

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий
Кафедра Информационных систем и технологий
Специальность 1-40 01 01 «Программное обеспечение информационных технологий»
Специализация 1-40 01 01 10 «Программное обеспечение информационных технологий
(программирование интернет-приложений)»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

Веб-приложение для управления библиотекой книг

Выполнил студент Мамаева Диана Александровна
(Ф.И.О.)

Руководитель проекта ст. препод. Дубовик М.В.
(учен. степень, звание, должность, подпись, Ф.И.О.)

Заведующий кафедрой к.т.н., доц. Смелов В.В.
(учен. степень, звание, должность, подпись, Ф.И.О.)

Консультанты ст. препод. Дубовик М.В.
(учен. степень, звание, должность, подпись, Ф.И.О.)

Нормоконтролер ст. препод. Дубовик М.В.
(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой _____

Содержание

ВВЕДЕНИЕ	3
1 Постановка задачи	4
1.1 Обзор аналогов.....	4
1.2 Постановку задач	6
2 Проектирование web-приложения	7
2.1 Архитектура веб-приложения	7
2.2 Описание работы программы	8
2.3 Проектирование базы данных	9
2.4 Итоги проектирования web-приложения	13
3 Разработка web-приложения.....	14
3.1 Разработка серверной части	14
3.2 Разработка клиентской части	18
4 Тестирование web-приложения.....	21
5 Руководство пользователя	23
5.1 Регистрация пользователя.....	23
5.2 Аутентификация и авторизация пользователя	24
5.3 Главная страница	24
5.4 Страница книги	25
5.5 Личная страница пользователя.....	26
5.6 Панель администратора	26
5.7 Удаление комментарием	28
Заключение	29
Список используемых источников	30
Приложение А	31

ВВЕДЕНИЕ

Современное информационное общество требует доступа к высококачественной литературе, и управление большими библиотеками книг может представлять сложность без эффективных инструментов и ресурсов. В свете этой проблемы становится актуальным и востребованным разработка функционального web-приложения для управления библиотекой книг.

Целью данного курсового проекта является создание такого web-приложения, которое позволит пользователям эффективно управлять библиотекой книг. Главная цель заключается в разработке полнофункционального приложения, обеспечивающего возможность регистрации и авторизации пользователей, поддержку ролей администратора и пользователя, добавление новых книг и информации о них, чтение и скачивание книг, редактирование информации, поиск книг по различным критериям, просмотр информации о книгах, оставление отзывов и ставление оценок, а также создание личных списков прочитанных книг, списка "хочу прочитать" и списка "любимое". Для достижения этих целей были выполнены следующие задачи:

- постановка задачи;
- проектирование web-приложения;
- разработка web-приложения;
- тестирование web-приложения;
- составление руководства пользователя.

Для разработки веб-приложения были выбраны современные технологии и инструменты, которые позволили скорость разработки и удобство использования приложения.

В данном проекте на стороне сервера были использованы два популярных инструмента: Express и Socket.IO. Для хранения данных была использована база данных MySQL, которая является одной из наиболее распространенных реляционных баз данных. Для работы с базой данных MySQL в проекте был использован ORM-фреймворк Prisma. Таким образом, Express обеспечивает обработку HTTP-запросов и маршрутизацию на стороне сервера, Socket.IO обеспечивает двунаправленную связь в режиме реального времени между клиентом и сервером, MySQL служит в качестве хранилища данных, а Prisma упрощает взаимодействие с базой данных и выполнение запросов. Эти инструменты совместно позволяют разрабатывать функциональное и эффективное web-приложение для управления библиотекой книг.

На стороне клиента для разработки пользовательского интерфейса был использован React Bootstrap. Bootstrap — это открытый и бесплатный HTML, CSS и JS фреймворк, который используется веб-разработчиками для быстрой вёрстки адаптивных дизайнов сайтов и веб-приложений[1].

1 Постановка задачи

Изучая, аналогичные веб-приложения, были найдены схожие по теме и функционалу приложения. Большая часть веб-приложений похожи друг на друга, и функционалом почти не отличаются. Так же были найдены статьи, которые помогли реализовать некоторые функции. Результат анализа представлен ниже.

1.1 Обзор аналогов

Goodreads.com - это популярная онлайн-платформа для любителей чтения книг. Она предоставляет возможность пользователям искать, открывать и оценивать книги. Goodreads.com также позволяет пользователям составлять личные списки прочитанных книг, добавлять отзывы и рекомендации, а также общаться с другими читателями через сообщества и группы обсуждения. Сайт предлагает разнообразные функции для удобного взаимодействия с книжным контентом и обмена литературными впечатлениями.

Интерфейс этого сайта можно увидеть на рисунке 1.1.



Рисунок 1.1 – Домашняя страница Goodreads

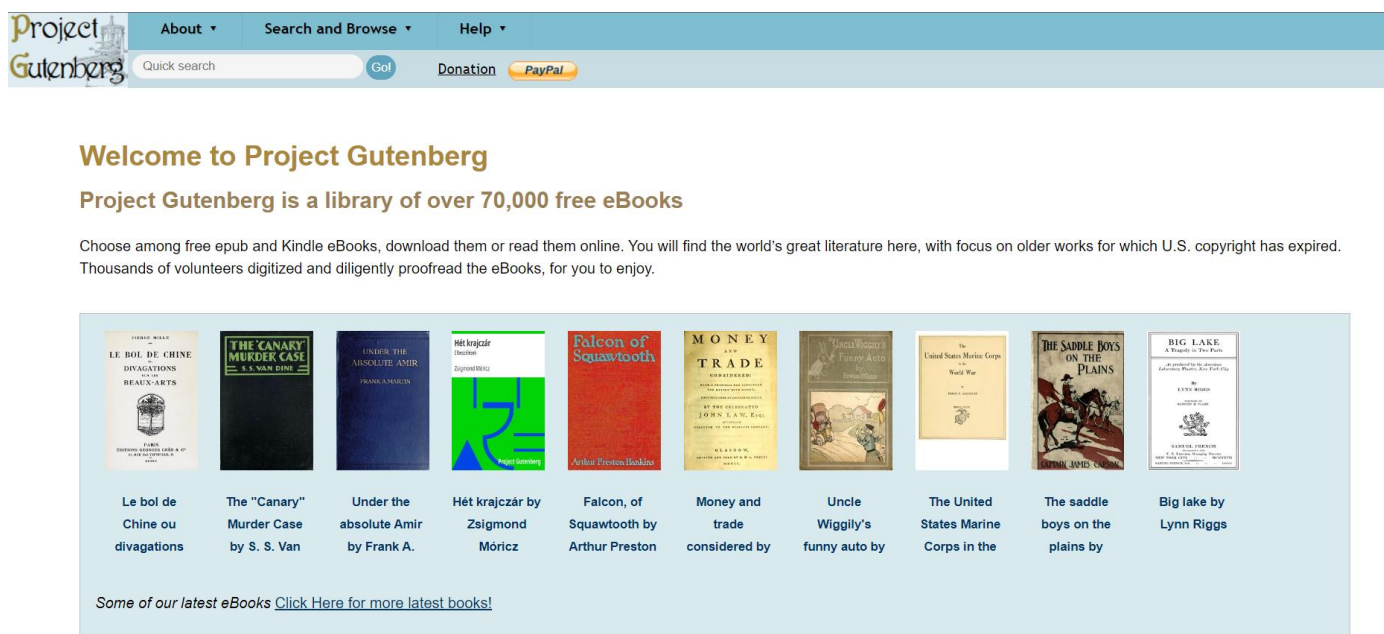
Преимущества данного сайта представлены ниже:

- Большой каталог книг;
- Сообщество чтения: Goodreads предоставляет возможность вступить в сообщество чтения, где пользователи могут обмениваться рекомендациями, отзывами и обсуждать книги с другими читателями;

— Личные списки и отслеживание прочитанного. Пользователи могут создавать личные списки прочитанных книг, отмечать книги, которые они хотят прочитать, и отслеживать свой прогресс чтения. Это помогает пользователям организовать свою литературную активность и следить за своими литературными интересами

Project Gutenberg - это онлайн-библиотека, которая предоставляет бесплатный доступ к большой коллекции электронных книг. Сайт характеризуется огромным выбором литературных произведений разных жанров, включая классику, научно-популярные и художественные книги. Пользователи могут бесплатно скачивать книги в различных форматах, таких как EPUB, MOBI, PDF и др. Project Gutenberg сосредоточен на предоставлении доступа к общественному достоянию литературы и является важным ресурсом для чтения книг в электронном формате.

Интерфейс этого сервиса можно увидеть на рисунке 1.2.



No fee or registration! Everything from Project Gutenberg is gratis, libre, and completely without cost to readers. If you find Project Gutenberg useful, please consider a small donation to help Project Gutenberg digitize more books, maintain its online presence, and improve Project Gutenberg programs and offerings. Other ways to help include digitizing, proofreading and formatting, or reporting errors.

Рисунок 1.2 – Домашняя страница Project Gutenberg

В целом, Project Gutenberg предоставляет бесплатный и удобный доступ к большой коллекции книг в общественном достоянии, что делает его ценным ресурсом для чтения и распространения литературного наследия. Рассмотрим некоторые преимущества.

Project Gutenberg имеет несколько преимуществ:

- Бесплатный доступ;
- Общественное достояние;

- Разнообразие книг. Коллекция Project Gutenberg сосредоточена на произведениях, находящихся в общественном достоянии. Это включает классические литературные работы, которые больше не защищены авторским правом. Пользователи могут получить доступ к произведениям, которые иначе могли бы быть недоступны или требовали бы платы.

- Различные форматы. Книги в Project Gutenberg доступны в различных форматах, включая EPUB, MOBI, PDF и другие. Это позволяет пользователям выбирать формат, который наиболее удобен для чтения на их устройствах, будь то электронные книги, смартфоны или планшеты.

1.2 Постановку задач

На основе проведенного анализа и учета требований, указанных в задании на курсовое проектирование, для разработки функционального web-приложения по управлению библиотекой книг могут быть поставлены следующие задачи:

- Разработка системы регистрации и авторизации: Задача состоит в создании функционала, позволяющего пользователям регистрироваться на сайте, авторизовываться и сохранять их учетные данные в безопасном виде.

