

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

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

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

Направленность Программная инженерия

БАКАЛАВРСКАЯ РАБОТА

на тему

**Программное обеспечение автоматизированной
системы управления книжным магазином**

Студент

(подпись, дата)

Бобков Максим Игоревич

(ФИО полностью)

Руководитель

(подпись, дата)

Князев В.Н.

(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

Такташкин Д.В.

(фамилия, инициалы)

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

Заведующий кафедрой

(подпись)

Козлов А.Ю.

(фамилия, инициалы)

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

Секретарь ГЭК

(подпись)

Попова Н.А.

(фамилия, инициалы)

Пенза, 2025

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
«20» декабря 2024 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Бобков Максим Игоревич гр. 21ВП2 факультета ВТ

<Фамилия, имя, отчество полностью>

2. Тема работы Программное обеспечение автоматизированной системы управления книжным магазином

Тема утверждена приказом ПГУ № _____ от “___” _____ 20__

3. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Князев В.Н.

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

1) Разработать программное обеспечение автоматизированной системы управления книжным магазином в форме мобильного программного приложения.

2) Среда выполнения приложения: операционная система Android 9 и выше

Среда выполнения сервера: операционная система Windows 10 и выше

Языковые средства разработки: C#, UML, SQL, Linq

Инструментальные средства разработки: Microsoft SQL Server 2019, Microsoft Visual Studio 2022, Visual Paradigm, Microsoft Project 2007

Технологии разработки: RUP, ООП, MVC, Entity Framework, MVVM

3) Основные функции мобильного приложения:

а) просмотр каталога книг;

б) покупка книг;

в) добавление, удаление и редактирования информации о книгах в каталоге;

г) отправка отзывов и вопросов;

д) фильтрация каталога;

е) просмотр статистики продаж.

5. Перечень подлежащих разработке вопросов

1) Анализ предметной области и постановка задачи

2) Проектирование

3) Реализация

4) Контроль качества программного обеспечения

5) Оформление пояснительной записки

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	08.04.2025 – 14.04.2025	
2	Проектирование программных средств	20%	15.04.2025 – 25.04.2025	
3	Реализация программных средств	30%	26.04.2025 – 12.05.2025	
4	Тестирование программных средств	20%	13.05.2025 – 22.05.2025	
5	Оформление пояснительной записки	20%	23.05.2025 – 03.06.2025	

Дата выдачи задания «20» декабря 2024г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

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

Реферат

Пояснительная записка содержит 191 листов, 64 рисунков, 25 таблиц, 31 использованных источника, 5 приложений.

МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, КНИЖНЫЙ МАГАЗИН, ANDROID, MS SQL SERVER, C#, MAUI, .NET, RUP, UML, MVC, MVVM

Объектом разработки является мобильное приложение для осуществления автоматизированного управления книжным магазином.

Цель работы – разработка мобильного программного приложения для поддержки процессов функционирования книжного магазина, позволяющего управлять каталогом книг, осуществлять покупку книг, добавлять книги в корзину и список отложенного, задавать интересующие вопросы, оставлять отзывы и просматривать статистику продаж.

Разработка серверной части происходила на языке программирования C#. В качестве среды разработки использовалась Microsoft Visual Studio 2022. В качестве СУБД была использована СУБД MS SQL Server.

Разработка приложения производилась на языке программирования C#. В качестве среды разработки использовалась среда программирования Microsoft Visual Studio 2022, операционная система Windows 10. Приложение предназначено для использования под управлением ОС Android версии 9.0 и выше, а его серверная часть работает под управлением операционной системы Windows 10 и выше.

					ПГУ 09.03.04 – 08БР212.05 ПЗ			
Изм.	Лист	№ докум.	Подп.	Дата	Программное обеспечение автоматизированной системы управления книжным магазином.	Лит.	Лист	Листов
Разраб.		Бобков М.И.					4	191
Пров.		Князев В.Н.			Пояснительная записка	Группа 21ВП2		
Н. контр.		Такташкин Д.В.						
Утверд.		Козлов А.Ю.						

