Министерство образования Иркутской области
Государственное бюджетное профессиональное
образовательное учреждение Иркутской области
«Иркутский авиационный техникум»
(ГБПОУИО «ИАТ»)

КП.09.02.07-3.24.211.14 ПЗ

ВЕБ-ПРИЛОЖЕНИЕ ЭЛЕКТРОННЫХ КНИГ «ФАНТАСТИКА»

Председатель ВЦК: _____ (А.С. Александрова)
(подпись, дата)

Руководитель: _____ (Н.Р. Карпова)
(подпись, дата)

Студент: _____ (А.С. Нестерук)
(подпись, дата)

Иркутск 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Тема космической фантастики остается одной из самых популярных и захватывающих тем в литературе и культуре.

Такой проект может быть актуален для широкого круга пользователей, в том числе для любителей научной фантастики, мировой космической тематики и тех, кто ищет новые способы доступа к литературе в цифровом формате.

Целью курсового проекта является разработка веб-приложение электронных книг тематики космической фантастики «Фантастика», которое обеспечить пользователям доступ к широкому выбору книг данной тематики, удобное чтение через веб-интерфейс и возможность взаимодействия с другими любителями космической фантастики.

Для реализации такого проекта были поставлены следующие задачи:

1) Провести предпроектное исследование.
2) Составить техническое задание.
3) Спроектировать веб-приложение.
4) Разработать веб-приложение.
5) Составить технологическую документацию для веб-приложения.

Выполнение данных задач поможет создать собственное веб-приложение для чтения книг.

Развитие подобного веб-приложения может способствовать расширению круга читателей данного жанра, поддерживать и развивать культуру чтения, а также повышать интерес к космосу и научным технологиям через призму литературы.

Веб-приложение «Фантастика» представляет собой электронную библиотеку книг, посвященных космической фантастике. Здесь собраны произведения различных авторов, от классиков жанра до современных писателей, чьи книги переносят нас в удивительные миры космоса. это виртуальная библиотека, посвященная космической фантастике. Читатели погружаются в невероятные и захватывающие истории, где мечты переплетаются с реальностью, а наука с фантазией.

1 Предпроектное исследование

1.1 Описание предметной области

Описание предметной области для веб-приложения «Фантастика»

В качестве предметной области выбрано веб-приложение «Фантастика», которое специализируется на чтении книг жанра космическая фантастика. Приложение предоставляет пользователям возможность читать, обсуждать и делиться своими впечатлениями от прочитанных книг в данном жанре.

Основная деятельность веб-приложения включает в себя:

1) Создание и поддержание библиотеки книг космической фантастики, включая классику жанра и новинки.

2) Организация комментариев для обсуждения книг и тем, связанных с космической фантастикой.

Для эффективной работы веб-приложения необходимо учитывать следующие аспекты:

1) Удобный и интуитивно понятный интерфейс для поиска и чтения книг.

2) Система рекомендаций, основанная на рейтинге пользователей.

3) Безопасная и удобная система комментирования и обсуждения книг.

Веб-приложение «Фантастика» ориентировано на следующие группы пользователей:

1) Любители космической фантастики, которые хотят читать и обсуждать книги в этом жанре.

2) Авторы и издатели, которые хотят рассказать о своих новых книгах и узнать мнение читателей.

3) Специалисты в области космической фантастики, которые хотят следить за последними тенденциями и событиями в жанре.

Функциональные возможности веб-приложения:

1) Ведение базы данных книг космической фантастики (добавление, редактирование, удаление).

2) Обеспечение логической непротиворечивости базы данных.

3) Реализация часто встречающихся запросов в готовом виде (поиск книг по автору, названию, тегам и т.д.).

1.2 Анализ инструментальных средств для разработки

При разработке веб-приложения электронных книг важно выбрать правильные инструменты для успешной разработки и эксплуатации программного продукта.

Перечень инструментов разработки, их использование на различных этапах и обоснование выбора данных средств:

Анализ инструментальных средств для разработки веб-приложения "Фантастика"

1) Языки программирования и среды разработки:

Frontend:

Языки программирования: JavaScript, HTML, CSS.

JavaScript: широко используемый язык программирования для создания интерактивных веб-страниц и динамического контента.

HTML: язык разметки, используемый для структурирования контента на веб-страницах.

CSS: язык стилей, используемый для оформления и макетирования веб-страниц.

Среда разработки: Visual Studio Code.

Visual Studio Code: популярный и функциональный текстовый редактор с открытым исходным кодом, поддерживающий множество языков программирования и предоставляющий множество расширений для удобства разработки.

Фреймворки: React.js.

React.js: популярный JavaScript-фреймворк для разработки пользовательских интерфейсов, предоставляющий мощные инструменты для создания динамических и масштабируемых приложений.

Backend:

Язык программирования: Node.js.

Node.js: JavaScript-платформа, основанная на движке Chrome V8, позволяющая создавать высокопроизводительные серверные приложения.

Среды разработки: Visual Studio Code.

Базы данных: PostgreSQL.

PostgreSQL: мощная, надежная и масштабируемая система управления базами данных с открытым исходным кодом, поддерживающая множество современных функций и стандартов.

2) Средства проектирования и верстки:

Дизайн: Figma.

Figma: веб-приложение для создания макетов и дизайна пользовательских интерфейсов, предоставляющее множество инструментов для совместной работы и удобства использования.

Верстка: Bootstrap.

Bootstrap: популярный набор инструментов для разработки веб-приложений, включающий в себя библиотеку CSS, JavaScript-фреймворк и шаблоны верстки, позволяющие быстро создавать адаптивные и современные веб-страницы.

3) Средства управления версиями:

Git и GitHub для контроля версий и разработки

Обоснование выбора инструментов:

JavaScript и Node.js: выбраны за широкие возможности веб-разработки, большое сообщество разработчиков и возможность использования одного языка на frontend и backend.

PostgreSQL: быстрая и масштабируемая реляционная база данных, удобная для хранения информации и добавленния информации.

Сравнительный анализ инструментов разработки позволяет выделить ключевые критерии, который представлен в таблице 1.

Таблица 1 – Критерии анализа инструментов

Критерий	Описание
Продуктивность и эффективность	Выбранные инструменты являются популярными и широко используемыми в сообществе разработчиков, что обеспечивает поддержку и эффективное решение задач.
Гибкость и масштабируемость	JavaScript и Node.js обладают гибкостью для реализации различных функционалов и легко масштабируются в процессе развития проекта.
Удобство и доступность	Средства разработки, выбранные для frontend и backend, обеспечивают удобную работу разработчиков, сокращая время разработки и обеспечивая высокое качество продукта.

Выбранный набор инструментов обеспечивает эффективное развертывание веб-приложения электронных книг, отвечая требованиям современных технологий и потребностям конкретного проекта. Полученный анализ поможет оптимизировать процесс разработки и обеспечить реализацию успешных и качественных решений.

2 Техническое задание

В начале разработки создавалось техническое задание, в котором указывались основные требования.

Для создания технического задания использовался стандарт ГОСТ 34.602-2020.

Согласно ГОСТ 34.602-2020 техническое задание должно включать следующие разделы:

1. общие сведения;
2. назначение и цели создания веб-приложения;
3. требования к системе в целом;
4. требования к структуре и функционированию веб-приложения;
5. требования к надежности;
6. требования к безопасности;
7. требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы;
8. требования к документированию;
9. состав и содержание работ по созданию веб-приложения.

Техническое задание на разработку веб-приложения представлено в приложении А.

3 Проектирование веб-приложение

3.1 Функциональная схема веб-приложение

Функциональное проектирование веб-приложения "Фантастика" для чтения книг жанра космическая фантастика включает в себя следующие пункты:

1. Регистрация и аутентификация пользователей:

Создание системы регистрации и входа пользователей с использованием электронной почты или логина и пароля.

2. Личный кабинет пользователя:

Отображение информации о пользователе, включая имя, электронную почту и дату создания аккаунта.

Сохранение прогресса чтения и последней прочитанной страницы для каждой книги.

3. Библиотека книг:

Отображение списка доступных книг жанра космическая фантастика с возможностью фильтрации и сортировки по различным параметрам (автор, название, и т.д.).

Детальная информация о каждой книге, включая описание, обложку, автора, год издания и рейтинг.

4. Чтение книг:

– Возможность чтения книг онлайн с адаптивным дизайном для различных устройств.

– Возможность добавления закладок и заметок в процессе чтения.

5. Комментарии:

– Возможность оставлять комментарии к книгам и участвовать в дискуссиях с другими пользователями.

На рисунке 1 представлена диаграмма прецедентов Uses CASE в разработке веб-приложения, представляющая визуальное представление функциональных требований приложения. Она описывает то, как различные пользователи (актеры) будут взаимодействовать с приложением и как приложение будет отвечать на их действия.

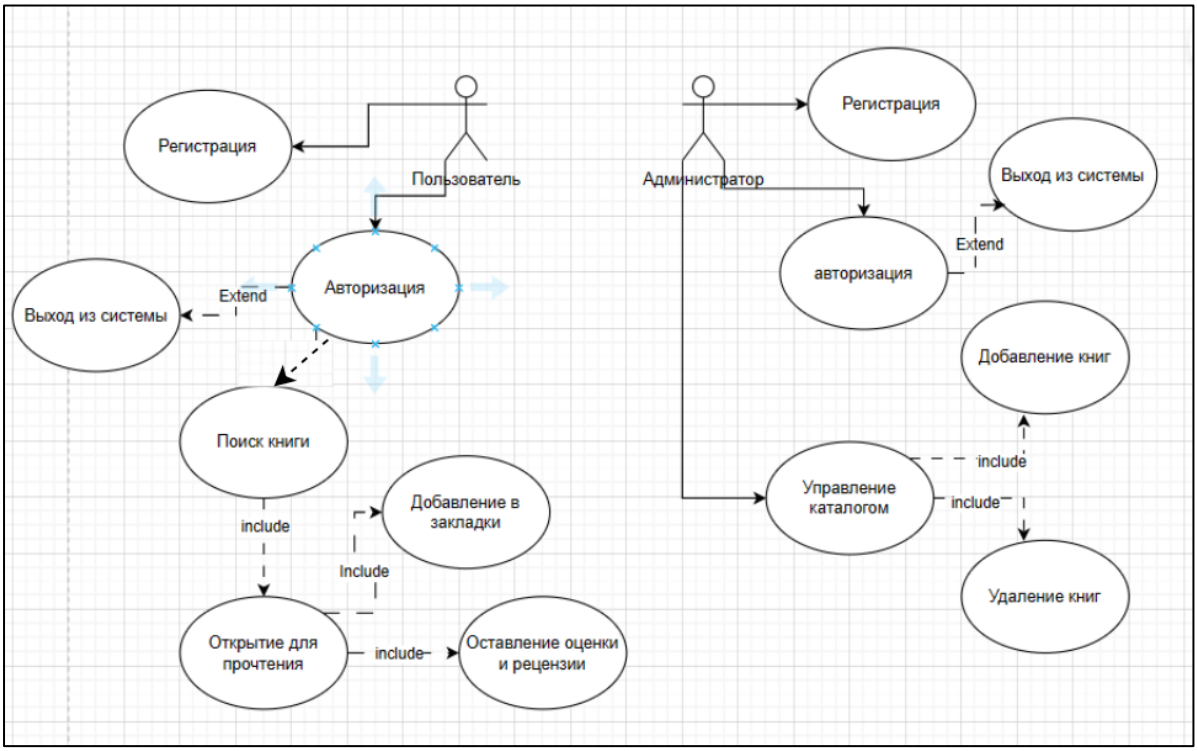

Рисунок 1 – Диаграмма прецедентов Uses CASE

На рисунке 2 представлена диаграмма потоков данных DFD которая является графическим инструментом для описания того, как данные перемещаются внутри веб-приложения. Она помогает визуализировать потоки информации, обработку данных и взаимодействие компонентов приложения.