- Создание системы ролей: Задача заключается в реализации механизма, позволяющего выделять роли администратора и пользователя. Администратору предоставляются дополнительные привилегии для управления книгами и данными библиотеки.

- Разработка функционала добавления и редактирования книг: Задача состоит в создании возможности добавления новых книг в библиотеку с указанием соответствующей информации (название, автор, жанр и др.). Также необходимо реализовать функционал редактирования информации о существующих книгах.

- Реализация функционала чтения и скачивания книг: Задача заключается в создании механизма, позволяющего пользователям читать книги онлайн и скачивать их в удобном формате.

- Разработка системы поиска книг: Задача состоит в создании возможности поиска книг в библиотеке по различным критериям, таким как название, автор, жанр и т.д. Это позволит пользователям быстро находить интересующие их книги.

- Реализация функционала отзывов и оценок: Задача заключается в создании механизма, позволяющего пользователям оставлять отзывы на книги и ставить им оценки. Также необходимо предусмотреть возможность просмотра отзывов и оценок других пользователей.

- Разработка функционала личных списков: Задача состоит в создании функционала, позволяющего пользователям создавать личные списки прочитанных книг, списки "хочу прочитать" и "любимое". Это позволит пользователям организовать и управлять своими литературными предпочтениями.

2 Проектирование web-приложения

Проект, разработанный в рамках данной работы, представляет собой клиент-серверную архитектуру, включающую два основополагающие части: клиентскую и серверную. Клиентская часть отвечает за взаимодействие с пользователем, предоставляет интерфейс для ввода данных и отображения результатов, в то время как серверный компонент обеспечивает обработку запросов, взаимодействие с базой данных и логику бизнес-процессов. Такое разделение функциональности позволяет эффективно управлять ресурсами и обеспечить надежную работу приложения.

2.1 Архитектура веб-приложения

Общая структура проекта клиентская часть и серверная.

Проект состоит из клиентской и серверной частей. Клиент представляет собой веб-интерфейс, построенный с использованием библиотеки React Bootstrap. Он позволяет клиенту отправлять запросы на сервер и отображать полученные ответы. На рисунке 2.1 предложена схема-развертывания приложения.

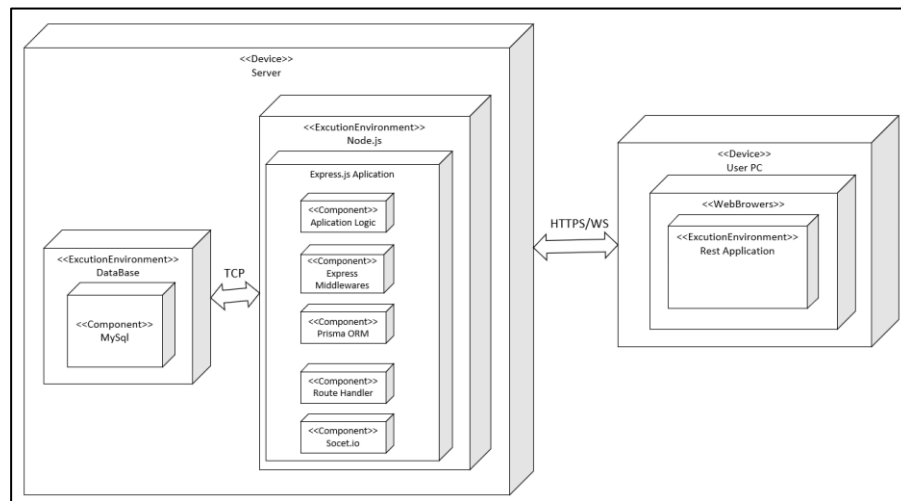


Рисунок 2.1 – Структурная схема приложения

Серверная часть реализована с использованием фреймворка Express. Он обрабатывает запросы, поступающие от клиента, и использует Prisma для взаимодействия с базой данных.

Взаимодействие между клиентом и сервером осуществляется по протоколу HTTPS. Клиент отправляет запросы на определенные маршруты сервера, а сервер обрабатывает эти запросы и возвращает соответствующие ответы.

React Bootstrap обеспечивает создание пользовательского интерфейса с использованием готовых компонентов, стилей и сеток, что упрощает разработку и создание современного дизайна веб-приложения.

Серверная часть использует различные промежуточные компоненты, такие как CORS для обработки запросов с разных источников, ErrorHandlerMiddleware для обработки ошибок. Кроме того, сервер использует пакет express-fileupload для обработки загрузки файлов, что позволяет клиентской части скачивать, читать книги, а другие файлы, по типу обложки книг и аватар пользователя.

Связь между клиентом и сервером в режиме реального времени реализована с использованием библиотеки Socket.IO. Она позволяет установить двустороннее соединение между клиентом и сервером, обеспечивая обновление данных на клиентской стороне без необходимости повторных запросов.

В результате, данная структура программного продукта обеспечивает эффективное и отзывчивое веб-приложение для управления отзывами и обработки данных из базы данных.

2.2 Описание работы программы

При запуске программы, сервер настраивается для прослушивания входящих запросов на определенном порту. Клиентское приложение загружается в веб-браузере пользователя, и пользователь может взаимодействовать с интерфейсом, предоставленным React Bootstrap.

Основной функционал приложения можно наглядно увидеть на use case-диаграмме, такая предоставлена на рисунке 2.2



Рисунок 2.2 – Use case-диаграмма

Когда пользователь выполняет определенные действия, например, отправляет запрос на добавление закладки книги, клиентское приложение формирует соответствующий HTTPS-запрос и отправляет его на сервер.

Сервер, используя фреймворк Express. Express представляет собой популярный веб-фреймворк, написанный на JavaScript и работающий внутри среды исполнения node.js. Этот модуль освещает некоторые ключевые преимущества этого фреймворка, установку среды разработки и выполнение основных задач веб-разработки и развёртывания[2]. принимает запрос и маршрутизирует его к соответствующему обработчику маршрута. Обработчик маршрута выполняет необходимые операции,

например, создает новую задачу в базе данных с использованием Prisma или работы с файлами. При успешном выполнении операции, сервер формирует ответ и отправляет его обратно клиентскому приложению.

Клиентское приложение получает ответ от сервера и обновляет свой интерфейс, отображая актуальные данные или информацию об успешном выполнении операции. В случае возникновения ошибок, сервер использует middleware для обработки и отправки соответствующего сообщения об ошибке обратно клиентскому приложению.

Кроме того, взаимодействие между клиентом и сервером в режиме реального времени осуществляется с использованием Socket.IO. Например, при появлении новой задачи, сервер отправляет уведомление о ней клиентскому приложению через WebSocket-соединение, и клиентское приложение мгновенно отображает это уведомление пользователю.

На рисунке 2.3 представлена UML-диаграмма, иллюстрирующая основные компоненты приложения и их взаимодействие:

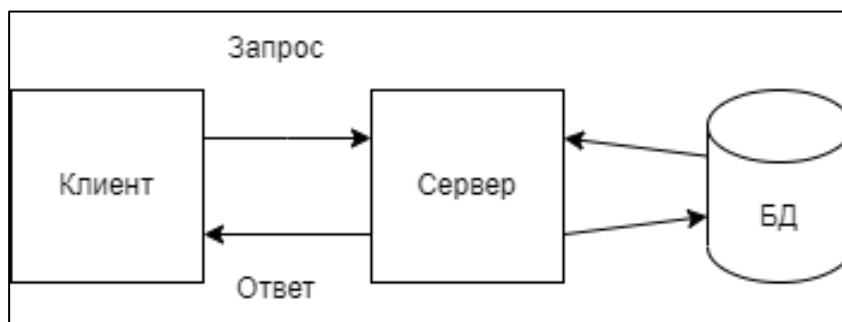


Рисунок 2.3 – Структурная схема приложения

Данная диаграмма показывает, что клиентское приложение взаимодействует с сервером через протокол HTTPS, а также осуществляет обмен данными с БД. Такая структура программного продукта позволяет пользователям взаимодействовать с сервером, отправлять запросы и получать ответы, а также обновлять данные в режиме реального времени с помощью сокетов. Это создает эффективное и отзывчивое веб-приложение для управления задачами и обработки данных из базы данных.

2.3 Проектирование базы данных

После проведения аналогов данного проекта, была спроектирована база данных, для реализации всех поставленных задач. Данная СУБД выбрана в качестве преимущественной из-за своей простоты создания таблиц и широкой поддержки веб-приложений. MySQL обладает простым и понятным языком запросов (SQL) и хорошо интегрируется с фреймворком Prisma, который используется в серверной части проекта.

С использованием MySQL, серверная часть может эффективно управлять данными, хранить информацию о пользователях, задачах и других сущностях,

необходимых для работы приложения. Создание таблиц и определение связей между ними происходит в соответствии с моделью данных, разработанной для проекта.

Схема созданного БД для данного проекта представлена на рисунке 2.4.

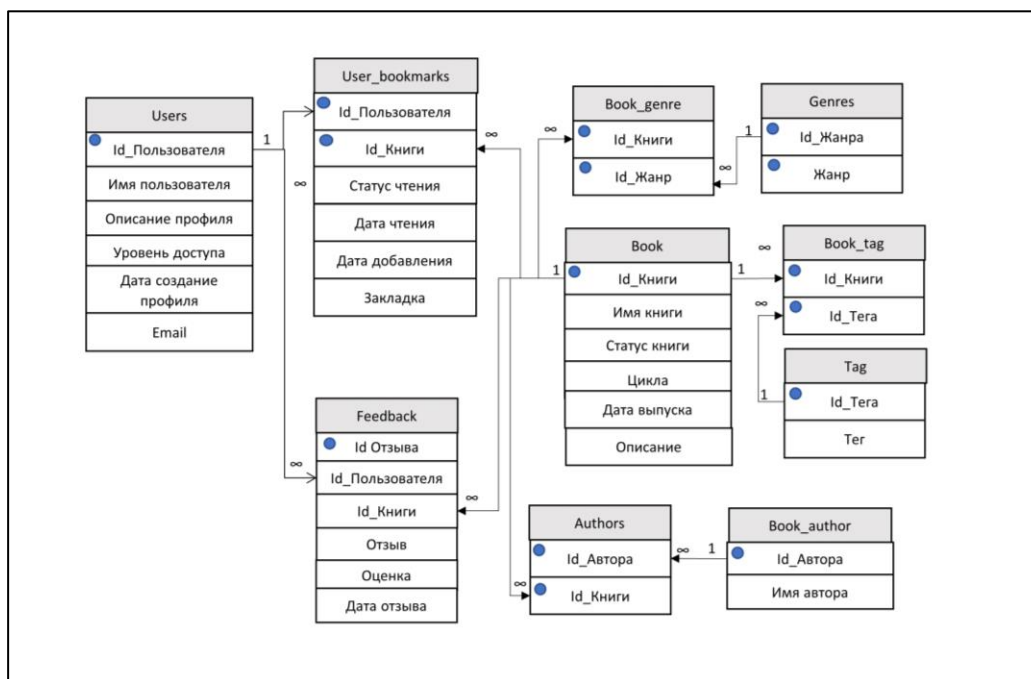


Рисунок 2.4 – Модель базы данных

Основой инфраструктуры базы данных является грамотно спроектированная модель, которая отображает связь пользовательских таблиц. Правильное и корректное взаимодействие их друг с другом как раз и заключается в схеме базы данных со связями, верно отображающими их положение.

