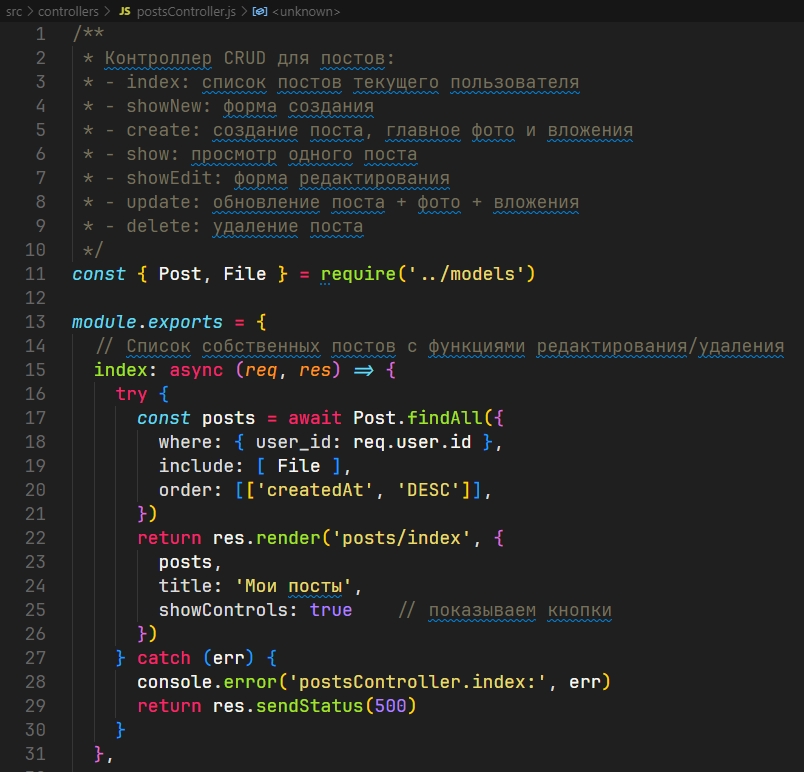


Рисунок 2. Пример контроллера postController.js

2.3.3 Шаблоны EJS и вёрстка страниц

Визуальная часть оформлена в EJS-шаблонах: у нас есть общий layout с хедером и футером, куда «встраиваются» конкретные представления страниц. В views/posts/index.ejs мы перебираем список постов, выводим заголовок, а в зависимости от флага showControls показываем или скрываем кнопки «Редактировать» и «Удалить» (Рис. 3 и 4). Для формирования единого стиля используется SCSS и Bootstrap — это позволяет быстро настроить сетку, кнопки и формы, а также легко менять тему оформления через переменные.

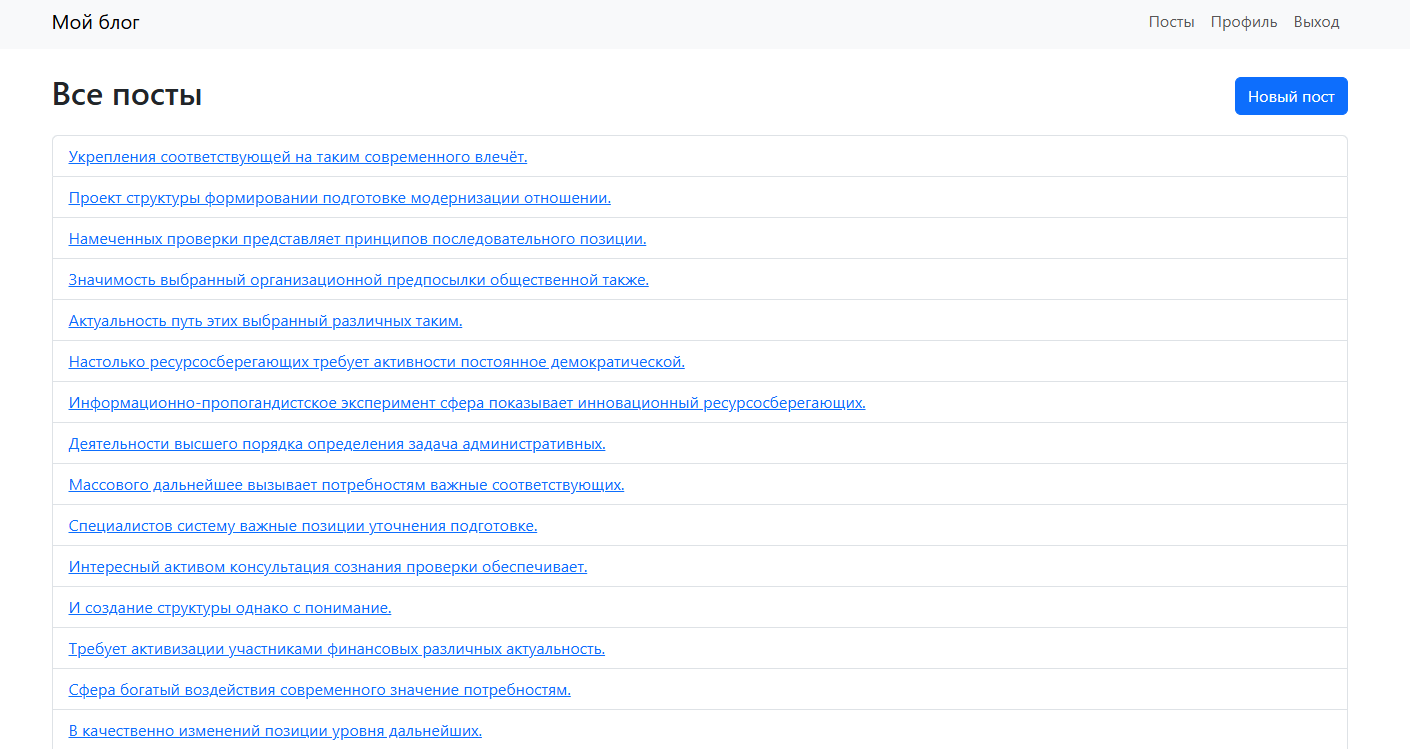


Рисунок 3. Все посты (нет кнопок редактирования).