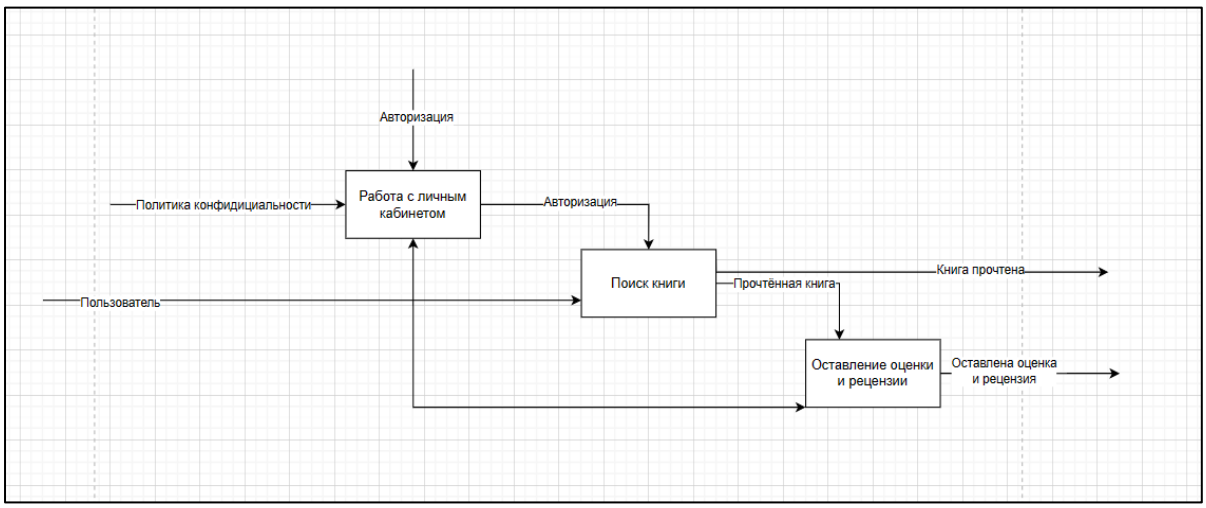

Рисунок 2 – Диаграмма потоков данных DFD

3.2 Структурная схема веб-приложение

На рисунке 3 представлена диаграмма классов, которая представляет собой структурную диаграмму, описывающая структуру классов в веб-приложении, их атрибуты, методы и взаимосвязи.

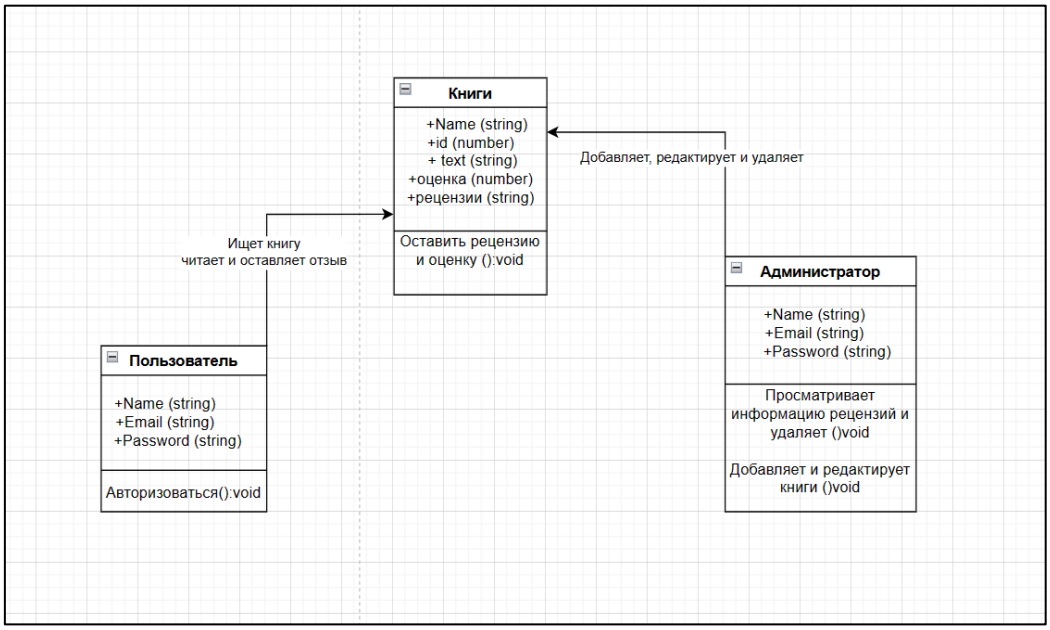

Рисунок 3 – Диаграмма классов

На рисунке 4 представлена диаграмма последовательности для веб-приложения электронных книг.

Диаграмма показывает, как пользователь взаимодействует с различными компонентами веб-приложения для чтения.

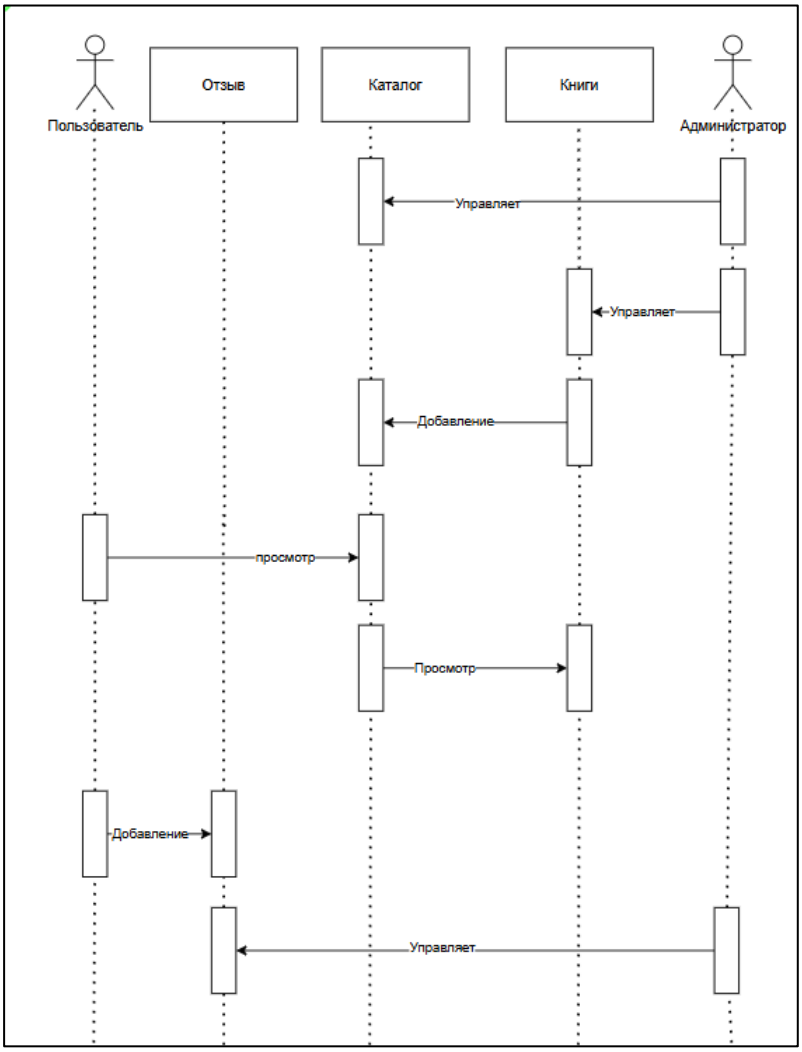

Рисунок 4 – Диаграмма последовательности

3.3 Проектирование баз данных

ER-модель веб-приложения электронных книг позволяет визуализировать структуру данных и связей между ними, а также способствует проектированию, нормализации базы данных и обеспечивает понимание комплексных взаимосвязей. Это важный инструмент при проектировании и разработке баз данных, он помогает создать эффективную структуру хранения и операций с данными. (Рисунок 5)

ER-модель содержит в себе 5 сущностей:

1. Сущность «Пользователи» (Users).
2. Сущность «Книга» (Books).
3. Сущность «Жанр» (Genre).
4. Сущность «Отзыв» (Review).

5. Сущность «Закладка» (BookMark).

Связь между сущностями:

Сущность «Пользователь» имеет связь с сущностью «Книги» с отношением многие-ко-многим через промежуточную таблицу «Отзыв».Промежуточная сущность содержит внешнее поле «UserId», которое ссылается на ID из таблицы «Пользователи» и поле «BookId», которое ссылается на ID из таблицы «Книги».

Сущность «Пользователь» имеет связь с сущностью «Книги» с отношением многие-ко-многим через промежуточную таблицу «Закладка». Промежуточная сущность содержит внешнее поле «UserId», которое ссылается на ID из таблицы «Пользователи» и поле «BookId», которое ссылается на ID из таблицы «Книги».

Нормальная форма в ER-модели:

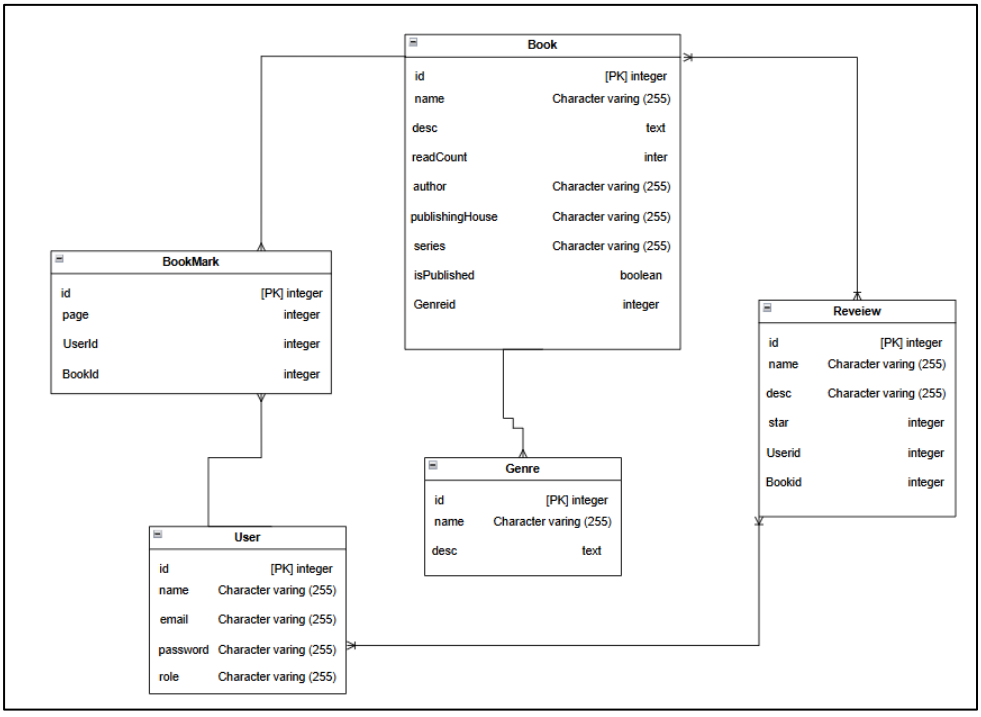

Рисунок 5 – ER-модель

Данная ER-модель соответствует третьей нормальной форме (3NF). Все атрибуты каждой сущности являются атомарными и не содержат повторяющихся групп. Нет транзитивных зависимостей между не ключевыми полями, и каждая сущность имеет один первичный ключ.

Описание ER-модели:

На таблицах 2 – 6 представлено описание ER-модели интернет-магазина.

Таблица 2 – Users

Поля	Тип данных	Значение
id	Int(PK)	Идентификатор пользователя (ключ.)
name	Varchar(255)	Имя пользователя
mail	Varchar(255)	Почта пользователя
password	Varchar(45)	Пароль пользователя
Price	Int	Дата создания пользователя
createdAt	TIMESTAMPTZ	Системные требования
updatedAt	TIMESTAMPTZ	Дата редактирования пользователя

Таблица 3 – Books

Поля	Тип данных	Значение
Id	Int(PK)	Идентификатор книги (ключ.)
name	Varchar(255)	Название книги
desc	Varchar(45)	Описание
readCount	Varchar(45)	Статистика прочтения
author	Varchar(255)	Автор книги
publishing House	Varchar(255)	Издательство
series	Varchar(255)	Серия книги
isPublished	boolean	Статус публикации
GenreId	Int	ID жанра

Таблица 4 – Genre

Поля	Тип данных	Значение
Id	Int(PK)	Идентификатор жанра (ключ.)
name	Varchar(255)	Название жанра
desc	text	Описание

Таблица 5 – Review

Поля	Тип данных	Значение
Id	Int(PK)	Идентификатор отзыва (ключ.)

name	Varchar(255)	Имя пользователя
desc	Varchar(45)	Описание
stars	Int	Кол-во звёзд
UserId	Int	ID автора отзыва
BookId	Int	ID книги из отзыва

Таблица 6 – BookMark

Поля	Тип данных	Значение
id	Int(PK)	Идентификатор закладки (ключ.)
page	Int	Страница
UserId	Int	ID пользователя
BookId	Int	ID книги

3.4 Проектирование пользовательского интерфейса

3.4.1 Разработка прототипов интерфейса

Для разработки прототипа дизайна сайта был использован инструмент для создания графического дизайна Figma. На рисунках 6-11 представлены прототипы страниц будущего веб-приложения.

