

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

Профиль подготовки: «Программное обеспечение средств вычислительной техники и автоматизированных систем»

БАКАЛАВРСКАЯ РАБОТА

на тему

**WEB-ПРИЛОЖЕНИЕ «КНИЖНЫЙ
ИНТЕРНЕТ-МАГАЗИН»**

Студент 5 30.05.25 Лукина Е.Ю.
(подпись, дата)

Руководитель И 30.05.25 Акифьев И.В.
(подпись, дата)

Нормоконтролёр А 30.05.25 А.С. Бычков
(подпись, дата)

Работа допущена к защите (протокол заседания кафедры от

28.05.25 г. № 16)

Заведующий кафедрой М М.А. Митрохин
(подпись)

Работа защищена с отметкой хорошо (протокол заседания ГЭК
от 16 июня 2025 г. № 40)

Секретарь ГЭК Е Е.И. Гурин
(подпись)

Пенза 2025

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
М.А. Митрохин
«06» 05 2025г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Лукина Екатерина Юрьевна гр.21ВВП2 факультета вычислительной техники.

2. Тема работы: Web-приложение «Книжный интернет-магазин»

Тема утверждена приказом ПГУ №601/о от "06" мая 2025г.

3. Руководитель бакалаврской работы к.э.н., доцент, каф. ВТ ПГУ Акифьев И.В.

4. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать web-приложение «Книжный интернет-магазин», позволяющее пользователям просматривать каталог книг, осуществлять поиск и фильтрацию, добавлять товары в корзину и оформлять заказы. Приложение должно обладать графическим и интуитивно понятным интерфейсом, для взаимодействия с пользователем. Система должна предоставлять функциональные возможности для авторизации, регистрации, управления пользовательским профилем, а также включать поддержку различных ролей (пользователь, автор, администратор). Разрабатываемое приложение также должно обеспечивать возможность авторам публиковать собственные книги, а администраторам — модерировать контент и обрабатывать заявки.

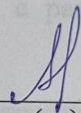
5. Перечень подлежащих разработке вопросов

Требуется произвести анализ предметных областей: изучить существующие web-приложения для продажи книг, выделить их достоинства и недостатки. Выбрать язык программирования, который подойдет для создания графического интерфейса. Определить архитектуру приложения и вспомогательные средства разработки.

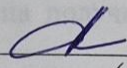
6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Изучение предметной области	32	06.05.2025 – 28.05.2025	
2	Анализ требований	32	07.05.2025 – 30.05.2025	
3	Создание макета графического интерфейса	30	09.05.2025 – 11.05.2025	
4	Выбор технологий, применяемых при проектировании и разработке	40	11.05.2025 – 13.05.2025	
5	Реализация графического интерфейса	66	14.05.2025 – 18.05.2025	
6	Разработка основной части программы	70	16.05.2025 – 23.05.2025	
7	Оформление пояснительной записки	54	24.05.2025 – 30.05.2025	

Дата выдачи задания "06" мая 2025 г.

Руководитель бакалаврской работы  06.05.25 Акифьев И.В.
(подпись, дата)

Задание к исполнению принял студент  06.05.25 Лукина Е.Ю.
(подпись, дата)

Работу к защите допустить
Декан факультета  16.06.2025 Фионова Л.Р.
(подпись, дата)

ПГУ 1.000301.10.001 ПЗ

Web-приложение «Бакалавр»
интернет-консультация
Пояснительная записка

ОБТ, каф. БТ
ф. 2150/71

Реферат

Бакалаврская работа содержит 111 листов, 37 рисунков, 10 источников и 3 приложения.

WEB-ПРИЛОЖЕНИЕ «Книжный интернет-магазин»

Цель работы — разработка полнофункционального web-приложения «Книжный интернет-магазин», предоставляющего пользователям возможность искать, просматривать, заказывать и оценивать книги в онлайн-формате, а также авторам — публиковать собственные произведения с возможностью модерирования со стороны администратора. Проект ориентирован на создание удобной и доступной платформы для читателей, независимых авторов и управляющего персонала.

Объектами разработки являются:

- web-приложение с клиентской частью на React и серверной частью на Flask;
- база данных на SQLite, содержащая информацию о книгах, пользователях, заказах и отзывах;
- система регистрации и входа с разграничением ролей (пользователь, автор, администратор);
- каталог книг с функциями фильтрации, поиска и отображения подробной информации;
- корзина, оформление заказов и история покупок;
- система подачи заявок на получение авторского статуса и добавление новых книг;
- панель администратора для управления пользователями, книгами, отзывами и заявками.

ПГУ 1.090301.10.001 ПЗ							
Изм	Лист	№ докум.	Подпись	Дата	Web-приложение «Книжный интернет-магазин». Пояснительная записка		
Разраб.		Лукина Е.Ю.		30.05.25			
Пров.		Акифьев И.В.		30.05.25	Литера		
					Лист		
					Листов		
Н. Контр.		Бычков А.С.		30.05.25	ФВТ, каф. ВТ		
Утв.		Митрохин М.А.		30.05.25	гр. 21ВВП1		

В результате проделанной работы реализовано web-приложение с полной пользовательской и административной логикой.

Реализованы:

- регистрация и аутентификация пользователей с разграничением прав доступа;
- публикация, редактирование и удаление книг для авторов;
- управление заявками на получение авторского статуса и модерация контента администраторами;
- функционал корзины, оформление заказов, просмотр истории;
- система отзывов с возможностью их удаления администратором;
- адаптивный интерфейс, обеспечивающий корректную работу на различных устройствах.

Проект демонстрирует возможность создания удобного и расширяемого web-приложения для онлайн-продажи книг, включающего поддержку независимых авторов и эффективную административную модерацию. Полученные результаты могут быть использованы как в учебных целях, так и в рамках реальных проектов электронной коммерции.

					ПГУ 1.090301.10.001 ПЗ	Лист
						2
Изм.	Лист	№ докум.	Подпись	Дата		

Содержание

Введение.....	5
1 Web-приложение «Книжный интернет-магазин».....	8
1.1 Анализ предметной области	8
1.1.1 Актуальность разработки	8
1.1.2 Анализ существующих аналогов.....	9
1.2 Постановка задачи.....	18
1.3 Выбор технологий и средств разработки приложения	20
1.3.1 Выбор технологий для клиентской части.....	21
1.3.2 Выбор технологий для серверной части.....	21
1.3.3 Выбор системы управления базами данных	21
1.3.4 Выбор формата обмена данными	22
1.4 Архитектура приложения.....	23
1.4.1 Архитектурный паттерн MVVM	23
1.4.2 Клиент-серверное взаимодействие и REST API.....	23
1.4.3 Основные серверные классы и модели.....	24
1.4.4 Управление доступом по ролям	24
1.4.5 Масштабируемость и расширяемость архитектуры.....	25
1.5 Разработка приложения	25
1.5.1 Разработка клиентской части.....	25
1.5.2 Разработка серверной части.....	30
1.6 Описание приложения	33
1.6.1 Руководство пользователя.....	33
2 Охрана труда.....	35
2.1 Эргономические требования	35
2.2 Организация рабочего места разработчика	35
2.3 Правовое соответствие	36
2.4 Требования к тестированию и качеству	37
Заключение	43
Список использованных источников	45

Приложение А – UML Диаграммы.....	46
Приложение Б – Листинг программы	51
Приложение С – Презентация.....	99

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		4

Перечень сокращений и обозначений

API (Application Programming Interface) – интерфейс программирования приложений, набор инструментов для взаимодействия между компонентами программного обеспечения.

MVVM (Model View ViewModel) – шаблон проектирования архитектуры приложения.

Web-приложение – клиент-серверное приложение, в котором клиент взаимодействует с веб-сервером при помощи браузера.

ORM (Object-Relational Mapping) – технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

JSON (JavaScript Object Notation) - текстовый формат обмена данными, основанный на JavaScript.

JWT (JSON Web Token) – это открытый стандарт для создания токенов доступа, основанный на формате JSON.

CORS (Cross-Origin Resource Sharing) – технология современных браузеров, которая позволяет предоставить веб-страницам доступ к ресурсам другого домена.

Фреймворк – программная платформа, которая упрощает разработку программного продукта, определяет структуру проекта.

ГОСТ – региональный стандарт, принятый Межгосударственным советом по стандартизации, метрологии и сертификации Содружества Независимых Государств.

Введение

В двадцать первом веке технологии играют ключевую роль практически во всех сферах деятельности человека, включая торговлю, образование, здравоохранение и культуру. С развитием интернета получили распространение различные онлайн-сервисы, что существенно изменило способ взаимодействия между продавцом и покупателем. Всё больше людей предпочитают приобретать товары через интернет, что так же актуально и в сфере продажи книг, где важны не только удобство выбора, но и скорость поиска нужной литературы.

Онлайн-магазины книг позволяют покупателю не только ознакомиться с ассортиментом, но и получать информацию о наличии товара, удобно сравнивать цены и быстро совершать покупку. Более того, они предоставляют информацию, которую сложно получить в физических магазинах: отзывы и рецензии на книги от других покупателей. При этом не все существующие web-решения обеспечивают необходимый уровень удобства и интуитивности. Многие системы страдают от перегруженного интерфейса, недостаточной адаптации под мобильные устройства или неэффективного поиска. Это говорит о наличии спроса на новые решения, более простые и удобные в использовании.

Разработка собственного web-приложения «Книжный интернет-магазин» позволяет спроектировать интерфейс, ориентированный на потребности реальных пользователей, наладить удобную систему поиска, фильтрации, оформления заказов, а также обеспечить безопасную работу с пользовательскими данными. Внедрение подобного программного решения может быть полезным для начинающих издательств или авторов, желающих продвигать свои книги через интернет.

Целью выпускной работы является:

- исследование предметной области и анализ существующих решений;
- разработка web-приложения для поиска, ознакомления и приобретения книг онлайн;
- создание удобного интерфейса пользователя с функциями авторизации, фильтрации и управления корзиной;

					ПГУ 1.090301.10.001 ПЗ	Лист
						6
Изм.	Лист	№ докум.	Подпись	Дата		

- проектирование структуры базы данных, обеспечивающей хранение информации о книгах, пользователях и заказах;
- реализация функционала личного кабинета, отзывов и рейтингов;
- обеспечение адаптивности интерфейса для корректной работы на различных устройствах.

В ходе выполнения работы будет проведён анализ существующих решений в сфере онлайн-продажи книг, сформулированы основные требования к разрабатываемому приложению, спроектирована архитектура системы, реализованы ключевые модули и проведено тестирование готового продукта.

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

1 Web-приложение «Книжный интернет-магазин»

1.1 Анализ предметной области

1.1.1 Актуальность разработки

Книга — это один из старейших и важнейших источников знаний, и, несмотря на развитие цифровых технологий, спрос на литературу не только не снижается, но и приобретает новые формы. Люди продолжают читать и покупать книги — как в бумажном, так и в электронном виде. Однако традиционные книжные магазины, особенно в небольших населённых пунктах, не всегда могут обеспечить широкий ассортимент, быстрый поиск нужной книги или удобный способ заказа. Более того, режим работы таких магазинов ограничен временем. Всё это формирует потребность в современных и доступных web-решениях для книжной торговли.

В условиях растущей популярности онлайн-покупок всё большее значение приобретают интернет-магазины, позволяющие за считанные минуты найти и заказать нужное издание. При этом не все существующие платформы соответствуют ожиданиям пользователей: где-то неудобная система поиска, где-то перегруженный интерфейс или, наоборот, ограниченный функционал. Многие из таких решений ориентированы исключительно на крупные сети и не учитывают потребности небольших книжных магазинов, частных продавцов или начинающих издателей.

Web-приложение "Книжный интернет-магазин", разрабатываемое в рамках данной бакалаврской работы, направлено на решение этих проблем. Его основное преимущество — простота, доступность и адаптированность под реальные нужды конечных пользователей: как продавцов, так и покупателей. Для продавца — это инструмент управления ассортиментом и заказами. Для покупателя — это удобный и понятный интерфейс с возможностью быстрого поиска, отзывов, фильтрации по жанрам, авторам и форматам книг.

Особую актуальность проект приобретает для независимых авторов и начинающих писателей, которые не обладают ресурсами для размещения своих

					ПГУ 1.090301.10.001 ПЗ	Лист
						8
Изм.	Лист	№ докум.	Подпись	Дата		

произведений на крупных коммерческих платформах. Разработка доступного web-приложения позволит таким авторам представить свои книги широкой аудитории, упростит взаимодействие с читателями, а также создаст дополнительные возможности для продвижения и распространения их творчества в онлайн-среде.

Простота, надёжность и практичность — вот те качества, которые делают данный проект востребованным среди пользователей и значимым в условиях современной электронной коммерции.

1.1.2 Анализ существующих аналогов

Среди аналогов книжных интернет-магазинов можно выделить следующие web-сервисы:

- ЛитРес - <https://www.litres.ru> [1];
- Лабиринт - <https://www.labirint.ru> [2];
- Яндекс Книги - <https://books.yandex.ru> [3].

Рассмотрим сайт ЛитРес.

Общий вид данного сервиса представлен на рисунке 1.1.

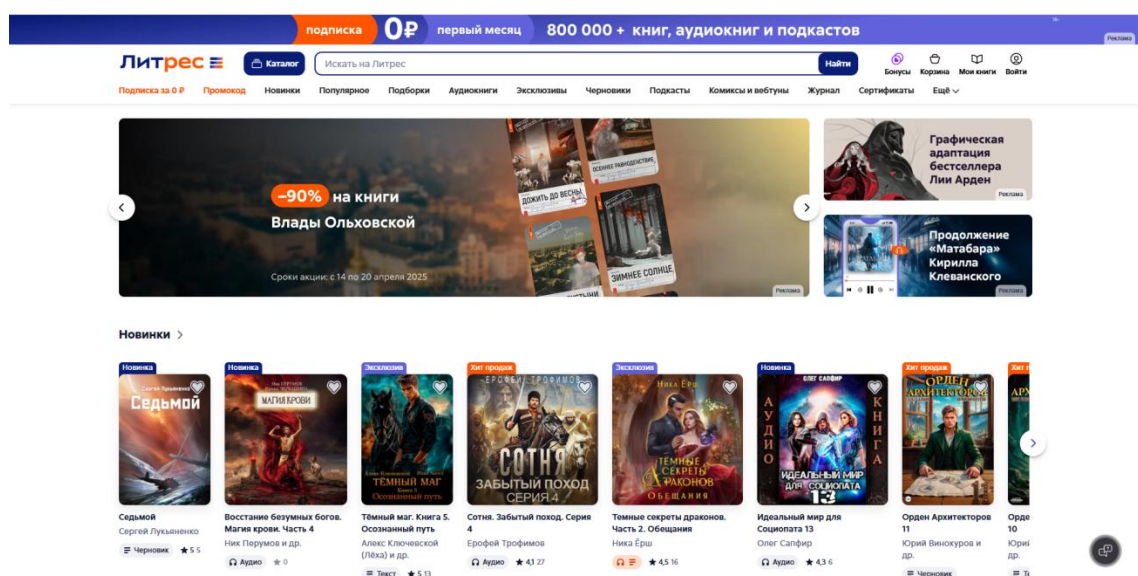


Рисунок 1.1 – Главная страница ЛитРес

Для визуального ознакомления с ассортиментом книг не обязательно проходить регистрацию. Также, без регистрации пользователя доступны

следующие функции: добавление книги в «избранное» (рисунок 1.2), добавление книги в корзину для дальнейшей покупки (рисунок 1.3).

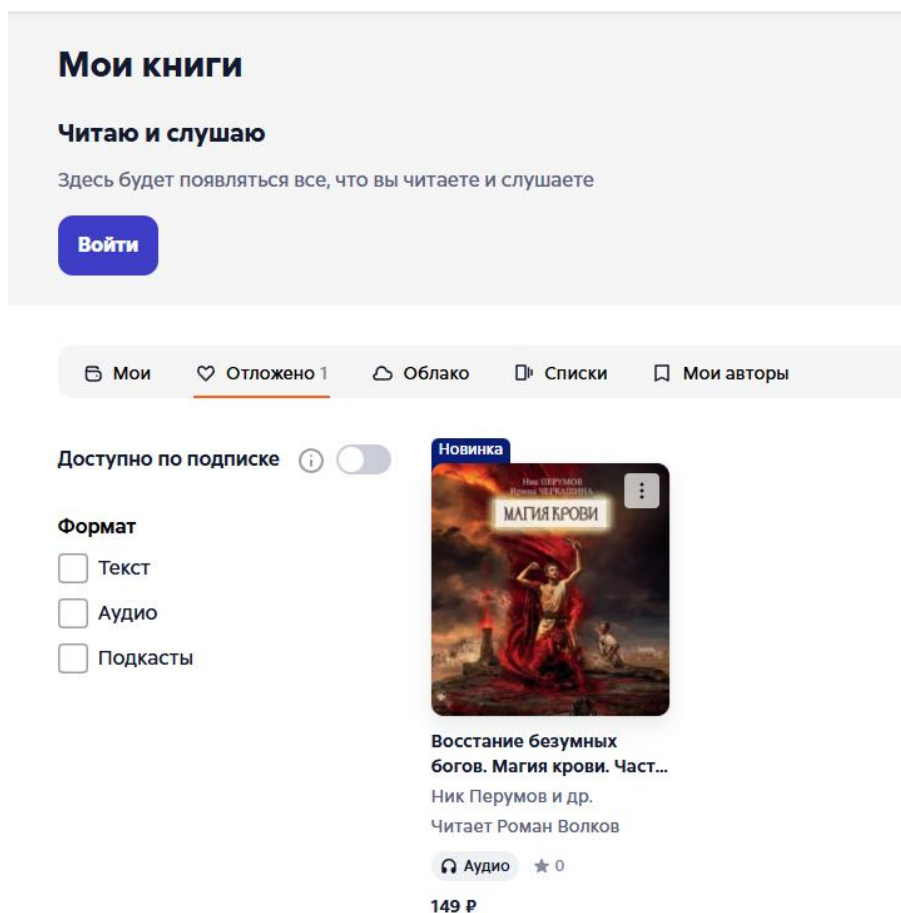


Рисунок 1.2 – Список «избранных» книг

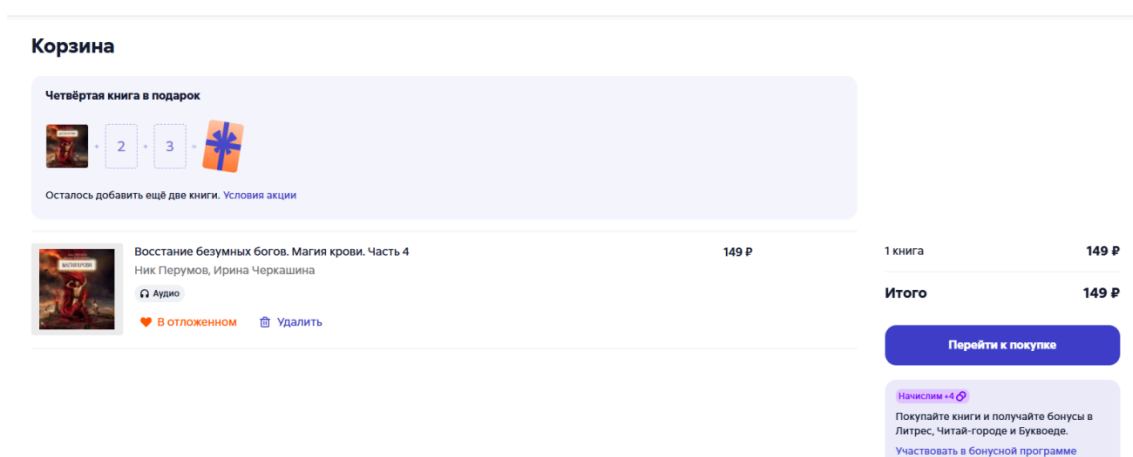


Рисунок 1.3 – Список товаров в корзине

После регистрации открывается полный список доступных функций, в том числе: просмотр списка приобретенных книг, возможность создания публичных

					ПГУ 1.090301.10.001 ПЗ	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

списков литературы, просмотр и редактирование списка подписок на авторов. Данная информация располагается в разделе «мои книги» (рисунок 1.4).

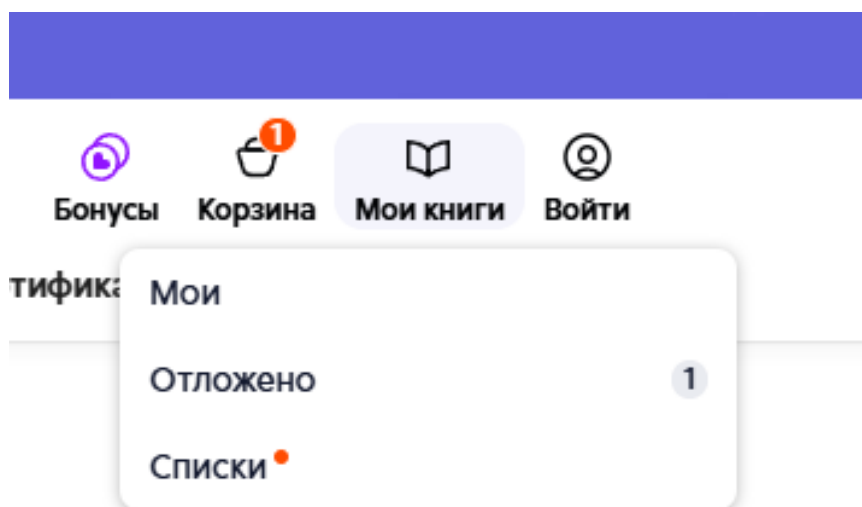


Рисунок 1.4 – Раздел «Мои книги»

Для совершения покупки также необходима авторизация. Доступно два способа покупки: по абонементу и обычная покупка.

Обычная покупка перенаправляет пользователя на экран выбора оплаты (рисунок 1.5) и после успешного подтверждения операции покупатель получает файл для скачивания.

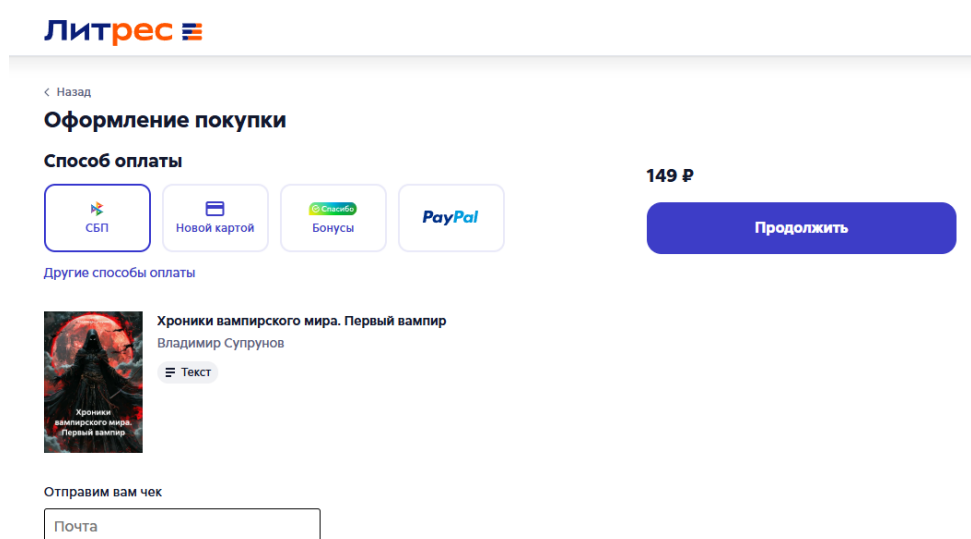


Рисунок 1.5 – Выбор оплаты

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		11

Покупка по абонементу предоставляет возможность приобрести любую из книг, участвующих в данной акции и стоящую до 700 рублей, накопительную скидку и две бесплатные книги из подборки для владельцев абонемента (рисунок 1.6).

Абонемент

Три книги на выбор, растущая скидка и многое другое за 399 ₽

Купить абонемент за 399 ₽

Купить в подарок

Каждый месяц для вас

Книга до 700 ₽

Одна любая электронная или аудиокнига до 700 ₽ из самого большого каталога в России

Персональная подборка

Две книги на выбор из персональной подборки

Книги навсегда ваши

Даже если абонемент закончится. Скачивайте, читайте и слушайте их на любом устройстве.

Дольше абонемент - больше скидка

Накапливаемая скидка на покупку книг, которые не входят в абонемент

10% сразу

20%

30%

с 3 месяца

с 6 месяца

Рисунок 1.6 – Описания условий «Абонемента»

Сервис предоставляет возможность оформления месячной подписки трех разных уровней, открывающей такие преимущества, как: возможность скачивать и читать любую книгу из ассортимента, открывает синхронизацию прогресса на всех устройствах для одного аккаунта, скидка 30% на весь каталог (рисунок 1.7).

