

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

<i>Факультет</i>	<i>Кафедра</i>
вычислительной техники	МОиПЭВМ
Направление подготовки	09.03.04 «Программная инженерия»
Направленность	Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

**Автоматизированная интеллектуальная
система продажи книг**

Студент	_____	Князев Данил Эдуардович
	(подпись, дата)	(ФИО полностью)
Руководитель	_____	Афонин А.Ю.
	(подпись, дата)	(фамилия, инициалы)
Нормоконтролер	_____	Такташкин Д.В.
	(подпись, дата)	(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой	_____	Козлов А.Ю.
	(подпись)	(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК	_____	Попова Н.А.
	(подпись)	(фамилия, инициалы)

Пенза, 2026

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Вычислительной техники

Кафедра

МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
А.Ю. Козлов «26»
декабря 2025 г

**ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА**

1. Студент Князев Данил Эдуардович гр. 22ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы Автоматизированная интеллектуальная система продажи книг

Тема утверждена приказом ПГУ №1538/о от "30" декабря 2025 г.

3. Руководитель работы Афонин Александр Юрьевич

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

Разработать автоматизированную интеллектуальную систему продажи книг, представляющую собой веб-приложение, которое объединяет функционал современного интернет-магазина с модулем интеллектуальных рекомендаций на основе обработки естественного языка.

Назначение системы: обеспечение удобного поиска, подбора и продажи книг пользователям через веб-интерфейс, а также предоставление администратору инструментов для управления ассортиментом и заказами.

Исходные данные: современные технологии веб-разработки, требования к функционалу типового книжного интернет-магазина, принципы микросервисной архитектуры и обработки естественного языка.

4. Перечень подлежащих разработке вопросов

1. Провести анализ предметной области
2. Сформулировать функциональные и нефункциональные требования к системе.
3. Спроектировать архитектуру системы
4. Разработать структуру базы данных.
5. Реализовать серверную часть на языке Go в виде микросервисов
6. Разработать клиентское веб-приложение на React с TypeScript.
7. Реализовать интеллектуальный модуль рекомендаций книг по текстовому запросу.
8. Обеспечить взаимодействие компонентов через REST API и событийную шину (Kafka).
9. Выполнить контейнеризацию и развертывание системы с помощью Docker Compose.
10. Провести модульное и функциональное тестирование разработанного программного обеспечения.
11. Оформить пояснительную записку и подготовить работу к защите.

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	07.04.2026 – 13.04.2026	
2	Проектирование программных средств	20%	14.04.2026 – 25.04.2026	
3	Реализация программных средств	30%	27.04.2026 – 11.05.2026	
4	Тестирование программных средств	20%	12.05.2026 – 21.05.2026	
5	Оформление пояснительной записки	20%	22.05.2026 – 02.06.2026	

Дата выдачи задания «26» декабря 2025 г.

Руководитель бакалаврской работы _____ 26.12.2025
(подпись, дата)

Задание к исполнению принял студент _____ 26.12.2025
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 124 листов, 25 таблиц, 32 рисунка, 20 использованных источников, 3 приложения.

АВТОМАТИЗИРОВАННАЯ СИСТЕМА, ИНТЕРНЕТ-МАГАЗИН,
КНИЖНЫЙ КАТАЛОГ, МИКРОСЕРВИСНАЯ АРХИТЕКТУРА,
ИНТЕЛЛЕКТУАЛЬНЫЙ ПОМОЩНИК, РЕКОМЕНДАЦИИ, GO, REACT,
POSTGRESQL

Объектом исследования является процесс автоматизации продажи книг в сфере электронной коммерции.

Целью работы является разработка автоматизированной интеллектуальной системы продажи книг, объединяющей функции книжного интернет-магазина и подбор литературы по текстовому запросу пользователя.

В ходе работы были проанализированы предметная область и аналоги, определены требования, спроектированы архитектура системы и база данных. Также реализованы серверные сервисы, клиентское веб-приложение и интеллектуальный чат-помощник.

Серверная часть разработана на Go с использованием микросервисной архитектуры, клиентская часть – на React и TypeScript. Для хранения данных используется PostgreSQL, запуск компонентов выполняется через Docker Compose.

Результатом работы является веб-приложение для просмотра каталога, оформления заказов, работы с отзывами, избранным и получения книжных рекомендаций через интеллектуального помощника.

ПГУ 09.03.04 – 08BP221.1 ПЗ

					ПГУ 09.03.04 – 08BP221.1 ПЗ								
Изм.	Лист	№ докум.	Подпись	Дата									
Разраб.		Князев Д.Э.			Автоматизированная интеллектуальная система продажи книг				Лит.	Лист	Листов		
Провер.		Афонин А.Ю.									4	124	
Реценз.									Группа 22ВП1				
Н. Контр.		Такташкин Д.В.											
Утверд.		Козлов А.Ю.											

Содержание

Введение	8
1 Анализ предметной области и постановка задачи	10
1.1 Анализ предметной области	10
1.2 Особенности электронной торговли книгами.....	11
1.3 Проблемы выбора книг пользователями.....	12
1.4 Анализ программного обеспечения аналогичного назначения	13
1.4.1 Лабиринт	14
1.4.2 Читай-город	15
1.4.3 Book24.....	16
1.4.4 Ozon, раздел «Книги».....	17
1.4.5 Сравнительная характеристика программного обеспечения аналогичного назначения	18
1.5 Постановка задачи на разработку.....	19
1.6 Планирование разработки и оценка бюджета.....	21
2 Анализ требований к информационной системе	29
2.1 Анализ требований информационной системы	29
2.2 Требования к серверному программному обеспечению	33
2.3 Требования к клиентскому программному обеспечению	34
2.4 Требования к интеллектуальному модулю рекомендаций	35
3 Проектирование информационной системы.....	37
3.1 Архитектура информационной системы.....	37
3.2 Проектирование базы данных	39
3.2.1 Концептуальное моделирование базы данных	39
3.2.2 Логическое моделирование базы данных	46

3.3	Проектирование серверного программного обеспечения	49
3.3.1	Назначение и функции серверного приложения	49
3.3.2	Проектирование архитектуры серверного приложения	51
3.3.3	Механизмы безопасности	52
3.4	Проектирование клиентского программного обеспечения	53
3.4.1	Назначение и функции клиентского приложения	53
3.4.2	Проектирование пользовательского интерфейса	54
3.4.3	Проектирование структуры клиентского приложения	55
3.5	Обоснование выбора средств разработки и технологий	56
4	Разработка программного обеспечения	59
4.1	Физическое моделирование базы данных	59
4.2	Разработка серверного приложения	60
4.2.1	Структура серверного приложения	60
4.2.2	Взаимодействие с базой данных	62
4.2.3	Реализация сервиса авторизации	64
4.2.4	Реализация каталога книг, отзывов и рейтингов	66
4.2.5	Реализация сервиса корзины и заказов	71
4.2.6	Реализация сервиса профиля пользователя	74
4.2.7	Реализация интеллектуального сервиса рекомендаций	76
4.3	Разработка клиентского приложения	80
4.3.1	Структура клиентского приложения	80
4.3.2	Разработка пользовательского интерфейса	85
4.4	Развертывание информационной системы	93
5	Тестирование	95
5.1	Функциональное тестирование	95

5.2 Модульное тестирование	98
5.3 Метрики качества кода	102
5.3.1 Общие сведения о метриках качества кода.....	102
5.3.2 Метрики SLOC	103
Заключение	105
Список используемых источников.....	107
Приложение А (справочное) Глоссарий основных терминов	109
Приложение Б (обязательное) Код программы	113
Приложение В (рекомендуемое) Скриншоты интерфейса	122

Введение

В современном мире электронная коммерция стала привычным и удобным способом покупки товаров, в том числе книг. Книжные интернет-магазины позволяют быстро просматривать каталог, искать нужные произведения, добавлять их в корзину и оформлять заказ. Однако при огромном ассортименте у пользователя часто возникает серьезная проблема выбора. Далеко не всегда человек заранее точно знает, какую именно книгу хочет прочитать [1].

Обычный поиск хорошо работает, если известно название, автор или конкретный жанр. Но на практике запросы пользователей чаще бывают размытыми. В таких случаях приходится вручную просматривать подборки, рейтинги и списки бестселлеров, что занимает много времени и не всегда приводит к удачному выбору.

Актуальность данной работы заключается в создании книжного интернет-магазина с интеллектуальным чат-помощником на базе модели OpenAI GPT-4o.

В отличие от стандартного поиска, такой помощник позволяет описать желаемую книгу простыми словами, а система подберет наиболее подходящие варианты из каталога и объяснит, почему именно эти книги были рекомендованы.

Цель выпускной квалификационной работы – разработать автоматизированную интеллектуальную систему продажи книг, которая объединяет обычный функционал интернет-магазина с умным подбором литературы по текстовому описанию пользователя.

Для достижения цели были поставлены следующие задачи:

- проанализировать предметную область и существующие решения;
- определить требования к системе;
- спроектировать архитектуру и базу данных;
- реализовать серверную часть на Go в виде микросервисов;
- разработать клиентское приложение на React + TypeScript;
- создать модуль интеллектуальных рекомендаций;

- провести тестирование и развернуть систему с помощью Docker Compose.

Объектом исследования является процесс автоматизации продажи книг в электронной коммерции. Предметом исследования – методы и средства разработки веб-системы книжного магазина с интеллектуальным модулем рекомендаций.

В результате работы должна быть реализована полноценная веб-система, позволяющая пользователям удобно выбирать, покупать книги и получать персонализированные рекомендации через чат-помощник.

1 Анализ предметной области и постановка задачи

1.1 Анализ предметной области

Разрабатываемая система относится к сфере книжной электронной торговли. Пользователь здесь работает с каталогом книг, подробными карточками товаров, корзиной, заказами, личным профилем, избранным, отзывами, рейтингами и интеллектуальным помощником подбора литературы.

Книжный интернет-магазин должен выполнять целый ряд ключевых операций: публиковать информацию о книгах, обеспечивать удобный поиск и фильтрацию, оформление заказов, хранение данных пользователей и управление товаром. Все эти функции логично распределены между клиентской частью, микросервисами и базой данных.

Каждая книга как товар содержит стандартный набор характеристик: название, автора, жанр, описание, цену, обложку и количество экземпляров на складе. Именно эти данные используются для отображения, поиска и фильтрации товаров в каталоге.

Однако на практике пользователь не всегда выбирает книгу только по формальным признакам. Часто запросы касаются настроения, атмосферы произведения, уровня сложности текста или сходства с уже прочитанными книгами.

В таких случаях классический поиск по названию, автору или ключевому слову оказывается недостаточно эффективным. Именно поэтому в систему добавлен интеллектуальный чат-помощник. Он принимает описание книги на естественном языке, сопоставляет его с имеющимся каталогом и выдает рекомендации вместе с пояснениями.

Таким образом, интеллектуальный модуль выступает естественным дополнением к обычному каталогу. Пользователь может как пользоваться стандартными инструментами поиска и фильтрации, так и описывать желаемую книгу своими словами.

Система предусматривает две основные роли. Зарегистрированный пользователь может просматривать каталог, работать с корзиной и заказами, управлять профилем, оставлять отзывы и оценки. Администратор же получает возможность добавлять, редактировать и удалять книги, а также управлять поступающими заказами.

Все данные хранятся в реляционной базе PostgreSQL. Серверная часть построена по микросервисной архитектуре и включает отдельные сервисы для авторизации, работы с каталогом, обработки заказов, профиля пользователя и интеллектуального подбора книг.

1.2 Особенности электронной торговли книгами

Электронная торговля книгами – это процесс продажи печатных и электронных изданий через информационные системы. Главным элементом такой системы выступает каталог книг, где собраны все необходимые сведения о каждом товаре: название, автор, описание, жанр, цена, количество доступных экземпляров, изображение обложки и другие характеристики, помогающие пользователю сделать выбор и оформить покупку.

Книжный интернет-магазин должен выполнять ряд ключевых функций. Пользователь должен иметь возможность удобно просматривать каталог и получать подробную информацию о каждой книге. Кроме того, система обязана обеспечивать качественный поиск и фильтрацию товаров, ведь ассортимент может насчитывать тысячи произведений разных жанров и авторов. Не менее важно, чтобы покупатель мог добавлять книги в корзину, изменять их количество и спокойно оформлять заказ. Также система должна надежно хранить информацию о пользователях, их заказах, избранных товарах и других действиях в магазине.

Для администратора необходимы удобные инструменты управления каталогом. Он должен иметь возможность добавлять новые книги, редактировать существующие позиции, изменять цену, описание, жанр и остатки

на складе. Не менее важна и работа с заказами: после оформления система сохраняет состав заказа, его статус и данные покупателя.

Отличительной особенностью книжного интернет-магазина является сложность навигации по каталогу. В отличие от большинства других товаров, книги выбирают не только по техническим характеристикам, но и по содержанию. Пользователь часто ищет не конкретное произведение, а книгу с определенным настроением, атмосферой, подходящую по уровню сложности или похожую на уже прочитанные. Например, запрос может звучать так: «хочу книгу похожую на детектив, но с элементами мистики» или «нужна классика, но более легкая для чтения».

Такие запросы сложно обработать только с помощью обычной фильтрации. Фильтр по жанру может сильно сузить список, но далеко не всегда может учитывать смысловой контекст. Поиск по названию или автору тоже часто не помогает, когда человек не знает точного произведения. Именно поэтому для повышения удобства выбора целесообразно использовать интеллектуальный модуль рекомендаций.

Интеллектуальная система рекомендаций существенно расширяет возможности книжного интернет-магазина. Она способна анализировать свободную формулировку пользователя, выделять основные признаки желаемой книги и подбирать наиболее подходящие варианты из каталога. В итоге пользователь получает не просто список книг, а осмысленные рекомендации с объяснением причин выбора. Это заметно упрощает процесс покупки и снижает вероятность того, что покупатель покинет сайт из-за трудностей с выбором.

1.3 Проблемы выбора книг пользователями

Основная сложность, с которой сталкиваются пользователи книжных интернет-магазинов, возникает именно тогда, когда у человека нет четкого представления о том, какую книгу он хочет приобрести. Если известно точное название или автор, обычный поиск легко решает данную проблему. Однако в большинстве случаев пользователь ориентируется на настроение, атмосферу

произведения или сходство с уже прочитанными книгами. В таких ситуациях стандартный поиск и фильтры показывают довольно слабую эффективность.

Фильтрация по жанру, например, не учитывает многие важные характеристики: сложность текста, темп повествования, эмоциональную окраску и сочетание нескольких признаков одновременно. Две книги одного и того же жанра могут сильно отличаться по возрастной аудитории, стилю изложения и основной тематике.

В обычном книжном-магазине роль помощника при выборе берет на себя продавец-консультант. Он уточняет предпочтения покупателя и предлагает подходящие для него варианты. В веб-приложении данную функцию можно логично возложить на интеллектуальный чат-помощник.

Помощник принимает запрос на простом языке и возвращает список наиболее подходящих книг. Такой подход особенно удобен, когда в одном запросе сочетается сразу несколько факторов: схожесть с конкретным произведением, нужный жанр, уровень сложности и желаемая атмосфера.

Еще одна важная проблема – доверие к рекомендациям. Пользователь оценивает книгу не только по описанию, но и по отзывам других читателей, рейтингу и количеству оценок. Поэтому хорошо продуманная система отзывов и рейтингов становится важным дополнением к интеллектуальному подбору и помогает пользователю принять более обоснованное решение перед покупкой.

1.4 Анализ программного обеспечения аналогичного назначения

Для сравнения с разрабатываемой системой были выбраны четыре популярных решения в сфере книжной электронной торговли: «Лабиринт», «Читай-город», Book24 и раздел «Книги» маркетплейса Ozon. Все они предоставляют пользователям каталог, поиск, фильтрацию, корзину, оформление заказов, отзывы и рекомендации.

Критериями сравнения стали наличие каталога книг, поиска, фильтрации, корзины, личного кабинета, избранного, отзывов, рейтингов, готовых подборок,

а также возможность интеллектуального подбора книг по свободному текстовому запросу.

1.4.1 Лабиринт

Лабиринт представляет собой книжный интернет-магазин, предназначенный для продажи книг, учебной литературы и сопутствующих товаров [2]. Система предоставляет пользователю каталог товаров, корзину, раздел отложенных товаров, доставку, оплату, рейтинги, новинки и скидки. На сайте также присутствуют редакционные подборки и разделы с рекомендациями, например «Главные книги» и материалы, где редакция собирает лучшие книги и важные события книжного рынка. Логотип компании представлен на рисунке 1.



Рисунок 1 – Логотип интернет-магазина Лабиринт

К преимуществам интернет-магазина Лабиринт можно отнести большой каталог книг, наличие отзывов и пользовательских оценок, систему рейтингов, скидки, а также редакционные рекомендации и тематические материалы.

Недостатком данного решения в контексте разрабатываемой системы является то, что подбор книг в основном строится на классических механизмах каталога, рейтингов, подборок и отзывов. Пользователь может искать книги по названию, автору или категории, однако возможность сформулировать сложный

запрос в свободной форме и получить персональную подборку книг с объяснением выбора не является основной функцией системы.

1.4.2 Читай-город

Читай-город – это сеть книжных магазинов, предоставляющая пользователям возможность приобретения книг, канцелярии, игр, подарков и других товаров [3]. В интернет-магазине представлен большой ассортимент книг, новинок и хитов, а также предусмотрены доставка, акции, программа лояльности и подарочные сертификаты. Главная страница интернет-магазина изображена на рисунке 2.

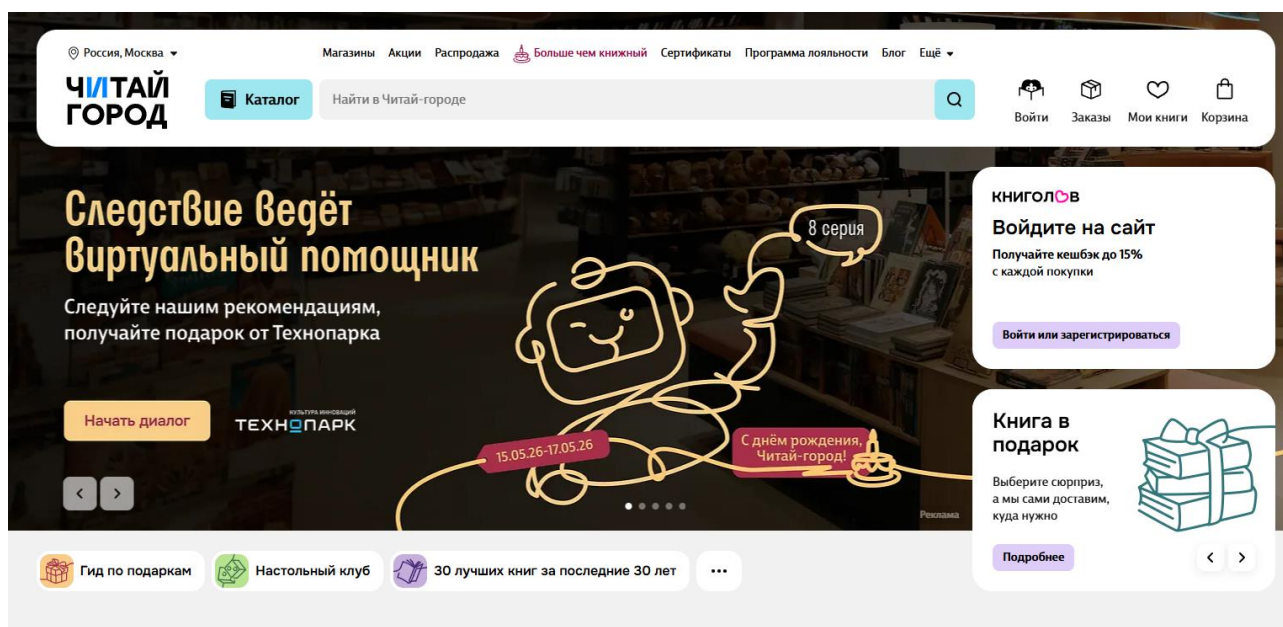


Рисунок 2 – Главная страница интернет-магазина Читай-город

Сильной стороной Читай-города является сочетание интернет-магазина с развитой розничной сетью. Пользователь может выбрать книгу на сайте и получить ее удобным способом: оформить доставку или забрать заказ в ближайшем магазине. Дополнительно сервис помогает ориентироваться в ассортименте за счет акций, подборок и разделов с популярными изданиями.

При этом рекомендации в системе в основном представлены в виде готовых подборок, топов и тематических разделов. Такой подход помогает пользователю выбрать книгу из уже сформированных списков, но не полностью

решает задачу индивидуального подбора, когда пользователь хочет описать желаемое произведение своими словами.

Особенностью разрабатываемой системы является наличие интеллектуального чат-помощника, который позволяет сформировать запрос в свободной форме и получить подборку книг из каталога с пояснением причин выбора.

1.4.3 Book24

Book24 – это книжный интернет-магазин, в котором пользователь может выбрать и заказать книги с доставкой. На сайте представлены новинки, бестселлеры, тематические подборки, рейтинги, обзоры и отзывы покупателей. Сервис также помогает пользователю определиться с выбором за счет списков книг от известных людей и подборок по различным темам. Главная страница книжного интернет-магазина Book24 представлена на рисунке 3.

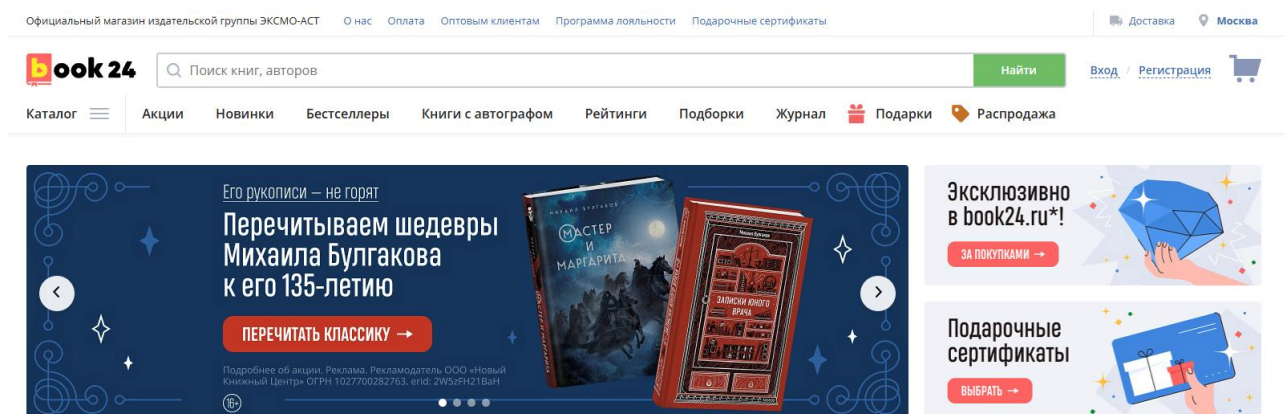


Рисунок 3 – Главная страница книжного интернет-магазина Book24

Сервис хорошо помогает пользователю ориентироваться в каталоге за счет подборок, обзоров, списков рекомендованных книг, рейтингов и отзывов. Такой подход позволяет выбирать книги не только по названию, автору или жанру, но и на основе редакционных материалов и мнений других покупателей [4].

Ограничением Book24 является то, что рекомендации в основном представлены в готовом виде. Пользователь может изучать подборки и обзоры,

но при сложном индивидуальном запросе ему все равно приходится самостоятельно сравнивать описания, рейтинги и отзывы. В разрабатываемой системе эту задачу должен решать интеллектуальный чат-помощник, подбирающий книги по свободному текстовому запросу.

1.4.4 Ozon, раздел «Книги»

Ozon является крупным маркетплейсом, в котором книги представлены как одна из товарных категорий [5]. Пользователь может искать книги, применять фильтры, читать отзывы, добавлять товары в корзину и оформлять доставку вместе с товарами других категорий. Логотип маркетплейса Ozon представлен на рисунке 4.



Рисунок 4 – Логотип маркетплейса Ozon

Главное преимущество Ozon заключается в масштабе платформы. Пользователю доступен большой ассортимент товаров, развитая система поиска и фильтрации, отзывы, рейтинги, рекомендации и удобная доставка. Кроме того, книгу можно приобрести вместе с товарами других категорий в рамках одного заказа.

При этом универсальность маркетплейса является и ограничением для книжной предметной области. Интерфейс и логика рекомендаций рассчитаны на разные группы товаров, поэтому система хорошо помогает найти конкретную книгу, но не заменяет специализированный подбор литературы по сюжету, стилю, настроению или сходству с ранее прочитанными произведениями.

1.4.5 Сравнительная характеристика программного обеспечения аналогичного назначения

В ходе анализа программного обеспечения аналогичного назначения были рассмотрены наиболее популярные решения, которые позволяют пользователям приобретать книги через интернет. Для сравнения были выбраны такие критерии, как наличие каталога, поиска, фильтрации, корзины, личного кабинета, избранных товаров, отзывов, подборок и интеллектуального подбора книг. Результаты сравнительного анализа представлены в таблице 1.

