**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 20](#_Toc13317)

[Заключение 27](#_Toc18601)

[Список используемых источников 28](#_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.

Также при работе с сайтом у администратора могут возникнуть ошибки. Ошибки, причина и их решение указаны в таблице 1

|  |  |  |
| --- | --- | --- |
| Ошибка | Причина | Решение |
| Пароль слишком простой. Используйте заглавные, строчные буквы, цифры и спецсимволы. | Пароль не соответствует требованиям (длина ≥ 8, есть заглавная, строчная буква, цифра и спецсимвол) | Придумать более сложный пароль, например: Admin@2024 |
| "Неверные данные" при входе администратора | Неверный логин, пароль или секретный ключ | Убедиться, что логин, пароль и 6-значный ключ указаны верно (без пробелов).  Проверить раскладку клавиатуры |
| "Ошибка отправки письма" при регистрации администратора | Неправильный email или проблемы со связью с почтовым сервером | Убедиться, что email указан верно.  Убедиться, что пароль приложения действителен и указана верная почта отправителя (например: user: 'example@mail.com',  pass: 'aaaa bbbb cccc dddd'  ) |

Таблица 1 – ошибки, причины и их решения

## **2 Руководство пользователя**

Руководство пользователя – это документ, который предоставляет подробную информацию о том, как использовать определенное оборудование, программу или сайт. В руководстве предоставляются инструкции по установке, порядок выполнения последовательности действий по настройке и использованию оборудования или ИТ-продукта, а также описание функций, возможностей и рекомендации по их использованию.

Руководство пользователя помогает новым пользователям быстрее освоиться с программой или сайтом, избежать ошибок и максимально эффективно использовать их возможности. В случае опытных пользователей руководство пользователя необходимо для обеспечения подробной информации о всех возможностях и функциях программы. Оно помогает пользователям освоить более сложные, «продвинутые» функции, узнать о возможных проблемах и их решениях, а также повысить производительность и эффективность работы. Руководство пользователя также может включать в себя советы по эксплуатационной безопасности и рекомендации по оптимальным настройкам для получения наилучших результатов.

Информационная система образовательного блога об электроэнергетике представляет собой современное веб‑приложение, созданное для обмена знаниями, повышения профессиональной компетенции и формирования сообщества специалистов и энтузиастов отрасли. Архитектура сайта разделена на несколько логических модулей, каждый из которых отвечает за определённый набор функций и гарантирует гибкость дальнейшего развития.

В информационной системе для блога об электроэнергетики существует три пользователя:

– зарегистрированный пользователь. Этот тип пользователя может не только просматривать блоги, он может оставлять отзыв с помощью комментария, а также оценивать блок с помощью звездного рейтинга;

– гость. Этот тип пользователя может просматривать блоги, но не может оставлять комментарии и оценивать блоги с помощью звездного рейтинга;

– администратор. Этот тип пользователя может создавать новые блоги, изменять их, а так же удалять блоги.

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.

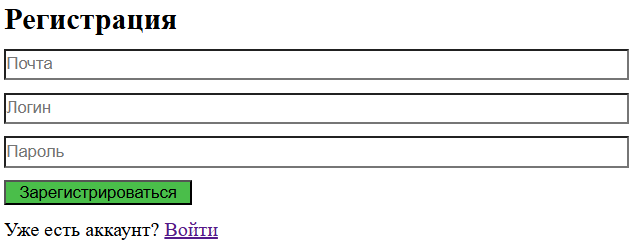


Рисунок 1 – Форма регистрации

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

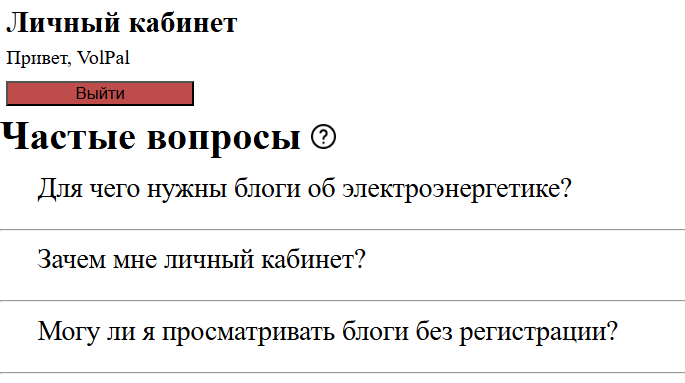


Рисунок 2 – Личный кабинет

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.

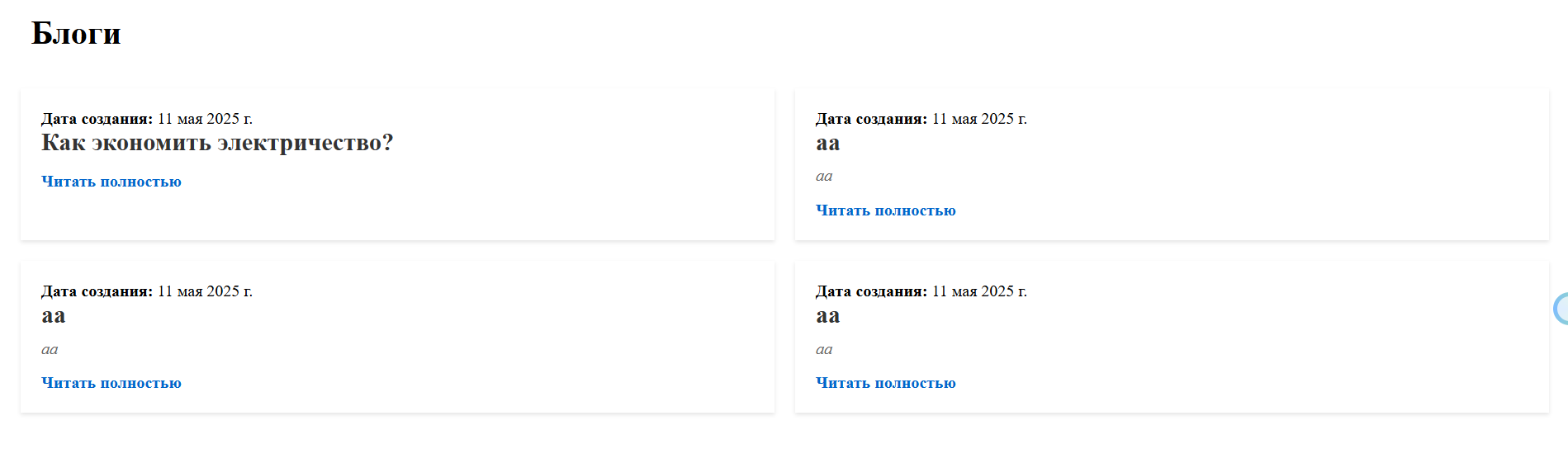


Рисунок 3 – Страница с блогами

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.