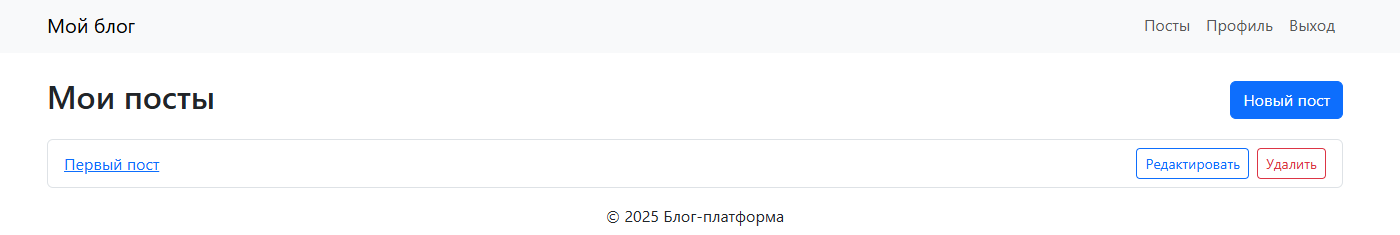


Рисунок 4. Мои посты (есть кнопки редактирования).

2.3.4 Обработка форм и загрузка изображений

Для парсинга данных из форм применяется стандартный middleware Express (express.urlencoded()), а за загрузку файлов отвечает Multer (Рис.5), настроенный на сохранение в папку public/uploads с уникальными именами. В шаблонах мы используем форму <form enctype="multipart/form-data">, где есть поле для основного изображения (name="image") и для дополнительных файлов (name="files" multiple) (Рис. 6). В контроллерах после сохранения поста мы проверяем, какие файлы пришли, и создаём соответствующие записи в таблице File, чётко разделяя главное изображение и вложения.

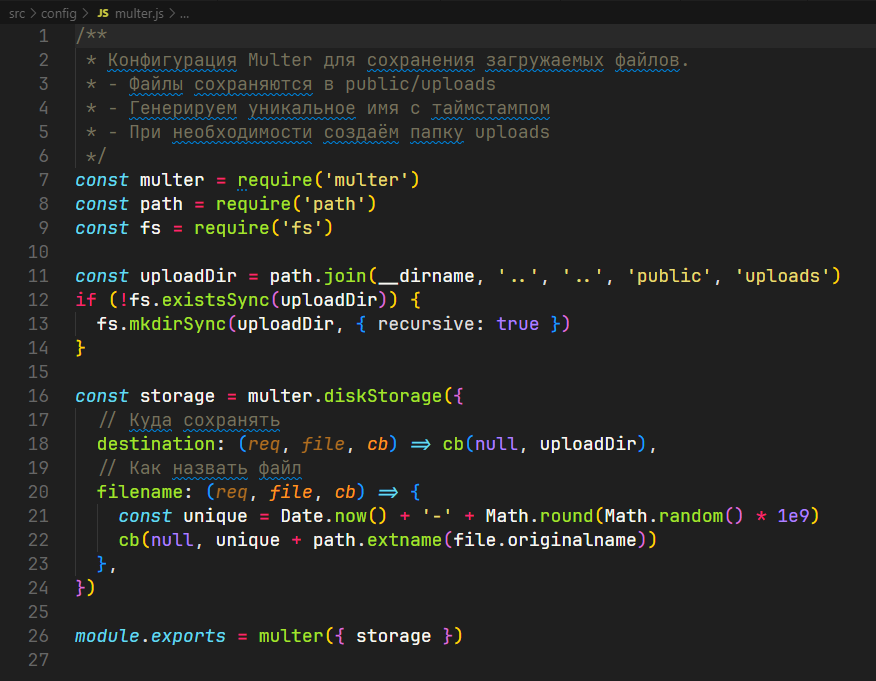


Рисунок 5. Реализация сохранения файлов на сервер.