Содержание

Введение.....	7
1 Анализ предметной области и требований	9
1.1 Анализ предметной области и постановка задачи	9
1.2 Анализ программных продуктов	10
1.2.1 Интернет-магазин «books.ru».....	10
1.2.2 Интернет-магазин «Лабиринт».....	11
1.2.3 Мобильное приложение «Читай-город».....	12
1.2.4 Сравнение аналогов	13
1.3 Анализ целевой аудитории.....	13
1.4 Анализ функциональных требований	16
1.5 Анализ нефункциональных требований	22
1.6 Планирование разработки и оценка бюджета	22
2 Проектирование мобильного приложения.....	42
2.1 Выбор технологий и инструментальных средств	42
2.2 Проектирование архитектуры программного обеспечения.....	43
2.3 Проектирование базы данных.....	45
2.3.1 Концептуальное проектирование базы данных	45
2.3.2 Логическое проектирование базы данных.....	47
2.4 Проектирование серверной части приложения.....	48
2.5 Проектирование интерфейса пользователя	49
3 Реализация программного обеспечения	66
3.1 Реализация базы данных.....	66
3.2 Разработка серверной части приложения	69
3.3 Реализация клиентской части приложения.....	72
4 Тестирование программного продукта.....	75
4.1 Функциональное тестирование черным ящиком	75
4.2 Модульное тестирование	76
4.3 Метрики кода	77
4.3.1 Метрики размера программы.....	78

4.3.2 Метрики Холстеда.....	79
Заключение	82
Список использованных источников	84
Приложение А (справочное). Глоссарий	87
Приложение Б (обязательное). Исходный код серверной части.....	89
Приложение В (обязательное). Исходный код мобильного приложения	128
Приложение Г (рекомендуемое). Результаты тестирования серверной части приложения	183
Приложение Д (рекомендуемое). Исходный код функций тестирования	187

Введение

В настоящее время информационные ресурсы общества являются главным фактором его развития. Они задают направления во всех сферах жизни человека, а также оказывают влияние на массовое сознание людей. Начало XXI века, считается развитием цифровых технологий. Ученные считают, что в будущем развитие технологий окажет влияние на все сферы жизни человека, как на экономическую, так и на социальную, а также и на весь мир в целом. На данный момент, цифровая среда проникла как в профессиональную сторону жизни людей, так и в повседневную [1].

В настоящее время, в нашей стране активно идет внедрения информационных технологий во все сферы деятельности человека. Данный процесс поддерживается на государственном уровне, программой «Цифровая экономика Российской Федерации». Данная программа была утверждена 9 мая 2017 года правительством Российской Федерации. Данный документ определяет стратегию развития «...цифровой экономики Российской Федерации, в которой данные в цифровой форме являются ключевым фактором производства во всех сферах социально-экономической деятельности, что повышает конкурентоспособность страны, качество жизни граждан, обеспечивает экономический рост». Также, данная программа направлена на «... повышение благосостояния и качества жизни граждан нашей страны путем повышения доступности и качества товаров и услуг, произведенных в цифровой экономике с использованием современных цифровых технологий, ...».

Также 9 мая 2017 года был подписан Указ Президента Российской Федерации «О стратегии развития информационного общества в Российской Федерации на 2017 – 2030 годы». В нем были поставлены основные цели и задачи политики Российской Федерации в сфере информационных технологий, а также объявлен курс на формирование национальной цифровой экономики [2].

Важной особенностью развития информационных технологий, является повсеместное использование мобильного устройства, который в свою очередь,

подключен к сети Интернет. Также стоит учесть, что мобильный интернет, с течением времени оттесняет доступ к сети через стационарные компьютеры, так как является более удобным [3]. Так, за 2023 год было зарегистрировано 227 миллионов сотовых мобильных подключений, что составляет 156,9% от общей численности населения. Это свидетельствует о высоком уровне мобильной связи. Также, в нашей стране на каждые 100 человек приходится более 164 мобильных телефона. В настоящее время, данные тенденции лишь усиливаются.