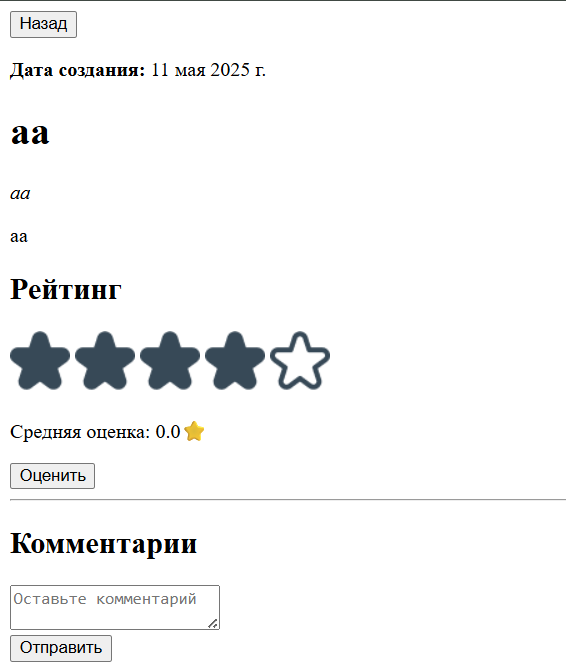


Рисунок 4 – Тестовый формат блога

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.

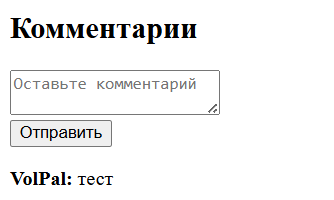


Рисунок 5 – Блок с комментариями

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

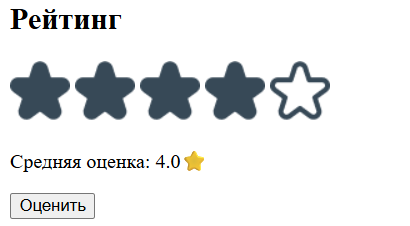


Рисунок 6 – Блок с звездным рейтингом

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.

Также при работе с сайтом у пользователя могут возникнуть ошибки. Ошибки, причина и их решение указаны в таблице 1

|  |  |  |
| --- | --- | --- |
| Ошибка | Причина | Решение |
| "Ошибка регистрации" при создании пользователя | Такой логин уже существует | Выбрать другое имя пользователя |
| "Неверные учетные данные" при входе обычного пользователя | Введены неправильные имя пользователя или пароль | Убедиться, что логин и пароль правильные.  Зарегистрироваться, если ещё не создана учётная запись |
| Ошибка при добавлении комментария: Комментарий не может быть пустым | Поле ввода комментария остался не заполненым | Заполнить поле комментария перед отправкой |

Таблица 1 – ошибки, причины и их решения

## **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

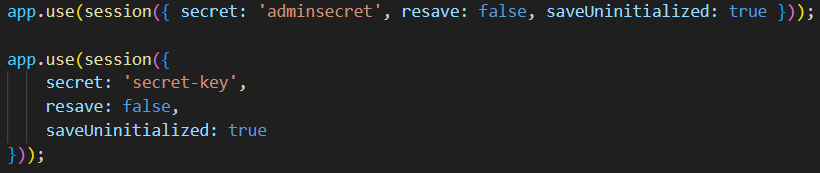


Рисунок 7 – повторное использование middleware-ов

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

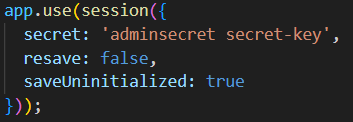
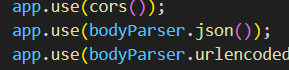


Рисунок 8 – исправленный middleware

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



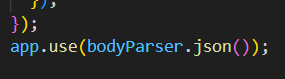


Рисунок 9 – повторяющийся app.use(bodyParser.json());

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.

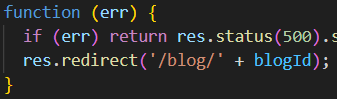


Рисунок 10 – использование redirect в REST

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.

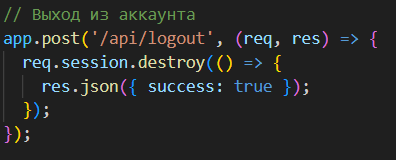


Рисунок 11 – функция выхода из аккаунта

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.

Перед началом рефакторинга необходимо иметь обширный набор тестов, который покрывает все аспекты функциональности приложения. Тесты должны запускаться автоматически после каждого изменения, чтобы убедиться в отсутствии регрессии.

Изменения в коде выполняются таким образом, чтобы не нарушить его работоспособность. Даже если в коде много «некрасивых» решений, приоритет отдается стабильности работы приложения. Также изменения, выполняемые в процессе рефакторинга, не должны видоизменять внешний интерфейс программы и ее поведение с точки зрения пользователя.

Выполнение преобразований небольшими последовательными шагами позволяет избежать ошибок, которые могут возникнуть при крупных и радикальных переработках. Эта практика также облегчает отслеживание внесенных изменений и откат нежелательных эффектов.

Рефакторинг кода проходит в следующих случаях:

– код сложен и труден для понимания. Плохая читаемость кода свидетельствует о высокой сложности его конструкций и отсутствии четкой структуры. Рефакторинг в этом случае направлен на упрощение структуры кода и на улучшение его логичности и организации. Таким образом код становится понятнее, что ускоряет разработку и облегчает обучение новых членов команды;

– в коде много дублирования. Частые повторения одинаковых или похожих фрагментов кода увеличивают вероятность ошибок при внесении изменений, так как каждый случай дублирования требует одинаковых корректировок. Это не только утяжеляет программу, но и влечет риски для ее поддержки и расширения. Рефакторинг с целью избавления от дублирования часто включает создание общих методов или классов. Сокращение дублированных частей упрощает процессы тестирования и сопровождения кода;

– производительность системы снижена. Анализ показывает, что медленная работа часто связана не с инфраструктурными ограничениями, а с неоптимальным кодом. Рефакторинг с акцентом на производительность может включать оптимизацию алгоритмов, устранение избыточных вычислений и уменьшение числа операций ввода-вывода. Улучшение эффективности кода часто ведет к лучшему пользовательскому опыту и сокращению ресурсов, необходимых для поддержки приложения.

– код содержит устаревшие решения. Устаревший код может стать причиной проблем с безопасностью, производительностью и интеграцией с другими системами. Рефакторинг с учетом последних достижений в области технологий дает возможность модернизировать приложение. Обновление кодовой базы обеспечивает лучшую поддержку, совместимость и предоставляет возможность использования нового функционала.

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.

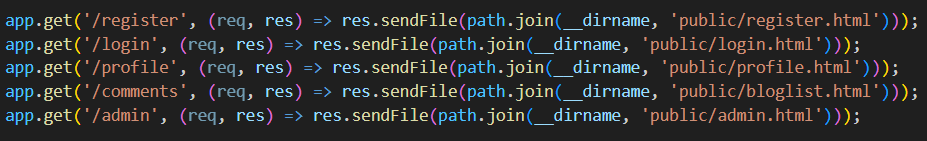


Рисунок 14 – код до рефакторинга

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.

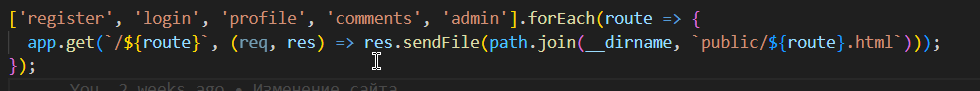


Рисунок 15 – код после рефакторинга

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.