Далее будет описана каждая таблица базы данных.

В таблице «users» хранятся все зарегистрированные пользователи. Описание структуры таблицы «users» представлено в таблице 2.1.

Таблица 2.1 – Структура таблицы «users»

Название столбца	Тип, ограничение целостности	Описание столбца
id	int, primary key	Идентификатор пользователя
EMAIL	varchar(50),unique	Электронная почта пользователя
pass	varchar(500)	Хеш пароля пользователя
ACCESS_LEVEL	varchar(20)	Роль пользователя
USER_NAME	varchar(20)	Имя пользователя
DATA_CREATE	DateTime	Время создания профиля
USER_INFO	Text	Описание профиля

В таблице «books» хранятся книги. Описание структуры таблицы «books» представлено в таблице 2.2.

Таблица 2.2 – Структура таблицы «books»

Название столбца	Тип, ограничение целостности	Описание столбца
ID	int, primary key	Идентификатор книги
BOOK_NAME	Text	Название книги
BOOK_SERIES	Text	Цикл книги
BOOK_DESCRIPTION	Text	Описание книги
CHAPTERS	int	Количество глав
DATA_RELEASE	DateTime	Дата выпуска
DATA_ADD	DateTime	Дата добавления в бд

Таблица «authors» содержит названия авторов книг. Её структура представлена в таблице 2.3.

Таблица 2.3 – Структура таблицы «authors»

Название столбца	Тип, ограничение целостности	Описание столбца
ID	Int, primary key	Идентификатор автора
AUTHOR_NAME	varchar(25), unique	Имя автора

В таблице «genres» содержит названия жанров. Описание структуры таблицы «genres» представлено в таблице 2.4.

Таблица 2.4 – Структура таблицы «genres»

Название столбца	Тип, ограничение целостности	Описание столбца
ID	Int, primary key	Идентификатор жанра
GENRE_NAME	varchar(25), unique	Название жанра

В таблице «tag» содержит название тегов используемый в книжках. Описание структуры таблицы «tag» представлено в таблице 2.5.

Таблица 2.5 – Структура таблицы «tag»

Название столбца	Тип, ограничение целостности	Описание столбца
ID	Int, primary key	Идентификатор тега
TAG_NAME	varchar(25), unique	Название тега

В таблице «book_author» описывается какие авторы присвоены книге. Описание структуры таблицы представлено в таблице 2.6.

Таблица 2.6 – Структура таблицы «book_author»

Название столбца	Тип, ограничение целостности	Описание столбца
ID	int, primary key	Идентификатор записи таблицы
BOOKID	int, foreign key	Идентификатор книги
AUTHORID	int, foreign key	Идентификатор автора

В таблице «book_genre» описывается какие жанры присвоены книге. Описание структуры таблицы представлено в таблице 2.7.

Таблица 2.7 – Структура таблицы «book_genre»

Название столбца	Тип, ограничение целостности	Описание столбца
ID	int, primary key	Идентификатор записи таблицы
BOOKID	int, foreign key	Идентификатор книги
GENREID	int, foreign key	Идентификатор жанра

В таблице «book_tag» описывается какие теги присвоены книге. Описание структуры таблицы представлено в таблице 2.8.

Таблица 2.8 – Структура таблицы «book_tag»

Название столбца	Тип, ограничение целостности	Описание столбца
ID	int, primary key	Идентификатор записи таблицы
BOOKID	int, foreign key	Идентификатор книги
TAGID	int, foreign key	Идентификатор тега

Таблица «feedback» содержит информацию об отзывах, оставленных пользователями, а также и оценку, которая влияет на рейтинг книги. Описание структуры таблицы представлено в таблице 2.9.

Таблица 2.9 – Структура таблицы «feedback»

Название столбца	Тип, ограничение целостности	Описание столбца
ID	Int, primary key	Идентификатор отзыва
ID_USER	Int, foreign key	Идентификатор пользователя
ID_BOOK	Int, foreign key	Идентификатор книги
FEEDBACK	text	Текст отзыва
RATING	float	Оценка
DATE	DateTime	Дата создания

В таблице «user_bookmarks» содержит информацию о закладках пользователей. Описание структуры таблицы представлено в таблице 2.10.

Таблица 2.10 – Структура таблицы «user_bookmarks»

Название столбца	Тип, ограничение целостности	Описание столбца
ID	Int, primary key	Идентификатор отзыва
ID_USER	Int, foreign key	Идентификатор пользователя
ID_BOOK	Int, foreign key	Идентификатор книги
STATUS_READING	varchar(25)	Статус чтения книги
DATA_READING	DateTime	Дата последнего чтения
DATA_ADD	DateTime	Дата добавления
MARKS	Varchar(1000)	Закладка, в книге

Таким образом при проектировании базы данных используется 10 таблиц, присутствует связь один-ко-многим и многие-ко-многим.

2.4 Итоги проектирования web-приложения

В результате данного проекта были разработаны основные компоненты, которые выполняют следующие функции:

Клиентское приложение: Клиентское приложение представляет собой веб-интерфейс на React Bootstrap, который обеспечивает пользовательскую интеракцию. Оно позволяет пользователям отправлять запросы на сервер, визуализировать ответы и взаимодействовать с функциональностью приложения. Клиентское приложение отображает данные, полученные от сервера, и обновляет интерфейс в соответствии с действиями пользователя.

Серверная часть: Серверная часть основана на фреймворке Express и обрабатывает запросы, поступающие от клиентского приложения. Она обеспечивает маршрутизацию запросов и взаимодействие с базой данных. Серверная часть выполняет следующие функции:

- Принимает запросы от клиентского приложения через HTTPS протокол;
- Маршрутизирует запросы к соответствующим обработчикам маршрутов;
- Обрабатывает запросы, взаимодействуя с базой данных для создания, чтения, обновления и удаления данных;
- Формирует ответы и отправляет их обратно клиентскому приложению.
- Обеспечивает Web-Socket соединение.

База данных: База данных хранит информацию о пользователях, задачах и других сущностях, необходимых для работы приложения. Она обеспечивает сохранение и извлечение данных, которые используются серверной частью приложения. База данных выполняет следующие функции:

- Хранит данные о пользователях, задачах и других сущностях;
- Обеспечивает доступ к данным для чтения, записи, обновления и удаления;
- Поддерживает целостность данных и обеспечивает безопасность доступа к данным.

Концептуально, основными компонентами проекта являются клиентское приложение, серверная часть и база данных. Клиентское приложение предоставляет пользовательский интерфейс и взаимодействует с сервером, отправляя запросы и получая ответы. Серверная часть обрабатывает запросы от клиента, взаимодействует с базой данных и возвращает результаты обратно клиентскому приложению. База данных хранит и обеспечивает доступ к данным, необходимым для функционирования приложения.

Таким образом, проект представляет собой систему, которая позволяет пользователям управлять задачами через веб-интерфейс. Клиентское приложение обеспечивает удобный интерфейс, а серверная часть и база данных обеспечивают хранение данных и выполнение операций над ними.

3 Разработка web-приложения

В процессе выполнения курсового проекта было разработано веб-приложение с использованием архитектурного стиля REST API.

REST расшифровывается как REpresentational State Transfer. Это был термин, первоначально введен Роем Филдингом (Roy Fielding), который также был одним из создателей протокола HTTP. Отличительной особенностью сервисов REST является то, что они позволяют наилучшим образом использовать протокол HTTP[3].

Проект включает два отдельных компонента: server и client, ответственных соответственно за серверную и клиентскую части программного средства.

3.1 Разработка серверной части

Серверная часть делится на 3 основных блока: Controller, Model, Route и другие сопутствующие им блоки по типу static, public, prisma. А так же блоки обработки ошибок error и middleware. Более наглядно структура показана на рисунке 3.1



Рисунок 3.1 – Структура проекта сервера

Входной точкой в приложение является файл `app.js`, в котором осуществляется запуск сервера для прослушивания входящих запросов, поступающих на определенный порт. В этом файле происходит подключение необходимых модулей для работы сервера, а также выполнение начальной инициализации приложения и регистрация маршрутизаторов.

`app.js` является основным файлом, который иницирует работу веб-сервера. В нем происходит создание экземпляра сервера с использованием фреймворка `Express.js`. Затем осуществляется настройка сервера, включая установку порта для прослушивания входящих запросов.

Далее происходит подключение необходимых модулей, которые используются в процессе работы сервера. Это может включать модули для работы с базой данных, обработки маршрутов, аутентификации и авторизации пользователей, обработки ошибок и другие.

После подключения модулей происходит инициализация приложения, включая настройку и конфигурацию необходимых компонентов и настройку маршрутов.

В результате выполнения кода в `app.js` сервер становится готовым к принятию входящих запросов и обработке их в соответствии с заданными маршрутами и логикой приложения. Это обеспечивает работоспособность и функциональность серверной части приложения, которая готова взаимодействовать с клиентской частью и обрабатывать запросы пользователей. В директории «`modules`» содержится модули для каждой предметной области приложений. В каждой такой папке есть файл роутера, сервиса и репозитория.

