**Министерство просвещения Республики Казахстан**

**Колледж TOO «Astana IT University»**

**Отчет**

**по производственной практике**

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_ Сейткерім Сабыржан Алтынтайұлы\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Группа\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ПО2207\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |
| --- | --- | --- |
|  |  |  |

**Астана 2024г.**

**СОДЕРЖАНИЕ**

|  |  |  |  |
| --- | --- | --- | --- |
| Введение | | | 2 |
|  | Собрание необходимого материала, используемого в работе | | 3 |
|  | Изучение программного обеспечение и технологий используемые в работе | | 5 |
|  | Сбор и анализ требований к производственной работе (проекту) | | 6 |
|  | Сбор и анализ требований к проекту. | | 10 |
|  | Разработка программного обеспечения (сборка компонент ПО) | | 14 |
|  |  |
| Заключение | | | 21 |
| Список использованной литературы | | | 22 |

**Введение**

Поизводственная работа является важным этапом подготовки студента к получению сомастоятельного опыта в области разработки програмного продукта. Она представляет собой исследовательский и практический проект, в ходе которого студент закрепляет теоретические знания и приобретает практические навыки в данной области. Производственная работа дает студенту возможность погрузиться в разработку собственного ПО.

Целью производственной работы является изучение основных аспектов разработки, а также приобретение практического опыта в этой области. Производственная работа была выполнена на основе практической и иследовательской деятельности во время теоретического обучения. В результате были закреплены ключевые компетенции и получен ценный опыт, необходимый для будущей профессиональной деятельности в области разработки программного обеспечения.

Для реализации целей производственной работы было предоставлено индивидуальные задачи для каждого студента. Индивидуальные задачи содержит следующее пункты:

1. Собрание необходимого материала, используемого в курсовой работе.
2. Изучение программного обеспечение используемые в курсовой работе.
3. Сбор и анализ требований к курсовой работе (проекту).
4. Создание компонент программного обеспечения (кодирование, отладка)
5. Разработка программного обеспечения (сборка компонент ПО)

**1 Собрание необходимого материала, используемого в курсовой работе**

* 1. **Введение в разработку приложений**

В данном подразделе собрана информация о современных методах и инструментах, используемых при разработке приложений. Были проанализированы основные этапы жизненного цикла разработки программного обеспечения, включая планирование, проектирование, реализацию, тестирование и развертывание. Также рассмотрены принципы объектно-ориентированного программирования и архитектурные шаблоны, применяемые при разработке приложений.

* 1. **Функции для платформы отельного бизнеса**

В этом подразделе представлен анализ основных функциональных требований к платформе для просмотра культурных событий и покупки билетов. Были определены основные функции, необходимые для удобного и эффективного использования приложения конечными пользователями. Среди таких функций: просмотр актуальной информации о номерах отеля, поиск отелей по различным критериям, покупка билетов онлайн, оплата через различные платежные системы, создание и управление личным профилем пользователя. Кроме того, проведен анализ дополнительных функций, которые могут повысить удобство использования приложения и привлечь больше пользователей.

**2 Изучение программного обеспечение и технологий используемые в курсовой работе**

Программное обеспечение в процессе работы зависит от характера выполняемой работы. Программное обеспечение можно условно разделить на основное, общее, и специализированное, необходимое для выполнения.

Программное обеспечение, установленное на компьютере представлено в «Таблица 1».

|  |  |
| --- | --- |
| **Вид программного обеспечения** | **Название программного обеспечения** |
| *Основное программное обеспечение* | |
| Операционная система | Windows 10 |
| Офисный пакет | Microsoft Office |
| Браузер | Google Chrome |
| *Специализированное программное обеспечение* | |
| Системы управления версиями | Git |
| Интегрированные среды разработки | Visual Studio Code |
| Программное обеспечение для работы с REST API | Postman |
| Программное обеспечения для работы с БД | PgAdmin |
| *Основное программное обеспечение* | |
| Операционная система | Windows 10, Ubuntu 22.04 |
| Антивирусная защита | Kaspersky |
| Офисный пакет | Microsoft Office |
| Браузер | Google Chrome, Mozila |
| *Специализированное программное обеспечение* | |
| *1. Для дизайна* | |
| Графический редактор | Figma |
| *2. Для программирования* | |
| Текстовый редактор | Visual Studio Code |
| Программное обеспечения для работы с БД | PgAdmin |

Таблица 1-Используемое программное обеспечение

**3 Сбор и анализ требований к производственной работе (проекту)**

В данном разделе представлен процесс сбора и анализа требований к приложению. Определение ключевых параметров и функциональных возможностей является критическим этапом для успешной разработки приложения.

Сбор требований:

Первый этап включал в себя определения основных целей приложения. В ходе работы были выделены следующие ключевые требования:

1. Цель приложения: Определение основной цели веб-сайта, такой как представление компании, продажа товаров или услуг, информационный ресурс и др.
2. Целевая аудитория: Идентификация целевой аудитории для оптимизации пользовательского опыта и адаптации контента.
3. Основные функциональности: Определение основных функций, таких как регистрация пользователей, поиск по сайту, интерактивные формы обратной связи и другие.
4. Дизайн и визуальное оформление: Уточнение предпочтений в дизайне, цветовой гамме, логотипе и других визуальных элементах.
5. Анализ требований: На этапе анализа требований проведен более детальный разбор выявленных потребностей для формирования концепции будущего веб-сайта.
6. Функциональный анализ: Разработка детальных сценариев использования, описание функций и взаимодействий между различными элементами сайта.
7. Структура приложения: Создание структуры приложения с учетом логического размещения информации, формирование навигационной системы.
8. Технические требования: Определение технических параметров, таких как выбор языков программирования, базы данных и других технологических аспектов.
9. Требования к безопасности: Установление мер безопасности для защиты данных пользователей и предотвращения возможных угроз.