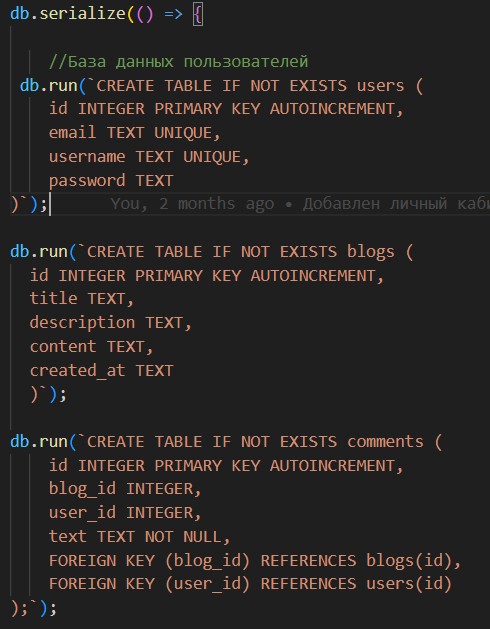


Рисунок 16 – код до рефакторинга

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 .

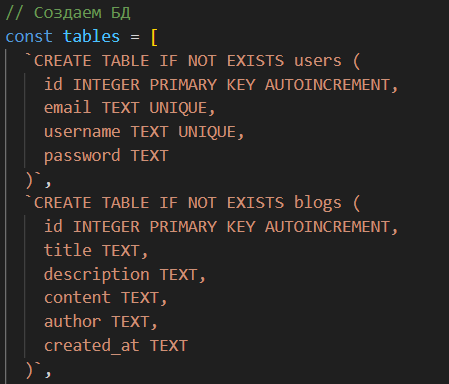


Рисунок 17 – код для добавления таблиц после рефакторинга

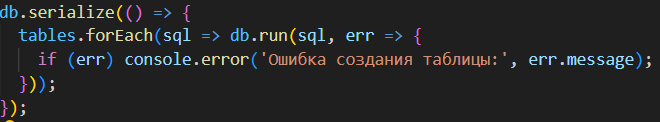


Рисунок 18 – код создания таблиц после рефакторинга

В script.js каждая функция делала fetch, res.json и так далее. Это не практично, так как это многократное дублирование, которое усложняет поддержку и увеличивает вероятность ошибок. На рисунке 19 изображен один из примеров, где в функциях использовались fetch и res.json.

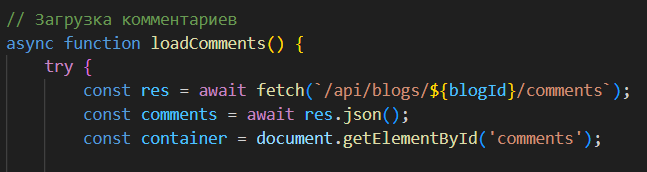


Рисунок 19 – код с fetch и res.json

Поэтому для облегчения поддержки и уменьшения вероятности ошибок была создана отдельная fetchJSON. Это позволит системе обрабатывать все ошибки, а также выводить понятные сообщения об ошибке. Обработчик ошибок представлен на рисунке 20.

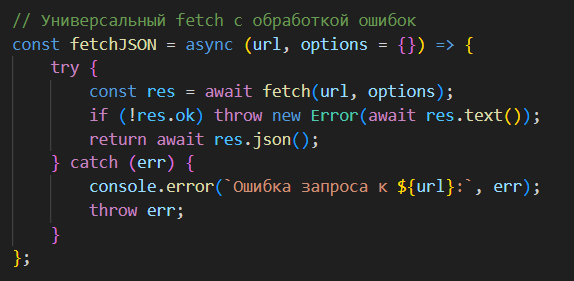


Рисунок 20 – обработчик ошибок

В исходной реализации функции deleteRating присутствует дублирующий запрос к /api/session, несмотря на наличие функции checkLogin(), которая уже выполняет аналогичную проверку авторизации. Данная функция представлена на рисунке 21.

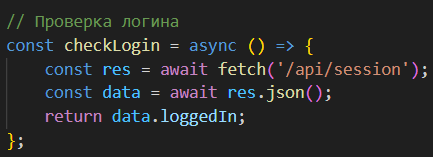


Рисунок 21 – проверка логина

Вместо дублирования кода используется повторное применение функции checkLogin(), что делает реализацию короче, чище и проще в поддержке: при изменении логики авторизации потребуется внести правки только в одном месте - в checkLogin(). Обновленная функция проверки логина продемонстрирована на рисунке 22. Также пример использования функции проверки логина в других функциях представлен рисунке 23.

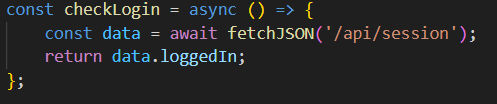


Рисунок 22 – обновленная проверка логина

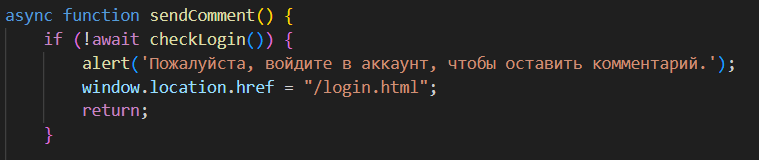


Рисунок 23 – пример использование функции проверки логина в функции отправки комментария

Изначально выполнение DOM-дерева не является асинхронным, в результате чего вложенные в него функции запускаются одновременно, не ожидая завершения друг друга. Это может привести к сбоям в случае зависимости одной функции от другой, визуальным ошибкам из-за несвоевременной загрузки данных, а также к трудноотлавливаемым сбоям выполнения. Первоначальный год изображен на рисунке 24.

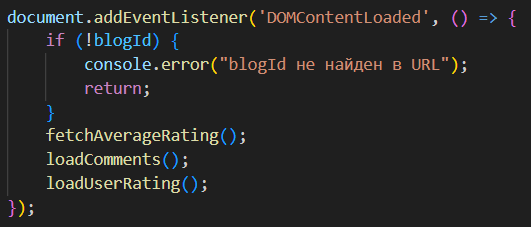


Рисунок 24 – первоначальный вид DOM-дерева

После рефакторинга DOM-дерево использует функцию асинхронности, что позволяет использовать функцию await. Это гарантирует, что все функции в DOM-дереве будут загружаться по очереди. Это позволит избежать меньше ошибок, а также поможет легче обновлять DOM-дерево. DOM-дерево после рефакторинга изображено на рисунке 25.

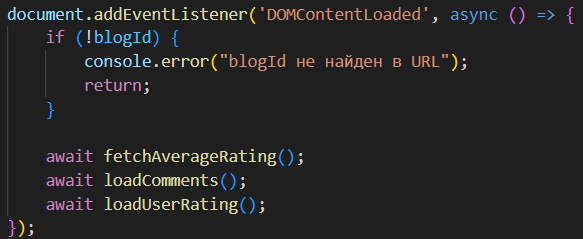


Рисунок 25 – DOM-дерево после рефакторинга

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

Испытания веб-приложения – это процесс всесторонней проверки работоспособности, стабильности, функциональности, безопасности и удобства использования веб-системы с целью выявления и устранения ошибок до внедрения в эксплуатацию.

