Не трогать

\

2

Не трогать

3

Содержание

Введение………………………………………………………………..……….....4

1. Теоретическая часть………………………………………………..………….5
   1. Техническое задание……………………………………………..………..5
   2. Описание выбранного инструментария……………………………..…...8
   3. Анализ существующих решений в предметной области………..…......10
2. Практическая часть…………………………………………………………...12
   1. Интерфейс…………………………………………………………………12
   2. Описание алгоритма…………………………………………………...…12
   3. Описание кода…………………………………………………………….13
   4. Тестирование…………………………………………………………...…24

Заключение…………………………………………………………………….…28

Список литературы………………………………………………………………29

Приложение А……………………………………………………………………30

Приложение Б………………………………………………………………….....31

Введение

способом получения знаний. В последние десятилетия наблюдается стремительный рост интереса к изучению иностранных языков, что связано с С развитием технологий и широким распространением интернета, система образования претерпевает значительные изменения, и онлайн-обучение становится всё более популярным глобализацией, международной интеграцией и необходимостью общения в многоязычном мире. Знание нескольких языков открывает новые горизонты для профессионального роста, расширяет культурный обмен и способствует более глубокому пониманию различных культур.

Современные студенты и профессионалы ищут удобные и доступные способы изучения языков, и онлайн-платформы предоставляют им такую возможность. Онлайн-обучение предлагает гибкость, позволяя учащимся самостоятельно планировать свое время и выбирать подходящий для себя темп обучения. Использование веб-технологий открывает доступ к разнообразным ресурсам: от видеолекций и интерактивных упражнений до форумов и мессенджеров для общения с носителями языка. Такие платформы способны адаптироваться под индивидуальные потребности и предпочтения каждого ученика, что значительно повышает мотивацию и эффективность обучения.

Целью данной курсовой работы является разработка веб-приложения для системы онлайн-обучения, ориентированной на изучение иностранных языков. В процессе работы будут определены ключевые функциональные компоненты приложения, такие как регистрация пользователей, доступ к учебным материалам, система тестирования и возможность взаимодействия между учащимися и преподавателями. Также будет рассмотрен вопрос о внедрении интерактивных методов обучения, таких как геймификация и использование мультимедиа, что поможет сделать процесс обучения более увлекательным и продуктивным.

В ходе работы будут исследованы современные технологии, используемые для разработки веб-приложений, такие как HTML, CSS, JavaScript, а также фреймворки и библиотеки, которые облегчают создание интерактивных интерфейсов. Будет уделено внимание проектированию пользовательского интерфейса, чтобы обеспечить максимальное удобство и доступность для пользователей. Важно также рассмотреть аспекты безопасности и защиты данных, особенно в контексте хранения личной информации учащихся.

Таким образом, разработка веб-приложения для онлайн-обучения иностранным языкам не только упрощает доступ к образовательным ресурсам, но и предоставляет учащимся возможность получать качественное образование в удобной и комфортной форме. Это приложение станет инструментом, который поможет пользователям эффективно изучать языки, расширять свои горизонты и достигать новых карьерных высот

1. Теоретическая часть

1.1 Техническое задание

2. Краткая характеристика отрасли применения

Система онлайн-обучения предназначена для студентов, учащихся и профессионалов, желающих изучать иностранные языки. Она может быть использована в образовательных учреждениях, языковых школах и в качестве самостоятельного инструмента обучения.

3. Основания для разработки.

3.1. Основания для проведения разработки.

Задание курсовой работы ГБПОУ НСО «Новосибирский химико-техно-

логический колледж им. д. и. Менделеева».

3.2. Наименование и условное обозначение.

Наименование разработки: Веб-приложение для онлайн-обучения иностранных языков. Условное обозначение: ВП-ООИЯ (Веб-приложение — Онлайн-Обучение Иностранным Языкам).

4. Назначение разработки

4.1. Функциональное назначение разработки.

Наименование разработки: Веб-приложение для онлайн-обучения иностранных языков. Условное обозначение: ВП-ООИЯ (Веб-приложение — Онлайн-Обучение Иностранным Языкам).4.2. Эксплуатационное назначение разработки.

4.2. Эксплуатационное назначение разработки.

Для пользования веб-приложения пользователями

4.3. Требования к серверной

- Управление

4.4. Состав выполняемых функций

- Функция кнопки «Вход» на главном экране отправляет на форму для входа в аккаунт;

- Функция кнопки «Регистрация» на главном экране отправляет на форму для регистрации аккаунта;

-Функция кнопки «Зарегистрироваться» Отправляет полученные данные на серверную часть

-Функция кнопки «Войти» Авторизует пользователя

-Функция кнопки «Выйти» Выход из учетной записи

- Функция кнопки «Запись» на главном экране отправляет на форму отзывов

- Функция кнопки «Чат» на главном экране отправляет на страницу чата

- Функция кнопки «Расписания» на главном экране отправляет на страницу расписания

4.5. Организация входных данных.

Запросы в формате JSON посредством REST.API с возможностью за-

грузки файлов из смежной системы.

4.6. Организация выходных данных.

Описание выходных данных аналогично описанию входных данных и

приведено в разделе 4.5 настоящего документа

5. Требования к информационной и программной совместимости.

5.1. Требования к исходным колам и языкам программирования.

Исходные коды программы должны быть написаны на языке JavaScript.

5.2. Требования к программный средствам, используемым программой.

Серверная часть:

- Выбор современного языка программирования и фреймворка для раз-

работки надежной и масштабируемой серверной части приложения;

- Требования к производительности, надежности и безопасности вы-

бранной технологии

Хранение данных:

- Выбор подходящей системы управления базами данных (СУБД) для

эффективного хранения информации о продуктах, заказах, клиентах и других сущностях;

Требования к масштабируемости, надежности и производительности

выбранной СУБД.

Клиентская часть:

- Выбор современного фреймворка или библиотеки для разработки ди-

памичного и удобного пользовательского интерфейса:

- Требования к производительности, адаптивности и пользовательскому

опыту выбранной технологии.

6. Требования к программной документации.

6.1. Состав программной документации.

Руководство пользователя и руководство программиста.

6.2. Календарный план работ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  этапа | Название этапа | Исполнители | Сроки  выполнения | Конечный  результат | Отметка о  выполнении  этапа |
| 1 | Сбор информации, определение целей и задач проекта | Гуров Дмитрий  Андреевич | С 8.11.24  .  По 12.11.24 | Электронный вариант информации | √ |
| 2 | Анализ требований  к веб-приложению | Гуров Дмитрий  Андреевич | С 12.11.24  .  По 15.11.24 | Электронный вариант информации | √ |
| 3 | Оформление технического задания | Гуров Дмитрий  Андреевич | С 15.11.24  .  По 20.11.24 | Техническое задание | √ |
| 4 | Разработка | Гуров Дмитрий  Андреевич | С 20.11.24  .  По 30.11.24 | Техническое задание | √ |
| 5 | Тестирование программного продукта | Гуров Дмитрий  Андреевич | С 30.11.24  .  По 4.12.24 | Тестирование  Приложения  На ошибки | √ |
| 6 | Вывод в эксплуатацию | Гуров Дмитрий  Андреевич | С 4.12.24  .  По 8.12.24 | Готовый программный продукт | √ |

Этапы разработки веб-приложения, исполнители, сроки выполнения этапов, конечный результат и отметка о выполнении (Таблица 1).

7. Условия эксплуатации.

7.1. Климатические условия эксплуатации.

Веб-приложение разработано для работы в стандартных климатических

условиях, соответствующих типичным рабочим условиям офисного помещения. Температурный диапазон: 10-35°С, относительная влажность: 30-70%.

7.2. Требования к видам обслуживания.

Для обеспечения бесперебойной работы веб-приложения необходимо регулярное техническое обслуживание серверной инфраструктуры, обновление программного обеспечения (включая библиотеки и зависимости), мониторинг производительности и резервное копирование данных.

7.3. Требования к численности и квалификации персонала.

Для обслуживания веб-приложения необходимо наличие квалифицированных системных администраторов, специалистов по информационной безопасности и разработчиков, обладающих опытом работы с технологиями

Node.js, PostgreSQL и React. Оптимальная численность персонала - не менее двух специалистов.

8. Порядок контроля и приемки.

