**Содержание**

[Введение 3](#_Toc26867)

[1 Руководство системному программисту 5](#_Toc24543)

[2 Руководство пользователя 8](#_Toc27231)

[3 Ревьюирование программного кода 12](#_Toc31427)

[4 Рефакторинг программного кода 15](#_Toc217)

[5 Программа и методика испытания веб-приложения 18](#_Toc13317)

[Заключение 19](#_Toc18601)

[Список используемых источников 20](#_Toc17651)

## Введение

Образование играет ключевую роль в жизни каждого человека, выступая не только источником новых знаний и расширения кругозора, но и инструментом формирования профессиональных навыков, развития научной мысли, технических достижений и обмена опытом внутри сообщества. Это особенно актуально для инженерных и технических направлений, где постоянное развитие, самообразование и доступ к достоверной информации являются основой профессионального роста.

С учётом увеличения численности населения, урбанизации, активной цифровизации и внедрения интеллектуальных технологий, возрастает необходимость создания удобных, надёжных и открытых информационных ресурсов, посвящённых электроэнергетике. Быстрое обновление нормативной базы, появление новых решений и усложнение отрасли требуют современных форм представления знаний. Цифровые образовательные платформы становятся важнейшим элементом системы подготовки и повышения квалификации специалистов энергетической отрасли.

Создание автоматизированной системы блога, посвящённого электроэнергетике, является своевременной и востребованной задачей. Такая система позволит эффективно распространять информацию о функционировании энергетических объектов, технологиях и нововведениях в отрасли. В условиях современного мира важно обеспечить быстрый и простой доступ к актуальным материалам без необходимости привлечения вспомогательного персонала.

Разработанное программное обеспечение для блога позволяет пользователям самостоятельно знакомиться с тематическими публикациями, упрощает навигацию и взаимодействие с материалами. Благодаря внедрению средств автоматизации и минимизации ручного администрирования достигается высокая автономность системы. Использование поисковых инструментов и фильтров способствует ускоренному доступу к нужной информации.

Цель данной преддипломной практики состоит в изучении и разработке руководства для разработчиков и пользователей для информационной системы блога об электроэнергетике, анализированию и оптимизации программного кода веб-приложения. Главной целью является оптимизировать программный код информационной системы блога об электроэнергетике для более быстрой и эффективной работе.

Основными задачами в данной преддипломной практики являются:

– разработать руководство разработчику и пользователю;

– изучить ревьюирование и рефакторинг програмного кода;

– выбрать нужный способ ревьюирования и рефакторинга програмного кода;

– провести тестирование веб-приложения.

Разработка автоматизированной системы для блога об электроэнергетике предполагает интеграцию различных технологий для создания эффективного и удобного программного обеспечения. Это включает в себя разработку интуитивно понятного интерфейса, интеграцию систем управления заказами.

Создание веб–сайта для блога об электроэнергетике является основной задачей для продвижения популяризации сферы образования об электроэнергетике. Конкуренция в сфере образования постоянно растёт, и каждый сайт по электроэнергетике демонстрирует свои решения для пользователей. Создание веб–страницы позволит пользователям не только ознакомиться с блогами об электроэнергетике, но и новостями, связанные с сайтом для блога об электроэнергетике.

Также необходимо учитывать, для какой аудитории разрабатывается программное обеспечение для блога об электроэнергетике. Это необходимо, так как у всех пользователей свои желания и предпочтения в образовании. Необходимо предоставить пользователям удобный и понятный интерфейс программного обеспечения, чтобы пользователям было понятно ориентироваться на веб–странице. Ни в коем случае нельзя делать программное обеспечение с большим количеством элементов, это может запутать пользователя и он не сможет разобраться в системе и просто уйдёт со страницы.

Разработка автоматизированной системы для блога об электроэнергетике имеет большой потенциал для улучшения процесса обучения, снижения затрат на обслуживание и повышения удовлетворенности пользователей. Продолжающееся развитие технологий и инноваций в этой области предоставляет возможности для создания все более совершенных и эффективных систем, отвечающих потребностям современных пользователей.