Для удобной реализации серверной части приложения был выбран фреймворк `Express.js`. Этот фреймворк позволяет разделить сервер на модели, контроллеры и роуты, обеспечивая более структурированный подход к разработке.

Модели представляют собой компоненты, отвечающие за взаимодействие с базой данных и представление данных в виде объектов. Они содержат определения схемы данных и методы для выполнения операций чтения, записи и обновления данных в базе. Рассмотрим один из методов модели `book`, `getBooks` – представляет собой метод принимающий номер страницы, параметры фильтрации и поиска. Данный метод связывается с `prisma.client` обращаясь к таблице `books`. В результате запроса к бд возвращает указанно страницу из списка отфильтрованный данных. Такой запрос можно увидеть в листинге 3.1

```
const books = await prisma.books.findMany({
  where,
  skip: skip(page),
  take: take,
});
```

Листинг 3.1 – Запрос метода `getBooks`

Запрос имеет 3 параметра `where`, `skip`, `take`. `Where` реализует поиск и фильтрацию данные, по заданным параметрам, а `skip` и `take` – пропускают вычисляемое количество значений и берут установленное заранее количество строк.

Контроллеры (controllers) являются частями Express.js, где реализуются функции для формирования ответов на запросы клиента. Они обрабатывают входящие запросы, взаимодействуют с моделями и формируют нужные данные для отправки клиенту. Контроллеры являются промежуточными слоями между маршрутами и моделями, отвечая за логику обработки запросов.

Роуты (routes) определяют пути URL и связанные с ними обработчики контроллеров. Они определяют, какой контроллер будет обрабатывать определенный тип запроса (GET, POST, PUT, DELETE) на определенном пути. Роуты могут также содержать параметры и динамические сегменты URL, чтобы обрабатывать различные варианты запросов. Основной роут `indexRoute.js` который распределяет запросы по разным роутам представлен на листинге 3.2

```
const Route = require('express').Router();
Route.use('/users', UserRoute);
Route.use('/books', BookRoute);
Route.use('/feedback', FeedbackRoute);
Route.use('/bookmark', BookmarkRoute);
Route.use('/files', fileRoute);
Route.use('/tag', tagRoute);
Route.use('/genre', genreRoute);
Route.use('/author', authorRoute);
module.exports = Route;
```

Листинг 3.2 – файл `indexRoute.js`

Файлы для обрабоотки ошибок представляют из себя пользовательский класс `ApiError`, код которого можно увидеть на листинге 3.3

```
class ApiError extends Error {
  constructor(status, message) {
    super();    this.status = status;    this.message = message;
  }
  static badRequest(message) {
    return new ApiError(404, message);
  }
  static unauthorized(message) {
    return new ApiError(401, message);
  }
  static forbidden(message) {
    return new ApiError(403, message);
  }
  static internal(message) {
    return new ApiError(500, message);
  }
}
module.exports = ApiError;
```

Листинг 3.3 – файл `ApiError.js`

Блок `middleware` представляет собой обработку ошибок и так же реализованы проверка авторизован ли пользователь и проверка на соответствие роли.

Рассмотрим код `authMiddleware`, представленный в листинге 3.4

```
module.exports = function (req, res, next) {
  if (req.method === 'OPTIONS') {
    return next();
  }
  try {
    const token = req.headers.authorization.split(' ')[1];
    if (!token) {
      return next(
        ApiError.unauthorized('Не авторизован(Не передан токен)');
      )
    }

    const decoded = verify(token, process.env.SECRET_KEY);
    req.user = decoded;
    console.log("decoded: ", decoded);
    return next();
  }
  catch (e) {
    return next(ApiError.unauthorized(
      'Не авторизован(Непредвиденная ошибка)' + e.message));
  }
}
```

Листинг 3.3 – файл `ApiError.js`

Данный код достает из заголовка запроса токен и декодирует его используя секретный ключ. Если же токен не был получен или не смог декодироваться то возникает ошибка.

В данном примере видно, что для регистрации, авторизации и аутентификации используется `jwt` – авторизация. Это довольно простой и защищенный способ авторизации пользователей.

Так же в серверной части используются статические файлы, такие как статические функции вычисляющие `skip` значения по номеру страницы.

Обработка статических файлов происходит за счет `express.static` он позволяет без проблем отображать файлы на клиентской части.

Так же на сервере реализован `web-socket`. Он используется для мгновенного отображение отзывов на странице книги. Это возможно с помощью модуля `socket.io`.

`Socket.IO` – это библиотека JavaScript для веб-приложений реального времени . Он обеспечивает двустороннюю связь в реальном времени между веб-клиентами и серверами. Он состоит из двух частей: клиентской библиотеки, которая запускается в браузере, и серверной библиотеки для `node.js`. Оба компонента имеют идентичный API.[4]

Код реализации web-socket предоставлен на листинге 3.4

```
const io = require("socket.io")(http, {
  cors: {
    origin: "http://localhost:3000",
    methods: ["GET", "POST"]
  }
});
io.on('connection', (socket) => {
  socket.on('disconnect', () => {
    console.log('user disconnected');
  });
  socket.on('message', (msg) => {
    feedback.addFeedback(msg.id, msg.id_Book, msg.raiting,
msg.text).then((res) => {
      io.emit('message', res);
    });
  });
});
});
```

Листинг 3.4 – Реализация web-socket

На в данном коде сокет прослушивает два состояния «message» и «disconnect». Сообщения принимаю сообщения и отправляют его всем клиентам в сети.

3.2 Разработка клиентской части

Клиентская часть разработана на react, что имеет собственную структура для реализации react api. Поэтому мы будем рассматривать лишь структуру папки src. Его структура предоставлена на рисунке 3.2

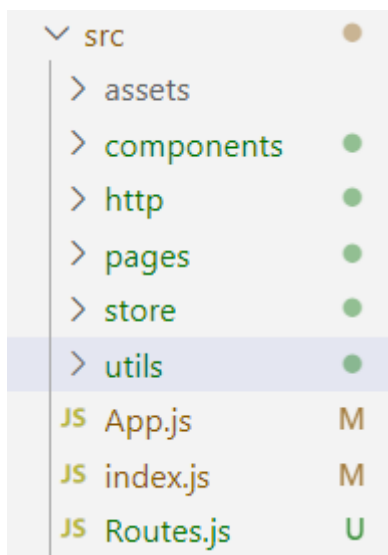


Рисунок 3.2 – Структура проекта клиента

Директория "src" содержит исходный код клиентской части приложения. Внутри нее располагаются следующие файлы и папки:

- components: Эта папка содержит компоненты, которые используются для создания пользовательского интерфейса. Компоненты представляют собой переиспользуемые блоки кода, которые описывают отдельные элементы интерфейса или части страницы. Примеры компонентов могут включать формы, таблицы, кнопки и другие интерактивные элементы.
- pages: Здесь находятся компоненты, которые представляют отдельные страницы или экраны приложения. Каждая страница обычно состоит из нескольких компонентов и определяет структуру и содержимое конкретной части интерфейса;
- http: В этой папке содержатся службы (services), которые обеспечивают взаимодействие клиентской части приложения с сервером. Сервисы выполняют функции отправки HTTP-запросов на сервер, обработки ответов и управления данными. Например, здесь может быть сервис для работы с API сервера, сервис для работы с авторизацией и другие;
- utils: В данной папке содержатся вспомогательные утилиты и функции, которые могут использоваться в разных частях клиентского приложения. Это могут быть функции для обработки данных, форматирования или других задач, которые упрощают разработку и повторное использование кода;
- store: Здесь находятся файлы реализующие некоторые классы. По типу UserStore и BookStore. Которые служат для передачи данных между компонентами в Context;
- App.js и index.js: Эти файлы являются основной точкой входа в клиентское приложение. App.js определяет основную структуру и маршрутизацию приложения, включая вложенные компоненты и их расположение. index.js отвечает за инициализацию приложения и его рендеринг в корневой элемент HTML-страницы.

В клиентской части приложения используется библиотека React Bootstrap, которая предоставляет набор предопределенных компонентов для создания пользовательского интерфейса на основе React. React Bootstrap комбинирует возможности React и стилей.

React Bootstrap предлагает множество других компонентов, таких как навигационное меню (Navbar), формы (Form), модальные окна (Modal) и многое другое. Вы можете ознакомиться с полным набором доступных компонентов и их возможностями в документации на официальном сайте React Bootstrap. Пример кода для импорта и использования компонента Navbar из React предоставлен в приложении А. В этом примере мы импортируем компоненты Navbar и Nav из библиотеки React Bootstrap и используем их для создания навигационного меню.

Так же для чтения файлов «.epub» был использован модуль ReactReader. React Reader — это реактивная оболочка для epub.js, использующая ветку v.03. epub.js — отличная библиотека, и это оболочка для нее. Эта оболочка упрощает использование в React-приложении[4].

Также обратите внимание, что EpubJS — это браузерный epub-ридер, и он работает, отображая текущую epub-главу в iframe, а затем с помощью CSS-колонок отображает текущую страницу[5].

Пример части кода компонента для чтения книг представлен на листинге 3.5

```
<Card>
  {alertshow && <Alert variant={variant} className="mt-3"
onClose={() => setAlertshow(false)} dismissible={alertText}</Alert>}
  <div style={{ height: '100vh' }}>
    <ReactReader locationChanged={locationChanged}
      {...location && { location }}
      url={process.env.REACT_APP_API_URL + 'Book' + data +
'.epub'}
      getRendition={rendition => (renditionRef.current =
rendition)}
      tocChanged={toc => (tocRef.current = toc)}
    />
  </div>
</Card>
```

Листинг 3.5 – Реализация ReactReader

В данном примере представлен код вывода на экран книги с возможностью перелистывать страницы и выбора глав.

Так же на клиенте как сервере реализован web-socket. Часть реализации представлен на листинге 3.6

```
const socket = io('http://localhost:5000');
useEffect(() => {
  try {
    socket.on('message', (msg) => {
      console.log(msg);
      setMessages([...messages, msg]);
      console.log(messages);
    });
  } catch (e) {
    setAlertshow(true);
    setVariant('danger');
    setAlertText('Произошла ошибка: ' +
e.response.data.message);
  }
}, [messages]);
```

Листинг 3.6 – Реализация web-socket

В данном коде клиент отправляет отзыв на сервер и получает сообщения от других пользователей.