8.1. Виды испытаний,

Функциональное тестирование: проверка функциональности веб-приложения, включая корректность отображения базы пользователей, возможность создания аккаунта, возможность войти в аккаунт и выйти.

8.2. Общие требования к приемке.

Интерфейс веб-приложения должен быть интуитивно понятным и удобным для пользователей, чтобы обеспечить приятный пользовательский опыт.

Веб-приложение должно соответствовать заявленным функциональным требованиям, таким как возможность отображения базы пользователей, возможность создания аккаунта, возможность войти в аккаунт, выйти, а также возможность записаться на курс, оставить и просмотреть отзывы.

9. Приложения.

* Инструкция по эксплуатации;
* Руководство программиста.

1.2 Выбор инструментария и его обоснования

1. Языки программирования

JavaScript

JavaScript является основным языком программирования для разработки как клиентской, так и серверной части веб-приложений. Он позволяет создавать динамические и интерактивные веб-страницы, а также обрабатывать данные на сервере. С помощью Node.js JavaScript используется для создания высокопроизводительных серверов, что позволяет разработчикам использовать один язык на всей стеке приложения. Это упрощает разработку, так как позволяет разработчикам использовать одни и те же концепции и синтаксис как на клиенте, так и на сервере.

HTML\\TML (HyperText Markup Language) — это язык разметки, который используется для создания структуры веб-страниц. Он позволяет определить различные элементы страницы, такие как заголовки, параграфы, списки, формы и ссылки. HTML является основой любого веб-приложения и обеспечивает семантическую структуру, которая важна для SEO и доступности.

CSS

CSS (Cascading Style Sheets) — это язык стилей, который используется для описания внешнего вида документа, написанного на HTML. CSS позволяет задавать цвета, шрифты, отступы, размеры и другие визуальные характеристики элементов на странице. С помощью CSS можно создавать адаптивные и отзывчивые интерфейсы, что особенно важно в условиях разнообразия устройств и экранов, с которых пользователи могут получать доступ к веб-приложению.

2. Фреймворки и библиотеки

Express.js

Express.js — это минималистичный веб-фреймворк для Node.js, который упрощает разработку серверных приложений. Он предоставляет множество удобных методов для обработки маршрутов, работы с запросами и ответами, а также управления middleware. Express позволяет разработчикам быстро создавать RESTful API и веб-приложения, обеспечивая гибкость и простоту настройки.

EJS (Embedded JavaScript)

EJS — это шаблонизатор для Node.js, который позволяет генерировать HTML-страницы с динамическим содержимым. Он позволяет встраивать JavaScript-код непосредственно в HTML, что упрощает создание интерфейсов. EJS поддерживает наследование шаблонов, что позволяет создавать сложные интерфейсы с повторно используемыми компонентами.

Helmet

Helmet — это библиотека для обеспечения безопасности приложений на Express.js. Она помогает устанавливать заголовки HTTP, которые защищают приложение от различных типов атак, таких как XSS (межсайтовый скриптинг) и кража данных. Helmet позволяет легко настраивать безопасные заголовки и минимизировать возможные уязвимости.

Bcrypt

Bcrypt — это библиотека для хэширования паролей, которая используется для безопасного хранения пользовательских паролей в базе данных. Она реализует алгоритм хэширования, который делает хранимые пароли защищёнными от атак, таких как атаки по словарю или перебору. Использование Bcrypt гарантирует, что даже в случае утечки данных, пароли остаются защищёнными.

3. База данных

PostgreSQL

PostgreSQL — это реляционная база данных с открытым исходным кодом, известная своей надёжностью, мощными возможностями и поддержкой сложных запросов. Она идеально подходит для хранения структурированных данных, таких как информация о пользователях, ролях и учебных материалах. PostgreSQL поддерживает транзакции, что гарантирует целостность данных, и обеспечивает возможности для работы с большими объёмами информации.

4. Инструменты разработки

Node.js

Node.js — это среда выполнения JavaScript на стороне сервера, которая позволяет разработчикам создавать масштабируемые приложения с высокой производительностью. Node.js использует неблокирующую модель ввода-вывода, что делает его идеальным для работы с асинхронными операциями, такими как обработка запросов и работа с базами данных.

Nodemon

Nodemon — это утилита, которая автоматически перезагружает сервер при внесении изменений в код. Это значительно упрощает процесс разработки, так как разработчикам не нужно вручную перезапускать сервер после каждого изменения.

Git

Git — это система контроля версий, которая позволяет разработчикам отслеживать изменения в коде и работать совместно над проектами. Она обеспечивает возможность отката к предыдущим версиям, создания веток для разработки новых функций и слияния изменений, что делает её неотъемлемой частью современного рабочего процесса.

VS Code

Visual Studio Code (VS Code) — это мощный редактор кода, который предоставляет множество инструментов для разработки, отладки и интеграции с системами контроля версий. Он поддерживает расширения, которые могут улучшить производительность и удобство работы.

5. Обоснование выбора инструментов

Выбор этих языков программирования и инструментов обусловлен несколькими факторами:

- JavaScript и Node.js: Позволяют разрабатывать как клиентскую, так и серверную логику на одном языке, что увеличивает скорость разработки и упрощает поддержку кода.

- Express.js: Обеспечивает гибкость и простоту в создании RESTful API, что идеально подходит для веб-приложений.

- EJS: Упрощает создание динамического контента, что делает его удобным для разработки интерфейсов.

- PostgreSQL: Подходит для хранения структурированных данных и обеспечивает мощные возможности для работы с запросами.

- Безопасность: Использование Helmet и Bcrypt обеспечивает защиту данных пользователей и предотвращает распространённые уязвимости.

Таким образом, выбранные технологии и инструменты обеспечивают создание эффективного, безопасного и удобного веб-приложения для онлайн-обучения иностранным языкам, позволяя пользователям получать доступ к качественным образовательным ресурсам и методам.

1.3.2 Существующие решения в предметной области

Перед тем, как начать разработку программного продукта, важно провести анализ существующих аналогов в соответствующей предметной области:

Duolingo. Duolingo — это многофункциональная платформа для изучения иностранных языков, которая предлагает пользователям интерактивные уроки и упражнения. Она обеспечивает удобный интерфейс и разнообразные методы обучения, включая игры и тесты.

Преимущества:

- Широкий функционал: Duolingo предлагает множество языков и разнообразные уроки, включая чтение, письмо, аудирование и разговорную практику.

- Большое количество пользователей: Платформа имеет значительное количество пользователей, что создает сообщество для обмена опытом и поддержки.

- Интуитивно понятный интерфейс: Легкий в использовании интерфейс делает процесс обучения увлекательным и доступным.

Недостатки:

- Ограниченная глубина изучения: Некоторые пользователи могут найти недостаток в глубоком освоении языка, так как акцент делается на игровые элементы.

- Отсутствие живого общения: Платформа не предоставляет возможности для живого общения с носителями языка.

Для повышения эффективности обучения следует внедрить в приложение функционал чатов и общения с преподавателями или носителями языка, а также уделить особое внимание вопросам безопасности данных.

Busuu. Busuu — это платформа для изучения языков, которая предлагает курсы с возможностью общения с носителями языка и получения обратной связи.

Преимущества:

- Актуальная информация: Платформа предоставляет пользователям актуальные материалы и уроки, адаптированные к уровню знаний.

- Легкость использования: Удобный интерфейс делает процесс обучения доступным для пользователей всех возрастов.

- Возможность обратной связи: Пользователи могут получать комментарии от носителей языка по выполненным заданиям.

Недостатки:

- Ограниченный бесплатный доступ: Некоторые функции доступны только по подписке, что может ограничивать пользователей.

- Недостаточная интеграция: Возможности интеграции с другими системами могут быть ограниченными.

Для удобства пользователей следует внедрить в приложение функционал актуализации материалов и обмена информацией о текущих событиях и изменениях в курсах.

Исходя из проведенного анализа программного обеспечения, можно заключить, что важным функционалом приложения должны быть следующие аспекты:

- Чат и обмен сообщениями с носителями языка и преподавателями.

- Легкость использования и интуитивно понятный интерфейс.

- Интеграция с расписанием занятий и курсами.

- Доступ к актуальным материалам и урокам.