					ПГУ 1.090301.10.001 ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

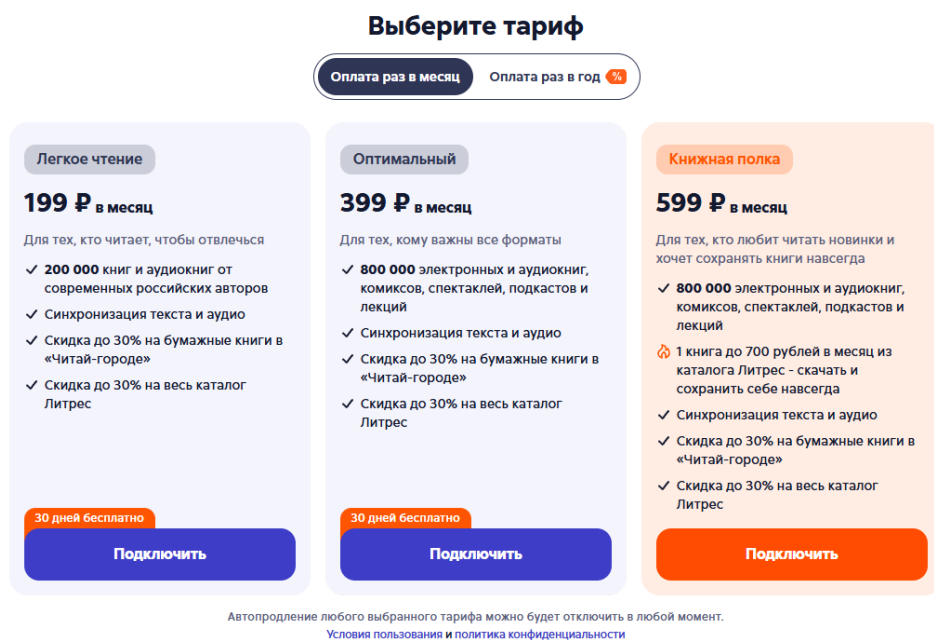


Рисунок 1.7 – Список тарифных планов месячной подписки

Помимо представленных функций для рядового пользователя, у сервиса ЛитРес имеется отдельный сайт для оформления себя в качестве автора: ЛитРес Самиздат (рисунок 1.8). Данное web-приложение является полу-автоматизированной системой для писателей, благодаря которой возможна публикация собственных книг на сайте ЛитРес и получение прямых выплат от каждой продажи.

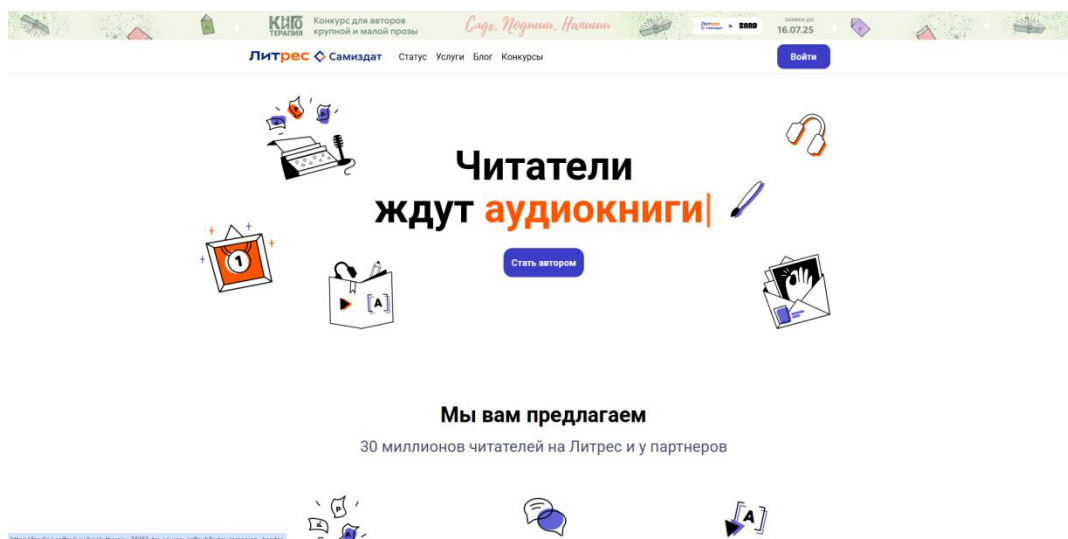


Рисунок 1.8 – Главная страница «ЛитРес: Самиздат»

Данный интернет-магазин обладает следующими преимуществами:

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		13

- продуманный и лаконичный интерфейс;
- большой ассортимент;
- наличие персональных рекомендаций;
- широкий выбор форматов книг;
- привязка к аккаунту;
- возможность бесплатного ознакомления с фрагментами из книги до покупки;

- система «абонемент» - выгодно для активных читателей;
- программа самиздата для авторов.

Из минусов данного сервиса можно выделить следующее:

- отсутствие возможности приобрести бумажные издания;
- некоторая часть функций доступна только при наличии платной подписки;

- ограничение на скачивание и копирование файлов;
- большая конкуренция для авторов;
- автор платит за публикацию своей книги;
- автор получает только часть прибыли.

Сайт Лабиринт.

Общий вид главной страницы сервиса представлен на рисунке 1.9.

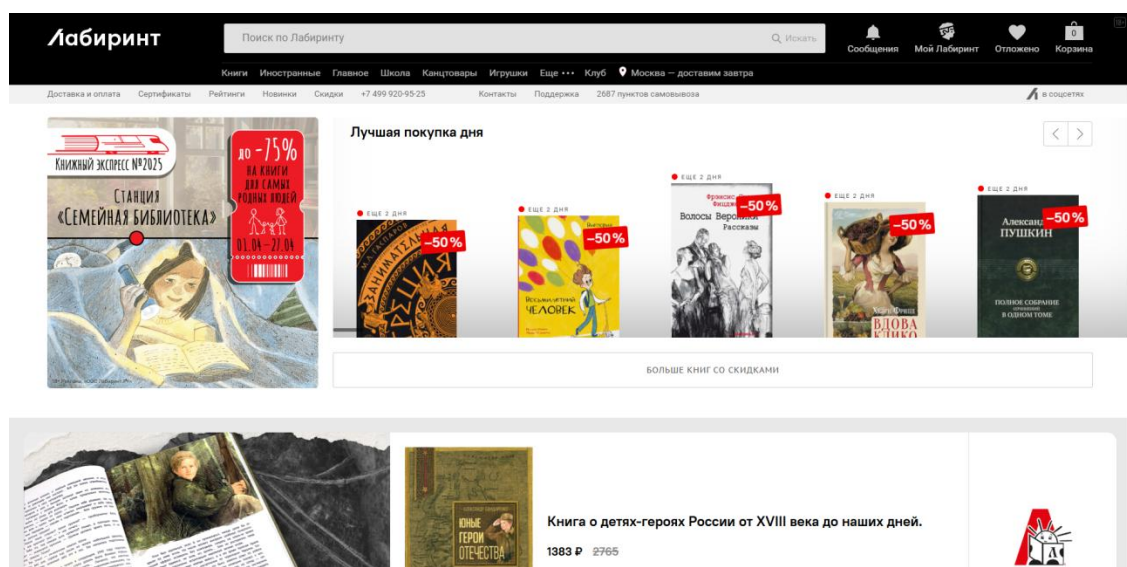


Рисунок 1.9 – Главная страница сайта Лабиринт

Для визуального ознакомления с ассортиментом книг регистрация не требуется — пользователю доступен просмотр обложек, кратких описаний, рейтингов, а также информации о наличии и цене товара.

Без регистрации пользователь может добавить книги в корзину, а также в отложенные (рисунок 1.10).

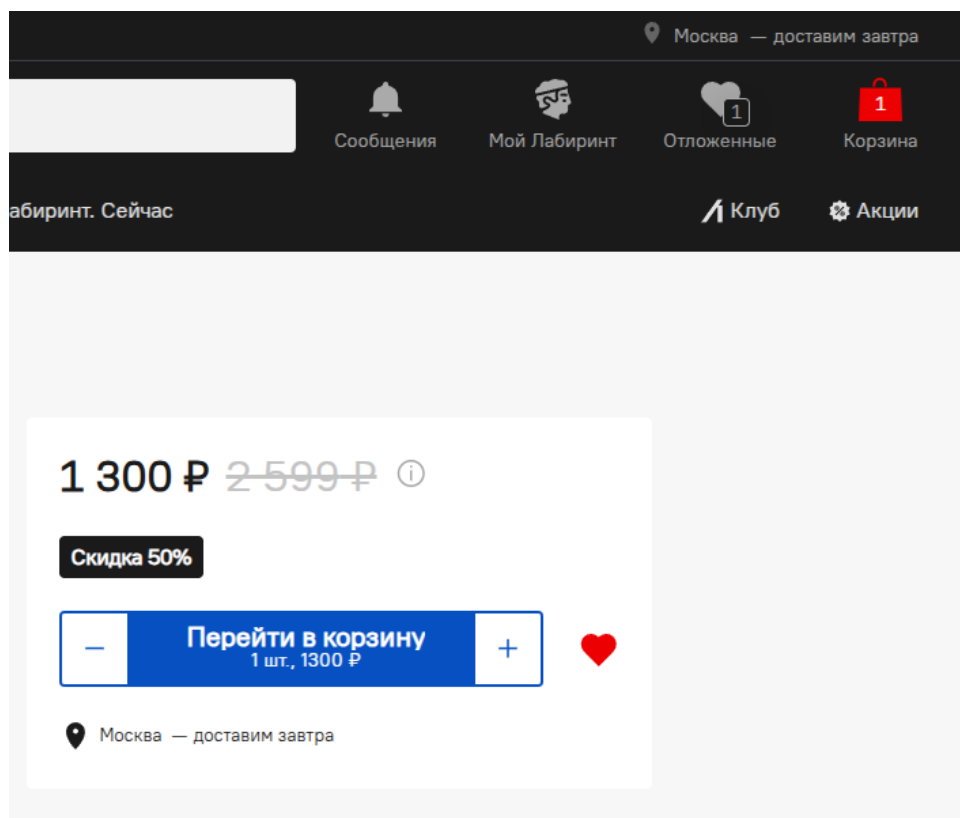


Рисунок 1.10 – Раздел «корзина» и «отложенные»

После авторизации становятся доступным оформление заказа, управление списками желаемого, возможность оставлять собственные рецензии и ставить оценки книгам (рисунок 1.11).

Рисунок 1.11 – Функция «написать рецензию»

Данный сервис распространяет книги только на бумажном носителе, поэтому при совершении покупки пользователю предлагается заполнить форму с адресом доставки (рисунок 1.12).

Рисунок 1.12 – Оформление заказа

Web-приложение Литрес обладает следующими преимуществами:

- большой ассортимент;
- удобная система фильтров;
- подробные карточки книг;
- наличие дополнительных товаров;
- самовывоз и доставка по всей России.

Из минусов данного сервиса можно выделить следующее:

- отсутствие электронных и аудиокниг
- сайт визуально перегружен;
- плохая адаптация под разные устройства;
- нет системы рекомендаций;
- нет возможности самопубликации.

Сайт Яндекс Книги.

Общий вид главной страницы сервиса представлен на рисунке 1.13.

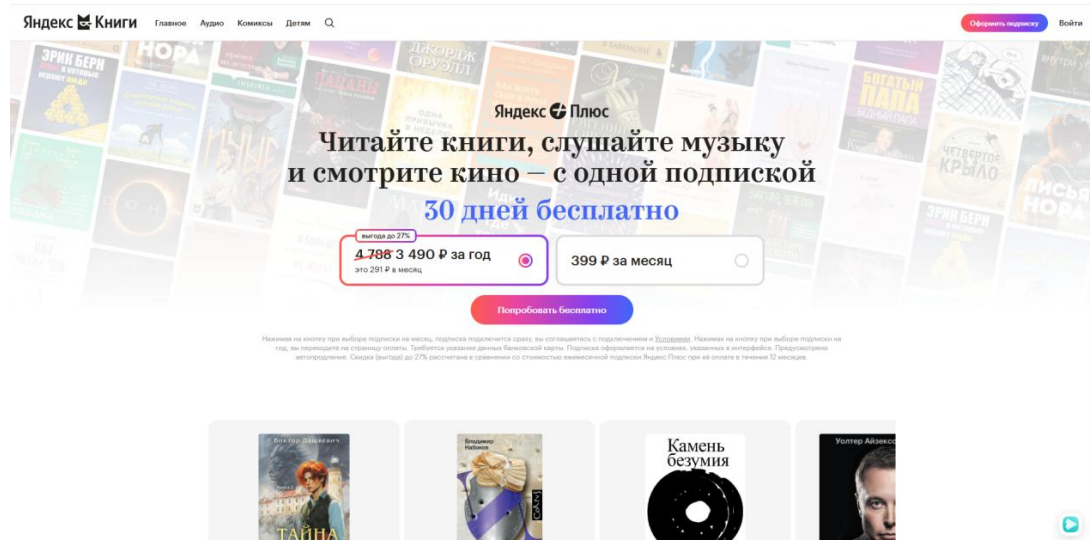


Рисунок 1.13 – Главная страница сайта Яндекс Книги

Без регистрации пользователь может просматривать каталог книг и их карточки.

После регистрации открываются функции просмотра бесплатного отрывка, добавление книги в отложенные, история просмотра книг и авторов и персонализированные рекомендации (рисунок 1.14).

Стандартной покупки электронной версии книги здесь не предусмотрено, пользователь должен приобрести платную подписку, открывающую возможность для скачивания любой понравившегося ему произведения из каталога (рисунок 1.15).

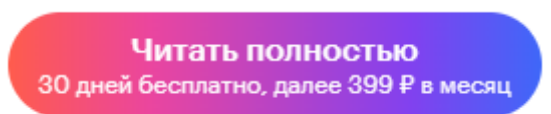


Рисунок 1.15 – Кнопка подключения платной подписки

Данный сервис обладает следующими преимуществами:

- хорошая навигация;
- интеграция с поисковой системой Яндекс для системы рекомендаций;
- адаптивный дизайн;
- минималистичный интерфейс.

Из минусов web-приложения Яндекс Книги можно выделить следующее:

- нельзя купить книгу без подписки;
- нет возможности самопубликации;
- ограничение на скачивание и копирование файлов;
- отсутствие возможности приобрести бумажные издания.

1.2 Постановка задачи

Требуется реализовать web-приложение, которое будет соответствовать следующим функциональным требованиям:

- регистрация и авторизация пользователей (создание нового аккаунта, вход в систему, восстановление пароля);
- система статусов (обычный пользователь, автор);
- работа с каталогом книг (отображение полного списка книг, просмотр информации о каждой книге: обложка, название, автор, жанр, аннотация, цена);
- поиск и фильтрация книг (по названию, автору, жанру, рейтингу, цене и другим параметрам);
- формирование заказа (добавление книг в корзину, оформление заказа, просмотр истории заказов);
- система отзывов и рейтингов (возможность оставить текстовый отзыв, поставить оценку, просматривать отзывы других пользователей);
- управление пользовательским профилем (просмотр и редактирование личных данных, смена пароля, удаление аккаунта);
- управление статусами пользователей и ассортиментом (для администратора: подтверждение/отклонение заявки на смену статуса «автор», проверка предложенных книг и информации о них);
- управление товарами (для автора: добавление новых книг, редактирование и удаление информации о товарах);
- адаптивный интерфейс (поддержка различных разрешений экранов, удобство при работе как с компьютера, так и с мобильного устройства).

Разрабатываемый онлайн сервис должен соответствовать следующим техническим требованиям:

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		18

- удобный и интуитивно понятный пользовательский интерфейс, обеспечивающий лёгкую навигацию по сайту;
- оформление страниц в едином стиле с акцентом на читаемость и простоту восприятия;
- поддержка адаптивной верстки для отображения на устройствах с различными диагоналями;
- использование клиент-серверной архитектуры;
- приложение должно работать в современных браузерах (Google Chrome, Mozilla Firefox, Microsoft Edge).

На рисунках 1.16 и 1.17 изображены диаграммы использования web-приложения с учетом ролевой модели доступа.

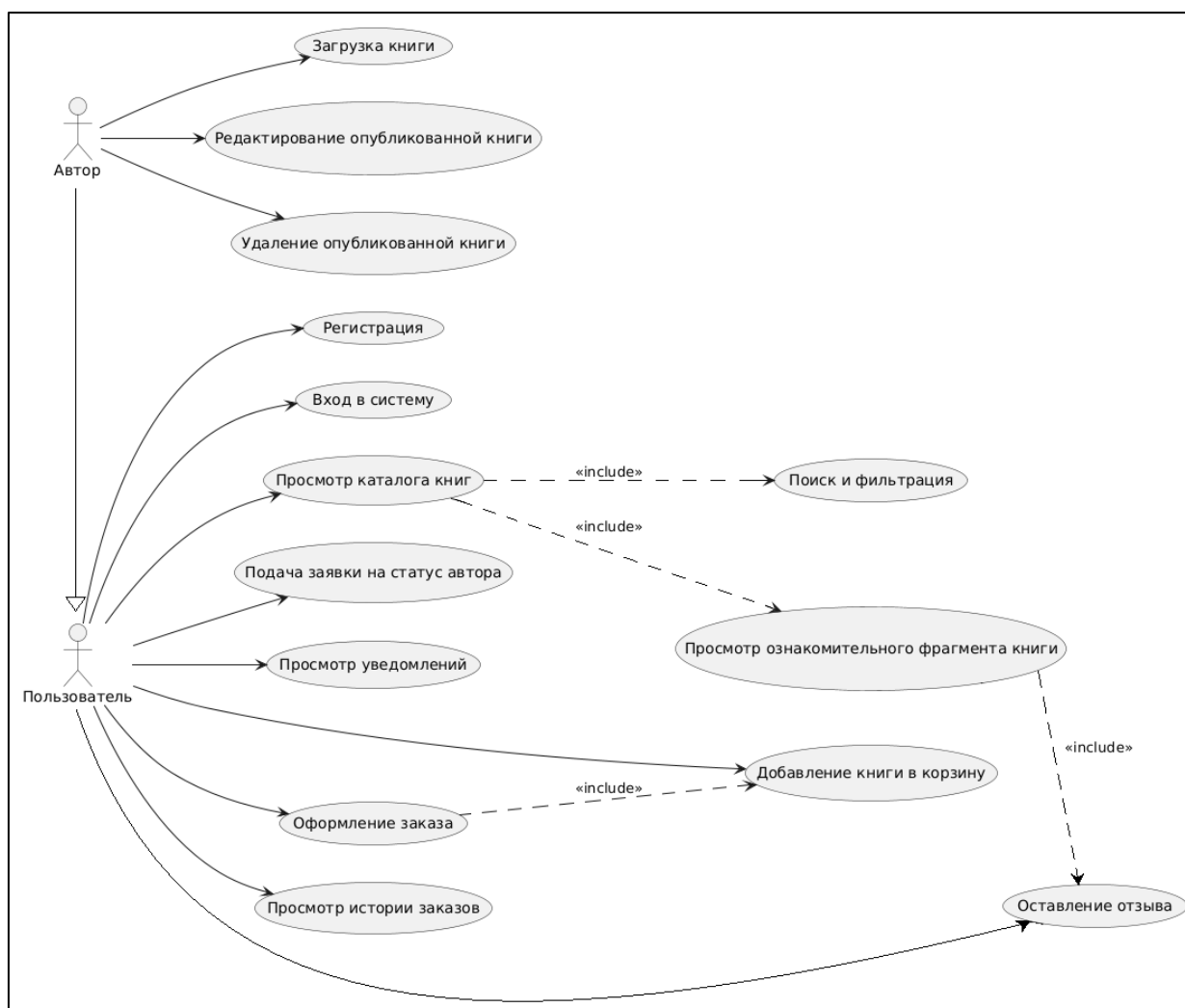


Рисунок 1.16 – Диаграмма вариантов использования web-приложения обычным пользователем и пользователем со статусом «автор»

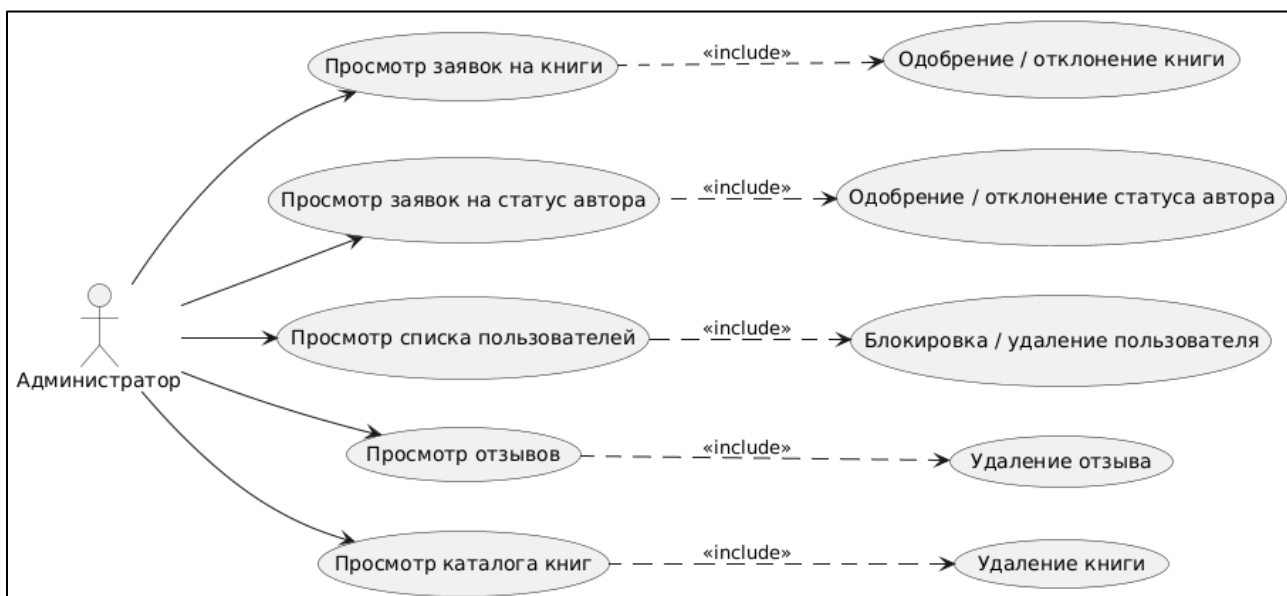


Рисунок 1.17 – Диаграмма использования web-приложения пользователя с правами администратора

1.3 Выбор технологий и средств разработки приложения

В данном разделе будут описаны технологии и инструменты, использованные при разработке web-приложения «Книжный интернет-магазин».

Ниже приведен список выбранных технологий и средств разработки:

- HTML5, CSS3;
- React;
- Python;
- Flask;
- SQLite;
- JSON.

1.3.1 Выбор технологий для клиентской части

HTML5 и CSS3 являются стандартными технологиями для создания структуры и оформления веб-страниц. HTML5 предоставляет расширенные возможности для семантической разметки, а CSS3 — для стилизации и адаптивного дизайна, что обеспечивает удобство использования на различных устройствах.

React — это JavaScript-библиотека для построения пользовательских интерфейсов, разработанная компанией Facebook. Она позволяет создавать динамичные и отзывчивые веб-приложения благодаря компонентному подходу и эффективному управлению состоянием [7].

Преимущества использования React включают:

- компонентную архитектуру, способствующую повторному использованию кода и упрощению сопровождения;
 - высокую производительность за счёт виртуального DOM и эффективного обновления интерфейса;
 - широкую поддержку сообщества и наличие множества готовых решений.
- [10]

1.3.2 Выбор технологий для серверной части

Python — высокоуровневый язык программирования с динамической типизацией, известный своей простотой и читаемостью кода. Он поддерживает различные парадигмы программирования и обладает богатой стандартной библиотекой, что делает его популярным выбором для веб-разработки.

Flask — это лёгкий и гибкий веб-фреймворк для Python, который предоставляет необходимые инструменты для создания веб-приложений и API. Он позволяет быстро начать разработку и легко масштабировать приложение по мере роста требований [4].

Преимущества использования Flask включают:

- простоту и гибкость, позволяющие быстро создавать и расширять приложения;
- возможность персонализации и выбора необходимых компонентов;
- полную документацию и активное сообщество, что обеспечивает поддержку при разработке. [8]

1.3.3 Выбор системы управления базами данных

SQLite — это встраиваемая реляционная система управления базами данных, которая хранит всю базу данных в одном файле. Она не требует отдельного сервера и легко интегрируется в приложения, что делает её

подходящей для небольших и средних проектов.

Преимущества использования SQLite включают:

- простоту установки и использования, без необходимости настройки сервер;
- надёжное хранение данных с поддержкой транзакций;
- поддержку различных платформ и широкое распространение в различных приложениях.