Таблица 1 – Сравнительная характеристика программного обеспечения аналогичного назначения

Критерий	Лабиринт	Читай-город	Book24	Ozon	Разрабатываемая система
Каталог книг	Да	Да	Да	Да	Да
Поиск по каталогу	Да	Да	Да	Да	Да
Фильтрация книг	Да	Да	Да	Да	Да
Просмотр карточки книги	Да	Да	Да	Да	Да
Корзина	Да	Да	Да	Да	Да
Оформление заказа	Да	Да	Да	Да	Да
Личный кабинет	Да	Да	Да	Да	Да
Избранные товары	Да	Да	Да	Да	Да
Отзывы и рейтинги	Да	Да	Да	Да	Да

Продолжение таблицы 1

Критерий	Лабиринт	Читай-город	Book24	Ozon	Разрабатываемая система
Управление каталогом администратором	Да	Да	Да	Да	Да
Готовые подборки книг	Да	Да	Да	Да	Да
Подбор по свободному текстовому запросу	Нет	Нет	Нет	Нет	Да
Объяснение причины рекомендации	Частично	Частично	Частично	Частично	Да
Интеллектуальный чат-помощник	Нет	Нет	Нет	Нет	Да

Как видно из таблицы 1, все рассмотренные системы хорошо реализуют базовые функции интернет-магазина. Однако ни одна из них не предоставляет полноценный интеллектуальный подбор книг по свободному текстовому запросу пользователя с объяснением причин рекомендаций.

1.5 Постановка задачи на разработку

Цель выпускной квалификационной работы – разработка автоматизированной интеллектуальной системы продажи книг, которая объединяет полный функционал интернет-магазина с интеллектуальным чат-помощником для подбора литературы.

Система должна обеспечивать полный цикл работы интернет-магазина и качественный подбор литературы через интеллектуальный чат-помощник.

Система представляет собой веб-приложение на клиент-серверной архитектуре. Клиентская часть предоставляет интерфейс для работы с каталогом, поиском, корзиной, заказами, профилем и чат-помощником. Серверная часть реализует бизнес-логику, обработку запросов и взаимодействие с базой данных.

Основной функционал включает просмотр и поиск книг, фильтрацию, работу с корзиной и избранным, оформление заказов, управление профилем, оставление отзывов и оценок. Для администратора предусмотрено управление каталогом и обработка заказов.

Ключевой особенностью проекта является интеллектуальный модуль рекомендаций. Пользователь может отправить запрос в свободной форме, а система подберет книги из каталога и пояснит причины выбора.

Для достижения цели необходимо решить следующие задачи:

- проанализировать предметную область книжной электронной коммерции и существующие решения;
- определить функциональные и нефункциональные требования;
- спроектировать архитектуру системы и структуру базы данных;
- реализовать серверные сервисы (авторизация, каталог, заказы, профиль, рекомендации);
- разработать клиентское веб-приложение;
- обеспечить взаимодействие компонентов через REST API;
- провести тестирование системы.

Серверная часть строится на Go с использованием микросервисной архитектуры. Для хранения данных применяется PostgreSQL. Клиентская часть разрабатывается на React с TypeScript. Развертывание компонентов осуществляется с помощью Docker Compose [6].

1.6 Планирование разработки и оценка бюджета

Планирование проекта включало определение последовательности задач, оценку трудозатрат и расчет стоимости разработки. Для этого был проведен функционально-стоимостный анализ.

Функционально-стоимостный анализ представляет собой метод оценки затрат на выполнение процессов с учетом используемых ресурсов и трудоемкости отдельных этапов. Такой подход позволяет определить наиболее ресурсоемкие этапы разработки [7].

Проведение функционально-стоимостного анализа обычно включает последовательное выполнение подготовительного, информационного, аналитического, исследовательского и рекомендательного этапов [8].

Все работы были разделены на этапы: анализ, проектирование, разработка, тестирование и оформление документации. Полный список задач с длительностью и предшественниками приведен в таблице 2.

Таблица 2 – Список задач проекта

№	Задача	Длительность	Предшественник
1	Начало работы над проектом	—	—
2	Анализ предметной области и постановка задачи	—	1
3	Анализ предметной области книжной электронной коммерции	2 дня	2
4	Анализ программного обеспечения аналогичного назначения	2 дня	3
5	Постановка задачи на разработку	1 день	4
6	Планирование и функционально-стоимостный анализ	2 дня	5

Продолжение таблицы 2

№	Задача	Длительность	Предшественник
7	Анализ требований к информационной системе	—	6
8	Определение функциональных требований	2 дня	7
9	Определение требований к серверной части	1 день	8
10	Определение требований к клиентской части	1 день	9
11	Определение требований к интеллектуальному модулю	1 день	10
12	Проектирование информационной системы	—	11
13	Проектирование архитектуры системы	2 дня	12
14	Проектирование базы данных	2 дня	13
15	Проектирование серверных сервисов	3 дня	14
16	Проектирование клиентского приложения	2 дня	15
17	Проектирование интеллектуального модуля рекомендаций	2 дня	16
18	Выбор средств разработки и технологий	1 день	17
19	Разработка информационной системы	—	18
20	Разработка сервиса авторизации	2 дня	19
21	Разработка сервиса каталога книг	4 дня	20

Окончание таблицы 2

№	Задача	Длительность	Предшественник
22	Разработка сервиса заказов и корзины	4 дня	21
23	Разработка сервиса профиля пользователя	3 дня	22
24	Разработка интеллектуального сервиса рекомендаций	4 дня	23
25	Разработка клиентского приложения	6 дней	24
26	Настройка базы данных и миграций	2 дня	25
27	Настройка Docker Compose и развертывания	2 дня	26
28	Тестирование информационной системы	—	27
29	Функциональное тестирование	3 дня	28
30	Тестирование интеллектуального чат-помощника	2 дня	29
31	Исправление выявленных ошибок	3 дня	30
32	Оформление документации	5 дней	31
33	Сдача проекта	—	32
34	Конец работы над проектом	—	33

Для наглядного отражения временных характеристик разработки программного обеспечения была построена диаграмма Ганта, приведенная на рисунке 5.

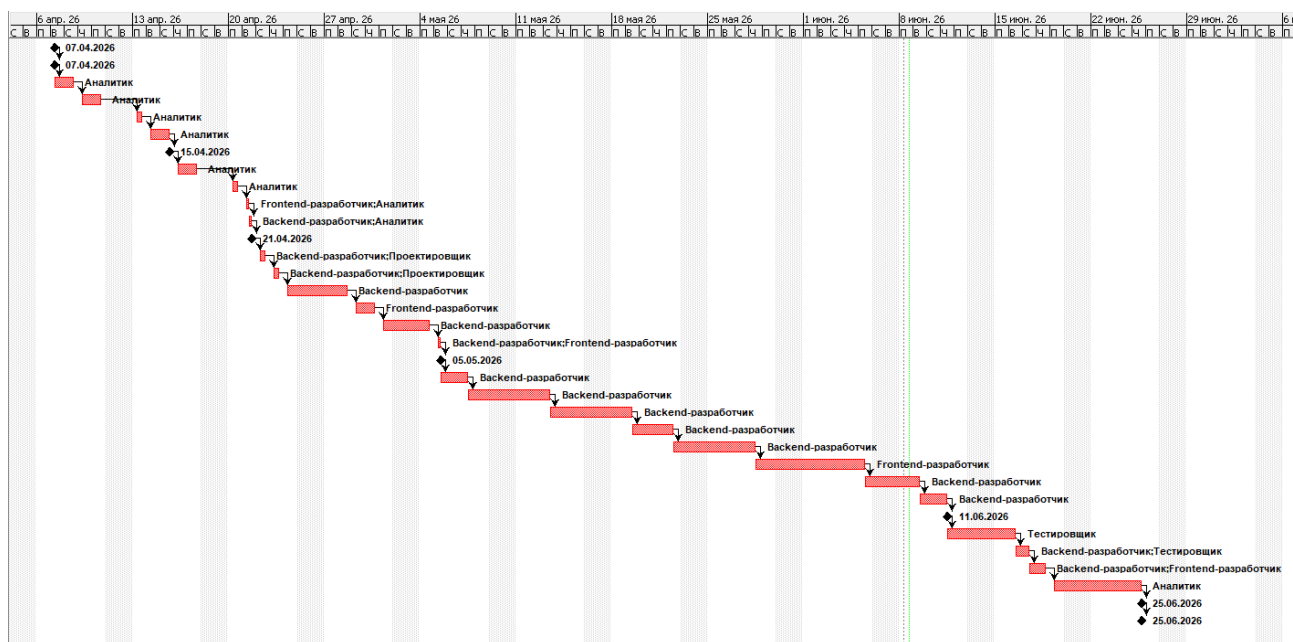


Рисунок 5 – Диаграмма Ганта

Следующим этапом является определение ресурсов и назначение их на задачи проекта. В таблице 3 представлен список используемых ресурсов.

Таблица 3 – Список используемых ресурсов

Название ресурса	Тип ресурса	Стандартная ставка	Сверхурочная ставка
Аналитик	Трудовой	270 руб/ч	—
Проектировщик	Трудовой	280 руб/ч	—
Backend-разработчик	Трудовой	320 руб/ч	—
Frontend-разработчик	Трудовой	289 руб/ч	—
Тестировщик	Трудовой	260 руб/ч	—

Для удобства расчета принято, что один рабочий день составляет 8 часов. Поскольку проект выполняется в рамках выпускной квалификационной работы, роли специалистов могут быть совмещены одним разработчиком.

В таблице 4 отображено распределение ресурсов по задачам проекта.

Таблица 4 – Распределение ресурсов по задачам

№	Задача	Ресурсы	Единицы
1	Начало работы над проектом	-	-
2	Анализ предметной области и постановка задачи на разработку	-	-
3	Анализ предметной области книжной электронной коммерции	Аналитик	100%
4	Анализ программного обеспечения аналогичного назначения	Аналитик	100%
5	Постановка задачи на разработку	Аналитик	100%
6	Планирование и функционально-стоимостный анализ	Аналитик	100%
7	Анализ требований к информационной системе	-	-
8	Определение функциональных требований	Аналитик	100%
9	Определение требований к серверной части	Аналитик	100%
10	Определение требований к клиентской части	Аналитик	100%
		Frontend-разработчик	50%
11	Определение требований к интеллектуальному модулю	Аналитик	100%
		Backend-разработчик	50%
12	Проектирование информационной системы	-	-
13	Проектирование архитектуры системы	Проектировщик	100%
		Backend-разработчик	50%

Продолжение таблицы 4

№	Задача	Ресурсы	Единицы
14	Проектирование базы данных	Проектировщик	100%
		Backend-разработчик	50%
15	Проектирование серверных сервисов	Backend-разработчик	100%
16	Проектирование клиентского приложения	Frontend-разработчик	100%
17	Проектирование интеллектуального модуля рекомендаций	Backend-разработчик	100%
18	Выбор средств разработки и технологий	Backend-разработчик	50%
		Frontend-разработчик	50%
19	Проектирование информационной системы завершено	-	-
20	Разработка сервиса авторизации	Backend-разработчик	100%
21	Разработка сервиса каталога книг	Backend-разработчик	100%
22	Разработка сервиса заказов и корзины	Backend-разработчик	100%
23	Разработка сервиса профиля пользователя	Backend-разработчик	100%
24	Разработка интеллектуального сервиса рекомендаций	Backend-разработчик	100%
25	Разработка клиентского приложения	Frontend-разработчик	100%
26	Настройка базы данных и миграций	Backend-разработчик	100%
27	Настройка Docker Compose и развертывания	Backend-разработчик	100%
28	Разработка информационной системы завершена	-	-
29	Тестирование	-	-
30	Функциональное тестирование	Тестирующий	100%
31	Тестирование интеллектуального чат-помощника	Тестирующий	100%
		Backend-разработчик	50%

Окончание таблицы 4

32	Исправление выявленных ошибок	Backend-разработчик	50%
		Frontend-разработчик	50%
33	Оформление документации	Аналитик	100%

На основе распределения ресурсов можно отметить, что основная нагрузка приходится на backend-разработчика, так как серверная часть системы разделена на несколько сервисов. Помимо стандартной бизнес-логики интернет-магазина, необходимо реализовать взаимодействие между сервисами, работу с базой данных и интеллектуальный модуль рекомендаций. Frontend-разработчик в большей степени задействован на этапе создания пользовательского интерфейса.

Для расчета общей стоимости разработки необходимо определить трудозатраты каждого специалиста. Расчет выполняется на основе длительности задач, восьмичасового рабочего дня и ставки специалиста.

В таблице 5 представлен предварительный расчет трудозатрат и стоимости работ.

Таблица 5 – Расчет трудозатрат и стоимости работ

Ресурс	Количество часов	Ставка	Стоимость
Аналитик	128 ч	270 руб/ч	34 560 руб
Проектировщик	48 ч	280 руб/ч	13 440 руб
Backend-разработчик	216 ч	320 руб/ч	69 120 руб
Frontend-разработчик	80 ч	289 руб/ч	23 120 руб
Тестировщик	40 ч	260 руб/ч	10 400 руб
Итого	512 ч	—	150 640 руб

На рисунке 6 представлена сводная информация о трудозатратах и стоимости проекта.

Название	Начало	Окончание	Менеджер	Отчётная дата	Стоимость	Работа
Книжный магазин	08.06.2026, 08:00	25.06.2026, 17:00	Админ	08.06.2026, 17:00	150640,01 Р	512 часов

Рисунок 6 – Сводные показатели проекта

В результате расчета получено, что ориентировочная стоимость разработки автоматизированной интеллектуальной системы продажи книг составляет 150 640 рублей. Наибольшую часть затрат составляет работа backend-разработчика, поскольку серверная часть включает несколько сервисов: авторизацию, каталог, заказы, профиль пользователя и интеллектуальные рекомендации. Также значимую часть занимает разработка клиентского приложения, так как от качества интерфейса напрямую зависит удобство использования системы.

Следует отметить, что при реальной коммерческой разработке стоимость подобной системы могла бы быть выше, так как дополнительно учитывались бы расходы на серверную инфраструктуру, доменное имя, поддержку, дизайн, сопровождение, интеграцию платежных систем и более глубокое тестирование безопасности. В рамках выпускной квалификационной работы расчет носит предварительный характер и отражает основные трудозатраты на создание работоспособного программного продукта.

2 Анализ требований к информационной системе

2.1 Анализ требований информационной системы

На основании постановки задачи на разработку были выделены следующие функциональные требования к системе:

- зарегистрироваться в системе;
- войти в систему;
- выйти из системы;
- просмотреть каталог книг;
- выполнить поиск книги;
- отфильтровать книги по жанру;
- просмотреть подробную информацию о книге;
- добавить книгу в корзину;
- изменить количество книги в корзине;
- удалить книгу из корзины;
- оформить заказ;
- просмотреть историю заказов;
- добавить книгу в избранное;
- удалить книгу из избранного;
- просмотреть профиль пользователя;
- изменить данные профиля пользователя;
- оставить отзыв к книге;
- выставить оценку книге от 1 до 5;
- отредактировать свой отзыв;
- удалить свой отзыв;
- отправить запрос интеллектуальному чат-помощнику;
- получить рекомендации книг;
- просмотреть объяснение рекомендации;
- добавить книгу в каталог;

- редактировать книгу в каталоге;
- удалить книгу из каталога;
- просмотреть список заказов;
- изменить статус заказа;
- удалить пользовательский отзыв администратором.

Диаграмма вариантов использования пользователя автоматизированной интеллектуальной системы продажи книг, представленная на рисунке 7.

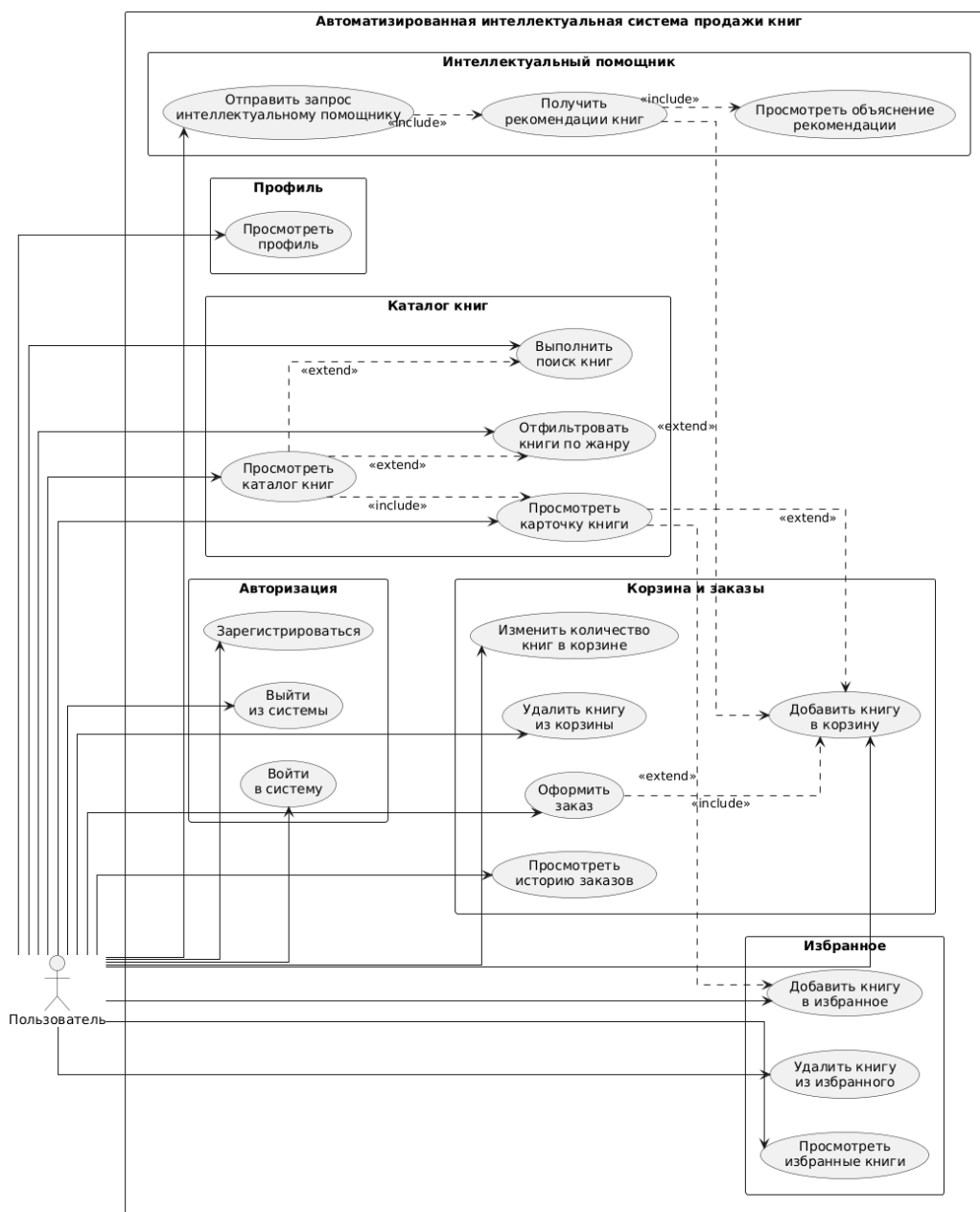


Рисунок 7 – Диаграмма вариантов использования пользователя автоматизированной интеллектуальной системы продажи книг

На рисунке 8 представлена диаграмма вариантов использования администратора.

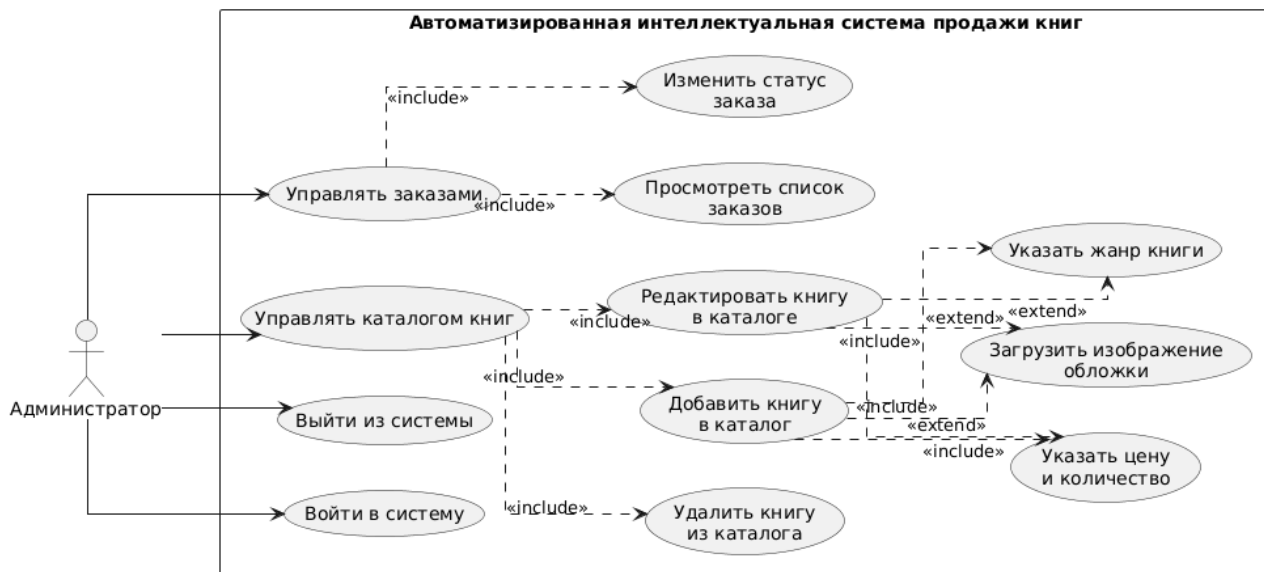


Рисунок 8 – Диаграмма вариантов использования администратора автоматизированной интеллектуальной системы продажи книг

На рисунке 9 изображена спецификация варианта использования «Оформить заказ».

Id: 1
Описание: Оформление заказа пользователем на основе книг, добавленных в корзину
Действующие лица: Пользователь, система
Предусловие: Пользователь авторизован, в корзине находится хотя бы одна книга
Основной поток: <ol style="list-style-type: none"> 1. Пользователь переходит в корзину. 2. Система отображает список добавленных книг, их количество и итоговую стоимость. 3. Пользователь проверяет состав корзины. 4. Пользователь при необходимости изменяет количество книг или удаляет лишние позиции. 5. Пользователь нажимает кнопку оформления заказа. 6. Система проверяет наличие выбранных книг в каталоге. 7. Система создаёт заказ и сохраняет его данные. 8. Система обновляет количество доступных экземпляров книг. 9. Система отображает пользователю информацию об успешном оформлении заказа.
Постусловие: Заказ создан и отображается в истории заказов пользователя

Рисунок 9 – Спецификация сценария варианта использования «Оформить заказ»

На рисунке 10 представлена спецификация варианта использования «Получить рекомендации книг».

<u>Id: 2</u>
Описание: Получение персонализированной подборки книг на основе текстового запроса пользователя
Действующие лица: Пользователь, система, интеллектуальный модуль
Предусловие: Пользователь авторизован, в каталоге имеются доступные книги
Основной поток: <ol style="list-style-type: none"> 1. Пользователь открывает страницу интеллектуального помощника. 2. Система отображает окно чата для ввода запроса. 3. Пользователь вводит текстовый запрос в свободной форме. 4. Система передаёт запрос в интеллектуальный модуль рекомендаций. 5. Интеллектуальный модуль анализирует смысл запроса пользователя. 6. Система получает из каталога книги, которые могут соответствовать запросу. 7. Интеллектуальный модуль формирует список рекомендуемых книг. 8. Система отображает пользователю рекомендации и краткое объяснение по каждой книге. 9. Пользователь при необходимости открывает карточку рекомендованной книги или добавляет её в корзину.
Постусловие: Пользователь получил список рекомендованных книг из каталога системы

Рисунок 10 – Спецификация сценария варианта использования «Получить рекомендации книг»

На рисунке 11 представлена спецификация варианта использования «Добавить книгу в каталог».

<u>Id: 3</u>
Описание: Добавление новой книги в каталог интернет-магазина администратором
Действующие лица: Администратор, система
Предусловие: Администратор авторизован и имеет права на управление каталогом
Основной поток: <ol style="list-style-type: none"> 1. Администратор переходит в раздел управления каталогом. 2. Система отображает список книг и элементы управления каталогом. 3. Администратор нажимает кнопку добавления новой книги. 4. Система открывает форму добавления книги. 5. Администратор вводит название, автора, описание, жанр, цену и количество экземпляров. 6. Администратор при необходимости загружает изображение обложки. 7. Администратор подтверждает добавление книги. 8. Система проверяет корректность введенных данных. 9. Система сохраняет новую книгу в базе данных. 10. Система отображает добавленную книгу в каталоге.
Постусловие: Новая книга добавлена в каталог и доступна пользователям для просмотра

Рисунок 11 – Спецификация сценария варианта использования «Добавить книгу в каталог»

В результате анализа функциональных требований были определены основные сценарии работы пользователя и администратора.

2.2 Требования к серверному программному обеспечению

Серверное программное обеспечение обрабатывает запросы от клиентского приложения, выполняет бизнес-логику, взаимодействует с базой данных и обеспечивает контроль прав доступа.