2. Практическая часть

2.1 Интерфейс

Интерфейс в контексте программного обеспечения и веб-приложений — это совокупность средств и методов, с помощью которых пользователи взаимодействуют с системой. Он включает в себя визуальные элементы, такие как кнопки, меню, формы, текст и изображения, а также логические структуры, которые определяют, как эти элементы работают вместе для обеспечения удобного и эффективного взаимодействия. В данном случае включены следующие блоки

- Основная часть – отображает основную информацию.

- Навигационное меню – обеспечивает возможность перемещения по разделам и функциям продукта.

- Кнопки – позволяет быстрый доступ к различным функциям или разделам приложения

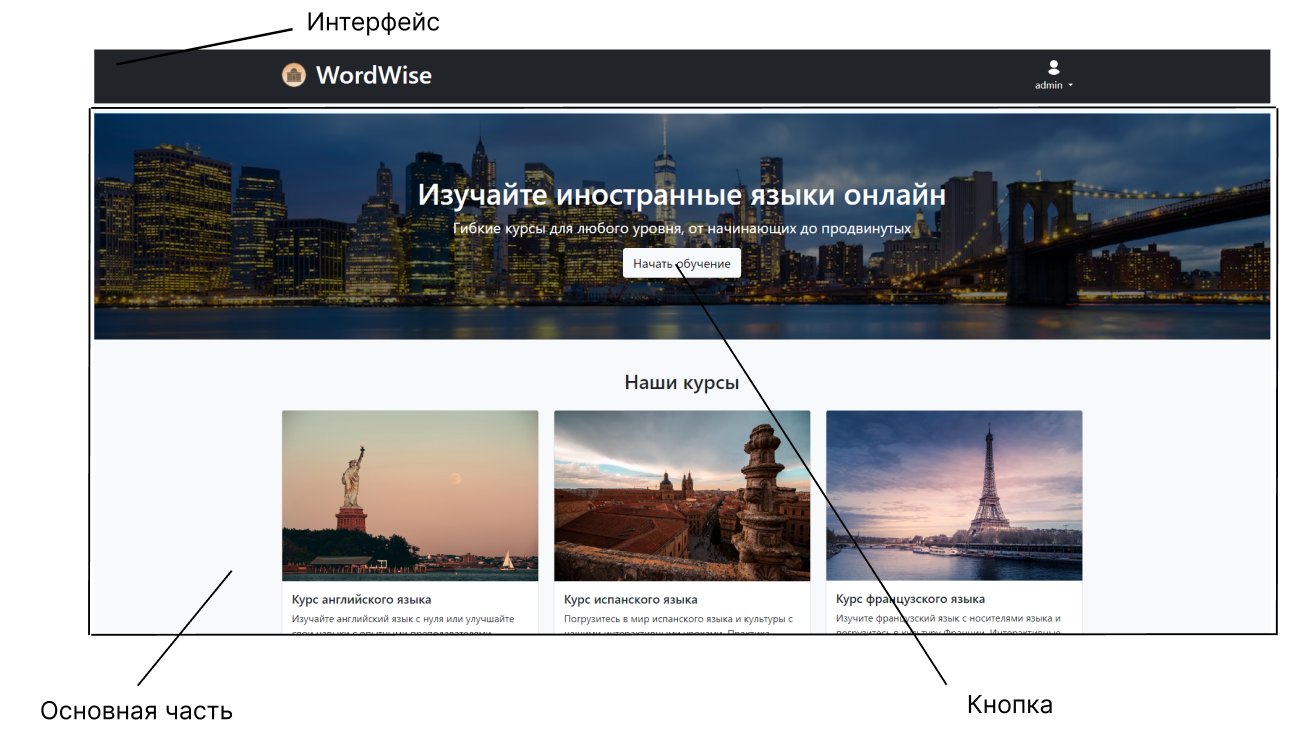


Рисунок 1 – «Интерфейс»

2.2 Описание алгоритма

Данная – диаграмма описывает алгоритм работы программного обеспечения.

После регистрации и авторизации пользователю доступны сервисы (Рисунок 2):

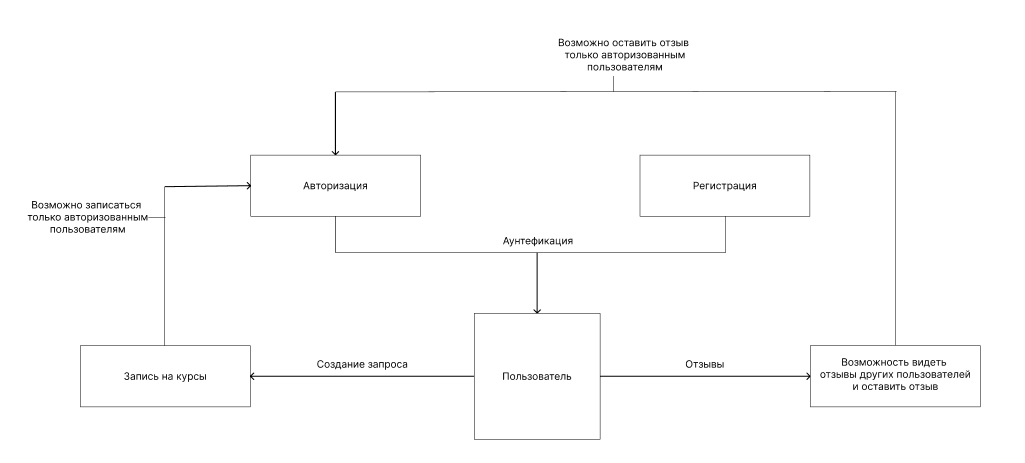


Рисунок 2 – «Алгоритм работы»

2.3 Описание кода

2.3.1 Controllers

Controllers — это ключевой компонент MVC (Model-View-Controller) архитектуры, который отвечает за обработку входящих HTTP-запросов. Они принимают данные от пользователя, вызывают соответствующие методы из моделей для работы с данными и выбирают, какое представление (view) отобразить. Например, если пользователь отправляет форму, контроллер обрабатывает эти данные и может перенаправить на другую страницу или вернуть результат.



Рисунок 3 – «AuthController»

Этот код представляет собой контроллер для аутентификации пользователей в веб-приложении, реализуя функциональность регистрации, входа и выхода

- Управляет аутентификацией: регистрацией, входом и выходом пользователей.

- Импортирует AuthService для бизнес-логики.

- Методы:

- register: обрабатывает регистрацию, проверяет данные, создает пользователя, устанавливает токен и перенаправляет.

- login: аутентифицирует пользователя, устанавливает токен и перенаправляет.

- logout: завершает сессию, удаляет токен и перенаправляет.

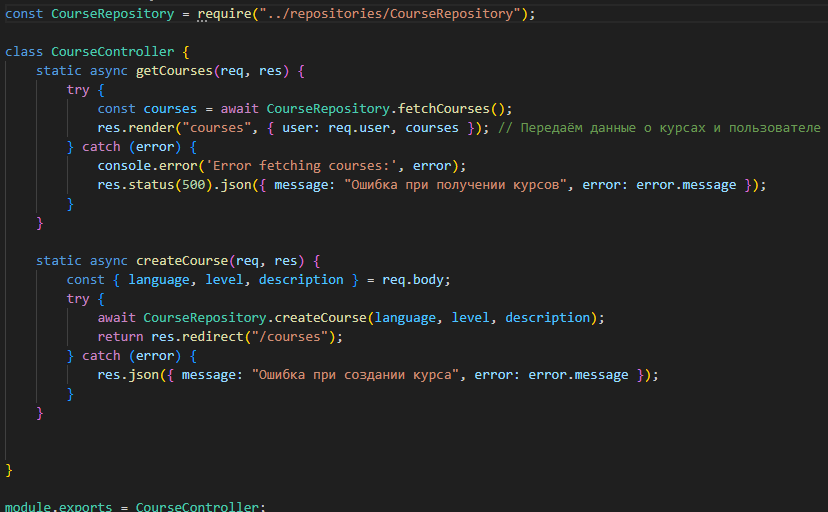


Рисунок 4 – «CourseController»

Этот код представляет собой контроллер для управления курсами в веб-приложении, реализуя функциональность получения и создания курсов.

- Управляет курсами: получение и создание.

- Импортирует CourseRepository для доступа к данным о курсах.