Во время работы курсовой работы, были выполнены различные задачи, направленные на применение теоретических знаний и навыков в разработке программного обеспечения.

Одной из основных задач было, на основе требований к программному продукту, создание инструментов с использованием фремворка React. Были разработаны прототипы и демонстрационные проекты, применяя знания html, css, javascript.

В процессе выполнения задачи была поставлена важная цель: разработка инструментов на базе требований к программному продукту с использованием PosgreSQL и Node.js.

React — это библиотека JavaScript, разработанная компанией Facebook, предназначенная для создания пользовательских интерфейсов. Основной упор в React делается на эффективность и удобство разработки интерактивных пользовательских интерфейсов. Вот несколько ключевых концепций и особенностей React:

1. Компоненты: Основная идея React — это разделение пользовательского интерфейса на небольшие, независимые компоненты. Каждый компонент представляет собой часть интерфейса и может содержать в себе логику и структуру. Эти компоненты легко могут быть повторно использованы и поддерживаемы.
2. Виртуальный DOM: React использует виртуальный DOM (Document Object Model) для оптимизации обновлений интерфейса. Когда состояние компонента изменяется, React создает виртуальное представление DOM, сравнивает его с текущим состоянием и обновляет только те части интерфейса, которые действительно изменились. Это позволяет сделать процесс обновления более эффективным.
3. JSX: Это расширение синтаксиса JavaScript, используемое в React для описания структуры пользовательского интерфейса. JSX делает код более понятным и удобным для написания. Он позволяет описывать компоненты в виде XML-подобных элементов.
4. Однонаправленный поток данных: Данные в React обычно передаются вниз по иерархии компонентов. Это означает, что родительский компонент может передавать данные дочернему компоненту, но не наоборот. Это сделано для упрощения отслеживания изменений и управления состоянием.
5. Состояние и свойства: Компоненты могут иметь состояние (state), которое может изменяться в процессе работы приложения. Кроме того, компоненты могут принимать входные данные, называемые свойствами (props), которые передаются им из родительских компонентов.
6. Жизненный цикл компонентов: React предоставляет методы жизненного цикла, которые позволяют компонентам выполнять определенные действия на разных этапах своей жизни, таких как момент создания, обновления или удаления.
7. React Hooks: Это новое введение в React, предоставляющее возможность использовать состояние и другие возможности React в функциональных компонентах, ранее доступных только классовым компонентам.

PostgreSQL - это мощная объектно-реляционная система управления базами данных (СУБД), разработанная для обработки больших объемов данных и обеспечения надежности, производительности и расширяемости. Вот более подробное описание PostgreSQL:

1. Открытый исходный код: PostgreSQL является проектом с открытым исходным кодом, что означает, что его исходный код доступен для общественности и может быть свободно изменен и распространен. Это способствует активному сообществу разработчиков и обеспечивает постоянное развитие и улучшение СУБД.
2. Объектно-реляционная модель данных: PostgreSQL предоставляет мощные возможности для работы с данными, основанными на объектно-реляционной модели. Это позволяет разработчикам создавать сложные и эффективные базы данных, которые могут содержать различные типы данных и устанавливать связи между ними.
3. Поддержка стандартов SQL: PostgreSQL полностью совместим с SQL (Structured Query Language), что делает его легко доступным и понятным для разработчиков, знакомых с языком запросов к базам данных. Он также поддерживает множество расширений и дополнений для более эффективной работы с данными.
4. Надежность и целостность данных: PostgreSQL обеспечивает высокий уровень надежности и целостности данных благодаря механизмам транзакций, контролю целостности и механизмам восстановления после сбоев. Это делает его идеальным выбором для приложений, где важна сохранность данных.
5. Производительность и масштабируемость: PostgreSQL обладает высокой производительностью и масштабируемостью, позволяя эффективно обрабатывать как небольшие, так и очень большие объемы данных. Он поддерживает параллельную обработку запросов, индексирование данных и оптимизацию запросов для улучшения производительности.
6. Расширяемость и гибкость: PostgreSQL предоставляет множество расширений и дополнений, которые позволяют расширить его функциональность и адаптировать под конкретные потребности проекта. Это включает в себя возможность создания пользовательских типов данных, функций и агрегатных функций.
7. Большое сообщество и поддержка: PostgreSQL имеет активное сообщество разработчиков и пользователей, что обеспечивает обмен знаниями, поддержку и разработку новых возможностей. Существует множество онлайн-ресурсов, форумов и документации, которые помогают разработчикам успешно использовать PostgreSQL в своих проектах.
8. В целом, PostgreSQL представляет собой надежную, производительную и мощную СУБД, которая подходит для широкого спектра приложений, начиная от небольших веб-сайтов до крупных корпоративных систем.