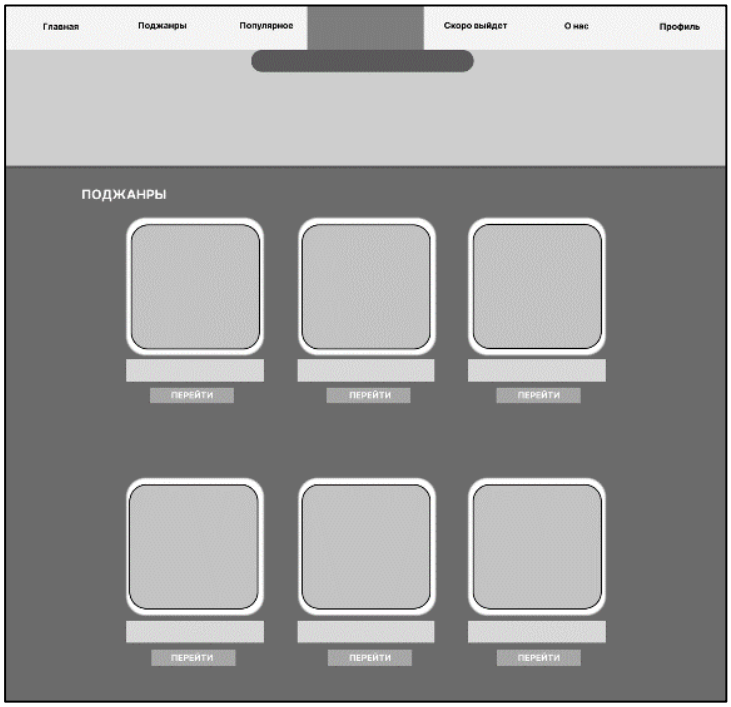

Рисунок 6 – Прототип главной страницы

Прототип главной страницы веб-приложения электронных книг, представленный на рисунке 6, имеет следующую структуру и элементы:

1) Верхняя панель с навигационным меню и поисковым полем.
2) Баннер вдохновляющий пользователей картинкой космоса.
3) Блок со списком поджанров, представленный в виде сетки с превью обложек поджанров.
4) Блоки со списком популярных книг и книг которые скоро выйдут представленный в виде миниатюрной сетки сетки с превью обложек книг.

Этот прототип представляет собой основу для создания удобной и функциональной главной страницы веб-приложения электронных книг, которая поможет пользователям быстро находить нужные книги.

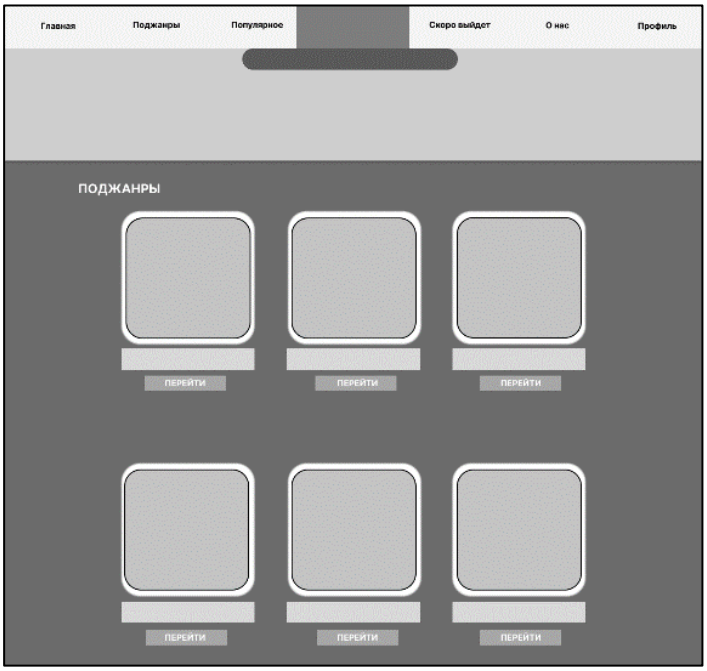

Рисунок 7 – Прототип страницы поджанров

На рисунке 7 показан прототип страницы поджанры для веб-приложения электронных книг. Страница имеет следующую структуру:

1. Верхняя панель с навигационным меню и поисковым полем.
2. Блок со списком поджанров, представленный в виде сетки с превью обложек поджанров.

Общий дизайн прототипа выглядит минималистичным, что должно облегчить навигацию и сделать процесс поиска поджанров максимально простым.

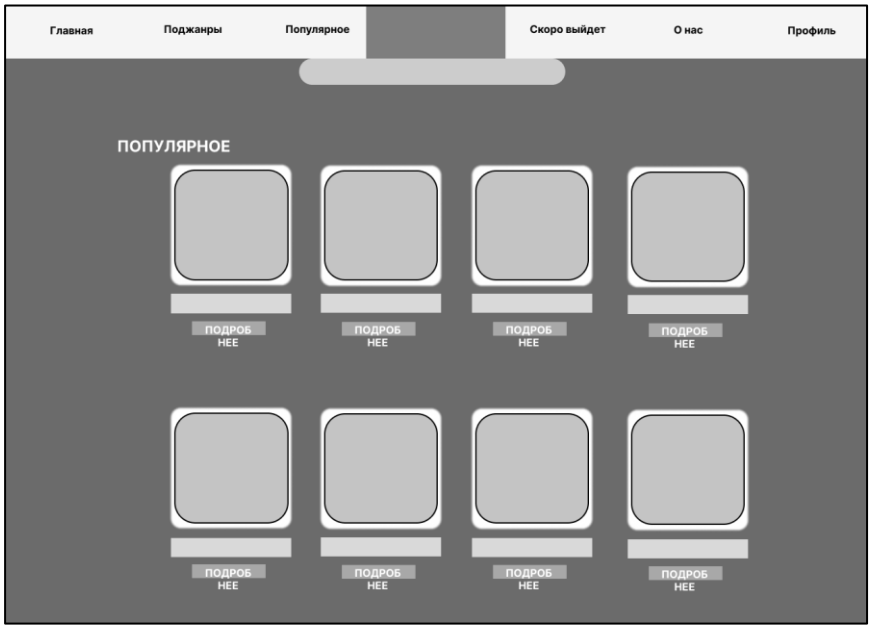

Рисунок 8 – Прототип страницы популярного

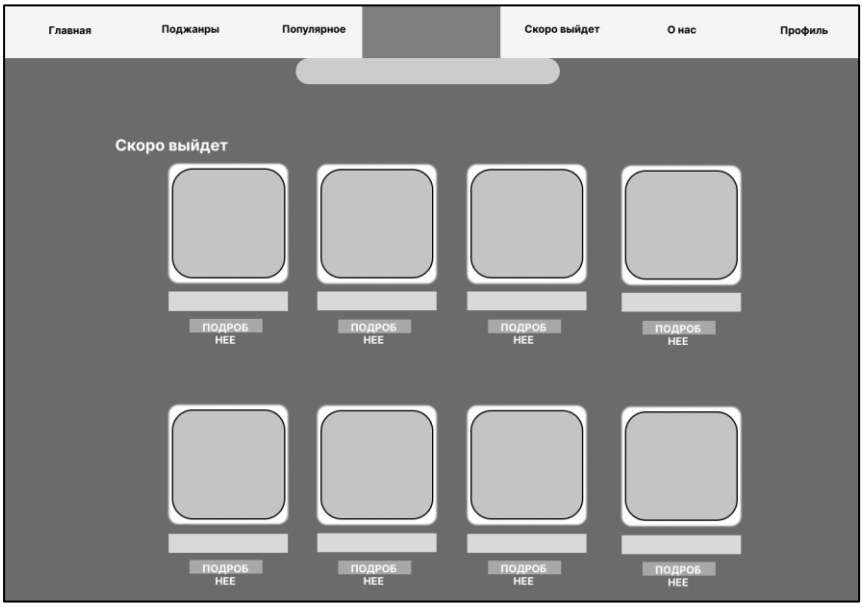

Рисунок 9 – Прототип страницы скоро выйдет

На рисунках 8-9 показаны прототипы страниц популярного и скоро выйдет для веб-приложения электронных книг. Страницы имеет следующую структуру:

1. Верхняя панель с навигационным меню и поисковым полем.
2. Блок со списком книг, представленный в виде сетки с превью обложек книг.

Общий дизайн прототипов один и тот же, выглядит минималистичным, что должно облегчить навигацию в поиске книг.

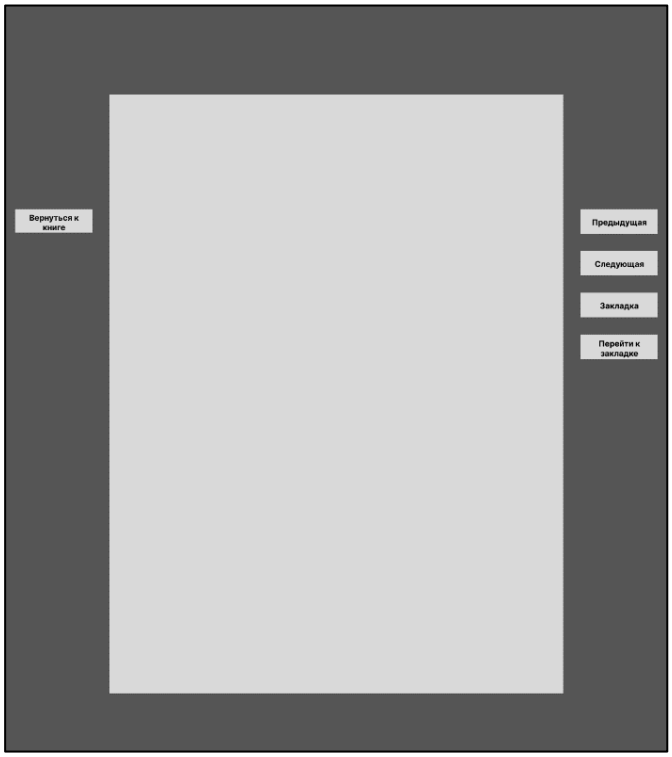

Рисунок 10– Прототип страницы чтения

На рисунке 10 показан прототип страницы чтения для веб-приложения электронных кииг. Страница имеет следующую структуру:

1. По центру блок с которого происходит чтение книги, это ключевой элемент страницы.
2. Слева расположена кнопка возврата к карточке товара.
3. Справа кнопки «предыдущая», «следующая», «закладка», «вернуться к закладке». С помощью этих кнопок будет происходить взаимодействие с книгой.

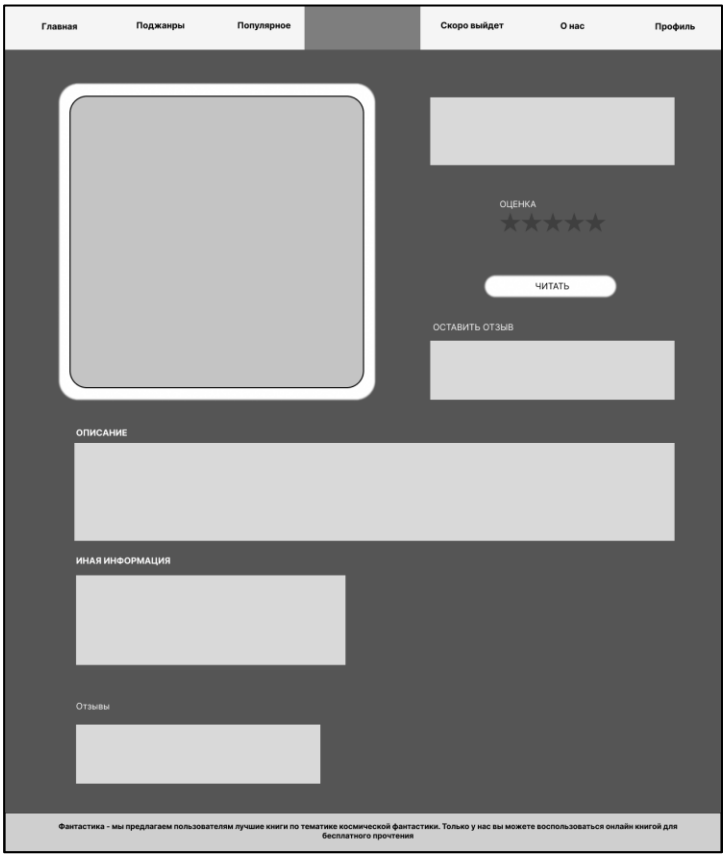

Рисунок 11– Прототип страницы карточки товара

На рисунке 11 показан прототип страницы карточки товара для ивеб-приложения электронных книг. Страница имеет следующую структуру:

1) Большое изображение обложки книги.
2) Справа от изображения обложки расположены название книги и её оценка.
3) Блок для написания отзыва.
4) Снизу от изображения обложки расположено краткое описание книги. Этот элемент должен привлечь читателя краткими захватывающими действиями в книге.
5) Блок с иной информацией, такой как номер в серии, именование автора и т.д.
6) В самом низу расположен блок с оставленными рецензиями и отзывами,
7) здесь читатели смогу обсуждать книгу между собой.