В рамках работы планируется изучить основные принципы работы автоматизированных систем для блога об электроэнергетике, а также провести обзор существующих технологий и применяемых решений в данной области. После этого будет проведён анализ требований потенциальных пользователей и определение функциональных и нефункциональных требований.

Объект исследования – блог об электроэнергетике.

Предмет исследования – сайт для блога об электроэнергетике и предоставление актуальной информации о нем.

## **1 Руководство системному программисту**

Информационная система для образовательного блога об электроэнергетике представляет собой сайт, который позволяет пользователям просматривать блоги, комментировать и оценивать их. Администраторам представлена возможность добавления блогов об электроэнергетике, изменения блогов, удаления, а так же модерации комментариев. Она состоит из нескольких компонентов, таких как главная страница, страница с блогами, личный кабинет, выставление оценки и комментариев, а так же админ панель, где администраторы смогут работать с блогами и модерировать комментарии. Так же есть система, позволяющая добавлять и удалять администраторов.

Для создания удобного и визуально привлекательного интерфейса образовательного блога об электроэнергетике необходимо использовать современные веб–технологии. К ним относятся:

– HTML, CSS, JavaScript – для верстки и стилизации страниц, а также для обеспечения динамических элементов интерфейса;

– адаптивная верстка – обеспечение корректного отображения сайта на мобильных устройствах, планшетах и компьютерах.

Бэкенд–часть системы отвечает за обработку пользовательских запросов, управление данными и обеспечение логики работы блога. Для этой части можно использовать:

– Node.js – для построения серверной части приложения. Node.js обеспечивает возможность разработки приложений с использованием JavaScript. Совместимость с JavaScript делает процесс разработки более удобным и доступным, позволяя разработчикам использовать в новых проектах уже сформированные навыки и опыт работы;

– SQLite – для хранения данных о пользователях, блогах, комментариях и рейтингах. Благодаря особенностям архитектуры SQLite работает быстро, особенно на чтение. Компоненты СУБД встроены в приложение и вызываются в том же процессе. Поэтому доступ к ним быстрее, чем при взаимодействии между разными процессами. Так же база данных состоит из табличных записей, связей между ними, индексов и других компонентов. В SQLite они хранятся в едином файле, который находится на том же устройстве, что и программа. Чтобы при работе не возникало ошибок, файл блокируется для сторонних процессов перед записью. Раньше это приводило к тому, что записывать данные в базу мог только один процесс единовременно. Но в новых версиях это решается перенастройкой режима работы СУБД.

Информационная система для блога об электроэнергетике имеет свою большую структуру. Структура данного проекта включает в себя следующие элементы:

– server.js – основной серверный файл, который помогает запускать приложение;

– ratings.db – файл базы данных, который хранит в себе все необходимые данные об пользователях, администраторов, блогахи так далее;

– public – папка, которая хранит в себе html, css и js файлы, а так же изображения на сайте;

– node\_modules – папка, которая хранит в себе все файлы и компоненты для работы node.js.

Для размещения веб–приложения выбран виртуальный выделенный сервер, предоставляющий возможность гибкой настройки серверной части, масштабирования проекта и высокой отказоустойчивости. Это особенно важно с учетом того, что приложение использует модульную архитектуру на базе Node.js и может требовать отдельного управления зависимостями и окружением. Для этот можно использовать Beget. [Beget](https://beget.com/ru/hosting/free" \t "https://skillbox.ru/media/marketing/9-besplatnykh-khostingov-dlya-saytov-razbiraemsya-kakoy-ispolzovat/_blank) – один из самых известных хостинг–провайдеров и регистраторов доменов в России.

Выбранная конфигурация VPS:

– процессор: 2 виртуальных ядра (vCPU, Intel Xeon);

– оперативная память (RAM): 4–8 ГБ;

– дисковое пространство: 60 ГБ SSD NVMe;

– сетевое подключение: 100 Мбит/с с возможностью расширения;

– серверное ПО: Node.js, SQLite.

Разработка проекта выполняется на пользовательском компьютере, соответствующем следующим требованиям:

– процессор: Intel Core i5 / AMD Ryzen 5;

– оперативная память: не менее 8 ГБ;

– диск: SSD 256 ГБ;

– рперационная система: Windows 10/11;

Программное обеспечение:

– среда разработки – Visual Studio Code;

– менеджер пакетов – npm / yarn;

– веб–браузеры – Google Chrome, Firefox.