Существует несколько видов тестирования, а именно:

– функциональное тестирование – проверка корректности работы функционала сайта или веб-приложения;

– тестирование производительности – измерение скорости загрузки страниц, а также способности сайта или приложения выдерживать большое количество пользователей и запросов;

– тестирование безопасности – проверка наличия уязвимостей и устойчивости к потенциальным атакам;

– тестирование совместимости – обеспечение корректной работы веб-приложения на разных браузерах, устройствах и операционных системах;

– тестирование юзабилити – оценка удобства и простоты использования сайта или приложения для пользователей;

– ручное тестирование – процесс поиска ошибок в программе без использования специальных ПО, силами человека.

Объектом тестирования является веб-приложение «Блог об электроэнергетике».

Комплектность системы:

– серверная часть: server.js, база данных ratings.db, папка public (HTML, CSS, JS);

– клиентская часть: HTML-страницы (index.html, blog.html, login.html, admin.html и др.);

– база данных: SQLite-файл с таблицами пользователей, блогов, комментариев, рейтингов;

– документация: руководство пользователя, руководство системному программисту, инструкция по установке.

Оценка результатов проведения тестирования разработки информационной системы для блога об электроэнергетике включает анализ результатов тестов и их соответствие ожиданиям и требованиям. Так же, оценка результатов проведения тестирования помогает понять текущее состояние сайта. Он включает в себя результаты тестирования, выявленные дефекты и рекомендации по их устранению. Такой отчет позволяет:

– оценить качество сайта;

– определить области, требующие улучшений;

– обеспечить прозрачность процесса тестирования;

– сформировать план действий для дальнейшего развития проекта.

Отчет о тестировании также служит важным инструментом для коммуникации между различными участниками проекта. Он помогает менеджерам проектов, разработчикам, тестировщикам и другим заинтересованным сторонам быть в курсе текущего состояния сайта и принимать обоснованные решения. Кроме того, отчет может быть использован для документирования процесса тестирования и хранения информации для будущих проектов. Вот несколько важных аспектов, которые следует учитывать при оценке результатов.

Успешность прохождения тестов. Необходимо проанализировать количество и типы ошибок, выявленных в процессе тестирования. Если тесты успешно проходят без ошибок, это может указывать на то, что информационная система по садоводству работает стабильно и соответствует своим требованиям. Если же при тестировании были обнаружены те или иные ошибки и уязвимости, то ошибки и уязвимости нужно исправить

Покрытие функциональности. Важно проверить, насколько полностью тестирование охватывает все функциональные возможности системы. Если тесты покрывают большую часть функций информационной системы по садоводству, это говорит о хорошем покрытии тестами.

Отклонения от требований. Если есть отклонения или несоответствия, важно документировать их и обратить внимание на их серьезность. Некоторые отклонения могут быть незначительными и легко устранимы, в то время как другие могут требовать значительных изменений в системе. Если не проверить на отклонения от требований, это может повлиять на будущее работы информационной системы.

Производительность и надежность. Бывают случаи, когда на информационную систему заходит много людей, из-за чего, может произойти понижение производительности или же полная неработоспособность сайта. Поэтому необходимо оценить результаты тестирования, чтобы определить производительность и надежность информационной системы по садоводству. Важно убедиться, что система работает эффективно и надежно в различных условиях и нагрузках.

Реакция на ошибки. Адекватная реакция на ошибки – критически важный аспект надежности программного обеспечения. Система должна не только обнаруживать нештатные ситуации, но и корректно реагировать на них, минимизируя негативное воздействие на пользователя и другие компоненты. Грамотная обработка ошибок подразумевает предоставление информативных сообщений, позволяющих пользователю понять проблему и предпринять необходимые действия, а также логирование для последующего анализа причин сбоев.

Соответствие пользовательскому интерфейсу. Оценить, насколько интуитивно понятен и удобен пользовательский интерфейс информационной системы по садоводству. Если пользователи успешно выполняют необходимые действия без затруднений, это может свидетельствовать о высокой эффективности разработки. Если же пользователю сложно найти какие-либо элементы или функции на сайте, то следует пересмотреть дизайн сайта и/или удалить не нужные элементы и функции

Надежность безопасности. Так же необходимо уделить внимание проверке системы на наличие уязвимостей и недостатков в области безопасности. Важно убедиться, что информационная система по садоводству защищена от несанкционированного доступа и не представляет рисков для пользователей и окружающей среды. Если же у системы есть какие-то уязвимости безопасности, то может произойти утечка данных клиентов или злоумышленники смогут взломать систему и использовать ее в своих целях и интересах.

Документация и отчётность. Так же нужно проверить качество документации, которая сопровождает информационную систему по садоводству. Она должна быть четкой, понятной и обеспечивать достаточно информации для поддержки и эксплуатации системы. Также убедиться, что созданы отчеты о результатах тестирования, которые содержат достаточно информации для оценки.

Внесение корректировок и улучшений. Результаты тестирования могут выявить слабые места или недочеты в информационной системе по садоводству. Важно, чтобы эти результаты использовались для улучшения разработки. Оцените, насколько эффективно команда разработчиков реагирует на обнаруженные проблемы, вносит исправления и выпускает обновления.

В ходе тестирования была проверена на выявление проблем с производительностью сайта. Это позволило нам определить, какие есть проблемы со скоростью загрузки веб-приложения и что нужно для этого сделать. Для тестирования на производительность использовался онлайн сервис PageSpeed Insights. Результат тестирования изображен на рисунке 26.