- Методы:

- getCourses: запрашивает курсы и отображает их в представлении.

- createCourse: создает новый курс и перенаправляет на страницу курсов.

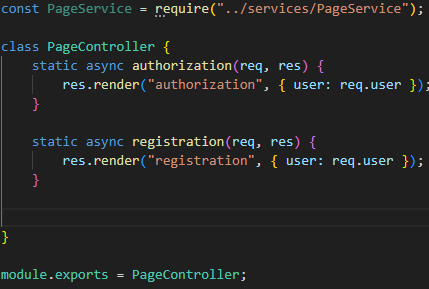


Рисунок 5 – «PageController»

Этот код представляет собой контроллер для управления страницами аутентификации и регистрации пользователей в веб-приложении.

- Управляет страницами аутентификации и регистрации.

- Импортирует PageService, хотя не использует его в данном коде.

- Методы:

- authorization: отображает страницу авторизации.

- registration: отображает страницу регистрации.

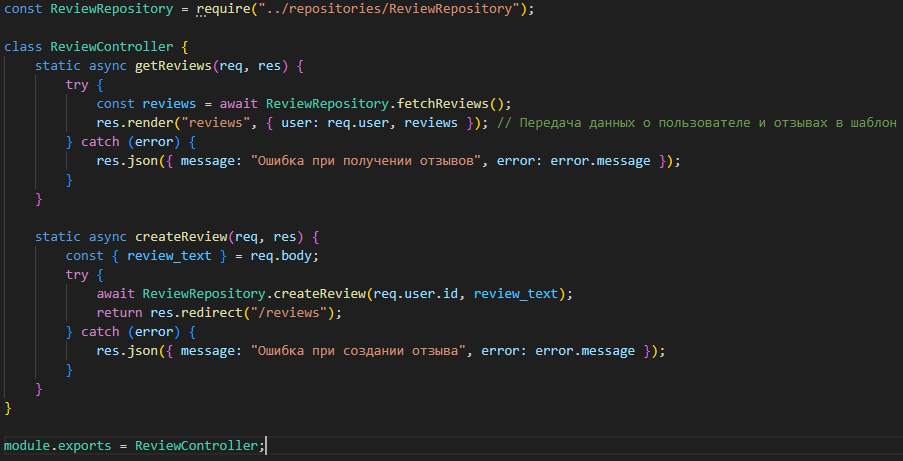


Рисунок 6 – «ReviewController»

Этот код представляет собой контроллер для управления отзывами в веб-приложении, реализуя функциональность получения и создания отзывов.

1. Импорт репозитория: const ReviewRepository = require("../repositories/ReviewRepository"); импортирует репозиторий отзывов, который отвечает за доступ к данным о отзывах.

2. Класс ReviewController: Определяет контроллер, содержащий статические методы для обработки запросов, связанных с отзывами.

3. Метод getReviews:

- Запрашивает отзывы, используя метод fetchReviews из ReviewRepository.

- Если запрос успешен, рендерит представление "reviews", передавая данные о текущем пользователе (из req.user) и полученные отзывы.

- В случае ошибки возвращает сообщение об ошибке в формате JSON.

4. Метод createReview:

- Принимает текст отзыва из тела запроса (req.body).

- Вызывает метод createReview из ReviewRepository для создания нового отзыва, передавая идентификатор пользователя и текст отзыва.

- После успешного создания перенаправляет пользователя на страницу с отзывами.

- Если возникает ошибка, возвращает сообщение об ошибке в формате JSON.

2.3.2 «Repositories»

Repositories служат для организации работы с данными. Они предоставляют абстракцию над слоями доступа к данным, что упрощает управление базой данных. Вместо того чтобы напрямую взаимодействовать с базой данных в контроллерах, разработчики используют репозитории. Это позволяет легче тестировать и изменять логику доступа к данным, а также улучшает читаемость кода.

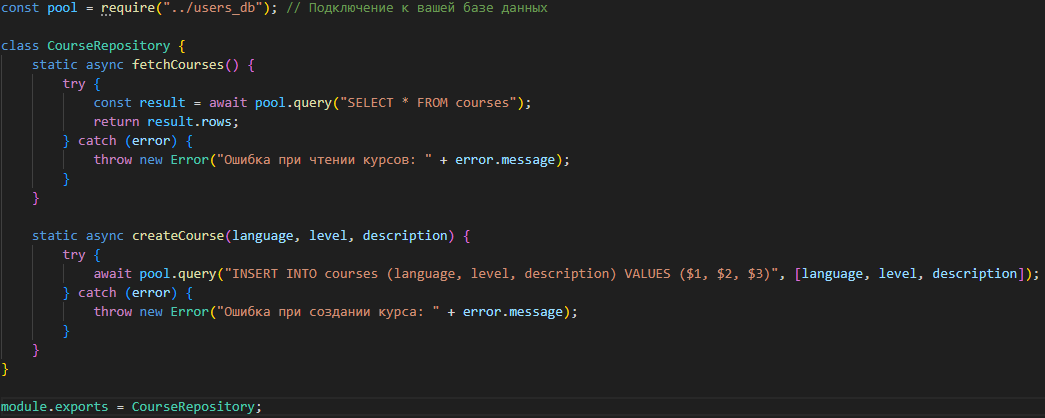


Рисунок 7 – «CourseRepository»

Этот код представляет собой репозиторий для управления курсами в базе данных, реализуя функции получения и создания курсов.

1. Импорт подключения к базе данных: const pool = require("../users\_db"); подключает пул соединений к базе данных, который используется для выполнения запросов.

2. Класс CourseRepository: Определяет репозиторий, который содержит статические методы для взаимодействия с таблицей курсов в базе данных.

3. Метод fetchCourses:

- Выполняет SQL-запрос для получения всех записей из таблицы "courses".

- Если запрос успешен, возвращает строки результата (курсы).

- В случае ошибки выбрасывает исключение с сообщением об ошибке.

4. Метод createCourse:

- Принимает параметры языка, уровня и описания курса.

- Выполняет SQL-запрос для вставки новой записи в таблицу "courses".

- Если возникает ошибка при выполнении запроса - сообщение об ошибке.

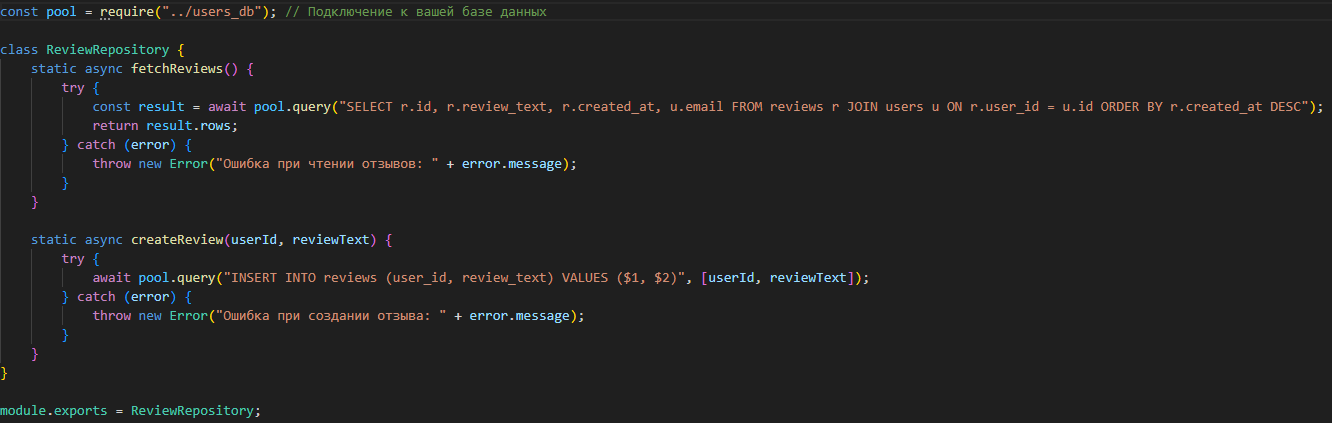


Рисунок 8 – «ReviewRepository»

1. Импорт подключения: const pool = require("../users\_db"); — подключает пул соединений к базе данных.

2. Класс ReviewRepository: Содержит статические методы для работы с таблицей отзывов.