Для запуска сервера необходимы следующие этапы:

– проверить на установленные зависимости. Для это необходимо в командной строке пропистаь команду npm intsall;

– убедиться в наличии файла базы данных ratings.db. Если его нет – создать файл и выполнить начальную инициализацию таблиц;

– настроить почтовый сервис в server.js. Для этого нужно указать логин и пароль для nodemailer;

– запуск сервера. Для этого необходимо ввести команду node server.js;

– проверить доступность веб-приложения по адресу http://localhost:3000

Если решили переименовать файл server.js, то в этом случае, перед запуска сервера, введите команду node новое название файла.js.

Информационная система для блога об электроэнергетике и имеет базу данных под названием ratings.db. Таблица включает в себя следующие данные:

– admins (администратор) – данная таблица хранит данные об администраторах;

– users (пользователи) – данная таблица хранит данные о пользователях;

– blogs (блоги) – данная таблица хранит данные о блогах;

– comments и ratings (комментарии, звездный рейтинг) – данная таблица хранит данные о комментариях и звездном рейтинге.

## **2 Руководство пользователя**

Информационная система для образовательного блога об электроэнергетике представляет собой сайт, который позволяет пользователям просматривать блоги, комментировать и оценивать их. Она состоит из нескольких компонентов, таких как главная страница, страница с блогами, личный кабинет, выставление оценки и комментариев.

Для объяснения работы пользователю информационной системе для блога о электроэнергии для этого необходимо расписать, как работает каждый элемент в данной системе.

Регистрация и вход.

Для начала пользователь может зарегистрироваться в веб-приложении. Регистрация позволит пользователю оставлять комментарии и оценивать блоки с помощью звездного рейтинга. Для регистрации необходимо перейти на страницу личного кабинета и нажать кнопку «Зарегистрироваться», после этого, откроется форма для регистрации. Форма регистрации изображена рисунке 1.

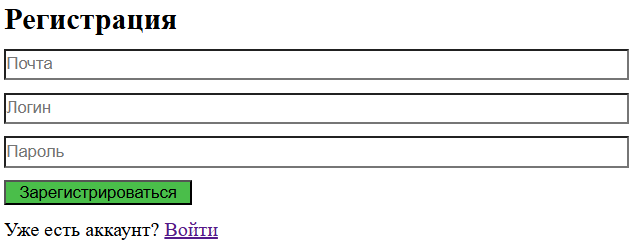


Рисунок 1 – Форма регистрации

После регистрации, пользователя перекинет в личный кабинет и сможет оставлять комментарии и оценивать блог с помощью звездного рейтинга. Личный кабинет изображен на рисунке 2

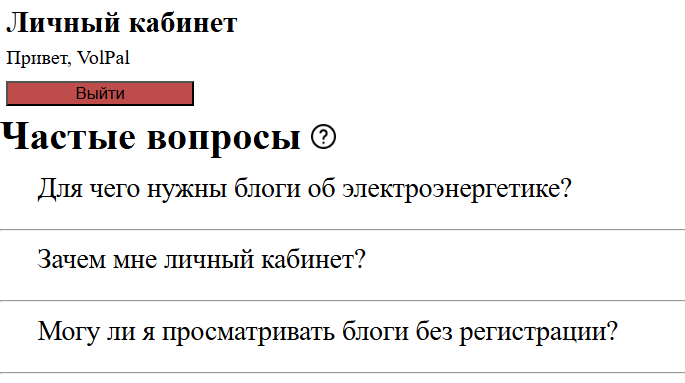


Рисунок 2 – Личный кабинет

Просмотр статей.

После регистрации пользователь сможет просмотреть все доступные блоги на главной странице, либо на странице «Блог». Все доступные статьи отображаются на главной странице или на странице «Блоги». Страница с блогами изображена на рисунке 3.