4 Тестирование web-приложения

Тестирование проекта является важной частью разработки программного обеспечения, поскольку оно помогает обнаруживать ошибки и уверенно утверждать, что система работает корректно. Для тестирования проекта можно использовать различные подходы и инструменты.

Первым на что может наткнуться пользователь окно входа и регистрации. В данном окне могут возникнуть ошибку входа пример такой ошибки представлено на рисунке 4.1

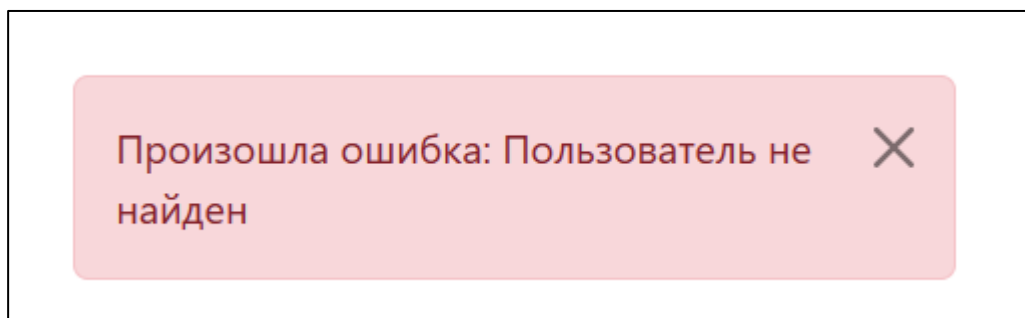


Рисунок 4.1 – Ошибка некорректных данных

Кроме этого поля также проверяются на корректное заполнение. Например, при входе почта и пароль должны быть длиной как минимум 3 символа. А почта должна быть корректной. Случай такой ошибки представлен на рисунке 4.3.

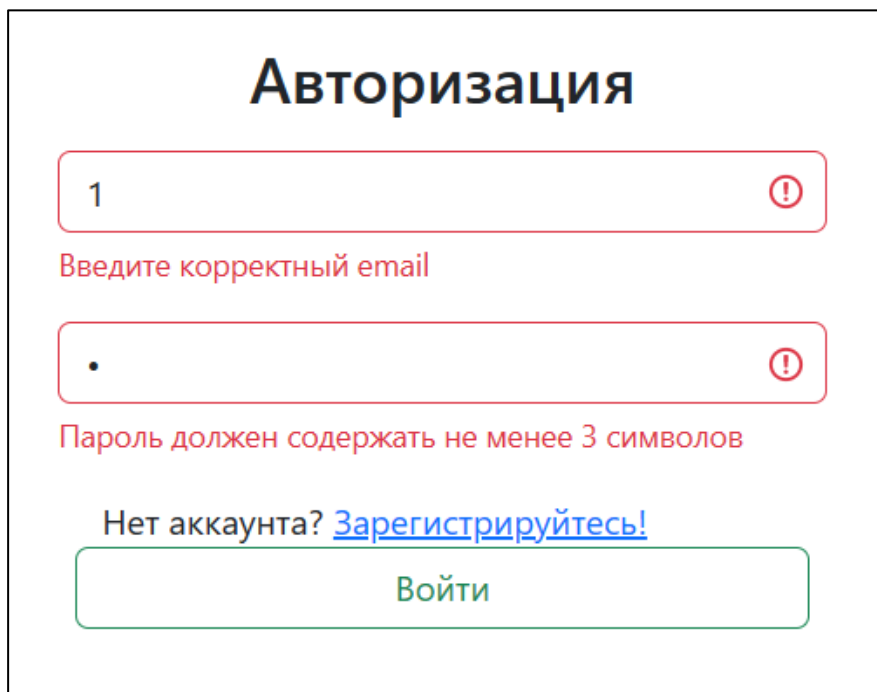
A screenshot of a login form titled "Авторизация" in bold black text. The form has two input fields. The first field contains the number "1" and has a red border with a red exclamation mark icon on the right. Below it is the red text "Введите корректный email". The second field contains a single dot "." and also has a red border with a red exclamation mark icon on the right. Below it is the red text "Пароль должен содержать не менее 3 символов". At the bottom, there is a link "Нет аккаунта? [Зарегистрируйтесь!](#)" in blue text, and a green button labeled "Войти" in green text.

Рисунок 4.2 – Ошибка заполнения полей password и email

В данном случае, валидация осуществляется с использованием простых проверок в функциях `validateEmail` и `validatePassword`, которые вызываются при потере фокуса (событие `onBlur`) полей ввода.

Ошибки могут возникать так же и в панели администратора, например при добавлении уже существующего тега. Пример представлен на рисунке 4.3

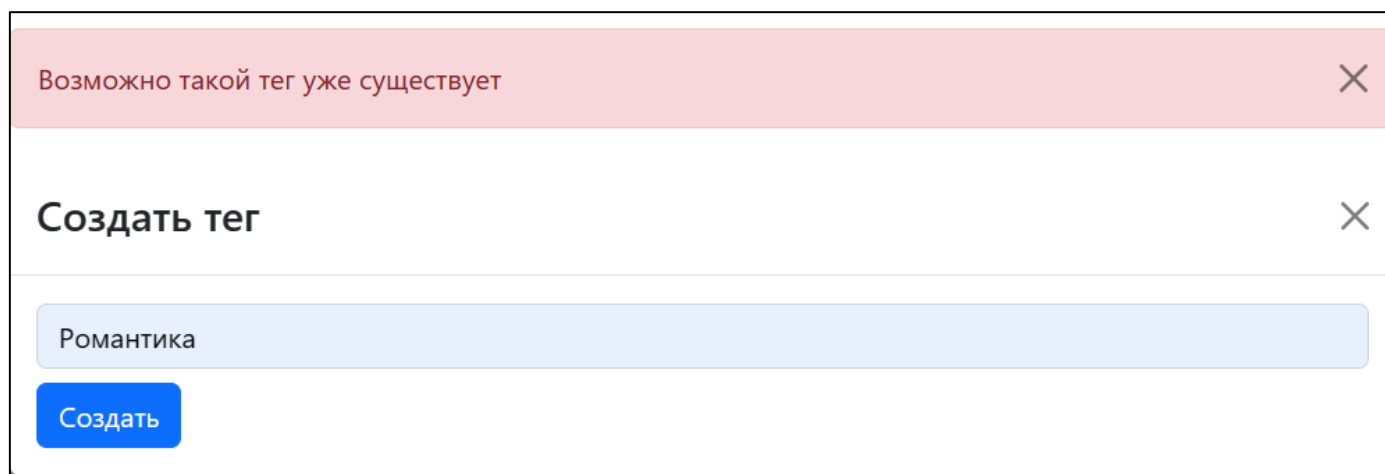


Рисунок 4.3– Ошибка создания тега

Таким образом мы не позволяем заполнить базу данных одинаковыми данными. В случае если сервер или бд будут не доступны, то возникнет ошибка. Это можно увидеть на рисунке 4.4

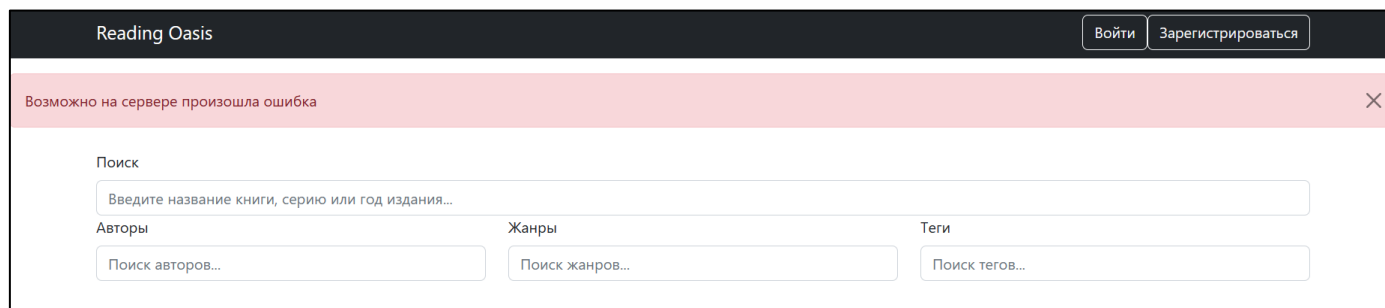


Рисунок 4.4 – Ошибка при неработающем сервере или бд

На клиенте так же могут возникнуть и другие непредвиденные ошибки. Но если пользователь попытается попасть на неразрешенный ему ресурс то, клиент просто вернет его на главную.

В результате тестирования приложения было обнаружено, что компоненты корректно рендерятся и взаимодействуют с пользовательскими действиями. Также было успешно протестировано сохранение и обновление прогресса чтения книги, а также отображение информации о текущей странице. В целом, приложение прошло основные тестовые сценарии и работает стабильно.