Взаимодействие между клиентом и сервером реализуется через REST API с использованием формата JSON. Таким образом передаются данные о пользователях, книгах, корзине, заказах, отзывах и рекомендациях.

Серверная часть разрабатывается на языке Go. Для хранения данных используется база данных PostgreSQL. Запуск и оркестрация всех компонентов системы осуществляется с помощью Docker Compose.

Архитектура серверной части построена по микросервисному принципу и включает пять независимых сервисов:

- auth – регистрация, авторизация и работа с JWT-токенами;
- catalog – управление книгами, отзывами и рейтингами;
- order – обработка корзины и заказов;
- profile – данные профиля пользователя и избранное;
- ai – интеллектуальный подбор книг.

К серверному обеспечению предъявляются следующие ключевые требования:

- регистрация и авторизация пользователей;
- формирование и проверка JWT-токенов;
- разграничение прав доступа между пользователем и администратором;
- обработка запросов к каталогу, включая поиск, фильтрацию и сортировку книг;
- добавление, редактирование и удаление книг администратором;

- работа с корзиной и оформление заказов;
- хранение истории заказов и изменение их статусов;
- управление профилем пользователя и избранными книгами;
- создание, редактирование и удаление отзывов;
- расчет среднего рейтинга книги;
- обработка запросов к AI-сервису и возврат рекомендаций;
- обработка ошибок и возврат корректных HTTP-ответов.

Пароли пользователей хранятся только в хешированном виде. При авторизации введенный пароль сравнивается с сохраненным хешем.

JWT-токен обязателен для доступа к защищенным маршрутам. Права доступа проверяются на серверной стороне. Обычный пользователь работает только со своими данными, а администратор дополнительно получает доступ к управлению каталогом, заказами и удалению чужих отзывов.

PostgreSQL хранит информацию о пользователях, книгах, корзине, заказах, позициях заказов, избранных книгах, отзывах и оценках. Для отзывов действует ограничение: один пользователь может оставить только один отзыв на одну книгу.

AI-сервис принимает текстовый запрос пользователя и возвращает рекомендации исключительно по книгам, присутствующим в каталоге. Результат передается клиентскому приложению в формате JSON.

2.3 Требования к клиентскому программному обеспечению

Клиентское программное обеспечение представляет собой веб-интерфейс, через который пользователи взаимодействуют с автоматизированной интеллектуальной системой продажи книг.

Клиентская часть разрабатывается на React с использованием TypeScript. React отвечает за построение динамического интерфейса, а TypeScript в свою очередь обеспечивает типизацию данных и компонентов.

Приложение должно содержать следующие основные страницы: регистрация, авторизация, каталог книг, карточка книги, корзина, оформление заказа, профиль пользователя, избранное, история заказов, отзывы и интеллектуальный чат-помощник. Для администратора предусмотрена отдельная панель управления каталогом и заказами.

Клиентское приложение взаимодействует с серверной частью исключительно через REST API, обмениваясь данными в формате JSON. JWT-токен сохраняется на стороне клиента и автоматически передается в заголовках запросов к защищенным маршрутам.

Выделяются следующие требования к клиентскому программному обеспечению:

- отображение форм регистрации и авторизации;
- просмотр каталога с поддержкой поиска, фильтрации и сортировки;
- отображение подробной карточки книги с описанием, отзывами и рейтингом;
- работа с корзиной;
- оформление и просмотр истории заказов;
- управление профилем и избранными книгами;
- создание, редактирование и удаление отзывов;
- интерфейс интеллектуального чат-помощника для отправки текстовых запросов и отображения рекомендаций с пояснениями;
- административная панель для управления книгами и заказами.

Таким образом, клиентское приложение предоставляет доступ к функциям системы через интерфейс, с которым пользователь взаимодействует для получения желаемого результата

2.4 Требования к интеллектуальному модулю рекомендаций

Интеллектуальный модуль рекомендаций предназначен для подбора книг по текстовому запросу пользователя. В отличие от стандартного поиска, он

позволяет работать с запросами в свободной форме и возвращать рекомендации только из книг, которые уже присутствуют в каталоге системы.

Пользователь может описать желаемую книгу своими словами. Запрос способен содержать название похожего произведения, жанр, настроение, сложность текста или несколько параметров одновременно.

К модулю предъявляются следующие требования:

- принимать текстовый запрос пользователя;
- получать список книг из каталога;
- анализировать название, автора, жанр и описание каждой книги;
- подбирать несколько наиболее подходящих книг;
- формировать краткое пояснение для каждой рекомендации;
- возвращать результат в формате JSON;
- не рекомендовать книги, которых нет в каталоге;
- корректно обрабатывать пустые или некорректные запросы;
- возвращать сообщение, если подходящих книг не найдено.

AI-сервис реализован как отдельный серверный компонент. Он получает запрос от клиентского приложения, обращается к данным каталога и формирует ответ с рекомендациями.

Результат работы должен содержать идентификатор книги, название, автора, изображение обложки, цену и текстовое пояснение причины рекомендации. Эти данные используются клиентским приложением для отображения карточек рекомендованных книг в интерфейсе чат-помощника.

Интеллектуальный модуль не должен изменять какие-либо данные в каталоге, корзине, заказах или профиле пользователя. Его задача ограничивается анализом запроса, подбором книг и формированием пояснений.

Таким образом, интеллектуальный модуль рекомендаций расширяет стандартный функционал интернет-магазина и делает процесс выбора книги более удобным.

3 Проектирование информационной системы

3.1 Архитектура информационной системы

Автоматизированная интеллектуальная система продажи книг проектируется с использованием трехуровневой клиент-серверной архитектуры.

Клиент-серверная архитектура – архитектурный стиль, при котором функции системы разделяются между клиентским приложением и серверной частью [9].

Трехуровневая архитектура включает 3 основных уровня: клиентское приложение, серверную часть и базу данных [9].

Клиентское приложение обеспечивает взаимодействие пользователя с системой через веб-интерфейс. Через интерфейс пользователь просматривает каталог книг, открывает карточку книги, работает с корзиной, оформляет заказы, просматривает профиль, избранные книги, отзывы, рейтинги и использует интеллектуальный чат-помощник.

Серверная часть выполняет обработку запросов, валидацию данных, бизнес-логику, аутентификацию и авторизацию пользователей. В системе серверная часть реализуется на языке Go и разделяется на 5 сервисов: auth, catalog, order, profile и ai.

База данных отвечает за хранение и управление данными системы.

Nginx используется как обратный прокси. Он принимает HTTP-запросы и направляет их к соответствующим сервисам. Обмен данными между клиентским приложением и серверной частью выполняется через REST API в формате JSON.

Для локального запуска компонентов используется Docker Compose. В конфигурации объединяются клиентское приложение, Nginx, сервисы, PostgreSQL, Kafka и вспомогательные контейнеры.

Архитектура информационной системы представлена на рисунке 12.

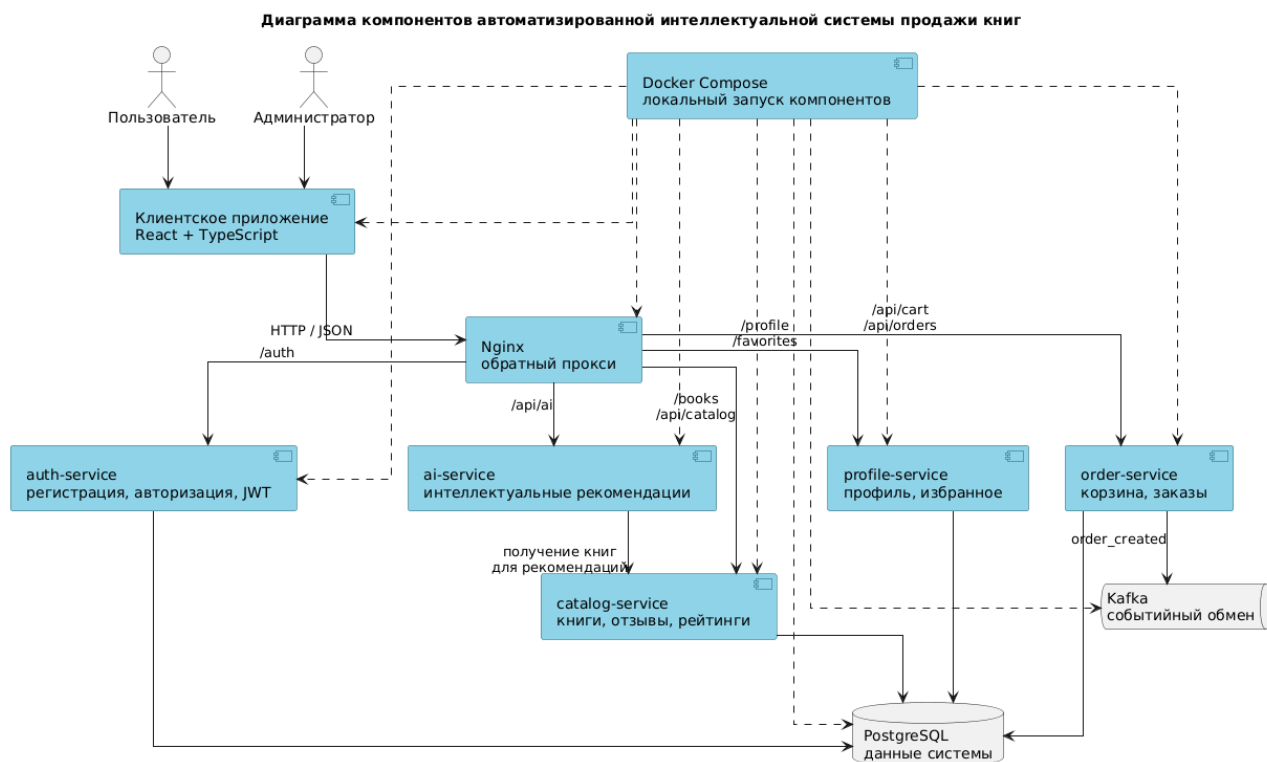


Рисунок 12 – Архитектура автоматизированной интеллектуальной системы продажи книг

Описание основных компонентов архитектуры представлено в таблице 6.

Таблица 6 – Описание компонентов архитектуры информационной системы

Компонент	Назначение
Клиентское приложение	Отображает каталог, карточку книги, корзину, заказы, профиль и интерфейс чат-помощника
Auth-сервис	Выполняет регистрацию, авторизацию и проверку JWT-токенов
Catalog-сервис	Обрабатывает книги, жанры, отзывы и рейтинги
Order-сервис	Обрабатывает корзину, оформление заказов и изменение статусов заказов
Profile-сервис	Хранит данные профиля пользователя и список избранных книг

Продолжение таблицы 6

Компонент	Назначение
AI-сервис	Подбирает книги по свободному текстовому запросу пользователя
PostgreSQL	Хранит данные пользователей, книг, заказов, корзины, отзывов, рейтингов и избранного
Kafka	Передаёт события между серверными сервисами
Docker Compose	Запускает клиентское приложение, серверные сервисы, базу данных и вспомогательные компоненты

Выбор такой клиент-серверной архитектуры обусловлен ее гибкостью и удобством разработки. Независимость компонентов позволяет проектировать и развивать их отдельно друг от друга не боясь, что изменение одного уровня может сильно повлиять на остальные. Кроме того, такая структура повышает безопасность системы, поскольку обработка данных и их хранение находятся на разных уровнях. Каждый запрос проходит через серверную часть, где осуществляется строгий контроль доступа и валидация, что существенно снижает риск несанкционированного доступа к данным напрямую из клиентского приложения.

3.2 Проектирование базы данных

3.2.1 Концептуальное моделирование базы данных

При проектировании базы данных были выделены основные сущности: пользователь, книга, позиция корзины, заказ, позиция заказа, избранная книга и отзыв, которые представлены на ERD-диаграмме концептуальной модели.

На рисунке 13 представлена ERD-диаграмма концептуальной модели базы данных.

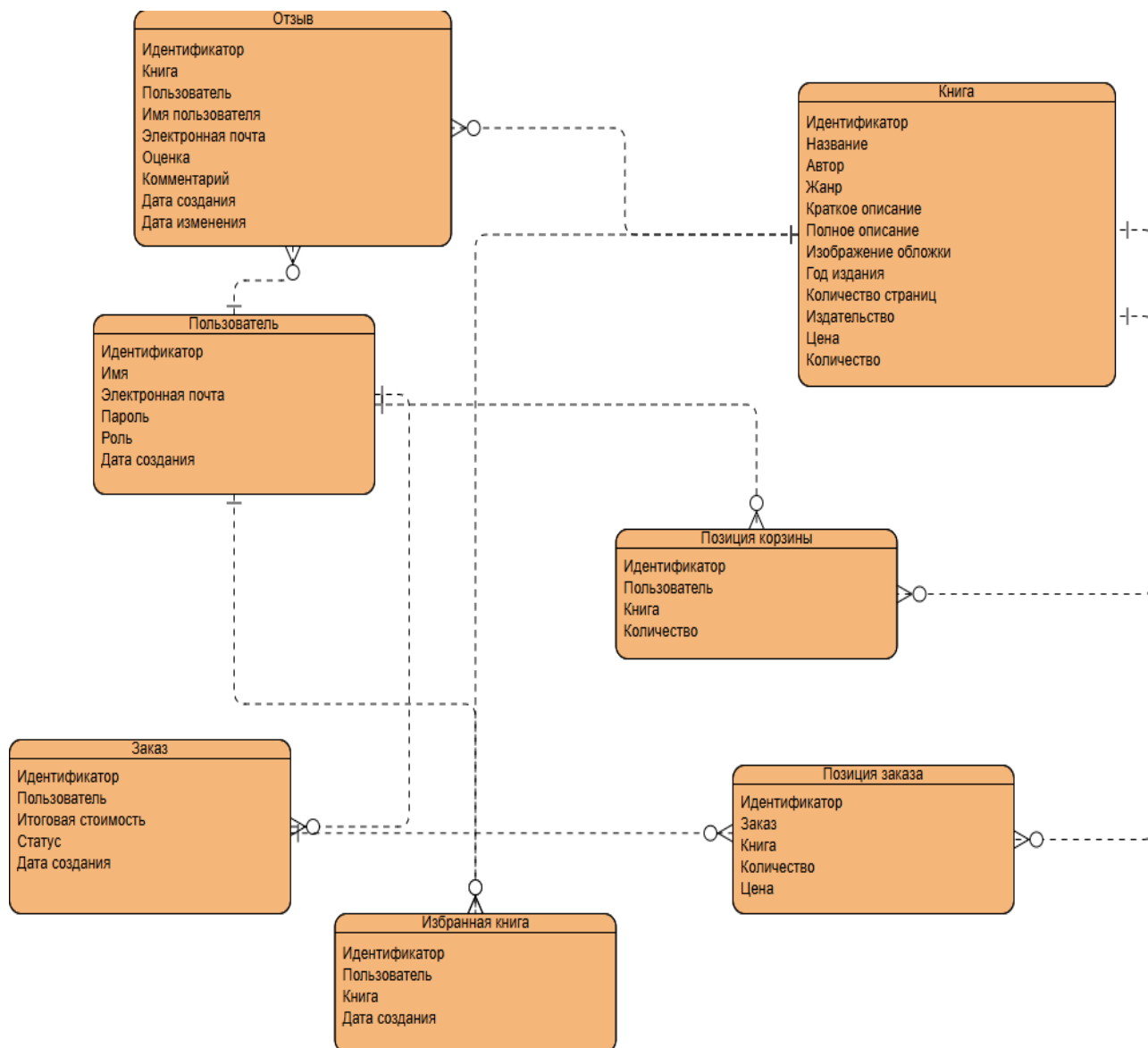


Рисунок 13 – ERD-диаграмма концептуальной модели базы данных

В таблице 7 представлено Описание сущностей представлено.

Таблица 7 – Описание сущностей

Название сущности	Описание	Категория
Пользователь	Хранит данные учетной записи пользователя	Стержневая
Книга	Хранит сведения о книге из каталога	Стержневая

Продолжение таблицы 7

Название сущности	Описание	Категория
Позиция корзины	Хранит книгу, добавленную пользователем в корзину	Характеристическая
Заказ	Хранит сведения об оформленном заказе	Стержневая
Позиция заказа	Хранит состав оформленного заказа	Характеристическая
Избранная книга	Хранит связь пользователя с избранной книгой	Характеристическая
Отзыв	Хранит оценку и комментарий пользователя к книге	Характеристическая

Также описание атрибутов, которые выделяют характеристики и свойства конкретной сущности, представлено в таблице 8.

Таблица 8 – Описание атрибутов

Название сущности	Атрибут	Описание атрибута	Ограничения
Пользователь	Идентификатор	Уникальный идентификатор пользователя	Первичный ключ
Пользователь	Имя	Имя пользователя	Не пустое значение
Пользователь	Электронная почта	Email для входа в систему	Уникальность
Пользователь	Пароль	Хеш пароля пользователя	Не хранится в открытом виде

Продолжение таблицы 8

Название сущности	Атрибут	Описание атрибута	Ограничения
Пользователь	Роль	Права пользователя в системе	user или admin
Пользователь	Дата создания	Дата регистрации пользователя	Не пустое значение
Книга	Идентификатор	Уникальный идентификатор книги	Первичный ключ
Книга	Название	Название книги	Не пустое значение
Книга	Автор	Автор книги	Не пустое значение
Книга	Жанр	Жанр книги	Не пустое значение
Книга	Краткое описание	Краткое описание книги в каталоге	Может отсутствовать
Книга	Полное описание	Описание книги на странице товара	Может отсутствовать
Книга	Изображение обложки	Ссылка на изображение книги	Может отсутствовать
Книга	Год издания	Год публикации книги	Может отсутствовать
Книга	Количество страниц	Объем книги в страницах	Может отсутствовать
Книга	Издательство	Название издательства	Может отсутствовать
Книга	Цена	Стоимость книги	Больше 0

Продолжение таблицы 8

Название сущности	Атрибут	Описание атрибута	Ограничения
Книга	Количество	Количество доступных экземпляров	Не меньше 0
Позиция корзины	Идентификатор	Уникальный идентификатор позиции корзины	Первичный ключ
Позиция корзины	Пользователь	Пользователь, добавивший книгу	Внешний ключ
Позиция корзины	Книга	Книга в корзине	Внешний ключ
Позиция корзины	Количество	Количество экземпляров	Больше 0
Заказ	Идентификатор	Уникальный идентификатор заказа	Первичный ключ
Заказ	Пользователь	Пользователь, оформивший заказ	Внешний ключ
Заказ	Итоговая стоимость	Общая сумма заказа	Больше 0
Заказ	Статус	Статус обработки заказа	Не пустое значение
Заказ	Дата создания	Дата оформления заказа	Не пустое значение
Позиция заказа	Идентификатор	Уникальный идентификатор позиции заказа	Первичный ключ

Продолжение таблицы 8

Название сущности	Атрибут	Описание атрибута	Ограничения
Позиция заказа	Заказ	Заказ, к которому относится позиция	Внешний ключ
Позиция заказа	Книга	Книга в составе заказа	Внешний ключ
Позиция заказа	Количество	Количество экземпляров	Больше 0
Позиция заказа	Цена	Цена книги при оформлении заказа	Больше 0
Избранная книга	Идентификатор	Уникальный идентификатор записи	Первичный ключ
Избранная книга	Пользователь	Пользователь, добавивший книгу в избранное	Внешний ключ
Избранная книга	Книга	Книга, добавленная в избранное	Внешний ключ
Избранная книга	Дата создания	Дата добавления книги в избранное	Не пустое значение
Отзыв	Идентификатор	Уникальный идентификатор отзыва	Первичный ключ
Отзыв	Книга	Книга, к которой относится отзыв	Внешний ключ
Отзыв	Пользователь	Автор отзыва	Внешний ключ
Отзыв	Имя пользователя	Имя автора отзыва	Не пустое значение

Окончание таблицы 8

Название сущности	Атрибут	Описание атрибута	Ограничения
Отзыв	Электронная почта пользователя	Email автора отзыва	Не пустое значение
Отзыв	Оценка	Оценка книги	От 1 до 5
Отзыв	Дата создания	Дата создания отзыва	Не пустое значение
Отзыв	Дата обновления	Дата последнего изменения отзыва	Не пустое значение

Описание связей между сущностями представлено в таблице 9.

Таблица 9 – Описание связей между сущностями

Сущность	Сущность	Описание связи	Степень связи	Обязательность
Пользователь	Позиция корзины	Пользователь может добавить несколько книг в корзину	1:M	Не обязательно
Книга	Позиция корзины	Книга может находиться в корзинах разных пользователей	1:M	Не обязательно
Пользователь	Заказ	Пользователь может оформить несколько заказов	1:M	Не обязательно
Заказ	Позиция заказа	Заказ содержит одну или несколько позиций	1:M	Обязательно
Книга	Позиция заказа	Книга может входить в разные заказы	1:M	Не обязательно

Продолжение таблицы 9

Сущность	Сущность	Описание связи	Степень связи	Обязательность
Пользователь	Избранная книга	Пользователь может добавить несколько книг в избранное	1:M	Не обязательно
Книга	Избранная книга	Книга может быть добавлена в избранное разными пользователями	1:M	Не обязательно
Пользователь	Отзыв	Пользователь может оставить отзывы к разным книгам	1:M	Не обязательно

Также стоит отметить, что интеллектуальный модуль не требует отдельной сущности в концептуальной модели. Он использует уже имеющиеся данные каталога: название, автора, жанр, описание и рейтинг книги. За счет этого модуль подбора работает с фактическими данными системы и не изменяет структуру базы данных.

Таким образом, концептуальная модель отражает основные данные, необходимые для работы книжного интернет-магазина. Она показывает связи между пользователями, книгами, корзиной, заказами, избранным и отзывами, а также создает основу для дальнейшего логического моделирования базы данных.

3.2.2 Логическое моделирование базы данных

Для построения логической модели базы данных были определены семь таблиц, которые участвуют в работе автоматизированной интеллектуальной системы продажи книг.

В соответствии с микросервисной архитектурой таблицы физически распределены по отдельным базам данных:

- bookstore_auth – таблица users;
- bookstore_catalog – таблицы books и reviews;
- bookstore_order – таблицы orders, order_items и cart_items;
- bookstore_profile – таблица favorites.

Связи между таблицами разных сервисов реализуются логически через UUID-идентификаторы, поскольку данные хранятся в отдельных базах.

Таблица users хранит учетные записи пользователей. Таблица books содержит все сведения о книгах каталога. Таблица reviews используется для хранения отзывов и оценок, на основе которых рассчитывается средний рейтинг книги. Таблицы orders и order_items фиксируют оформленные заказы и их состав, причем цена книги сохраняется на момент покупки. Таблица cart_items отвечает за содержимое корзины пользователя, а таблица favorites за список избранных книг.

Для таблиц favorites и reviews установлено ограничение уникальности по паре полей user_id и book_id. Это исключает возможность добавить одну и ту же книгу в избранное несколько раз или оставить несколько отзывов на одну книгу.

ERD-диаграмма модели базы данных представлена на рисунке 14.

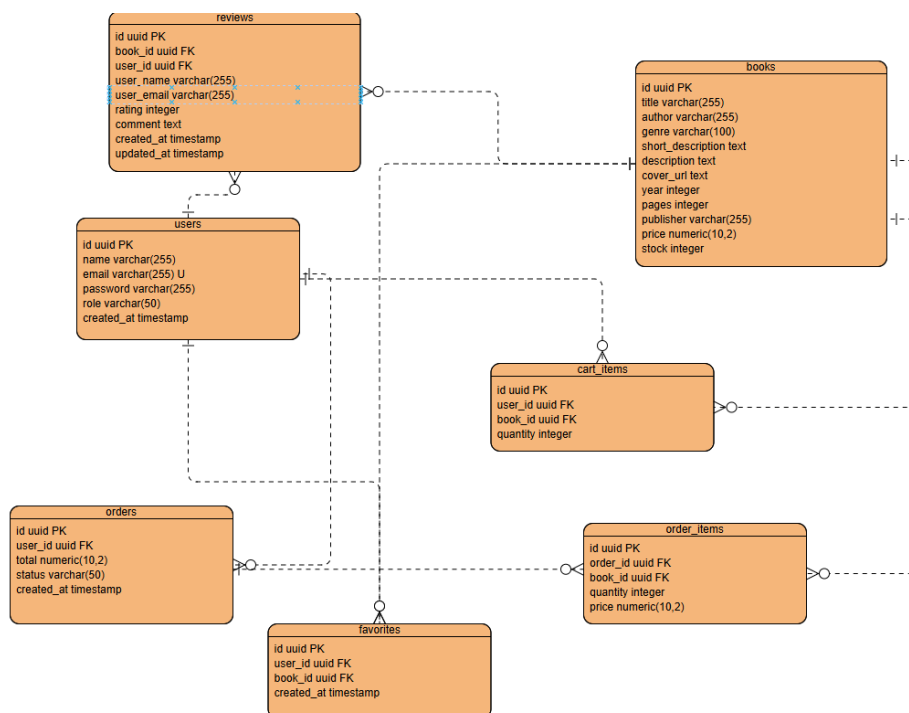


Рисунок 14 – ERD-диаграмма логической модели базы данных

В таблице 10 представлено описание таблиц логической модели базы данных.