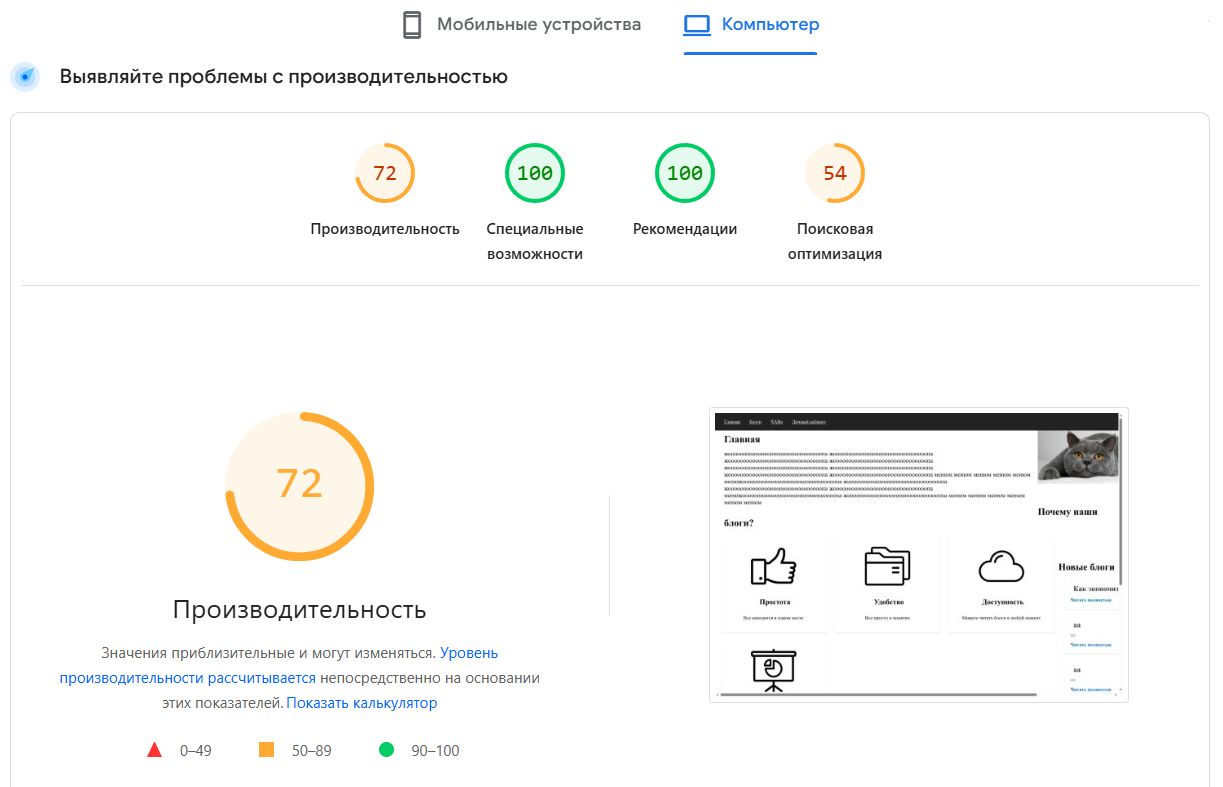


Рисунок 26 – результат тестирования веб-приложения.

Это тестирование показало, что веб-приложения «Блог об электроэнергетике» есть проблемы с производительностью. Проблемы возникли у двух параметров, а именно у Cumulative Layout Shift и у Speed Index.

Cumulative Layout Shift (совокупное смещение макета) – это величина, на которую смещаются видимые элементы области просмотра при загрузке. Чем ниже показатель, тем лучше.

Speed Index (индекс скорости загрузки) показывает, как быстро на странице появляется контент. Так же, чем ниже, тем лучше. Показатели после тестирования изображены на рисунке 27.

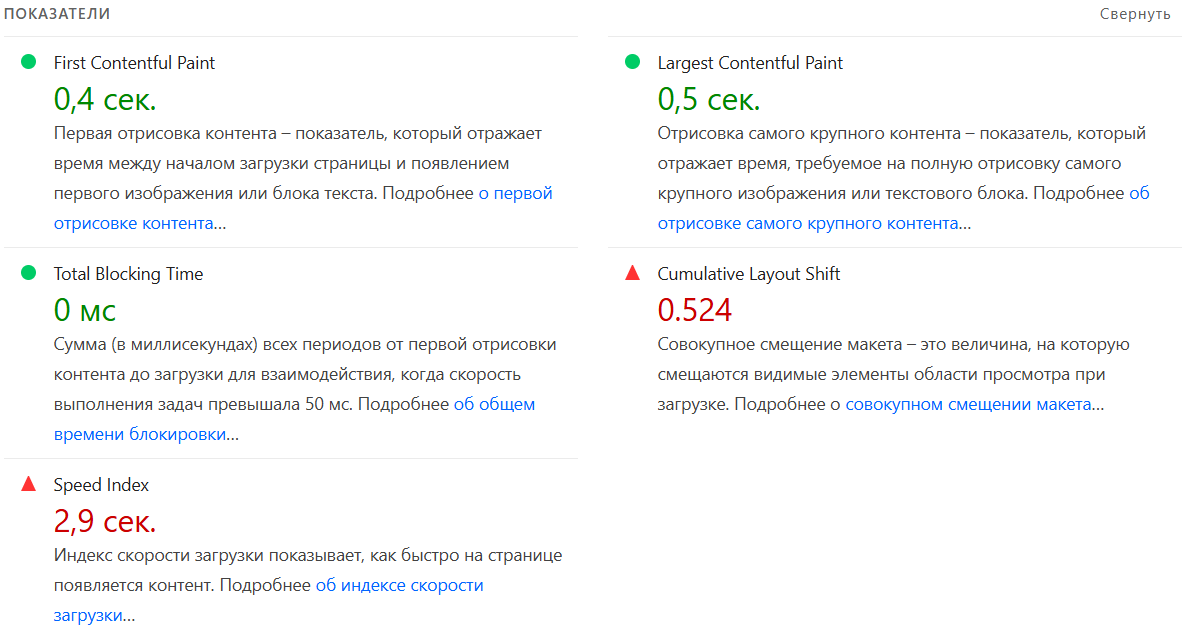


Рисунок 27 – итоговые показатели после тестирования

Промежуточном итоге можно сказать, что в основном веб-приложение прошло все тесты единственное, что необходимо сделать это повысит производительность сайта.

Так же проводилось ручное тестирование, чтобы определить, работают ли все функции в информационной системе для блога об электроэнергетике. Это было необходимо, чтобы убедиться, что все функции системы и работа с базой данных работает корректно и без перебоев. Так же, без ручного тестирования, мы не смогли бы понять, где могут находиться ошибки в информационной системе и это повлияло бы на стабильность работы и привлечение новых пользователей. В тестировании,участвовали регистрация и удаление нового администратора, добавление блога и так далее.

Первым, что необходимо проверить в информационной системе для блога об электроэнергетике – регистрация и удаление нового администратора, работает ли система проверки полей ввода данных. Форма регистрации нового администратора изображена на рисунке 28.

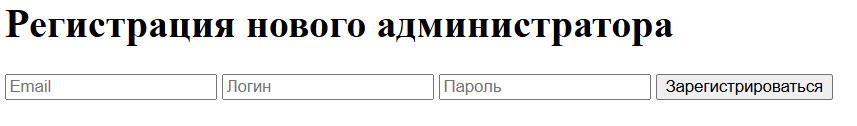


Рисунок 28 – форма регистрации нового администратора

Информационная система для блога об электроэнергетике предоставляет компании возможность зарегистрировать нового администратора в системе. Это позволит новому администратору добавлять, редактировать и удалять блоги Для регистрации в системе, необходимо ввести следующие данные:

– имя пользователя;

– электронная почта;

– придумать пароль.

Если попробовать нажать кнопку «Зарегистрироваться» с пустыми полями ввода, то сайт запросит ввести соответствующие данные. Это сделано для того, чтобы в базе данных не создавались пустые регистрационные данные. Ошибка изображена на рисунке 29.

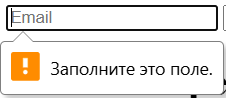


Рисунок 28 – ошибка при вводе пустых значений

Далее мы вводим недостающие данные. Если мы попробуем зарегистрировать пользователя с паролем меньше, чем 8 символов, выдается соответствующая ошибка. На рисунке 29 изображена ошибка с паролем.