1.3.4 Выбор формата обмена данными

JSON (JavaScript Object Notation) — это текстовый формат обмена данными, основанный на синтаксисе JavaScript. Он широко используется в веб-разработке как стандартный способ передачи информации между клиентом и сервером, особенно в архитектурах с разделением на frontend и backend.

В рамках разрабатываемого web-приложения JSON используется для:

- обмена данными между React (frontend) и Flask (backend) через REST API;
- отправки данных форм при регистрации, входе в систему, оформлении заказа;
- получения и отображения информации о книгах, заказах, пользователях;
- передачи ошибок, уведомлений и других ответов от сервера.

Преимущества JSON:

- формат легко читается как человеком, так и машиной;
- поддерживается всеми языками программирования, включая JavaScript и Python;
- JSON-компактен и быстро обрабатывается;
- легко сериализует объекты, списки, вложенные структуры.

1.4 Архитектура приложения

1.4.1 Архитектурный паттерн MVVM

В разработке клиентской части приложения используется архитектурный паттерн MVVM (Model-View-ViewModel). Этот паттерн способствует разделению бизнес-логики, логики представления и пользовательского интерфейса, что упрощает тестирование, обслуживание и развитие приложения. MVVM состоит из трех компонентов:

- Model: представляет данные и бизнес-логику приложения;
- View: отвечает за отображение информации пользователю и взаимодействие с ним;
- ViewModel: служит посредником между Model и View, обрабатывая данные и команды.

Использование MVVM позволяет создавать более модульный и поддерживаемый код, особенно в сочетании с библиотекой React, которая способствует компонентному подходу в разработке пользовательского интерфейса [6].

На рисунке 19 показаны три компонента использующиеся в паттерне MVVM и отображено взаимодействие между ними.

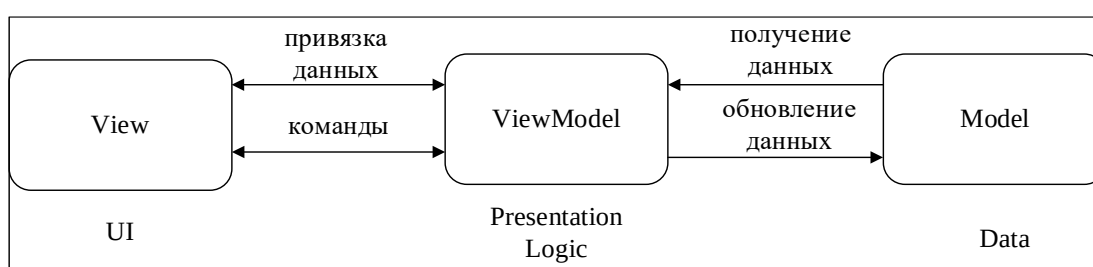


Рисунок 1.18 – Интерфейс для фактической нагрузки

1.4.2 Клиент-серверное взаимодействие и REST API

Клиентская часть приложения общается с сервером через REST API, реализованный во Flask. Сервер обрабатывает запросы от клиента, обращается к базе данных и возвращает результат в формате JSON. База данных, построенная

на SQLite, хранит основную информацию — сведения о книгах, заказах, пользователях и их ролях. [5]

Система предусматривает три типа пользователей: обычный пользователь, автор и администратор. Все они имеют различные права доступа и функциональные возможности. Это реализовано через встроенную систему авторизации и ролей, где каждая роль определяет набор доступных действий. При входе в систему пользователю выдаётся JWT-токен, который сохраняется на клиенте и используется для последующих запросов.

1.4.3 Основные серверные классы и модели

С точки зрения серверной логики, архитектура включает несколько ключевых компонентов:

- класс User — представляет пользователя системы, содержит такие поля, как id, username, email, password_hash и role, именно через это поле определяется доступ пользователя к различным разделам сайта;
- класс Book — модель книги, содержащая название, описание, цену, автора и статус модерации (например, «ожидает одобрения» или «опубликована»);
- класс Order и OrderItem — отражают структуру заказов, позволяя хранить как общие сведения о заказе, так и список конкретных книг, входящих в него;
- класс AuthorRequest — отдельная сущность, предназначенная для хранения заявок от пользователей, желающих получить статус автора.

1.4.4 Управление доступом по ролям

Взаимодействие между пользователем и системой организовано таким образом, чтобы предоставить доступ к нужной функциональности в зависимости от его роли:

- обычные пользователи могут просматривать каталог, оформлять заказы и подавать заявку на получение статуса автора;
- авторы получают возможность управлять своими книгами, а также получают уведомления при покупке одной из них;
- администратор имеет доступ к заявкам пользователей и контролю модерации новых книг, но не получает уведомлений о покупках.

Все роли обрабатываются централизованно на сервере, а их проверка на клиенте осуществляется через сохранённый JWT-токен и контекст пользователя.

1.4.5 Масштабируемость и расширяемость архитектуры

Благодаря использованию единой базы данных и чёткой архитектуры компонентов, система остаётся устойчивой к расширению.

Появление новых функций, страниц или ролей (например, модератора или рецензента) не требует переработки ядра приложения и может быть реализовано в рамках текущей архитектурной схемы.

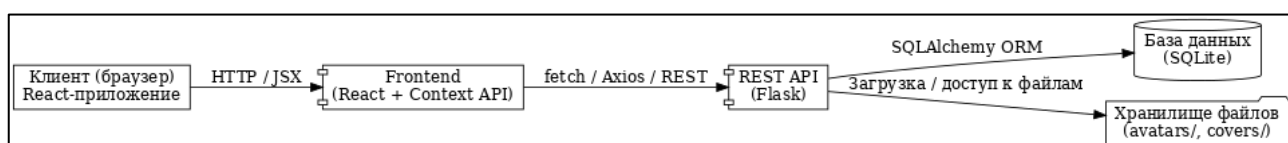


Рисунок 1.19 – Схема хранения загружаемых файлов

1.5 Разработка приложения

1.5.1 Разработка клиентской части

Клиентская часть web-приложения реализована с использованием библиотеки React. Приложение состоит из отдельных компонентов, каждый из которых отвечает за конкретный раздел пользовательского интерфейса. Компоненты сгруппированы по назначению: страницы (страницы-контейнеры), компоненты интерфейса (Header, Footer), и контексты состояния (UserContext, CartContext, FavoritesContext).

Компонент `App.jsx` является корневым и осуществляет маршрутизацию между страницами. С помощью библиотеки `react-router-dom` реализована логика маршрутов: `/catalog`, `/book/:id`, `/cart`, `/checkout`, `/profile`, `/login`, `/register`, `/favorites`, `/admin`, `/admin/requests`, `/admin/books`, `/admin/users`. В этом же компоненте подключаются провайдеры контекстов, таких как `UserContext`, `CartContext` и `FavoritesContext`.

Компонент `Header.jsx` отвечает за отображение навигационной панели. Он изменяет содержимое в зависимости от роли пользователя: для администратора

отображается ссылка на панель управления, для обычного пользователя — на корзину, профиль и избранное.

Контекст `UserContext.jsx` содержит глобальное состояние текущего пользователя и токена авторизации:

```
const login = async (email, password) => {
  const response = await fetch('http://localhost:5000/login', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ email, password })
  });
  const data = await response.json();
  if (data.token) {
    setToken(data.token);
    setUser(data.user);
  }
};
```

Также реализованы функции входа в систему, выхода и обновления данных пользователя. Компоненты, использующие этот контекст, могут получать доступ к информации о пользователе без необходимости повторных запросов.

Контекст `CartContext.jsx` управляет состоянием корзины. Он предоставляет функции для добавления и удаления книг, сохранения и очистки содержимого корзины:

```
const addToCart = (book) => {
  setCart((prevCart) => [...prevCart, book]);
};
```

Контекст `FavoritesContext.jsx` реализует логику избранных книг. Книги, добавленные в избранное, сохраняются локально и синхронизируются с сервером при необходимости.

Страница `Catalog.jsx` отображает список книг. Реализована фильтрация и сортировка по различным параметрам (жанру, алфавиту, новизне). Используются хуки состояния для хранения списка книг и параметров фильтрации. Запросы к серверу выполняются через fetch и axios.

Страница `Book.jsx` отображает подробную информацию о выбранной книге, предоставляет возможность добавить её в корзину или избранное. Помимо этого, здесь реализована форма добавления нового отзыва и их вывод:

```
const handleSubmitReview = async () => {
  await fetch(`http://localhost:5000/book/${bookId}/review`, {
```

					ПГУ 1.090301.10.001 ПЗ	Лист
						26
Изм.	Лист	№ докум.	Подпись	Дата		

```

method: 'POST',
headers: {
  'Content-Type': 'application/json',
  Authorization: `Bearer ${token}`
},
body: JSON.stringify({ rating, text })
});
};

```

Страница `Cart.jsx` отображает содержимое корзины. Здесь можно изменить количество экземпляров книги, удалить товар или перейти к оформлению заказа.

Страница `Checkout.jsx` реализует форму подтверждения заказа с отображением списка товаров, общей стоимости и кнопкой "Подтвердить заказ".

Страница `Profile.jsx` отображает личные данные пользователя и позволяет загрузить аватар. Также здесь отображаются заказы и книги пользователя, если он автор.

Компонент `AddBook.jsx` используется авторами для добавления новых книг. В форму включены такие поля, как название, псевдоним автора, жанр, описание, файл обложки, файл с ознакомительным фрагментом и файл с текстом книги, а также имеется обработка некорректного заполнения формы:

```

const handleSubmit = async e => {
  e.preventDefault();
  if (!title.trim() || !description.trim() ||
!price.trim() || isNaN(price)) {
    alert('Пожалуйста, заполните все поля и укажите корректную
цену.');
```

```

    return;
  }
  if (!coverFile) {
    alert('Пожалуйста, загрузите обложку книги.');
```

```

    return;
  }
  if (!bookFile) {
    alert('Пожалуйста, загрузите файл книги.');
```

```

    return;
  }
  if (!pseudonymId) {
    alert('Пожалуйста, выберите псевдоним.');
```

```

    return;
  }

  const fd = new FormData();
  fd.append('cover',          coverFile);
  fd.append('fragment',       fragmentFile || '');
  fd.append('book',           bookFile);

```

```

        fd.append('title',          title.trim());
        fd.append('pseudonym_id',   pseudonymId);
        fd.append('genre',          genre.trim());
        fd.append('description',     description.trim());
        fd.append('publication_year', publicationYear.trim());
        fd.append('price',           price.trim());

        const res = await
        fetch('http://localhost:5000/author/books', {
            method: 'POST',
            headers: { Authorization: `Bearer ${token}` },
            body: fd
        });

        if (res.ok) {
            setOverlay(true);
        } else {
            const data = await res.json();
            alert(data.error || 'Ошибка отправки');
        }
    };
};

```

Компонент `EditBooks.jsx` используется для редактирования уже опубликованных книг авторами. Реализована проверка прав доступа и вывод текущих данных книги.

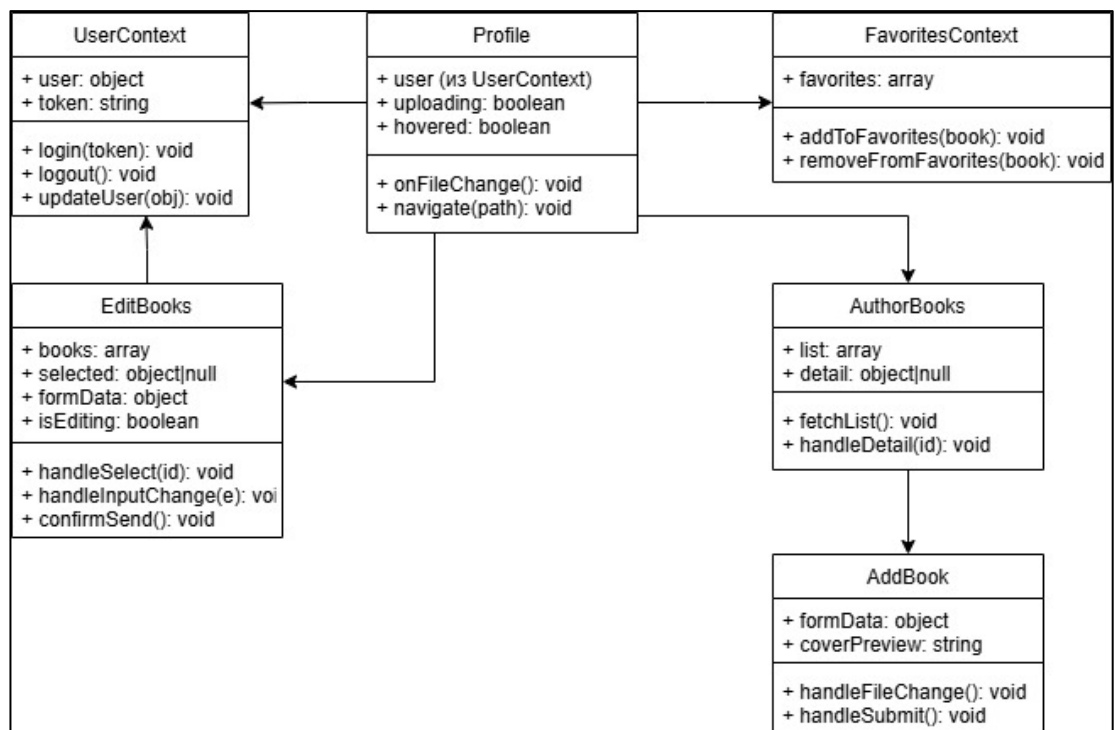


Рисунок 1.20 – Диаграмма классов пользователей

Панель администратора представлена в `AdminDashboard.jsx`. Внутри используются компоненты `NewBooksTab` (заявки на книги), `RequestsTab` (заявки на статус автора) и `UsersListTab` (список зарегистрированных пользователей). Для каждого типа запроса реализованы кнопки "одобрить" и "отклонить". Администратор может просматривать список пользователей, блокировать и удалять их.

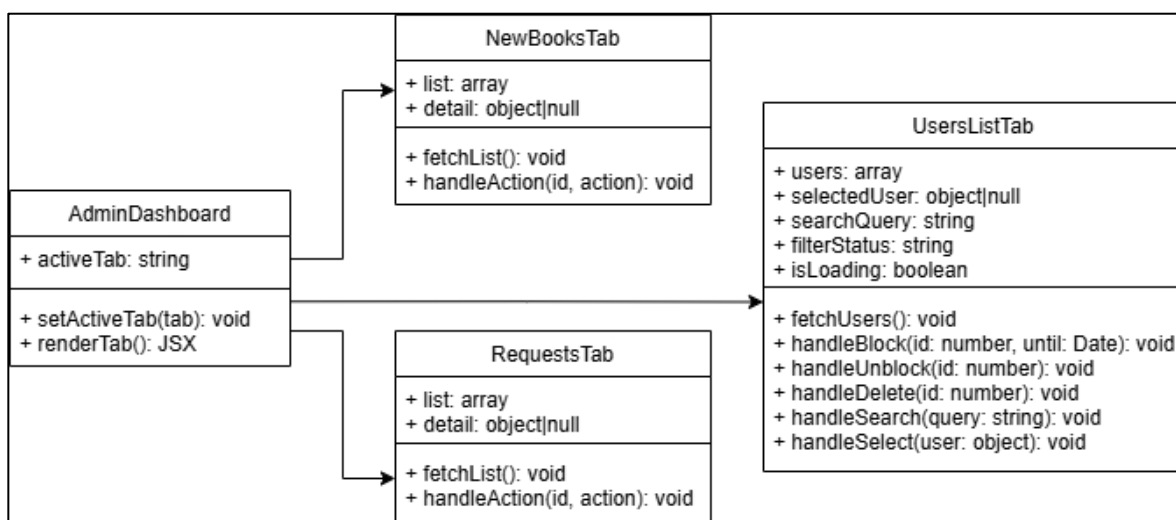


Рисунок 1.21 – Диаграмма классов пользователя с правами администратора

1.5.2 Разработка серверной части

Серверная часть реализована на языке Python с использованием фреймворка Flask. В файле `app.py` создаётся и конфигурируется объект Flask-приложения. Подключаются модули маршрутов, база данных и система CORS.

Маршруты сгруппированы в директорию `routes`. В `books.py` реализована логика:

- получения списка книг (с фильтрацией);
- добавления новой книги (проверка авторских прав);
- редактирования и удаления книги (автор или администратор);
- работы с отзывами: добавление, удаление, получение списка.

Следующим образом реализован данный маршрут:

```

@app.route('/books')
def get_books():
    books = Book.query.all()
    return jsonify([
        {
            'id': b.id,

```

```

        'title':      b.title,
        'author':     b.author,
        'genre':      b.genre,
        'description': b.description,
        'price':      b.price,
        'image_url':  b.image_url,
        'status':     b.status
    } for b in books])

```

Файл `users.py` содержит маршруты:

- регистрация и авторизация пользователя;
- обновление профиля;
- подача заявки на авторство;
- проверка прав и ролей;
- действия администратора над пользователями (удаление, блокировка).

```

@app.route('/register', methods=['POST'])
def register():
    data = request.get_json()
    hashed_password=
    bcrypt.hashpw(data['password'].encode('utf-8'),
    bcrypt.gensalt())
    new_user = User(email=data['email'], name=data['name'],
    password=hashed_password)
    db.session.add(new_user)
    db.session.commit()
    return jsonify({"message": "Пользователь зарегистрирован"})

```

База данных описана в файле `models.py`. Используется библиотека SQLAlchemy для ORM. Здесь описаны следующие модели:

- `User` — содержит имя, email, пароль (в зашифрованном виде), роль, статус автора;
- `Book` — заголовок, описание, автор, дата публикации, обложка;
- `Order` и `OrderItem` — описание заказов и включённых в них книг;
- `Review` — текст отзыва, оценка, автор и связь с книгой;
- `Favorite` — таблица избранных книг;
- `AuthorRequest` — заявки на статус автора, включая причину и статус рассмотрения.

Конфигурация приложения и секретные ключи определены в файле `config.py`. Подключение базы данных осуществляется через SQLite. Пример реализации:

					ПГУ 1.090301.10.001 ПЗ	Лист
						30
Изм.	Лист	№ докум.	Подпись	Дата		

```
class Config:
    SECRET_KEY = 'your_secret_key'
    SQLALCHEMY_DATABASE_URI = 'sqlite:///database.db'
    SQLALCHEMY_TRACK_MODIFICATIONS = False
```

Все ответы от сервера возвращаются в формате JSON. Для авторизации используется JWT (JSON Web Token), который проверяется на каждом защищённом маршруте.

Таким образом, реализация backend обеспечивает надёжную работу с данными, поддержку всех ролей пользователей и безопасную авторизацию. Представленная ниже диаграмма классов отражает отношения между объектами.

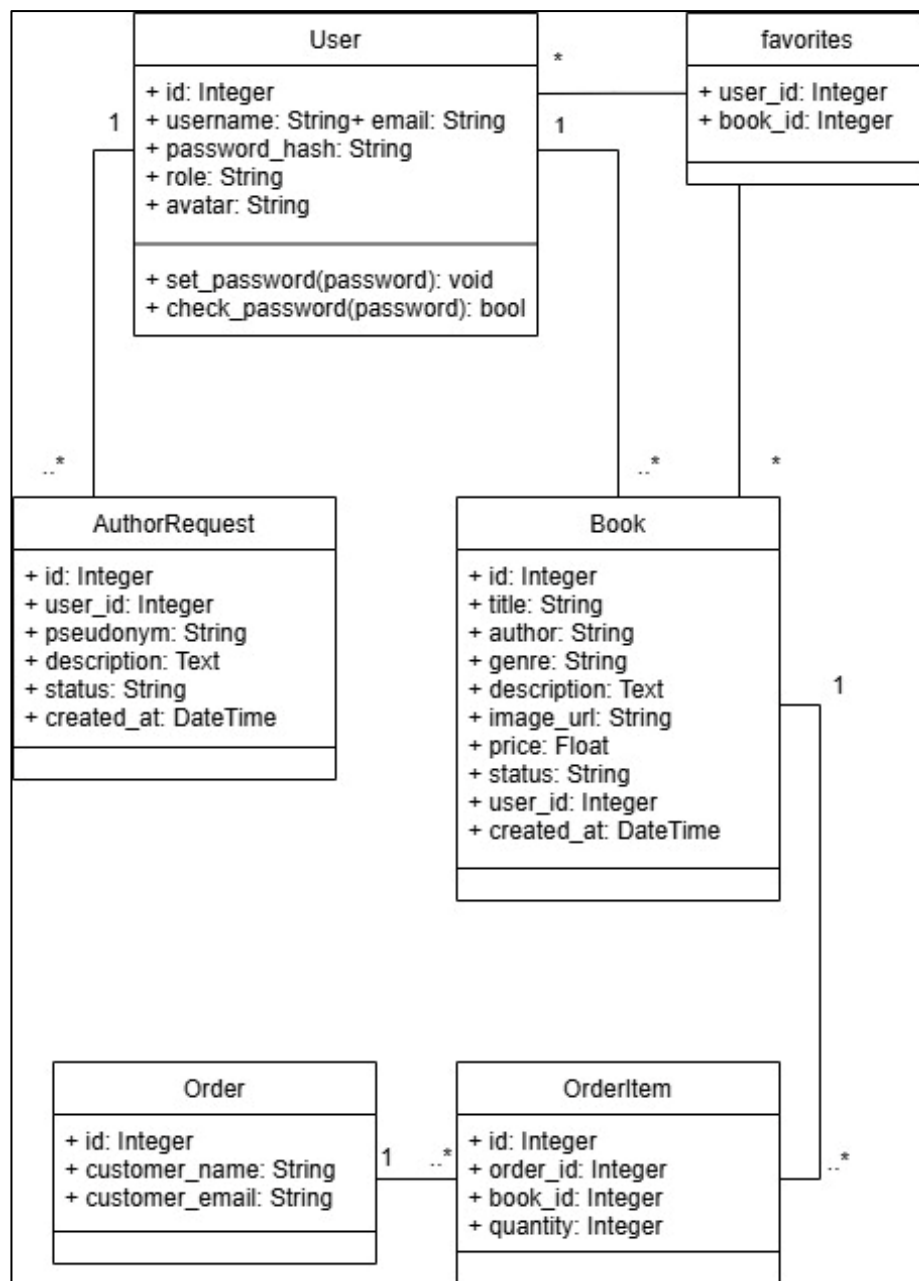


Рисунок 1.22 – Диаграмма классов backend

1.6 Описание приложения

1.6.1 Руководство пользователя

При входе на сайт пользователь попадает на главную страницу, где отображаются карточки книг. Незарегистрированному пользователю доступен ограниченный функционал — он может просматривать каталог книг и добавлять их в корзину, однако оформление заказа, добавление отзывов и доступ к личному кабинету доступны только после регистрации. При попытке совершения действий, требующих авторизации, пользователю отображается всплывающее окно с предложением войти в аккаунт или пройти регистрацию. Пользователи могут зарегистрироваться с указанием имени, электронной почты и пароля. При входе проверяется корректность введенных данных. Интерфейс формы входа представлен на рисунке 1.23.

Зарегистрироваться'." data-bbox="228 428 895 647"/>

Рисунок 1.23 – Форма входа в профиль

Каждая карточка содержит изображение обложки, название, автора, цену и кнопки "Добавить в корзину", а также "Добавить в избранное". Главная страница содержит навигационное меню и доступ к профилю пользователя.

На главной странице и в каталоге реализована функция поиска, позволяющая найти нужное произведение по ключевым словам. Здесь доступны фильтры по жанру, автору и алфавиту, что облегчает навигацию по каталогу. При нажатии на карточку книги пользователь попадает на её индивидуальную страницу, где представлена подробная информация: полное описание, жанр, отзывы других

пользователей, кнопки добавления в корзину и избранное. Ниже отображается форма для оставления отзыва.

Раздел "Корзина" содержит список всех выбранных книг, отображает их цену и итоговую сумму. Здесь же пользователь может удалить товар или перейти к оформлению заказа. Интерфейс корзины показан на рисунке 1.26.

Раздел "Оформление заказа" содержит форму с контактной информацией и кнопкой подтверждения. После оформления заказ сохраняется в базе данных, пользователь получает уведомление и возможность скачать файл купленной литературы. Интерфейс оформления заказа представлен на рисунке 1.27.

Оформление заказа

Имя

Email

Ваш заказ:

Дальняя планета 600 Р

Оформить

Рисунок 1.27 – Оформление заказа

В личном кабинете пользователь может видеть свои данные, изменить аватар и просмотреть историю заказов. Также здесь размещена кнопка подачи заявки на авторство. Интерфейс профиля представлен на рисунке 1.28.