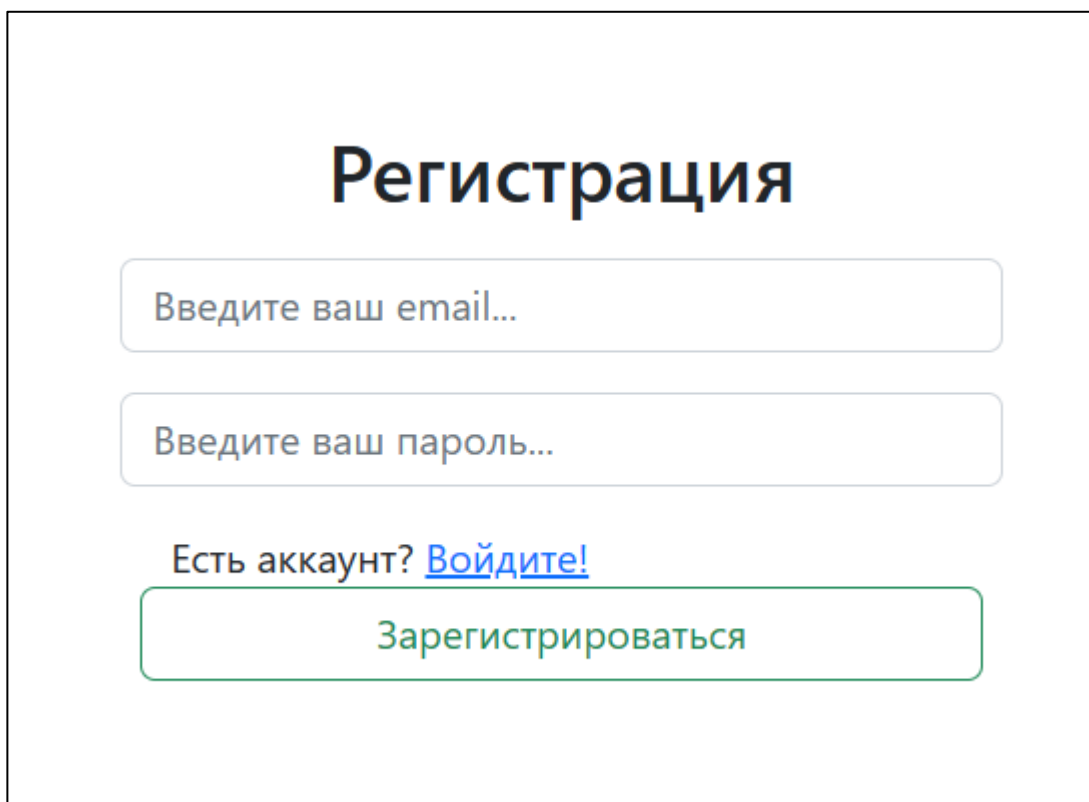
5 Руководство пользователя

5.1 Регистрация пользователя

Для создания аккаунта нужно следовать следующим шагам:

- Зайдите на главную страницу приложения.
- Нажмите на ссылку "Регистрация" или "Создать аккаунт".
- Заполните необходимые поля в форме регистрации, включая имя пользователя, адрес электронной почты и пароль.
- Подтвердите правильность введенной информации.
- Нажмите кнопку "Зарегистрироваться" для создания аккаунта.
- Если все данные введены правильно, вы будете перенаправлены на страницу входа в систему.

Форма для регистрации предоставлена на рисунке 5.1.



The image shows a registration form with the title "Регистрация" in a large, bold, black font. Below the title are two input fields: "Введите ваш email..." and "Введите ваш пароль...". Below these fields is a link "Есть аккаунт? [Войдите!](#)". At the bottom is a green button with the text "Зарегистрироваться".

Рисунок 5.1 – Окно регистрации

Форма имеет валидацию поэтому нужно соблюдать пару правил, почта, должна быть корректной, а пароль не менее 3 символов. Если не соблюдать валидацию зарегистрироваться не получится. Так же надо помнить на одну почту один аккаунт. При регистрации в localStorage сохраняются данные зарегистрированного пользователя.

5.2 Аутентификация и авторизация пользователя

Для входа в систему необходимо ввести свою электронную почту, пароль и нажать на кнопку "Войти". Это позволит системе проверить наличие пользователя с указанными данными. Все поля обязательны для заполнения.

Если аутентификация прошла успешно и были соблюдены все вышеуказанные правила, пользователь будет перенаправлен на главную страницу, что означает успешную авторизацию. На рисунке 5.2 представлен скриншот формы авторизации.

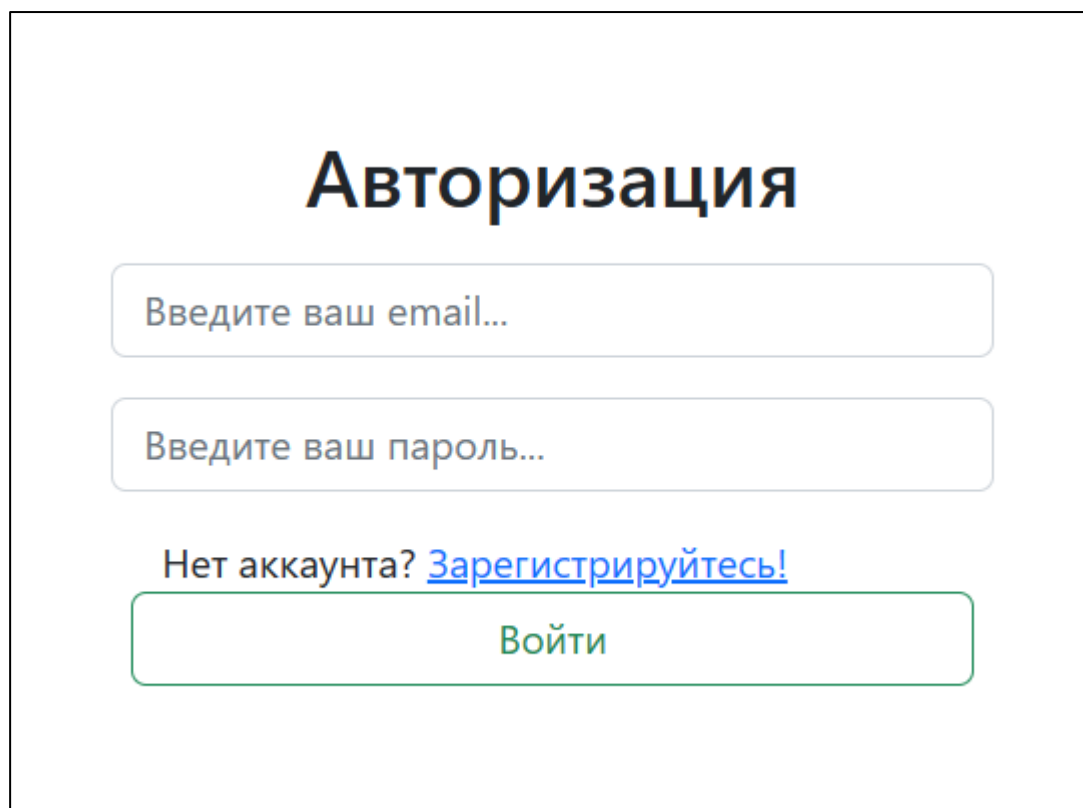
Скриншот веб-формы авторизации. В центре экрана крупным черным шрифтом написано слово "Авторизация". Ниже него расположены два белых поля с серой обводкой. Первое поле содержит текст "Введите ваш email...", второе — "Введите ваш пароль...". Под этими полями находится ссылка "Нет аккаунта? [Зарегистрируйтесь!](#)" в синем цвете. В самом низу формы находится зеленая кнопка с белым текстом "Войти".

Рисунок 5.2 – Форма авторизации

После успешного входа пользователь получит доступ к большей части функций, и так же к редактированию собственного профиля.

Для использования некоторых функций приложения необходимо пройти процесс авторизации. Если пользователь не авторизован, он не сможет получить доступ к определенным страницам и функциям приложения. Для получения доступа к полному функционалу необходимо пройти процесс авторизации.

5.3 Главная страница

При успешном логине пользователь попадает на главную страницу. На этой странице можно совершить поиск и фильтрацию по различным параметрам, таким как жанры, теги, авторы, серии, названия. Так же на странице виден рейтинг книг, который вычисляется по отзывам других пользователей.

Данная страница предоставлена на рисунке 5.3.

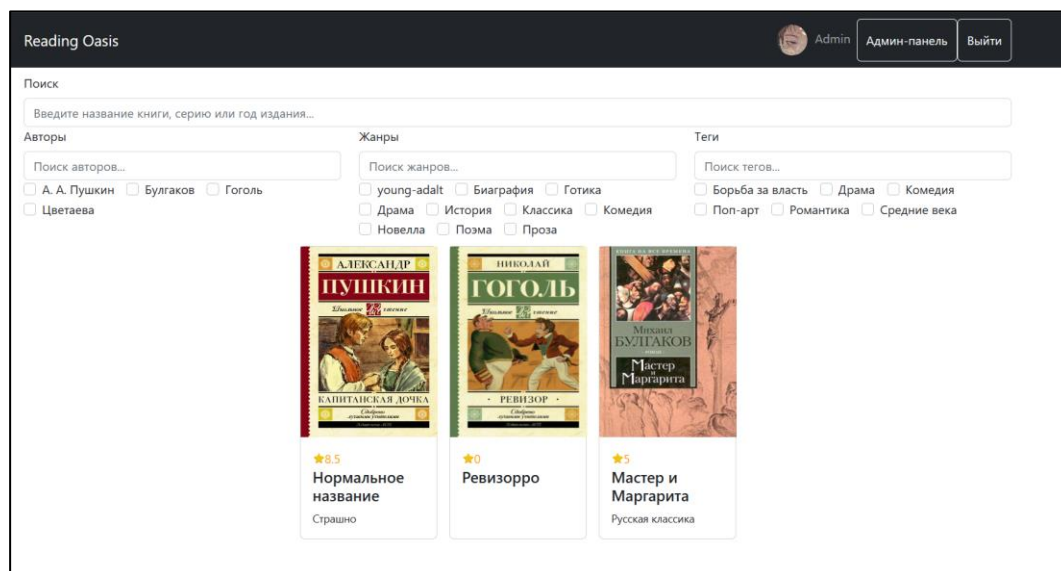


Рисунок 5.3 – Страница поиска

На данной странице можно перейти к чтению книг или просмотреть собственные списки перейдя в профиль пользователя.