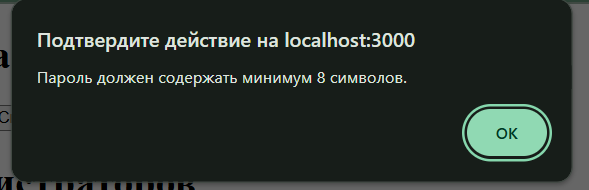


Рисунок 29 – ошибка с паролем

После ввода всех необходимых данных нажимаем кнопку зарегистрироваться. После регистрации нам система скажет, что администратор зарегистрирован, и он уведомлен по почте. В этом сообщении прикреплен файл, в котором указаны все регистрационные данные, такие как логин, пароль и секретный ключ, который генерируется автоматически. На рисунке 30 изображено сообщение после регистрации.

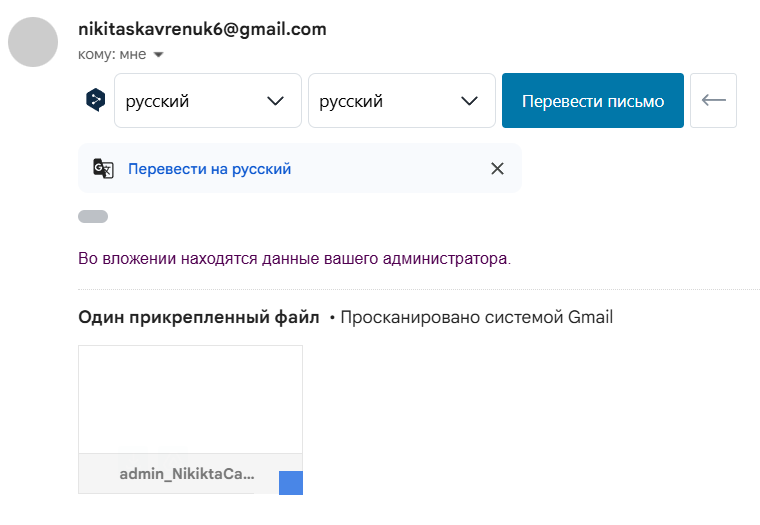


Рисунок 30 – сообщение после регистрации

Теперь мы должны проверить функцию удаления администратора. Эта функция необходима так и примеру, если сотрудник был уволен, то его личный кабинет администратора также необходимо удалить. Для этого на странице регистрации нового администратора есть блок со всеми зарегистрированными администраторами, который изображен на рисунке 31.

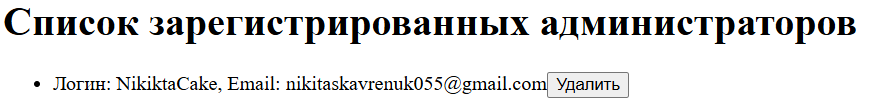


Рисунок 31 – блок со всеми зарегистрированными администраторами

Чтобы утолить администратора, необходимо нажать кнопку удалить. После чего нас спросят, действительно ли хотите удалить администратора, подтверждаем это действие. После удаления администратора, придет на почту сообщения о том, то, что он больше не является администратором на сайте. Данное сообщение изображено на рисунке 32.

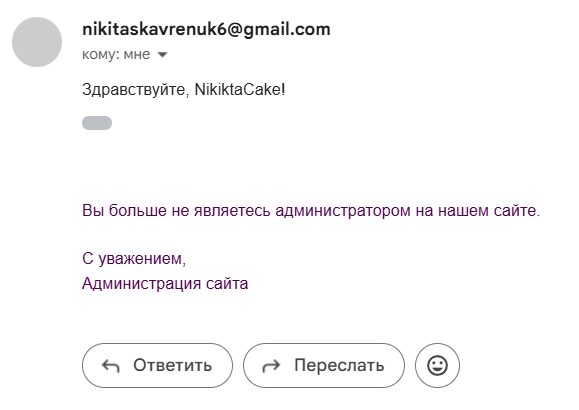


Рисунок 32 – сообщение о том то, что не являетесь администратором на сайте.

В промежуточном итоге, мы можем выделить, что форма регистрации нового аккаунта работает корректно, система не регистрирует аккаунт с пустыми данными, так же система не регистрирует пользователя без электронной почты и пароля, удаление работает так же корректно. Новый пользователь был зарегистрирован и внесен в базу данных администраторов блога об электроэнергетике

Далее необходимо проверить добавление нового блога, Для этого необходимо войти с учетной записи администратора. После входа администратора встретит админ панель, Которая поможет в дальнейшем администратору добавлять, редактировать и удалять блоги. Данная админ панель продемонстрирована на рисунке 33.

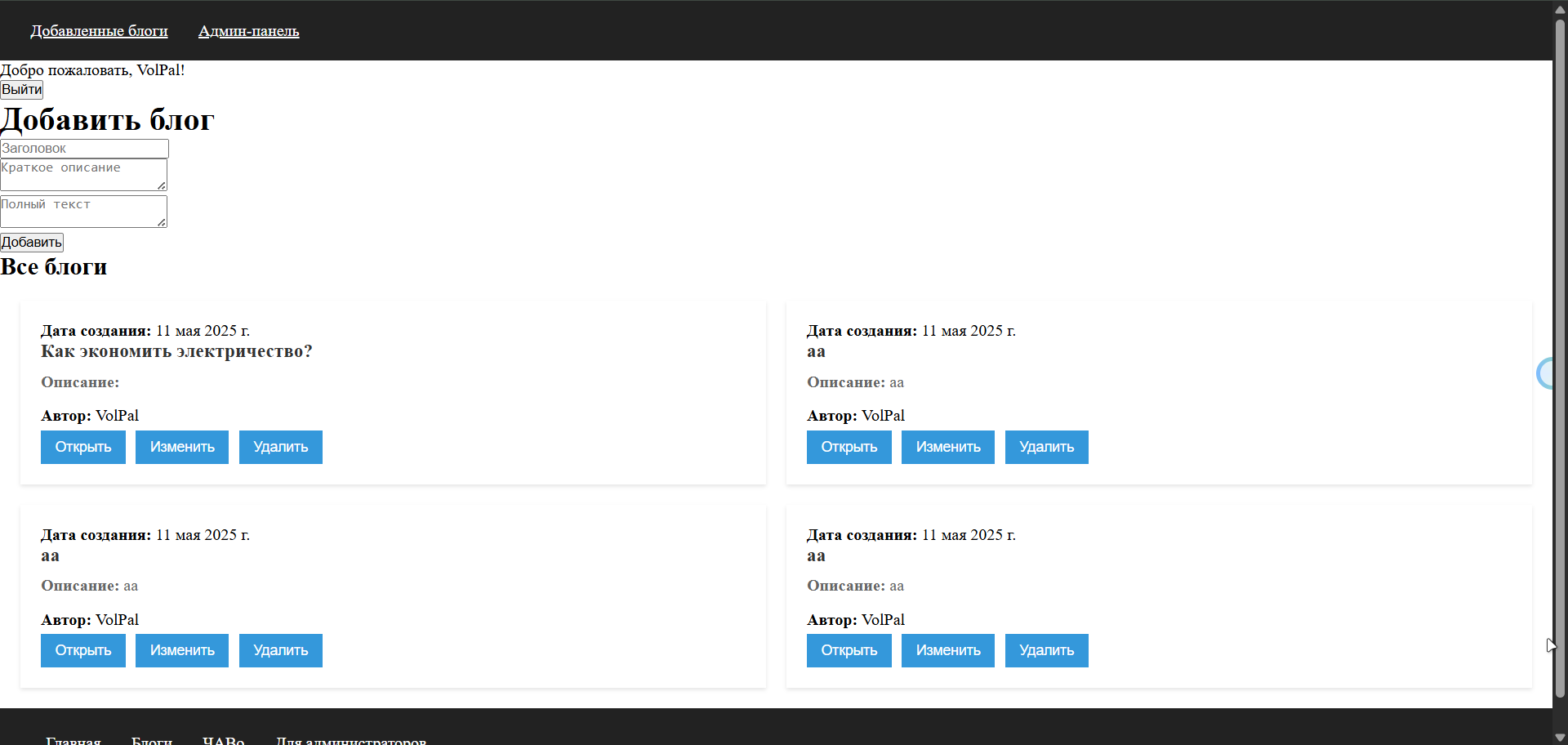


Рисунок 33 – админ панель

Чтобы добавить новый блок, необходимо ввести данные в небольшую форму. Форма имеет 3 пункта заголовок, краткое описание и полный текст. Форма добавления блога изображена на рисунке 34.