Таблица 10 – Описание таблиц логической модели базы данных

Таблица	Назначение
users	Хранение данных пользователей
books	Хранение сведений о книгах каталога
cart_items	Хранение книг, добавленных в корзину
orders	Хранение оформленных заказов
order_items	Хранение состава заказа
favorites	Хранение избранных книг пользователя
reviews	Хранение отзывов и оценок книг

В таблице 11 представлено описание ключевых полей и связей логической модели базы данных.

Таблица 11 – Описание ключевых полей и связей логической модели базы данных

Таблица	Первичный ключ	Внешние ключи	Ограничения
users	id	–	email unique
books	id	–	price > 0, stock >= 0
cart_items	id	user_id, book_id	quantity > 0
orders	id	user_id	total > 0
order_items	id	order_id, book_id	quantity > 0, price > 0

Продолжение таблицы 11

Таблица	Первичный ключ	Внешние ключи	Ограничения
favorites	id	user_id, book_id	unique(user_id, book_id)
reviews	id	user_id, book_id	rating от 1 до 5, unique(user_id, book_id)

Такая структура логической модели полностью соответствует микросервисной архитектуре проекта и обеспечивает целостность данных при работе с каталогом, заказами, корзиной, избранным и отзывами.

3.3 Проектирование серверного программного обеспечения

3.3.1 Назначение и функции серверного приложения

Серверное приложение является важной частью автоматизированной интеллектуальной системы продажи книг. Оно принимает запросы от клиентского приложения, выполняет бизнес-логику, проверяет права доступа и обеспечивает взаимодействие с данными системы.

Серверная часть реализуется на языке Go. Обмен данными между клиентским приложением и сервером выполняется через REST API в формате JSON. Благодаря этому клиентская часть может получать данные каталога, отправлять запросы на оформление заказов, работать с профилем, отзывами и интеллектуальными рекомендациями через единый программный интерфейс.

В отличие от клиентской части, серверное приложение отвечает за обработку и проверку данных. На сервере выполняются операции авторизации, работа с каталогом книг, корзиной, заказами, профилем пользователя, избранными книгами, отзывами и рейтингами. Серверная часть передает запросы в интеллектуальный модуль, который подбирает книги по текстовому описанию пользователя.

В ходе проектирования серверного программного обеспечения была разработана диаграмма деятельности, представленная на рисунке 15.

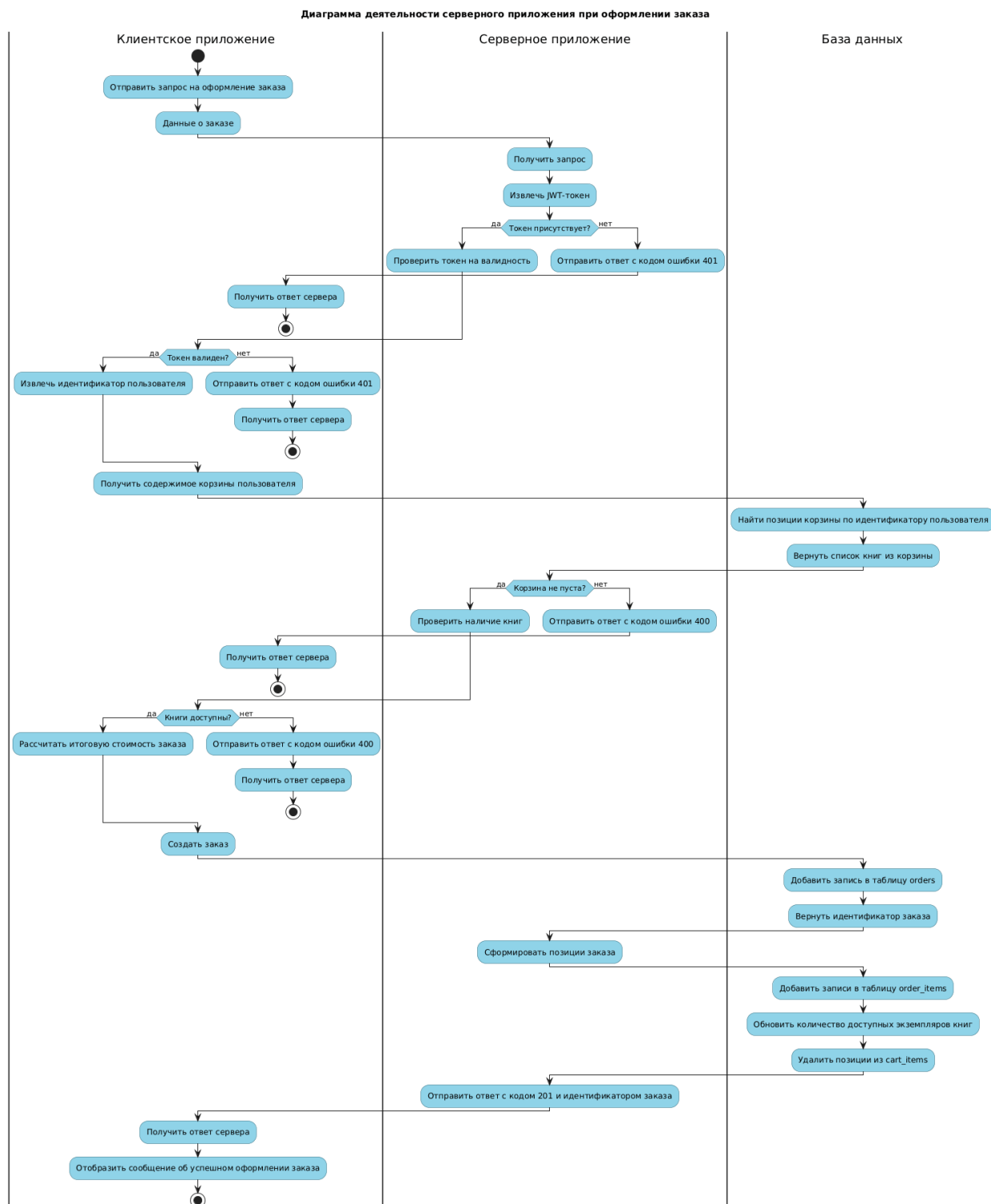


Рисунок 15 – Диаграмма деятельности серверного приложения при оформлении заказа

Диаграмма деятельности отображает обработку заказа на серверной стороне. Пользователь отправляет запрос на оформление заказа, сервер проверяет авторизацию, получает содержимое корзины, проверяет наличие книг, создает заказ, сохраняет позиции заказа и очищает корзину пользователя.

3.3.2 Проектирование архитектуры серверного приложения

При разработке серверного приложения планируется использовать принципы чистой архитектуры. Данный подход предполагает разделение приложения на несколько независимых уровней, каждый из которых отвечает за свою область ответственности.

Основная идея чистой архитектуры заключается в том, что бизнес-логика не должна зависеть от деталей реализации – таких как база данных, HTTP-фреймворк или внешние библиотеки. Это делает код более удобным для разработки и расширения.

В серверной части выделяются следующие уровни:

- уровень представления (handlers);
- уровень приложения (service);
- уровень домена (business logic);
- уровень инфраструктуры (repository, database, external services).

Уровень представления принимает HTTP-запросы и формирует ответы. Уровень приложения координирует выполнение операций. Уровень домена содержит основные бизнес-правила системы. Уровень инфраструктуры обеспечивает работу с базой данных и другими внешними компонентами.

Дополнительно серверная часть разделена на пять микросервисов: auth, catalog, order, profile и ai. Каждый сервис решает свою задачу и организован по тем же слоям чистой архитектуры. Такой подход позволяет изолировать разные части предметной области и уменьшить влияние изменений в одном сервисе на остальные.

3.3.3 Механизмы безопасности

Для ограничения доступа к функциям серверного приложения используется аутентификация на основе JWT-токенов.

JSON Web Token (JWT) – открытый стандарт для создания токенов доступа, основанный на формате JSON [10]. В разрабатываемой системе токен создается сервером после успешной авторизации пользователя и подписывается секретным ключом.

JWT-токен передается клиентскому приложению и используется при обращении к защищенным маршрутам. Без токена пользователь не может работать с корзиной, заказами, профилем, избранными книгами и отзывами.

В системе предусмотрено разделение прав доступа по ролям. Пользователь с ролью client получает доступ к функциям покупки книг, оформления заказов, работы с профилем и отзывами. Пользователь с ролью admin получает доступ к управлению каталогом книг, заказами и удалению пользовательских отзывов.

Важно отметить, что пароли пользователей не хранятся в базе данных в открытом виде. При регистрации пароль преобразуется с помощью криптографической хеш-функции. При авторизации сервер сравнивает введенный пароль с сохраненным хешем.

Соль – строка входных данных, которая передается хеш-функции вместе с паролем для вычисления хеша [11]. Использование хеширования снижает риск восстановления исходного пароля при утечке базы данных.

Защита данных также обеспечивается проверкой прав на уровне серверной части. Пользователь может изменять только собственный профиль, работать только со своей корзиной и заказами, редактировать и удалять только свои отзывы.

Для отзывов используется дополнительное ограничение: один пользователь может оставить только один отзыв на одну книгу. Оценка проверяется на сервере и должна находиться в диапазоне от 1 до 5.

Административные операции выполняются только после проверки роли пользователя. К таким операциям относятся добавление, редактирование и удаление книг, изменение статуса заказа и удаление отзывов других пользователей.

3.4 Проектирование клиентского программного обеспечения

3.4.1 Назначение и функции клиентского приложения

Клиентское приложение предназначено для взаимодействия пользователя с автоматизированной интеллектуальной системой продажи книг. Оно предоставляет веб-интерфейс, через который выполняются основные действия: просмотр каталога, работа с карточкой книги, корзиной, заказами, профилем, избранным, отзывами и интеллектуальным чат-помощником.

В отличие от серверной части, клиентское приложение не выполняет бизнес-логику напрямую, а отображает данные, передает действия пользователя на сервер и показывает результат выполнения операций. За счет этого пользователь получает удобный доступ к функциям системы через страницы и компоненты интерфейса.

Для обычного пользователя клиентская часть обеспечивает сценарии выбора и покупки книги, а также работу с рекомендациями. Пользователь может найти книгу через каталог или обратиться к чат-помощнику, получить подходящие варианты и перейти к карточке выбранного произведения.

Для администратора клиентское приложение предоставляет отдельные возможности управления системой. Оно позволяет добавлять и редактировать книги, изменять статус заказов и управлять отзывами пользователей через интерфейс приложения.

Клиентская часть обрабатывает три основных состояния интерфейса: загрузку, ошибки и успешное выполнение. Это реализовано на страницах каталога, оформления заказа, создания отзыва и в чат-помощнике.

3.4.2 Проектирование пользовательского интерфейса

Пользовательский интерфейс реализован в виде одностраничного веб-приложения. Такое решение позволяет сделать переходы между разделами без полной перезагрузки страницы. Это существенно повышает скорость работы и улучшает пользовательский опыт.

Структура интерфейса делится на 3 части:

- публичную;
- пользовательскую;
- административную.

К публичной части относятся страницы регистрации, авторизации, каталога и карточки книги. После входа в систему пользователю становятся доступны корзина, оформление заказа, профиль, избранное, история заказов, отзывы и интеллектуальный помощник. Административный раздел доступен только пользователю с ролью `admin`.

Основным элементом навигации является верхнее меню. Через него пользователю доступны все вышеперечисленные разделы. Такой подход делает их доступными из любой части приложения.

В интерфейсе используются карточки книг, формы, кнопки действий, модальные окна и сообщения о результате операции. Карточки применяются для отображения книг и рекомендаций, а формы – для регистрации, авторизации, редактирования профиля, добавления книги и создания отзыва.

Важным является отображение состояния выполняемых действий. При загрузке данных, отправке отзыва, оформлении заказа или обращении к интеллектуальному помощнику интерфейс должен показывать загрузку, ошибку или успешный результат. За счет этого пользователь понимает, что его действие было обработано системой.

Поля ввода проверяются на клиентской стороне до отправки запроса на сервер. Проверка применяется к электронной почте, паролю, количеству товара, цене, оценке и текстовым полям. Таким образом, пользовательский интерфейс

проектируется с учетом удобной навигации, разделения ролей и понятной обратной связи от системы.

3.4.3 Проектирование структуры клиентского приложения

Клиентское приложение проектируется как Single Page Application. Это означает, что основная страница загружается один раз, а дальнейшее изменение интерфейса происходит за счет обновления отдельных компонентов без полной перезагрузки.

Структура клиентской части строится по модульному принципу. Каждый модуль отвечает за свою задачу: маршрутизацию, отображение страниц, выполнение запросов к серверу, хранение состояния, проверку данных и работу с авторизацией.

В проектируемом клиентском приложении выделяются следующие модули:

- модуль маршрутизации;
- модуль страниц;
- модуль общих компонентов;
- сервисный слой;
- слой управления состоянием;
- модуль валидации;
- модуль авторизации.

Модуль маршрутизации определяет, какая страница должна отображаться при изменении адреса в браузере. Модуль страниц содержит основные экраны приложения: каталог, карточку книги, корзину, профиль, избранное, историю заказов, интеллектуального помощника и административную панель.

Модуль общих компонентов включает элементы, которые используются повторно: кнопки, поля ввода, модальные окна, индикаторы загрузки, сообщения об ошибках и компонент рейтинга. За счет этого интерфейс становится более единообразным, а код не дублируется в разных частях приложения.

Сервисный слой отвечает за обмен данными с серверной частью. В нем размещаются функции для обращения к сервисам auth, catalog, order, profile и ai. Слой управления состоянием хранит данные, которые нужны нескольким компонентам: токен авторизации, информацию о пользователе, содержимое корзины, избранные книги и результаты запросов.

Модуль авторизации хранит JWT-токен и ограничивает доступ к защищенным страницам. Модуль валидации проверяет данные форм до отправки на сервер, например электронную почту, пароль, количество товара, цену и текст отзыва.

Структура клиентского приложения представлена на рисунке 16.

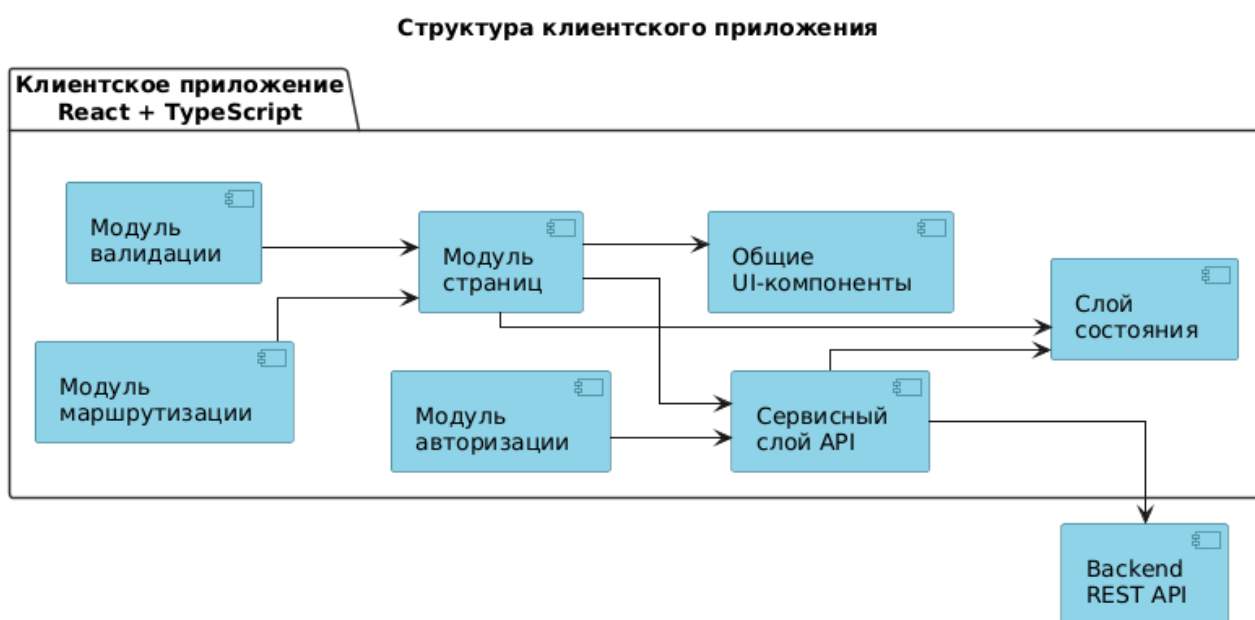


Рисунок 16 – Структура клиентского приложения

Подобная структура разделяет страницы, общие компоненты, API-запросы и состояние приложения. Это снижает связанность клиентского кода и упрощает добавление новых функций без изменения уже существующих компонентов.

3.5 Обоснование выбора средств разработки и технологий

Для разработки автоматизированной интеллектуальной системы продажи книг выбраны технологии, соответствующие веб-приложению с клиент-серверной и микросервисной архитектурой.

Серверная часть реализуется на языке программирования Go. Данный язык подходит для разработки сетевых сервисов, обработки HTTP-запросов и создания независимых серверных приложений. В проекте на Go реализуются сервисы auth, catalog, order, profile и ai.

Для хранения данных используется PostgreSQL. СУБД применяется для хранения пользователей, книг, заказов, корзины, избранных книг, отзывов и оценок. В проекте PostgreSQL разделен по базам данных сервисов, что соответствует микросервисной структуре серверной части.

Для работы с базой данных используется GORM. Библиотека применяется для описания моделей, подключения к PostgreSQL и автоматического создания таблиц через механизм миграций.

Клиентская часть разрабатывается с использованием React и TypeScript. React используется для построения интерфейса одностраничного приложения. TypeScript применяется для типизации компонентов, данных и API-запросов.

Для обмена данными между клиентским приложением и серверными сервисами используется REST API. Формат передачи данных – JSON. Такой формат применяется при регистрации, авторизации, загрузке каталога, оформлении заказа, создании отзыва и получении рекомендаций.

Для аутентификации используется JWT. После входа пользователь получает токен, который передается при обращении к защищенным маршрутам. Такой механизм применяется для корзины, заказов, профиля, избранного, отзывов и административных функций.

Для маршрутизации запросов используется Nginx. Он выполняет роль обратного прокси и направляет запросы клиентского приложения к нужному серверному сервису.

Для локального запуска компонентов применяется Docker Compose. В конфигурации объединяются frontend-контейнер, серверные сервисы, PostgreSQL, Kafka и вспомогательные компоненты.

Kafka используется как брокер сообщений для событийного взаимодействия. В проекте она применяется для обработки событий, связанных с заказами.

Выбранный набор технологий соответствует структуре проекта. Go используется для серверных сервисов, React и TypeScript – для клиентского приложения, PostgreSQL – для хранения данных, Docker Compose – для запуска компонентов, Nginx – для маршрутизации запросов, JWT – для защиты пользовательских функций.

4 Разработка программного обеспечения

4.1 Физическое моделирование базы данных

Физическая модель базы данных реализована в PostgreSQL. В проекте используется один Docker-контейнер PostgreSQL, внутри которого созданы отдельные базы данных для каждого микросервиса: `bookstore_auth`, `bookstore_catalog`, `bookstore_order` и `bookstore_profile`.

Такое разделение полностью соответствует микросервисной архитектуре системы. Каждый сервис работает только со своей базой данных.

Распределение таблиц по базам данных представлено в таблице 12.

Таблица 12 – Распределение таблиц по базам данных сервисов

База данных	Сервис	Таблицы
<code>bookstore_auth</code>	<code>auth</code>	<code>users</code>
<code>bookstore_catalog</code>	<code>catalog</code>	<code>books</code> , <code>reviews</code>
<code>bookstore_order</code>	<code>order</code>	<code>orders</code> , <code>order_items</code> , <code>cart_items</code>
<code>bookstore_profile</code>	<code>profile</code>	<code>favorites</code>

Физическая структура таблиц создается автоматически средствами GORM. При запуске каждого сервиса вызывается метод `AutoMigrate`, который создает и обновляет таблицы на основе Go-моделей.

Для сервиса `auth` выполняется миграция модели `User`. Для сервиса `catalog` – модели `Book` и `Review`. Для сервиса `order` – модели `Order`, `OrderItem` и `CartItem`. Для сервиса `profile` – модель `Favorite`.

Связи между данными разных сервисов реализованы через UUID-идентификаторы. Например, поле `user_id` в таблице `orders` ссылается на пользователя из сервиса `auth`, а поле `book_id` в таблице `reviews` – на книгу из сервиса `catalog`.

В таблицах `favorites` и `reviews` добавлены ограничения уникальности по паре (`user_id`, `book_id`). Это исключает повторное добавление одной и той же книги в избранное или повторный отзыв от одного пользователя. Поле `rating` в отзывах принимает значения только от 1 до 5.

4.2 Разработка серверного приложения

4.2.1 Структура серверного приложения

Серверное приложение реализовано на языке Go и разделено на пять прикладных сервисов: `auth-service`, `catalog-service`, `order-service`, `profile-service` и `ai-service`. Go отлично подходит для создания масштабируемых и производительных серверных приложений благодаря своей простоте, встроенной поддержке конкурентности и эффективной работе с сетевыми запросами [12].

Также в рабочее пространство входит общий модуль `shared`, в котором размещены повторно используемые компоненты.

Структура серверного приложения представлена на рисунке 17.

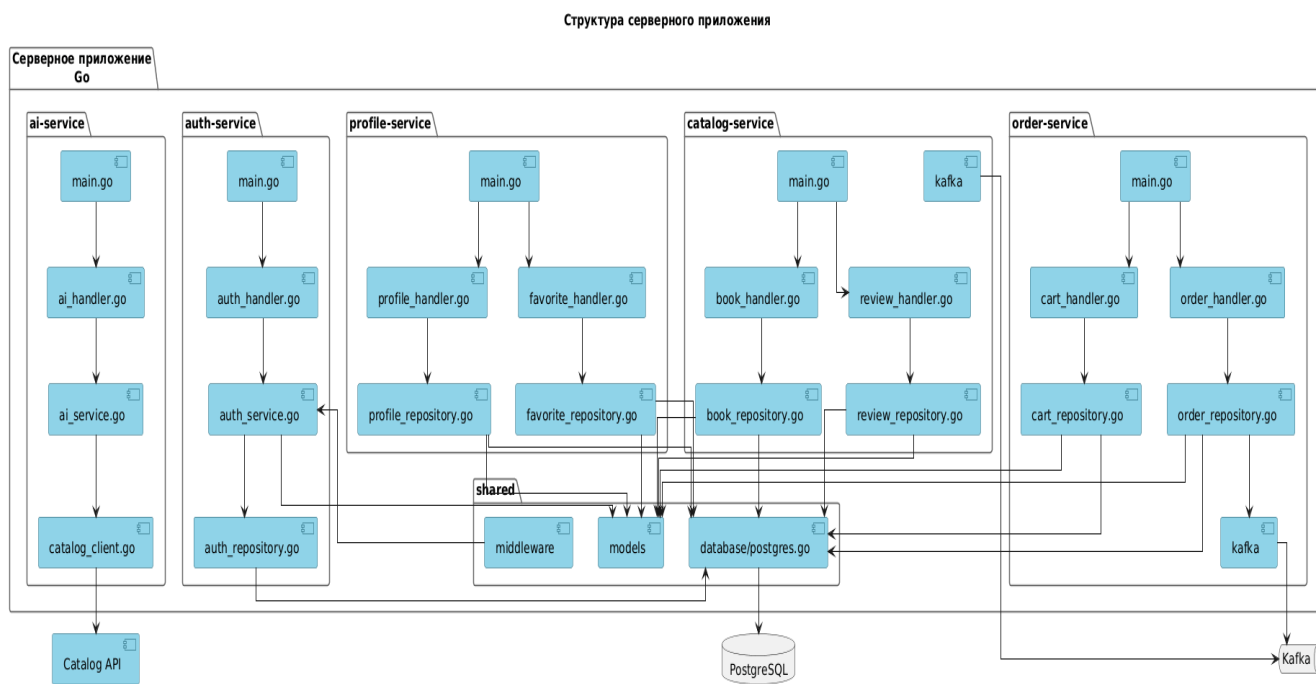


Рисунок 17 – Структура серверного приложения

Каждый сервис отвечает за отдельную часть предметной области. Auth-service обрабатывает регистрацию, вход в систему, данные пользователя и смену пароля. Catalog-service работает с каталогом книг, жанрами, карточками товаров, отзывами и рейтингами, а также выполняет административные операции с книгами. Order-service отвечает за корзину, оформление заказов, историю покупок и изменение статусов заказов. Profile-service объединяет функции, связанные с профилем пользователя и избранными книгами. Ai-service используется для интеллектуального подбора литературы по текстовому запросу пользователя.

В серверном приложении применяются два способа взаимодействия между компонентами. HTTP-запросы используются для синхронного обмена данными между сервисами, а Kafka применяется для передачи события после оформления заказа. Это необходимо, чтобы catalog-service мог обновить количество доступных экземпляров книги.

В таблице 13 представлено назначение компонентов серверного приложения.