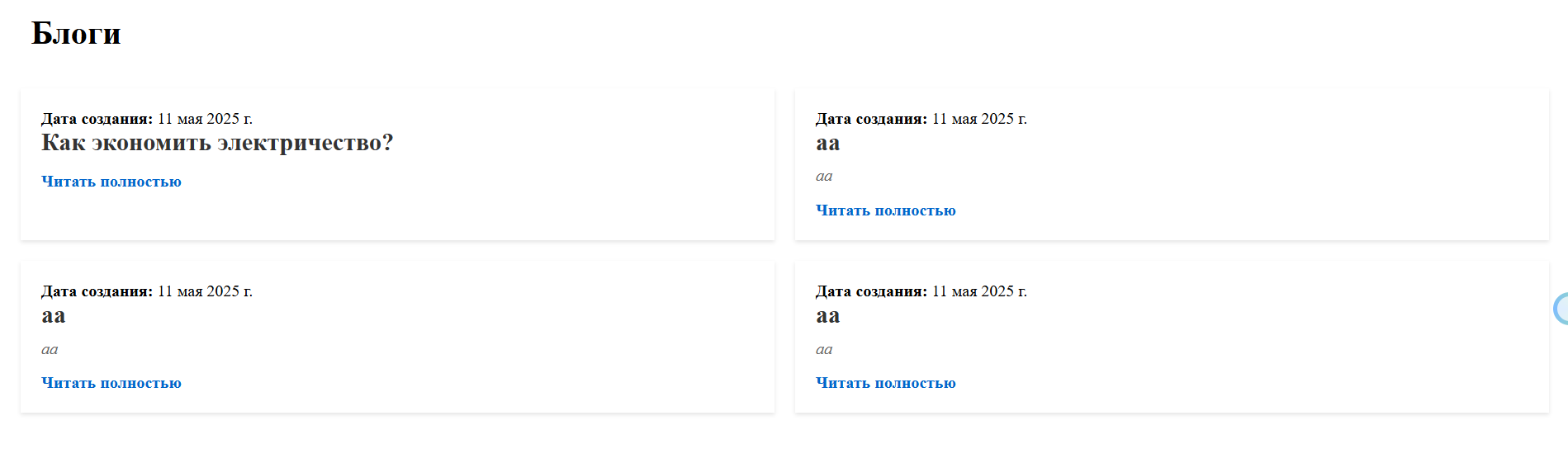


Рисунок 3 – Страница с блогами

При нажатии на «Читать полностью», откроется выбранный блог и пользователь сможет просмотреть блог. Ниже статьи отображаются комментарии пользователей и средняя оценка. Тестовый формат блога изображен на рисунке 4.

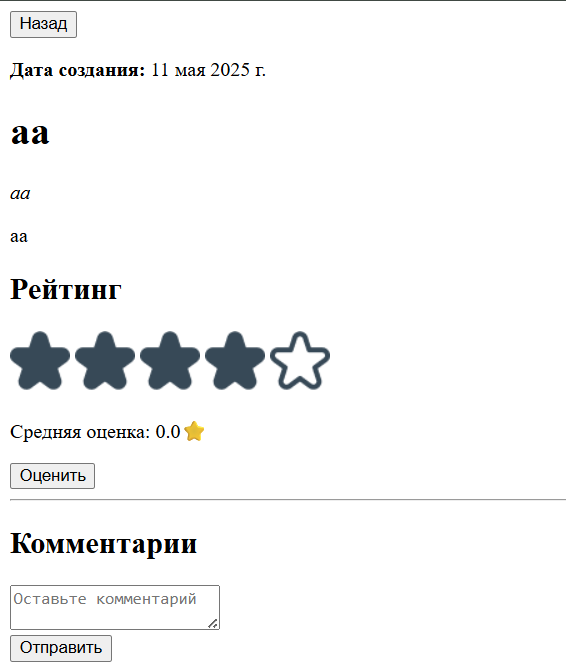


Рисунок 4 – Тестовый формат блога

Комментирование блога.

После прочтения статьи, пользователь сможет оставить свой отзыв о нем, написав комментарий специальное выведенном блоке. Чтобы оставить комментарий, пользователю необходимо авторизоваться. Внизу страницы с блогом находится форма ввода комментария. После ввода отзыва нажмите кнопку «Отправить». Комментарий появится сразу после отправки. Блок с комментарием изображен на рисунке 5.