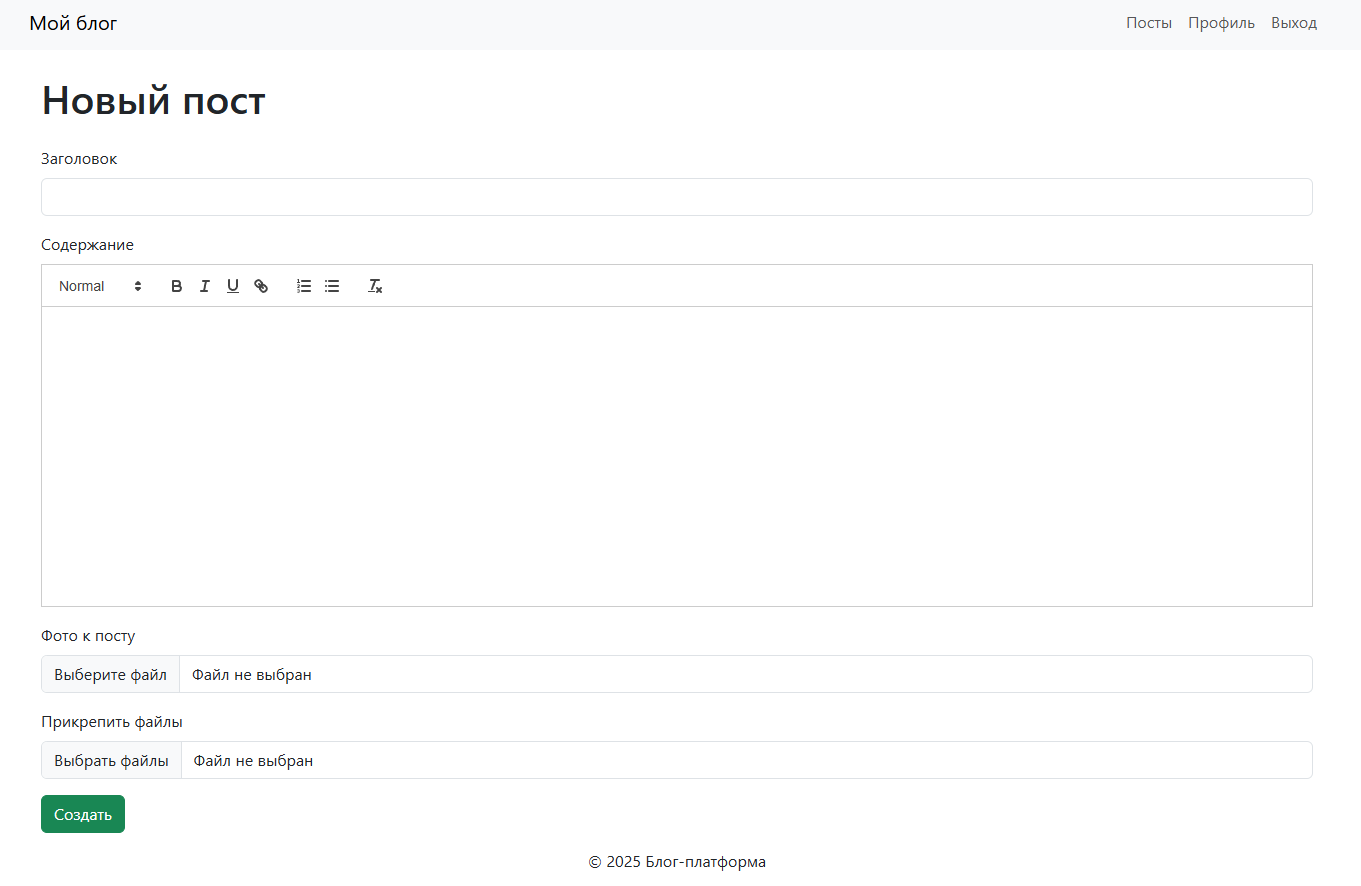


Рисунок 6. Форма создания/редактирования (new.ejs, edit.ejs).

2.3.5 Интеграция WYSIWYG-редактора

Для удобства пользователей в формах создания и редактирования поста используется Quill.js. По подключению его скрипта и стилей мы инициализируем редактор на контейнере #editor, а при отправке формы автоматически копируем сгенерированный HTML в скрытое поле content. Такой подход даёт гибкость форматирования текста без прямого ввода HTML пользователем.

* + 1. Тема и персонализация

Пользователь может выбрать светлую/тёмную тему и цвета фона/текста постов в настройках профиля (Рис. 7):

* Настройки сохраняются в таблице Users;
* Значения применяются через CSS-переменные в <html>.

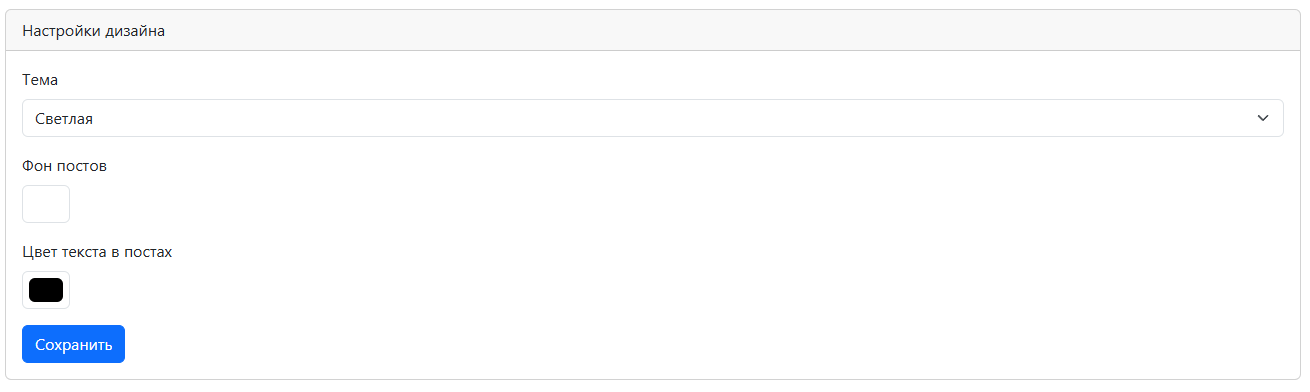


Рисунок 7. Настройка тем.

* + 1. Аутентификация и авторизация

За безопасность входа и защиты приватных маршрутов отвечает Passport.js с локальной стратегией: проверка email и пароля (bcrypt), сериализация пользователя в сессии и десериализация при каждом запросе. Middleware ensureAuthenticated гарантирует, что доступ к страницам создания и редактирования постов будет только у авторизованных.

* Passport.js с LocalStrategy: вход по email/password, хеширование bcrypt (Рис. 8.
* express-session для хранения сессий.
* Middleware ensureAuthenticated защищает приватные маршруты.