Диаграмма навигации:

Диаграмма навигации приложения состоит из следующих ключевых элементов: Главная, Поджанры, Популярное, Скоро выйдет, Профиль. Для неавторизированных пользователей недоступна ни одна страница.

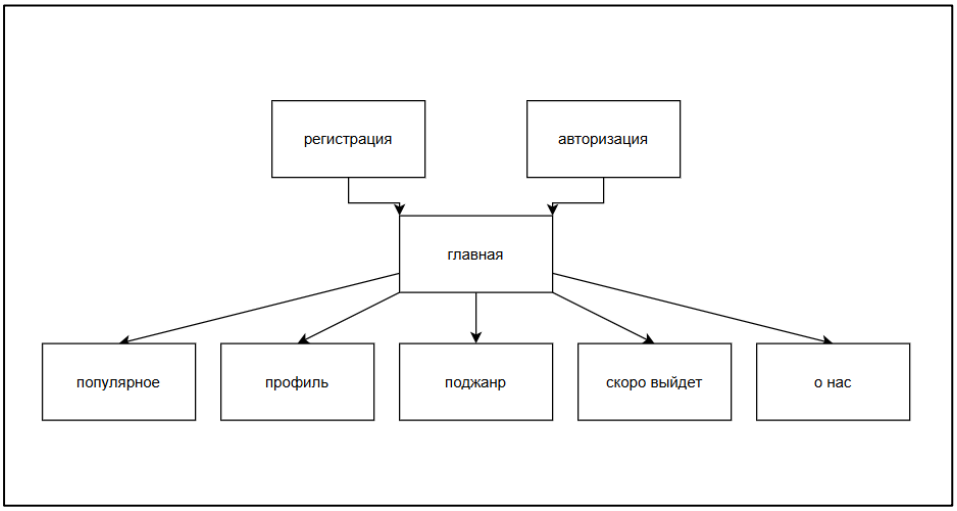

Рисунок 12 – Диаграмма навигации

Вывод по созданному прототипу:

Созданный прототип обеспечивает удовлетворение основных потребностей пользователей и легкость взаимодействия с приложением. Навигация интуитивно понятна, а дизайн отвечает современным требованиям. Прототип представляет собой эффективное средство для дальнейшей разработки и реализации веб-приложения «Фантастика».

3.4.2 Выбор цветовой гаммы и шрифтов

Для обеспечения единого стиля и визуальной привлекательности пользовательского интерфейса приложения «Фантастика» были выбраны определенные цвета и шрифты.

Цветовая гамма:

Основной цвет кнопок: #004C63 (Сапфирный с оттенком изумрудного). Этот цвет используется для выделения ключевых элементов и действий в интерфейсе, обеспечивая контраст и акцент.

Задний фон: #0f2c34 (Полуночный). Этот цвет выбран для фона основных блоков и панелей, создавая тёмный, загадочный общий вид приложения.

Рисунок 13 – Цветовая гамма

Шрифты:

Основной шрифт: Inter. Для основного текста и заголовков был выбран шрифт Inter, это вариативный шрифт, разработанный для компьютерных экранов.

Inter имеет высокую высоту x, чтобы помочь в удобочитаемости текста со смешанным регистром и строчными буквами. Обеспечивает четкость и легкость восприятия текста.

Этот выбор цветовой гаммы и шрифтов позволяет создать стильный и привлекательный интерфейс приложения, который одновременно удобен для использования и соответствует современным требованиям дизайна.

3.4.3 Разработка элементов интерфейса

Для обеспечения удобства использования приложения «Фантастика» были разработаны следующие элементы: «Кнопка», «Строка поиска».

Кнопка представляет собой элемент интерфейса, предназначенный для выполнения определенного действия или перехода на другую страницу. В приложении использована кнопка с фоновым цветом #004C63 и текстом белого цвета, написанным шрифтом Inter. Кнопка имеет скругленные углы и эффект наведения для повышения интерактивности.

Рисунок 14 – Кнопка

Строка поиска предназначена для пользовательского ввода текстовой информации, такой как имя автора, название книги, названия поджанра. В приложении использовано поле ввода с белым фоном. Текст в строке поиска отображается шрифтом Regular.

Рисунок 15 – Строка поиска

3.4.4 Разработка дизайн-макетов

Главная страница:

Главная страница встречает нас баннер с изображением космоса, что добавляет загадочности в стиль сайта. Следом идёт блок с боджанрами, с тематическими изображениями. Блок популярное с книгами, имеющими большее количество прочтений. Блок скоро выйдет с изображениями неопубликованных книг.

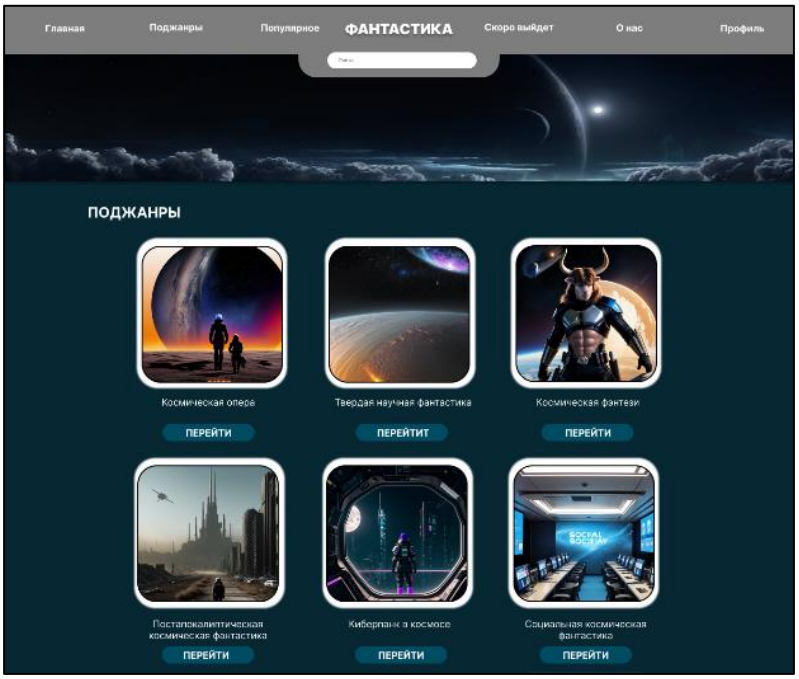

Рисунок 16 – Дизайн главной страницы

Поджанры:

Страница имеет один блок с поджанрами, что облегчает навигацию и поиск.

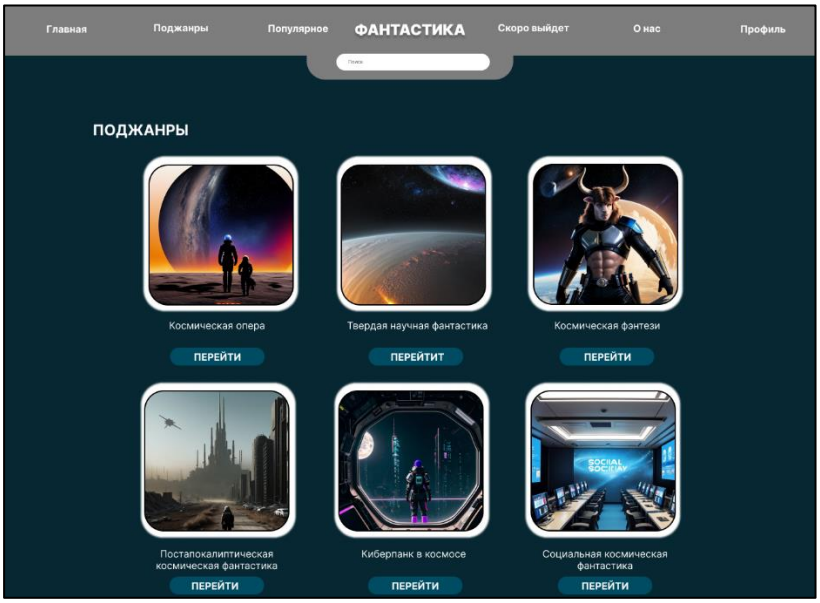

Рисунок 17 – Дизайн страницы поиска

Популярное и скоро выйдет:

Страницы популярного и скоро выйдет (рис. 18-19) имеют один основной дизайн и представляют из себя вывод информации блоков с главной страницы на отдельные.

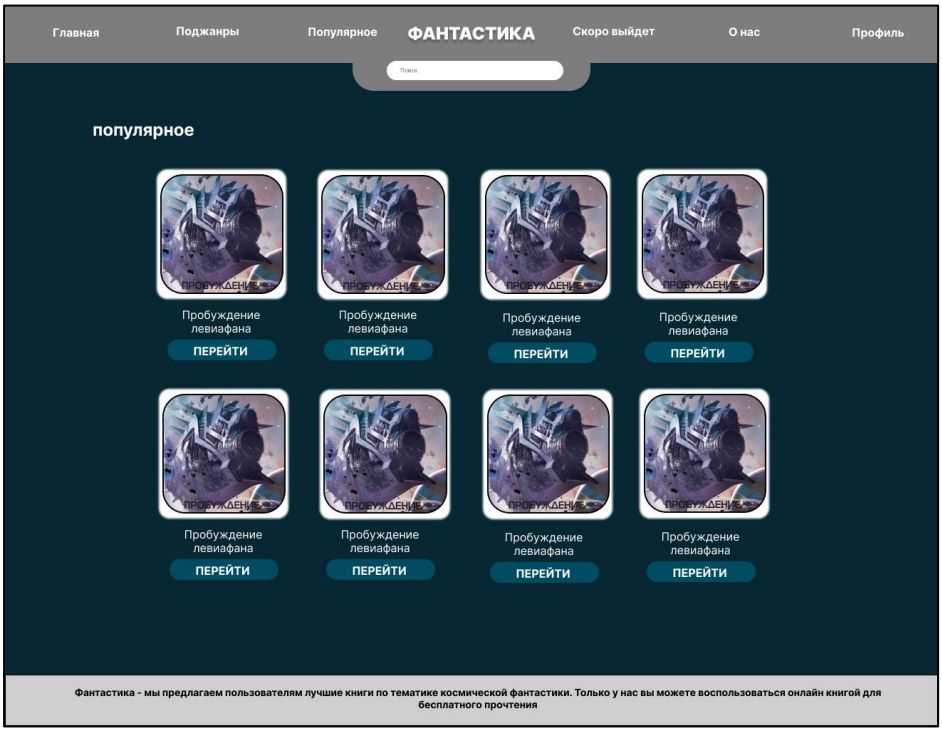

Рисунок 18 – Дизайн популярное

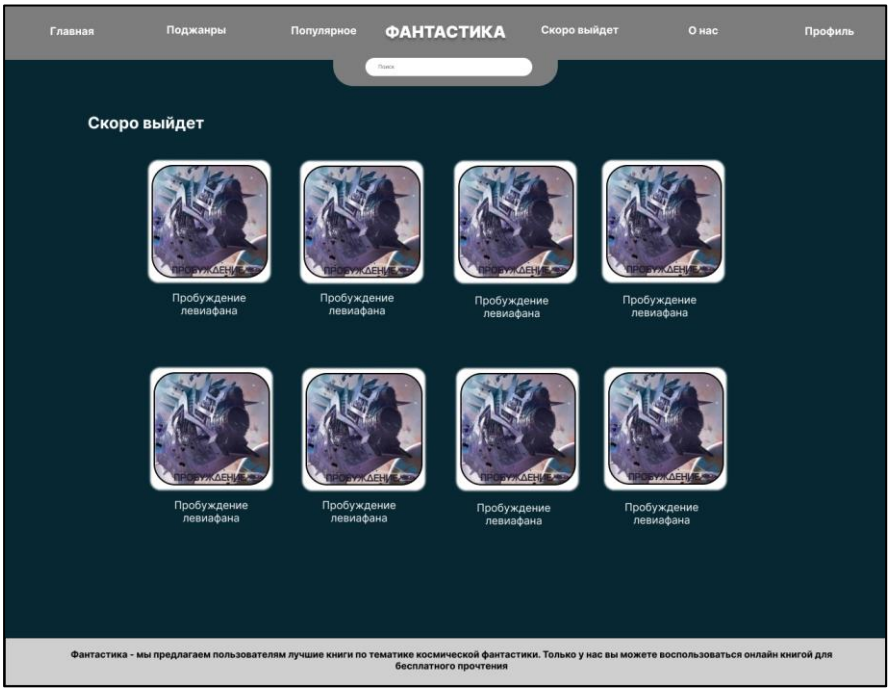

Рисунок 19 – Дизайн скоро выйдет

Карточка товара:

Страница с информацией о книге и её кратким описанием. Большая картинка с превью книги. Сапфирная кнопка с надписью «Читать». Блок с характеристиками. Раздел с оставленными комментариями.