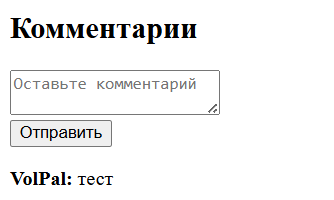


Рисунок 5 – Блок с комментариями

Оценка статей.

Пользователь может оставить оценку с помощью звездного рейтинга. Оценки выставляются в виде звёздочек от 1 до 5. Пользователь может оценить статью один раз, при этом он может изменить свою оценку позже. После выставления оценки, необходимо нажать кнопку «Сохранить». После отправки оценки, средний рейтинг меняется. Блок с звездным рейтингом изображен на рисунке 6

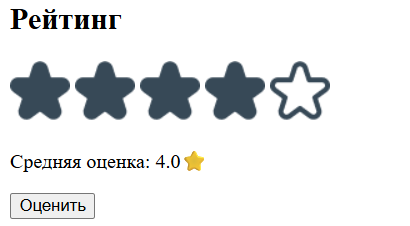


Рисунок 6 – Блок с звездным рейтингом

Ограничения и требования

Комментирование и оценка недоступны без регистрации. Не допускаются оскорбительные или спам-сообщения – они будут удаляться модератором. Добавление, редактирование и удаление блогов недоступны пользователю, это доступно только администратору.

Так же таблица с проблемами и их решением, описать группы пользователей

## **3 Ревьюирование программного кода**

Ревьюрирование кода – практика, при которой разработчики смотрят и оценивают код, написанный другими. Это важная часть создания программного обеспечения, которая помогает повысить качество, улучшить читаемость и обнаружить потенциальные проблемы.

Основная цель процесса – обнаружение ошибок и уязвимостей, а также улучшение качества кода. Для эффективного код-ревью необходимо установить ясные критерии оценки, определить роли и ответственности разработчиков, а также использовать инструменты и системы управления версиями, которые упрощают процесс.

Существуют различные подходы:

– парное программирование. Оно предполагает работу двух разработчиков над одним и тем же кодом, что позволяет обнаруживать ошибки на ранней стадии и обмениваться знаниями. Коллеги сотрудничают в режиме реального времени – один пишет код (драйвер), а другой просматривает его (навигатор). Такой способ часто используют команды разработчиков, потому что товарищи по команде совместно находят наиболее эффективное решение проблемы;

– обзоры «из-за плеча». Два разработчика – автор и рецензент – сотрудничают лично или удаленно через общий экран, и автор объясняет и аргументирует выбранные решения. Рецензент задает вопросы и вносит предложения. Автор может внести небольшие изменения во время обзора, а более крупные исправления отложить на потом;

– инструментальные обзоры. Использование инструментов позволяет сократить время и обеспечить создание кода самого высокого качества. Рецензии с помощью инструментов могут автоматически собирать измененные файлы и вычленять различия, упрощать получение отзывов и обсуждение с помощью комментариев. Также они могут включать статическое тестирование безопасности приложений (SAST), чтобы помочь выявить и устранить уязвимости;

– электронная почта. Передача по электронной почте часто используется для решения незначительных проблем и проверки небольших фрагментов.

Для ревьюрирование кода я использовал инструментальный обзор, Потому что данный метод быстрее, удобнее и этот способ дает рекомендации по исправлению кода.

Первая проблема – повторное использование middleware-ов. Код представлен на рисунке 7

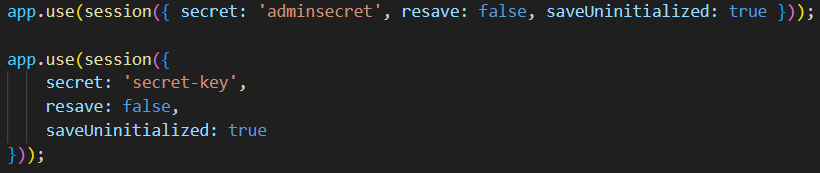


Рисунок 7 – повторное использование middleware-ов

Решение данной проблемы заключается в сокращении кода в один middleware. Исправленный код изображен на рисунке 8.