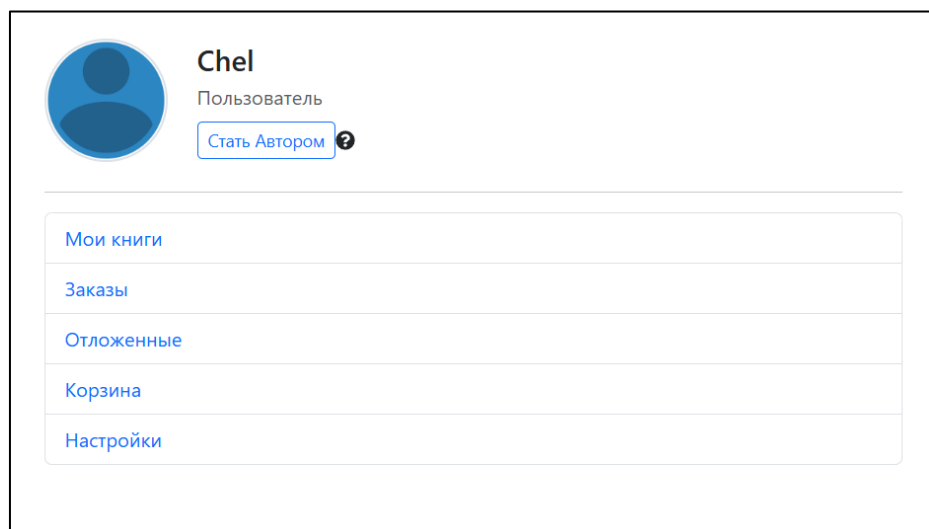


Рисунок 1.28 – Профиль обычного пользователя

Пользователь, подавший заявку и получивший статус автора, получает доступ к расширенному функционалу (рисунок 1.29). Он может добавлять новые книги, редактировать уже опубликованные, управлять своими произведениями через личный кабинет. Интерфейс формы добавления книги представлен на рисунке 1.30.

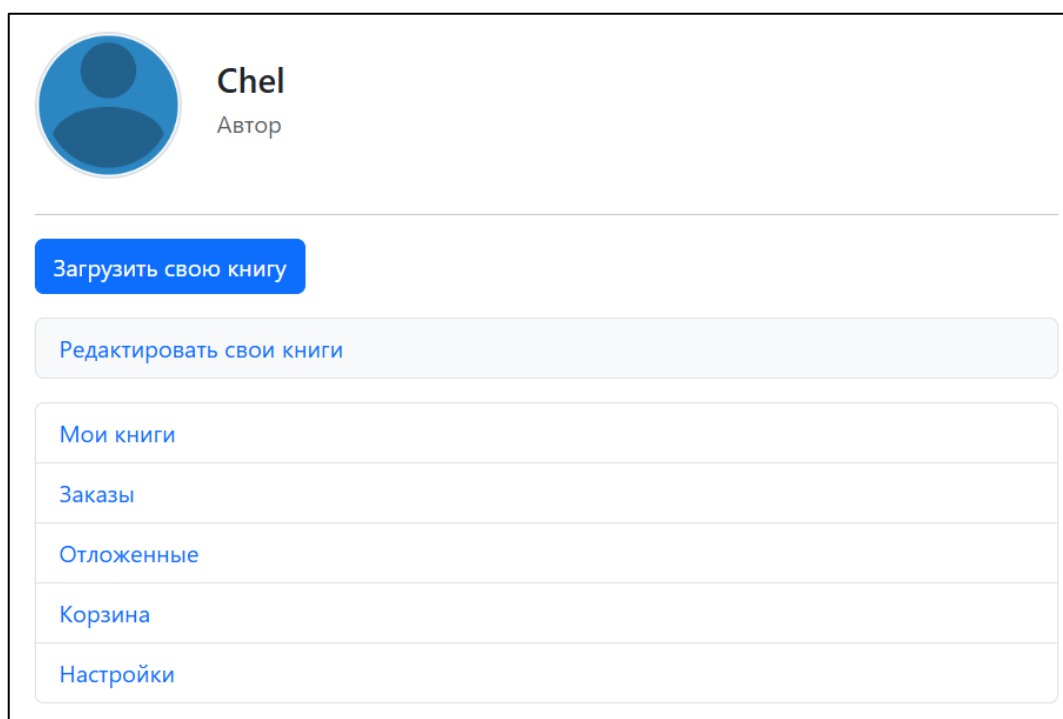


Рисунок 1.29 – Профиль пользователя со статусом «автор»

Добавить книгу

Загрузить обложку

Название книги

Псевдоним

Выберите псевдоним...

Жанр

Описание

Год издания

Цена (₽)

Ознакомительный фрагмент (необязательно)

Choose file

No file chosen

Загрузить книгу *

Choose file

No file chosen

Отправить

Рисунок 1.30 – Форма загрузки книги

Для публикации книги необходимо указать название, описание, выбрать жанр, загрузить обложку и прикрепить текст в электронном виде. Также доступна функция редактирования ранее размещённых материалов. На рисунке 1.31 продемонстрирован внешний вид карточки книги с подробной информацией, на рисунке 1.32 - интерфейс редактирования книги. После заполнения формы пользователь нажимает на кнопку «Отправить на модерацию», но данное действие требует дополнительного подтверждения, поскольку книга исчезнет из каталога на время проверки (рисунок 1.33).

Космический сигнал



Космический сигнал

Псевдоним: Вася Пулкин

Жанр: фантасти

Описание: История о том, как далекий космический сигнал дошёл до Земли

Издание: 2024г.

Цена: 500 Р

Скачать литературу

Редактировать

Назад

Рисунок 1.31 – Карточка книги с подробной информацией в списке опубликованных книг конкретного пользователя

Редактировать книгу

Название книги

Космический сигнал

Псевдоним

Выберите псевдоним...

Жанр

фэнтези

Описание

История о том, как далекий космический сигнал дошёл до Земли

Год издания

2024

Цена (₽)

500

Сменить обложку

Choose file No file chosen

Загрузить ознакомительный фрагмент

Choose file No file chosen

Загрузить литературу

Choose file No file chosen

Отправить на модерацию

Назад

Рисунок 1.32 – Форма редактирования книги автором

Внимание! Книга отправится на модерацию и будет временно недоступна для покупки.

Вы уверены?

Нет, назад

Я понял(а), продолжить

Рисунок 1.33 – Всплывающее предупреждение при нажатии на кнопку «Отправить на модерацию»

На странице книги пользователи могут оставлять отзывы с оценками. Отзывы отображаются публично, а администратор имеет возможность удалять неподобающие записи. Интерфейс формы отзыва представлен на рисунке 1.31.

Системой предусмотрен пользователь с правами администратора, в аккаунт которого можно зайти с помощью специального логина и пароля. Данный пользователь обладает уникальным функционалом, но при этом большая часть функционала, доступного обычным пользователям, например, возможность оформить заказ и изменить информацию профиля, ему недоступна.

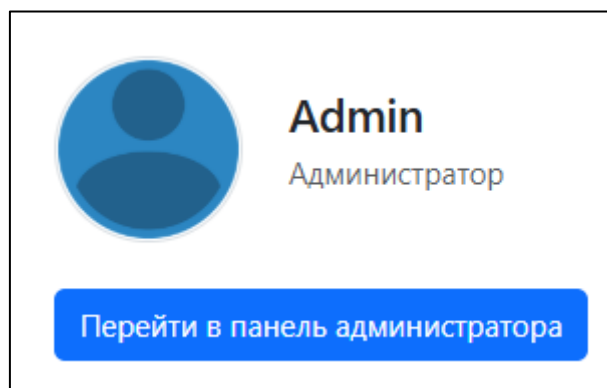


Рисунок 1.34 – Профиль администратора

Администратор имеет доступ к отдельной панели, где отображаются вкладки: "Заявки на книги", "Заявки на авторство" и "Пользователи". В первых двух вкладках администратор может просматривать, одобрять, отклонять или удалять записи. Вкладка «Пользователи» предназначена для управления списком зарегистрированных на сайте пользователей: администратор может удалить выбранный аккаунт или временно заблокировать в случае нарушения пользователем правил пользования. Интерфейс панели администратора представлен на рисунках 1.35 – 1.36.

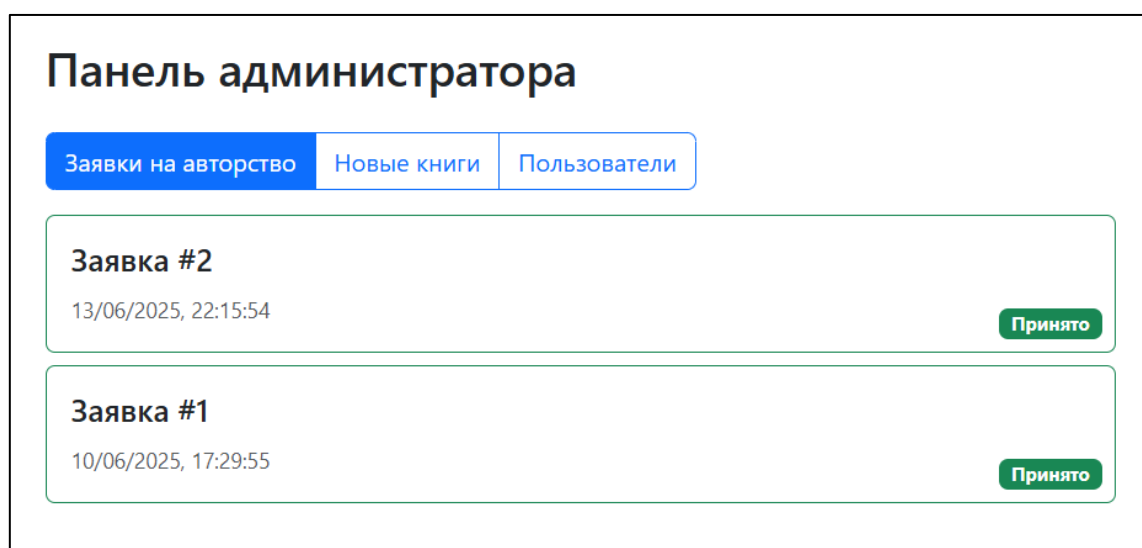


Рисунок 1.35 – Общий вид панели администратора

Панель администратора

[Заявки на авторство](#)[Новые книги](#)[Пользователи](#)

Книга «Мертвые души»



Мертвые души

Псевдоним: Николай Гоголь

Подача: 13/06/2025, 22:28:02

Жанр: Роман, Сатира, Фикшн

Описание:

В уездный городок приезжает отставной чиновник Чичиков, задумавший экстравагантную финансовую аферу. Его странные действия по покупке «мертвых душ» вносят изрядное смущение в тихую жизнь российской провинции...

Издание: 1842г.

Цена: 0 ₽

[Скачать литературу](#)

На рассмотрении

Одобрить

Отклонить

Назад

Рисунок 1.36 – Подробное описание отправленной автором книги

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

38

2 Охрана труда

2.1 Эргономические требования

В рамках разработки и использования web-приложения «Книжный интернет-магазин» особое внимание было уделено вопросам эргономики пользовательского интерфейса. Это позволяет минимизировать зрительную нагрузку, снизить вероятность переутомления пользователей и повысить общее удобство взаимодействия с системой.

Интерфейс сайта разработан с учётом требований визуального комфорта: применяются нейтральные цвета, крупные и читаемые шрифты, элементы управления размещены логично и последовательно. Обеспечена поддержка масштабирования интерфейса и адаптивная верстка, благодаря чему сайт корректно отображается на различных типах устройств — от настольных компьютеров до мобильных телефонов.

Важной задачей стало обеспечение читабельности контента. Минимизировано количество отвлекающих элементов и визуального шума. Навигация интуитивно понятна, а логическая структура страниц снижает когнитивную нагрузку на пользователя.

2.2 Организация рабочего места разработчика

Работа по созданию программного обеспечения ведётся за персональным компьютером, поэтому необходимо учитывать санитарные и гигиенические требования к организации рабочего места. Основными мерами по обеспечению комфортной и безопасной среды являются:

- установка монитора на уровне глаз на расстоянии 60–70 см от пользователя;
- обеспечение равномерного освещения без бликов и отражений на экране;
- использование регулируемого кресла с поддержкой поясницы;
- регулярные перерывы в работе: не реже чем один раз в час на 5–10 минут с выполнением упражнений для глаз и кистей рук.

Разработчику следует придерживаться сбалансированного режима труда и отдыха, что особенно важно при длительном нахождении за экраном. Также необходимо контролировать уровень шума, температуру воздуха и вентиляцию помещения в соответствии с нормами СанПиН.

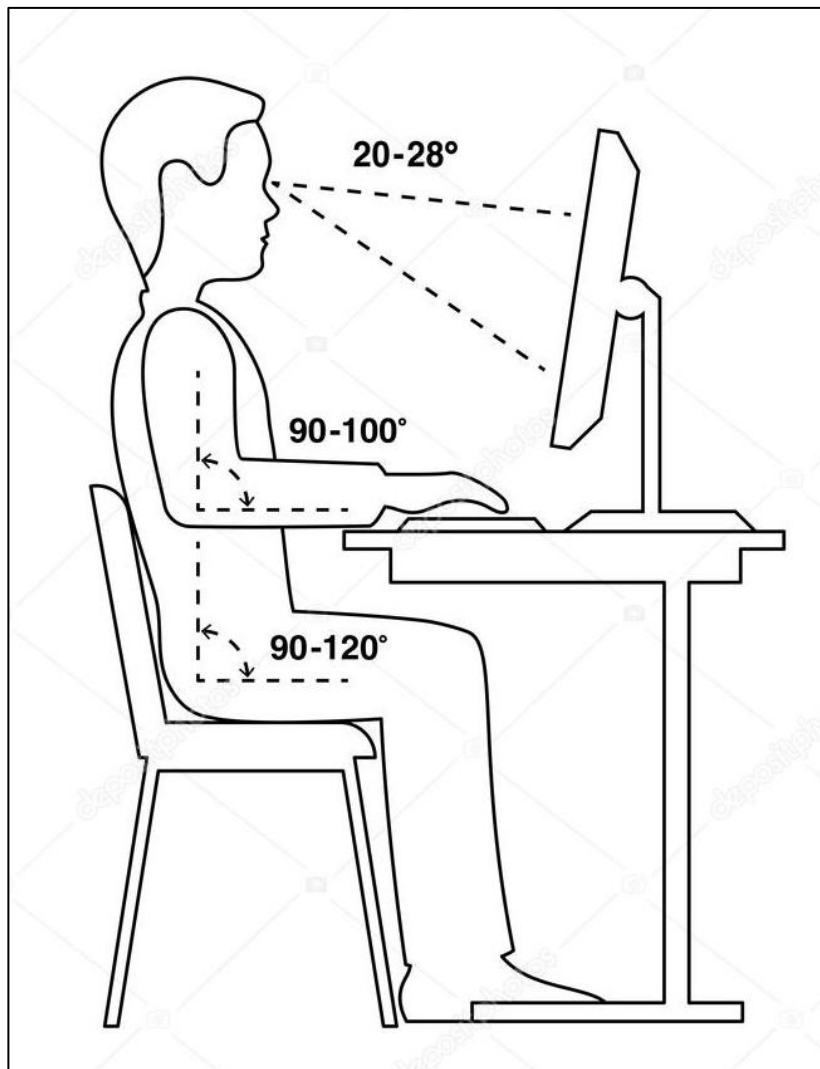


Рисунок 2.1 – Правильная посадка за ПК

2.3 Правовое соответствие

В процессе разработки web-приложения учитывались требования Федерального закона №152-ФЗ «О персональных данных». Приложение предусматривает сбор и временное хранение личных данных пользователей (имя, электронная почта, сведения о заказах), которые используются исключительно для целей идентификации и выполнения функционала сервиса.

Поскольку проект находится на стадии локальной разработки и тестирования, подключение защищённого протокола HTTPS на данный момент не реализовано и должно быть обеспечено при дальнейшем развертывании на продакшн-сервере или при использовании стороннего хостинга.

Тем не менее, предприняты базовые меры правовой и информационной безопасности:

- реализована система авторизации с разграничением прав доступа по ролям (пользователь, автор, администратор);
- действия, совершаемые пользователями, в том числе в административной панели, проверяются на стороне сервера и логируются;
- реализована структура, позволяющая при необходимости добавить пользовательское соглашение и политику конфиденциальности в интерфейс сайта и предусмотреть их подтверждение при регистрации пользователя.

2.4 Требования к тестированию и качеству

Немаловажным аспектом охраны труда является обеспечение устойчивой и безошибочной работы программного продукта. Ошибки в работе интерфейса, потеря данных и нестабильное поведение сайта могут привести к стрессу как у пользователей, так и у персонала, сопровождающего проект.

В целях повышения качества и надёжности были реализованы следующие меры:

- проведено тестирование интерфейса на различных устройствах и браузерах (кроссбраузерность);
- реализована проверка пользовательского ввода и вывод сообщений об ошибках в доступной форме;
- предусмотрены механизмы отката и перезагрузки ключевых компонентов при сбоях;
- протестированы сценарии отказоустойчивости и некорректного поведения со стороны пользователя (пустые поля, неверные данные, сбои подключения);

– реализована система уведомлений и сообщений об ошибках с пояснениями, позволяющими избежать недоразумений.

Таким образом, все этапы проектирования и внедрения web-приложения сопровождались мерами, направленными на соблюдение норм охраны труда как для разработчиков, так и для конечных пользователей.

					ПГУ 1.090301.10.001 ПЗ	Лист
						42
Изм.	Лист	№ докум.	Подпись	Дата		

Заключение

В ходе выполнения выпускной квалификационной работы было разработано web-приложение «Книжный интернет-магазин», предназначенное для поиска, ознакомления и приобретения книг в онлайн-формате. Основной акцент был сделан на создании интуитивно понятного интерфейса, обеспечении быстрого доступа к литературе, а также организации полноценной системы взаимодействия между покупателем, автором и администратором ресурса.

Клиентская часть приложения была реализована с использованием библиотеки React, что позволило добиться высокой отзывчивости интерфейса и адаптивности на разных устройствах. Серверная логика построена с помощью Python-фреймворка Flask, а в качестве хранилища данных использована реляционная база SQLite. В приложении реализована поддержка авторизации, корзины, оформления заказов, добавления и просмотра отзывов, подачи заявок на авторство, а также административная панель для модерирования контента и управления пользователями.

Разработанное web-приложение охватывает ключевые пользовательские сценарии: просмотр каталога книг, фильтрация и поиск, добавление книг в корзину и оформление заказа, а также работа с отзывами и авторскими публикациями. Пользователям предоставляется простой и доступный инструмент для взаимодействия с платформой, а авторами — удобный канал для публикации собственных произведений.

С технической точки зрения приложение обладает модульной архитектурой, что обеспечивает лёгкость его масштабирования и поддержки. Проведённое тестирование подтвердило стабильную работу системы, а соблюдение принципов охраны труда при разработке обеспечило комфорт и безопасность как для разработчика, так и для конечного пользователя.

В дальнейшем возможна доработка проекта с учётом следующих направлений:

- интеграция платёжных систем для онлайн-оплаты заказов;
- подключение облачного хранилища для размещения текстов книг;

- расширение системы рекомендаций на основе предпочтений пользователей;
- внедрение панели аналитики для авторов и администраторов;
- повышение уровня информационной безопасности и переход на защищённое соединение HTTPS.

Разработка данного web-приложения не только позволила закрепить полученные в процессе обучения знания, но и сформировала полноценный программный продукт, который может быть использован в качестве основы для коммерческого проекта или платформы поддержки начинающих авторов.

					ПГУ 1.090301.10.001 ПЗ	Лист
						44
Изм.	Лист	№ докум.	Подпись	Дата		

Список использованных источников

1. ЛитРес. – Режим доступа: <https://www.litres.ru/> (дата обращения: 02.05.2025).
2. Лабиринт. – Режим доступа: <https://www.labirint.ru/> (дата обращения: 02.05.2025).
3. Яндекс Книги. – Режим доступа: <https://books.yandex.ru/> (дата обращения: 02.05.2025).
4. Хабр: «Мега-Учебник Flask» / Алексей Мерсер. – Режим доступа: <https://habr.com/ru/articles/817369/> дата обращения: 10.05.2025).
5. SkillFactory «REST API – что это такое». – Режим доступа: <https://blog.skillfactory.ru/glossary/rest-api/> (дата обращения: 10.05.2025).
6. GitVerse.ru: «Что такое MVVM: шаблон, как работает, преимущества и примеры». – Режим доступа: <https://gitverse.ru/blog/articles/development/116-cto-takoe-mvvm-shablon-kak-rabotaet-preimushstva-i-primery-ispolzovaniya> (дата обращения: 13.05.2025).
7. «React: современные шаблоны для разработки приложений» / Дон Гриффитс и Дэвид Гриффитс; пер. на русский. – М.: Питер, 2023. – 480 с. (дата обращения 13.05.2025).
8. Oilman, A.M. Flask. Введение и разработка веб-приложений на Python (перевод). – PDF-версия. – 2019. – 300 с. (дата обращения 14.05.2025).
9. u.wikipedia.org: «Локализация программного обеспечения». – Режим доступа: https://ru.wikipedia.org/wiki/Локализация_программного_обеспечения (дата обращения 15.05.2025).
10. «React быстро. 2-е междунар. изд.» / А. Мардан. – М.: Питер, 2025. – 800 с. (дата обращения 17.05.2025).