5.4 Страница книги

На данной странице можно узнать более подробно информацию о книге, прочитать её. Страница представлена на рисунке 5.4

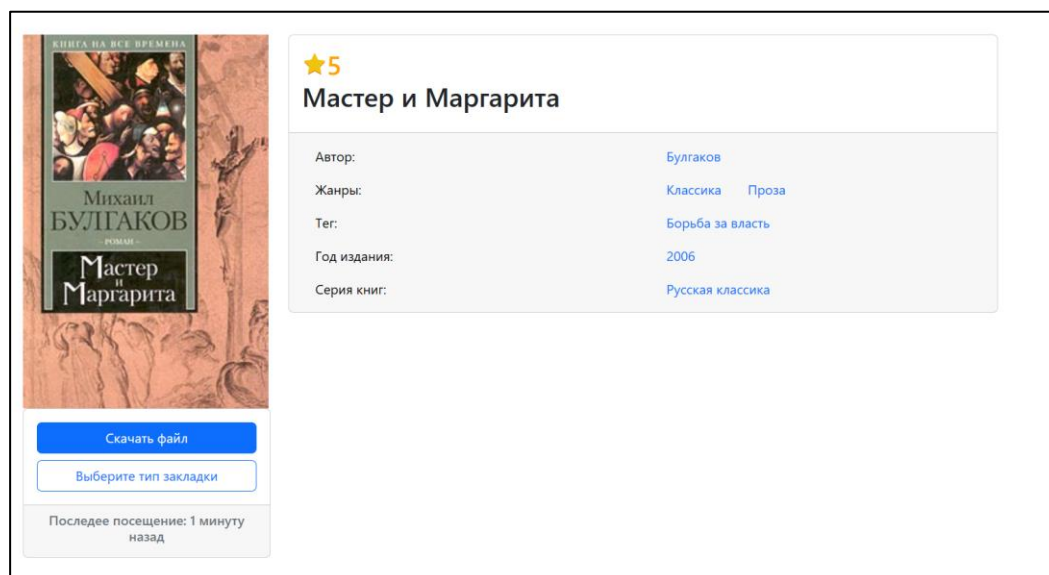


Рисунок 5.3 – Страница поиска

Так же на странице можно добавить закладку и оставить отзыв.

5.5 Личная страница пользователя

Нажав на аватарку или имя можно попасть на страницу пользователя. Эта страница представлена на рисунке 5.5.

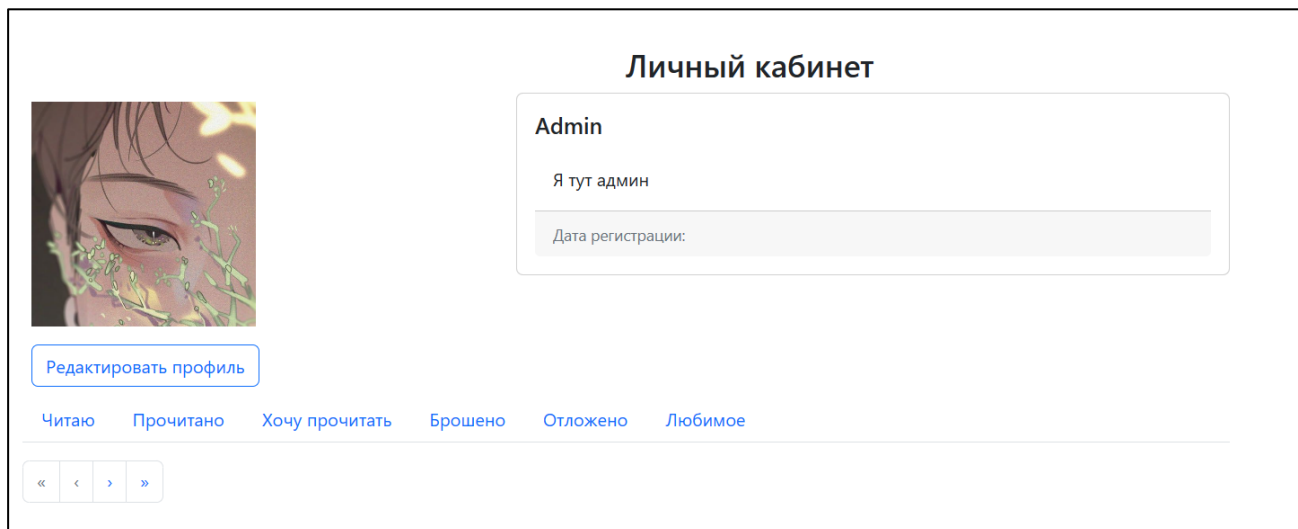


Рисунок 5.5 – Личная страница пользователя

На этой странице представлена основная информация о пользователе, так же можно изменить информацию о пользователе, или загрузить картинку.

5.6 Панель администратора

В данном курсовом проекте была разработана возможность администрирования. Страница панели администратора представлена на рисунке 5.6.

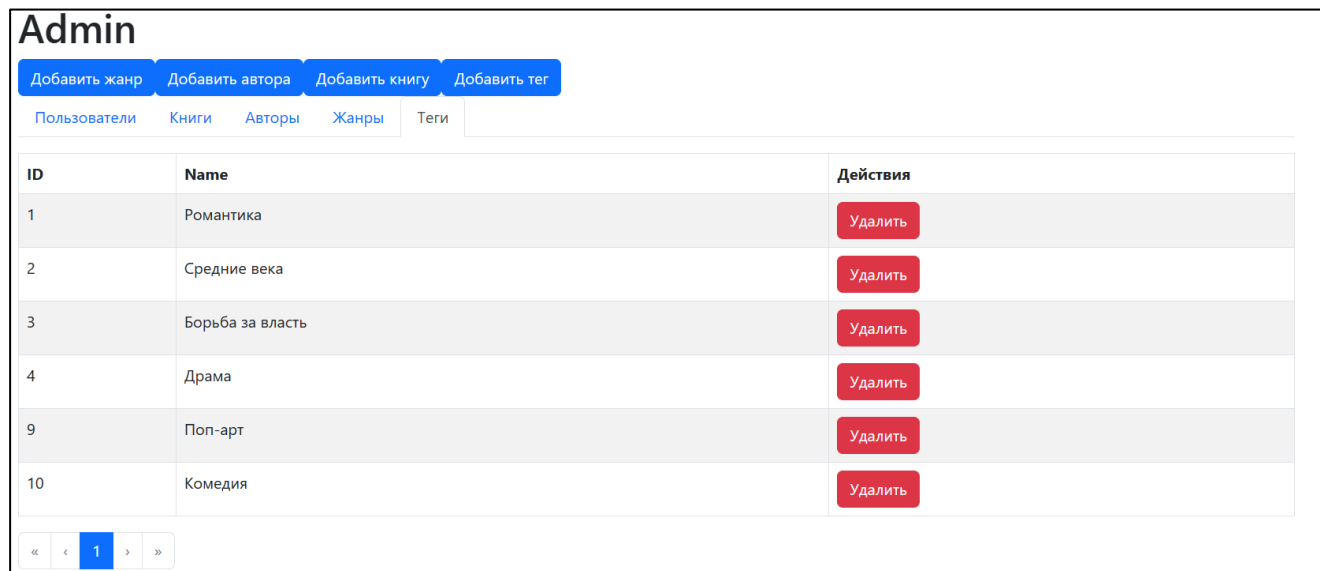


Рисунок 5.6 – Панель администратора

Администратор может добавлять и удалять жанры, авторов, теги, книги пользователей. Так же можно изменить книги. Страница имеет переключения по старикам таблиц. На рисунке 5.7 представлена форма создания жанра.



Рисунок 5.7 – Форма создания записи

При изменении книги мы перейдем на страницу изменения книги. Это представлено на странице 5.8

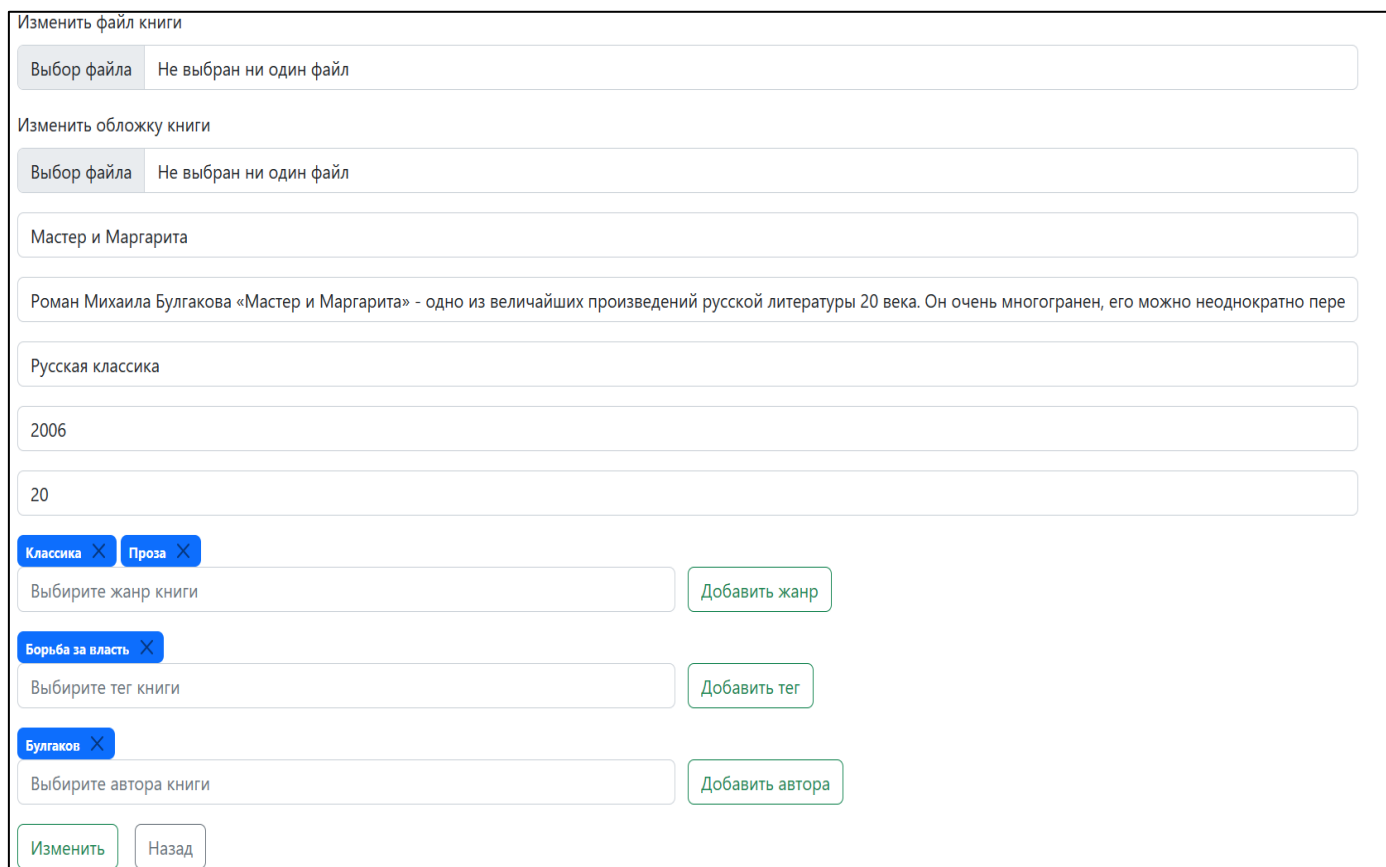


Рисунок 5.8 – Форма изменения книги

Тут можно добавить книге жанров, тегов и авторов, а также обновить обложку и файл.