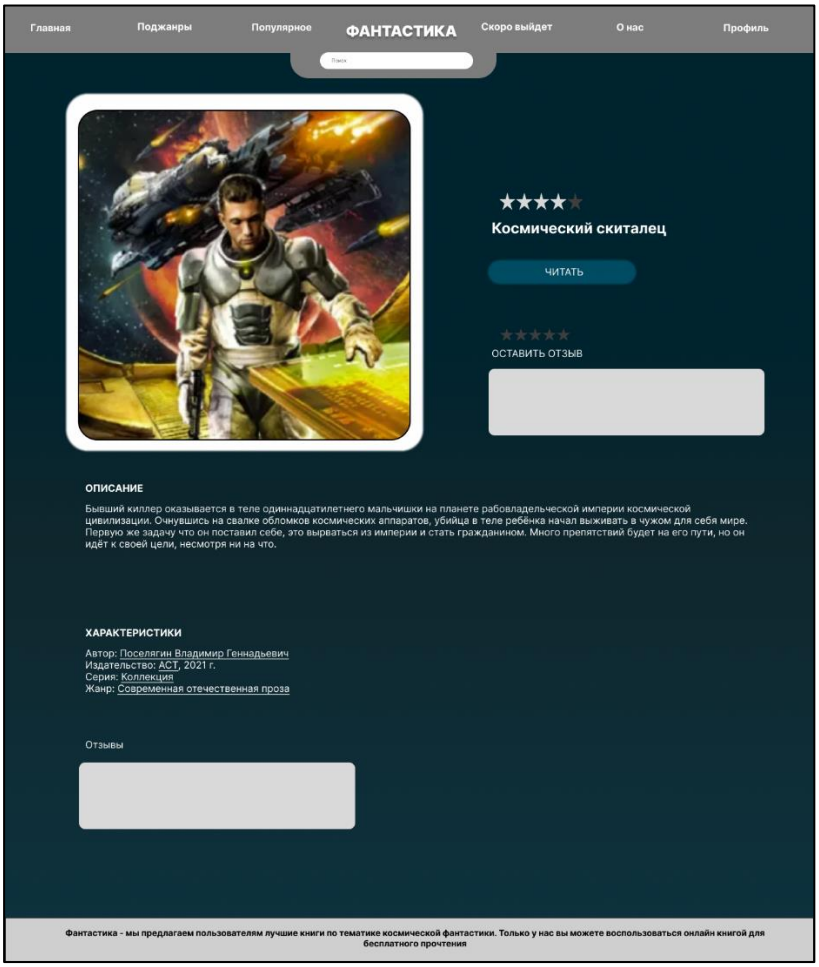

Рисунок 20 – Дизайн страницы карточки товара

Профиль:

Страница профиля имеет простой дизайн, на ней указаны имя профиля, электронная почта и дата создания аккаунта. Имеется кнопка «Выйти из аккаунта».

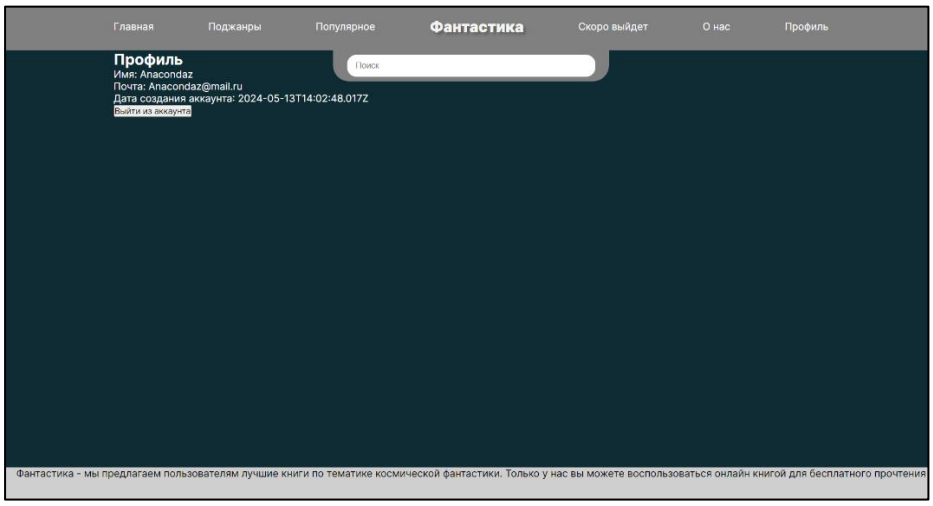

Рисунок 21 – Дизайн страницы профиля

4 Разработка веб-приложение

4.1 Разработка веб-интерфейса

При разработке веб-приложения был использован язык программирования Javascript совместно с NodeJs - серверная платформа для работы с JavaScript. Задействована была библиотека для веб-сервера express.js.

Серверная часть приложения была необходима для работы со статическими файлами (pdf-файлы документы, превью книги) и выдача информации о книги пользователю.

Реализован был Rest API для полноценной интеграцией с клиентской частью на чтение, редактирование, запись и удаление записей из Базы данных. Также был реализован концепт CRUD.

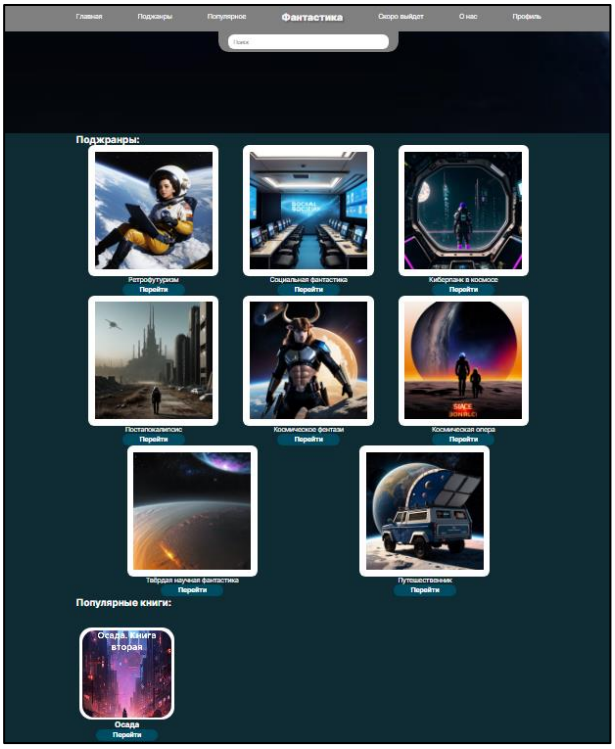

Рисунок 22– Страница “Главная"

Отрывок кода страницы по работе с книгами представлен на рисунках 23-24. С полным кодом страницы можно ознакомиться в приложении Б.

```jsx
import { useEffect, useState } from "react"
import { bookService } from "../../services/book"
import { Preview } from "../genre/Card/component/Preview"
import '../genre/style.css'
import { useNavigate } from "react-router-dom"

export const PopularBooks = () => {

    const navigate = useNavigate()

    const [data, setData] = useState(null)
    const [isLoading, setIsLoading] = useState(true)
    const [, setError] = useState(null)

    useEffect(() => {
        bookService.popularList().then(setData).catch(setError).finally(() => setIsLoading(false))
    }, [])

    return (
        isLoading ? <h2>Загрузка...</h2> :
        <>
            <h1>Популярные книги:</h1>
            <div className="genre-books-container">
                {data.rows.map(item =>
                    <div className="GenreBookCard" key={item.id}>
                        <Preview id={item.id} />
                        <h3>{item.name}</h3>
                        <button className="btn" onClick={() => navigate(`/book/${item.id}`)}>Перейти</button>
                    </div>
```

Рисунок 23 – Код клиентской части поиска книг

```
    }
        return res.status(400).json(err)
    }catch(err){
        return res.json(data)
        const data = await Book.findAndCountAll({ offset, limit, order: [['createdAt', 'DESC']]
        const {limit, offset} = getPagination(req)
    try{
async function getPopularList(req,res){
```

Рисунок 24 – Код серверной части поиска книг

В личном кабинете отображается личная информация о пользователе и красная кнопка для выхода из личного кабинета, а также админ-панель для пользователей со статусом администратора в базе данных. На рисунке 25 представлена страница личного кабинета.

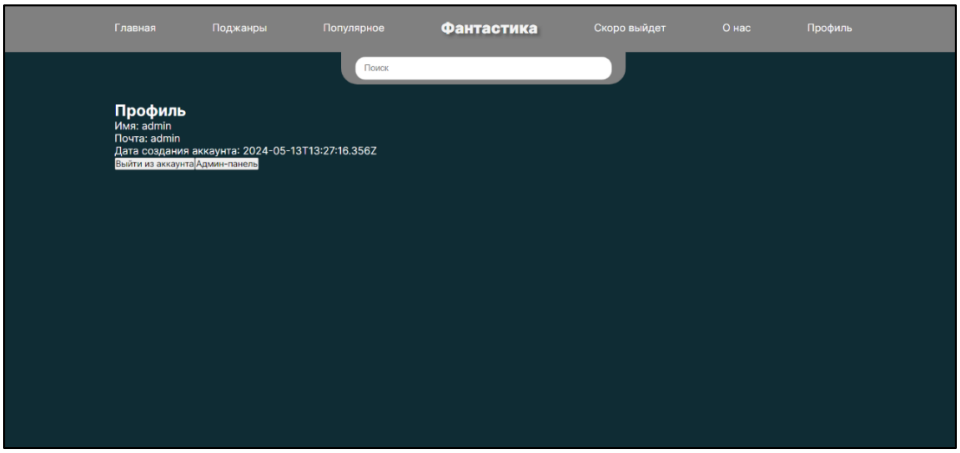

Рисунок 25 – Страница профиля

Отрывок кода страницы профиля представлен на рисунках 26-27.

```
async me(req,res){
    try{
        const {id: userId} = req.user
        const {id, name, email, createdAt, role} = await User.findOne({where: {id: userId}})
        return res.json({id, name, email, createdAt, role})
    }catch(err){
        return res.status(400).json(err)
    }
}
```

Рисунок 26 – Код страницы серверной части личного кабинета

```
export const Profile = () => {

    const navigate = useNavigate()

    const [data, setData] = useState({})
    const [isLoading, setIsLoading] = useState(true)
    const [ , setError] = useState(null)

    useEffect(() => {
        userService.getMe().then(setData).catch(setError).finally(() => setIsLoading(false))
    }, [])

    const {role} = jwtDecode(localStorage.getItem('token'))

    return (
        <Wrapper enableHeader>
            {isLoading ?
                <h2>Загрузка...</h2>
            :
                <>
                    <h2>Профиль</h2>

                    <p>Имя: {data?.name}</p>
                    <p>Почта: {data?.email}</p>
                    <p>Дата создания аккаунта: {data?.createdAt}</p>
                    <button onClick={() => {
                        localStorage.removeItem('token')
                        navigate('/sign-in')
                        window.location.reload()
                    }}>Выйти из аккаунта</button>

                    {role === "ADMIN" && <button onClick={() => navigate('/admin')}>Админ-панель</button>}
                </>
            }

        </Wrapper>
    )
}
```

Рисунок 27 – Код страницы клиентской части личного кабинета

4.2 Разработка базы данных веб-приложение

Разработка базы данных приложения реализовывалась с помощью библиотеки из серверной части приложения Sequelize orm. База данных состоит из 6 таблиц:

Пользователи, Книги, Жанры, Отзывы и Закладки.