Приложение А

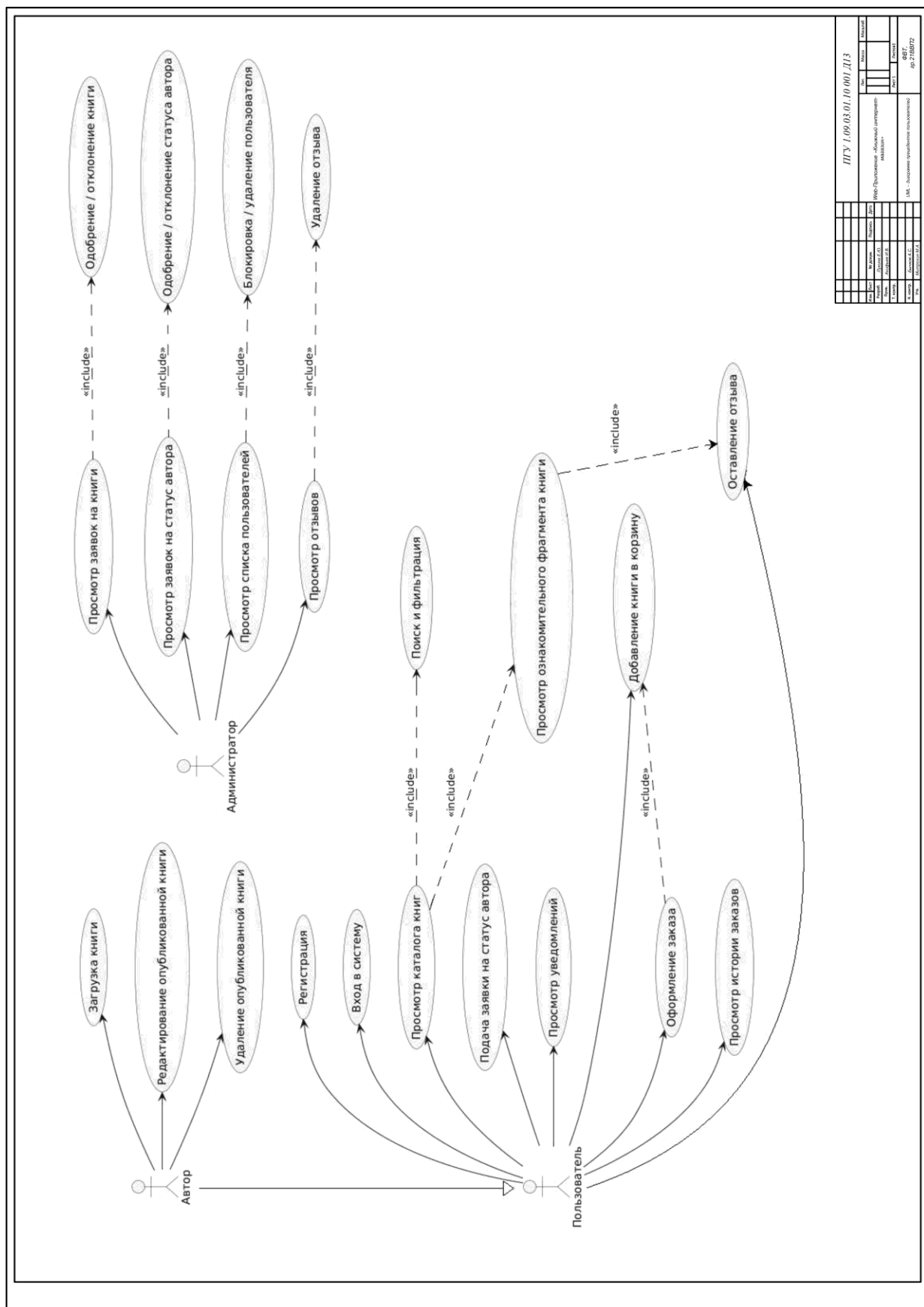
(Обязательное)

Web-приложение «Книжный интернет-магазин»

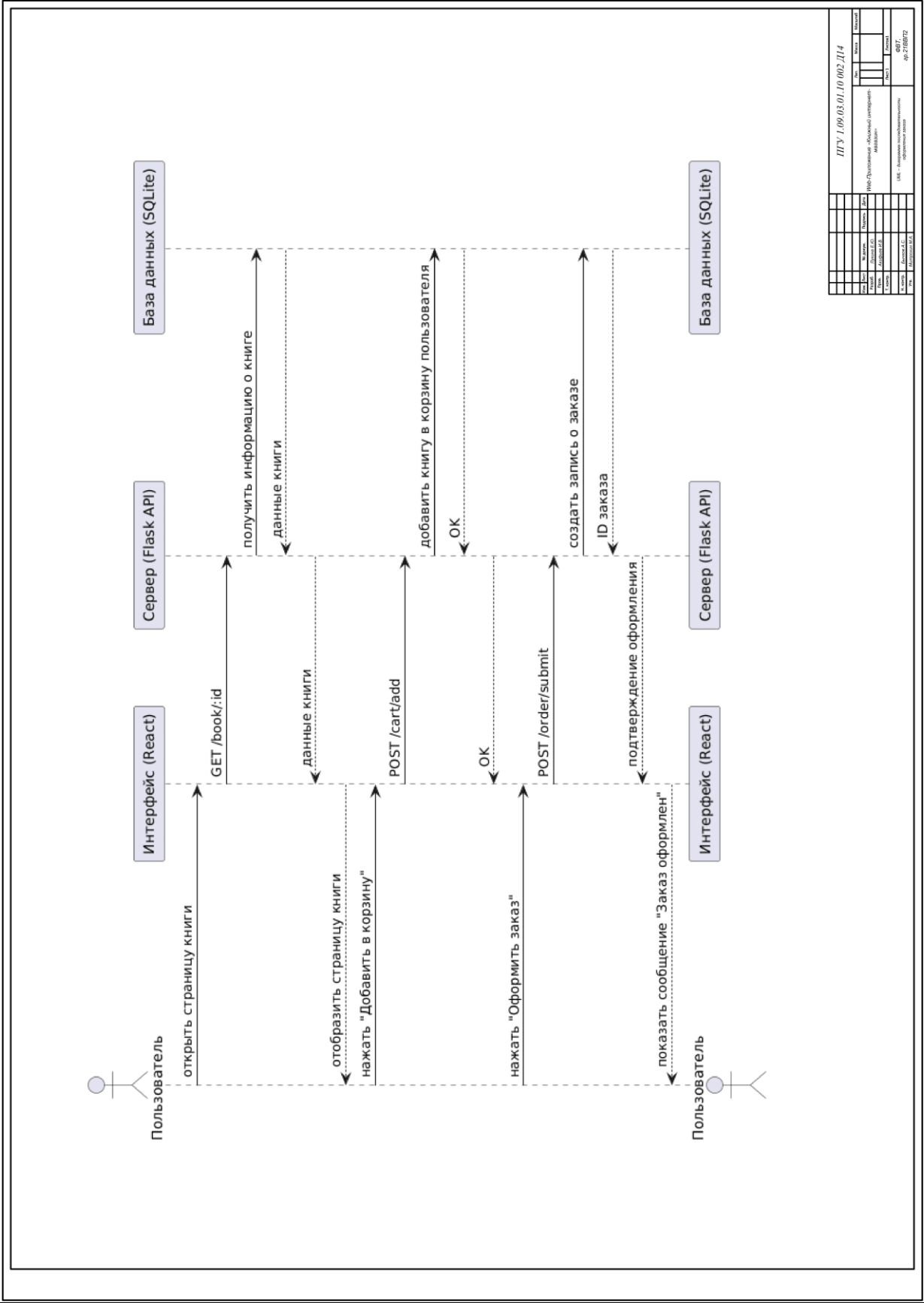
UML-диаграммы

					ПГУ 1.090301.10.001 ПЗ	Лист
						46
Изм.	Лист	№ докум.	Подпись	Дата		

Приложение А.1 – UML – диаграмма прецедентов пользователей

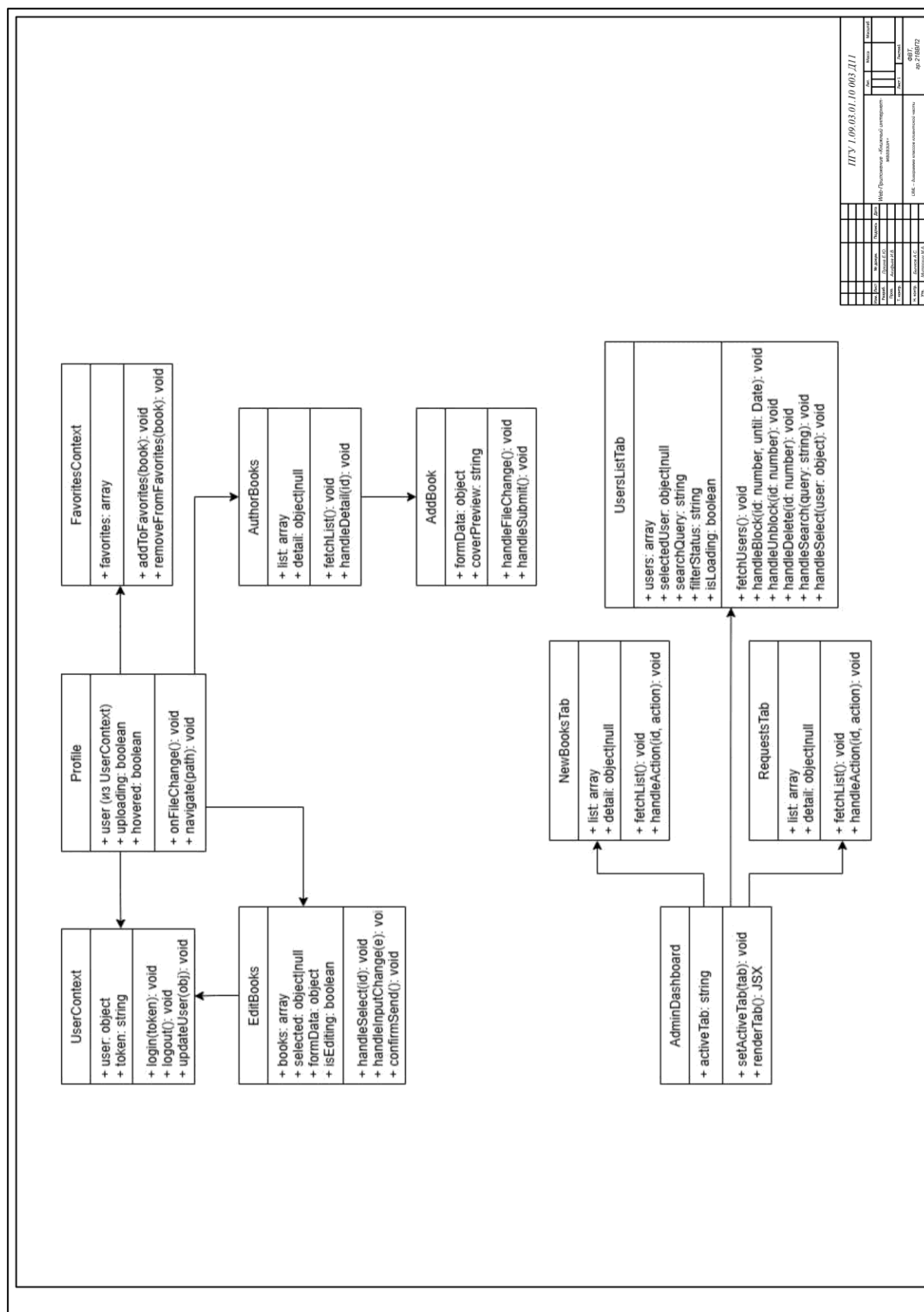


Приложение А.2 – UML – диаграмма последовательности оформления заказа



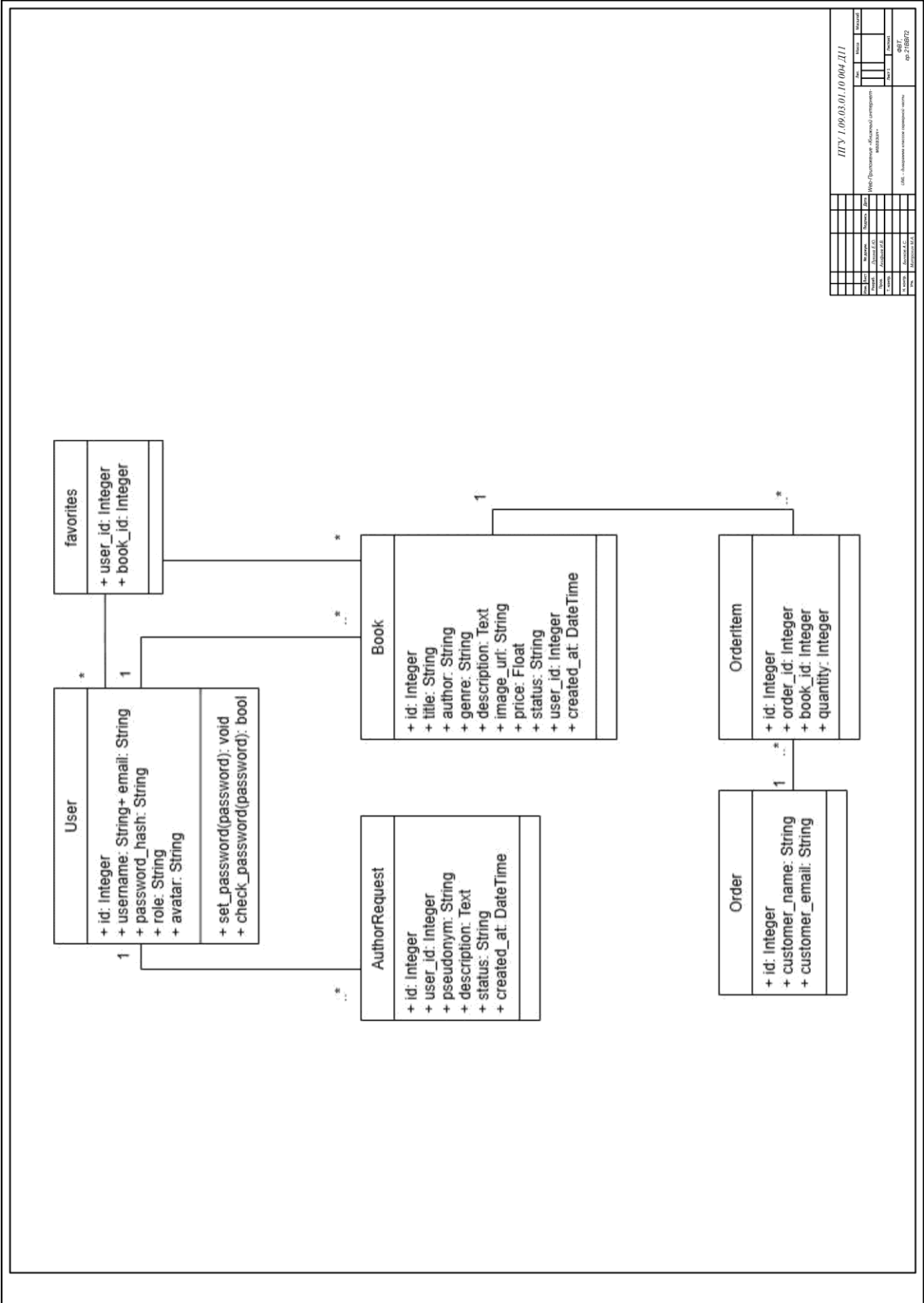
ПГУ 1.09.03.01.10.002.Л14									
Имя	Фамилия	Отчество	Дата	Время	Место	Страна	Город	Улица	Почта
Иван	Иванов	Иванович	10.09.2014	10:00	Москва	Россия	Москва	ул. Пушкина	ivanov@yandex.ru
Имя	Фамилия	Отчество	Дата	Время	Место	Страна	Город	Улица	Почта
Иван	Иванов	Иванович	10.09.2014	10:00	Москва	Россия	Москва	ул. Пушкина	ivanov@yandex.ru
Имя	Фамилия	Отчество	Дата	Время	Место	Страна	Город	Улица	Почта
Иван	Иванов	Иванович	10.09.2014	10:00	Москва	Россия	Москва	ул. Пушкина	ivanov@yandex.ru
Имя	Фамилия	Отчество	Дата	Время	Место	Страна	Город	Улица	Почта
Иван	Иванов	Иванович	10.09.2014	10:00	Москва	Россия	Москва	ул. Пушкина	ivanov@yandex.ru
Имя	Фамилия	Отчество	Дата	Время	Место	Страна	Город	Улица	Почта
Иван	Иванов	Иванович	10.09.2014	10:00	Москва	Россия	Москва	ул. Пушкина	ivanov@yandex.ru

Приложение А.3 – UML – диаграмма классов клиентской части



ПГУ 1.09.03.01.10.003.111									
Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия	Имя	Фамилия
Место	Место	Место	Место	Место	Место	Место	Место	Место	Место
Дата	Дата	Дата	Дата	Дата	Дата	Дата	Дата	Дата	Дата
Подпись	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись	Подпись
Лист 49									

Приложение А.4 – UML – диаграмма классов серверной части



Приложение Б

(Обязательное)

Web-приложение «Книжный интернет-магазин»

Листинг программы

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		51

Файл «app.py»

```
import os
import time
from flask import Flask, request, jsonify, send_from_directory
from flask_cors import CORS
from flask_sqlalchemy import SQLAlchemy
from flask_jwt_extended import (
    JWTManager, create_access_token,
    jwt_required, get_jwt_identity, get_jwt
)
from werkzeug.utils import secure_filename
from werkzeug.security import generate_password_hash,
check_password_hash
from datetime import datetime

def allowed_file(filename, allowed_exts):
    ext = filename.rsplit('.', 1)[1].lower() if '.' in filename else ''
    return ext in allowed_exts

from models import db, Book, Order, OrderItem, User,
AuthorRequest, Pseudonym

# — Конфигурация приложения —

app = Flask(__name__)
app.config.update({
    'JWT_SECRET_KEY': '12345',
    'SQLALCHEMY_DATABASE_URI': 'sqlite:///books.db',
    'SQLALCHEMY_TRACK_MODIFICATIONS': False
})

# Папки для загрузок
BASE_DIR = os.path.dirname(__file__)
UPLOAD_AVATAR_FOLDER = os.path.join(BASE_DIR, 'static', 'avatars')
UPLOAD_COVER_FOLDER = os.path.join(BASE_DIR, 'static', 'covers')
UPLOAD_FRAGMENT_FOLDER = os.path.join(BASE_DIR, 'static',
    'fragments')
UPLOAD_BOOK_FOLDER = os.path.join(BASE_DIR, 'static', 'books')

# Разрешённые расширения
ALLOWED_COVER_EXT = {'png', 'jpg', 'jpeg'}
ALLOWED_BOOK_EXT = {'epub', 'fb2', 'mobi', 'pdf', 'doc', 'docx'}

app.config['UPLOAD_AVATAR_FOLDER'] = UPLOAD_AVATAR_FOLDER
app.config['UPLOAD_COVER_FOLDER'] = UPLOAD_COVER_FOLDER
app.config['UPLOAD_FRAGMENT_FOLDER'] = UPLOAD_FRAGMENT_FOLDER
app.config['UPLOAD_BOOK_FOLDER'] = UPLOAD_BOOK_FOLDER

# CORS: разрешаем запросы с фронта http://localhost:3000
CORS(
    app,
    supports_credentials=True,
```

					ПГУ 1.090301.10.001 ПЗ	Лист
						52
Изм.	Лист	№ докум.	Подпись	Дата		

```

resources={r"/*": {"origins": "http://localhost:3000"}},
allow_headers=["Content-Type", "Authorization"],
methods=["GET", "POST", "PUT", "DELETE", "OPTIONS"]
)

# Инициализация базы данных и JWT
db.init_app(app)
with app.app_context():
    db.create_all()

jwt = JWTManager(app)

# Создаём папки, если их нет
for folder in [UPLOAD_AVATAR_FOLDER, UPLOAD_COVER_FOLDER,
UPLOAD_FRAGMENT_FOLDER, UPLOAD_BOOK_FOLDER]:
    os.makedirs(folder, exist_ok=True)

# ——— Вспомогательные функции ———

def allowed_cover(filename):
    """
    Проверяет расширение файла обложки.
    """
    ext = filename.rsplit('.', 1)[1].lower() if '.' in filename else ''
    return ext in ALLOWED_COVER_EXT

def create_admin():
    """
    Создаёт администратора с логином 'Admin' и паролем '123456789',
    если он ещё не существует.
    """
    admin = User.query.filter_by(username='Admin').first()
    if not admin:
        admin = User(
            username='Admin',
            email='admin@example.com',
            password_hash=generate_password_hash('123456789'),
            role='admin'
        )
        db.session.add(admin)
        db.session.commit()
    print('✅ Администратор создан: Admin / 123456789')
    else:
        print('❗ Администратор уже существует')

# ——— Обработчики JWT-ошибок ———

@jwt.unauthorized_loader
def missing_token_callback(err):
    return jsonify({"error": "Отсутствует токен"}), 401

@jwt.invalid_token_loader

```

```

def invalid_token_callback(err):
    return jsonify({"error": "Неверный токен"}), 422

@jwt.expired_token_loader
def expired_token_callback(jwt_header, jwt_payload):
    return jsonify({"error": "Токен истёк"}), 401

# — Статические файлы —

@app.route('/uploads/<filename>')
def uploaded_avatar(filename):
    return send_from_directory(app.config['UPLOAD_AVATAR_FOLDER'],
                               filename)

@app.route('/static/covers/<path:filename>')
def serve_cover(filename):
    return send_from_directory(app.config['UPLOAD_COVER_FOLDER'],
                               filename)

@app.route('/static/fragments/<path:filename>')
def serve_fragment(filename):
    return send_from_directory(app.config['UPLOAD_FRAGMENT_FOLDER'],
                               filename)

@app.route('/static/books/<path:filename>')
def serve_book(filename):
    return send_from_directory(app.config['UPLOAD_BOOK_FOLDER'],
                               filename)

# — Проверка доступности username —

@app.route('/check-username', methods=['POST'])
def check_username():
    data = request.get_json(silent=True) or {}
    username = data.get('username', '').strip()
    if not username:
        return jsonify({'available': False})
    exists = User.query.filter_by(username=username).first()
    return jsonify({'available': not bool(exists)})

# — Регистрация и логин —

@app.route('/register', methods=['POST'])
def register():
    data = request.get_json(silent=True) or {}
    u = data.get('username', '').strip()
    e = data.get('email', '').strip()
    p = data.get('password', '')
    if not all([u, e, p]):
        return jsonify(error="Не все поля заполнены"), 400
    if User.query.filter_by(username=u).first():
        return jsonify(error="Имя пользователя уже занято"), 409
    if User.query.filter_by(email=e).first():

```

					ПГУ 1.090301.10.001 ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		54

```

return jsonify(error="Пользователь с таким email уже существует"),
409
user = User(username=u, email=e)
user.set_password(p)
db.session.add(user)
db.session.commit()
return jsonify(message="Регистрация успешна")

@app.route('/login', methods=['POST'])
def login():
    data = request.get_json() or {}
    u = User.query.filter_by(username=data.get('username')).first()
    if u and u.check_password(data.get('password')):
        additional = {
            'username': u.username,
            'role': u.role,
            'email': u.email or '',
            'phone': u.phone or '',
            'avatar': u.avatar or ''
        }
        token = create_access_token(
            identity=str(u.id),
            additional_claims=additional
        )
        return jsonify(token=token), 200
    return jsonify(error='Неверные данные'), 401

# — Маршруты книг —

@app.route('/books')
def get_books():
    """
    Возвращает список всех книг (можно позже фильтровать по
    'status=approved').
    """
    books = Book.query.all()
    return jsonify([
        {
            'id': b.id,
            'title': b.title,
            'author': b.author,
            'genre': b.genre,
            'description': b.description,
            'price': b.price,
            'image_url': b.image_url,
            'status': b.status
        } for b in books])

@app.route('/add-test-books')
def add_test_books():
    """
    Добавляет пару тестовых книг (без обложки).
    """

```

```

b1 = Book(title="Мастер и Маргарита", author="Булгаков",
genre="Роман",
price=499.0, description="Роман о дьяволе в Москве.",
image_url="", status='approved')
b2 = Book(title="Преступление и наказание", author="Достоевский",
genre="Классика",
price=399.0, description="Классика русской литературы.",
image_url="", status='approved')
db.session.add_all([b1,b2])
db.session.commit()
return jsonify(message="Тестовые книги добавлены")

@app.route('/book/<int:book_id>')
def get_book(book_id):
    """
    Детали конкретной книги (по id).
    """
    b = Book.query.get(book_id)
    if not b:
        return jsonify(error="Книга не найдена"), 404
    return jsonify({
        'id': b.id,
        'title': b.title,
        'author': b.author,
        'genre': b.genre,
        'description': b.description,
        'price': b.price,
        'image_url': b.image_url,
        'status': b.status
    })

# — Оформление заказа —

@app.route('/order', methods=['POST'])
@jwt_required()
def create_order():
    current = get_jwt_identity()
    data = request.get_json(silent=True) or {}
    name = data.get('name', '').strip()
    email = data.get('email', '').strip()
    items = data.get('items', [])
    if not all([name, email, items]):
        return jsonify(error="Недостаточно данных"), 400
    o = Order(customer_name=name, customer_email=email)
    db.session.add(o)
    db.session.commit()
    for it in items:
        bid = it.get('id')
        if Book.query.get(bid):
            db.session.add(OrderItem(order_id=o.id, book_id=bid, quantity=1))
    db.session.commit()
    return jsonify(message="Заказ успешно создан")

```

— Обновление профиля —

```
@app.route('/update-profile', methods=['POST'])
@jwt_required()
def update_profile():
    user_id = int(get_jwt_identity())
    u = User.query.get(user_id)
    data = request.get_json(force=True) or {}
    if not u.check_password(data.get('current_password', '')):
        return jsonify(error='Неверный текущий пароль'), 403

    if data.get('username') and data['username'] != u.username:
        if User.query.filter_by(username=data['username']).first():
            return jsonify(error='Имя пользователя уже занято'), 409
        u.username = data['username']
    if data.get('email'): u.email = data['email']
    if data.get('phone'): u.phone = data['phone']
    if data.get('new_password'): u.set_password(data['new_password'])

    db.session.commit()

    additional = {
        'username': u.username,
        'role': u.role,
        'email': u.email or '',
        'phone': u.phone or '',
        'avatar': u.avatar or ''
    }
    new_token = create_access_token(identity=str(u.id),
                                    additional_claims=additional)
    return jsonify(message='Профиль обновлён успешно',
                    token=new_token), 200
```

— Загрузка аватара —

```
@app.route('/upload-avatar', methods=['POST'])
@jwt_required()
def upload_avatar():
    user_id = int(get_jwt_identity())
    u = User.query.get(user_id)
    if 'avatar' not in request.files:
        return jsonify(error='No file'), 400
    file = request.files['avatar']
    if file.filename == '' or '.' not in file.filename:
        return jsonify(error='Invalid filename'), 400
    ext = file.filename.rsplit('.', 1)[1].lower()
    if ext not in ALLOWED_COVER_EXT:
        return jsonify(error='Invalid extension'), 400

    os.makedirs(app.config['UPLOAD_AVATAR_FOLDER'], exist_ok=True)
    fname = secure_filename(f"{u.id}_{int(time.time())}.{ext}")
    path = os.path.join(app.config['UPLOAD_AVATAR_FOLDER'], fname)
    file.save(path)
```

```

u.avatar = f"/uploads/{fname}"
db.session.commit()

additional = {
    'username': u.username,
    'role':     u.role,
    'email':    u.email or '',
    'phone':    u.phone or '',
    'avatar':   u.avatar or ''
}
new_token = create_access_token(identity=str(u.id),
additional_claims=additional)
return jsonify(avatar_url=u.avatar, token=new_token), 200

# — Избранные книги (favorites) —

@app.route('/favorites', methods=['GET'])
@jwt_required()
def get_favorites():
    user_id = get_jwt_identity()
    u = User.query.get(int(user_id))
    favs = u.favorites.all()
    return jsonify([
        {
            'id':          b.id,
            'title':       b.title,
            'author':      b.author,
            'genre':       b.genre,
            'description': b.description,
            'price':       b.price,
            'image_url':   b.image_url
        } for b in favs])

@app.route('/favorites/<int:book_id>', methods=['POST'])
@jwt_required()
def toggle_favorite(book_id):
    user_id = get_jwt_identity()
    u = User.query.get(int(user_id))
    b = Book.query.get(book_id)
    if not b:
        return jsonify(error="Книга не найдена"), 404
    if u.favorites.filter_by(id=book_id).first():
        u.favorites.remove(b)
        action = 'removed'
    else:
        u.favorites.append(b)
        action = 'added'
    db.session.commit()
    return jsonify(message=f'favorite_{action}', book_id=book_id), 200

# — Заявки на статус «Автор» —

@app.route('/request-author', methods=['POST'])