Node.js - это среда выполнения JavaScript, построенная на движке V8 JavaScript от Google Chrome, которая позволяет разработчикам создавать серверные приложения и сетевые приложения с использованием JavaScript. Вот более подробное описание Node.js:

1. Асинхронная и событийно-ориентированная архитектура: Одним из ключевых преимуществ Node.js является его асинхронная и событийно-ориентированная архитектура, которая позволяет обрабатывать большое количество запросов без блокировки потоков. Это позволяет создавать масштабируемые и производительные приложения, способные обрабатывать тысячи одновременных соединений.
2. Единая языковая среда: Node.js позволяет использовать JavaScript как на клиентской стороне (в браузере), так и на серверной стороне, что упрощает разработку и сопровождение приложений. Разработчики могут использовать тот же язык программирования для создания как фронтенда, так и бэкенда, что уменьшает сложность проекта.
3. Модульность и пакетный менеджер npm: Node.js предоставляет мощную систему модулей, которая позволяет организовывать код приложения в небольшие, независимые модули, повторно используемые в разных частях приложения. Пакетный менеджер npm (Node Package Manager) позволяет разработчикам управлять зависимостями и устанавливать сторонние библиотеки и инструменты с помощью простых команд.
4. Веб-сервер и API: Node.js обеспечивает возможность создания высокопроизводительных веб-серверов и API для обработки HTTP запросов. Благодаря своей асинхронной архитектуре, Node.js может обрабатывать большое количество запросов одновременно, что делает его идеальным выбором для создания высоконагруженных веб-приложений.
5. Работа с базами данных: Node.js предоставляет множество модулей для работы с различными базами данных, такими как MongoDB, PostgreSQL, MySQL и другие. Это позволяет разработчикам создавать мощные и масштабируемые приложения с поддержкой различных типов хранилищ данных.
6. Сообщество и поддержка: Node.js имеет активное сообщество разработчиков и пользователей, которые постоянно вносят вклад в его развитие и поддерживают его. Существует множество ресурсов, форумов и документации, которые помогают разработчикам успешно использовать Node.js в своих проектах.
7. Node.js является мощным и гибким инструментом для создания высокопроизводительных серверных приложений и API, и его популярность и простота использования делают его одним из лидеров на рынке серверной разработки.

**4 Сбор и анализ требований к курсовой работе (проекту).**

После изучение потребностей к программному продукту, было назначена работа по созданию компонент программного обеспечения.

Первый этап это было создание и написание структуры БД, и создание связи и самой БД:

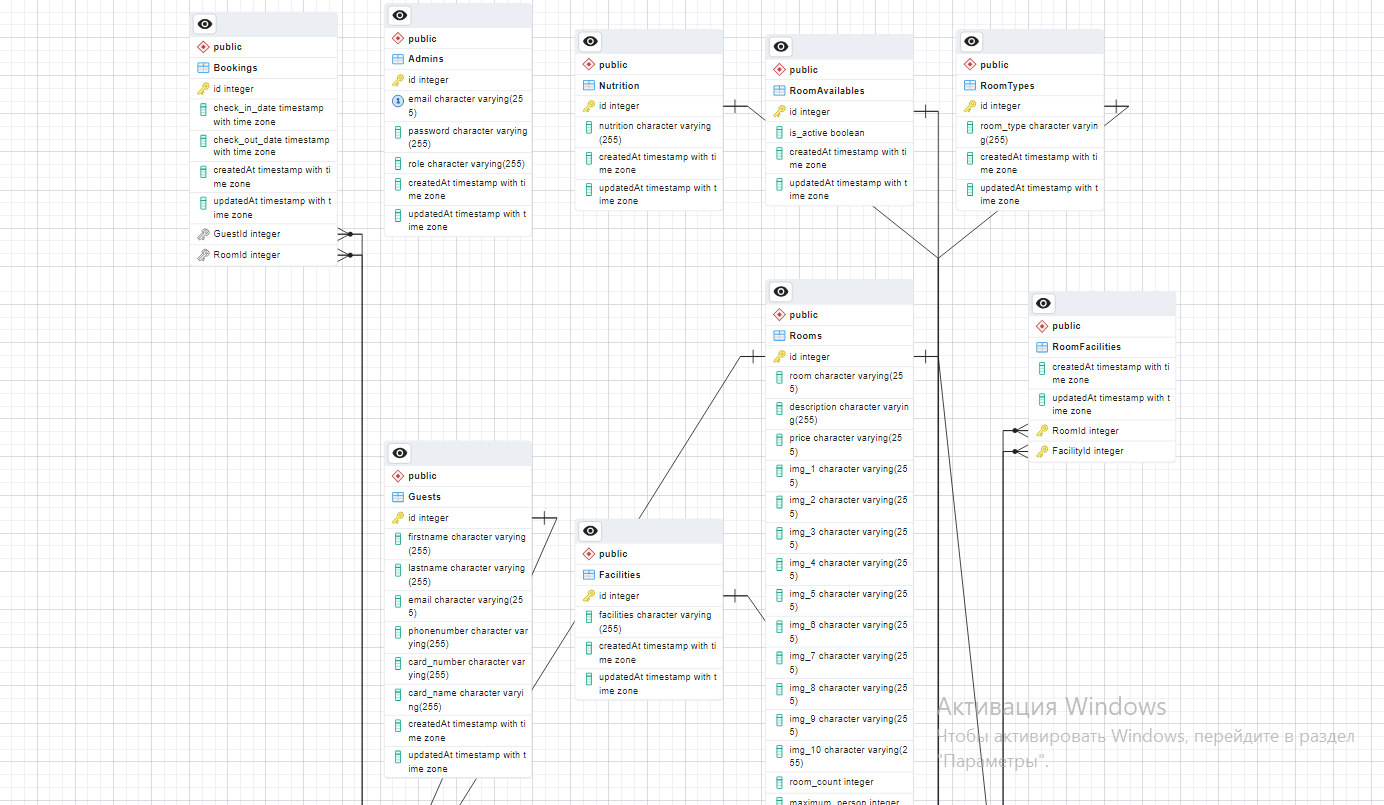


Рисунок 2-ERD диаграмма БД

Следующее создание сервера и API, используя сборку Nodejs:

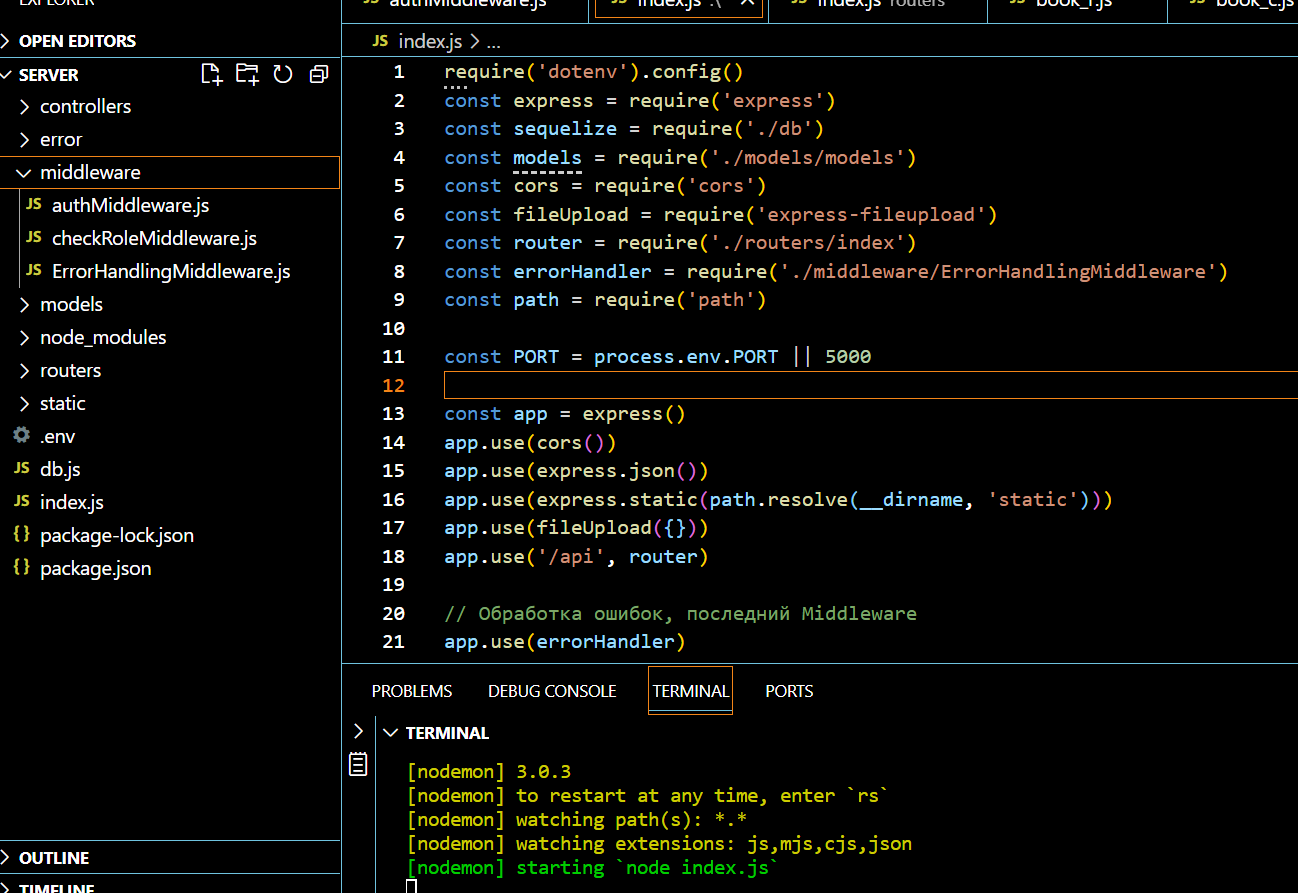


Рисунок 3-Запуск сервера приложения

Установка необходимых зависемостей и библиотек для работы и запуска бэкенд части:

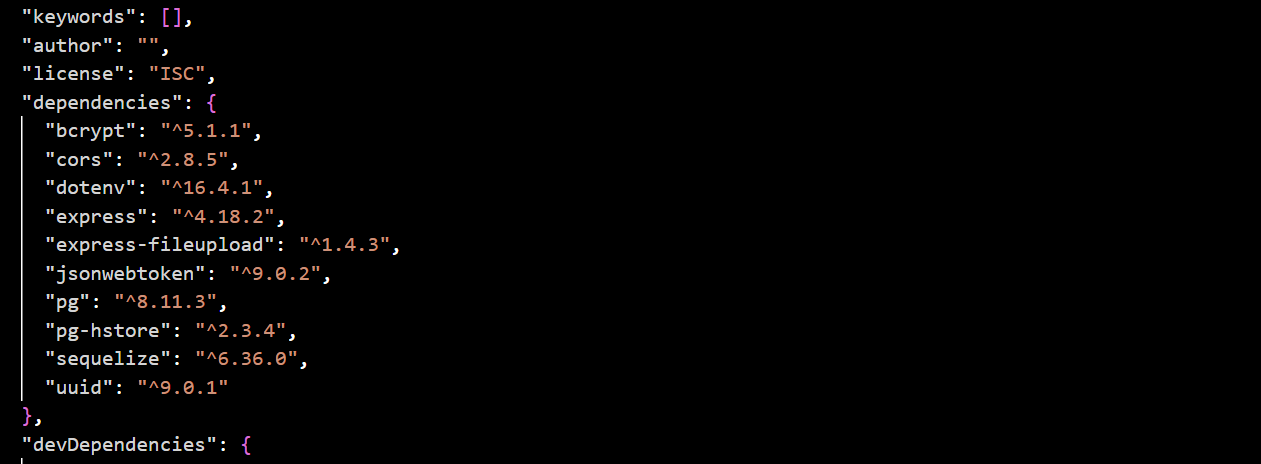


Рисунок 4-Установленные зависимости

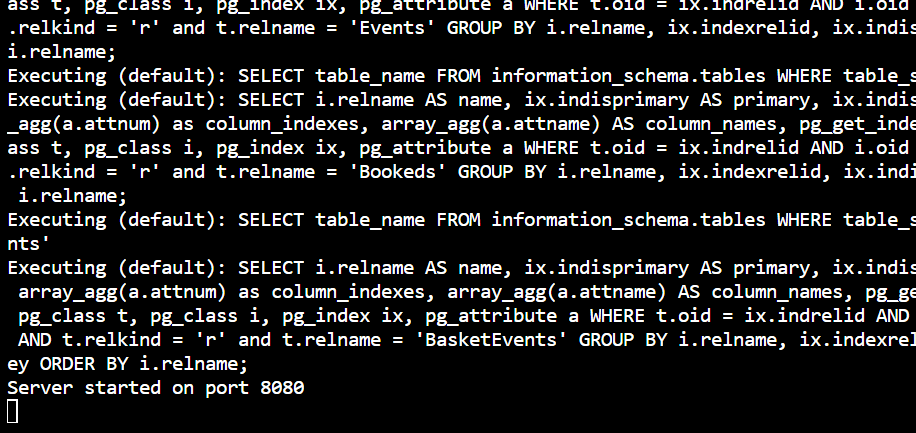


Рисунок 5-Синхронизация моделей с БД и запуск сервера

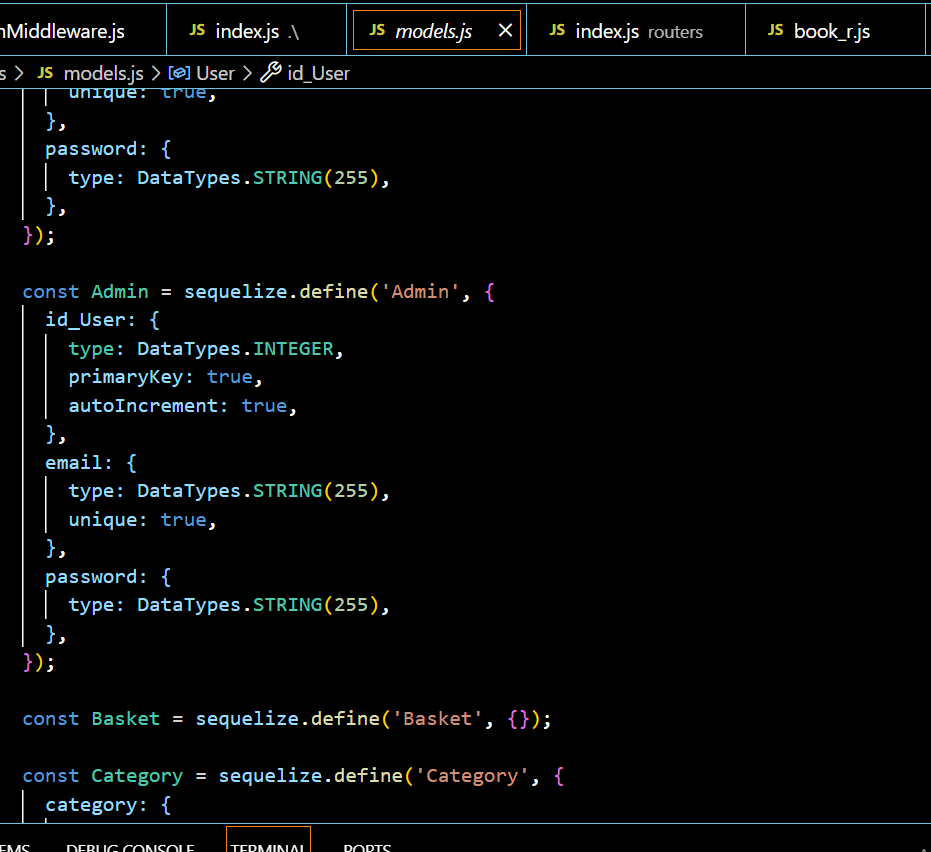
Дальше разработка дизайна и логотипа в графическом редакторе Figma:

Рисунок 6-Создание моделей БД

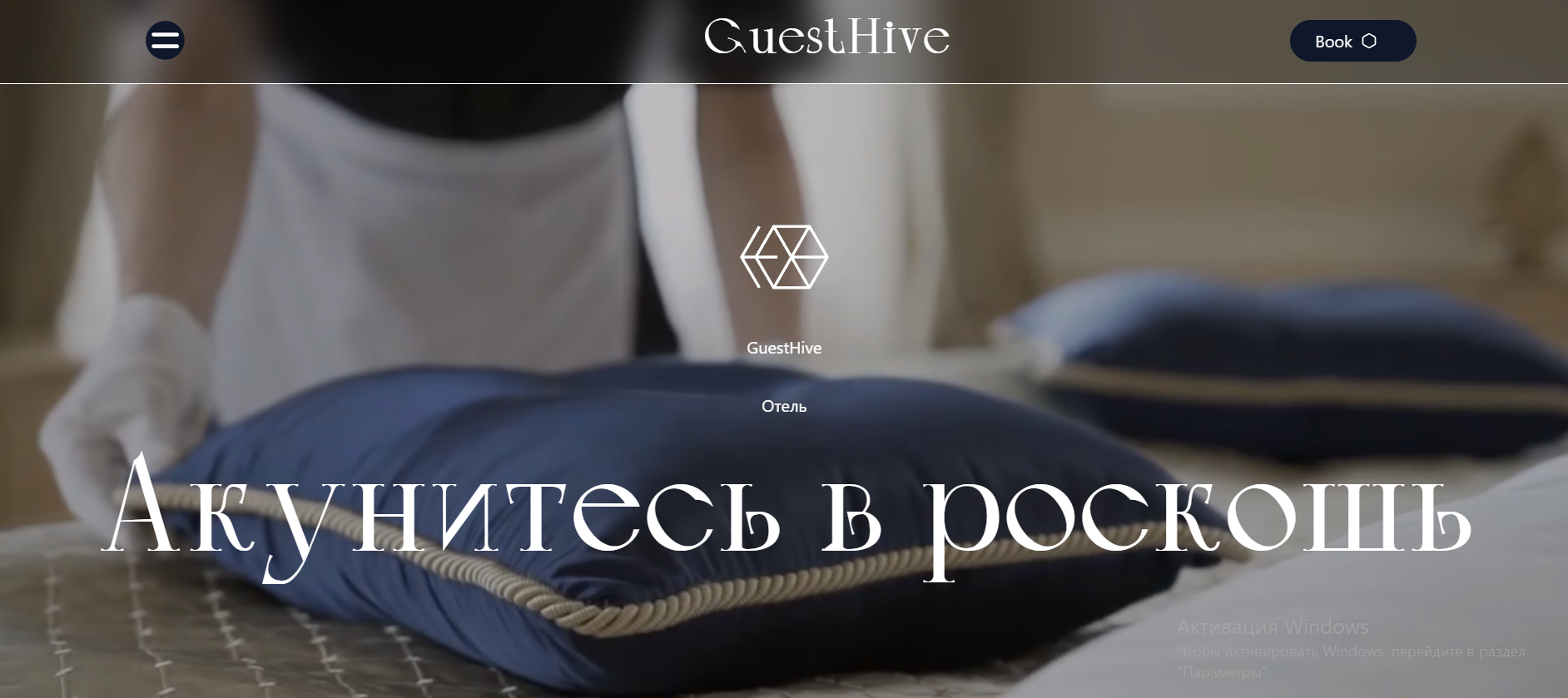


Рисунок 7-Создание клиентской части



Рисунок 8-Создание клиентской части

**5 Разработка программного обеспечения (сборка компонент ПО)**

После подготовки основных компонентов ПО, началась, активная, разработка собственного ПО в виде приложения. В ходе этой разработки студент приобрел дополнительные знания, которые представлены ниже:

Библиотеки Node.js

1. Express - это минималистичный и гибкий веб-фреймворк для Node.js, который предоставляет удобные средства для создания веб-приложений и API.



Рисунок 8-Использование express для создания бэкендной части приложения

1. Sequelize - это обширная ORM (Object-Relational Mapping) библиотека для Node.js, которая позволяет разработчикам взаимодействовать с базами данных с помощью объектов JavaScript, вместо написания SQL запросов вручную.

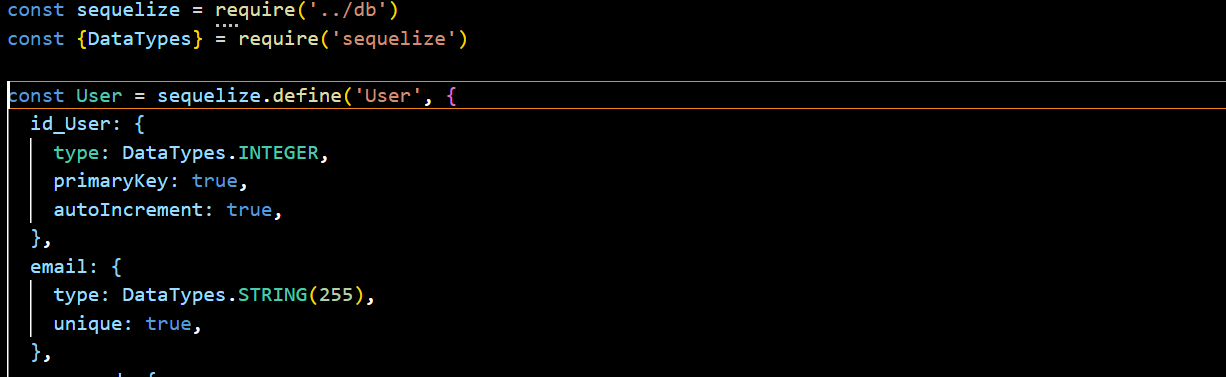


Рисунок 9-Использование sequelize для создания моделей БД

1. dotenv - это библиотека для Node.js, которая загружает переменные окружения из файла .env в процесс приложения.