Таблицы баз данных представлены на рисунках 28–32.

```
book-reader=# SELECT column_name, data_type from information_schema.columns WHERE table_name = 'Reviews';
 column_name |        data_type
-------------+--------------------------
 stars       | integer
 updatedAt   | timestamp with time zone
 UserId      | integer
 id          | integer
 createdAt   | timestamp with time zone
 name        | character varying
 desc        | character varying
(7 ёЄЁюъ)
```

Рисунок 28 - Таблица «Отзывы»

```
book-reader=# SELECT column_name, data_type from information_schema.columns WHERE table_name =
 column_name |         data_type
-------------+---------------------------
 id          | integer
 page        | integer
 createdAt   | timestamp with time zone
 updatedAt   | timestamp with time zone
 BookId      | integer
 UserId      | integer
(6 ёЄЁюъ)
```

Рисунок 29 - Таблица «Закладки»

```
book-reader=# SELECT column_name, data_type from information_schema.columns WHERE table_name =
   column_name    |         data_type
------------------+---------------------------
 id               | integer
 readCount        | integer
 isPublished      | boolean
 createdAt        | timestamp with time zone
 updatedAt        | timestamp with time zone
 GenreId          | integer
 publishingHouse  | character varying
 name             | character varying
 desc             | text
 pdfPath          | character varying
 series           | character varying
 author           | character varying
(12 ёЄЁюъ)
```

Рисунок 30 - Таблица «Книги»

```
book-reader=# SELECT column_name, data_type from information_schema.columns WHERE table_name =
 column_name |         data_type
-------------+---------------------------
 id          | integer
 createdAt   | timestamp with time zone
 updatedAt   | timestamp with time zone
 name        | character varying
 desc        | character varying
(5 ёЄЁюъ)
```

Рисунок 31 - Таблица «Жанры»

```
book-reader=# SELECT column_name, data_type from information_schema.columns WHERE table_name =
 column_name |         data_type
-------------+---------------------------
 id          | integer
 createdAt   | timestamp with time zone
 updatedAt   | timestamp with time zone
 role        | character varying
 password    | character varying
 name        | character varying
 email       | character varying
(7 ёєЁюъ)
```

Рисунок 32 – Таблица «Пользователи»

4.3 Разработка веб-приложение

Для подключения веб-приложения к базе данных в sequelize, необходимо настроить конфигурационный файл .env, который расположен в корне проекта. В этом файле указываются данные для подключения к базе данных, такие как имя базы данных, логин, пароль и хост.

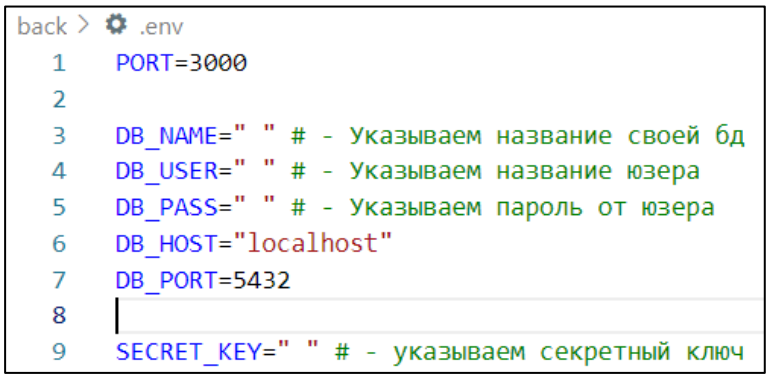

Рисунок 33 – Подключение к базе данных

Ниже предоставлен код моделей Базы данных

Рисунок 34 - Модели базы данных

5 Документирование программного продукта

5.1 Руководство пользователя веб приложения

Чтобы запустить проект необходимо открыть терминал и написать «npm install» после того, как все пакеты будут установлены ввести команду «npm run watch». Переходим в другой терминал и вводим «composer update», после этого вводим «php artisan serve». В консоли появится ссылка: «http://127.0.0.1:8000». Кликаем по ней и переходим в веб-приложение на главную страницу.

После установки nodejs и postgres, необходимо скачать репозиторий с ветками проекта:

Открываем терминал и пишем:

« git clone https://github.com/AnacondazNS/kurs.git », Далее переходим в папку front командой «cd front» , устанавливаем nodejs-модули «npm i» и запускаем проект «npm run dev». В консоли появится ссылка: «http://127.0.0.1:8000». Кликаем по ней и переходим в веб-приложение на страницу регистрации.

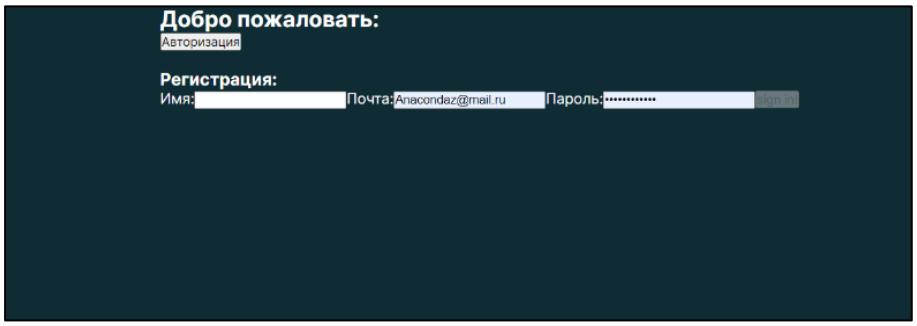

Рисунок 35 - Страница регистрации

Чтобы перейти на страницу «Главная» необходимо авторизироваться. Страница авторизации представлена на рисунке 25. Здесь нам предлагают ввести логин или Email и пароль.

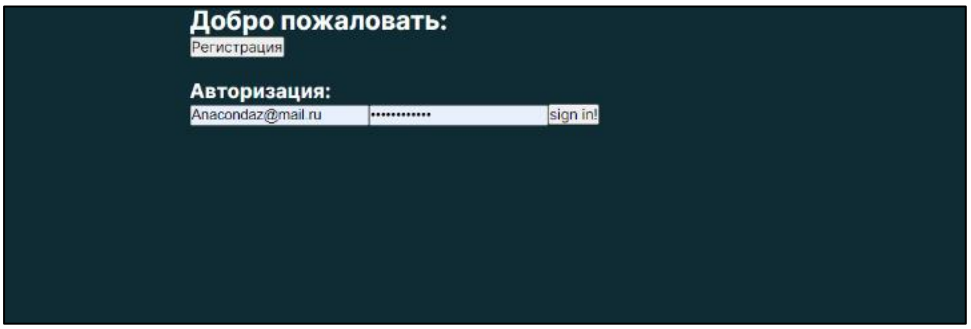

Рисунок 35 - Страница авторизации

Далее нас автоматически перебрасывает на главную страницу. Главная страница представлена на рисунке 26.

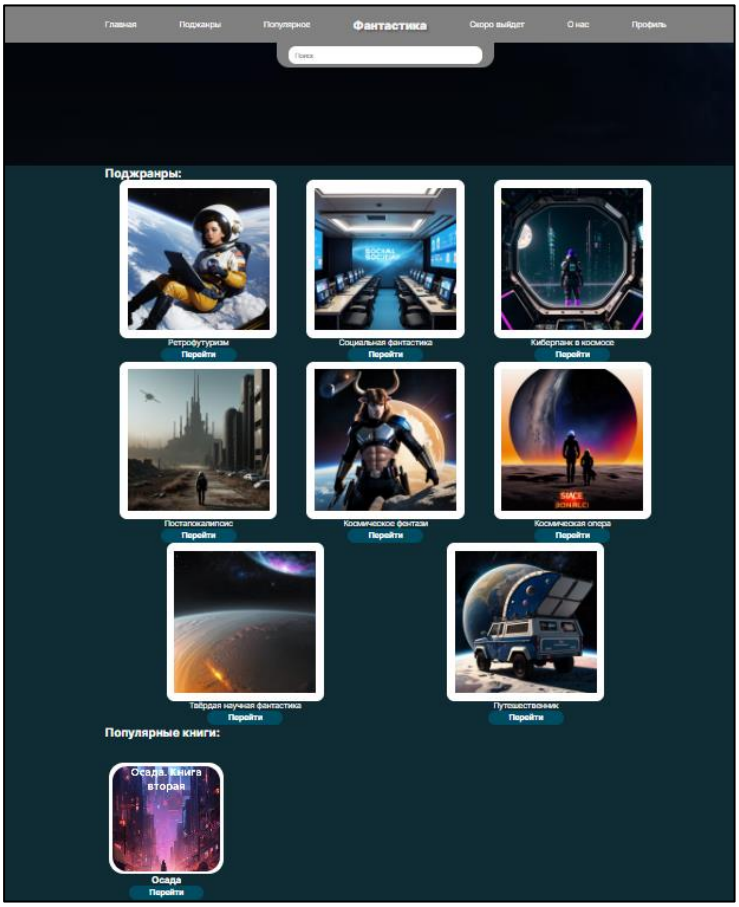

Рисунок 37 - Главная страница

На главной странице нам предоставляется возможность выбрать поджанр который мы хотим прочитать, также указываются популярные книги. Открываем поджанр и запускаем понравившуюся книгу - рисунок 28.

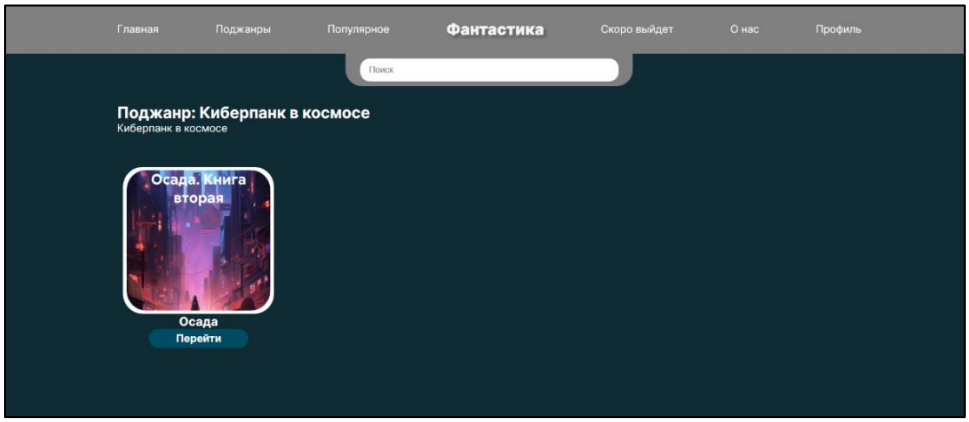

Рисунок 38 - Страница поджанра

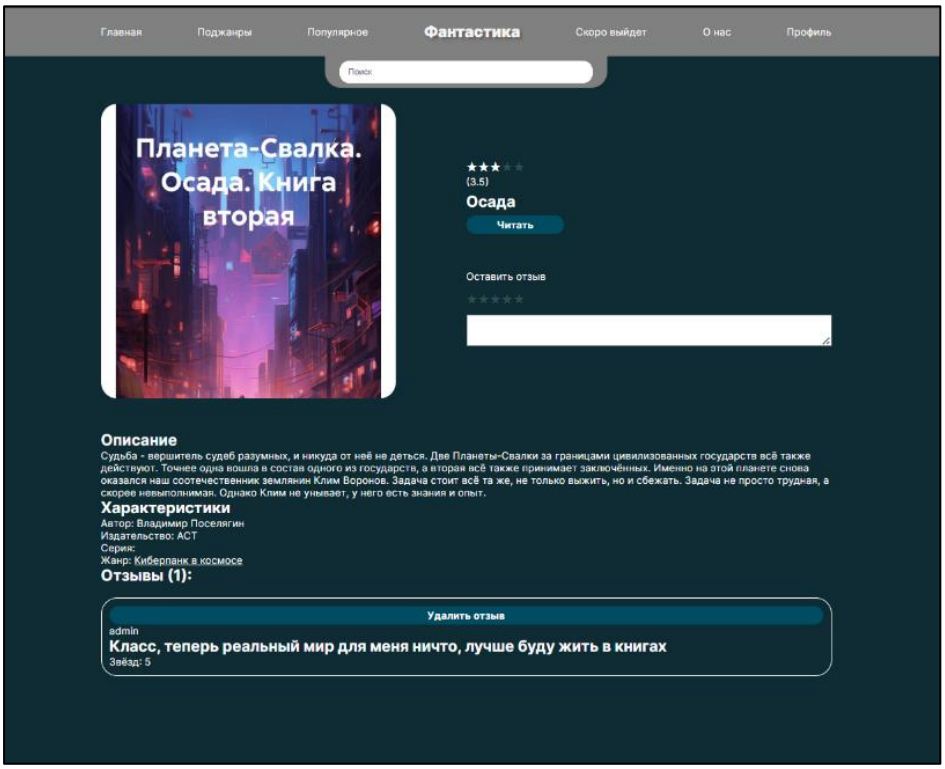

Рисунок 39 - Карточка товара

В открывшиеся книге оставляем отзыв и рецензию. Переходим по кнопке «Читать», где пользователя перенаправляет на страницу для прочтения. Страница для прочтения представлена на рисунке 29