```

```

@jwt_required()
def request_author():
    user_id = int(get_jwt_identity())
    data = request.get_json(silent=True) or {}
    pseudonym = data.get('pseudonym', '').strip()
    description = data.get('description', '').strip()
    if not pseudonym or not description:
        return jsonify(error='Заполните все поля'), 400
    if AuthorRequest.query.filter_by(user_id=user_id,
        status='pending').first():
        return jsonify(error='У вас уже есть заявка на рассмотрение'), 400
    req = AuthorRequest(user_id=user_id, pseudonym=pseudonym,
        description=description)
    db.session.add(req)
    db.session.commit()
    return jsonify(message='Заявка отправлена'), 200

@app.route('/request-author', methods=['GET'])
@jwt_required()
def get_my_request():
    user_id = int(get_jwt_identity())
    r = AuthorRequest.query.filter_by(user_id=user_id,
        status='pending') \
        .order_by(AuthorRequest.created_at.desc()) \
        .first()
    if not r:
        return jsonify({'exists': False}), 200
    return jsonify({
        'exists': True,
        'id': r.id,
        'pseudonym': r.pseudonym,
        'description': r.description,
        'status': r.status,
        'created_at': r.created_at.isoformat()
    }), 200

# — Панель администратора: заявки на авторство —

@app.route('/admin/requests')
@jwt_required()
def get_author_requests():
    claims = get_jwt()
    if claims.get('role') != 'admin':
        return jsonify(error='Доступ запрещён'), 403
    reqs =
    AuthorRequest.query.order_by(AuthorRequest.created_at.desc()).all()
    return jsonify([
        {
            'id': r.id,
            'username': r.user.username,
            'pseudonym': r.pseudonym,
            'description': r.description,
            'status': r.status,

```

```

'created_at': r.created_at.isoformat()
} for r in reqs]), 200

@app.route('/admin/requests/<int:req_id>')
@jwt_required()
def get_author_request(req_id):
    claims = get_jwt()
    if claims.get('role') != 'admin':
        return jsonify(error='Доступ запрещён'), 403
    r = AuthorRequest.query.get_or_404(req_id)
    return jsonify({
        'id': r.id,
        'user_id': r.user_id,
        'username': r.user.username,
        'pseudonym': r.pseudonym,
        'description': r.description,
        'status': r.status,
        'created_at': r.created_at.isoformat()
    }), 200

@app.route('/admin/requests/<int:req_id>/approve',
methods=['POST'])
@jwt_required()
def approve_request(req_id):
    claims = get_jwt()
    if claims.get('role') != 'admin':
        return jsonify(error='Доступ запрещён'), 403

    r = AuthorRequest.query.get_or_404(req_id)
    r.status = 'approved'

    user = User.query.get(r.user_id)
    user.role = 'author'

    # Сохраняем псевдоним из заявки, если его ещё нет
    if r.pseudonym:
        existing = Pseudonym.query.filter_by(user_id=r.user_id,
name=r.pseudonym).first()
        if not existing:
            p = Pseudonym(user_id=r.user_id, name=r.pseudonym)
            db.session.add(p)

    db.session.commit()
    return jsonify(message='Принято'), 200

@app.route('/admin/requests/<int:req_id>/reject',
methods=['POST'])
@jwt_required()
def reject_request(req_id):
    claims = get_jwt()
    if claims.get('role') != 'admin':
        return jsonify(error='Доступ запрещён'), 403
    r = AuthorRequest.query.get_or_404(req_id)

```

```

r.status = 'rejected'
db.session.commit()
return jsonify(message='Отклонено'), 200

# — Панель администратора: заявки на публикацию книг —

@app.route('/admin/book-requests', methods=['GET'])
@jwt_required()
def get_book_requests():
    claims = get_jwt()
    if claims.get('role') != 'admin':
        return jsonify(error='Доступ запрещён'), 403

    # Убрали фильтр по status='pending'
    all_books = Book.query.order_by(Book.created_at.desc()).all()

    result = []
    for b in all_books:
        author_user = User.query.get(b.user_id)
        result.append({
            'id': b.id,
            'title': b.title,
            'author': b.author,
            'image_url': b.image_url,
            'publication_year': b.publication_year,
            'created_at': b.created_at.isoformat(),
            'status': b.status
        })
    return jsonify(result), 200

@app.route('/admin/book-requests/<int:book_id>', methods=['GET'])
@jwt_required()
def get_book_request_detail(book_id):
    claims = get_jwt()
    if claims.get('role') != 'admin':
        return jsonify(error='Доступ запрещён'), 403

    b = Book.query.get_or_404(book_id)
    # Даже если статус != pending, позволяем просмотреть детали
    author_user = User.query.get(b.user_id)
    return jsonify({
        'id': b.id,
        'title': b.title,
        'author': b.author,
        'genre': b.genre or '',
        'description': b.description,
        'price': b.price,
        'image_url': b.image_url,
        'fragment_url': b.fragment_url or '',
        'book_url': b.book_url or '',
        'publication_year': b.publication_year,
        'status': b.status,
        'created_at': b.created_at.isoformat()
    })

```

```

    }), 200

@app.route('/admin/book-requests/<int:book_id>/approve',
methods=['POST'])
@jwt_required()
def approve_book_request(book_id):
    claims = get_jwt()
    if claims.get('role') != 'admin':
        return jsonify(error='Доступ запрещён'), 403

    b = Book.query.get_or_404(book_id)
    if b.status != 'pending':
        return jsonify(error='Эту заявку уже обработали'), 400

    b.status = 'approved'
    db.session.commit()
    return jsonify(message='Книга одобрена'), 200

@app.route('/admin/book-requests/<int:book_id>/reject',
methods=['POST'])
@jwt_required()
def reject_book_request(book_id):
    claims = get_jwt()
    if claims.get('role') != 'admin':
        return jsonify(error='Доступ запрещён'), 403

    b = Book.query.get_or_404(book_id)
    if b.status != 'pending':
        return jsonify(error='Эту заявку уже обработали'), 400

    b.status = 'rejected'
    db.session.commit()
    return jsonify(message='Книга отклонена'), 200

# — Маршруты автора: список, создание и редактирование книг —

@app.route('/author/books')
@jwt_required()
def get_my_books():
    """
    Возвращает все книги пользователя (любого статуса).
    """
    user_id = int(get_jwt_identity())
    books = Book.query.filter_by(user_id=user_id).all()
    return jsonify([
        {
            'id': b.id,
            'title': b.title,
            'author': b.author,
            'genre': b.genre,
            'description': b.description,
            'price': b.price,
            'image_url': b.image_url,
            'fragment_url': b.fragment_url,
        }
    ])

```

```

'book_url':          b.book_url,
'publication_year':  b.publication_year,
'status':            b.status,
'created_at':        b.created_at.isoformat()
} for b in books]), 200

@app.route('/author/books', methods=['POST'])
@jwt_required()
def create_book():
    user_id = int(get_jwt_identity())
    u = User.query.get(user_id)
    if not u or u.role != 'author':
        return jsonify(error='Доступ запрещён'), 403

    # выбор псевдонима
    pseud_id = request.form.get('pseudonym_id')
    if not pseud_id:
        return jsonify(error='Не выбран псевдоним'), 400
    pseud = Pseudonym.query.get(int(pseud_id))
    if not pseud or pseud.user_id != user_id:
        return jsonify(error='Неверный псевдоним'), 400
    author_name = pseud.name

    # проверяем файлы
    if 'cover' not in request.files:
        return jsonify(error='Не прикреплено поле cover'), 400
    cover = request.files['cover']
    if cover.filename == '' or not allowed_file(cover.filename,
        ALLOWED_COVER_EXT):
        return jsonify(error='Неверное расширение обложки'), 400

```

Файл «models.py»

```

from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash,
check_password_hash
from datetime import datetime

db = SQLAlchemy()

# таблица many-to-many для «отложенных»
favorites_table = db.Table(
    'favorites',
    db.Column('user_id', db.Integer, db.ForeignKey('user.id'),
primary_key=True),
    db.Column('book_id', db.Integer, db.ForeignKey('book.id'),
primary_key=True)
)

class Book(db.Model):
    __tablename__ = 'book'
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(200), nullable=False)
    author = db.Column(db.String(100), nullable=False)
    genre = db.Column(db.String(100), nullable=True)

```

```

        description      = db.Column(db.Text, nullable=False)
        price             = db.Column(db.Float, nullable=True)
        image_url         = db.Column(db.String, nullable=True)
        fragment_url      = db.Column(db.String(500), nullable=True)    #
        путь к ознакомительному фрагменту
        book_url          = db.Column(db.String(500), nullable=False)    #
        путь к полному файлу книги
        status            = db.Column(db.String(20), default='pending')#
        pending, approved, rejected
        user_id           = db.Column(db.Integer,
db.ForeignKey('user.id'), nullable=True)
        created_at        = db.Column(db.DateTime,
default=datetime.utcnow)
        publication_year= db.Column(db.Integer, nullable=True)          #
        год издания книги

        user = db.relationship('User', backref='books')

class Order(db.Model):
    __tablename__ = 'order'
    id             = db.Column(db.Integer, primary_key=True)
    customer_name  = db.Column(db.String(100), nullable=False)
    customer_email = db.Column(db.String(120), nullable=False)
    items          = db.relationship('OrderItem', backref='order',
lazy=True)

class OrderItem(db.Model):
    __tablename__ = 'order_item'
    id            = db.Column(db.Integer, primary_key=True)
    order_id      = db.Column(db.Integer, db.ForeignKey('order.id'),
nullable=False)
    book_id       = db.Column(db.Integer, db.ForeignKey('book.id'),
nullable=False)
    quantity      = db.Column(db.Integer, default=1)

class User(db.Model):
    __tablename__ = 'user'
    id            = db.Column(db.Integer, primary_key=True)
    username      = db.Column(db.String(64), unique=True,
nullable=False)
    email         = db.Column(db.String(120), unique=True,
nullable=False)
    phone         = db.Column(db.String(20), nullable=True)
    password_hash = db.Column(db.String(128), nullable=False)
    role          = db.Column(db.String(20), default='user')    #
    user / author / admin
    avatar        = db.Column(db.String(200), nullable=True)

    # отношение к «отложенным»
    favorites = db.relationship(
        'Book',
        secondary=favorites_table,
        backref=db.backref('favorited_by', lazy='dynamic'),

```

```

        lazy='dynamic'
    )

    def set_password(self, password):
        self.password_hash = generate_password_hash(password)

    def check_password(self, password):
        return check_password_hash(self.password_hash, password)

class AuthorRequest(db.Model):
    __tablename__ = 'author_requests'
    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'),
        nullable=False)
    pseudonym = db.Column(db.String(100), nullable=False)
    description = db.Column(db.Text, nullable=False)
    status = db.Column(db.String(20), default='pending') #
    pending, approved, rejected
    created_at = db.Column(db.DateTime, default=datetime.utcnow)

    user = db.relationship('User', backref='author_requests')

class Pseudonym(db.Model):
    __tablename__ = 'pseudonym'
    __table_args__ = (db.UniqueConstraint('user_id', 'name',
        name='_user_pseudonym_uc'),)

    id = db.Column(db.Integer, primary_key=True)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'),
        nullable=False)
    name = db.Column(db.String(100), nullable=False)
    created_at = db.Column(db.DateTime, default=datetime.utcnow)

    user = db.relationship('User',
        backref=db.backref('pseudonyms', lazy='dynamic'))

```

Файл «Profile.jsx»

```

import React, { useContext, useRef, useState } from 'react';
import { UserContext } from '../context/UserContext';
import { FaQuestionCircle, FaPlus } from 'react-icons/fa';
import { useNavigate } from 'react-router-dom';
import './Profile.css';

export default function Profile() {
    const { user, token, login, updateUser } =
    useContext(UserContext);
    const [hovered, setHovered] = useState(false);
    const [uploading, setUploading] = useState(false);
    const inputRef = useRef();
    const navigate = useNavigate();

    if (!user) return <div className="container mt-4">Вы не
    авторизованы</div>;

```

```

// Обработчик выбора файла
const onFileChange = async (e) => {
  const file = e.target.files[0];
  if (!file) return;

  const fd = new FormData();
  fd.append('avatar', file);
  setUploading(true);

  const res = await fetch('http://localhost:5000/upload-avatar', {
    method: 'POST',
    mode: 'cors', // разрешаем CORS
    credentials: 'include', // на всякий случай, если будут
куки
    headers: {
      Authorization: `Bearer ${token}`
    },
    body: fd
  });

  setUploading(false);

  if (!res.ok) {
    const err = await res.json().catch(() => ({}));
    alert(err.error || 'Ошибка загрузки');
    return;
  }

  const data = await res.json();
  // обновляем токен и URL аватарки
  login(data.token);
  const fullUrl = `http://localhost:5000${data.avatar_url}`;
  updateUser({ avatar: fullUrl });
};

const backend = 'http://localhost:5000';
const avatarSrc = user.avatar
  ? (user.avatar.startsWith('http') ? user.avatar :
`${backend}${user.avatar}`)
  : 'https://via.placeholder.com/100';

return (
  <div className="container mt-4">
    <div className="d-flex align-items-center mb-4">
      <div
        className="avatar-container"
        onMouseEnter={() => setHovered(true)}
        onMouseLeave={() => setHovered(false)}
        onClick={() => inputRef.current.click()}
      >
        <img src={avatarSrc} alt="" />

        {(hovered || uploading) && (

```

```

        <>
        <div className="avatar-overlay">
            {uploading ? '...' : <FaPlus size={24} />}
        </div>
        <div className="avatar-tooltip">
            {uploading ? 'Загрузка...' : 'Изменить фото
профиля'}
        </div>
    </>
)}

<input
    type="file"
    accept=".jpg,.jpeg,.png"
    className="d-none"
    ref={inputRef}
    onChange={onFileChange}
/>
</div>

<div className="ms-4">
    <h4 className="mb-1">{user.username}</h4>
    <p className="text-muted mb-2">
        {{
            user: 'Пользователь',
            author: 'Автор',
            admin: 'Администратор'
        }}[user.role]}
    </p>
    {user.role === 'user' && (
        <div className="d-flex align-items-center">
            <button
                className="btn btn-outline-primary btn-sm"
                onClick={() => navigate('/request-author')}
            >
                Стать Автором
            </button>
            <FaQuestionCircle
                title="Нажмите, чтобы подать заявку на статус
автора и получить возможность публиковать свои книги после
модерации"
            />
        </div>
    )}
</div>
</div>

{user.role === 'admin' ? (
    <button
        className="btn btn-primary"
        onClick={() => navigate('/admin')}
    >
        Перейти в панель администратора

```

```

        </button>
      ) : (
        <>
          <hr />

          {user.role === 'author' && (
            <>
              <button
                className="btn btn-primary mb-3"
                onClick={() => navigate('/author/books')}
              >
                Загрузить свою книгу
              </button>

              <ul className="list-group mb-3">
                <li
                  className="list-group-item bg-light text-
primary"
                  onClick={() => navigate('/author/edit-books')}
                  style={{ cursor: 'pointer' }}
                >
                  Редактировать свои книги
                </li>
              </ul>
            </>
          )}

          <ul className="list-group">
            {[
              ['Мои книги', '/my-books'],
              ['Заказы', '/orders'],
              ['Отложенные', '/favorites'],
              ['Корзина', '/cart'],
              ['Настройки', '/settings']
            ].map(([title, url]) => (
              <li key={url} className="list-group-item">
                <a href={url} className="text-decoration-none">
                  {title}
                </a>
              </li>
            ))}
          </ul>
        </>
      )}
    </div>
  );
}

```

Файл «Register.jsx»

```

import React, { useState } from 'react';
import { Link } from 'react-router-dom';
import { FaCheckCircle, FaTimesCircle } from 'react-icons/fa';

export default function Register() {

```

```

    const [form, setForm] = useState({ username: '', email: '',
password: '' });
    const [message, setMessage] = useState(null);
    const [usernameValid, setUsernameValid] = useState(null);

    const handleChange = (e) => {
        setForm({ ...form, [e.target.name]: e.target.value });

        if (e.target.name === 'username') {
            checkUsername(e.target.value);
        }
    };

    const checkUsername = (name) => {
        fetch('http://localhost:5000/check-username', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify({ username: name })
        })
            .then(res => res.json())
            .then(data => setUsernameValid(data.available))
            .catch(() => setUsernameValid(null));
    };

    const handleSubmit = (e) => {
        e.preventDefault();

        fetch('http://localhost:5000/register', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(form)
        })
            .then(res => res.json())
            .then(data => setMessage(data.message || data.error))
            .catch(() => setMessage('Ошибка подключения'));
    };

    return (
        <div className="container mt-4">
            <h2 className="mb-4">Регистрация</h2>
            {message && <div className="alert alert-
info">{message}</div>}
            <form onSubmit={handleSubmit}>
                <div className="mb-3">
                    <label>Имя пользователя</label>
                    <div className="d-flex align-items-center">
                        <input
                            type="text"
                            name="username"
                            className="form-control me-2"
                            required
                            value={form.username}
                            onChange={handleChange}

```

```

        />
        {usernameValid === true && <FaCheckCircle
className="text-success" />}
        {usernameValid === false && <FaTimesCircle
className="text-danger" />}
      </div>
    </div>
    <div className="mb-3">
      <label>Email</label>
      <input type="email" name="email" className="form-
control" required onChange={handleChange} />
    </div>
    <div className="mb-3">
      <label>Пароль</label>
      <input type="password" name="password" className="form-
control" required onChange={handleChange} />
    </div>
    <button className="btn btn-primary"
type="submit">Зарегистрироваться</button>
  </form>

  <div className="mt-3">
    <span>Уже есть аккаунт? </span>
    <Link to="/login" className="btn btn-link">Войти</Link>
  </div>
</div>
);
}

```

Файл «Login.jsx»

```

import React, { useState, useContext } from 'react';
import { Link, useNavigate } from 'react-router-dom';
import { UserContext } from '../context/UserContext';

export default function Login() {
  const { login } = useContext(UserContext);
  const navigate = useNavigate();

  const [form, setForm] = useState({ username: '', password: '' });
  const [message, setMessage] = useState(null);

  const handleChange = e => {
    setForm({ ...form, [e.target.name]: e.target.value });
  };

  const handleSubmit = e => {
    e.preventDefault();

    fetch('http://localhost:5000/login', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },
      body: JSON.stringify(form)
    })
  }
}

```

```

.then(res => res.json())
.then(data => {
  if (data.token) {
    // save into context + localStorage
    login(data.token);
    setMessage('Вход выполнен');
    // redirect to home or profile
    navigate('/');
  } else {
    setMessage(data.error || 'Ошибка входа');
  }
})
.catch(() => setMessage('Ошибка подключения'));
};

return (
  <div className="container mt-4">
    <h2 className="mb-4">Вход</h2>
    {message && <div className="alert alert-
info">{message}</div>}
    <form onSubmit={handleSubmit}>
      <div className="mb-3">
        <label>Имя пользователя</label>
        <input
          type="text"
          name="username"
          className="form-control"
          value={form.username}
          onChange={handleChange}
          required
        />
      </div>
      <div className="mb-3">
        <label>Пароль</label>
        <input
          type="password"
          name="password"
          className="form-control"
          value={form.password}
          onChange={handleChange}
          required
        />
      </div>
      <button className="btn btn-success"
type="submit">Войти</button>
    </form>

    <div className="mt-3">
      <span>Нет аккаунта? </span>
      <Link to="/register" className="btn btn-
link">Зарегистрироваться</Link>
    </div>
  </div>

```

```
);
}
```

Файл «Settings.jsx»

```
import React, { useState, useContext, useEffect, useCallback }
from 'react';
import { UserContext } from '../context/UserContext';
import { FaCheckCircle, FaTimesCircle, FaEdit, FaTrash, FaPlus }
from 'react-icons/fa';

export default function Settings() {
  const { user, token, login } = useContext(UserContext);

  const [username, setUsername] = useState(user?.username || '');
  const [usernameValid, setUsernameValid] = useState(null);
  const [usernameMessage, setUsernameMessage] = useState('');

  const [phone, setPhone] = useState(user?.phone || '');
  const [phoneValid, setPhoneValid] = useState(null);

  const [email, setEmail] = useState(user?.email || '');
  const [emailValid, setEmailValid] = useState(null);

  const [newPassword, setNewPassword] = useState('');
  const [repeatPassword, setRepeatPassword] = useState('');
  const [passwordsMatch, setPasswordsMatch] = useState(null);

  const [modalOpen, setModalOpen] = useState(false);
  const [confirmPassword, setConfirmPassword] = useState('');
  const [passwordError, setPasswordError] = useState(null);
  const [successMessage, setSuccessMessage] = useState(null);

  const [pseudonyms, setPseudonyms] = useState([]);
  const [newPseudo, setNewPseudo] = useState('');
  const [editingPseudoId, setEditingPseudoId] = useState(null);
  const [editingPseudoName, setEditingPseudoName] = useState('');

  const fetchPseudonyms = useCallback(() => {
    fetch('http://localhost:5000/author/pseudonyms', {
      headers: { Authorization: `Bearer ${token}` }
    })
      .then(res => res.json())
      .then(data => setPseudonyms(Array.isArray(data) ? data :
[]))
      .catch(() => setPseudonyms([]));
  }, [token]);

  useEffect(() => {
    setUsername(user?.username || '');
    setPhone(user?.phone || '');
    setEmail(user?.email || '');
    if (user?.role === 'author') {
      fetchPseudonyms();
    }
  })
}
```

```

    }, [user, fetchPseudonyms]);

// Pseudonyms CRUD
function handleAddPseudo() {
    const name = newPseudo.trim();
    if (!name) return;
    fetch('http://localhost:5000/author/pseudonyms', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',
            Authorization: `Bearer ${token}`
        },
        body: JSON.stringify({ name })
    })
    .then(res => res.ok ? res.json() : Promise.reject())
    .then(() => {
        setNewPseudo('');
        fetchPseudonyms();
    })
    .catch(() => alert('Ошибка при добавлении псевдонима'));
}

function handleEditPseudo(id) {
    const name = editingPseudoName.trim();
    if (!name) return;
    fetch(`http://localhost:5000/author/pseudonyms/${id}`, {
        method: 'PUT',
        headers: {
            'Content-Type': 'application/json',
            Authorization: `Bearer ${token}`
        },
        body: JSON.stringify({ name })
    })
    .then(res => res.ok ? res.json() : Promise.reject())
    .then(() => {
        setEditingPseudoId(null);
        setEditingPseudoName('');
        fetchPseudonyms();
    })
    .catch(() => alert('Ошибка при редактировании псевдонима'));
}

function handleDeletePseudo(id) {
    if (!window.confirm('Удалить этот псевдоним?')) return;
    fetch(`http://localhost:5000/author/pseudonyms/${id}`, {
        method: 'DELETE',
        headers: { Authorization: `Bearer ${token}` }
    })
    .then(res => res.ok ? res.json() : Promise.reject())
    .then(() => fetchPseudonyms())
    .catch(() => alert('Ошибка при удалении псевдонима'));
}

```

```

// Форматирование телефона: вводим только цифры локальной части
const formatPhone = (input) => {
  let digits = input.replace(/\D/g, '').slice(0, 11);
  if (digits.startsWith('8') || digits.startsWith('7')) {
    digits = digits.slice(1);
  }
  digits = digits.slice(0, 10);
  if (!digits) return '';
  let result = '+7';
  if (digits.length > 0) result += ' ' + digits.slice(0, 3);
  if (digits.length >= 4) result += ' ' + digits.slice(3, 6);
  if (digits.length >= 7) result += '-' + digits.slice(6, 8);
  if (digits.length >= 9) result += '-' + digits.slice(8, 10);
  return result;
};

// Валидация телефона: пустое – нейтрально, иначе по шаблону
const validatePhone = (formatted) => {
  if (!formatted) {
    setPhoneValid(null);
    return;
  }
  const regex = /^\\+7 \\d{3} \\d{3}-\\d{2}-\\d{2}$/;
  setPhoneValid(regex.test(formatted));
};

// Валидация e-mail
const validateEmail = (mail) => {
  if (!mail) {
    setEmailValid(null);
    return;
  }
  const regex = /^[^\\s@]+@[^\\s@]+\\. [^\\s@]+$/;
  setEmailValid(regex.test(mail));
};

// Валидация username + проверка на сервере
const validateUsername = (name) => {
  const regex = /^[a-zA-Z0-9_]+$/;
  if (name.length > 20) {
    setUsernameValid(false);
    setUsernameMessage('Имя должно содержать не более 20 СИМВОЛОВ');
  } else if (!regex.test(name)) {
    setUsernameValid(false);
    setUsernameMessage('Допустимы только буквы, цифры и _');
  } else if (name === user.username) {
    setUsernameValid(true);
    setUsernameMessage('');
  } else {
    fetch('http://localhost:5000/check-username', {
      method: 'POST',
      headers: { 'Content-Type': 'application/json' },