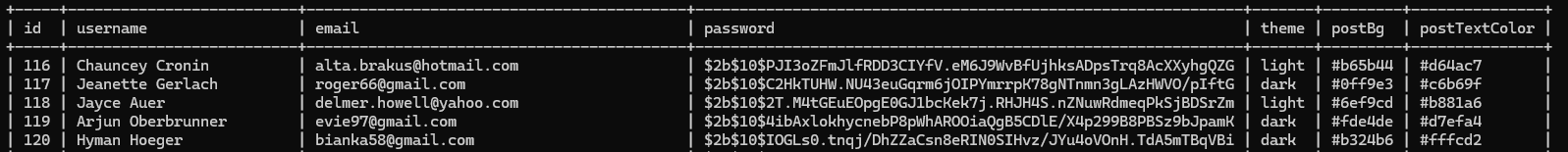


Рисунок 8. Хэширование bcrypt.

* + 1. Профиль и настройки

В личном кабинете (/profile) отображаются имя, email и общее число записей пользователя (Рис. 9). Раздел «Настройки» (/profile/settings) позволяет выбрать светлую или тёмную тему, а также установить произвольные цвета фона и текста постов. После сохранения пользователь получает мгновенный отклик — все страницы перерисовываются с новыми параметрами, благодаря использованию CSS-переменных.

* GET /profile — вывод имени, email и числа постов, кнопка «Настройки».
* GET/POST /profile/settings — форма выбора theme, postBg, postTextColor, отправка и сохранение в БД.

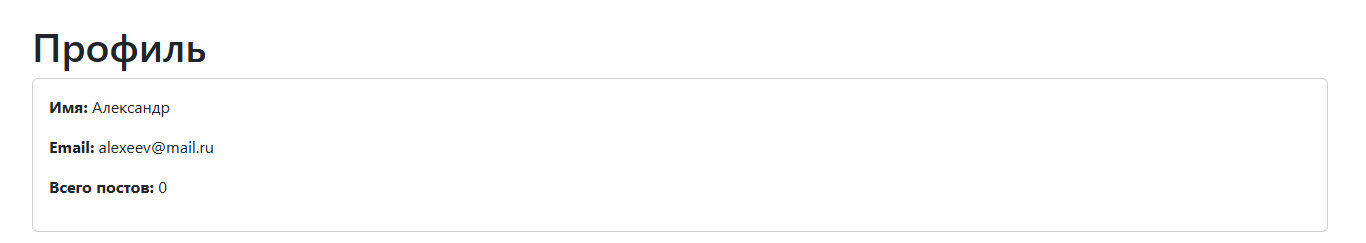


Рисунок 9. Профиль с отображением личной информации.

* + 1. Seed-данные

Для удобства тестирования и демонстрации платформы реализованы сидеры (Рис. 8) через Sequelize CLI, которые с помощью Faker-RU генерируют 20 реальных пользователей и по одному посту на каждого. Запуск npx sequelize db:seed:all наполняет базу, позволяя сразу проверить отображение элементов и работу CRUD-функционала на множестве записей:

* Пользователи: 20 записей с реалистичными именами и email.
* Посты: по одному на каждого пользователя.



Рисунок 10. Seeder для генерации постов.

## 2.4 Размещение веб-приложения на хостинге

Размещение веб-приложения на хостинге позволяет сделать его доступным для пользователей в сети Интернет. В рамках данного проекта в качестве платформы был использован виртуальный сервер (VPS), предоставляемый хостинг-провайдером Beget, поскольку обычный виртуальный хостинг не поддерживает запуск приложений на Node.js.

Подключение к серверу

После заказа VPS в панели управления Beget был предоставлен внешний IP-адрес и доступ по SSH. Подключение осуществлялось через программу PuTTY, где в качестве имени пользователя использовался root, а доступ осуществлялся по паролю, установленному вручную через консоль VNC. Также использовалась утилита WinSCP для передачи файлов проекта на сервер.

Установка необходимых компонентов

На сервер была установлена актуальная версия Node.js и пакетный менеджер npm, с помощью которых был выполнен запуск проекта. Дополнительно был установлен менеджер процессов pm2, обеспечивающий автоматический перезапуск и автозапуск приложения при сбоях и перезагрузке сервера. Основные команды (Рис. 11):

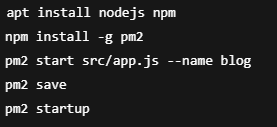


Рисунок 11. Команды для запуска сервера.

Настройка базы данных

Для работы с базой данных был установлен и настроен сервер MySQL(Рис. 12). Была создана отдельная база данных и пользователь с соответствующими правами доступа. Sequelize ORM использовался для управления миграциями и структурой БД.



Рисунок 12. Команды для установки и настройки MySQL.

Запуск приложения

После установки зависимостей и настройки конфигураций (config/config.js) проект был запущен с помощью команды npm run dev, а затем — переведён в фоновый режим через pm2. Сервер был настроен на прослушивание HTTPS-порта 5000, что позволило получить к нему доступ по адресу: https://217.114.6.130:5000/

# Заключение

В рамках выполнения курсового проекта была разработана современная, функциональная и надёжная блог-платформа, представляющая собой полнофункциональное веб-приложение. Реализация охватила все ключевые этапы жизненного цикла разработки — от проектирования архитектуры и моделирования базы данных до финальной вёрстки пользовательского интерфейса, настройки систем безопасности и обеспечения масштабируемости. Платформа предназначена для создания, редактирования и публикации пользовательских записей, поддерживает работу с вложениями и предоставляет гибкие возможности персонализации.

Основу приложения составляют три сущности — пользователи (User), записи (Post) и файлы (File), реализованные с помощью Sequelize ORM и MySQL. Это обеспечило чёткую структуру базы данных, удобство масштабирования и поддержки. Миграции и seed-данные через Sequelize CLI ускоряют развертывание и настройку тестового окружения.

Серверная логика написана на базе Express.js. Вся маршрутизация организована по принципам REST-архитектуры, а логика обработки запросов строго разделена на контроллеры, сервисы и модели, что соответствует MVC-подходу и способствует поддерживаемости и читаемости кода. Такая структура позволяет легко масштабировать проект, дополняя его новыми функциональными модулями без нарушения архитектурной целостности.