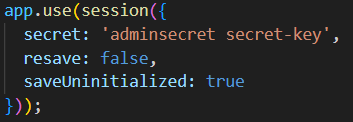
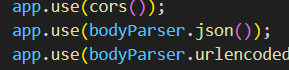


Рисунок 8 – исправленный middleware

Вторая проблема заключается в том, то, что мы использовали app.use(bodyParser.json()); два раза. Проблема изображена на рисунке 9.



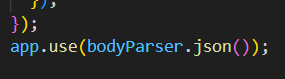


Рисунок 9 – повторяющийся app.use(bodyParser.json());

Решением данной проблемы является удалением одного из app.use(bodyParser.json());, а именно второго.

Проблема номер три – использование res.redirect в API. Это не харакетрно для REST, оно харакетрно для серверных маршрутов. Проблема изображена на рисунке 10.

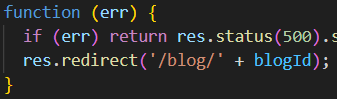


Рисунок 10 – использование redirect в REST

Решением данной проблемы является замена redirect на json.

Также была найдена еще одна проблема с повторением функции, а именно повторялось функция выхода из личного кабинета. Функция выхода из акккаунта изображена на рисунке 11.

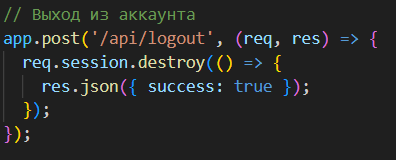


Рисунок 11 – функция выхода из аккаунта

Чтобы немного улучшить скорость работы системы, необходимо было убрать одну лишнюю функцию выхода из аккаунта.

Во время ревьюрирования программного кода информационной системы для блога об электроэнергетики была обнаружена уязвимость, а именно уязвимость к XSS-атакам.

XSS – довольно распространенная уязвимость, которую можно обнаружить на множестве веб-приложений. Ее суть довольно проста, злоумышленнику удается внедрить на страницу JavaScript-код, который не был предусмотрен разработчиками. Этот код будет выполняться каждый раз, когда жертвы будут заходить на страницу приложения, куда этот код был добавлен.

Данная уязвимость была обнаружена в функции для загрузки комментария. Функция для загрузки комментария изображена на рисунке 12.



Рисунок 12 - функция для загрузки комментариев

Для исправления данной уязвимости необходимо исправить код, сделать в более защищенном, используя функции textContent и createTextNode. Исправленный код изображен на рисунке 13.



Рисунок 13 - исправленный код загрузки комментариев

Благодаря textContent и createTextNode, они автоматически экранируют любые HTML-теги, введённые пользователями. Это предотвращает внедрение <script> и других опасных конструкций.

## **4 Рефакторинг программного кода**

Рефакторинг – изменение во внутренней структуре ПО, имеющее целью облегчить понимание его работы и упростить модификацию, не затрагивая наблюдаемого поведения. Рефакторинг позволяет нам улучшить читаемость, поддерживаемость и эффективность кода, делая его более гибким и легким для внесения будущих изменений.

Существует несколько причин, почему возникает необходимость. проведения рефакторинга программного кода:

– снижение технического долга. Технический долг – это накопленные проблемы и недостатки в коде, которые могут замедлять разработку и усложнять поддержку продукта. Рефакторинг помогает устранить эти проблемы и снизить технический долг, что способствует более плавному и эффективному процессу разработки;

– улучшение читаемости и тестируемости кода. Улучшение читаемости не только сократит время на чтение кода, но и поспособствует более легкому развитию и поддержке программного обеспечения в долгосрочной перспективе, а также облегчит написание автоматических тестов даже тем людям, у которых разработка не является основным направлением;

– повышение эффективности разработчиков. Когда код унифицирован, структурирован и понятен, можно с легкостью внедрять новые функции или поддерживать существующие. Кроме того, благодаря этому разработчики могут быстрее ориентироваться в коде и вносить изменения без риска нежелательного влияния на другие части программного обеспечения, что в конечном итоге снижает количество ошибок.

В рефакторинге участвовало два файла server.js и script.js. В server.js было много повторяющихся ссылок на кокретные файлы, что могло повлиять на скорость работы системы. Код до рефакторинга изображенен на рисунке 14.