Сегодня, рынок бумажных книг, демонстрирует стабильный спрос, у клиентов любого возраста. Кто-то покупает книги в целях образования, для кого-то это способ скрасить свой досуг. Таким образом, книга занимает далеко не последнюю роль в жизни большинства людей.

На данный момент, практически у каждого человека имеется мобильное устройство, которое имеет выход в сеть Интернет. Поэтому его использование является привычным для большинства граждан. Это касается и людей, работающих в сфере услуг. Например, администратору книжного магазина намного удобнее выполнять свои обязанности при помощи телефона, не используя компьютер или ноутбук. Также, менеджеру по работе с клиентами, проще отвечать на вопросы покупателей с телефоном в руках, чем использовать громоздкий компьютер.

Клиентам, в свою очередь, удобно осуществлять выбор книг в каталоге, не ходя в магазин. А при помощи удобной фильтрации каталога книг, это будет занимать меньше времени, чем то, которое затрачивается на поиск интересующей книги в магазине. Покупка книг, осуществляется в несколько кликов и не требует большого времени.

Исходя из всего вышеперечисленного следует, что проект разработки программного обеспечения автоматизированного управления книжным магазином для мобильного приложения является перспективным.

1 Анализ предметной области и требований

1.1 Анализ предметной области и постановка задачи

Разрабатываемое мобильное приложение должно обеспечить пользователю удобный и интуитивно понятный интерфейс, для покупки книг в книжном магазине. С помощью, разрабатываемого приложения, пользователи могут просматривать каталог доступных для покупки книг, добавлять их в корзину для осуществления покупки. Также, пользователи имеют возможность отсортировать каталог книг, что позволит быстрее найти необходимую книгу в каталоге. Пользователи могут добавить понравившиеся книги в список отложенных товаров, чтобы после вернуться к этим книгам, не искав их повторно в каталоге. Также пользователи могут задать интересующий их вопрос, и впоследствии получить на него ответ и оставлять отзывы на книги.

Таким образом, в приложении должны быть представлены следующие основные функции:

- регистрация/вход пользователя;
- просмотр каталога книг;
- сортировка каталога книг;
- поиск книги в каталоге;
- добавление книги/книг в корзину;
- добавление книги/книг в список отложенного;
- оформить и оплатить покупку;
- просмотр истории, совершенных покупок;
- задать вопрос;
- оставить отзыв.

Помимо обычных пользователей в системе будет присутствовать администратор. Он имеет следующие функции: добавить книгу в каталог, удалить книгу из каталога, редактировать информацию о книге в каталоге, анализировать статистику продаж.

Также в системе будет присутствовать менеджер по работе с клиентами, который будет отвечать на интересующие пользователей вопросы и осуществлять модерацию отзывов.

1.2 Анализ программных продуктов

В настоящее время существует целый ряд интерне-магазинов для покупки печатных книг. Рассмотрим наиболее известные из них и проведем их сравнительный анализ.

1.2.1 Интернет-магазин «books.ru»

Первый книжный интернет-магазин в России. Открылся в 1996 году для поиска и покупки книг на разных языках, а также с доставкой в разные точки мира [4]. Перейдя по ссылке, пользователь попадает на главную страницу. На ней представлен каталог книг, список рекомендаций, а также поле для поиска интересующей книги.

Главная страница интернет-магазина «books.ru» представлена на рисунке 1.



Рисунок 1 – Главная страница интернет-магазина «books.ru»

К достоинствам интернет-магазина относится наличие у каждого пользователя личного кабинета, удобного интерфейса для работы с корзиной, а также наличия списка отложенных товаров.

К недостаткам данного интернет-магазина относится неудобная фильтрация книг, а также перегруженный интерфейс. Кроме того, хотя интернет-магазин и имеет мобильную версию, полноценного мобильного приложения для него не существует.

1.2.2 Интернет-магазин «Лабиринт»

Компания лабиринт работает в сфере книгоиздания, логистики и книгораспространении. В 2006 году был открыт интернет-магазин «Лабиринт» для продажи книг [5]. Перейдя по ссылке, пользователь попадает на главную страницу. На ней представлен каталог книг и поле для поиска.