Вопросы безопасности реализованы с особым вниманием. Для аутентификации используется Passport.js с локальной стратегией, а bcrypt обеспечивает безопасное хеширование пользовательских паролей. Также в проекте применяется Helmet, автоматически добавляющий надёжные HTTP-заголовки, включая Content Security Policy (CSP), что защищает от XSS-атак и других распространённых угроз в веб-среде.

Пользовательский интерфейс разработан с применением EJS-шаблонов, что обеспечивает быструю генерацию HTML на стороне сервера. Дизайн оформлен с использованием SCSS и Bootstrap, что позволило быстро создать адаптивную, кроссбраузерную и эстетически приятную визуальную часть платформы. Дополнительная гибкость достигается за счёт использования CSS-переменных, благодаря которым каждый пользователь может индивидуализировать внешний вид своего блога: выбирать между светлой и тёмной темами, настраивать фон и цветовую палитру текста.

Особое внимание уделено удобству создания и редактирования контента. Встроенный WYSIWYG-редактор Quill обеспечивает интуитивно понятный интерфейс для форматирования текста, вставки изображений и других элементов, что делает процесс написания постов максимально комфортным даже для начинающих пользователей. За загрузку изображений и других файлов отвечает Multer, который гарантирует безопасную и надёжную передачу вложений на сервер с возможностью последующей обработки.

Проект изначально задумывался как модульный и легко расширяемый. Благодаря грамотно организованной структуре кода, следованию принципам чистой архитектуры и использованию проверенных технологий, внедрение новых функций — таких как система комментариев, лайков и рейтингов, REST API для мобильных клиентов, полнотекстовый поиск или интеграция с социальными сетями — не требует глобальной переработки существующего функционала.

Таким образом, разработанная блог-платформа не только полностью удовлетворяет требованиям, предъявляемым к курсовой работе, но и демонстрирует глубокое понимание принципов современного веб-разработки.

# Список использованной литературы

1. Официальный сайт PHP [Электронный ресурс]. – Режим доступа: https://www.php.net/manual/ru/, свободный. – Дата обращения: 11.05.2025.
2. Документация Laravel на русском [Электронный ресурс]. – Режим доступа: https://laravel.ru/docs, свободный. – Дата обращения: 08.05.2025.
3. Документация Composer [Электронный ресурс]. – Режим доступа: https://getcomposer.org/doc/, свободный. – Дата обращения: 01.05.2025.
4. Хабр. Раздел PHP [Электронный ресурс]. – Режим доступа: https://habr.com/ru/hub/php/, свободный. – Дата обращения: 03.05.2025.
5. Документация MySQL [Электронный ресурс]. – Режим доступа: https://dev.mysql.com/doc/, свободный. – Дата обращения: 15.05.2025.
6. MDN Web Docs (HTML, CSS и JavaScript) [Электронный ресурс]. – Режим доступа: https://developer.mozilla.org/ru/docs/Web, свободный. – Дата обращения: 29.05.2025.
7. Pro Git. Книга на русском языке [Электронный ресурс]. – Режим доступа: https://git-scm.com/book/ru/v2, свободный. – Дата обращения: 22.05.2025.
8. Node.js на русском [Электронный ресурс]. – Режим доступа: https://nodejs.org/ru/, свободный. – Дата обращения: 13.05.2025.
9. Документация Visual Studio Code [Электронный ресурс]. – Режим доступа: https://code.visualstudio.com/docs, свободный. – Дата обращения: 01.05.2025.

**ПРИЛОЖЕНИЕ**

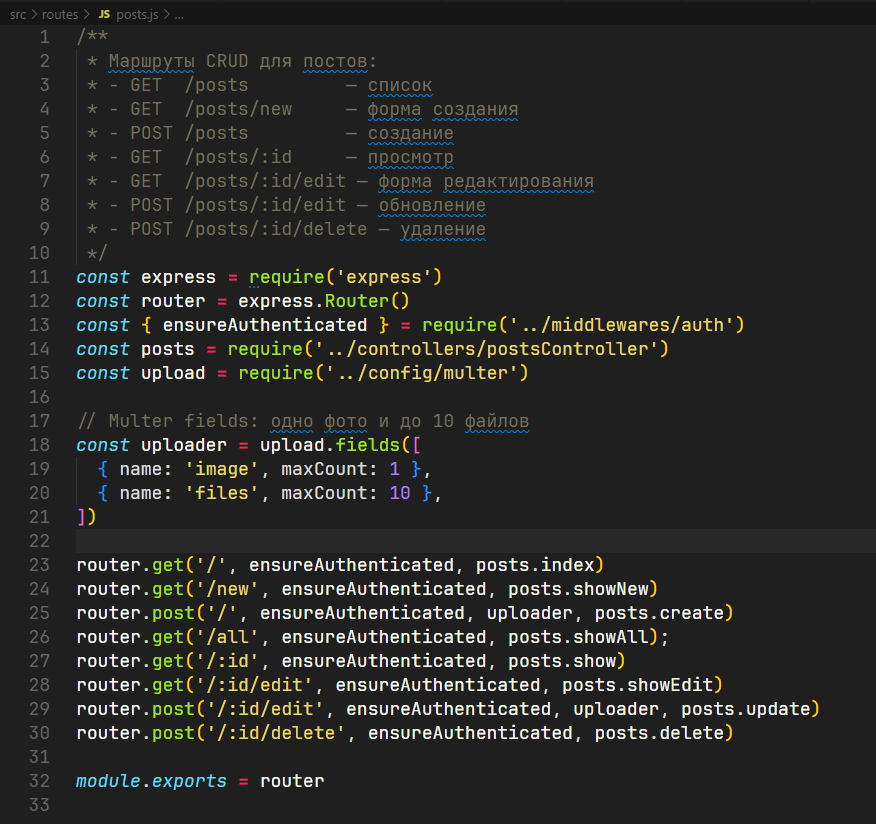


Рисунок 1. Пример маршрута posts.js.