```

```

        body: JSON.stringify({ username: name }),
    })
    .then(res => res.json())
    .then(data => {
        if (data.available) {
            setUsernameValid(true);
            setUsernameMessage('');
        } else {
            setUsernameValid(false);
            setUsernameMessage('Пользователь с таким именем уже
существует');
        }
    });
    }
};

// Подсветка совпадения паролей
useEffect(() => {
    if (!newPassword && !repeatPassword) {
        setPasswordsMatch(null);
    } else {
        setPasswordsMatch(newPassword === repeatPassword);
    }
}, [newPassword, repeatPassword]);

const openModal = (e) => {
    e.preventDefault();
    // Не открывать, если есть ошибки валидации
    if (
        usernameValid === false ||
        phoneValid === false ||
        emailValid === false ||
        passwordsMatch === false
    ) return;
    setModalOpen(true);
    setPasswordError(null);
};

const cancelModal = () => {
    setModalOpen(false);
    setConfirmPassword('');
    setPasswordError(null);
};

const submitUpdate = () => {
    if (!confirmPassword) {
        setPasswordError('Введите текущий пароль');
        return;
    }
    fetch('http://localhost:5000/update-profile', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json',

```

```

        Authorization: `Bearer ${token}`,
    },
    body: JSON.stringify({
        username,
        phone,
        email,
        current_password: confirmPassword,
        new_password: newPassword || null,
    }),
  ))
  .then(res => res.json())
  .then(data => {
    if (data.error) {
      setPasswordError(data.error);
    } else {
      if (data.token) {
        localStorage.setItem('token', data.token);
        login(data.token);
      }
      setSuccessMessage('Изменения успешно сохранены');
      setModalOpen(false);
    }
  });
};

if (!user) return <div className="container mt-4">Вы не
авторизованы</div>;

return (
  <div className="container mt-5 d-flex justify-content-center">
    <div style={{ width: '100%', maxWidth: '500px' }}>
      <h3 className="mb-4 text-center">Настройки профиля</h3>
      {successMessage && <div className="alert alert-
success">{successMessage}</div>}

      <form onSubmit={openModal}>
        {/* Username */}
        <div className="mb-3">
          <label>Имя пользователя</label>
          <div className="d-flex align-items-center">
            <input
              type="text"
              className="form-control me-2"
              value={username}
              onChange={e => {
                setUsername(e.target.value);
                validateUsername(e.target.value);
              }}
              maxLength={20}
              required
            />
            {usernameValid && <FaCheckCircle className="text-
success" />}

```

```

        {usernameValid === false && <FaTimesCircle
className="text-danger" />}
      </div>
      {usernameMessage && <small className="text-
danger">{usernameMessage}</small>}
    </div>

    {/* Phone */}
    <div className="mb-3">
      <label>Номер телефона</label>
      <div className="d-flex align-items-center">
        <input
          type="text"
          className="form-control me-2"
          value={phone}
          onChange={e => {
            const formatted = formatPhone(e.target.value);
            setPhone(formatted);
            validatePhone(formatted);
          }}
          placeholder="+7 XXX XXX-XX-XX"
        />
        {phoneValid && <FaCheckCircle className="text-
success" />}
        {phoneValid === false && <FaTimesCircle
className="text-danger" />}
      </div>
      {phoneValid === false && (
        <small className="text-danger">
          Введите номер в формате +7 XXX XXX-XX-XX
        </small>
      )}
    </div>

    {/* Email */}
    <div className="mb-3">
      <label>Электронная почта</label>
      <div className="d-flex align-items-center">
        <input
          type="email"
          className="form-control me-2"
          value={email}
          onChange={e => {
            setEmail(e.target.value);
            validateEmail(e.target.value);
          }}
          placeholder="example@mail.com"
        />
        {emailValid && <FaCheckCircle className="text-
success" />}
        {emailValid === false && <FaTimesCircle
className="text-danger" />}
      </div>
    </div>

```

```

        {emailValid === false && (
          <small className="text-danger">Введите корректный е-
mail</small>
        )}
      </div>

      <hr className="my-4" />

      {/* Change password */}
      <div className="mb-3">
        <label>Изменение пароля</label>
        <input
          type="password"
          className="form-control mb-2"
          placeholder="Новый пароль"
          value={newPassword}
          onChange={e => setNewPassword(e.target.value)}
        />
        <input
          type="password"
          className="form-control"
          placeholder="Повторите новый пароль"
          value={repeatPassword}
          onChange={e => setRepeatPassword(e.target.value)}
        />
        {passwordsMatch === false && (
          <div className="text-danger mt-1">Пароли не
совпадают</div>
        )}
      </div>

      <hr className="my-4" />

      {/* Псевдонимы */}
      {user.role === 'author' && (
        <div className="mb-3">
          <label>Псевдонимы</label>
          <ul className="list-group mb-2">
            {pseudonyms.map(p => (
              <li key={p.id} className="list-group-item d-flex
align-items-center">
                {editingPseudoId === p.id ? (
                  <>
                    <input
                      type="text"
                      className="form-control me-2"
                      value={editingPseudoName}
                      onChange={e =>
setEditingPseudoName(e.target.value)}
                    />
                    <button
                      className="btn btn-sm btn-primary me-2"
                      type="button"

```

```

        onClick={() => handleEditPseudo(p.id)}
      >
        Сохранить
      </button>
      <button
        className="btn btn-sm btn-secondary"
        type="button"
        onClick={() => setEditingPseudoId(null)}
      >
        Отмена
      </button>
    </>
  ) : (
    <>
      <span className="me-auto">{p.name}</span>
      <button
        className="btn btn-sm btn-outline-
primary me-2"
        type="button"
        onClick={() => {
          setEditingPseudoId(p.id);
          setEditingPseudoName(p.name);
        }}
      >
        <FaEdit />
      </button>
      <button
        className="btn btn-sm btn-outline-
danger"
        type="button"
        onClick={() => handleDeletePseudo(p.id)}
      >
        <FaTrash />
      </button>
    </>
  )}
</li>
)}}
</ul>
<div className="d-flex">
  <input
    type="text"
    className="form-control me-2"
    placeholder="Новый псевдоним"
    value={newPseudo}
    onChange={e => setNewPseudo(e.target.value)}
  />
  <button
    className="btn btn-outline-success"
    type="button"
    onClick={handleAddPseudo}
  >
    <FaPlus />

```

```

        </button>
      </div>
    </div>
  )}

  <button type="submit" className="btn btn-primary mt-3 w-
100">
    Сохранить изменения
  </button>
</form>

{modalOpen && (
  <div className="position-fixed top-0 start-0 w-100 h-100
bg-dark bg-opacity-50 d-flex justify-content-center align-items-
center">
    <div className="bg-white p-4 rounded shadow" style={{
width: '100%', maxWidth: '400px' }}>
      <h5 className="mb-3">Подтвердите изменения</h5>
      <label className="form-label">Введите текущий
пароль</label>
      <input
        type="password"
        className="form-control mb-2"
        value={confirmPassword}
        onChange={e => setConfirmPassword(e.target.value)}
      />
      {passwordError && <div className="text-danger mb-
2">{passwordError}</div>}
      <div className="d-flex justify-content-end">
        <button className="btn btn-secondary me-2"
onClick={cancelModal}>
          Отмена
        </button>
        <button className="btn btn-primary"
onClick={submitUpdate}>
          Подтвердить
        </button>
      </div>
    </div>
  </div>
)}
</div>
</div>
);
}

```

Файл «Favorites.jsx»

```

import React, { useContext } from 'react';
import { FavoritesContext } from '../context/FavoritesContext';
import { Link } from 'react-router-dom';
import { FaHeart, FaRegHeart } from 'react-icons/fa';

export default function Favorites() {

```

```

    const { favorites, toggleFavorite, isFavorite } =
    useContext(FavoritesContext);

    if (!favorites.length) {
      return (
        <div className="container mt-4">
          <h2>Отложенные книги</h2>
          <p>У вас нет отложенных книг.</p>
        </div>
      );
    }

    return (
      <div className="container mt-4">
        <h2 className="mb-4">Отложенные книги</h2>
        <div className="row">
          {favorites.map(book => (
            <div key={book.id} className="col-md-4 mb-4">
              <div className="card h-100 position-relative">
                /* Кнопка «избранное» */
                <button
                  onClick={() => toggleFavorite(book)}
                  className="btn btn-link position-absolute"
                  style={{ top: '10px', right: '10px', zIndex: 2 }}
                  aria-label="Удалить из отложенных"
                >
                  {isFavorite(book.id)
                    ? <FaHeart className="text-danger" size={24} />
                    : <FaRegHeart className="text-muted" size={24} />
                }
              </button>

              /* Изображение и заголовок — только они кликабельны */
              {book.image_url && (
                <Link to={` /book/${book.id}`}>
                  <img
                    src={book.image_url}
                    className="card-img-top"
                    alt={book.title}
                  />
                </Link>
              )}
              <div className="card-body d-flex flex-column">
                <Link to={` /book/${book.id}`} className="text-decoration-none text-dark">
                  <h5 className="card-title">{book.title}</h5>
                </Link>
                <h6 className="card-subtitle mb-2 text-muted">{book.author}</h6>
                <p className="card-text flex-grow-1">{book.description}</p>
              </div>
            </div>
          )}
        </div>
      </div>
    );
  );
}

```

```

        <div className="d-flex justify-content-between
align-items-center">
            <span className="fw-bold">{book.price} ₸</span>
        </div>
    </div>
</div>
</div>
    )})
</div>
</div>
);
}

```

Файл «Catalog.jsx»

```

import React, { useEffect, useState, useContext } from 'react'
import { Link, useNavigate } from 'react-router-dom'
import { FavoritesContext } from '../context/FavoritesContext'
import { CartContext } from '../context/CartContext'
import { FaHeart, FaRegHeart, FaShoppingCart } from 'react-
icons/fa'
import { toast, ToastContainer } from 'react-toastify'
import 'react-toastify/dist/ReactToastify.css'

export default function Catalog() {
    const [books, setBooks] = useState([])
    const [loading, setLoading] = useState(true)
    const [error, setError] = useState(null)

    const { toggleFavorite, isFavorite } =
useContext(FavoritesContext)
    const { addToCart, isInCart } = useContext(CartContext)
    const navigate = useNavigate()

    useEffect(() => {
        const loadBooks = async () => {
            try {
                const res = await fetch('http://localhost:5000/books')
                if (!res.ok) throw new Error(`Сервер вернул
${res.status}`)
                setBooks(await res.json())
            } catch (err) {
                setError(err)
            } finally {
                setLoading(false)
            }
        }
        loadBooks()
    }, [])

    const handleFavorite = (book) => {
        const wasFav = isFavorite(book.id)
        toggleFavorite(book)
        if (!wasFav) {
            toast.success('Успешно отложено!')
        }
    }
}

```

```

    }
    // при удалении тост не показываем
  }

  const handleCart = (book) => {
    if (isInCart(book.id)) {
      navigate('/cart')
    } else {
      addToCart(book)
      toast.success('Добавлено в корзину!')
    }
  }

  if (loading) {
    return <div className="container mt-4"><p>Загрузка...</p></div>
  }
  if (error) {
    return <div className="container mt-4"><p>Ошибка при загрузке: {error.message}</p></div>
  }
  if (!books.length) {
    return <div className="container mt-4"><p>Каталог пока пуст</p></div>
  }

  return (
    <div className="container mt-4">
      <h2 className="mb-4">Каталог книг</h2>
      <div className="row">
        {books.map(book => (
          <div key={book.id} className="col-md-4 mb-4">
            <div className="card h-100 position-relative">
              {/* Избранное */}
              <button
                className="btn btn-link position-absolute top-0
end-0 m-2 p-1"
                onClick={() => handleFavorite(book)}
              >
                {isFavorite(book.id)
                  ? <FaHeart className="text-danger" />
                  : <FaRegHeart className="text-secondary" />}
              </button>

              {/* Обложка */}
              <Link to={`\book/${book.id}`} className="text-decoration-none text-dark">
                {book.image_url && (
                  <img
                    src={book.image_url}
                    className="card-img-top"
                    alt={book.title}
                  />

```

```

    ))
  </Link>

  <div className="card-body d-flex flex-column">
    <Link to={`book/${book.id}`} className="text-decoration-none text-dark flex-grow-1">
      <h5 className="card-title">{book.title}</h5>
      <h6 className="card-subtitle mb-2 text-muted">{book.author}</h6>
      <p className="card-text">{book.description}</p>
    </Link>
    <div className="d-flex justify-content-between align-items-center mt-3">
      <span className="fw-bold">{book.price} ₸</span>
      <button
        className={`btn btn-sm btn-${isInCart(book.id) ? 'success' : 'primary'}`}
        onClick={() => handleCart(book)}
      >
        {isInCart(book.id)
          ? <>В корзине <small>Перейти</small></>
          : <><FaShoppingCart /> В корзину</>}
      </button>
    </div>
  </div>
</div>
))}
</div>
{ /* Тосты */ }
<ToastContainer
  position="bottom-right"
  autoClose={2000}
  hideProgressBar={false}
  newestOnTop={false}
  pauseOnHover
/>
</div>
)
}

```

Файл «Book.jsx»

```

import React, { useEffect, useState, useContext } from 'react';
import { useParams } from 'react-router-dom';
import { CartContext } from '../context/CartContext';

export default function Book() {
  const { id } = useParams();
  const [book, setBook] = useState(null);
  const { addToCart } = useContext(CartContext);

  useEffect(() => {
    fetch(`http://localhost:5000/book/${id}`)
      .then(res => res.json())

```

```

        .then(data => setBook(data))
        .catch(err => console.error('Ошибка при загрузке книги:',
err));
    }, [id]);

    if (!book) return <p>Загрузка...</p>;

    return (
        <div className="container mt-4">
            <h2>{book.title}</h2>
            <h5 className="text-muted">{book.author}</h5>
            <img src={book.image_url} alt={book.title} style={{
maxWidth: '300px' }} />
            <p className="mt-3">{book.description}</p>
            <p className="fw-bold">{book.price} ₽</p>
            <button className="btn btn-primary" onClick={() =>
addToCart(book)}>
                Добавить в корзину
            </button>
        </div>
    );
}

```

Файл «AuthorBooks.jsx»

```

import React, { useState, useEffect, useContext, useCallback }
from 'react';
import { UserContext } from '../context/UserContext';
import { useNavigate } from 'react-router-dom';

const statusLabels = {
    pending: 'На рассмотрении',
    approved: 'Принято',
    rejected: 'Отклонено'
};

const statusColors = {
    pending: 'warning',
    approved: 'success',
    rejected: 'danger'
};

export default function AuthorBooks() {
    const { user, token } = useContext(UserContext);
    const navigate = useNavigate();

    useEffect(() => {
        if (!user || user.role !== 'author') {
            navigate('/', { replace: true });
        }
    }, [user, navigate]);

    const [list, setList] = useState([]);
    const [detail, setDetail] = useState(null);

    const fetchList = useCallback(() => {

```

```

    fetch('http://localhost:5000/author/books', { headers: {
      Authorization: `Bearer ${token}` } })
      .then(r => r.ok ? r.json() : Promise.reject())
      .then(data => {
        const sorted = data
          .slice()
          .sort((a, b) => new Date(b.created_at || b.createdAt) -
            new Date(a.created_at || a.createdAt));
        setList(sorted);
      })
      .catch(() => setList([]));
  }, [token]);

  useEffect(fetchList, [fetchList]);
  useEffect(() => { if (detail === null) fetchList(); }, [detail,
    fetchList]);

  if (detail) {
    return (
      <div className="container mt-4">
        <h2 className="mb-4">Загруженные книги</h2>
        <button
          className="btn btn-success mb-3 d-block mx-auto"
          onClick={() => navigate('/author/books/add')}
        >
          Добавить книгу
        </button>

        <div className={`card mb-3 border-
          ${statusColors[detail.status]} position-relative`}>
          <div className="card-body d-flex align-items-start">
            {detail.image_url && (
              <img
                src={`http://localhost:5000${detail.image_url}`}
                alt={detail.title}
                className="me-4"
                style={{ width: '300px', height: '300px',
                  objectFit: 'cover', flexShrink: 0 }}
              />
            )}
            <div style={{ flex: 1, overflowWrap: 'break-word',
              wordBreak: 'break-word' }}>
              <h5 className="card-title fw-
                bold">{detail.title}</h5>
              <p><small className="text-muted">Псевдоним:
                {detail.author}</small></p>
              <hr style={{ width: '100%', margin: '0.5rem 0',
                borderTop: '1px solid #6c757d' }} />
              <p><strong>Жанр:</strong> {detail.genre || '-'}</p>
              <p style={{ whiteSpace: 'pre-wrap', overflowWrap:
                'break-word', wordBreak: 'break-word' }}>
                <strong>Описание:</strong><br
              />{detail.description}
            </div>
          </div>
        </div>
      </div>
    );
  }

```

```

        </p>
        {detail.publication_year && (
            <p><strong>Издание:</strong>
{detail.publication_year}г.</p>
        )}
        <p><strong>Цена:</strong> {detail.price !=
undefined ? detail.price + ' ₸' : '-'}</p>
        {detail.fragment_url && (
            <p>
                <a
href={`http://localhost:5000${detail.fragment_url}`}
                download
                target="_blank"
                rel="noopener noreferrer"
            >
                Скачать ознакомительный фрагмент
            </a>
        </p>
        )}
        {detail.book_url && (
            <p>
                <a
href={`http://localhost:5000${detail.book_url}`}
                download
                target="_blank"
                rel="noopener noreferrer"
            >
                Скачать литературу
            </a>
        </p>
        )}
    </div>
</div>
<span
    className={`badge bg-${statusColors[detail.status]}
position-absolute bottom-0 end-0 m-2 fs-5`}
    >
        {statusLabels[detail.status]}
    </span>
</div>

    <button className="btn btn-outline-secondary" onClick={()
=> setDetail(null)}>
        Назад
    </button>
</div>
);
}

return (
    <div className="container mt-4">

```

```

<h2 className="mb-4">Загруженные книги</h2>
<button
  className="btn btn-success mb-3 d-block mx-auto"
  onClick={() => navigate('/author/books/add')}
>
  Добавить книгу
</button>

{list.length === 0 ? (
  <p className="text-center">У вас пока нет загруженных
книг.</p>
) : (
  list.map(b => (
    <div
      key={b.id}
      className={`card mb-2 border-${statusColors[b.status]}
position-relative`}
      style={{ cursor: 'pointer' }}
      onClick={() => {
        fetch(`http://localhost:5000/author/book/${b.id}`, {
headers: { Authorization: `Bearer ${token}` } })
          .then(r => r.ok ? r.json() : Promise.reject())
          .then(data => setDetail(data));
      }}
    >
      <div className="card-body d-flex align-items-center">
        {b.image_url && (
          <img
            src={`http://localhost:5000${b.image_url}`}
            alt={b.title}
            className="me-3"
            style={{ width: '150px', height: '150px',
objectFit: 'cover', flexShrink: 0 }}
          />
        )}
        <div>
          <h5 className="card-title">{b.title}</h5>
          <p><small className="text-muted">Подача:
{(b.created_at || b.createdAt) ? new Date(b.created_at ||
b.createdAt).toLocaleString() : ''}</small></p>
          <p className="card-text mb-1"><small
className="text-muted">Псевдоним: {b.author}</small></p>
          <p className="card-text"><small className="text-
muted">Жанр: {b.genre || '-'}</small></p>
          {b.publication_year && (
            <p className="card-text"><small className="text-
muted">Издание: {b.publication_year}г.</small></p>
          )}
        </div>
      </div>
      <span className={`badge bg-${statusColors[b.status]}
position-absolute bottom-0 end-0 m-2`}>
        {statusLabels[b.status]}
      </span>
    </div>
  )
  )
}

```

```

        </span>
      </div>
    ))
  })
</div>
);
}

```

Файл «AdminDashbroad.jsx»

```

import React, { useState, useEffect, useContext } from 'react';
import { UserContext } from '../context/UserContext';
import { useNavigate } from 'react-router-dom';
import RequestsTab from '../components/RequestsTab';
import NewBooksTab from '../components/NewBooksTab';

const AdminDashboard = () => {
  const { user } = useContext(UserContext);
  const navigate = useNavigate();

  const [activeTab, setActiveTab] = useState('requests');

  useEffect(() => {
    if (!user || user.role !== 'admin') {
      navigate('/');
    }
  }, [user, navigate]);

  const renderTab = () => {
    switch (activeTab) {
      case 'requests':
        return <RequestsTab />;
      case 'books':
        return <NewBooksTab />;
      case 'users':
        return <div>Здесь будет список пользователей</div>;
      default:
        return null;
    }
  };

  return (
    <div className="container mt-4">
      <h2 className="mb-4">Панель администратора</h2>
      <div className="btn-group mb-3" role="group">
        <button
          className={`btn btn-outline-primary ${activeTab ===
'requests' ? 'active' : ''}`}
          onClick={() => setActiveTab('requests')}
        >
          Заявки на авторство
        </button>
        <button
          className={`btn btn-outline-primary ${activeTab ===
'books' ? 'active' : ''}`}

```

```

        onClick={() => setActiveTab('books')}
      >
        НОВЫЕ КНИГИ
      </button>
      <button
        className={`btn btn-outline-primary ${activeTab ===
'users' ? 'active' : ''}`}
        onClick={() => setActiveTab('users')}
      >
        Пользователи
      </button>
    </div>
    {renderTab()}
  </div>
);
};

```

export default AdminDashboard;

Файл «NewBooksTab.jsx»

```

import React, { useState, useEffect, useContext, useCallback }
from 'react';
import { UserContext } from '../context/UserContext';

```

```

const statusLabels = {
  pending: 'На рассмотрении',
  approved: 'Одобрено',
  rejected: 'Отклонено'
};

```

```

const statusColors = {
  pending: 'warning',
  approved: 'success',
  rejected: 'danger'
};

```

```

export default function NewBooksTab() {
  const { token } = useContext(UserContext);
  const [list, setList] = useState([]);
  const [detail, setDetail] = useState(null);

```

```

  const fetchList = useCallback(() => {
    fetch('http://localhost:5000/admin/book-requests', {
      headers: { Authorization: `Bearer ${token}` }
    })
      .then(res => res.json())
      .then(data =>
        setList(
          data
            .slice()
            .sort((a, b) => new Date(b.created_at) - new
Date(a.created_at))
        )
      )
      .catch(() => setList([]));

```

```

    }, [token]));

useEffect(fetchList, [fetchList]);
useEffect(() => {
    if (detail === null) fetchList();
}, [detail, fetchList]);

const handleAction = async (id, action) => {
    const res = await fetch(
        `http://localhost:5000/admin/book-requests/${id}/${action}`,
        { method: 'POST', headers: { Authorization: `Bearer
${token}` } }
    );
    if (res.ok) {
        setDetail(prev => ({
            ...prev,
            status: action === 'approve' ? 'approved' : 'rejected'
        }));
    }
};

if (detail) {
    const dateStr = new Date(detail.created_at).toLocaleString();
    return (
        <div>
            <h3>Книга «{detail.title}»</h3>

            <div className={`card mb-3 position-relative border-
${statusColors[detail.status]}`} >
                <div className="card-body d-flex align-items-start">
                    {detail.image_url && (
                        <img
                            src={`http://localhost:5000${detail.image_url}`}
                            alt="Cover"
                            className="me-4"
                            style={{ width: '300px', height: '300px',
objectFit: 'cover', flexShrink: 0 }}
                        />
                    )}
                    <div style={{ flex: 1, overflowWrap: 'break-word',
wordBreak: 'break-word' }} >
                        <h5 className="card-title fw-
bold">{detail.title}</h5>
                        <p><small className="text-muted">Псевдоним:
{detail.author}</small></p>
                        <p><small className="text-muted">Подача:
{dateStr}</small></p>
                        <hr style={{ width: '120px', margin: '0.5rem 0',
borderTop: '1px solid #6c757d' }} />

                        <p><strong>Жанр:</strong> {detail.genre || '-'}</p>
                        <p style={{ whiteSpace: 'pre-wrap', overflowWrap:
'break-word', wordBreak: 'break-word' }} >