3. Метод fetchReviews:

- Выполняет SQL-запрос для получения всех отзывов, объединяя таблицы "reviews" и "users".

- Возвращает отзывы или выбрасывает исключение при ошибке.

4. Метод createReview:

- Принимает идентификатор пользователя и текст отзыва.

- Выполняет SQL-запрос для добавления нового отзыва или выбрасывает исключение при ошибке.

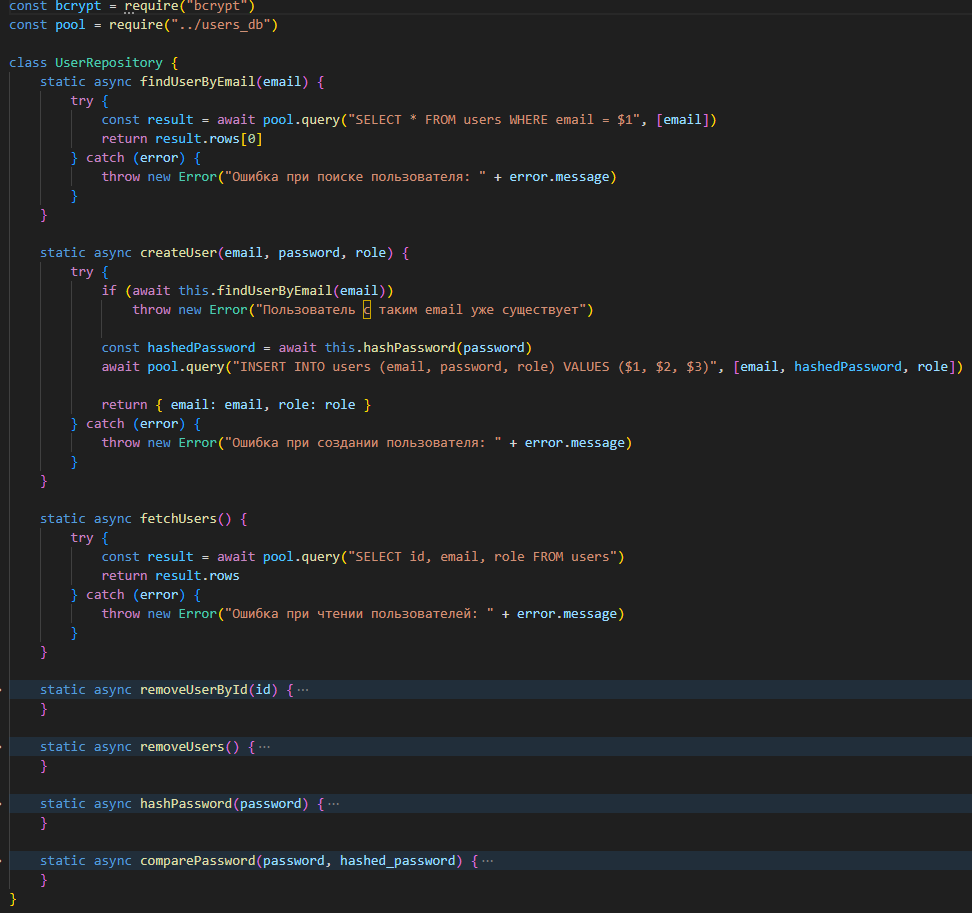


Рисунок 9 – «UserRepository»

Этот код представляет собой репозиторий для управления пользователями в базе данных, реализуя функции поиска, создания, получения и удаления пользователей.

1. Импорт модулей:

- const bcrypt = require("bcrypt") — для хэширования паролей.

- const pool = require("../users\_db") — подключает пул соединений к базе данных.

2. Класс UserRepository: Содержит статические методы для работы с таблицей пользователей.

3. Метод findUserByEmail:

- Ищет пользователя по email и возвращает его или undefined.

4. Метод createUser:

- Проверяет существование пользователя, хэширует пароль и добавляет нового пользователя в базу.

5. Метод fetchUsers:

- Получает всех пользователей из базы.

6. Метод removeUserById:

- Удаляет пользователя по идентификатору.

7. Метод removeUsers:

- Удаляет всех пользователей из таблицы.

8. Метод hashPassword:

- Хэширует пароль с использованием bcrypt.

9. Метод comparePassword:

- Сравнивает введенный пароль с хэшированным.

2.3.3 «Routes»

Routes определяют маршрутизацию в приложении, связывая URL-адреса с конкретными контроллерами и действиями. Это упрощает организацию структуры приложения и позволяет легко добавлять новые страницы или изменять существующие. Например, маршрут может указывать, что запрос на "/users" должен обрабатываться контроллером пользователей.

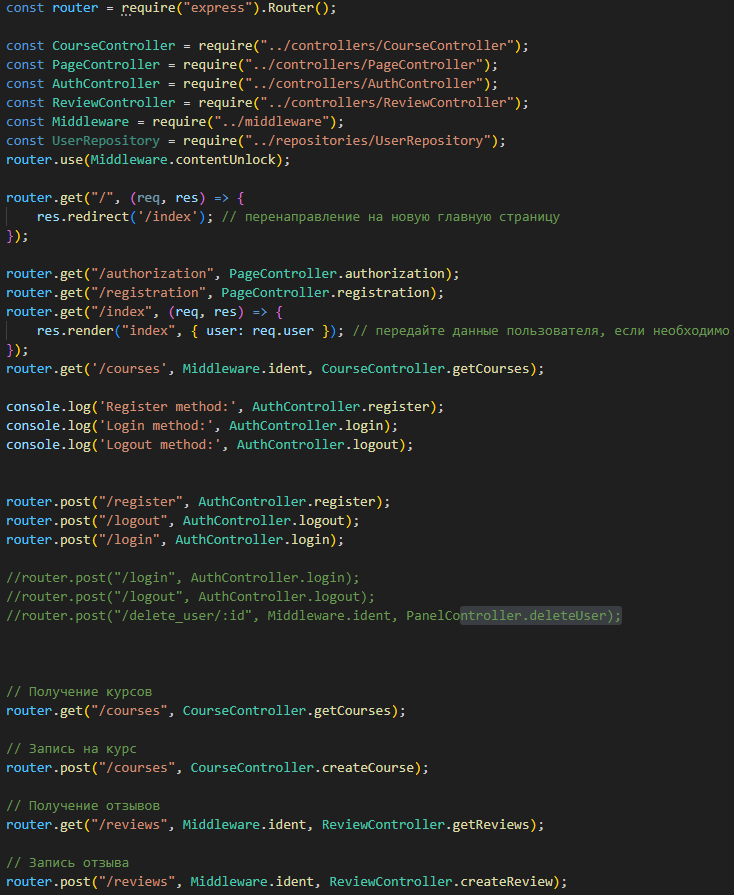


Рисунок 10 – «Routes»

Этот код представляет собой маршрутизатор для веб-приложения на основе Express, который управляет маршрутизацией запросов и связывает их с соответствующими контроллерами.

1. Импорт библиотек и модулей:

- const router = require("express").Router(); — создает новый экземпляр маршрутизатора.

- Импортируются контроллеры для управления курсами, страницами, аутентификацией и отзывами.

- Импортируется промежуточное ПО (middleware) для обработки запросов.

2. Использование промежуточного ПО:

- router.use(Middleware.contentUnlock); — применяет промежуточное ПО, которое, вероятно, управляет доступом к контенту.

3. Определение маршрутов:

- router.get("/", (req, res) => { res.redirect('/index'); }); — перенаправляет корневой маршрут на страницу "index".

- router.get("/authorization", PageController.authorization); — отображает страницу авторизации.

- router.get("/registration", PageController.registration); — отображает страницу регистрации.

- router.get("/index", (req, res) => { res.render("index", { user: req.user }); }); — рендерит главную страницу и передает данные о пользователе.

- router.get('/courses', Middleware.ident, CourseController.getCourses); — запрашивает курсы, используя метод контроллера, и требует аутентификации.

4. Аутентификация:

- router.post("/register", AuthController.register); — обрабатывает запросы на регистрацию.

- router.post("/logout", AuthController.logout); — обрабатывает запросы на выход из системы.

- router.post("/login", AuthController.login); — обрабатывает запросы на вход в систему.

5. Работа с курсами:

- router.post("/courses", CourseController.createCourse); — обрабатывает запросы на создание нового курса.