Таблица 13 – Назначение компонентов серверного приложения

Сервис	Назначение	Основные компоненты
auth-service	Регистрация и авторизация пользователей	handler, service, repository
catalog-service	Каталог книг, отзывы и рейтинги	book handler, review handler, repositories
order-service	Корзина и заказы	cart handler, order handler, repositories
profile-service	Профиль пользователя и избранное	profile handler, favorite handler, repositories

Продолжение таблицы 13

Сервис	Назначение	Основные компоненты
ai-service	Интеллектуальный подбор книг	ai handler, ai service, catalog client
shared	Общие пакеты серверного приложения	middleware, models, database

Разделение серверной части по сервисам позволяет уменьшить связанность компонентов. Изменение логики корзины не требует изменения сервиса авторизации, а доработка интеллектуального помощника никак не затрагивает хранение заказов. Данная структура соответствует микросервисной архитектуре проекта и упрощает дальнейшее расширение системы [13].

4.2.2 Взаимодействие с базой данных

Взаимодействие серверных сервисов с PostgreSQL организовано через общий пакет shared/database. В нем находится файл postgres.go, который используется всеми сервисами для подключения к базе данных.

Для работы с PostgreSQL применяется библиотека GORM. Она используется в сервисах auth-service, catalog-service, order-service и profile-service. Сервис ai-service не имеет собственной базы данных и получает необходимые данные о книгах через catalog-service.

GORM – популярная ORM-библиотека для языка программирования Go, которая упрощает взаимодействие с реляционными базами данных. Она позволяет работать с таблицами базы данных через Go-структуры и автоматически выполнять миграции моделей. [14].

Каждый сервис работает со своей отдельной базой данных. Такое разделение полностью соответствует микросервисной архитектуре проекта.

Использование баз данных сервисами представлено в таблице 14.

Таблица 14 – Использование баз данных серверными сервисами

Сервис	База данных	Основные данные
auth-service	bookstore_auth	пользователи, роли, учетные данные
catalog-service	bookstore_catalog	книги, отзывы, рейтинги
order-service	bookstore_order	корзина, заказы, позиции заказов
profile-service	bookstore_profile	профиль пользователя, избранные книги
ai-service	не используется	данные получает через catalog-service

При запуске каждого сервиса выполняется автоматическая миграция соответствующих моделей. Ниже приведен пример миграции моделей в order-service.

Ниже приведен код в листинге 1, в котором выполняется подключение order-service к базе данных и миграция моделей заказов.

Листинг 1 – Подключение к базе данных и миграция моделей в order-service

```
database.InitDB("bookstore_order")
defer database.Close()

if err := database.DB.AutoMigrate(
    &models.Order{},
    &models.OrderItem{},
    &models.CartItem{},
); err != nil {
    log.Fatalf("Migration error: %v", err)
}
```

Сервис order-service использует базу bookstore_order. В миграцию передаются 3 модели: Order, OrderItem и CartItem. Эти структуры используются для хранения оформленных заказов, состава заказа и текущей корзины пользователя.

4.2.3 Реализация сервиса авторизации

Авторизация пользователей реализована в auth-service. Данный сервис работает с учетными записями пользователей и отвечает за регистрацию, вход в систему, получение профиля, изменение имени и смену пароля. Для хранения данных используется база данных bookstore_auth.

При запуске сервиса выполняется миграция модели User. Также создаются начальные учетные записи, которые используются для проверки пользовательского и административного доступа. Маршруты сервиса авторизации представлены в таблице 15.

Таблица 15 – Маршруты сервиса авторизации

Метод	Маршрут	Назначение	Доступ
GET	/health	Проверка работоспособности сервиса	публичный
POST	/auth/register	Регистрация пользователя	публичный
POST	/auth/login	Авторизация пользователя	публичный
POST	/auth/ping	Проверка ответа обработчика	публичный
GET	/auth/profile	Получение профиля пользователя	пользователь
PATCH	/auth/profile	Изменение имени пользователя	пользователь
PATCH	/auth/password	Изменение пароля	пользователь

Публичные маршруты используются для регистрации и входа. Маршруты профиля относятся к защищенным, поэтому для обращения к ним требуется JWT-токен.

Регистрация маршрутов сервиса авторизации представлена в листинге 2.

Листинг 2 – Регистрация маршрутов сервиса авторизации

```
authGroup := r.Group("/auth")
{
    authGroup.POST("/register", handler.Register)
    authGroup.POST("/login", handler.Login)
    authGroup.POST("/ping", handler.Ping)
}

authProtected := r.Group("/auth")
authProtected.Use(middleware.AuthRequired())
{
    authProtected.GET("/profile", handler.GetProfile)
    authProtected.PATCH("/profile", handler.UpdateProfile)
    authProtected.PATCH("/password", handler.UpdatePassword)
}
```

В коде выше маршруты разделены на 2 группы. Первая группа доступна без токена, а вторая использует `middleware.AuthRequired()`. За счет этого обработчики профиля вызываются только после проверки авторизации пользователя.

При регистрации пользователь передает имя, электронную почту и пароль. Входные данные проверяются перед созданием учетной записи: имя ограничивается по длине, электронная почта проверяется по формату, а пароль должен содержать не менее 8 символов. После успешной проверки создается пользователь с ролью `client`.

При входе сервис получает пользователя по email, сравнивает введенный пароль с сохраненным хешем и при успешной проверке формирует JWT-токен.

Код генерации токена приведен в листинге 3.

Листинг 3 – Генерация JWT-токена

```
func GenerateJWT(userID uuid.UUID, name, email, role string)
(string, error) {
    claims := models.UserClaims{
        UserID: userID,
```

```

        Name:    name,
        Email:   email,
        Role:    role,
        RegisteredClaims: jwt.RegisteredClaims{
            ExpiresAt: jwt.NewNumericDate(time.Now().Add(24 *
time.Hour)),
            IssuedAt:  jwt.NewNumericDate(time.Now()),
        },
    },
}

token := jwt.NewWithClaims(jwt.SigningMethodHS256, claims)
return token.SignedString(jwtSecret)
}

```

Токен содержит идентификатор пользователя, имя, email и роль. Срок действия токена составляет 24 часа. Эти данные позволяют другим сервисам определять пользователя и проверять его права доступа без повторной авторизации.

Важно отметить, что пароль пользователя не хранится в открытом виде. При регистрации он хешируется, а при входе сравнивается с сохраненным хешем через bcrypt. При ошибке авторизации сервис возвращает одинаковое сообщение для неверной почты и неверного пароля, чтобы не раскрывать наличие учетной записи в системе.

4.2.4 Реализация каталога книг, отзывов и рейтингов

Catalog-service отвечает за работу с книжным каталогом, отзывами и рейтингами. Через данный сервис клиентское приложение получает список книг, жанры, карточку выбранной книги, отзывы пользователей и среднюю оценку. Администратору доступны операции добавления, редактирования и удаления книг.

Маршруты сервиса каталога книг представлены в таблице 16.

Таблица 16 – Маршруты сервиса каталога книг

Метод	Маршрут	Назначение	Доступ
GET	/books	Получение списка книг	публичный

Продолжение таблицы 16

Метод	Маршрут	Назначение	Доступ
GET	/health	Проверка работоспособности сервиса	публичный
GET	/books/genres	Получение списка жанров	публичный
GET	/books/:id	Получение книги по идентификатору	публичный
GET	/books/:id/reviews	Получение отзывов книги	публичный
GET	/books/:id/rating	Получение рейтинга книги	публичный
POST	/books	Добавление книги	администратор
POST	/books/batch	Добавление списка книг	администратор
PUT	/books/:id	Редактирование книги	администратор
DELETE	/books/:id	Удаление книги	администратор
DELETE	/books/all	Удаление всех книг	администратор
POST	/api/catalog/books/:id/reviews	Создание отзыва	пользователь
PUT	/api/catalog/books/:id/reviews/:reviewId	Редактирование отзыва	пользователь

Окончание таблицы 16

Метод	Маршрут	Назначение	Доступ
DELETE	/api/catalog/books/:id/reviews/:reviewId	Удаление отзыва	пользователь или администратор
POST	/books/:id/decrease-stock	Уменьшение остатка книги	внутренний маршрут
POST	/books/:id/increase-stock	Увеличение остатка книги	внутренний маршрут

Разделение маршрутов в catalog-service сделано так, чтобы просмотр книг и отзывов был доступен без авторизации, а любые изменения каталога (добавление, редактирование и удаление книг) – только пользователю с ролью администратора.

В листинге 4 приведен код регистрации маршрутов.

Листинг 4 – Регистрация маршрутов сервиса каталога

```
// Публичные роуты (без JWT)
public := r.Group("/books")
{
    public.GET("", handler.GetBooks)
    public.GET("/genres", handler.GetGenres)
    public.GET("/:id", handler.GetBookByID)
    public.GET("/:id/reviews", handler.GetBookReviews)
    public.GET("/:id/rating", handler.GetBookRating)
}

// Внутренний роут для сервис-к-сервис вызовов (защищен API
ключом)
internal := r.Group("/books")
internal.Use(middleware.InternalOnly())
{
    internal.POST("/:id/decrease-stock", handler.DecreaseStock)
    internal.POST("/:id/increase-stock", handler.IncreaseStock)
}

// Защищенные роуты (только для админов)
admin := r.Group("/books")
admin.Use(middleware.AuthRequired())
```

```

admin.Use(middleware.AdminOnly())
{
    admin.POST("", handler.CreateBook)
    admin.POST("/batch", handler.CreateBooksBatch)
    admin.PUT("/:id", handler.UpdateBook)
    admin.DELETE("/all", handler.DeleteAllBooks)
    admin.DELETE("/:id", handler.DeleteBook)
}

reviews := r.Group("/api/catalog/books")
reviews.Use(middleware.AuthRequired())
{
    reviews.POST("/:id/reviews", handler.CreateReview)
    reviews.PUT("/:id/reviews/:reviewId", handler.UpdateReview)
    reviews.DELETE("/:id/reviews/:reviewId", handler.DeleteReview)
}

```

Доступ к маршрутам определяется через `middleware`. Для действий со стороны администратора используются проверки `AuthRequired` и `AdminOnly`, а для внутренних операций с остатками применяется `InternalOnly`.

Метод получения списка книг поддерживает полнотекстовый поиск, фильтрацию по жанру и автору, различные варианты сортировки и пагинацию. Код этого метода представлен в листинге 5.

В листинге 5 представлен код фильтрации и сортировки каталога.

Листинг 5 – Фильтрация и сортировка книг

```

func (r *BookRepository) GetFiltered(search, genre, author, sort
string, page, limit int) ([]*models.Book, int64, error) {
    query := r.applyFilters(database.DB.Model(&models.Book{}),
search, genre, author)

    var total int64
    if err := query.Count(&total).Error; err != nil {
        return nil, 0, err
    }

    dataQuery := r.applyFilters(r.booksWithRatingQuery(), search,
genre, author)

    switch sort {
    case "price_asc":
        dataQuery = dataQuery.Order("price ASC")
    case "price_desc":
        dataQuery = dataQuery.Order("price DESC")
    case "title_asc":
        dataQuery = dataQuery.Order("title ASC")
    }
}

```

```

    case "title_desc":
        dataQuery = dataQuery.Order("title DESC")
    default:
        dataQuery = dataQuery.Order("title ASC")
}

if page < 1 {
    page = 1
}
if limit < 1 || limit > 100 {
    limit = 12
}

offset := (page - 1) * limit
dataQuery = dataQuery.Offset(offset).Limit(limit)

var books []*models.Book
if err := dataQuery.Find(&books).Error; err != nil {
    return nil, 0, err
}

return books, total, nil
}

```

Данный метод принимает параметры `search`, `genre`, `author`, `sort`, `page` и `limit`. Сначала определяется общее количество найденных книг, затем применяется сортировка и ограничение выборки по страницам.

Отзывы содержат оценку от 1 до 5 и текстовый комментарий. Средний рейтинг рассчитывается по оценкам из отзывов, поэтому отдельная таблица рейтинга не используется.

После оформления заказа `order-service` передает событие через `Kafka`. `Catalog-service` получает его и уменьшает количество доступных экземпляров книг. Благодаря этому обновление остатков выполняется без участия клиентского приложения.

Таким образом, в `catalog-service` реализована работа с книгами, отзывами и рейтингами на уровне маршрутов, репозиториях и проверок доступа. Публичные операции доступны без авторизации, пользовательские действия требуют JWT-токен, административные маршруты ограничены ролью `admin`, а изменение остатков выполняется через внутренние механизмы взаимодействия сервисов.

4.2.5 Реализация сервиса корзины и заказов

В order-service реализована обработка корзины и заказов. В данном разделе рассматриваются маршруты сервиса, проверка корзины перед оформлением заказа, расчет итоговой суммы и отправка события после успешного создания заказа.

В корзине хранится идентификатор книги и количество экземпляров. Важно обратить внимание, что цена книги не передается с клиентской части при оформлении заказа. Перед созданием заказа сервис получает актуальные данные о книге из каталога, проверяет остаток и рассчитывает итоговую сумму на серверной стороне.

В таблице 17 представлены маршруты сервиса корзины и заказов.

Таблица 17 – Маршруты сервиса корзины и заказов

Метод	Маршрут	Назначение	Доступ
GET	/health	Проверка работоспособности сервиса	публичный
POST	/api/cart/add	Добавление книги в корзину	пользователь
GET	/api/cart	Получение корзины пользователя	пользователь
PUT	/api/cart/:book_id	Изменение количества книги в корзине	пользователь
DELETE	/api/cart/:book_id	Удаление книги из корзины	пользователь
POST	/api/orders/checkout	Оформление заказа	пользователь

Продолжение таблицы 17

Метод	Маршрут	Назначение	Доступ
GET	/api/orders	Получение заказов пользователя	пользователь
GET	/api/orders/admin/all	Получение всех заказов	администратор
PUT	/api/orders/admin/:id/status	Изменение статуса заказа	администратор
DELETE	/api/orders/admin/:id	Удаление заказа	администратор
GET	/internal/orders/can-review	Проверка покупки книги пользователем	внутренний маршрут

Пользовательские операции требуют JWT-токен. Административные маршруты дополнительно проверяют роль пользователя, а внутренний маршрут /internal/orders/can-review используется catalog-service для проверки права на создание отзыва.

В листинге 6 приведен код настройки маршрутов order-service.

Листинг 6 – Регистрация маршрутов сервиса корзины и заказов

```
api := r.Group("/api")
api.Use(middleware.AuthRequired())
{
    cart := api.Group("/cart")
    {
        cart.POST("/add", handler.AddToCart)
        cart.GET("", handler.GetCart)
        cart.PUT("/:book_id", handler.UpdateCart)
        cart.DELETE("/:book_id", handler.RemoveFromCart)
    }

    orders := api.Group("/orders")
    {
        orders.POST("/checkout", handler.Checkout)
        orders.GET("", handler.GetUserOrders)
    }
}
```

```

        orders.GET("/admin/all", middleware.AdminOnly(),
handler.GetAllOrders)
        orders.PUT("/admin/:id/status", middleware.AdminOnly(),
handler.UpdateOrderStatus)
        orders.DELETE("/admin/:id", middleware.AdminOnly(),
handler.DeleteOrder)
    }
}

internal := r.Group("/internal/orders")
internal.Use(middleware.InternalOnly())
{
    internal.GET("/can-review", handler.CanReviewBook)
}

```

В листинге 6 маршруты разделены на три группы. Группа `/api/cart` отвечает за операции с корзиной, `/api/orders` – за оформление и просмотр заказов, а `/internal/orders` содержит сервисный маршрут, который не вызывается напрямую клиентским приложением.

Оформление заказа начинается с получения корзины пользователя. Если корзина пуста, сервис возвращает ошибку. Для каждой позиции запрашиваются данные книги из каталога, после чего проверяется остаток и рассчитывается итоговая сумма заказа.

Ниже в листинге 7 приведен основной участок оформления заказа.

Листинг 7 – Проверка корзины и расчет суммы заказа

```

cartItems := cartRepo.Get(userID)
if len(cartItems) == 0 {
    c.JSON(http.StatusBadRequest, gin.H{"error": "Корзина пуста"})
    return
}

total := 0.0
orderItems := []kafka.OrderItemEvent{}

for _, item := range cartItems {
    book, err := getCatalogBook(item.BookID.String())
    if err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": "Товар не
найден в каталоге"})
        return
    }

    if book.Stock < item.Quantity {
        c.JSON(http.StatusBadRequest, gin.H{

```

```

        "error":      fmt.Sprintf("Недостаточно товара '%s' на
складе", book.Title),
        "book_title": book.Title,
        "available":  book.Stock,
        "requested":  item.Quantity,
    })
    return
}

total += book.Price * float64(item.Quantity)

orderItems = append(orderItems, kafka.OrderItemEvent{
    BookID:    item.BookID.String(),
    Quantity:  item.Quantity,
    Price:     book.Price,
})
}

```

Вышеприведенный код показывает, что итоговая сумма рассчитывается только на сервере. Клиент передает идентификатор книги и количество, но не передает цену. Это исключает возможность изменить стоимость заказа через клиентский запрос.

После проверки всех позиций создается заказ со статусом `new`. В заказ сохраняются пользователь, итоговая сумма, дата создания и список позиций. Каждая позиция содержит идентификатор книги, количество и цену на момент оформления.

После успешного создания заказа корзина пользователя очищается, а сервис формирует событие Kafka с данными заказа. Для администратора предусмотрено изменение статуса заказа. Допустимыми значениями являются `new`, `paid`, `shipped`, `delivered` и `cancelled`. При передаче другого значения обработчик возвращает ошибку запроса.

4.2.6 Реализация сервиса профиля пользователя

Profile-service реализует операции, связанные с профилем пользователя и избранными книгами. В данном разделе рассматриваются маршруты сервиса, получение данных из других компонентов и хранение избранного.

Для хранения данных используется база `bookstore_profile`. В ней сохраняются только записи избранных книг. Данные учетной записи не

дублируются, так как имя, email и роль пользователя запрашиваются из auth-service.

В таблице 18 представлены основные операции сервиса профиля.

Таблица 18 – Маршруты сервиса профиля пользователя

Метод	Маршрут	Назначение	Доступ
GET	/health	Проверка работоспособности сервиса	публичный
GET	/profile	Получение профиля пользователя	пользователь
GET	/favorites	Получение списка избранных книг	пользователь
POST	/favorites	Добавление книги в избранное	пользователь
DELETE	/favorites/:book_id	Удаление книги из избранного	пользователь
GET	/recommendations	Получение рекомендаций для профиля	пользователь

Маршруты профиля закрыты через JWT-токен. Идентификатор пользователя извлекается из токена, поэтому клиент не передает user_id в теле запроса или параметрах URL.

Ниже в листинге 8 представлена регистрация защищенных маршрутов профиля.

Листинг 8 – Регистрация маршрутов сервиса профиля

```
protected := r.Group("/")
protected.Use(middleware.AuthRequired())
{
```

```

protected.GET("/profile", profileHandler.GetProfile)
protected.GET("/favorites", profileHandler.GetFavorites)
protected.POST("/favorites", profileHandler.AddFavorite)
protected.DELETE("/favorites/:book_id",
profileHandler.DeleteFavorite)
protected.GET("/recommendations",
profileHandler.GetRecommendations)
}

```

Все маршруты группы `protected` защищены middleware `AuthRequired`. После успешной проверки JWT-токена пользователь получает доступ к своему профилю, списку избранных книг и персональным рекомендациям.

При запросе профиля сервис собирает информацию из нескольких источников: основные данные учетной записи запрашиваются из `auth-service`, количество заказов — из `order-service`, а количество избранных книг — из собственной базы. В ответ возвращается обобщенная информация и статистика по активности пользователя.

При добавлении книги в избранное сервис проверяет корректность UUID и запрашивает существование книги в `catalog-service`. Если книга найдена, создается запись связи. При получении списка избранного сервис берет сохраненные идентификаторы и запрашивает по ним актуальные данные книг из каталога.

`profile-service` выступает агрегатором персональной информации пользователя. Он не хранит и не изменяет данные каталога, заказов или учетных записей, а только объединяет информацию из разных сервисов для удобного отображения в клиентском приложении

4.2.7 Реализация интеллектуального сервиса рекомендаций

Интеллектуальный подбор книг реализован в `ai-service`. Собственная база данных в сервисе не используется.

Для обработки естественного языка и генерации рекомендаций с пояснениями применяется модель OpenAI GPT-4o. Выбор обусловлен высоким качеством понимания русского языка, способностью к рассуждениям и удобным

API. Взаимодействие с моделью осуществляется через официальный OpenAI SDK.

Ai-service получает данные о книгах из catalog-service, а текстовый ответ формирует через LLM-провайдера. В конфигурации задаются модель, API-ключ, тайм-аут, максимальное количество кандидатов и размер ответа.

Ниже в листинге 9 представлена настройка маршрута интеллектуального сервиса.

Листинг 9 – Регистрация маршрута интеллектуального сервиса

```
r.GET("/health", func(c *gin.Context) {
    c.JSON(http.StatusOK, gin.H{"status": "ok", "service": "ai"})
})

api := r.Group("/api/ai")
api.Use(middleware.AuthRequired())
{
    api.POST("/chat", aiHandler.Chat)
}
```

Основной маршрут `/api/ai/chat` защищен аутентификацией. Проверочный маршрут `/health` доступен без токена.

Обработка запроса чат-помощника выполняется в методе `Chat`, код которого приведен в листинге 10.

Листинг 10 – Обработка запроса интеллектуального помощника

```
func (h *AIHandler) Chat(c *gin.Context) {
    var request service.ChatRequest
    if err := c.ShouldBindJSON(&request); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "Неверный
формат запроса"})
        return
    }

    response, err := h.aiService.Chat(c.Request.Context(),
request)
    if err != nil {
        log.Printf("Ошибка AI-чата: %v", err)

        status := http.StatusBadGateway
        message := "AI-помощник временно недоступен. Попробуйте
еще раз позже."

        if errors.Is(err, context.DeadlineExceeded) {
            status = http.StatusGatewayTimeout
```

```

        message = "AI-помощник не успел ответить вовремя.
        Попробуйте сформулировать запрос чуть короче."
    }

    c.JSON(status, gin.H{"error": message})
    return
}

c.JSON(http.StatusOK, response)
}

```

В листинге 10 показана базовая обработка запроса. Сначала проверяется структура входных данных, после чего запрос передается в ai-service. При ошибке внешнего провайдера обработчик возвращает понятное сообщение, чтобы пользователь видел причину недоступности функции.

Входные и выходные данные интеллектуального сервиса представлены в таблице 19.

Таблица 19 – Структура данных интеллектуального сервиса

Структура	Поле	Назначение	Структура
ChatRequest	message	Текстовый запрос пользователя	ChatRequest
ChatRequest	userId	Идентификатор пользователя	ChatRequest
ChatRequest	history	История сообщений	ChatRequest
ChatResponse	answer	Текстовый ответ помощника	ChatResponse
ChatResponse	books	Список рекомендованных книг	ChatResponse
ChatResponse	clarifyingQuestion	Уточняющий вопрос при широком запросе	ChatResponse

Продолжение таблицы 19

Структура	Поле	Назначение	Структура
ResponseBook	id	Идентификатор книги	ResponseBook
ResponseBook	title	Название книги	ResponseBook
ResponseBook	author	Автор книги	ResponseBook
ResponseBook	genre	Жанр книги	ResponseBook
ResponseBook	reason	Причина рекомендации	ResponseBook
ResponseBook	rating_average	Средний рейтинг книги	ResponseBook
ResponseBook	reviews_count	Количество отзывов	ResponseBook

Перед подбором книг сервис определяет тип запроса. В коде предусмотрены варианты: запрос о конкретной книге, запрос рекомендации, уточняющий вопрос, обычный разговор, сообщение вне книжной тематики и слишком общий запрос. Если запрос слишком общий, сервис не подбирает случайные книги, а возвращает уточняющий вопрос. Например, пользователь может уточнить настроение, жанр, темп или автора.

Для поиска кандидатов сервис обращается к catalog-service через HTTP-клиент. Из каталога возвращаются книги с названием, автором, жанром, описанием, ценой, остатком, средним рейтингом и количеством отзывов.

После получения кандидатов сервис формирует запрос к LLM-провайдеру.

В ответе ожидается текстовое объяснение и список рекомендаций с идентификаторами книг. Далее результат сопоставляется с найденными книгами каталога. За счет этого пользователю не возвращаются книги, которых нет в системе.

В результате ai-service выполняет подбор книг, формирует пояснения и возвращает ответ клиентскому приложению, не изменяя данные каталога, корзины, заказов или профиля.

4.3 Разработка клиентского приложения

4.3.1 Структура клиентского приложения

Клиентское приложение реализовано в директории frontend и представляет собой веб-интерфейс автоматизированной интеллектуальной системы продажи книг. Оно разработано на React и TypeScript, а сборка проекта выполняется с использованием Vite.

Структура клиентского приложения построена по модульному принципу. Основной код размещается в директории src, где отдельно выделены страницы, общие компоненты, контекст авторизации, API-клиент, типы данных и вспомогательные функции.

Входной точкой приложения является файл main.tsx. Он подключает корневой компонент App и запускает клиентское приложение в браузере. В App.tsx задается маршрутизация между страницами и общий каркас интерфейса.

Страницы приложения размещены в директории pages. В ней находятся компоненты Home, Catalog, BookDetail, Cart, Orders, Profile, AIAssistant, AdminPanel, Login и Register. Такое разделение позволяет хранить основные пользовательские сценарии отдельно друг от друга.