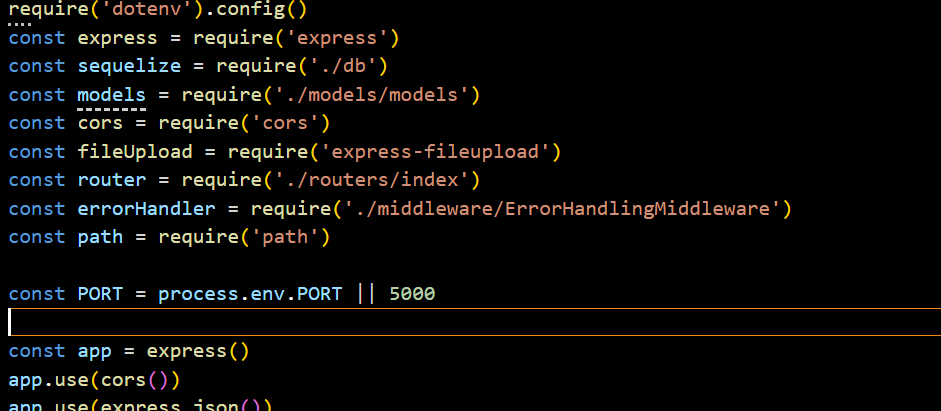


Рисунок 10-Использование dotenv для конфигурации переменных окружения

1. express-fileupload - это middleware для Express, который обеспечивает простое и удобное загрузку файлов на сервер.

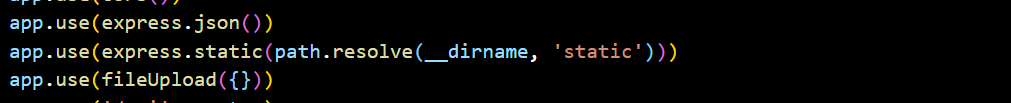


Рисунок 11-Использование express-fileupload в коде

1. bcrypt - это библиотека для хеширования паролей в Node.js. Она основана на алгоритме bcrypt, который является одним из наиболее безопасных методов хеширования паролей.

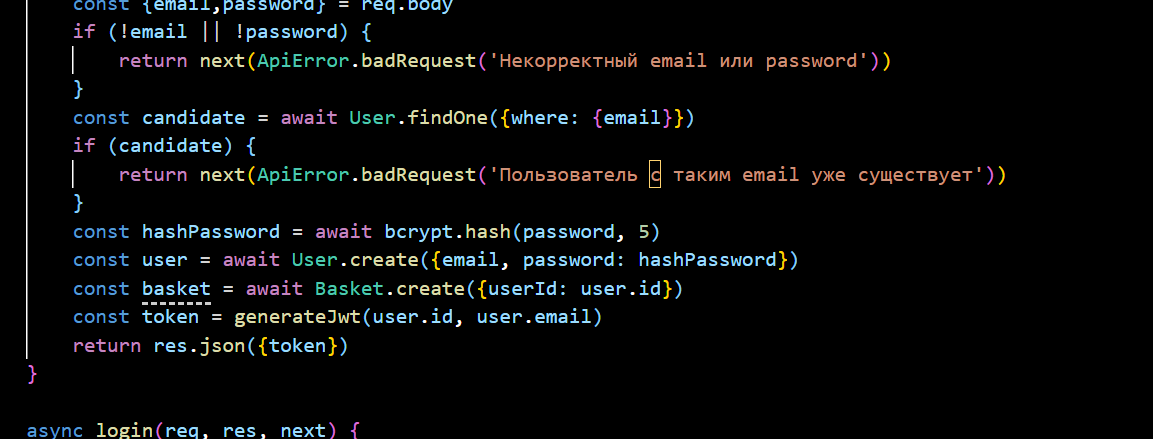


Рисунок 12-Пример использований bcrypt для хэширования

1. Библиотека uuid для JavaScript предоставляет простой способ создания уникальных идентификаторов, известных как UUID (Universally Unique Identifier).