6. Работа с отзывами:

- router.get("/reviews", Middleware.ident, ReviewController.getReviews); — получает отзывы с обязательной аутентификацией.

- router.post("/reviews", Middleware.ident, ReviewController.createReview); — обрабатывает создание нового отзыва с обязательной аутентификацией.

2.3.4 «Services»

Сервисы в архитектуре веб-приложения выполняют бизнес-логику, отделяя ее от контроллеров и репозиториев. Они обеспечивают чистую и организованную структуру кода, что упрощает его поддержку и тестирование.

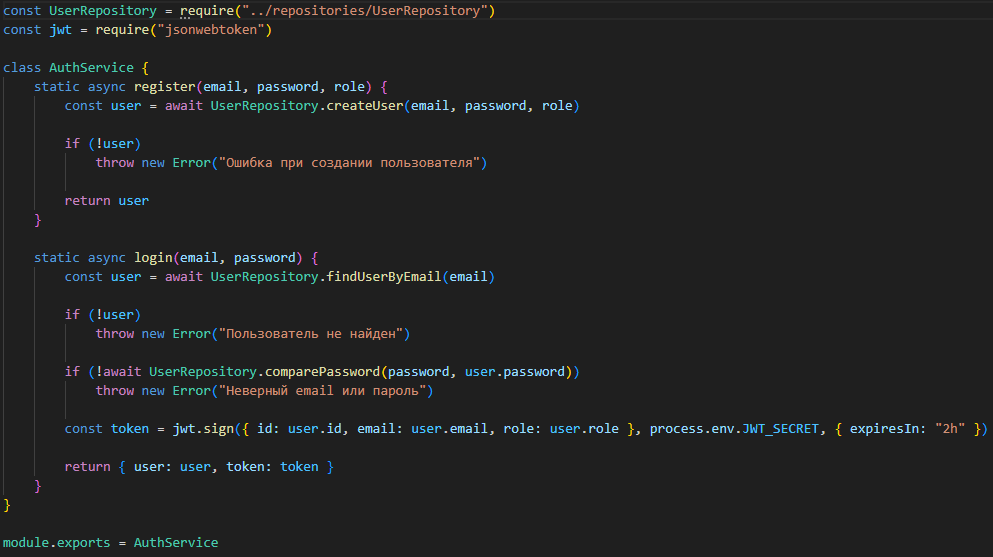


Рисунок 10 – «AuthService»

Этот код представляет собой сервис аутентификации пользователей, реализующий функции регистрации и входа в систему.

1. Импорт модулей: Импортирует UserRepository для доступа к данным пользователей и jsonwebtoken для работы с JWT.

2. Класс AuthService: Содержит методы для аутентификации.

3. Метод register:

- Регистрирует нового пользователя, вызывая createUser.

- Возвращает созданного пользователя или выбрасывает ошибку.

4. Метод login:

- Ищет пользователя по email.

- Сравнивает введенный пароль с хэшированным.

- Генерирует JWT с данными пользователя, сроком действия 2 часа.

- Возвращает объект с пользователем и токеном или выбрасывает ошибку.

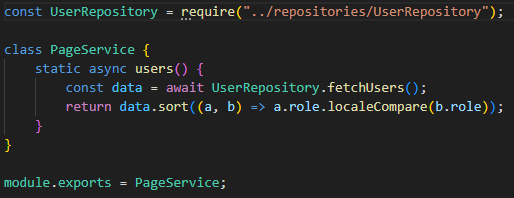


Рисунок 10 – «PageService»

Этот код представляет собой сервис страниц (PageService), который управляет пользователями.

1. Импорт репозитория: Импортирует UserRepository для доступа к данным пользователей.

2. Метод users:

- Вызывает fetchUsers из UserRepository для получения всех пользователей.

- Сортирует пользователей по роли в алфавитном порядке.

- Возвращает отсортированный список пользователей.

2.3.5 «Server»

Server.js в веб-приложении обычно отвечает за настройку и запуск сервера. Основные функции включают:

1. Инициализацию приложения: Создает экземпляр приложения (например, с помощью Express).

2. Настройку маршрутов: Импортирует и связывает маршрутизаторы.

3. Подключение к базе данных: Устанавливает соединение с базой данных.

4. Настройку промежуточного ПО: Подключает необходимые middleware.

5. Запуск сервера: Устанавливает сервер на прослушивание порта.

6. Обработку ошибок: Управляет ошибками на уровне сервера.

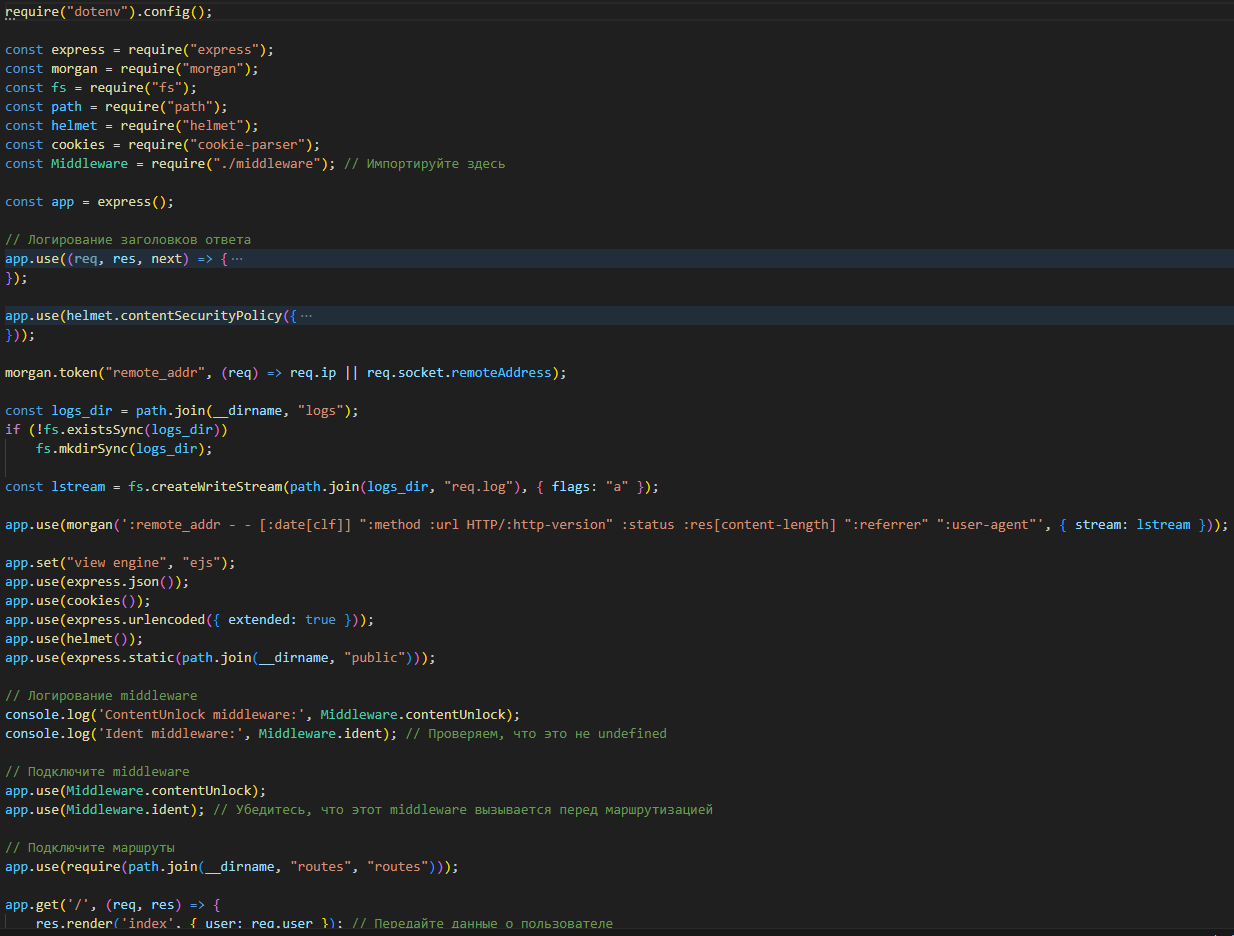


Рисунок 11 – «Server»

Этот код настраивает веб-сервер с использованием Express и выполняет следующие функции:

1. Импорт модулей: Загружает необходимые библиотеки и конфигурации.