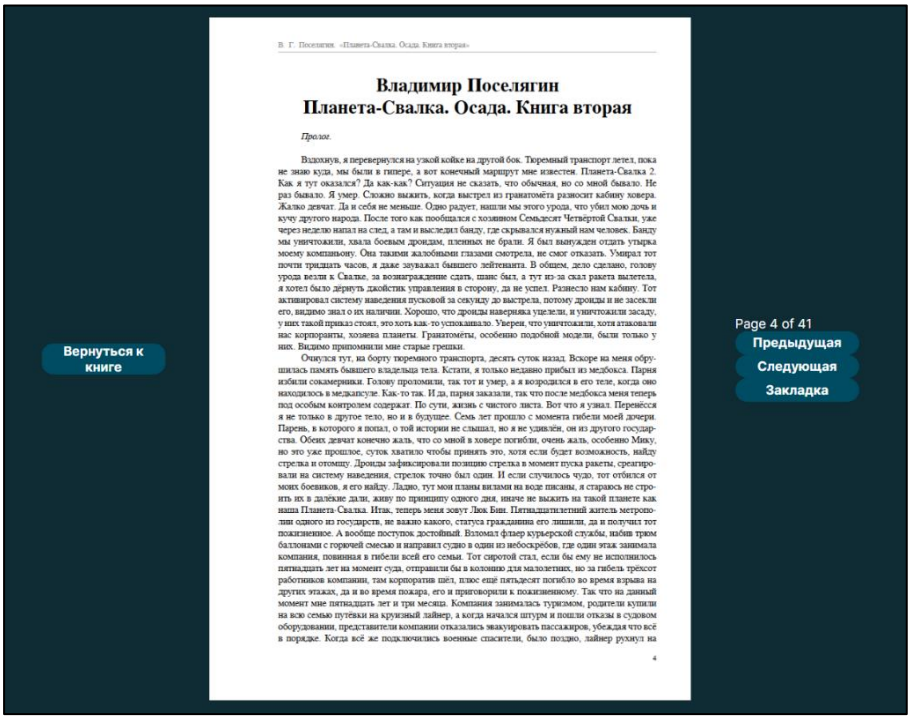

Рисунок 40 - Страница для чтения

Если пользователь авторизовался как администратор, то пользователь может удалять отзывы переходя на страницу книги. Пример удаления отзыва представлен на рисунке 30.

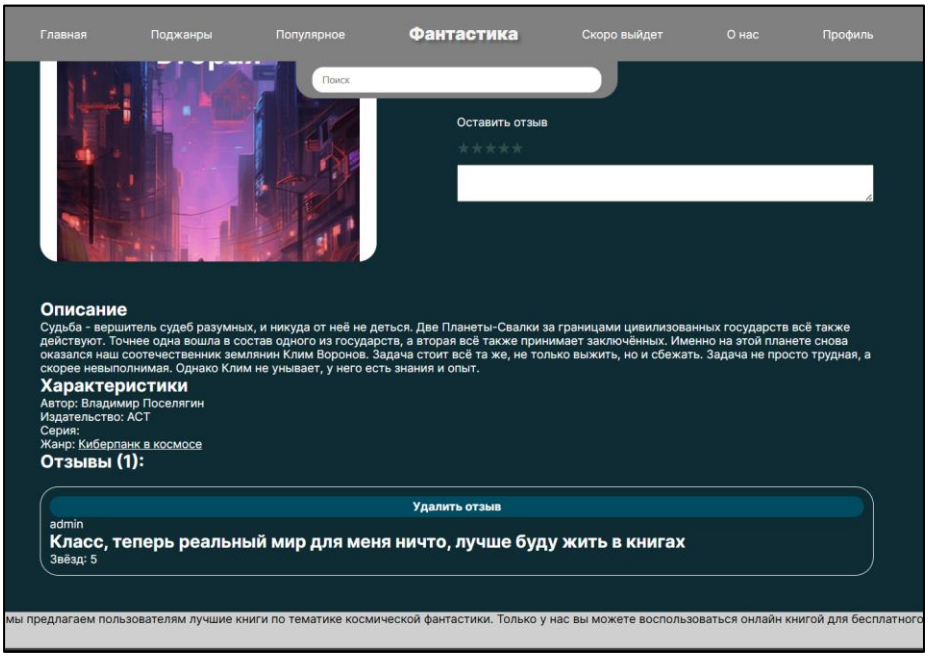

Рисунок 41 - удаление отзыва

Также у пользователя авторизированным как администратор в профиле добавляется административная панель, с которой появляется возможность добавлять и редактировать книги, а также добавлять и удалять список поджанров. Пример с использованием административной панели представлена на рисунках 31-34

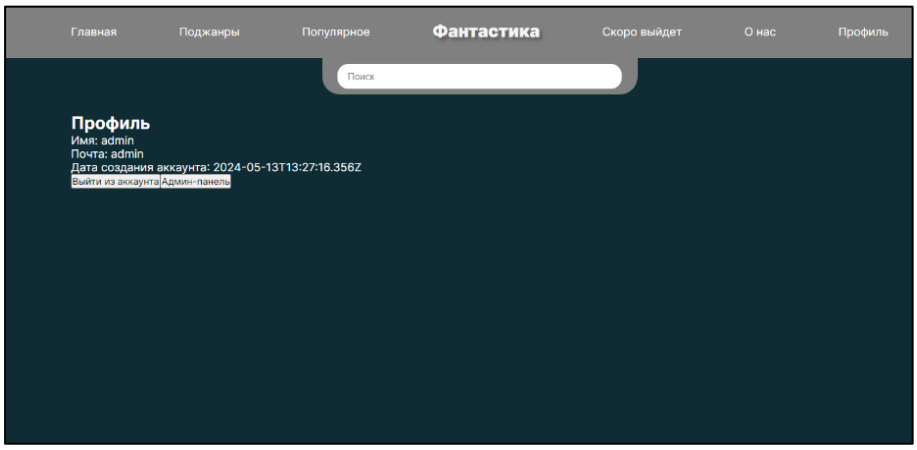

Рисунок 42 - Добавлена кнопка для административной панели

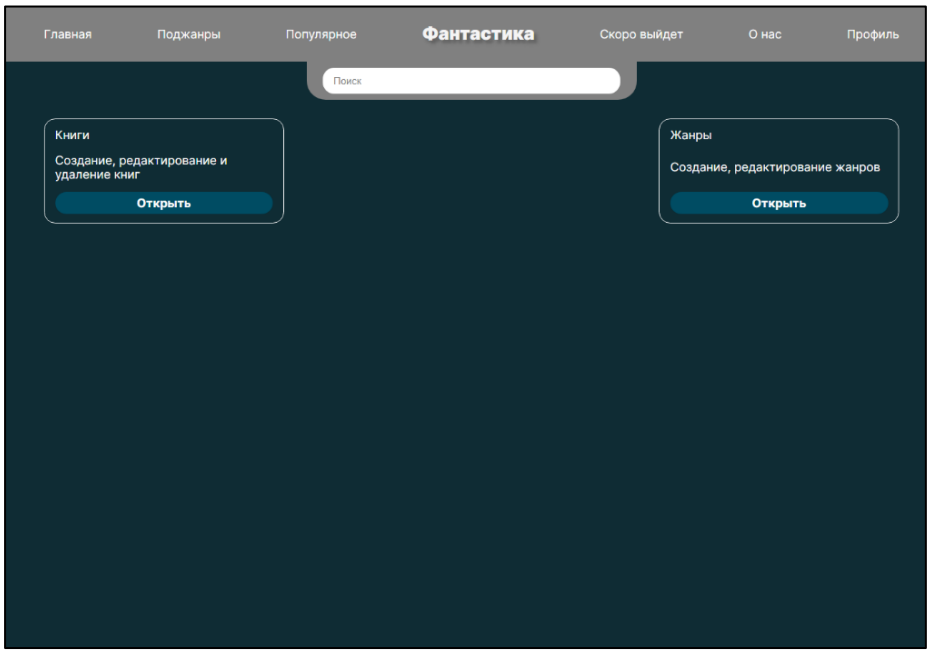

Рисунок 43 - Выбор категории редактирования

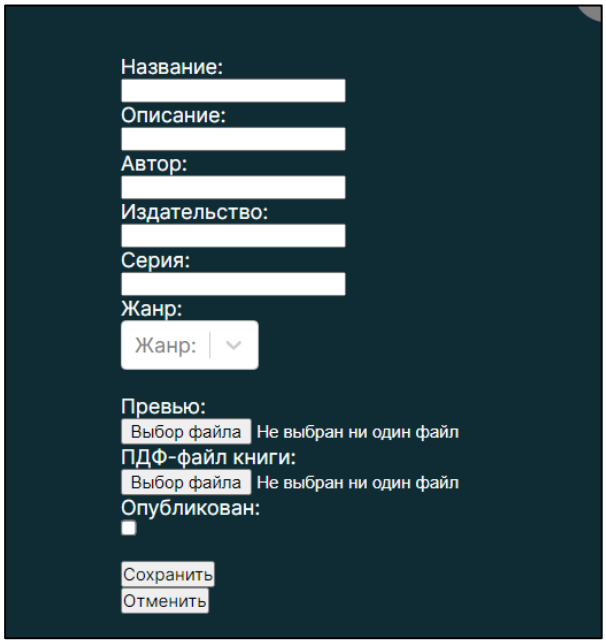

Рисунок 44 - Создание книги

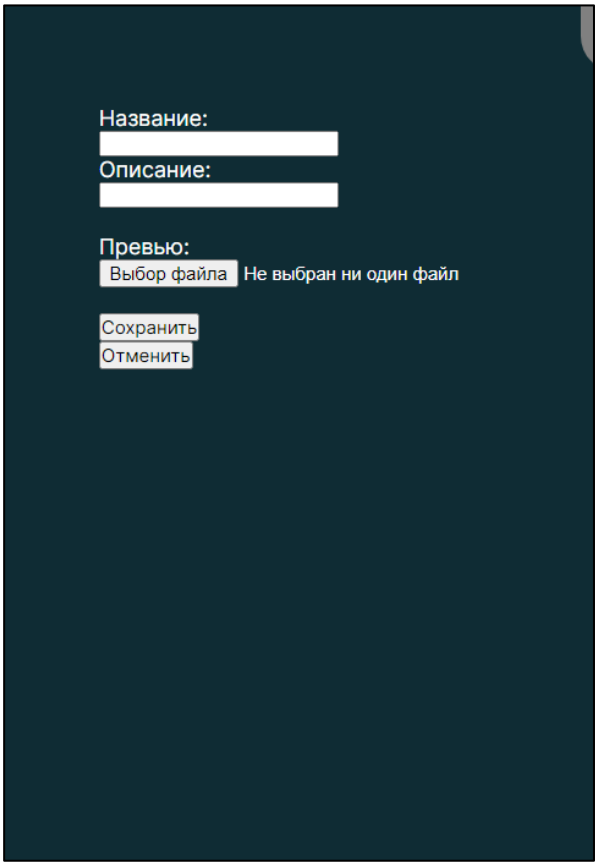

Рисунок 45 - создание жанра

ЗАКЛЮЧЕНИЕ

В процессе курсового проекта была поставлена цель разработать веб-приложение электронных книг «Фантастика». В результате работы был создан дизайн пользовательского интерфейса и back-end часть приложения, обеспечивающее возможность бесплатного прочтения книг.