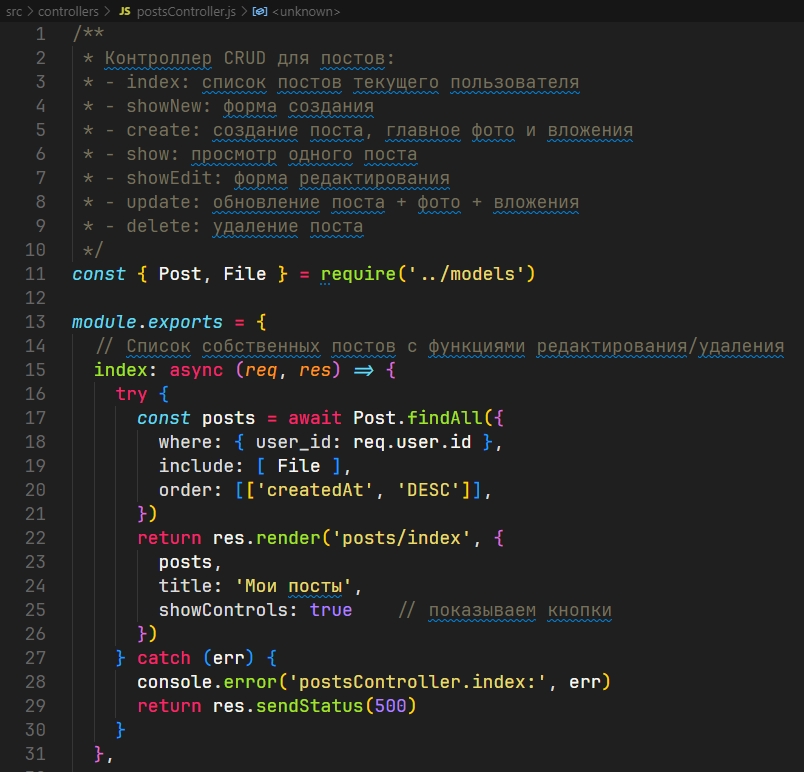


Рисунок 2. Пример контроллера postController.js.

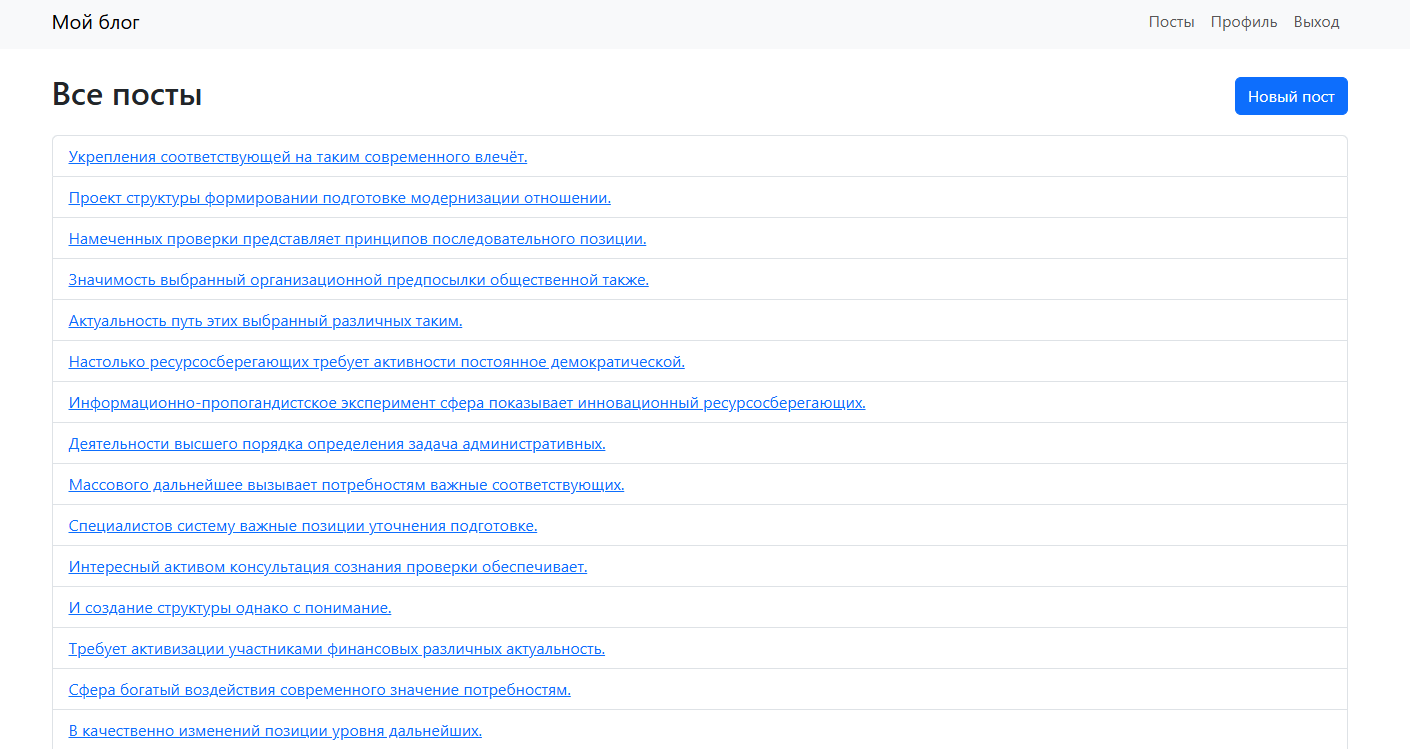


Рисунок 3. Все посты (нет кнопок редактирования).