Главная страница интернет-магазина «Лабиринт» представлена на рисунке 2.



Рисунок 2 – Главная страница интернет-магазина «Лабиринт»

К достоинствам интернет-магазина относится наличие у каждого пользователя личного кабинета, удобного интерфейса для работы с корзиной, наличия списка отложенных товаров, а также удобной фильтрации каталога книг по жанрам.

К недостаткам данного интернет-магазина относится неудобного отображения каталога, пользователю не всегда удобно работать с представленным каталогом. Кроме того, интернет-магазин имеет мобильную версию, также во всплывающем окне он предлагает установить мобильное приложение, но мобильное приложение не устанавливается из-за отсутствия его на маркетплейсах.

1.2.3 Мобильное приложение «Читай-город»

«Читай город» – это самая большая в России сеть книжных магазинов и интернет-магазина. Данная кампания была основана в 2008 году [6]. Перейдя в приложение, пользователь попадает на главную страницу. На ней представлен каталог книг и поле для поиска.

Главная страница приложения «Читай-город» представлена на рисунке 3.

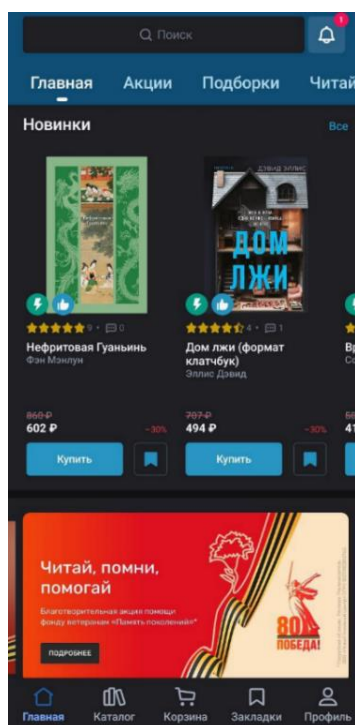


Рисунок 3 – Главная страница интернет-магазина «Читай-город»

К достоинствам приложения относится наличие у каждого пользователя личного кабинета, удобного интерфейса для работы с корзиной, наличия списка отложенных товаров, удобной фильтрации каталога книг по жанрам и интуитивно понятного интерфейса.

К недостаткам данного приложения относится то, что данное приложение специализируется не только на печатных изданиях, но также и на канцтоварах, игрушках и сувенирах.

1.2.4 Сравнение аналогов

На данном этапе был произведен сравнительный анализ рассмотренных ранее интернет-магазинов и мобильного приложения. В результате анализа была сформирована таблица 1.

Таблица 1 – Сравнение аналогов

Мобильное приложение/ интернет-магазин	Фильтрация книг	Удобство интерфейса	Наличие мобильного приложения	Отображение каталога книг	Продажа только печатной продукции
books.ru	Неудобная	Перегруженный	Нет	Удобное	нет
Лабиринт	Удобная	Интуитивно-понятный	Нет	Неудобное	нет
Читай-город	Удобная	Интуитивно-понятный	Есть	Удобное	нет
Разрабатываемое приложение	Удобная	Интуитивно-понятный	Есть	Удобное	да

Таким образом, в результате анализа систем аналогичного назначения было принято решение о разработке собственного приложения, которое будет лишено недостатков, выявленных у рассмотренных выше аналогов.

1.3 Анализ целевой аудитории

Основной целевой аудиторией являются покупатели и работники книжного магазина. Согласно отчету президента Российской академии образования, тренд на чтение бумажных книг и текстов, без учета учебной литературы, сохраняется. Так, издательский холдинг «Эксмо-АСТ» сообщает, что в 2022 году оборот печатных книг вырос на 7,5%, составив 90,7 млрд рублей.

В 2023 году рынок бумажных книг в России увеличился на 9% в денежном и на 3% в физическом выражении. В 2024 году продажи бумажных книг без учета учебной литературы выросли на 1% в натуральном и на 7% в рублевом выражении. Несмотря на рост цен на книги, читатели предпочитают покупать печатные и текстовые издания [7, 8].