Из языков программирования основным для разработки был выбран JavaScript, с использованием библиотеки React.js. Его применение в моем проекте было очень полезным и удобным. React.js предоставляет множество полезных функций и инструментов, которые значительно упростили разработку моего веб-приложения. Эта библиотека значительно упрощает разработку и тестирование веб-приложения. Общение с сервером реализовано с помощью Restfull API, для работы с данными из базы данных и статических файлов. Использование Node.js упростило написание кода для чтения системой PDF файлов и работой с ними, с помощью готовых .lib - файлов. Все необходимые диаграммы и модели помогли мне разработать нужный интерфейс. Для того, чтобы поставленная цель была решена, были выполнены следующие задачи:

- Предпроектное исследование.
- Написание технического задания.
- Проектирование интернет-магазина.
- Разработка интернет-магазина.
- Документирование интернет-магазина.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. https://ru.react.js.org/docs/react-component.html (дата обращения: 17.03.2024). – Текст: электронный.
2. https://nodejsdev.ru/guides/webdraftt/static/?ysclid=lwwhcxl1sz386859085 (дата обращения: 17.03.2024). – Текст: электронный.
3. https://getbootstrap.ru/docs/5.1/layout/containers/ (дата обращения: 23.03.2024). – Текст: электронный.
4. https://habr.com/ru/articles/522078/ (дата обращения: 28.03.2024). – Текст: электронный.
5. https://git-scm.com/downloads (дата обращения: 13.04.2024). – Текст: электронный.
6. https://nodejs.org/en (дата обращения: 18.03.2024). – Текст: электронный.
7. https://www.postgresql.org/download/ (дата обращения: 18.03.2024). – Текст: электронный.
8. https://node-postgres.com/features/connecting (дата обращения: 20.03.2024). – Текст: электронный.
9. https://habr.com/ru/articles/565062/ (дата обращения: 20.03.2024). – Текст: электронный.
10. https://habr.com/ru/articles/483202/ (дата обращения: 20.03.2024). – Текст: электронный.

Приложение А Техническое задание

Министерство образования Иркутской области

Государственное бюджетное профессиональное образовательное учреждение

Иркутской области

«Иркутский авиационный техникум»

(ГБПОУИО «ИАТ»)

ТЕХНИЧЕСКОЕ ЗАДАНИЕ

ВЕБ-ПРИЛОЖЕНИЕ ЭЛЕКТРОННЫХ КНИГ «ФАНТАСТИКА»

Руководитель: _____ (Н.Р. Карпова)

(подпись, дата)

Студент: _____ (А.С.Нестерук)

(подпись, дата)

Иркутск 2024

1 Общие сведения

Наименование работы: веб-приложение электронных книг «Фантастика».

Исполнитель: студент Иркутского авиационного техникума, отделения ИКТ, группы ВЕБ-21-1, Нестерук А.С.

Разработка веб-приложения проходит в рамках курсового проекта по МДК.09.01 «Проектирование и разработка веб-приложений», на основании приказа №284-у от 29 января 2024 года.

Сроки разработки веб-приложения с 29.01.2024 по 13.05.2024 года.

2 Назначение и цели создания веб-приложения

Назначение создания веб-приложения для чтения книг заключается в чтении книг онлайн. Цель создания веб-приложения обеспечение удобного выбора и чтения книг через интернет.

3 Требования к веб-приложению в целом

3.1 Требования к структуре и функционированию веб-приложения

1. Раздел «Личный кабинет»:

1.1 Отображение информации о пользователе, включая имя, электронную почту и историю просмотров.

1.2 Возможность редактирования личных данных и настроек учетной записи.

1.3 Возможность быстрого перехода в другие разделы сайта.

2. Раздел «Библиотека»:

2.1 Отображение списка доступных книг жанра космическая фантастика с возможностью фильтрации и сортировки по различным параметрам (автор, название, и т.д.).

3. Раздел «Чтение»:

3.1 Возможность чтения книг онлайн с адаптивным дизайном

3.2 Возможность добавления закладок и заметок в процессе чтения.

4. Раздел «Комментарии»:

3.2 Возможность оставлять комментарии к книгам и участвовать в дискуссиях с другими пользователями.

3.2 Требования к надежности

Для обеспечения надежности необходимо проверять корректность получаемых данных и реализовать валидность полей. Входные данные поступают в виде значений с клавиатуры. Эти значения отображаются в отдельных полях таблицы.

3.3 Требования к безопасности

Для обеспечения безопасности в веб-приложении, необходимо реализовать разграничение прав доступа.

3.3 Требования к эксплуатации, техническому обслуживанию, ремонту и хранению компонентов системы

Минимальные системные требования для рабочей станции:

1) Процессор: Intel Pentium 4 2.0Ghz / AMD XP 2200+;
2) Оперативная память: 16 гб;
3) Жёсткий диск: 256 гб;
4) Операционная система: Windows, Linux, MacOS.

4 Требования к документированию

Основным документам, регламентирующими использование веб-приложения является руководство пользователя.

Основным документам, регламентирующими разработку веб-приложения является техническое задание.

5 Состав и содержание работ по созданию веб-приложения

В таблице 1 представлены плановые сроки начала и окончания работы по созданию веб-приложения

Таблица 1 – Плановые сроки по созданию веб-приложения

№ п/п	Содержание этапа или стадии выполнения КП	Сроки выполнения	
		Начало	Окончание
1	Предпроектное исследование предметной области (выбор темы, постановка цели, задач, описание области применения, исследование предметной области)	30.01.24	09.02.24
2	Разработка технического задания (выбор архитектуры программного обеспечения, выбор типа пользовательского интерфейса, выбор языка и среды программирования)	09.02.24	19.02.24
3	Проектирование программного обеспечения. (разработка структурной и функциональной схемы ПО, проектирование базы данных)	19.02.24	04.03.24

4	Оформление пунктов пояснительной записки (введение, предпроектное исследование, техническое задание и проектирование веб-приложения)	04.03.24	11.03.24
5	Разработка (программирование) и отладка программного продукта	11.03.24	29.04.24
6	Составление программной документации (оформление ПЗ, руководство пользователя и презентации)	29.04.24	13.05.24

ПРИЛОЖЕНИЕ Б ЛИСТИНГ КОДА ПОИСКА КНИГ

```
    //Предоставляем путь к статическим файлам веб-приложения
    const PATH_STATIC = path.join(__dirname, '..', '..', 'static')

    module.exports = new class BookController{
        async list(req,res){
            try{
                const { role} = req.user

                const {ignorePublishStatus} = req.query

                const {limit, offset} = getPagination(req)

                const where = {}
                where.isPublished = true

                if(role && role === "ADMIN" && ignorePublishStatus){
                    delete where.isPublished
                }

    //создаём запрос на список книг и их колличество
                const data = await Book.findAndCountAll({offset, limit, where})
                return res.json(data)
            }catch(err){
                return res.status(400).json(err)
            }
        }

        async unpublishList(req,res){
            try{
                const {limit, offset} = getPagination(req)

                const data = await Book.findAndCountAll({offset, limit, where:
{isPublished: false}})
                return res.json(data)
            }catch(err){
                return res.status(400).json(err)
            }
        }
    //Извлекаем данные из базы данных и создаём обьект обновления который замещает
устаревшие данные
        async update(req,res){
            try{
                const {id} = req.params
                const {name, desc, author, publishingHouse, series, genreId,
isPublish} = req.body
```

```
                const updObj = {}
                if(name){
                    updObj.name = name
                }
                if(desc){
                    updObj.desc = desc
                }
                if(author){
                    updObj.author = author
                }
                if(publishingHouse){
                    updObj.publishingHouse = publishingHouse
                }
                if(series){
                    updObj.series = series
                }
                if(genreId){
                    updObj.GenreId = genreId
                }
                if(isPublish && (Boolean(isPublish) === false || Boolean(isPublish)
=== true)){
                    updObj.isPublished = isPublish
                }

                const obj = await Book.findOne({where: {id}})
                if(!obj){
                    return res.status(404).json({message: "NOT_FOUND"})
                }

                const upd = await Book.update(updObj,{where: {id}})

                if(req.files){
                    const {preview, pdf} = req.files

                    if(pdf){
                        try{
                            if(pdf && pdf.mimetype === "application/pdf"){
                                fs.unlinkSync(path.join(PATH_STATIC,         'books',
`${id}.pdf`))
                                pdf.mv(path.join(PATH_STATIC, 'books', `${id}.pdf`))
                            }
                        }catch(err){
                            console.log(err)
                        }
                    }

                    if(preview){
                        try{
```

```
                            if(preview && (preview.mimetype === "image/jpeg") ||
preview.mimetype === "image/jpg"){
                                fs.unlinkSync(path.join(PATH_STATIC, 'previews',
`${id}.jpg`))
                                preview.mv(path.join(PATH_STATIC, 'previews',
`${id}.jpg`))
                            }
                        }catch(err){
                            console.log(err)
                        }
                    }
                }

                return res.json({message: "UPDATED"})
            }catch(err){
                console.error(err)
                return res.status(400).json(err)
            }
        }

        async getPDFByBookId(req,res){
            try{
                const {id} = req.params
                const userId = req.user.id
                // определяем путь к пдф файлу
                const pathName = path.join(__dirname, '..', '..', 'static', 'books',
`${id}.pdf`)
                // читает файл пдф
                const file = fs.readFileSync(pathName,'utf-8')
                if(!file){
                    return res.status(404).json({message: "NOT_FOUND"})
                }
                // счётчик колличества прочтений
                const book = await Book.findOne({where: {id}})
                book.readCount++
                await book.save()

                return res.sendFile(pathName)
            }catch(err){
                return res.status(400).json(err)
            }
        }
    //создание книги и проверка на все заполненные поля
        async createBook(req,res,next){
            try{
                const {name, desc, author, publishingHouse, series, genreId,
isPublish} = req.body

                const {preview, pdf} = req.files
```

```
                if(!name || !desc || !author || !publishingHouse || !series ||
!genreId){
                    throw new Error({message: "CHOOSE_FIELDS"})
                }
                const result = await Book.create({name, desc, author, publishingHouse,
series, GenreId: genreId, isPublished: isPublish})
                //перемещаем пдф в статику
                try{
                    if(pdf && pdf.mimetype === "application/pdf"){
                        pdf.mv(path.join(PATH_STATIC, 'books', `${result.id}.pdf`))
                    }
                }catch(err){
                    console.log(err)
                }
                //перемещаем превью в статику
                try{
                    if(preview    &&    (preview.mimetype    ===    "image/jpeg")    ||
preview.mimetype === "image/jpg"){
                        preview.mv(path.join(PATH_STATIC,                'previews',
`${result.id}.jpg`))
                    }
                }catch(err){
                    console.log(err)
                }
                return res.json(result)
            }catch(err){
                console.log(err)
                return res.status(400).json(err)
            }
        }
    // удаление книги, после идёт процесс удаления пдф и превью
        async deleteById(req,res){
            try{
                const {id} =  req.params

                await Book.destroy({where: {id}})

                try{
                    fs.unlinkSync(path.join(PATH_STATIC, 'books', `${id}.pdf`))
                    fs.unlinkSync(path.join(PATH_STATIC, 'previews', `${id}.jpg`))
                }catch(err){
                    console.error(err)
                }

                return res.json({message: "DELETED"})

            }catch(err){
                console.log(err)
```

```
                return res.status(400).json(err)
            }
        }

        async setBookMark(req,res) {
            try{
                const {id} = req.params
                const {page} = req.body
                const userId = req.user.id


                let bookmark = await Bookmark.findOne({where: {BookId: id, UserId:
userId}})
                if(bookmark){
                    if(bookmark.page === page){
                        await bookmark.destroy()
                        return res.json({message: "REMOVED"})
                    }

                    bookmark.page = page
                    await bookmark.save()

                    return res.json({message: "UPDATED"})
                }

                bookmark = await Bookmark.create({page, BookId: id, UserId: userId})

                return res.json({message: "SETTED"})
            }catch(err){

                return res.status(400).json(err)
            }
        }

        async getBookMark(req,res){
            try{
                const {id} = req.params
                const userId = req.user.id

                const bookmark = await Bookmark.findOne({where: {BookId: id, UserId:
userId}})
                if(!bookmark){
                    return res.status(404).json({message: "NOT_FOUND"})
                }

                return res.json(bookmark)
            }catch(err){
```

```
                return res.status(400).json(err)
            }
        }

        async getByGenreId(req,res){
            try{

                const {id} = req.params
                const {limit, offset} = getPagination(req)

                const data = await Book.findAndCountAll({offset, limit, where:
{GenreId: id, isPublished: true}})
                return res.json(data)
            }catch(err){
                return res.status(400).json(err)
            }
        }

        async getById(req,res){
            try{
                const {id} = req.params

                const data = await Book.findOne({
                    where: {id},
                    include: [
                        {
                            model: Genre,
                            attributes: ['id', 'name', 'desc']
                        }
                    ]
                })
                if(!data){
                    return res.status(404).json({message: "NOT_FOUND"})
                }

                if(!data.dataValues.isPublished){
                    return res.status(404).json({message: "NOT_FOUND"})
                }

                const reviews = await Review.findAll({
                    include: [{
                        model: Book,
                        where: { id: id },
                    }]
```