Общие элементы интерфейса вынесены в директорию components. В ней находятся Header, Notification, PrivateRoute и StarRating. Header отвечает за верхнее меню, Notification используется для отображения сообщений пользователю, PrivateRoute ограничивает доступ к защищенным страницам, а StarRating применяется при отображении и выставлении оценки книге.

Работа с серверной частью находится в файле api/client.ts. В нем создается экземпляр axios и описываются группы запросов: authAPI, booksAPI, cartAPI, orderAPI, profileAPI и aiAPI.

На рисунке 18 изображена диаграмма компонентов, которая отображает структуру клиентского приложения.

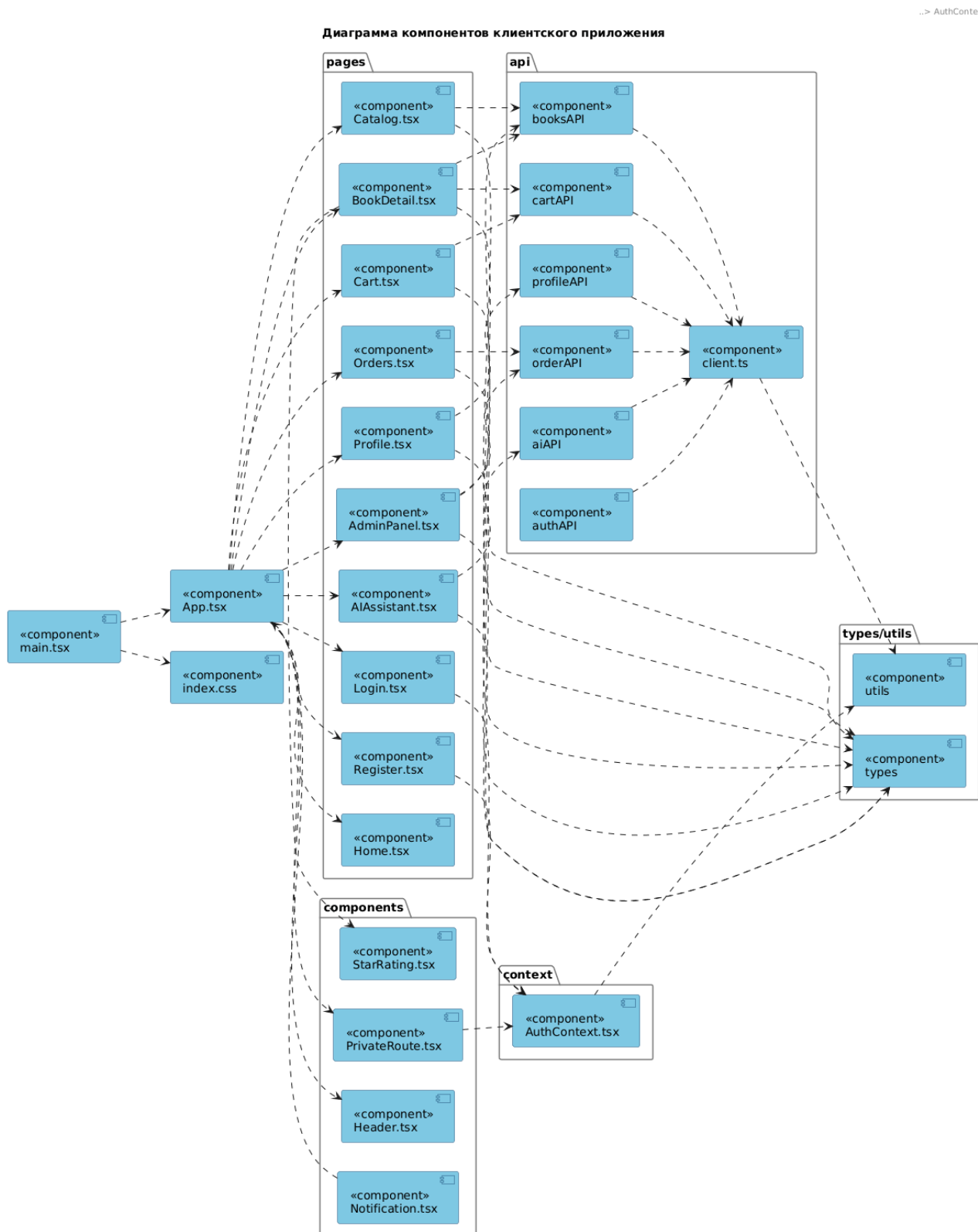


Рисунок 18 – Диаграмма компонентов клиентского приложения

В таблице 20 представлено описание основных компонентов клиентского приложения.

Таблица 20 – Назначение компонентов клиентского приложения

Файл или модуль	Назначение
main.tsx	Входная точка клиентского приложения. Подключает корневой компонент и запускает React-приложение в браузере.
App.tsx	Определяет основные маршруты приложения и подключает страницы. В нем также используется общий компонент верхнего меню и защищенные маршруты.
index.css	Содержит общие стили клиентского приложения.
pages/Home.tsx	Главная страница приложения. Используется как начальная точка для пользователя.
pages/Catalog.tsx	Страница каталога книг. Отвечает за отображение списка книг, поиск, фильтрацию и сортировку.
pages/BookDetail.tsx	Страница карточки книги. Отображает подробные сведения о книге, отзывы, рейтинг и действия пользователя.
pages/Cart.tsx	Страница корзины. Позволяет просматривать добавленные книги, изменять количество и удалять позиции.
pages/Orders.tsx	Страница заказов. Отображает историю заказов пользователя.
pages/Profile.tsx	Страница профиля пользователя. Отображает данные учетной записи, избранные книги и связанную статистику.

Продолжение таблицы 20

Файл или модуль	Назначение
pages/AIAssistant.tsx	Страница интеллектуального чат-помощника. Отправляет текстовый запрос пользователя и отображает рекомендации книг.
pages/AdminPanel.tsx	Административная панель. Используется для управления каталогом, заказами и пользовательскими отзывами.
pages/Login.tsx	Страница входа пользователя в систему.
pages/Register.tsx	Страница регистрации нового пользователя.
components/Header.tsx	Верхнее меню приложения. Обеспечивает переход между основными разделами.
components/PrivateRoute.tsx	Компонент защиты маршрутов. Ограничивает доступ к страницам, требующим авторизации.
components/Notification.tsx	Компонент отображения уведомлений, ошибок и сообщений об успешном выполнении действия.
components/StarRating.tsx	Компонент отображения и выбора оценки книги в виде звездного рейтинга.
components/profile	Набор компонентов, связанных с отображением данных профиля пользователя.
context/AuthContext.tsx	Хранит состояние авторизации: пользователя, токен, признак входа в систему и роль администратора.
api/client.ts	Единый API-клиент на основе axios. Добавляет JWT-токен в запросы, обрабатывает ошибку авторизации и содержит группы запросов к серверным сервисам.
authAPI	Группа запросов для регистрации, входа, получения профиля и смены пароля.

Окончание таблицы 20

Файл или модуль	Назначение
booksAPI	Группа запросов для каталога книг, жанров, карточки книги, отзывов, рейтингов и административных операций с книгами.
cartAPI	Группа запросов для добавления книги в корзину, получения корзины, изменения количества и удаления позиции.
orderAPI	Группа запросов для оформления заказа, получения истории заказов и административного управления заказами.
profileAPI	Группа запросов для профиля пользователя, избранных книг и рекомендаций профиля.
aiAPI	Группа запросов для обращения к интеллектуальному чат-помощнику.
types	Модуль с типами данных, которые используются страницами, компонентами и API-клиентом.
utils/authStorage	Вспомогательные функции для хранения, чтения и удаления данных авторизации.

Диаграмма отражает структуру клиентской части, где `main.tsx` запускает приложение, `App.tsx` связывает страницы и маршруты, а отдельные страницы обращаются к серверным сервисам через API-клиент. Также важным элементом является `AuthContext`, который хранит данные авторизации и используется защищенными маршрутами, верхним меню и страницами входа.

4.3.2 Разработка пользовательского интерфейса

Пользовательский интерфейс клиентского приложения разработан с учетом основных сценариев работы книжного интернет-магазина. Он должен быть логичным, понятным и удобным для пользователя [15].

В приложении реализованы следующие страницы: страницы регистрации и авторизации, главная страница с каталогом книг, карточка книги, корзина и оформление заказа, профиль пользователя, избранные книги, история заказов, интеллектуальный чат-помощник и административная панель.

При первом открытии веб-приложения пользователь увидит страницу регистрации.

Внешний вид страницы регистрации представлен на рисунке 19.

The image shows a user registration page for a bookstore application. The page is divided into two main sections: a left sidebar and a right main content area. The sidebar contains a header with a star icon and the text 'НОВЫЙ ЧИТАТЕЛЬ', a login section with a circular icon and the text 'РЕГИСТРАЦИЯ Войти в мир Laga', and a section titled 'ПОСЛЕ РЕГИСТРАЦИИ' with text about the catalog and profile. The main content area is titled 'СОЗДАНИЕ АККАУНТА' and 'Присоединяйтесь к сообществу'. It contains a registration form with fields for name, email, password, and password confirmation, followed by a 'Зарегистрироваться' button. At the bottom, there are two buttons: 'У меня уже есть аккаунт' and 'Вернуться в каталог'. Numbered callouts 1 through 7 point to specific elements: 1 points to the name input field, 2 points to the email input field, 3 points to the password input field, 4 points to the password confirmation input field, 5 points to the registration button, 6 points to the 'У меня уже есть аккаунт' button, and 7 points to the 'Вернуться в каталог' button.

Рисунок 19 – Страница регистрации пользователя

Основные элементы страницы:

- 1 – поле ввода имени пользователя;
- 2 – поле ввода электронной почты пользователя;
- 3 – поле ввода пароля;
- 4 – поле подтверждения пароля;

5 – кнопка регистрации;

6 – кнопка перехода на страницу входа;

7 – кнопка перехода в каталог.

Страница регистрации используется для создания новой учетной записи. Пользователь вводит имя, электронную почту и пароль. После заполнения формы данные отправляются на сервер.

На рисунке 20 представлен внешний вид страницы авторизации.

The image shows a user authentication interface. On the left, there's a section titled 'ВОЗВРАЩЕНИЕ В LARA' with a 'ЛИЧНЫЙ ВХОД' button and a heading 'Снова к своим книгам'. Below this is a paragraph about continuing a conversation with an AI assistant. Further down are two boxes: 'ПРОФИЛЬ' (Profile) and 'ПОМОЩНИК' (Assistant). At the bottom left are two buttons: 'Каталог и корзина всегда под рукой' and 'Сменить пароль можно прямо в профиле'. On the right, the 'ВХОД' (Login) section is titled 'Добро пожаловать обратно' and asks the user to enter their email and password. It includes input fields for 'EMAIL' (containing 'your@email.com') and 'ПАРОЛЬ' (password, masked with dots). Below these is a large orange 'Войти' (Login) button. A note below the button says: 'Если не получается войти, проверьте email, раскладку клавиатуры и регистр символов в пароле. Пароль можно изменить в профиле после входа.' At the bottom right are two buttons: 'Создать аккаунт' (Create account) and 'Вернуться в каталог' (Return to catalog). Numbered callouts point to specific elements: 1 points to the email input field, 2 points to the password input field, 3 points to the 'Войти' button, 4 points to the 'Создать аккаунт' button, and 5 points to the 'Вернуться в каталог' button.

Рисунок 20 – Страница авторизации пользователя

Основные компоненты страницы авторизации:

1 – поле ввода имени пользователя;

2 – поле ввода пароля;

3 – кнопка входа в клиентское приложение;

4 – кнопка перехода на страницу регистрации.

5 – кнопка возвращения в каталог

Страница авторизации предназначена для входа зарегистрированного пользователя в систему. После успешного входа пользователь получает доступ к защищенным разделам: корзине, заказам, профилю, избранным книгам, отзывам и интеллектуальному помощнику.

На рисунке 21 представлена страница каталога.

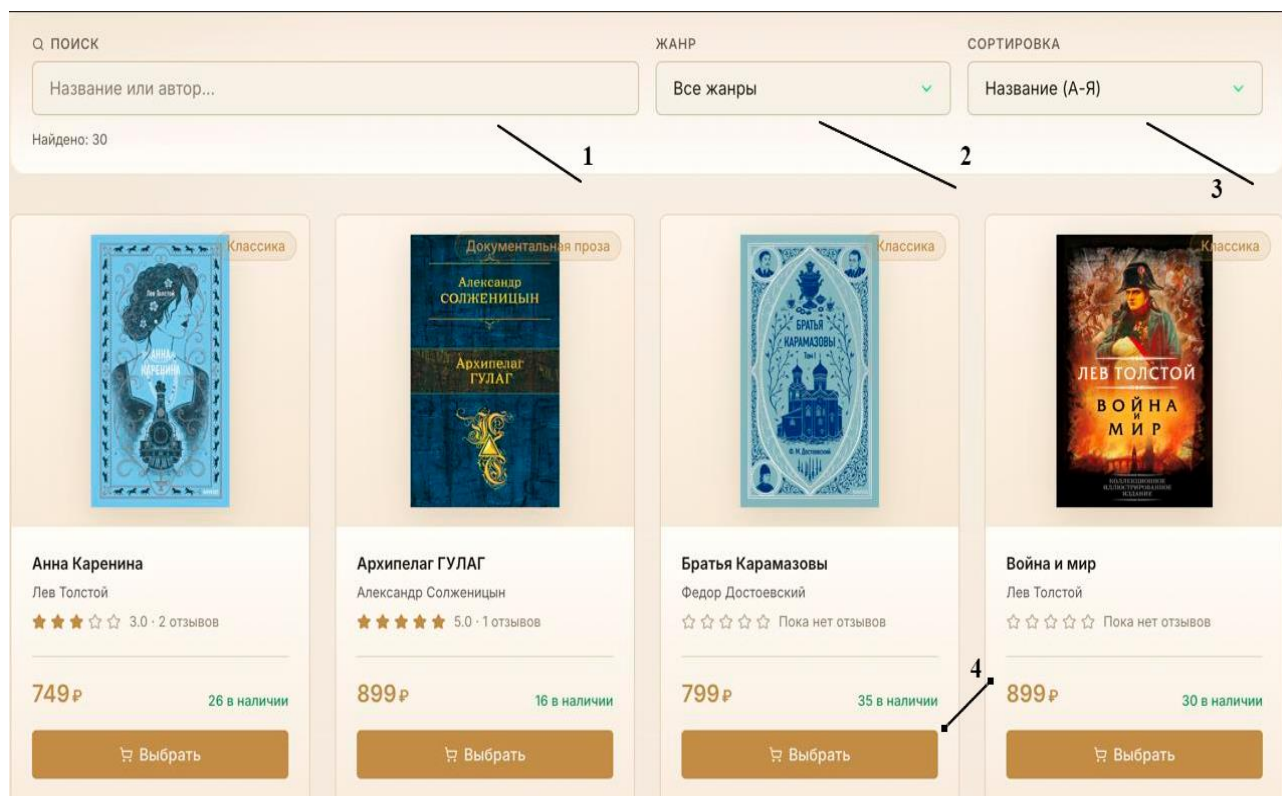


Рисунок 21 – Страница каталога книг

Основные компоненты страницы авторизации:

- 1 – поле поиска книг по названию или автору;
- 2 – фильтр по жанру;
- 3 – выпадающий список сортировки;
- 4 – кнопка добавления книги в корзину.

Страница каталога отображает список доступных книг. Пользователь может использовать поиск, чтобы найти произведение по названию или автору. Доступна фильтрация по жанру и сортировка для удобного поиска. Также карточки книг содержат основную информацию: обложку книги, название, автора, рейтинг, цену и кнопку добавления в корзину.

Ниже на рисунке 22 представлен внешний вид карточки книги.

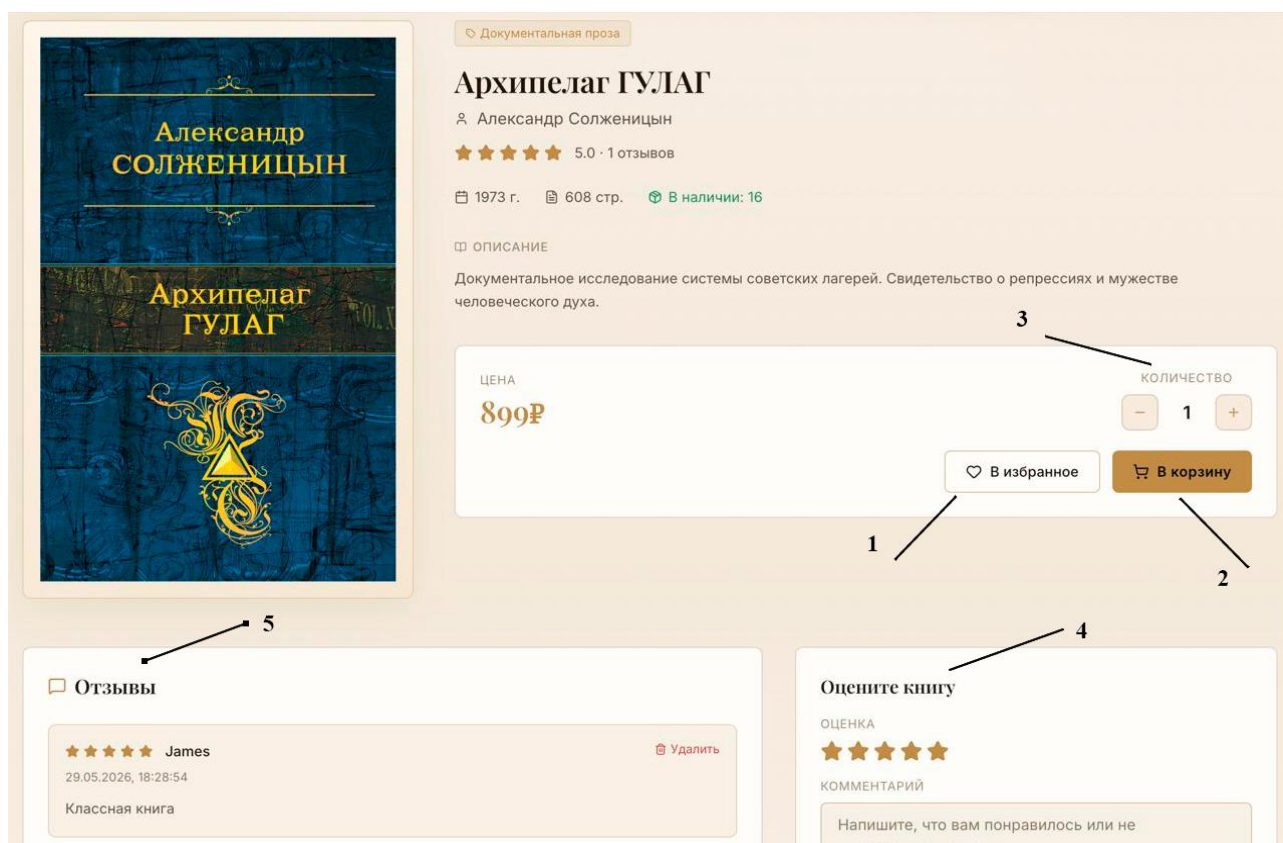


Рисунок 22 – Страница карточки книги

Основные элементы карточки книг:

- 1 – кнопка добавления книги в избранное;
- 2 – кнопка добавления книги в корзину;
- 3 – выбор количества экземпляров;
- 4 – блок оставления отзыва и оценки книги;
- 5 – раздел с отзывами пользователей.

Карточка позволяет пользователю подробно изучить основную составляющую книги. На странице отображается обложка, название, автор, рейтинг, год издания, количество страниц, цена и наличие товара на складе. Пользователю доступен следующий функционал:

возможность может добавить книгу в корзину или избранное, выбрать необходимое количество экземпляров, а также оставить отзыв и оценку после покупки.

Внешний вид страницы корзины представлен на рисунке 23.

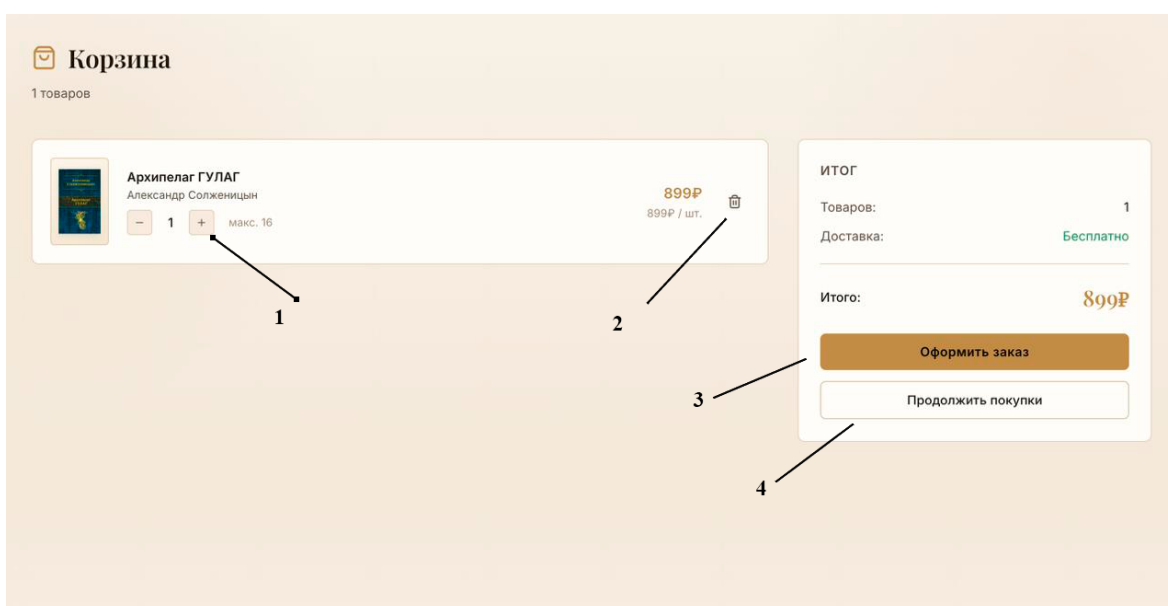


Рисунок 23 – Страница корзины пользователя

Основные элементы страницы:

- 1 – кнопки изменения количества товара;
- 2 – кнопка удаления позиции;
- 3 – кнопка оформления заказа;
- 4 – кнопка продолжения покупок.

Страница корзины содержит список выбранных товаров с возможностью изменения их количества и удаления. На странице также отображается итоговая стоимость заказа и кнопки оформления заказа или продолжения покупок.

На рисунке 24 представлена страница истории оформленных заказов.



Рисунок 24 – Страница истории заказов

Основные компоненты истории заказов:

- 1 – индикатор статуса заказа;
- 2 – идентификатор заказа;
- 3 – текущий статус заказа;
- 4 – сумма заказа;
- 5 – дата и время оформления заказа.

Страница истории заказов отображает все оформленные пользователем заказы. Для каждого заказа показывается его идентификатор, статус, сумма, дата и время оформления. Также отображается визуальный индикатор прохождения заказа по этапам: Новый – Оплачен – Отправлен – Доставлен.

Страница профиля пользователя представлена на рисунке 25.

ПРОФИЛЬ
Администратор

ИМЯ: Администратор

EMAIL: admin@gmail.com

С нами с 30 мая 2026 г.

ЗАКАЗОВ 1

ИЗБРАННОЕ 2

БЕЗОПАСНОСТЬ
Смена пароля

ТЕКУЩИЙ ПАРОЛЬ

НОВЫЙ ПАРОЛЬ

ПОВТОРИТЕ НОВЫЙ ПАРОЛЬ

Для безопасности используйте не менее 8 символов и не повторяйте текущий пароль.

Обновить пароль

История заказов

ЗАКАЗ	ДАТА	СТАТУС	СУММА
90fd3419-7bb...	08.06.2026, 02:30:03	Новый	899₽
Архипелаг ГУЛАГ Александр Солженицын			1 × 899₽ 899₽

Рисунок 25 – Личные данные пользователя и статус заказа

На данной странице есть следующие элементы:

- 1 – кнопка сохранения имени пользователя;
- 2 – блок с информацией о профиле;
- 3 – раздел истории заказов;

4 – блок смены пароля.

Страница профиля содержит основную информацию об учетной записи, возможность смены пароля, просмотр истории и статуса заказа.

В приложение В также изображены список избранных книг и рекомендации. Отдельная страница реализована для интеллектуального чат-помощника. Внешний вид страницы чат-помощника представлен на рисунке 26.