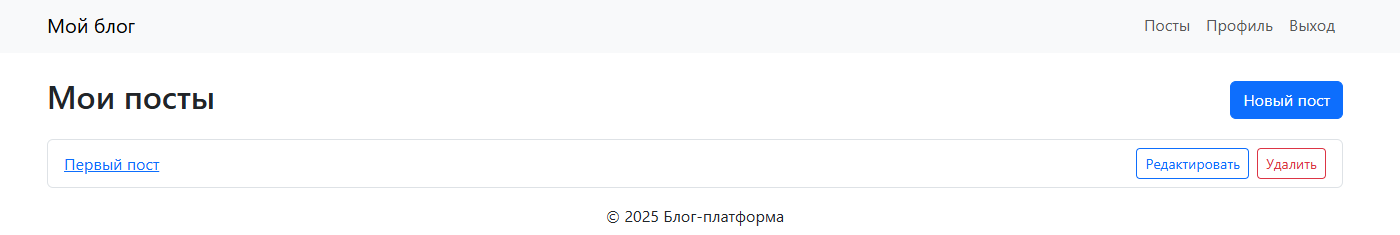


Рисунок 4. Мои посты (есть кнопки редактирования).

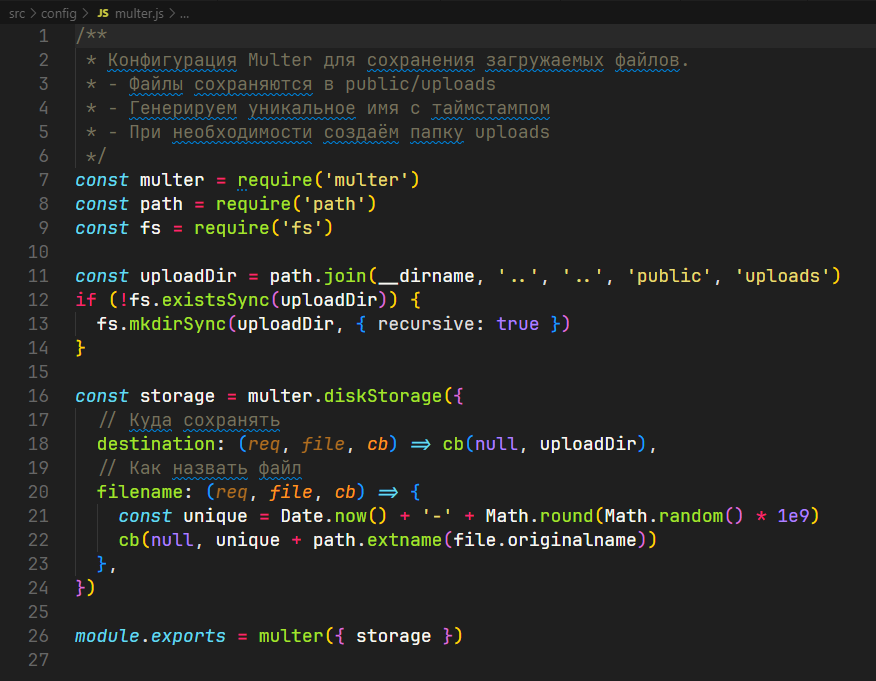


Рисунок 5. Реализация сохранения файлов на сервер.