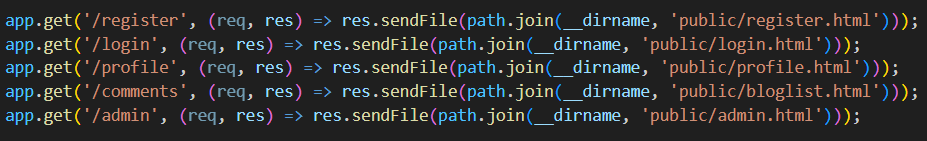


Рисунок 14 - код до рефакторинга

После рефакторинга код стал короче и проще в развитии. Все повторяющиеся маршруты были объединены в один цикл forEach. forEach используется для итерации элементов массива и выполнения указанной функции один раз для каждого элемента. Чтобы добавить новый маршрут, достаточно просто добавить его в массив. Код после рефакторинга изображен на рисунке 15.

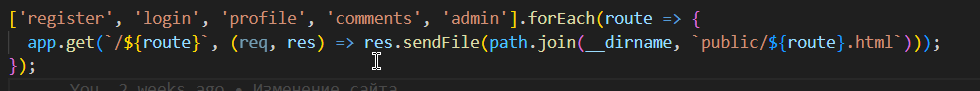


Рисунок 15 - код после рефакторинга

До рефакторинга программного кода таблиц базы данных информационные системы для блога об электроэнергетике, все таблицы были прописаны раздельно. Это может повлиять на скорость создания и подключения к таблицам в базе данных. На рисунке 16 продемонстрирован код базы данных до рефакторинга.

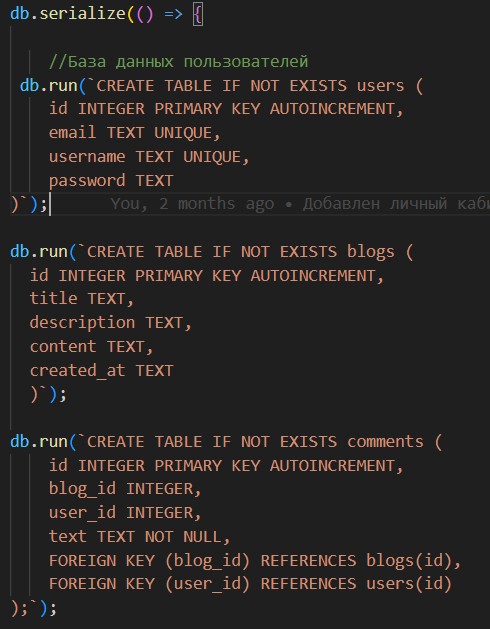


Рисунок 16 - код до рефакторинга

После рефакторинга, создание всех таблиц было объединено в одну функцию const tabels, к которой подключена функция db.serialize. Благодаря этому создание и подключение таблиц станет быстрее. Код создание таблиц после рефакторинга изображен на рисунках 17 и 18 .

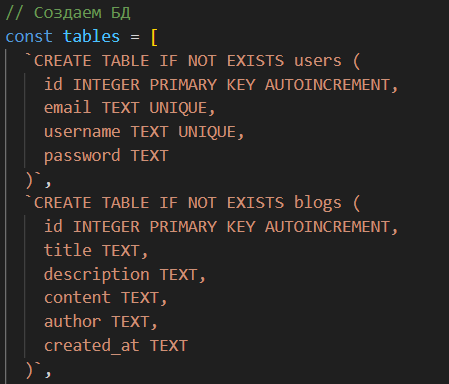


Рисунок 17 - код для добавления таблиц после рефакторинга

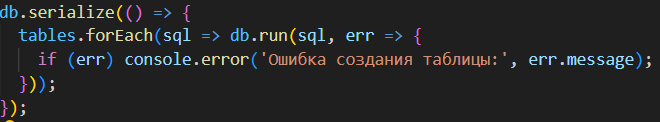


Рисунок 18 - код создания таблиц после рефакторинга

## **5 Программа и методика испытания веб-приложения**

Что такое испытания веб-приложения, процесс тестирования.

## Заключение

## Список используемых источников