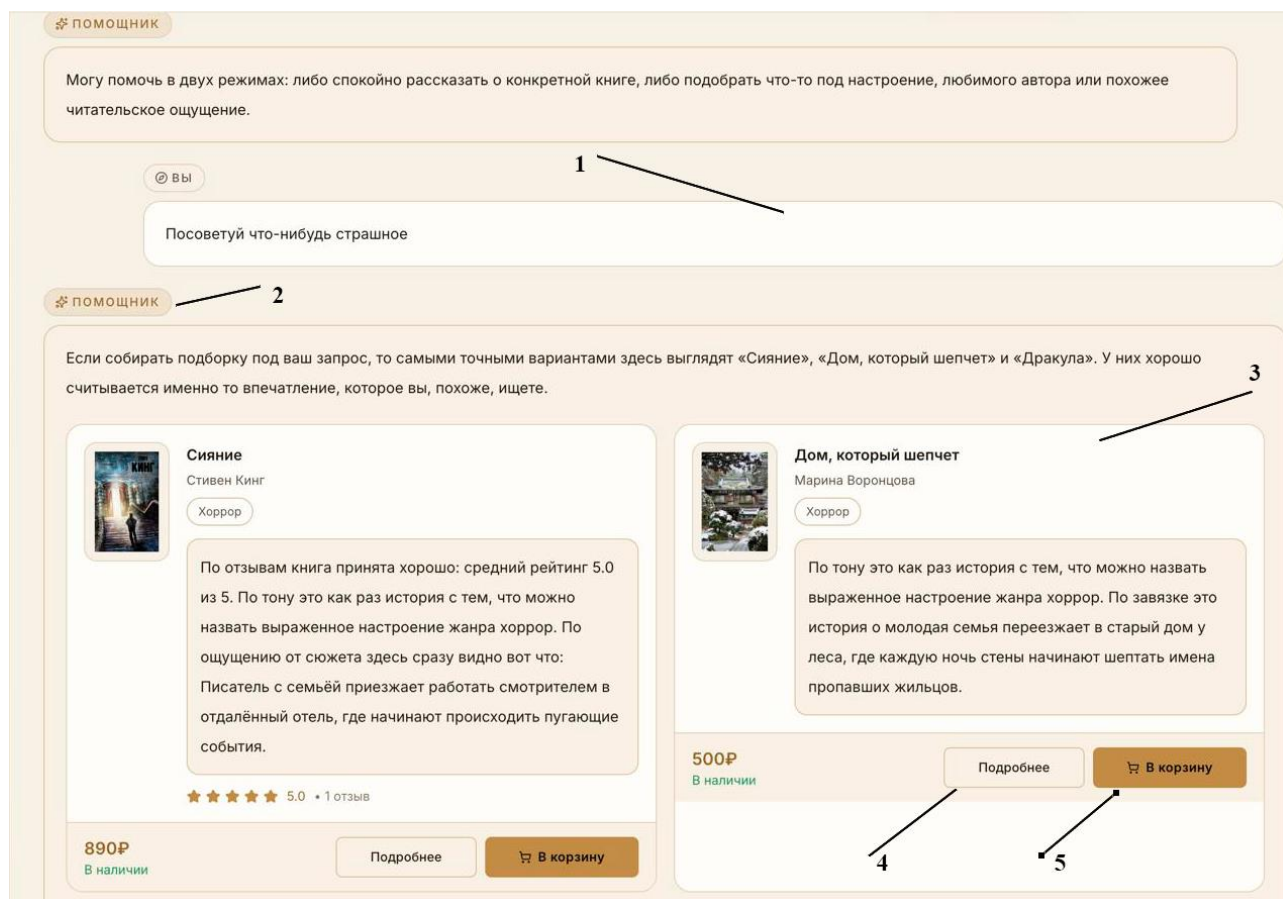


Рисунок 26 – Страница интеллектуального чат-помощника

Основные элементы страницы чат помощника:

- 1 – сообщение пользователя;
- 2 – ответ интеллектуального помощника;
- 3 – блок с рекомендациями книг;
- 4 – кнопка перехода на страницу книги;
- 5 – кнопка добавления книги в корзину.

Пользователь вводит запрос в свободной форме, а система возвращает ему список рекомендованных книг с пояснениями. Из рекомендаций можно сразу перейти к книге или добавить ее в корзину. Внешний вид административной панели представлен на рисунках 27.

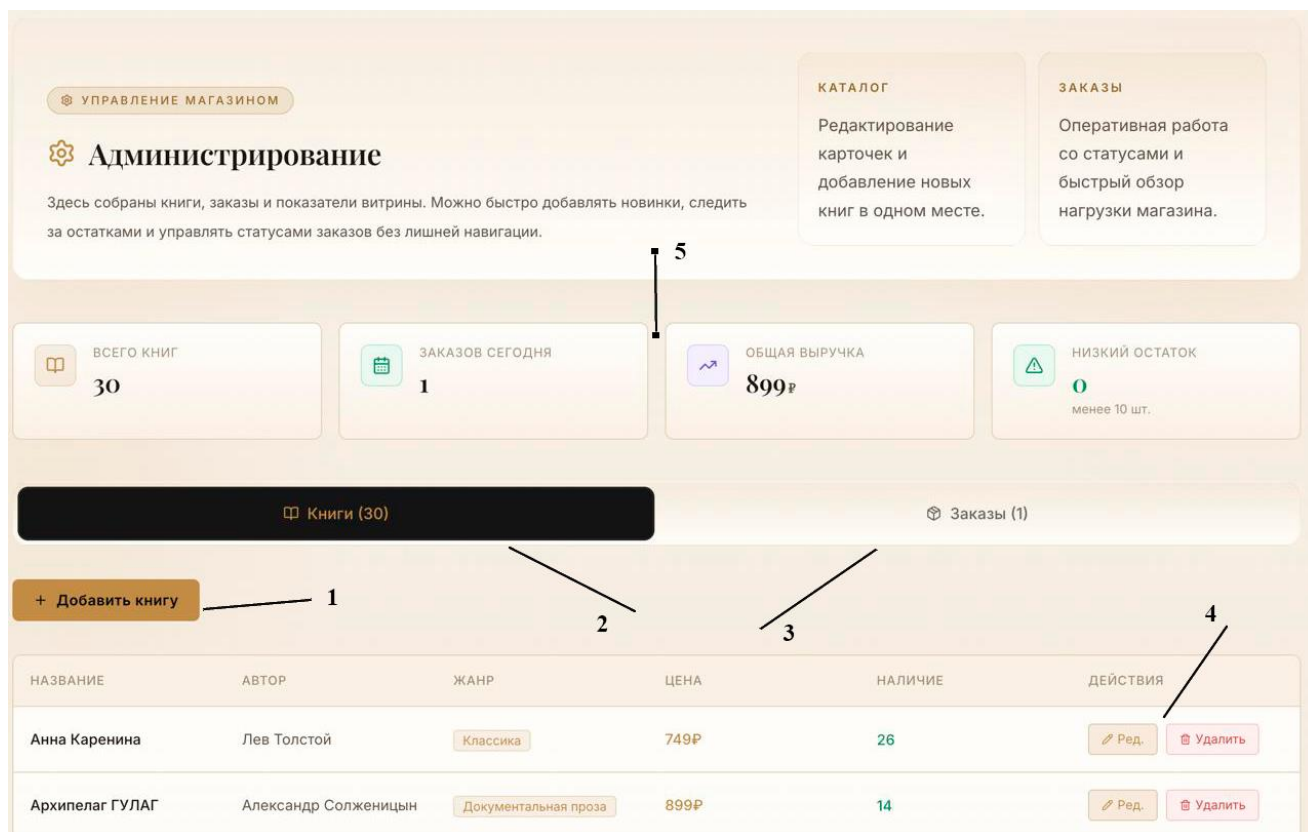


Рисунок 27 – Внешний вид административной панели

Основные элементы панели администратора:

- 1 – кнопка добавления новой книги;
- 2 – вкладка управления книгами;
- 3 – вкладка управления заказами;
- 4 – кнопки редактирования и удаления книг;
- 5 – блок с общей статистикой магазина.

Для администратора предусмотрена административная панель. В ней доступны общая информация, добавление и редактирование книг, просмотр всех заказов и изменение их статусов. Более детальные скриншоты отдельных разделов административной панели представлены в приложении В.

В результате пользовательский интерфейс охватывает все основные сценарии работы с системой: от поиска и выбора книги до оформления заказа и работы с рекомендациями. Благодаря продуманной навигации и разделению ролям пользователь может быстро выполнять нужные действия, а администратор эффективно управлять каталогом и заказами.

4.4 Развертывание информационной системы

Развертывание автоматизированной интеллектуальной системы продажи книг выполнено с помощью Docker и Docker Compose.

Docker представляет собой платформу для разработки, доставки и запуска приложений в контейнерах. Каждый контейнер включает приложение вместе со всеми его зависимостями, что обеспечивает изоляцию окружения и высокую воспроизводимость системы [16] docker.

Управление всеми контейнерами осуществляется через Docker Compose. Состав основных контейнеров и их назначение представлены в таблице 21.

Таблица 21 – Контейнеры информационной системы

Контейнер	Назначение	Порт
auth	Регистрация, авторизация и работа с учетными записями	8001
catalog	Каталог книг, отзывы, рейтинги и остатки	8002
order	Корзина, оформление заказов и история покупок	8003
profile	Профиль пользователя и избранные книги	8004

Продолжение таблицы 21

Контейнер	Назначение	Порт
ai	Интеллектуальный подбор книг	8005
frontend	Клиентское веб-приложение	80
postgres	Хранение данных сервисов	5432
pgadmin	Веб-интерфейс для работы с PostgreSQL	80
zookeeper	Вспомогательный сервис для Kafka	2181
kafka	Брокер сообщений для событий между сервисами	9092

Запуск системы выполняется из корневой директории проекта командой:
`docker compose up –build`

Использование Docker и Docker Compose позволяет добиться единообразия среды на разных компьютерах, существенно упрощает процесс развертывания и дает возможность быстро поднимать систему как для разработки, так и для демонстрации.

5 Тестирование

5.1 Функциональное тестирование

Функциональное тестирование представляет собой проверку программного продукта на соответствие его функциональным требованиям. Оно направлено на подтверждение того, что система выполняет заданные функции: регистрацию, авторизацию, работу с каталогом, корзиной, заказами, отзывами и интеллектуальным помощником [17].

В ходе тестирования проверялись пользовательские и административные сценарии. Для пользователя выполнялись проверка входа, просмотра каталога, добавления книги в корзину, оформления заказа и отправки запроса чат-помощнику. Для администратора проверялись управление книгами и изменение статуса заказа.

Перед началом тестирования система запускалась через Docker Compose. После запуска проверялась доступность клиентского приложения, серверных сервисов и базы данных. Результаты фиксировались по каждому сценарию: действие, ожидаемый результат и фактический результат.

Основное внимание уделялось следующим группам функций:

- регистрация и авторизация пользователя;
- просмотр каталога книг;
- поиск, фильтрация и сортировка книг;
- просмотр карточки книги;
- работа с корзиной;
- оформление заказа;
- просмотр истории заказов;
- работа с профилем и избранными книгами;
- создание, редактирование и удаление отзывов;
- получение рекомендаций через интеллектуального помощника;
- управление каталогом и заказами администратором.

Результаты функционального тестирования представлены в таблице 22.

Таблица 22 – Результаты функционального тестирования

№	Проверяемая функция	Действие	Ожидаемый результат	Результат
1	Регистрация пользователя	Ввести имя, email и пароль, нажать кнопку регистрации	Пользователь создан, выполняется переход к входу или в систему	Успешно
2	Авторизация пользователя	Ввести корректные email и пароль	Пользователь входит в систему, отображаются защищенные разделы	Успешно
3	Ошибка авторизации	Ввести неверные данные входа	Система отображает сообщение об ошибке	Успешно
4	Просмотр каталога	Открыть страницу каталога	Отображается список книг	Успешно
5	Поиск книги	Ввести поисковый запрос	Отображаются книги, соответствующие запросу	Успешно
6	Фильтрация книг	Выбрать жанр в фильтре	В каталоге отображаются книги выбранного жанра	Успешно
7	Сортировка книг	Выбрать вариант сортировки	Порядок книг изменяется согласно выбранному параметру	Успешно
8	Просмотр карточки книги	Открыть книгу из каталога	Отображаются подробные данные книги, рейтинг и отзывы	Успешно

Продолжение таблицы 22

№	Проверяемая функция	Действие	Ожидаемый результат	Результат
9	Добавление в корзину	Нажать кнопку добавления книги в корзину	Книга появляется в корзине пользователя	Успешно
10	Изменение количества	Изменить количество книги в корзине	Количество и итоговая сумма обновляются	Успешно
11	Удаление из корзины	Удалить книгу из корзины	Позиция удаляется из корзины	Успешно
12	Оформление заказа	Оформить заказ из корзины	Создается заказ, корзина очищается	Успешно
13	История заказов	Открыть страницу заказов	Отображается список заказов пользователя	Успешно
14	Добавление в избранное	Добавить книгу в избранное	Книга отображается в профиле пользователя	Успешно
15	Удаление из избранного	Удалить книгу из избранного	Книга удаляется из списка избранного	Успешно
16	Создание отзыва	Оставить оценку и комментарий к книге	Отзыв отображается на странице книги	Успешно
17	Редактирование отзыва	Изменить текст или оценку своего отзыва	Отзыв обновляется	Успешно
18	Удаление отзыва	Удалить собственный отзыв	Отзыв исчезает из списка	Успешно

Окончание таблицы 22

№	Проверяемая функция	Действие	Ожидаемый результат	Результат
19	Запрос к AI-помощнику	Отправить текстовый запрос о подборе книги	Отображается ответ и список рекомендованных книг	Успешно
20	Административное добавление книги	Добавить книгу через панель администратора	Книга появляется в каталоге	Успешно
21	Изменение статуса заказа	Изменить статус заказа в панели администратора	Статус заказа обновляется	Успешно
22	Ограничение доступа	Открыть административную панель обычным пользователем	Доступ запрещен или выполняется перенаправление	Успешно

В ходе функционального тестирования основные сценарии работы системы были выполнены успешно. Проверка показала, что пользователь может зарегистрироваться, войти в систему, выбрать книгу, оформить заказ, работать с профилем, отзывами и рекомендациями. Административные функции доступны только пользователю с соответствующей ролью, что подтверждает корректную работу разграничения доступа.

5.2 Модульное тестирование

Модульное тестирование – это тестирование отдельных частей программы в изоляции от остальных компонентов. Оно позволяет проверить корректность работы конкретных модулей и функций до их интеграции в общую систему [18].

Целью тестирования проводилось для проверки отдельных частей серверного приложения без запуска полного пользовательского сценария. Основное внимание уделялось функциям, которые отвечают за обработку данных, проверку входных параметров и выполнение бизнес-логики

Проект построен как набор отдельных сервисов: auth, catalog, order, profile и ai, а также содержит общий модуль shared. Такое разделение позволяет тестировать компоненты независимо друг от друга и быстрее находить ошибки в конкретной части системы.

Для модульного тестирования были выбраны наиболее важные участки системы:

- проверка регистрации и авторизации пользователя;
- генерация и проверка JWT-токена;
- хеширование и сравнение пароля;
- фильтрация и сортировка книг;
- расчет итоговой суммы заказа;
- проверка допустимых статусов заказа;
- проверка оценки книги в диапазоне от 1 до 5;
- обработка пустого или некорректного запроса к интеллектуальному помощнику.

Результаты модульного тестирования представлены в таблице 23.

Таблица 23 – Результаты модульного тестирования

№	Проверяемый модуль	Проверяемое действие	Ожидаемый результат	Результат
1	auth-service	Регистрация с корректными данными	Пользователь создается с ролью client	Успешно
2	auth-service	Регистрация с некорректным email	Возвращается ошибка валидации	Успешно

Продолжение таблицы 23

№	Проверяемый модуль	Проверяемое действие	Ожидаемый результат	Результат
3	auth-service	Вход с верным паролем	Формируется JWT-токен	Успешно
4	auth-service	Вход с неверным паролем	Возвращается ошибка авторизации	Успешно
5	shared/middleware	Проверка запроса без токена	Доступ к защищенному маршруту запрещен	Успешно
6	catalog-service	Фильтрация книг по жанру	Возвращаются книги выбранного жанра	Успешно
7	catalog-service	Сортировка книг по цене	Список книг возвращается в нужном порядке	Успешно
8	catalog-service	Создание отзыва с оценкой выше 5	Возвращается ошибка	Успешно
9	order-service	Расчет суммы заказа	Итоговая сумма соответствует цене и количеству книг	Успешно
10	order-service	Оформление заказа с пустой корзиной	Возвращается ошибка	Успешно
11	order-service	Передача недопустимого статуса заказа	Возвращается ошибка запроса	Успешно
12	profile-service	Добавление книги в избранное	Создается связь пользователя и книги	Успешно

Окончание таблицы 23

№	Проверяемый модуль	Проверяемое действие	Ожидаемый результат	Результат
13	profile-service	Добавление книги с некорректным UUID	Возвращается ошибка формата идентификатора	Успешно
14	ai-service	Пустой запрос к помощнику	Возвращается сообщение об ошибке или уточнение	Успешно
15	ai-service	Слишком общий запрос	Возвращается уточняющий вопрос	Успешно

На рисунке 28 представлены результаты модульного тестирования.

```

danil@MacBookDan auth % cd "/Users/danil/Desktop/goRepo/bookstore demo/ai"
go test ./... -v
--- PASS: TestParseStructuredResponseAcceptsFencedJSON (0.00s)
=== RUN TestDetectIntent/follow_up_from_history
=== PAUSE TestDetectIntent/follow_up_from_history
=== RUN TestDetectIntent/recommendation_request
--- PASS: TestCleanBookReferenceFromBookPrompt (0.00s)
--- PASS: TestCleanBookReference (0.00s)
=== PAUSE TestDetectIntent/recommendation_request
--- PASS: TestBuildConversationContextDoesNotLeakOldTopicIntoNewRecommendation (0.00s)
=== RUN TestDetectIntent/recommendation_request_with_napodobie_should_not_become_author_query
--- PASS: TestCleanBookReferenceWithNeedPhrase (0.00s)
=== PAUSE TestDetectIntent/recommendation_request_with_napodobie_should_not_become_author_query
=== RUN TestDetectIntent/existing_recommendation_with_romance_wording
--- PASS: TestLooksLikeAuthorRequestDoesNotTreatPopularityAsAuthorQuery (0.00s)
=== PAUSE TestDetectIntent/existing_recommendation_with_romance_wording
--- PASS: TestMapLLMResponseHidesBooksForConversation (0.00s)
--- PASS: TestShouldUseHistoryContextForRecommendation (0.00s)
=== RUN TestDetectIntent/recommendation_request_with_inline_book_should_beat_old_history
--- PASS: TestMapLLMResponseFallsBackWhenRecommendationIDsDoNotMatch (0.00s)
--- PASS: TestCleanBookReferenceWithoutQuotes (0.00s)
--- PASS: TestShouldReplaceAnswerForPopularityRejectsAuthorStyleText (0.00s)
--- PASS: TestMapLLMResponseMapsRecommendationsAndQuestion (0.00s)
--- PASS: TestBuildEffectiveMessageBookFromHist (0.00s)
=== PAUSE TestDetectIntent/recommendation_request_with_inline_book_should_beat_old_history
--- PASS: TestFindCandidateBooksReturnsOnlyRequestedAuthorBooks (0.00s)
--- PASS: TestBuildFallbackAnswerForSpecificBookDoesNotDuplicateEto (0.00s)
--- PASS: TestFindCandidateBooksExcludesAnchorBookForSimilarRequest (0.00s)
=== RUN TestDetectIntent/popularity_request
=== PAUSE TestDetectIntent/popularity_request
=== RUN TestDetectIntent/preference_answer_after_clarifying
=== PAUSE TestDetectIntent/preference_answer_after_clarifying
=== CONT TestDetectIntent/small_talk
=== CONT TestDetectIntent/recommendation_request_with_napodobie_should_not_become_author_query
=== CONT TestDetectIntent/follow_up_from_history
=== CONT TestDetectIntent/popularity_request
=== CONT TestDetectIntent/specific_book_question
=== CONT TestDetectIntent/too_general_request
=== CONT TestDetectIntent/preference_answer_after_clarifying
=== CONT TestDetectIntent/recommendation_request
=== CONT TestDetectIntent/existing_recommendation_with_romance_wording
=== CONT TestDetectIntent/recommendation_request_with_inline_book_should_beat_old_history
--- PASS: TestDetectIntent (0.00s)
--- PASS: TestDetectIntent/small_talk (0.00s)
--- PASS: TestDetectIntent/follow_up_from_history (0.00s)
--- PASS: TestDetectIntent/popularity_request (0.00s)
--- PASS: TestDetectIntent/recommendation_request_with_napodobie_should_not_become_author_query (0.00s)
--- PASS: TestDetectIntent/specific_book_question (0.00s)
--- PASS: TestDetectIntent/too_general_request (0.00s)
--- PASS: TestDetectIntent/preference_answer_after_clarifying (0.00s)
--- PASS: TestDetectIntent/recommendation_request (0.00s)
--- PASS: TestDetectIntent/existing_recommendation_with_romance_wording (0.00s)
--- PASS: TestDetectIntent/recommendation_request_with_inline_book_should_beat_old_history (0.00s)
PASS
ok      github.com/JustTheMercer/bookstore-microservices/ai/internal/service 0.372s

```

Рисунок 28 – Результаты модульного тестирования

В ходе модульного тестирования проверялись как корректные, так и ошибочные сценарии. За счет этого удалось убедиться, что отдельные функции сервисов обрабатывают входные данные ожидаемым образом и возвращают корректные ответы при ошибках. Такой подход дополняет функциональное тестирование, так как позволяет проверить внутреннюю логику системы отдельно от пользовательского интерфейса.

5.3 Метрики качества кода

5.3.1 Общие сведения о метриках качества кода

Метрики качества программного кода используются для количественной оценки отдельных характеристик программы [19]. Они предоставляют возможность анализа объема текущего кода, его структуру, сложность, а также выявить участки, которые могут потребовать дальнейшей доработки.

Метрики являются важным инструментом для разработчиков, но не дают полной оценки качества программного продукта, так как удобство использования, корректность бизнес-логики и надежность системы проверяются другими видами тестирования. Однако они помогают получить общее представление о размере проекта и распределении кода между его компонентами.

В рамках данной работы для оценки использовалась метрика SLOC. Она показывает количество строк исходного кода без учета пустых строк и комментариев. Данная метрика подходит для общей оценки объема разработанной системы, так как проект включает серверные сервисы, общий модуль и клиентское приложение.

Для подсчета строк кода использовалась утилита cloc, которая формирует отчет по языкам программирования и показывает количество файлов, пустых строк, комментариев и строк исходного кода.

5.3.2 Метрики SLOC

Метрика SLOC используется для оценки объема исходного кода программного продукта [20]. Она показывает количество строк кода без учета пустых строк и комментариев. В данной работе эта метрика применяется для анализа общего размера проекта и распределения кода между серверной и клиентской частями.

Подсчет строк исходного кода выполнялся с помощью утилиты cloc. В анализ были включены серверные сервисы auth, catalog, order, profile, ai, общий модуль shared, а также исходный код клиентского приложения в директории frontend/src. Команда запуска представлена в листинге 13.

Листинг 13 – Команда подсчета строк исходного кода

```
cloc auth catalog order profile ai shared frontend/src
```

На рисунке 29 представлен результат подсчета строк исходного кода.

```
danil@MacBookDan bookstore demo % cloc auth catalog order profile ai shared frontend/src
111 text files.
101 unique files.
11 files ignored.

github.com/AlDanial/cloc v 2.08 T=0.05 s (1860.0 files/s, 252541.1 lines/s)
-----
Language               files      blank      comment      code
-----
Go                      68         1267         99          7078
TypeScript              26          362         13          4066
CSS                     1           59          1           448
JSON                   1            0           0           222
Dockerfile              5           45          2           51
SUM:                   101        1733        115        11865
-----
```

Рисунок 29 – Результат подсчета строк исходного кода

Также результаты подсчета SLOC представлены в таблице 24.

Таблица 24 – Результаты подсчета строк исходного кода

Язык	Количество файлов	Пустые строки	Комментарии	Строки кода
Go	68	1267	99	7078
TypeScript	26	362	13	4066

Продолжение таблицы 24

Язык	Количество файлов	Пустые строки	Комментарии	Строки кода
CSS	1	59	1	448
JSON	1	0	0	222
Dockerfile	5	45	2	51
Итого	101	1733	115	11865

По результатам подсчета видно, что наибольший объем занимает код на языке Go – 7078 строк. Это связано с тем, что серверная часть состоит из нескольких сервисов: авторизации, каталога, заказов, профиля и интеллектуальных рекомендаций. Каждый сервис содержит обработчики, бизнес-логику, модели, репозитории и вспомогательные компоненты.

Клиентская часть написана на TypeScript и содержит 4066 строк кода. В нее входят страницы интерфейса, общие компоненты, API-клиент, контекст авторизации и типы данных. Также в проекте присутствуют CSS-файлы для оформления интерфейса, JSON-файлы конфигурации и Dockerfile-файлы для контейнеризации компонентов.

Полученные значения показывают, что проект имеет умеренный объем исходного кода и разделен между серверной и клиентской частями. Основная часть кода приходится на серверные сервисы, так как именно там реализована основная бизнес-логика системы.

Заключение

В ходе выполнения выпускной квалификационной работы была разработана автоматизированная интеллектуальная система продажи книг, объединяющая функционал современного интернет-магазина с интеллектуальным чат-помощником для подбора литературы по текстовому запросу пользователя.

В ходе работы были решены все поставленные задачи. Проанализирована предметная область книжной электронной торговли и существующие аналоги. Определены функциональные и нефункциональные требования к системе. Спроектирована трехуровневая клиент-серверная архитектура на основе микросервисов. Разработана база данных, реализованы сервисы на языке Go, клиентское веб-приложение на React с TypeScript, а также интеллектуальный модуль рекомендаций.