На рисунке 4 представлены доли мобильных операционных систем в мировом рынке [9].



Рисунок 4 – Доли мобильных операционных систем в мировом рынке

Проанализировав рынок можно сделать вывод, что большинство людей используют операционную систему Android, вторым по численности является операционная система iOS.

На рисунке 5 представлены доли мобильных операционных систем на российском рынке [10].

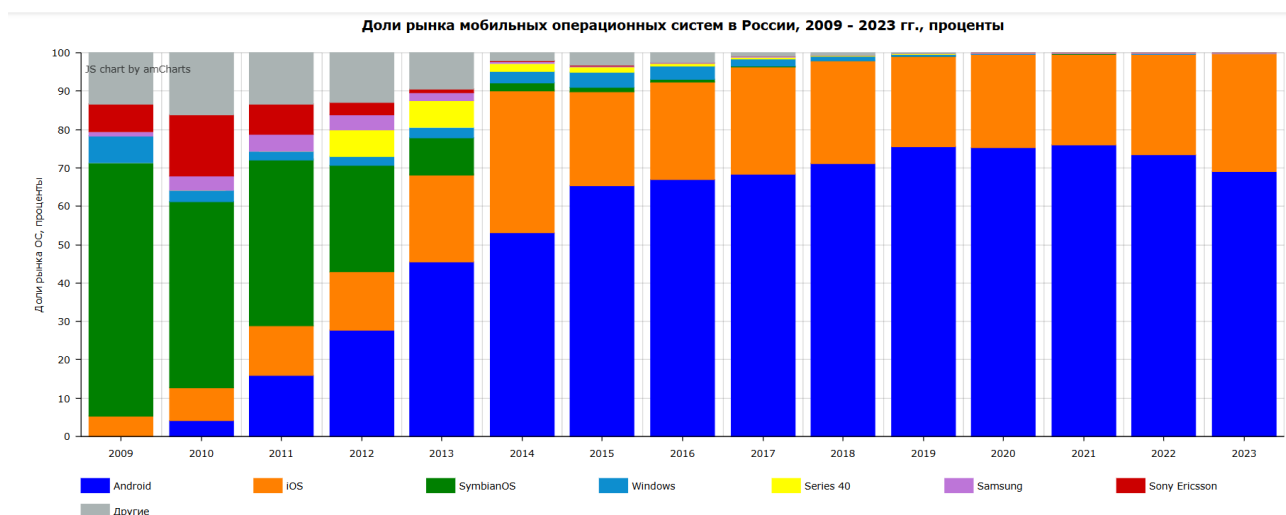


Рисунок 5 – Доли мобильных операционных систем на российском рынке

Проанализировав российский рынок мобильных операционных систем, можно сделать вывод, что несмотря на то, что с 2021 года доля операционной

системы Android постепенно падает, данная операционная система занимает лидирующие места.

Также, в России существуют свои мобильные операционные системы. Самые популярные из них это:

- «Astra Linux Mobile» – один из лидеров российских аналогов Android, главной особенностью является высокая безопасность;
- «Аврора» – российский аналог Android, обладающий высокой безопасностью и конфиденциальностью, но доступна только для юридических лиц;
- «Ред ОС М» – новая российская мобильная операционная система, аналог Android, разработана на базе операционной системы Linux, обладает хорошей безопасностью, в массовое пользование не запущена так, как проходит проверку функциональности на работниках государственных ведомств [11].

На рисунке 6 продемонстрирована статистика распространенных версий Android [12].