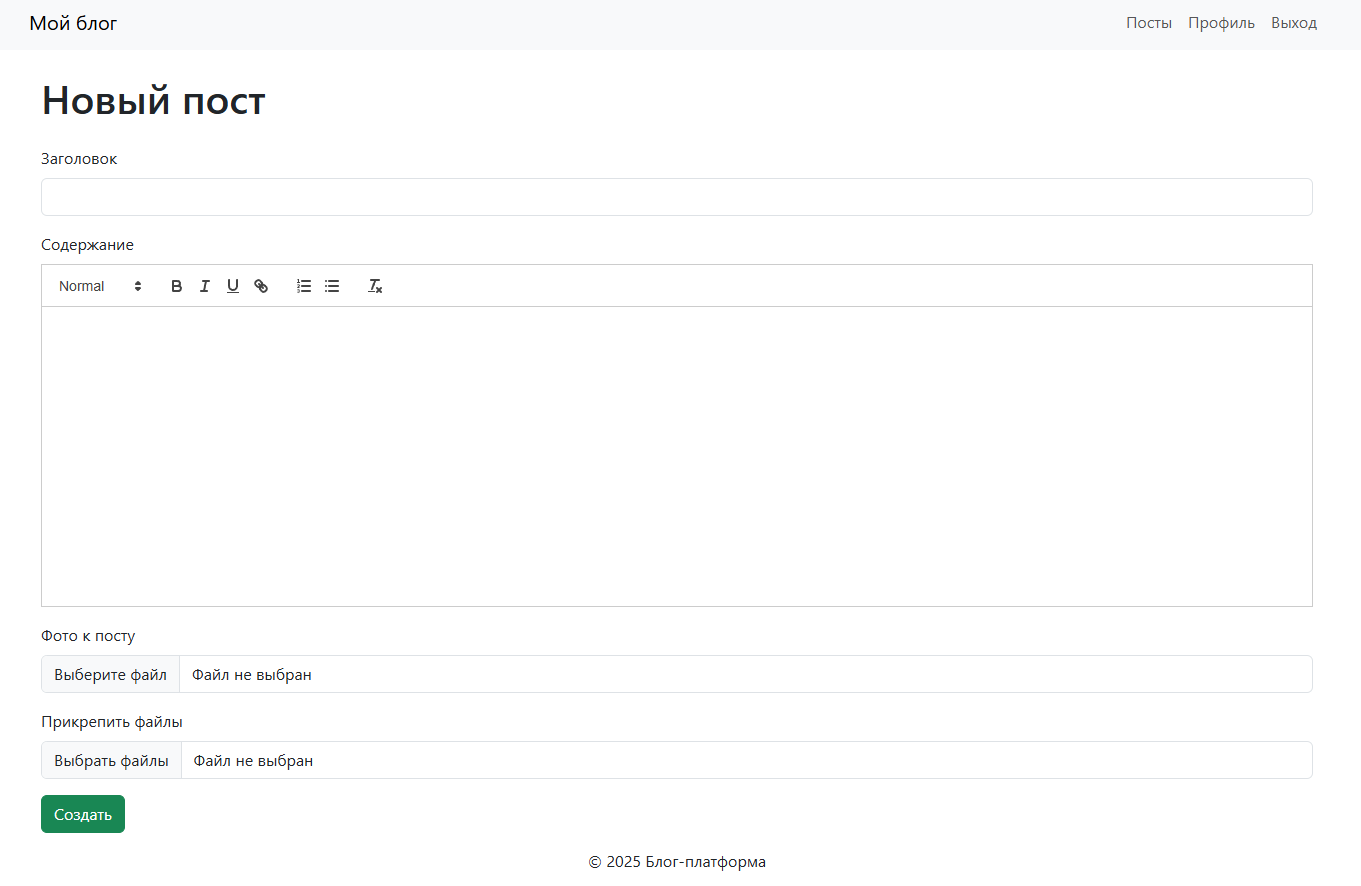


Рисунок 6. Форма создания/редактирования (new.ejs, edit.ejs).

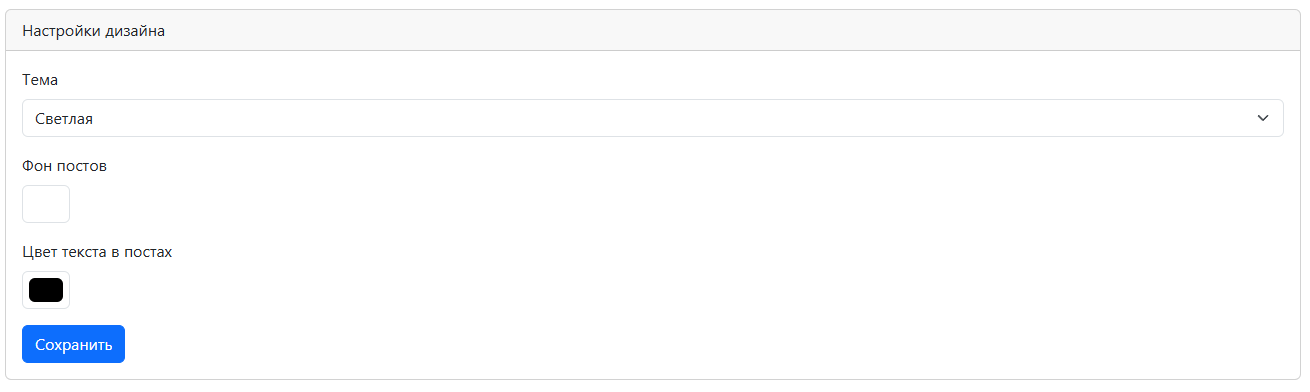


Рисунок 7. Настройка тем.

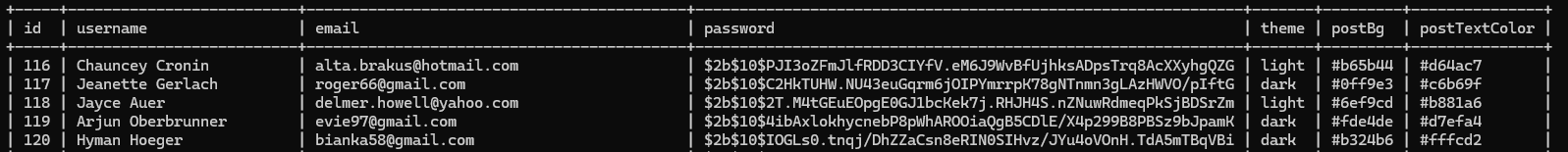


Рисунок 8. Хэширование bcrypt.

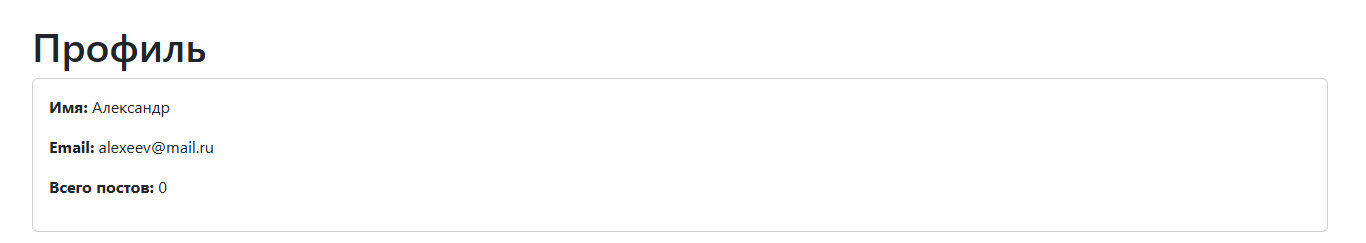


Рисунок 9. Профиль с отображением личной информации.



Рисунок 10. Seeder для генерации постов.

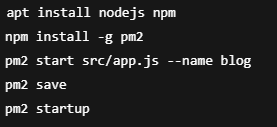


Рисунок 11. Команды для запуска сервера.



Рисунок 12. Команды для установки и настройки MySQL.