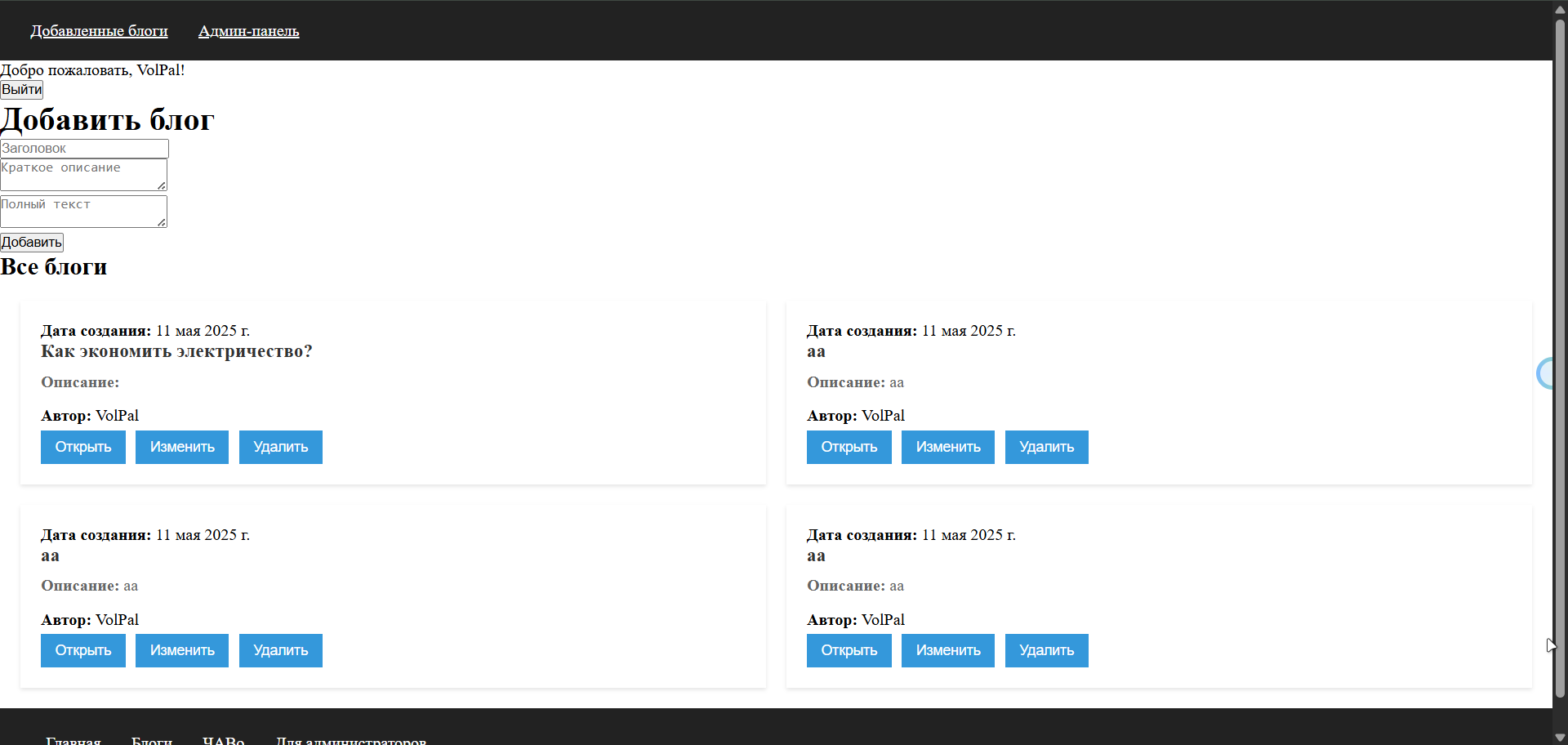


Рисунок 34 – форма добавления блога

После ввода всех необходимых данных форму нажимаем кнопку добавить, после чего новый блог появится ниже.

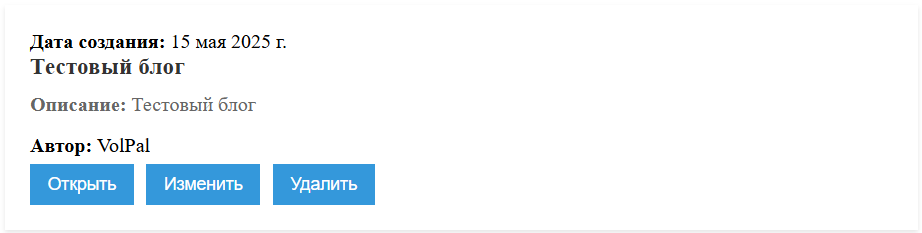


Рисунок 35 – новый блог

В общем итоге, можно выделить следующие выводы, все тесты прошли успешно и без проблем. Регистрация администратора проходит успешно и новый администратор добавляется в базу данных, администратор может может войти в свою учетную запись, блог добавляются в систему. Все тесты завершились успешно, продемонстрировав корректность и стабильность работы системы.

## Заключение

В ходе выполнения преддипломной практики была успешно реализована разработка автоматизированной системы веб-приложения для блога, посвящённого электроэнергетике. Основной целью работы стало создание удобной, доступной и функциональной платформы, обеспечивающей возможность получения, обмена и обсуждения информации в сфере энергетики. Задачи, поставленные в рамках практики, включая проектирование архитектуры, написание руководства для разработчиков и пользователей, анализ и оптимизацию программного кода, а также проведение испытаний системы, были выполнены в полном объёме.

Разработанное веб-приложение позволяет публиковать статьи, оставлять комментарии, ставить оценки, а также взаимодействовать с материалами при помощи удобного пользовательского интерфейса. Благодаря реализации адаптивного дизайна система корректно отображается как на компьютерах, так и на мобильных устройствах. Функционал платформы спроектирован с учётом потребностей целевой аудитории — студентов, преподавателей, специалистов в области энергетики и всех заинтересованных пользователей.

В результате рефакторинга исходный код был улучшен, устранены дублирования, повышена читаемость и структурированность программных модулей, что позволило оптимизировать производительность веб-приложения и упростить его сопровождение. Также была реализована система тестирования, охватывающая ключевые компоненты функциональности, что позволило удостовериться в корректности и устойчивости работы системы при различных условиях эксплуатации.