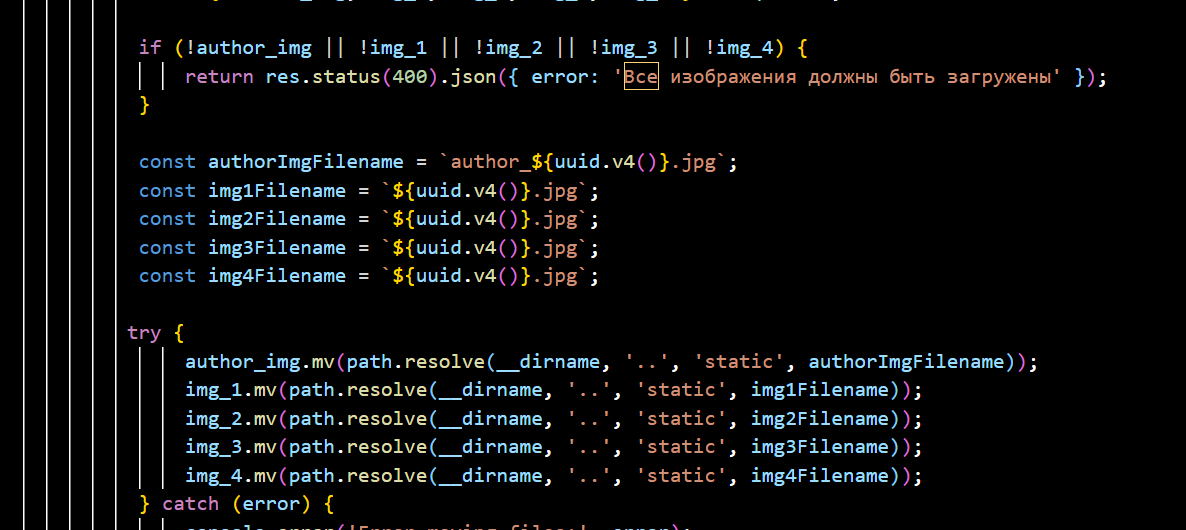


Рисунок 13-Пример использования uuidy для хранения названия картинок картинок

1. Библиотека @react-navigation предоставляет инструменты для создания и управления навигацией между экранами в приложении React Native. Она позволяет легко определять стеки, табы, модальные окна и другие типы навигации.
2. react-native-reanimated - это библиотека для React Native, предоставляющая мощные инструменты для создания анимаций и интерактивных пользовательских интерфейсов. Она основана на библиотеке Reanimated 2, которая предоставляет высокопроизводительные анимации, работающие на нативном уровне.

Работа с HTTP и HTTPS:

Ознакомление с протоколами HTTP и HTTPS было важным шагом в понимании того, как работают веб–сайты. HTTP (Hypertext Transfer Protocol) и HTTPS (Hypertext Transfer Protocol Secure) являются протоколами передачи данных между клиентом и сервером. HTTPS обеспечивает шифрование данных для защиты конфиденциальности.

Типы Запросов и API:

Научившись использовать различные виды запросов (GET, PUT, DELETE, PATCH, POST, UPDATE), можно взаимодействовать с REST API. Это открыло возможность обмена данными между фронтендом и бэкендом.

Переменные Окружения:

Environment variables (переменные окружения) являются важным средством безопасности и конфигурации. Они позволяют хранить чувствительную информацию, такую как ключи API, вне исходного кода, что обеспечивает безопасность приложения.

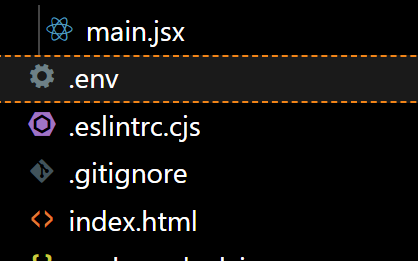


Рисунок 16-Создание переменных окружения в файле .env

PERN Stack:

PERN представляет собой стек технологий, включающий PosgreSQL, Express.js, React-Native и Node.js. Он предоставляет полный набор инструментов для создания современных веб–приложений.

**Заключение**

Заключение для курсовой работы по теме "Платформа просмотра культурных событий и покупки билетов":

В результате проведенного исследования и разработки платформы для просмотра культурных событий и покупки билетов были достигнуты следующие результаты и выводы.

Первоначально были определены основные требования к функциональности и дизайну платформы на основе анализа рынка и потребностей пользователей. Затем была проведена разработка архитектуры приложения, включая выбор технологий и инструментов для реализации.

В ходе разработки были использованы современные инструменты и технологии, такие как React Native для создания мобильного приложения, PostgreSQL для хранения данных о мероприятиях и билетах, а также Node.js для создания серверной части приложения. Также были применены библиотеки и фреймворки, такие как Express для создания API и Reanimated для реализации высокопроизводительных анимаций.

Полученное приложение предоставляет пользователям удобный и интуитивно понятный интерфейс для просмотра культурных событий, ознакомления с информацией о них и приобретения билетов. Оно обладает высокой производительностью и отзывчивостью благодаря использованию современных технологий и библиотек.

Таким образом, разработанная платформа представляет собой полезный инструмент для организации культурных мероприятий и удовлетворения потребностей пользователей в доступе к ним. Дальнейшее развитие проекта может включать в себя расширение функциональности, оптимизацию производительности и улучшение пользовательского опыта на основе обратной связи от пользователей.

**Литература**

1. Brown, Sarah. "Understanding User Experience: How to Create Great Products." Publisher Y, 2018. - Данная книга рассматривает принципы проектирования интерфейса и создания положительного пользовательского опыта при использовании приложений, что может быть полезно при разработке пользовательского интерфейса для платформы просмотра культурных событий.
2. Johnson, Mark. "Database Management: A Comprehensive Guide." Publisher Z, 2019. - Эта книга предоставляет информацию о методах организации и управления базами данных, что может быть полезно при выборе и настройке базы данных для хранения информации о культурных событиях и билетах.
3. Online Documentation and Tutorials:
4. [React.js Documentation](https://reactjs.org/docs/getting-started.html)
5. [Tailwind CSS Documentation](https://tailwindcss.com/docs)
6. [Vite Documentation](https://vitejs.dev/guide/)
7. Официальная документация PostgreSQL (https://www.postgresql.org/docs/)
8. Официальная документация Node.js (https://nodejs.org/en/docs/)