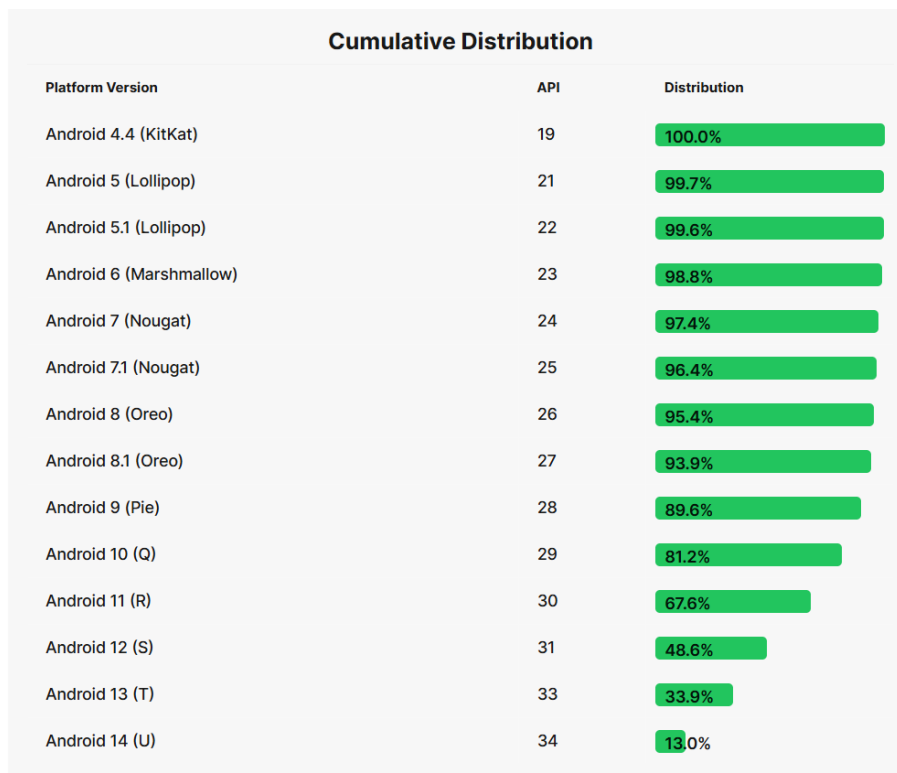


Рисунок 6 – Статистика распространённых версий Android

На основе данных статистики, можно сделать вывод, что версия Android 9 довольно распространена среди пользователей. Поэтому мобильное приложение будет разрабатываться на операционной системе Android, таким образом, чтобы оно функционировало на версии Android 9 и выше.

1.4 Анализ функциональных требований

Основными пользователями нашего приложения выступают: покупатель, администратор и менеджер по работе с клиентами. Покупатель, работая с мобильным приложением, должен иметь возможность просмотра и фильтрации каталога. Пользователь может осуществлять поиск книг в каталоге. Добавлять понравившиеся книги в корзину, для дальнейшей покупки или добавлять книги в список отложенного для дальнейшей покупки. Покупатель может посмотреть свои покупки в личном кабинете. Также покупатель может оставить отзыв на книгу и задать любой интересующий его вопрос. Покупатель может посмотреть все вопросы, которые он задал, а также ответы на вопросы.

Менеджер книжного магазина, работая с мобильным приложением, должен иметь возможность просмотра всех отзывов и вопросов. Также менеджер имеет возможность модерирования отзывов, для поддержания порядка и уважительной атмосферы на платформе. Менеджер может отвечать, на заданные пользователями вопросы.

Администратор книжного магазина, работая с мобильным приложением, должен иметь возможность как менеджера, так и покупателя, а также он может осуществлять добавление книг в каталог, удаления книг из каталога, редактирование информации о книге в каталоге и просмотра статистики продаж.

Также все пользователи имеют возможность регистрации и аутентификации в системе, для доступа к функционалу приложения.

На основе всего вышесказанного была создана диаграмма вариантов использования, представленная на рисунке 7 [13].



Рисунок 7 – Диаграмма вариантов использования мобильного приложения

На основе диаграммы вариантов использования, составим спецификации для основных вариантов использования: «Зарегистрироваться в системе» (рисунок 8), «Оформить и оплатить покупку» (рисунок 9), «Добавить книгу в каталог» (рисунок 10).