2. Настройка приложения:

- Логирует заголовки ответа.

- Настраивает политику безопасности с помощью Helmet.

- Создает директорию для логов и настраивает логирование запросов с помощью Morgan.

3. Настройка middleware:

- Устанавливает EJS как движок шаблонов.

- Обрабатывает JSON и URL-кодированные данные.

- Использует cookie-parser.

- Подключает пользовательские промежуточные функции (middleware).

4. Подключение маршрутов: Загружает маршруты из файла routes/routes.

5. Главный маршрут: Отображает главную страницу и передает данные о пользователе.

6. Запуск сервера: Запускает сервер на порту, указанном в переменных окружения.

7. Маршрут для курсов: Обрабатывает запросы на /courses и отображает страницу курсов.

2.3 Тестирование

Таблица 2 – «Тест-кейс №1»

|  |  |
| --- | --- |
| Номер | 1 |
| Заголовок | Регистрация |
| Предусловие | http://localhost:3044/registration |
| Шаг | Ожидаемый результат: |
| В доступной странице вводим свои данные в формы «Электронная почта» и «Пароль | Если введено все правильно – переносит на главную страницу и заносит данные в базу.  Если введено неправильно (Пропуск формы или неправильная почта) - ошибка |

Результат (Рисунок 13);

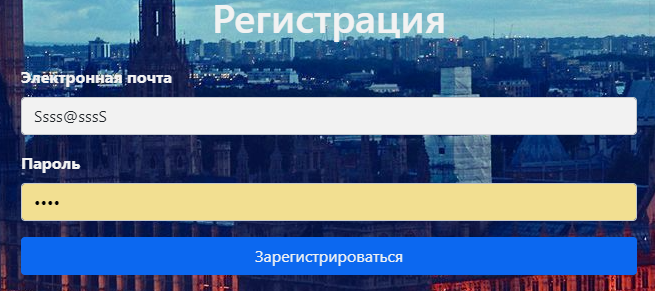


Рисунок 12 – «Регистрация»

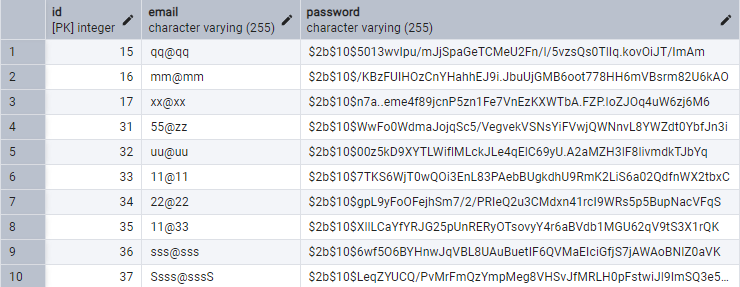


Рисунок 13 – «База данных пользователей»

Таблица 3 – «Тест-кейс №2»

|  |  |
| --- | --- |
| Номер | 2 |
| Заголовок | Вход |
| Предусловие | http://localhost:3044/authorization |
| Шаг | Ожидаемый результат: |
| В доступной странице вводим свои данные в формы «Электронная почта» и «Пароль | Если введено все правильно – переносит на главную страницу.  Если введено неправильно -(Пропуск формы или неправильная почта или пароль) - ошибка |

Результат (Рисунок 14);

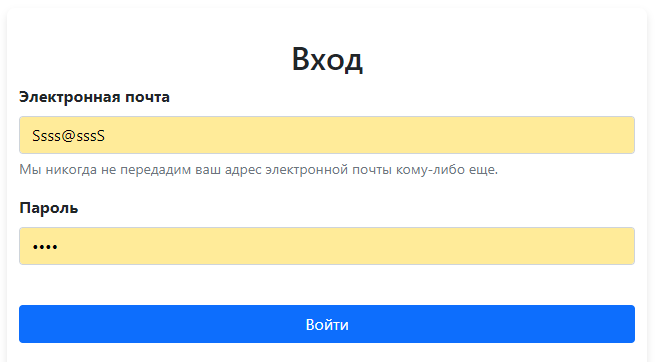


Рисунок 13 – «Вход»

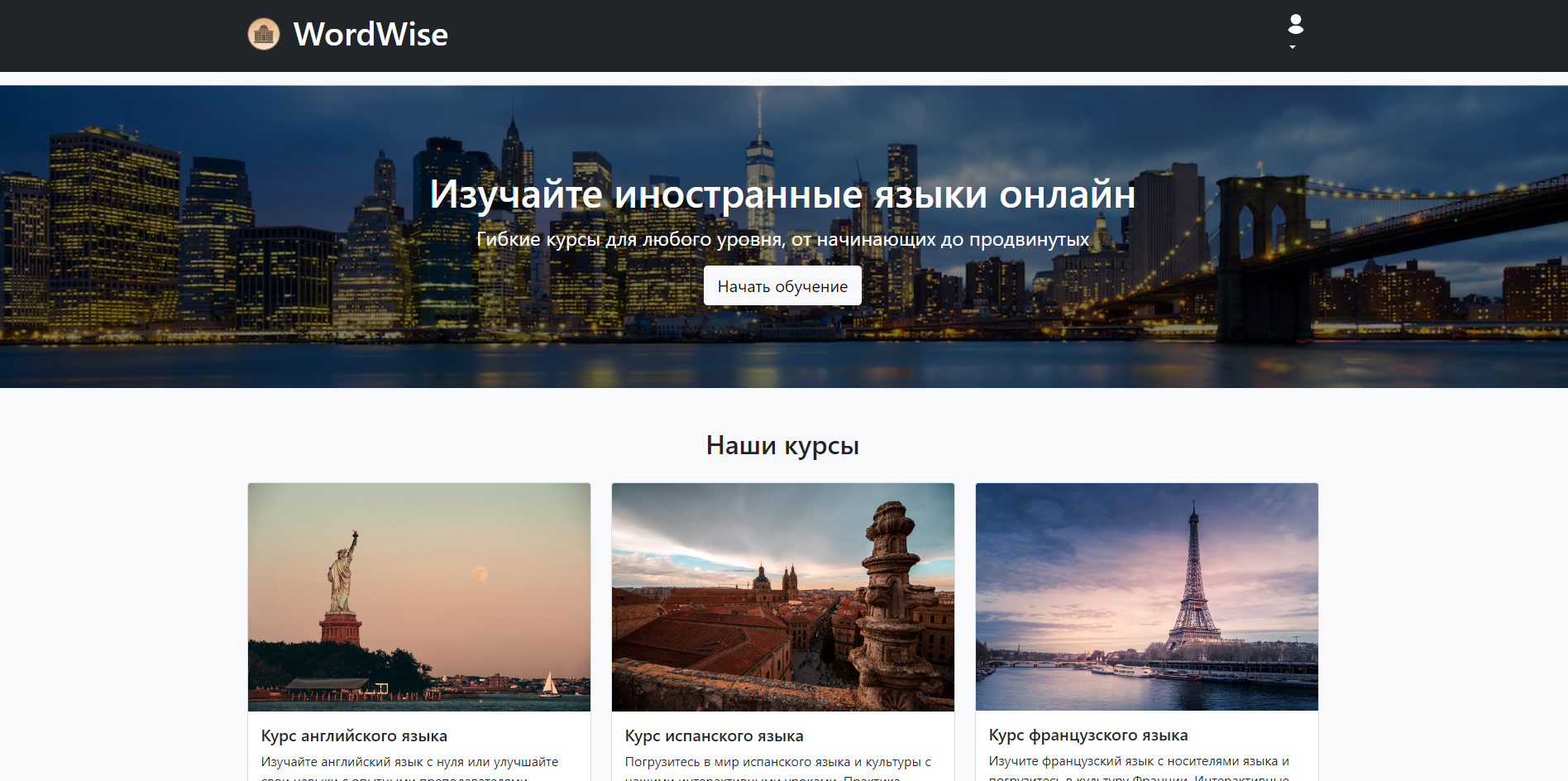


Рисунок 14 – «Главная страница»