5.7 Удаление комментарием

Администратор, может удалять комментарии у книги. Кнопка удаление представлена на рисунке 5.9.

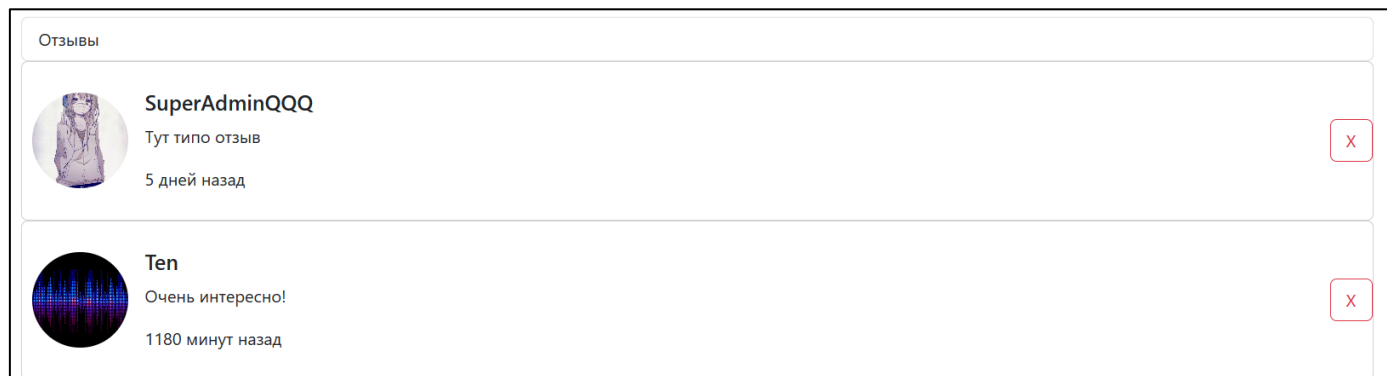


Рисунок 5.9 – Кнопка удаление

Администратор, как и пользователь может модерировать собственные списки книг. Пример таких списков представлен на рисунке 5.10

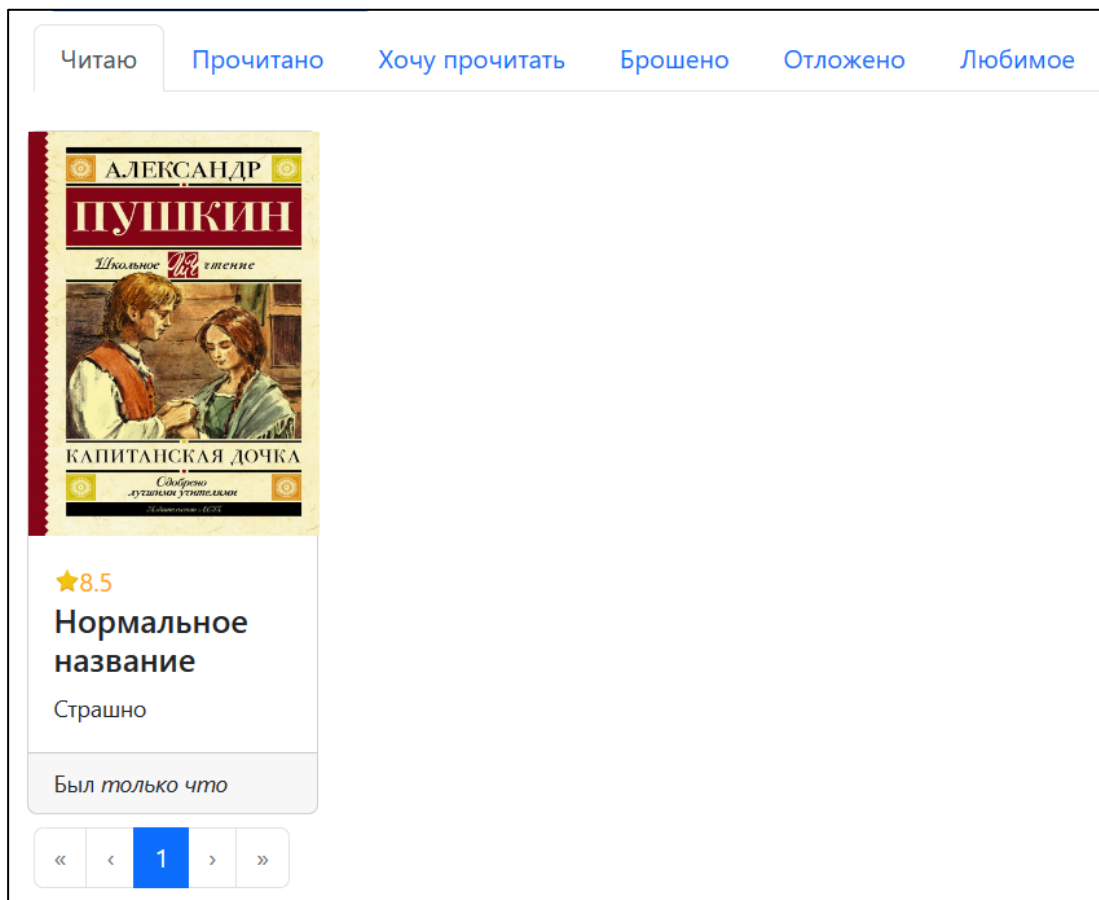


Рисунок 5.10 Личные списки книг

Тут представлены различные списки книг, так же видно когда последний обновлялась закладка.

Заключение

В процессе решения поставленной задачи была достигнута поставленная цель по Функциональное web-приложение, разработанное с использованием технологий React, Bootstrap, Express, и Mysql, представляет собой полноценную библиотеку для чтения и управления информацией о книгах. Оно обладает следующими основными возможностями:

- Регистрация и авторизация: Пользователи могут создавать учетные записи и входить в систему для доступа к функциональности приложения. Это обеспечивает безопасность и персонализированный опыт использования;
- Роли администратора и пользователя: Приложение поддерживает две роли - администратора и пользователя. Администратор имеет расширенные привилегии, такие как добавление и редактирование книг, управление пользователями и отзывами. Пользователи могут просматривать информацию о книгах, оставлять отзывы и устанавливать оценки;
- Добавление и редактирование книг: Администратор может добавлять новые книги в библиотеку, предоставляя информацию о них, такую как название, автор, жанр и другие характеристики. Также имеется возможность редактировать информацию о существующих книгах;
- Чтение и скачивание книг: Пользователи могут читать книги онлайн и скачивать их для офлайн-чтения. Приложение обеспечивает удобный интерфейс для чтения и предоставляет возможность загрузки книг на устройства пользователей;
- Поиск книг: В приложении реализован поиск книг по различным критериям, включая название, автор, жанр и другие параметры. Это позволяет пользователям быстро находить интересующие их книги;
- Информация о книгах: Пользователи могут просматривать подробную информацию о каждой книге, включая обложку, описание, рейтинги, отзывы и другие характеристики. Это помогает им принимать решение о прочтении книги;
- Отзывы и оценки: Пользователи могут оставлять отзывы на книги и ставить им оценки. Это позволяет пользователям делиться своим мнением о книгах и помогает другим пользователям принять решение о прочтении;
- Личные списки: Пользователи могут создавать личные списки прочитанных книг, книг, которые они хотят прочитать, а также список своих любимых книг. Это помогает пользователям организовать свою библиотеку и делиться рекомендациями с другими.

Таким образом, функциональное web-приложение предоставляет все необходимые возможности для удобного управления библиотекой книг. Оно сочетает в себе современные технологии разработки с понятным и интуитивным пользовательским интерфейсом, что делает его привлекательным и удобным в использовании для пользователей.

Список используемых источников

1. Что такое Bootstrap и зачем он нужен? статья [Электронный ресурс]. – Режим доступа: <https://itchief.ru/bootstrap/introduction> Дата доступа: 07.04.2023
2. Веб-фреймворк Express статья [Электронный ресурс] Режим доступа: https://developer.mozilla.org/ru/docs/Learn/Server-side/Express_Nodejs Дата доступа: 14.05.2023
3. REST API статья [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/articles/483202/>– Дата доступа: 10.05.2023.
4. ReactReader документация [Электронный ресурс].– Режим доступа: <https://www.npmjs.com/package/react-reader>.– Дата доступа: 14.05.2023
5. Scket.io документация [Электронный ресурс]. – Режим доступа: <https://coderlessons.com/tutorials/kompiuternoe-programmirovanie/uznaite-socket-io/socket-io-kratkoe-rukovodstvo>– Дата доступа: 15.05.2023

Приложение А

```

<Navbar bg="dark" variant="dark">
  <Container>
    <Navbar.Brand href={BOOKS_ROUTE}>Reading Oasis</Navbar.Brand>
    <Nav className="justify-content-end">
      {user.isAuthenticated ? (
        <>
          <Link to={`/${USER_ROUTE}/${id}`}
            className="nav-link"
            title="Перейти в профиль">
            <img
              src={`/${process.env.REACT_APP_API_URL}/User${id}.jpg`} className="rounded-circle"
              style={{ width: '40px', height: '40px', objectFit: 'cover', marginRight: '10px' }}
              alt="Avatar" />
            <span className="d-none d-md-inline">{name}</span>
          </Link>
          {role === 'ADMIN' &&
            <Button variant="outline-light"
              href={ADMIN_ROUTE} className="ml-2 d-flex align-items-center">
              <span className="d-inline-block text-truncate">Админ-панель</span>
            </Button>
            <Button variant="outline-light"
              onClick={() => logout()} className="ml-2">Выйти</Button>
          </>
          ) : (
            <>
              <Button variant="outline-light"
                href={LOGIN_ROUTE} className="mr-2">Войти</Button>
              <Button variant="outline-light"
                href={REGISTRATION_ROUTE}>Зарегистрироваться</Button>
            </>
          )}
        </Nav>

```