Разработка и внедрение данной платформы способствует распространению знаний в области электроэнергетики, что особенно актуально в условиях цифровизации образования и модернизации отрасли. Система имеет потенциал для дальнейшего масштабирования и расширения функциональных возможностей: в будущем возможно добавление форумов, системы уведомлений, интеграции с социальными сетями и аналитики поведения пользователей.

Таким образом, реализация проекта показала практическую значимость и целесообразность применения цифровых технологий в области технического образования. Разработанное программное обеспечение успешно решает поставленные задачи, отвечает современным требованиям к веб-приложениям и обладает высоким потенциалом для дальнейшего развития.

## Список используемых источников

1 Как сделать чтобы Footer всегда был внизу, но не фиксированным? // Хабр URL: [https://qna.habr.com/q/540713](https://qna.habr.com/q/540713" \t "_new) (дата обращения: 07.05.2025) – 120 с.

2 CSS прижать footer к низу окна браузера // ProWeb URL: [https://proweb63.ru/help/all-about-css/css-footer-pribit](https://proweb63.ru/help/all-about-css/css-footer-pribit" \t "_new) (дата обращения: 06.05.2025) – 250 с.

3 Основные операции с данными. Добавление данных. Команда INSERT // METANIT.COM URL: [https://metanit.com/sql/sqlite/3.1.php](https://metanit.com/sql/sqlite/3.1.php" \t "_new) (дата обращения: 30.04.2025) – 404 с.

4 Что такое код-ревью и зачем его проводят? // cloud4y URL: [https://www.cloud4y.ru/blog/what-is-code-review/](https://www.cloud4y.ru/blog/what-is-code-review/" \t "_new) (дата обращения: 30.04.2025) – 562 с.

5 9 хороших сервисов проверки кода для разработчиков // GeekBrains URL: [https://gb.ru/blog/9-horoshih-servisov-proverki-koda/](https://gb.ru/blog/9-horoshih-servisov-proverki-koda/" \t "_new) (дата обращения: 30.04.2025) – 315 с.

6 Рефакторинг кода, и как его не бояться // Хабр URL: [https://habr.com/ru/articles/781538/](https://habr.com/ru/articles/781538/" \t "_new) (дата обращения: 30.04.2025) – 275 с.

7 Рефакторинг кода // SkeelFactory URL: [https://blog.skillfactory.ru/glossary/refaktoring-koda/](https://blog.skillfactory.ru/glossary/refaktoring-koda/" \t "_new) (дата обращения: 30.04.2025) – 198 с.

8 Как сменить адрес репозитория Git в Visual Studio для отправки обновлений по новому адресу // StackOverflow URL: [https://ru.stackoverflow.com/questions/Как-сменить-адрес-репозиторий-git-в-visual-studio-для-отправки-обновлений-по-нов](https://ru.stackoverflow.com/questions/%D0%9A%D0%B0%D0%BA-%D1%81%D0%BC%D0%B5%D0%BD%D0%B8%D1%82%D1%8C-%D0%B0%D0%B4%D1%80%D0%B5%D1%81-%D1%80%D0%B5%D0%BF%D0%BE%D0%B7%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D0%B9-git-%D0%B2-visual-studio-%D0%B4%D0%BB%D1%8F-%D0%BE%D1%82%D0%BF%D1%80%D0%B0%D0%B2%D0%BA%D0%B8-%D0%BE%D0%B1%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9-%D0%BF%D0%BE-%D0%BD%D0%BE%D0%B2" \t "_new) (дата обращения: 30.04.2025) – 230 с.

9 Что такое XSS-уязвимость и как тестировщику не пропустить ее // Хабр URL: [https://habr.com/ru/articles/511318/](https://habr.com/ru/articles/511318/" \t "_new) (дата обращения: 30.04.2025) – 290 с.

10 Flexbox: современный подход к верстке // MDN Web Docs URL: [https://developer.mozilla.org/ru/docs/Web/CSS/CSS\_Flexible\_Box\_Layout/Basic\_Concepts\_of\_Flexbox](https://developer.mozilla.org/ru/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox" \t "_new) (дата обращения: 21.04.2025) – 340 с.

11 Основы работы с SQLite: создание и использование баз данных // SQLite Tutorial URL: https://www.sqlitetutorial.net/sqlite-basics/ (дата обращения: 25.04.2025) – 410 с.

12 Организация пользовательских сессий в Express.js // ExpressJS Documentation URL: https://expressjs.com/en/resources/middleware/session.html (дата обращения: 02.05.2025) – 260 с.

13 Создание простого REST API на Node.js и Express // Medium URL: https://medium.com/@xyz/create-simple-rest-api-nodejs-express-12345 (дата обращения: 05.05.2025) – 280 с.

14 Лучшие практики работы с JavaScript Fetch API // JavaScript.info URL: https://javascript.info/fetch-api (дата обращения: 10.05.2025) – 350 с.

15 Как организовать аутентификацию и авторизацию на сайте // Dev.to URL: https://dev.to/abc/how-to-setup-authentication-and-authorization-5678 (дата обращения: 12.05.2025) – 320 с.

16 Позиционирование элементов в CSS: absolute, fixed, relative и static // HTML Academy URL: https://htmlacademy.ru/blog/css/css-positioning (дата обращения: 22.04.2025) – 300 с.

17 Как организовать хранение и обработку комментариев на сайте // habr.com URL: https://habr.com/ru/post/123456/ (дата обращения: 28.04.2025) – 420 с.

18 Введение в работу с базами данных SQLite на Node.js // Node.js Guide URL: https://nodejs.org/en/docs/guides/nodejs-with-sqlite/ (дата обращения: 03.05.2025) – 360 с.

19 Работа с сессиями в Express и хранение данных пользователя // Scotch.io URL: https://scotch.io/tutorials/easy-node-authentication-setup (дата обращения: 08.05.2025) – 330 с.

20 Как использовать Fetch API для отправки POST-запросов // CSS-Tricks URL: https://css-tricks.com/using-fetch/ (дата обращения: 11.05.2025) – 280 с.

21 Создание системы регистрации и логина на Node.js с использованием Express и SQLite // Medium URL: https://medium.com/@xyz/nodejs-express-sqlite-authentication-98765 (дата обращения: 13.05.2025) – 370 с.

22 Работа с формами в HTML и отправка данных через JavaScript // W3Schools URL: https://www.w3schools.com/js/js\_forms.asp (дата обращения: 15.05.2025) – 210 с.

23 Принципы REST API и их реализация на Node.js // FreeCodeCamp URL: https://www.freecodecamp.org/news/rest-api-design-best-practices-build-a-rest-api/ (дата обращения: 14.05.2025) – 400 с.

24 Безопасность веб-приложений: основы XSS и CSRF // OWASP URL: https://owasp.org/www-community/attacks/xss/ (дата обращения: 16.05.2025) – 360 с.

25 Отладка JavaScript-кода в браузере: советы и трюки // Google Developers URL: https://developers.google.com/web/tools/chrome-devtools/javascript (дата обращения: 21.04.2025) – 290 с.