Таблица 4 – «Тест-кейс №3»

|  |  |
| --- | --- |
| Номер | 3 |
| Заголовок | Запись на курсы по иностранным языкам |
| Предусловие | Обязательно быть авторизованным  http://localhost:3044/courses |
| Шаг | Ожидаемый результат: |
| В доступной странице заполняем форму записи.  В пункте «Язык» выбираем нужный из предложенных.  В пункте «Уровень» выбираем нужный из предложенных.  Пункт «Описание» - не обязателен.  После нажимаем «Записаться» | Запись создана и отправлена в базу.  Записи показываются на сайте. |

Результат (Рисунок 15);

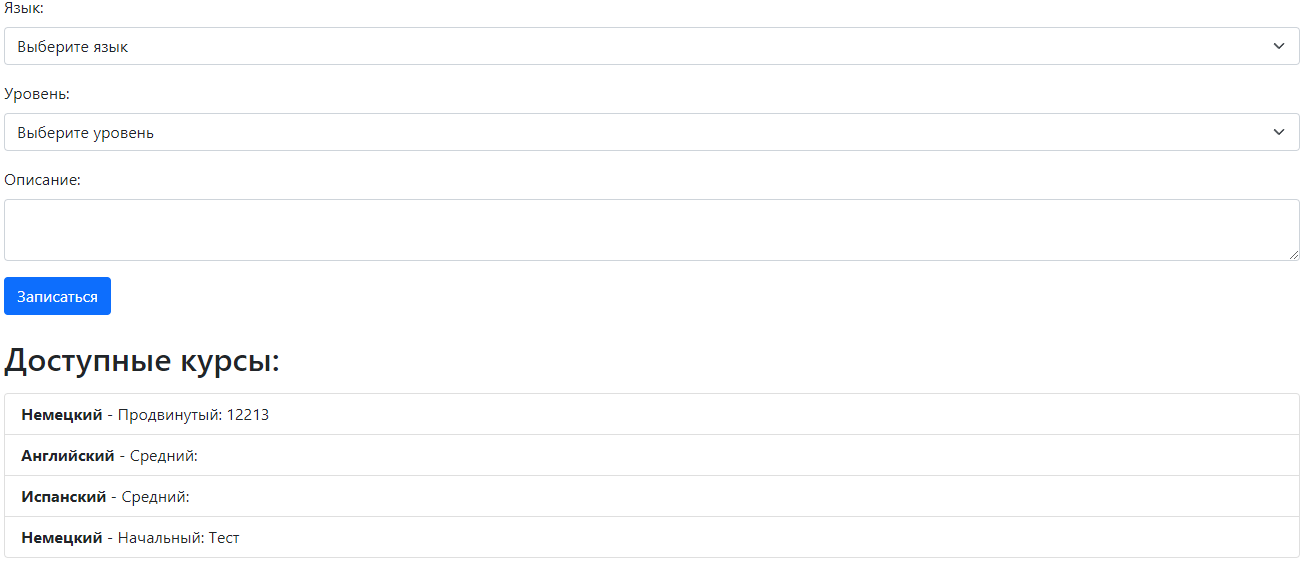


Рисунок 14 – «Запись на курсы»

Таблица 5 – «Тест-кейс №4»

|  |  |
| --- | --- |
| Номер | 3 |
| Заголовок | Чат |
| Предусловие | Обязательно быть авторизованным  http://localhost:3044/reviews |
| Шаг | Ожидаемый результат: |
| В доступной странице заполняем форму «Сообщение».  После нажимаем «Отправить» | Запись создана и отправлена в базу.  Сообщение показывается на сайте. |

Результат (Рисунок 16);

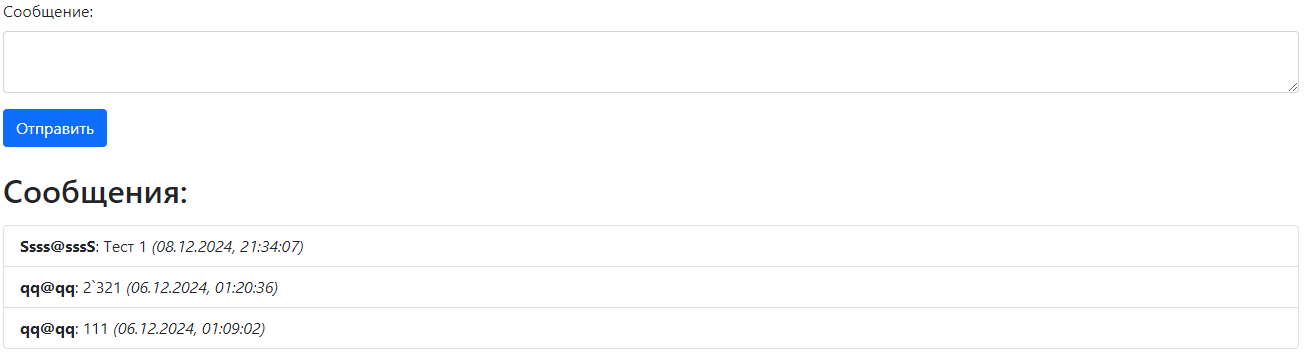


Рисунок 16 – «Чат»

Заключение

По результатам данной курсовой работы было разработано программное обеспечение для онлайн-обучения иностранных языков. Приложение предоставляет функционал рассылки уведомлений об изменениях в расписании занятий, регистрацию, чата для коммуникации, систему записи пользователей на курсы.

Оценка инструментов программирования показала, что использование Java с фреймворком Express для создания backend является оптимальным выбором для разработки данного приложения. Хотя приложение успешно реализует основной функционал, всегда есть возможность улучшения и расширения его возможностей. Например, можно добавить новые функции для интерактивного обучения, улучшить систему уведомлений и интеграцию с другими платформами, такими как видеоконференции и мессенджеры.

В целом, данный курсовой проект позволил применить теоретические знания по программированию на практике, освоить использование современных технологий разработки веб-приложений и создать полезное программное обеспечение, способное оптимизировать процесс изучения иностранных языков.

Можно утверждать с уверенностью, что основная цель: разработка веб-приложения для онлайн-обучения языкам достигнута. Все задачи выполнены, включая:

- Проектирование базы данных для хранения информации о расписании занятий, пользователях, сообщениях в чате, записи на курсы и других сущностях.

- Настройка парсера для получения актуальной информации о расписании с официальных образовательных ресурсов.

- Создание API для взаимодействия веб-приложения с базой данных.

-Разработка веб-приложения с учетом требований безопасности, производительности и удобства использования.

-Тестирование и отладка разработанных приложений.

-Подготовка документации и отчетности по результатам работы.

Список литературы

1. Flanagan, D. (2020). JavaScript: The Definitive Guide. 7th Edition. O'Reilly Media.

2. Node.js Foundation. (n.d.). Node.js Documentation. Retrieved from https://nodejs.org/en/docs/

3. StrongLoop. (2015). Express - Node.js web application framework. Retrieved from https://expressjs.com/

4. EJS. (n.d.). Embedded JavaScript. Retrieved from https://ejs.co/

5. PostgreSQL Global Development Group. (n.d.). PostgreSQL Documentation. Retrieved from https://www.postgresql.org/docs/

6. Git. (n.d.). Git Documentation. Retrieved from https://git-scm.com/doc

7. Microsoft. (n.d.). Visual Studio Code Documentation. Retrieved from https://code.visualstudio.com/docs

Приложение А

Ссылка на приложение:

Приложение Б

Инструкция по эксплуатации (Таблица 6)

|  |  |
| --- | --- |
| Регистрация | 1.Перейдите на страницу регистрация  2. Введите логин. (обязательно с @)  3. Введите пароль.  4. Нажмите зарегестироваться. |
| Авторизация | 1.Перейдите на страницу Авторизация.  2. Введите логин.  3. Введите пароль.  4. Нажмите авторизация.  5. При ошибке попробуйте снова, вы где то ошиблись. |
| Чат | 1. Обязательно авторизуйтесь.  2.Перейдите на страницу Чат.  3. Вы можете общаться с другими пользователями в реальном времени.  4. Для отправки сообщения нажмите «отправить» |
| Запись | 1. Обязательно авторизуйтесь.  2. Перейдите на страницу Запись.  2. Заполните форму записи. |
| Расписание | 1. Перейдите на старицу расписание.  2. Следите за расписанием что бы не пропустить занятие. |