```

```

        <strong>Описание:</strong><br
/>{detail.description}
    </p>
    {detail.publication_year && (
        <p><strong>Издание:</strong>
{detail.publication_year}г.</p>
    )}
    <p><strong>Цена:</strong> {detail.price} Р</p>

    {detail.fragment_url && (
        <p>
            <a

href={`http://localhost:5000${detail.fragment_url}`}
            download
            target="_blank"
            rel="noopener noreferrer"
        >
            Скачать ознакомительный фрагмент
        </a>
    </p>
    )}
    {detail.book_url && (
        <p>
            <a

href={`http://localhost:5000${detail.book_url}`}
            download
            target="_blank"
            rel="noopener noreferrer"
        >
            Скачать литературу
        </a>
    </p>
    )}
</div>
</div>

<span
    className={`badge bg-${statusColors[detail.status]}
position-absolute bottom-0 end-0 m-2 fs-5`}
>
    {statusLabels[detail.status]}
</span>
</div>

<div className="d-flex gap-3">
    {detail.status === 'pending' && (
        <>
            <button
                className="btn btn-success"
                onClick={() => handleAction(detail.id, 'approve')}
            >

```

```

        Одобрить
      </button>
      <button
        className="btn btn-danger"
        onClick={() => handleAction(detail.id, 'reject')}
      >
        Отклонить
      </button>
    </>
  )}
  <button
    className="btn btn-outline-secondary"
    onClick={() => setDetail(null)}
  >
    Назад
  </button>
</div>
</div>
);
}

return (
  <div>
    {list.map(b => {
      const dateStr = new Date(b.created_at).toLocaleString();
      return (
        <div
          key={b.id}
          className={`card mb-2 position-relative border-
            ${statusColors[b.status]}`}
          style={{ cursor: 'pointer' }}
          onClick={() =>
            fetch(`http://localhost:5000/admin/book-
            requests/${b.id}`, {
              headers: { Authorization: `Bearer ${token}` }
            })
              .then(res => res.json())
              .then(data => setDetail(data))
              .catch(() => {})}
        >
          <div className="card-body d-flex align-items-center">
            {b.image_url && (
              <img
                src={`http://localhost:5000${b.image_url}`}
                alt={b.title}
                className="me-3"
                style={{ width: '150px', height: '150px',
                  objectFit: 'cover', flexShrink: 0 }}
              />
            )}
          </div>
          <h5 className="card-title">{b.title}</h5>

```

```

        <p><small className="text-muted">Подача:
{dateStr}</small></p>
        <p className="card-text mb-1"><small
className="text-muted">Псевдоним: {b.author}</small></p>
        <p className="card-text"><small className="text-
muted">Жанр: {b.genre || '-'}</small></p>
        {b.publication_year && (
        <p className="card-text"><small className="text-
muted">Издание: {b.publication_year}г.</small></p>
        )}
    </div>
</div>
<span
    className={`badge bg-${statusColors[b.status]}
position-absolute bottom-0 end-0 m-2`}
    >
    {statusLabels[b.status]}
</span>
</div>
);
    )}
</div>
);
}

```

Файл «AddBook.jsx»

```

import React, { useState, useContext, useEffect, useCallback }
from 'react';
import { UserContext } from '../context/UserContext';
import { useNavigate } from 'react-router-dom';
import './AddBook.css';

export default function AddBook() {
    const { user, token } = useContext(UserContext);
    const navigate = useNavigate();

    useEffect(() => {
        if (!user || user.role !== 'author') {
            navigate('/', { replace: true });
        }
    }, [user, navigate]);

    const [coverFile, setCoverFile] = useState(null);
    const [coverPreview, setCoverPreview] = useState(null);
    const [fragmentFile, setFragmentFile] = useState(null);
    const [bookFile, setBookFile] = useState(null);
    const [title, setTitle] = useState('');
    const [pseudonyms, setPseudonyms] = useState([]);
    const [pseudonymId, setPseudonymId] = useState('');
    const [genre, setGenre] = useState('');
    const [description, setDescription] = useState('');
    const [publicationYear, setPublicationYear] = useState('');
    const [price, setPrice] = useState('');
    const [overlay, setOverlay] = useState(false);

```

```

const fetchPseudonyms = useCallback(() => {
  fetch('http://localhost:5000/author/pseudonyms', {
    headers: { Authorization: `Bearer ${token}` }
  })
  .then(res => res.json())
  .then(data => setPseudonyms(data))
  .catch(() => setPseudonyms([]));
}, [token]);

useEffect(() => {
  fetchPseudonyms();
}, [fetchPseudonyms]);

// Обработка выбора файла обложки
const handleCoverChange = e => {
  const file = e.target.files[0];
  if (!file) return;
  const ext = file.name.split('.').pop().toLowerCase();
  if (!['png', 'jpg', 'jpeg'].includes(ext)) {
    alert('Допустимы только форматы PNG и JPG.');
```

return;

```

  }
  const reader = new FileReader();
  reader.onload = () => {
    const img = new Image();
    img.onload = () => {
      if (img.width < 300 || img.height < 300 || img.width >
1000 || img.height > 1000) {
        alert('Изображение должно быть от 300×300 до
1000×1000.');
```

setCoverFile(null);

setCoverPreview(null);

```

      } else {
        setCoverFile(file);
        setCoverPreview(reader.result);
      }
    };
    img.src = reader.result;
  };
  reader.readAsDataURL(file);
};

// Обработка выбора ознакомительного фрагмента
const handleFragmentChange = e => {
  const file = e.target.files[0];
  if (!file) return;
  const ext = file.name.split('.').pop().toLowerCase();
  if (!['epub', 'fb2', 'mobi', 'pdf', 'doc', 'docx'].includes(ext)) {
    alert('Неподдерживаемый формат фрагмента.');
```

return;

```

  }
  setFragmentFile(file);

```

```

};

// Обработка выбора книги
const handleBookChange = e => {
  const file = e.target.files[0];
  if (!file) return;
  const ext = file.name.split('.').pop().toLowerCase();
  if (!['epub', 'fb2', 'mobi', 'pdf', 'doc', 'docx'].includes(ext)) {
    alert('Неподдерживаемый формат книги.');
```

return;

```
  }
  setBookFile(file);
};

const handleSubmit = async e => {
  e.preventDefault();
  if (!title.trim() || !description.trim() || !price.trim() ||
  isNaN(price)) {
    alert('Пожалуйста, заполните все поля и укажите корректную
цену.');
```

return;

```
  }
  if (!coverFile) {
    alert('Пожалуйста, загрузите обложку книги.');
```

return;

```
  }
  if (!bookFile) {
    alert('Пожалуйста, загрузите файл книги.');
```

return;

```
  }
  if (!pseudonymId) {
    alert('Пожалуйста, выберите псевдоним.');
```

return;

```
  }

  const fd = new FormData();
  fd.append('cover', coverFile);
  fd.append('fragment', fragmentFile || '');
  fd.append('book', bookFile);
  fd.append('title', title.trim());
  fd.append('pseudonym_id', pseudonymId);
  fd.append('genre', genre.trim());
  fd.append('description', description.trim());
  fd.append('publication_year', publicationYear.trim());
  fd.append('price', price.trim());

  const res = await fetch('http://localhost:5000/author/books',
  {
    method: 'POST',
    headers: { Authorization: `Bearer ${token}` },
    body: fd
  });
});

```

```

    if (res.ok) {
      setOverlay(true);
    } else {
      const data = await res.json();
      alert(data.error || 'Ошибка отправки');
    }
  };

  return (
    <div className="container mt-4">
      <h2 className="mb-4">Добавить книгу</h2>
      <form onSubmit={handleSubmit} style={{ display: overlay ?
'none' : 'block' }} encType="multipart/form-data">
        {/* Обложка */}
        <div className="mb-3 text-center">
          <label htmlFor="coverInput" className="btn btn-outline-
secondary">
            {coverPreview ? 'Изменить обложку' : 'Загрузить
обложку'}
          </label>
          <input id="coverInput" type="file"
accept=".png, .jpg, .jpeg" className="d-none"
onChange={handleCoverChange} />
        </div>
        {coverPreview && (
          <div className="mb-3 text-center">
            <div className="cover-preview"><img src={coverPreview}
alt="Cover Preview" /></div>
            </div>
          </div>
        )}

        {/* Поля */}
        <div className="mb-3">
          <label>Название книги</label>
          <input required type="text" className="form-control"
value={title} onChange={e => setTitle(e.target.value)} />
        </div>
        <div className="mb-3">
          <label>Псевдоним</label>
          <select
            required
            className="form-select"
            value={pseudonymId}
            onChange={e => setPseudonymId(e.target.value)}
          >
            <option value="">Выберите псевдоним...</option>
            {pseudonyms.map(p => (
              <option key={p.id} value={p.id}>{p.name}</option>
            ))}
          </select>
        </div>
        <div className="mb-3">
          <label>Жанр</label>

```

```

        <input type="text" className="form-control"
value={genre} onChange={e => setGenre(e.target.value)} />
    </div>
    <div className="mb-3">
        <label>Описание</label>
        <textarea required className="form-control" rows="6"
value={description} onChange={e => setDescription(e.target.value)}
/>
    </div>
    <div className="mb-3">
        <label>Год издания</label>
        <input
            type="number"
            className="form-control"
            value={publicationYear}
            onChange={e => setPublicationYear(e.target.value)}
        />
    </div>
    <div className="mb-3">
        <label>Цена (₽)</label>
        <input required type="number" min="0" className="form-
control" value={price} onChange={e => setPrice(e.target.value)} />
    </div>

    {/* Фрагмент и книга */}
    <div className="mb-3">
        <label className="form-label">Ознакомительный фрагмент
(необязательно)</label>
        <input type="file"
accept=".epub,.fb2,.mobi,.pdf,.doc,.docx" className="form-control"
onChange={handleFragmentChange} />
    </div>
    <div className="mb-3">
        <label className="form-label">
            Загрузить книгу <span className="text-danger">*</span>
        </label>
        <input required type="file"
accept=".epub,.fb2,.mobi,.pdf,.doc,.docx" className="form-control"
onChange={handleBookChange} />
    </div>

```

Приложение С

(Обязательное)

Web-приложение «Книжный интернет-магазин»

Презентация

					ПГУ 1.090301.10.001 ПЗ	Лист
						99
Изм.	Лист	№ докум.	Подпись	Дата		

Тема работы: Web-приложение «Книжный интернет магазин»

Автор работы:

студент гр.21ВВП2 Лукина Е.Ю.

Научный руководитель:

к.э.н., доцент, каф. ВТ ПГУ

Акифьев И.В.

Пенза 2025

1

Актуальность разработки

С развитием электронной торговли растёт спрос на удобные онлайн-сервисы. Книжные интернет-магазины позволяют быстро находить нужную литературу, оформлять заказы и получать отзывы, но многие существующие решения перегружены и неудобны. Разработка собственного web-приложения помогает создать более простую, адаптивную и доступную платформу как для читателей, так и для авторов.



2

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

100

Цель работы

Разработка web-приложения «Книжный интернет-магазин» с удобным интерфейсом для поиска, покупки и публикации книг, включающего функционал для пользователей, авторов и администратора.

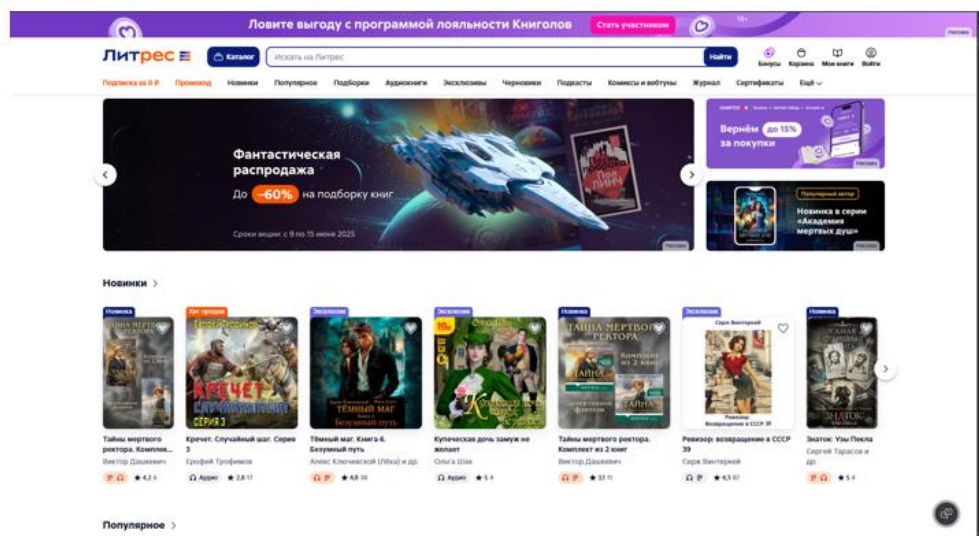
Объект работы

Приложение, выполняющее следующие функции:

- каталогизация и отображение книг с возможностью фильтрации;
- оформление заказов и управление корзиной;
- публикация и модерация книг со стороны авторов и администратора;
- реализация личного кабинета с ролевой моделью.

3

Аналоги



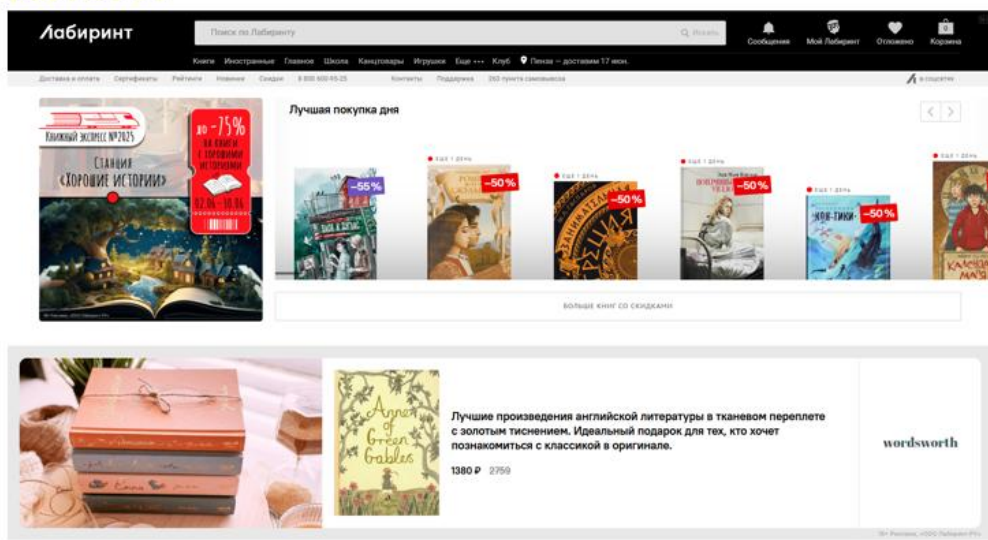
4

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

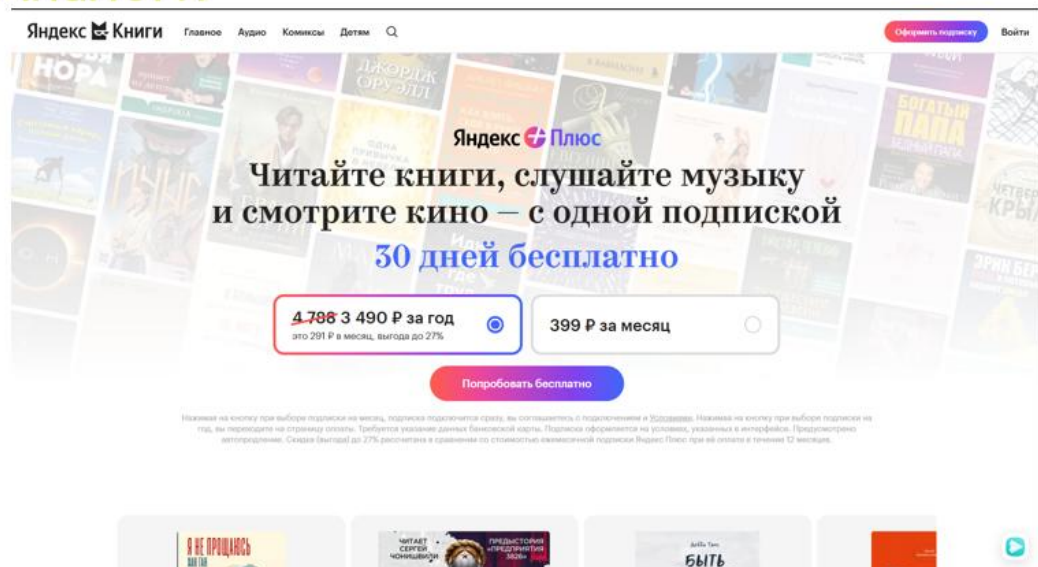
Лист
101

Аналоги



5

Аналоги



6

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

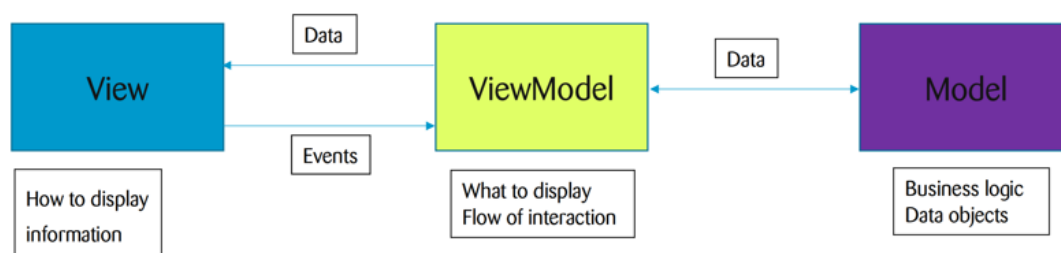
102

Стек технологий



7

Паттерн Model – View - ViewModel



8

Изм.	Лист	№ докум.	Подпись	Дата

ПГУ 1.090301.10.001 ПЗ

Лист

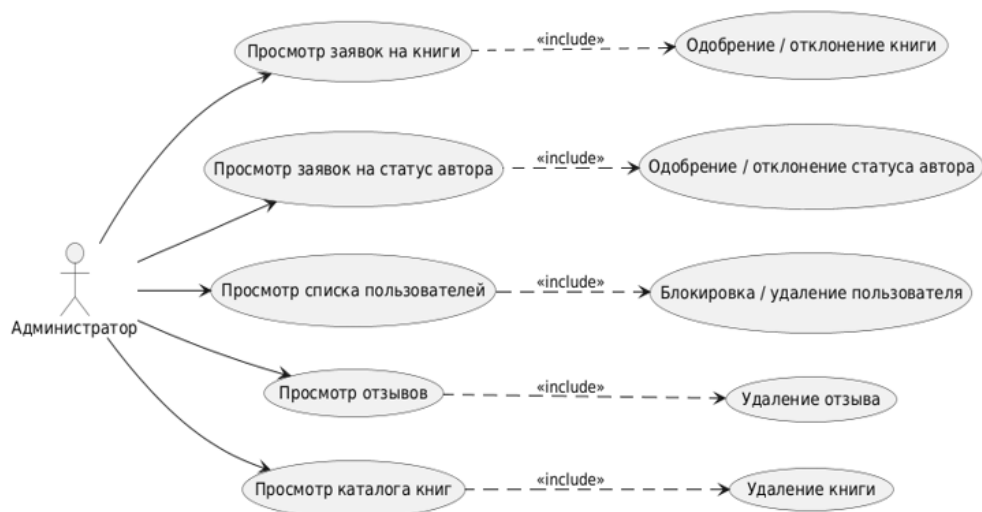
103

Диаграмма вариантов использования



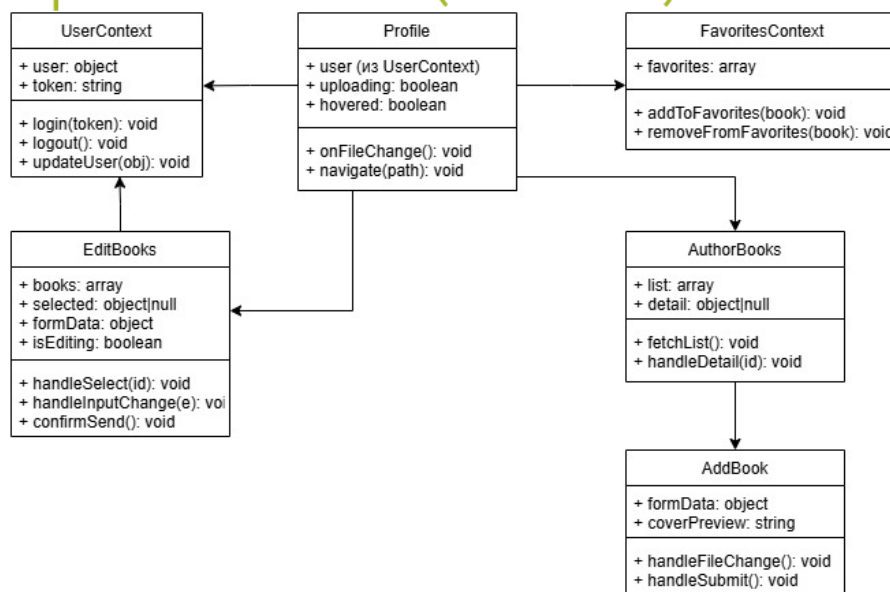
9

Диаграмма вариантов использования



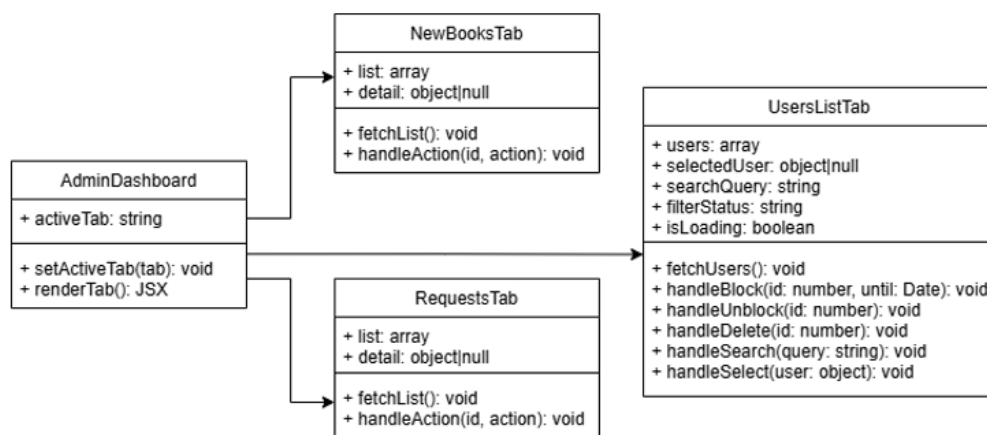
10

Диаграмма классов (frontend)



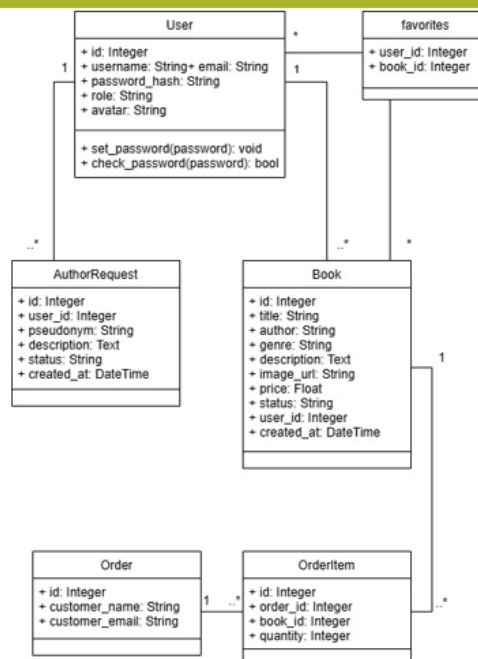
11

Диаграмма классов (frontend)



12

Диаграмма классов (backend)



13

Заключение

В ходе выполнения выпускной квалификационной работы было разработано web-приложение «Книжный интернет-магазин» с клиентской частью на React и серверной частью на Flask. Интерфейс разработан с учётом принципов адаптивности и удобства, реализована поддержка ролевой модели.

Разработанное приложение позволяет:

- просматривать каталог книг с фильтрацией и поиском;
- оформлять заказы через корзину;
- регистрироваться и управлять личным кабинетом;
- оставлять отзывы о книгах;
- авторам публиковать, редактировать и удалять свои книги;
- администраторам управлять пользователями и модерировать заявки.

Возможно усовершенствовать систему, добавив:

- интеграцию онлайн-оплаты заказов;
- расширенную аналитику для авторов;
- внедрение рекомендательной системы;
- автоматическую систему уведомлений и рассылок.

14

Спасибо за внимание!

15

					ПГУ 1.090301.10.001 ПЗ	Лист
						107
Изм.	Лист	№ докум.	Подпись	Дата		