Результатом работы стала работоспособная система, включающая следующие возможности:

- просмотр и поиск книг с фильтрацией и сортировкой;
- работу с корзиной и оформление заказов;
- систему отзывов и рейтингов;
- личный профиль и избранное;
- интеллектуальный чат-помощник, позволяющий получать рекомендации книг по описанию на естественном языке с пояснением причин выбора;
- административную панель для управления каталогом и заказами.

Система реализована с использованием современных технологий: микросервисной архитектуры, Docker и Docker Compose, PostgreSQL, Kafka и JWT-авторизации. Интеллектуальный модуль рекомендаций реализован с использованием модели OpenAI GPT-4o, что обеспечило высокое качество подбора литературы и генерации пояснений.

Проведено модульное и функциональное тестирование, подтвердившее корректность работы основных сценариев.

Разработанное приложение может быть использовано как готовый программный продукт для малого и среднего бизнеса в сфере книжной торговли, а также как основа для дальнейшего развития и масштабирования.

В будущем возможно расширение функционала системы за счет добавления платежных систем, внедрения рекомендаций на основе истории покупок, добавления системы уведомлений и мобильной версии приложения, а также улучшения функционала и доработки модели чат-помощника.

Таким образом, поставленная задача достигнута в полном объеме. Разработанная автоматизированная интеллектуальная система продажи книг соответствует современным требованиям к веб-приложениям электронной коммерции и демонстрирует эффективное сочетание традиционных функций интернет-магазина с инновационными возможностями искусственного интеллекта.

Список используемых источников

1. ГОСТ Р 51303–2023. Торговля. Термины и определения. URL: <https://files.stroyinf.ru/Data/802/80250.pdf> (дата обращения: 14.04.2026).
2. Лабиринт. Книжный интернет-магазин [Электронный ресурс]. URL: <https://www.labirint.ru/> (дата обращения: 17.04.2026).
3. Читай-город. Федеральная сеть книжных магазинов [Электронный ресурс]. URL: <https://www.chitai-gorod.ru/> (дата обращения: 17.04.2026).
4. Book24. Книжный интернет-магазин [Электронный ресурс]. URL: <https://book24.ru/> (дата обращения: 17.04.2026).
5. Книги купить на Ozon [Электронный ресурс]. URL: <https://www.ozon.ru/category/knigi-16500/> (дата обращения: 17.04.2026).
6. Руководство по Docker Compose для начинающих [Электронный ресурс] Хабр. URL: <https://habr.com/ru/companies/ruvds/articles/450312/> (дата обращения: 19.04.2026).
7. Функционально-стоимостной анализ бизнес-процессов [Электронный ресурс] // Business Studio. URL: https://www.businessstudio.ru/articles/article/funktsionalno_stoimostnyy_analiz_biznes_protseessov/ (дата обращения: 9.05.2026).
8. Рабочий план проведения функционально-стоимостного анализа [Электронный ресурс] // Корпоративный менеджмент. URL: <https://www.cfin.ru/management/controlling/fsa/stages.shtml> (дата обращения: 10.05.2026).
9. Ричардс М., Форд Н. Фундаментальный подход к программной архитектуре: паттерны, свойства, проверенные методы. – СПб.: Питер, 2023. – 448 с.
10. Подробнее о JWT / Хабр [Электронный ресурс]. URL: <https://habr.com/ru/articles/842056/> (дата обращения 18.05.2026).

11. Соль | Энциклопедия «Касперского» [Электронный ресурс]. URL: <https://encyclopedia.kaspersky.ru/glossary/salt/?ysclid=mbnmrjb4xf890591596> (дата обращения 19.05.2026).
12. Донован, А. Язык программирования Go / А. Донован, Б. Керниган. – Санкт-Петербург : Питер, 2020. – 400 с.
13. Ньюман, С. Создание микросервисов. Проектирование, разработка и сопровождение / С. Ньюман. – 2-е изд. – Санкт-Петербург : Питер, 2022. – 592 с.
14. Object-Relational Mapping (ORM) в Golang [Электронный ресурс]. URL: <https://habr.com/ru/companies/otus/articles/780902/> (дата обращения 20.05.2026).
15. Круг, С. Не заставляйте меня думать. Веб-юзабилити и здравый смысл / С. Круг. – 3-е изд. – Москва : Эксмо, 2021. – 256 с.
16. Гадзурас Эммануил Г13 Docker Compose для разработчика. Упростите разработку и оркестрацию многоконтейнерных приложений. – М.: ДМК Пресс, 2023. – 220 с.
17. Куликов, С. С. K90 Тестирование программного обеспечения. Базовый курс / С. С. Куликов. – 3-е изд. – Минск: Четыре четверти, 2020. – 312 с.
18. Савин, Р. С. Тестирование dot com. Руководство по тестированию программного обеспечения / Р. С. Савин. – Санкт-Петербург: 2020. – 432 с.
19. Программный код и его метрики [Электронный ресурс] // Хабр. – 2011. – URL: <https://habr.com/ru/companies/intel/articles/106082/> (дата обращения: 03.06.2026).
20. Макконнелл, С. Совершенный код. Полное руководство по разработке программного обеспечения / С. Макконнелл. – 2-е изд. – Санкт-Петербург : Питер, 2021. – 896 с.

Приложение А
(справочное)
Глоссарий основных терминов

Таблица А – Глоссарий терминов

Термин	Определение
Go	Язык программирования, используемый для разработки высокопроизводительной серверной части. В проекте применяется для реализации всех микросервисов.
Gin	Легкий веб-фреймворк для языка Go. Используется для настройки HTTP-маршрутов и обработки запросов в серверных сервисах.
GORM	ORM-библиотека для Go. Применяется для работы с PostgreSQL, описания моделей данных и автоматических миграций таблиц.
JWT (JSON Web Token)	Открытый стандарт для создания компактных токенов, используемых для аутентификации и авторизации. В системе хранит идентификатор пользователя, имя, email и роль.
REST API	Архитектурный стиль взаимодействия между клиентом и сервером через HTTP-запросы с использованием стандартных методов.
Микросервис	Независимый программный компонент, реализующий отдельную бизнес-функцию. Система разделена на пять микросервисов: auth, catalog, order, profile и ai.
Docker Compose	Инструмент для определения и запуска многоконтейнерных приложений. Используется для совместного запуска всех компонентов системы одной командой.
PostgreSQL	Реляционная система управления базами данных. В проекте применяется для хранения данных пользователей, книг, заказов, корзины, отзывов и избранного.
Клиентская часть	Веб-интерфейс приложения, разработанный на React с TypeScript. Обеспечивает взаимодействие пользователя с системой.

Продолжение таблицы А

Термин	Определение
Серверная часть	Совокупность микросервисов на языке Go, реализующих бизнес-логику, обработку запросов и взаимодействие с базой данных.
Роль пользователя	Атрибут учетной записи, определяющий права доступа. В системе используются роли client (пользователь) и admin (администратор).
Каталог книг	Подсистема хранения и представления информации о книгах с поддержкой поиска, фильтрации, сортировки и просмотра карточек товаров.
Корзина	Временное хранилище выбранных пользователем книг перед оформлением заказа.
Заказ	Оформленная покупка, содержащая список книг, итоговую сумму, статус и дату создания.
Отзыв	Текстовая оценка книги пользователем (от 1 до 5) с комментарием. Один пользователь может оставить только один отзыв на книгу.
Рейтинг книги	Среднее значение оценок, рассчитанное на основе всех пользовательских отзывов.
Избранное	Персональный список книг пользователя для быстрого доступа.
Интеллектуальный помощник	Модуль системы, осуществляющий подбор книг по текстовому запросу пользователя на естественном языке с формированием пояснений.
RAG (Retrieval-Augmented Generation)	Подход к генерации ответов, при котором большая языковая модель использует предварительно извлеченные данные из каталога книг.
React	Библиотека JavaScript для разработки пользовательских интерфейсов. Используется для разработки клиентской части как Single Page Application.

Окончание таблицы А

Термин	Определение
TypeScript	Надмножество JavaScript с поддержкой статической типизации. Применяется в клиентской части для повышения надежности кода.
OpenAI GPT-4o	Многофункциональная большая языковая модель от компании OpenAI, применяемая в ai-service для генерации рекомендаций книг по текстовому описанию пользователя.
LLM (Large Language Model)	Большая языковая модель — класс искусственных нейронных сетей, обученных на огромных объёмах текстовых данных. Используется для обработки естественного языка в интеллектуальном чат-помощнике.
Функциональное тестирование	Проверка системы на соответствие функциональным требованиям через пользовательские сценарии.
Модульное тестирование	Проверка отдельных модулей и функций программы в изолированной среде.

Приложение Б
(обязательное)
Код программы

```

func GetBookRating(c *gin.Context) {
bookID, ok := parseBookID(c)
if !ok {
return
}

summary, err := reviewRepo.GetRatingSummary(bookID)
if err != nil {
c.JSON(http.StatusInternalServerError, gin.H{"error": "Ошибка
получения рейтинга"})
return
}

c.JSON(http.StatusOK, newRatingSummaryResponse(summary))
}

func CreateReview(c *gin.Context) {
bookID, ok := parseBookID(c)
if !ok {
return
}

if _, err := bookRepo.GetByID(bookID); err != nil {
c.JSON(http.StatusNotFound, gin.H{"error": "Книга не найдена"})
return
}

userID, ok := parseUserID(c)
if !ok {
return
}

var payload reviewPayload
if err := c.ShouldBindJSON(&payload); err != nil {
c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
return
}

existing, err := reviewRepo.GetByUserAndBook(userID, bookID)
if err != nil {
c.JSON(http.StatusInternalServerError, gin.H{"error": "Ошибка
проверки существующего отзыва"})
return
}
if existing != nil {
c.JSON(http.StatusConflict, gin.H{"error": "Вы уже оставили отзыв
на эту книгу"})
return
}

canReview, err := canUserReviewBook(userID, bookID)

```

```

if err != nil {
    c.JSON(http.StatusBadGateway, gin.H{"error": "Не удалось
    проверить, покупал ли пользователь эту книгу"})
    return
}
if !canReview {
    c.JSON(http.StatusForbidden, gin.H{"error": "Оставлять отзывы
    можно только после доставки книги"})
    return
}

review := models.Review{
    ID: uuid.New(),
    BookID: bookID,
    UserID: userID,
    Username: getContextString(c, "name"),
    UserEmail: getContextString(c, "email"),
    Rating: payload.Rating,
    Comment: strings.TrimSpace(payload.Comment),
}

if err := reviewRepo.Create(&review); err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error": "Ошибка
    создания отзыва"})
    return
}

c.JSON(http.StatusCreated, newReviewResponse(review))
}

func CreateReview(c *gin.Context) {
    bookID, ok := parseBookID(c)
    if !ok {
        return
    }

    if _, err := bookRepo.GetByID(bookID); err != nil {
        c.JSON(http.StatusNotFound, gin.H{"error": "Книга не
        найдена"})
        return
    }

    userID, ok := parseUserID(c)
    if !ok {
        return
    }

    var payload reviewPayload
    if err := c.ShouldBindJSON(&payload); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": err.Error()})
        return
    }
}

```

```

// Проверка существующего отзыва
existing, err := reviewRepo.GetByUserAndBook(userID, bookID)
if err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error":
"Ошибка проверки существующего отзыва"})
    return
}
if existing != nil {
    c.JSON(http.StatusConflict, gin.H{"error": "Вы уже
оставили отзыв на эту книгу"})
    return
}

// Проверка факта покупки
canReview, err := canUserReviewBook(userID, bookID)
if err != nil {
    c.JSON(http.StatusBadGateway, gin.H{"error": "Не удалось
проверить покупку"})
    return
}
if !canReview {
    c.JSON(http.StatusForbidden, gin.H{"error": "Оставлять
отзывы можно только после доставки книги"})
    return
}

review := models.Review{
    ID:      uuid.New(),
    BookID:  bookID,
    UserID:  userID,
    Username: getContextString(c, "name"),
    UserEmail: getContextString(c, "email"),
    Rating:  payload.Rating,
    Comment: strings.TrimSpace(payload.Comment),
}

if err := reviewRepo.Create(&review); err != nil {
    c.JSON(http.StatusInternalServerError, gin.H{"error":
"Ошибка создания отзыва"})
    return
}

c.JSON(http.StatusCreated, newReviewResponse(review))
}

func CanReviewBook(c *gin.Context) {
    userID, err := uuid.Parse(c.Query("user_id"))
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "Неверный
user_id"})
        return
    }

```

```

    }

    bookID, err := uuid.Parse(c.Query("book_id"))
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "Неверный
book_id"})
        return
    }

    purchased, err := orderRepo.HasPurchasedBook(userID, bookID)
    if err != nil {
        c.JSON(http.StatusInternalServerError, gin.H{"error":
"Ошибка проверки покупки книги"})
        return
    }

    c.JSON(http.StatusOK, canReviewResponse{CanReview: purchased})
}

func (h *ProfileHandler) AddFavorite(c *gin.Context) {
    userID, ok := utils.GetUserIDFromContext(c)
    if !ok {
        return
    }

    var req addFavoriteRequest
    if err := c.ShouldBindJSON(&req); err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"error": "Неверный
формат запроса"})
        return
    }

    favorite, err :=
h.profileService.AddFavorite(c.Request.Context(), userID,
req.BookID)
    if err != nil {
        switch {
            case errors.Is(err, service.ErrInvalidBookID):
                c.JSON(http.StatusBadRequest, gin.H{"error": "Неверный
book_id"})
            case errors.Is(err, service.ErrBookNotFound):
                c.JSON(http.StatusNotFound, gin.H{"error": "Книга не
найдена"})
            default:
                c.JSON(http.StatusInternalServerError, gin.H{"error":
"Не удалось добавить книгу в избранное"})
        }
        return
    }

    c.JSON(http.StatusCreated, gin.H{
        "message": "Книга добавлена в избранное",
    })
}

```

```

        "book_id":      favorite.BookID,
        "favorited_at": favorite.CreatedAt,
    })
}

func normalizeHistory(history []ConversationEntry)
[]ConversationEntry {
    normalized := make([]ConversationEntry, 0, min(len(history), 10))
    for _, entry := range history {
        role := strings.TrimSpace(strings.ToLower(entry.Role))
        content := strings.TrimSpace(entry.Content)
        if content == "" {
            continue
        }
        if role != "user" && role != "assistant" {
            continue
        }
        normalized = append(normalized, ConversationEntry{Role: role,
Content: truncateText(content, 500)})
    }

    if len(normalized) > 10 {
        return normalized[len(normalized)-10:]
    }

    return normalized
}

func buildEffectiveMessage(message string, history
[]ConversationEntry, intent queryIntent) string {
    switch intent {
    case intentFollowUp:
        book := findMentionedBook(history)
        if book != "" {
            return message + " " + book
        }

        context := latestMeaningfulUserMessage(history)
        if context != "" {
            return message + " " + context
        }
    case intentSpecificBook:
        lower := strings.ToLower(strings.TrimSpace(message))
        book := findMentionedBook(history)
        if book != "" && (strings.Contains(lower,
"подробнее") strings.Contains(lower, "эта
книга") strings.Contains(lower, "она")) {
            return message + " " + book
        }
    case intentRecommendation:
        if looksLikePreferenceAnswer(message, history) {
            context := latestMeaningfulUserMessage(history)

```

```

    if context != "" {
        return message + " " + context
    }
    book := findMentionedBook(history)
    if book != "" {
        return message + " " + book
    }
}

return message
}

func findMentionedBook(history []ConversationEntry) string {
    for i := len(history) - 1; i >= 0; i-- {
        if history[i].Role != "user" {
            continue
        }
        content := strings.TrimSpace(history[i].Content)
        if content == "" {
            continue
        }

        lower := strings.ToLower(content)
        if looksLikeBookQuestion(content) ||
            strings.Contains(lower, "про ") ||
            strings.Contains(lower, "наподобие ") ||
            strings.Contains(lower, "похожее на ") ||
            strings.Contains(lower, "похоже на ") {
            return cleanBookReference(content)
        }
    }

    return ""
}

func (s *AIService) findCandidateBooks(message string, intent
queryIntent, history []ConversationEntry) ([]catalog.Book, error)
{
    if intent == intentSpecificBook {
        book, err := s.findSpecificBook(message, history)
        if err != nil {
            return nil, err
        }
        if book != nil {
            return []catalog.Book{*book}, nil
        }
    }

    seen := make(map[string]catalog.Book)
    shouldBroadenPool := intent == intentRecommendation || intent ==
intentFollowUp
    var purchasePopularity map[string]int

```

```

var requestedAuthor string

addBooks := func(books []catalog.Book) {
    for _, book := range books {
        if book.ID == "" {
            continue
        }
        if _, exists := seen[book.ID]; exists {
            continue
        }
        seen[book.ID] = book
    }
}

primary, err := s.catalogClient.SearchBooks(message,
detectGenre(message), s.maxCandidates)
if err != nil {
    return nil, err
}
addBooks(primary)

if shouldBroadenPool {
    broaderLimit := min(max(s.maxCandidates*8, 24), 100)
    broaderBooks, err := s.catalogClient.SearchBooks("",
detectGenre(message), broaderLimit)
    if err == nil {
        addBooks(broaderBooks)
        if looksLikeAuthorRequest(message) {
            requestedAuthor = findRequestedAuthor(message, broaderBooks)
            if requestedAuthor != "" {
                addBooks(filterBooksByAuthor(broaderBooks, requestedAuthor))
            }
        }
    }
}

result := make([]catalog.Book, 0, len(seen))
for _, book := range seen {
    result = append(result, book)
}

if requestedAuthor != "" {
    result = filterBooksByAuthor(result, requestedAuthor)
}

result = filterBooksForQuery(message, result)
purchasePopularity = s.loadPurchasePopularity(max(len(result)*3,
20))

sort.Slice(result, func(i, j int) bool {
    leftScore := candidateScore(message, result[i],
purchasePopularity[result[i].ID])

```

```
    rightScore := candidateScore(message, result[j],  
purchasePopularity[result[j].ID])  
    return leftScore > rightScore  
  })  
  
  return result, nil  
}
```

Полный код приложения приведен на GitHub:
<https://github.com/JustTheMercer/bookstore-microservices>

Приложение В
(рекомендуемое)
Скриншоты интерфейса

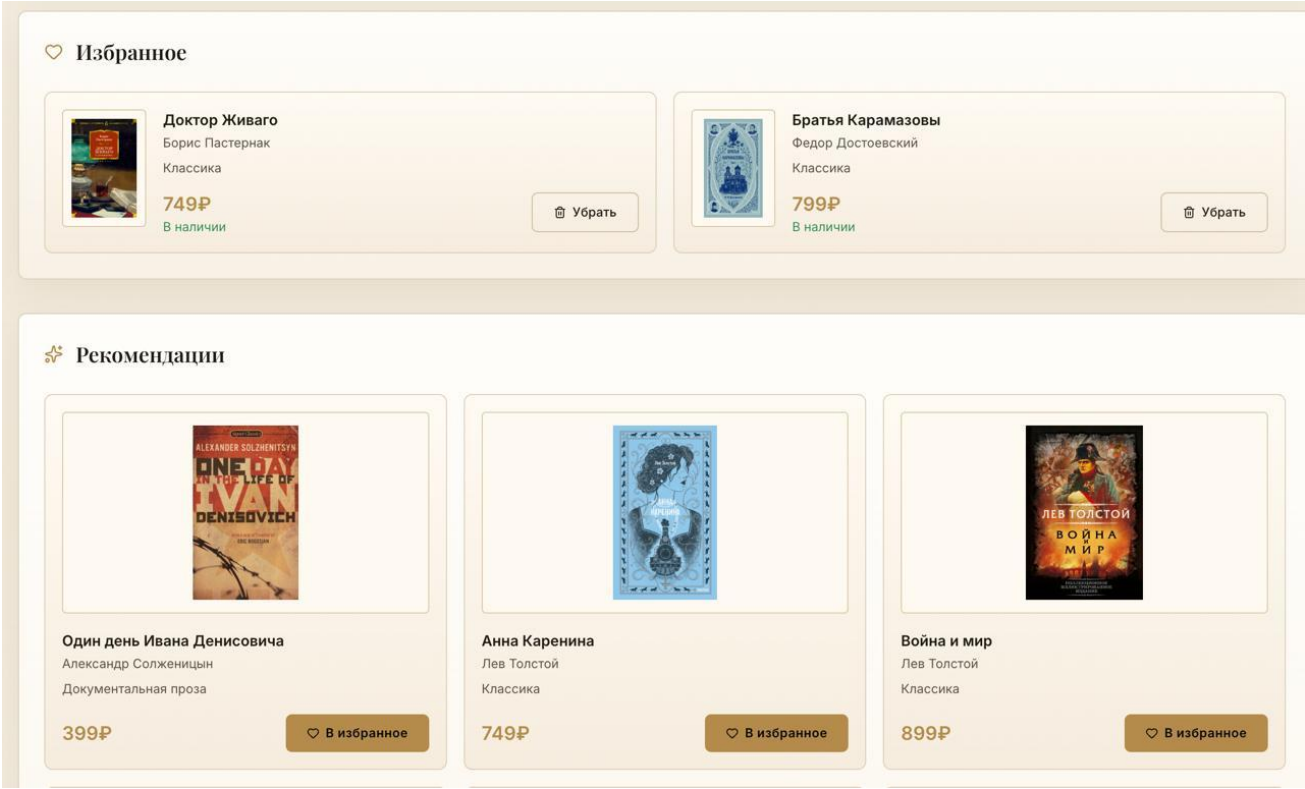


Рисунок В.1 – Страница избранных книг и рекомендаций

Новая книга

НАЗВАНИЕ

Название книги

АВТОР

Имя автора

ЖАНР

Жанр

КРАТКОЕ ОПИСАНИЕ

Краткая аннотация книги

ПОЛНОЕ ОПИСАНИЕ

Подробное описание книги

ГОД ИЗДАНИЯ

2024

СТРАНИЦ

0

ИЗДАТЕЛЬСТВО

Название издательства

ССЫЛКА НА ОБЛОЖКУ

<https://example.com/cover.jpg>

Рисунок В.2 – Добавление новой книги в административной панели

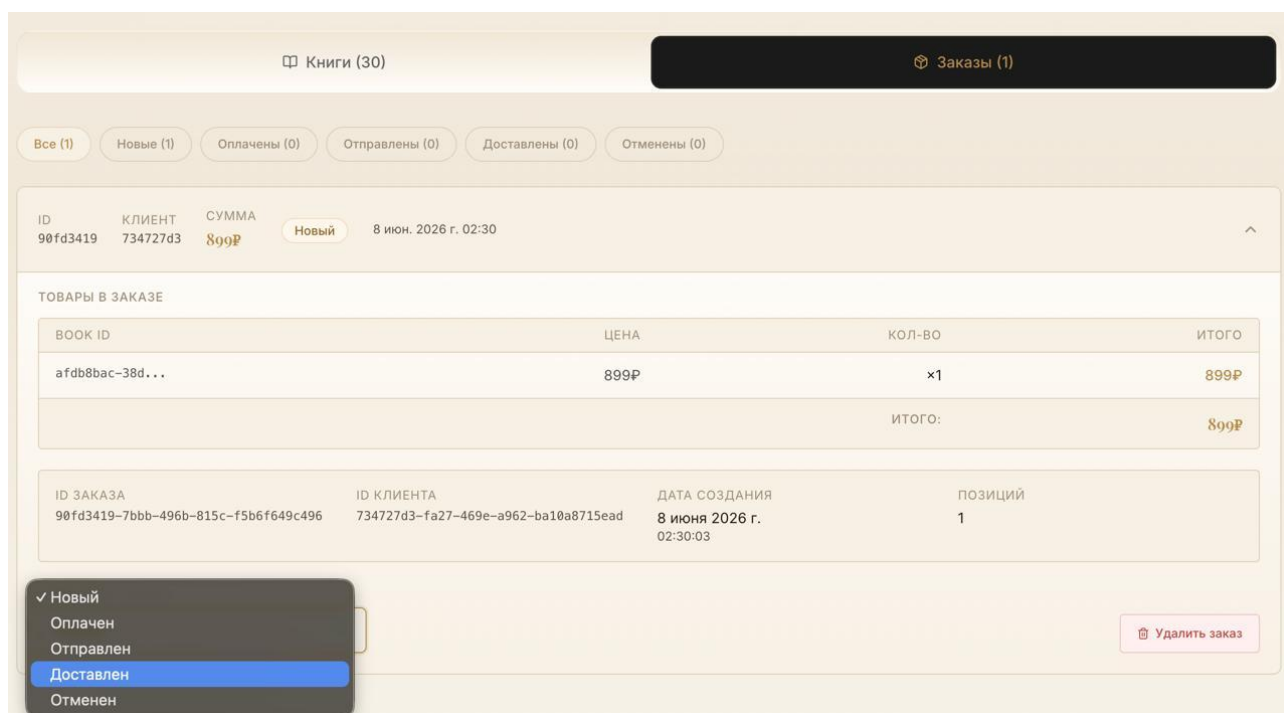


Рисунок В.3 – Управление заказами в административной панели