Наименование: зарегистрироваться в системе
ID: 1
Краткое описание: пользователь регистрируется в системе
Действующие лица: пользователь, СУБД
Предусловие: приложение запущено
Основной поток: <ol style="list-style-type: none"> 1. Пользователь выбирает регистрацию 2. Пользователь вводит логин и пароль 3. Система проверяет логин и пароль на корректность 4. Если логин и пароль корректны <ol style="list-style-type: none"> 4.1. Система запрашивает у СУБД данные об аккаунте 4.2. СУБД передает сведения о существовании аккаунта 4.3. Если такого аккаунта нет <ol style="list-style-type: none"> 4.3.1. Система добавляет аккаунт в БД 4.3.2. Система осуществляет вход для нового пользователя 4.4. Иначе система выводит сообщение о том, что указанный логин занят 5. Иначе система выводит сообщение о том, что указанные данные некорректны
Постусловие: регистрация выполнена

Рисунок 8 – Спецификация варианта использования «Зарегистрироваться в системе»

Наименование: добавить книгу в корзину
ID: 2
Краткое описание: покупатель добавляет книгу в корзину
Действующие лица: покупатель, СУБД
Предусловие: приложение запущено, покупатель зарегистрирован в системе, книги присутствуют в каталоге
Основной поток: <ol style="list-style-type: none"> 1. покупатель выбирает добавить книгу в корзину 2. Система проверяет доступность наличия количества книг 3. Если книги доступны для покупки <ol style="list-style-type: none"> 3.1. СУБД добавляет данные в БД 4. Иначе система выводит сообщение о недоступности добавления книги в корзину
Постусловие: книга добавлена в корзину

Рисунок 9 – Спецификация варианта использования «Добавить книгу в корзину»

Наименование: добавить книгу в каталог
ID: 3
Краткое описание: администратор добавляет книгу в каталог
Действующие лица: администратор, СУБД
Предусловие: приложение запущено, администратор зарегистрирован в системе
Основной поток: <ol style="list-style-type: none"> 1. Администратор выбирает добавление книги в каталог 2. Администратор указывает поля с информацией о книге 3. Если информация указана корректна <ol style="list-style-type: none"> 3.1. СУБД добавляет данные в БД 4. Иначе система выводит сообщение о некорректно введенных значениях
Постусловие: книга добавлена в каталог

Рисунок 10 – Спецификация варианта использования «Добавить книгу в каталог»

Также, на основе диаграммы вариантов использования, составим спецификации для основных вариантов использования. На рисунках были представлены спецификации «Написать отзыв на книгу» (рисунок 11) и «Ответить на вопрос покупателя» (рисунок 12).

Наименование: написать отзыв
ID: 4
Краткое описание: покупатель пишет отзыв о книге
Действующие лица: покупатель, СУБД
Предусловие: приложение запущено, покупатель зарегистрирован в системе
Основной поток:
1. Покупатель выбирает написание отзыва
2. Покупатель вводит текст отзыва
3. СУБД добавляет данные отзыва в БД
4. Система выводит сообщение об успешном написании отзыва
Постусловие: отзыв добавлен в БД

Рисунок 11 – Спецификация варианта использования «Написать отзыв на книгу»

Наименование: ответить на вопросы покупателя
ID: 5
Краткое описание: менеджер отвечает на вопросы покупателей
Действующие лица: менеджер, СУБД
Предусловие: приложение запущено, менеджер зарегистрирован в системе
Основной поток:
1. Менеджер выбирает ответить на вопросы
2. Если список вопросов не пуст
2.1. Менеджер пишет ответ к выбранному вопросу
2.2. СУБД добавляет текст ответа в БД
2.3. Система выводит сообщение об успешном написании ответа
3. Система выводит сообщение об отсутствии вопросов
Постусловие: ответ добавлен в БД

Рисунок 12 – Спецификация варианта использования «Ответить на вопрос покупателя»

Рассмотрим некоторые варианты использования поподробнее. Для вариантов использования «Добавить книгу в каталог» и «Добавить отзыв» составлены диаграммы деятельности. Вариант использования добавления книги администратором книжного магазина в каталог представлен на рисунке 13, вариант использования добавления отзыва покупателя представлен на рисунке 14.

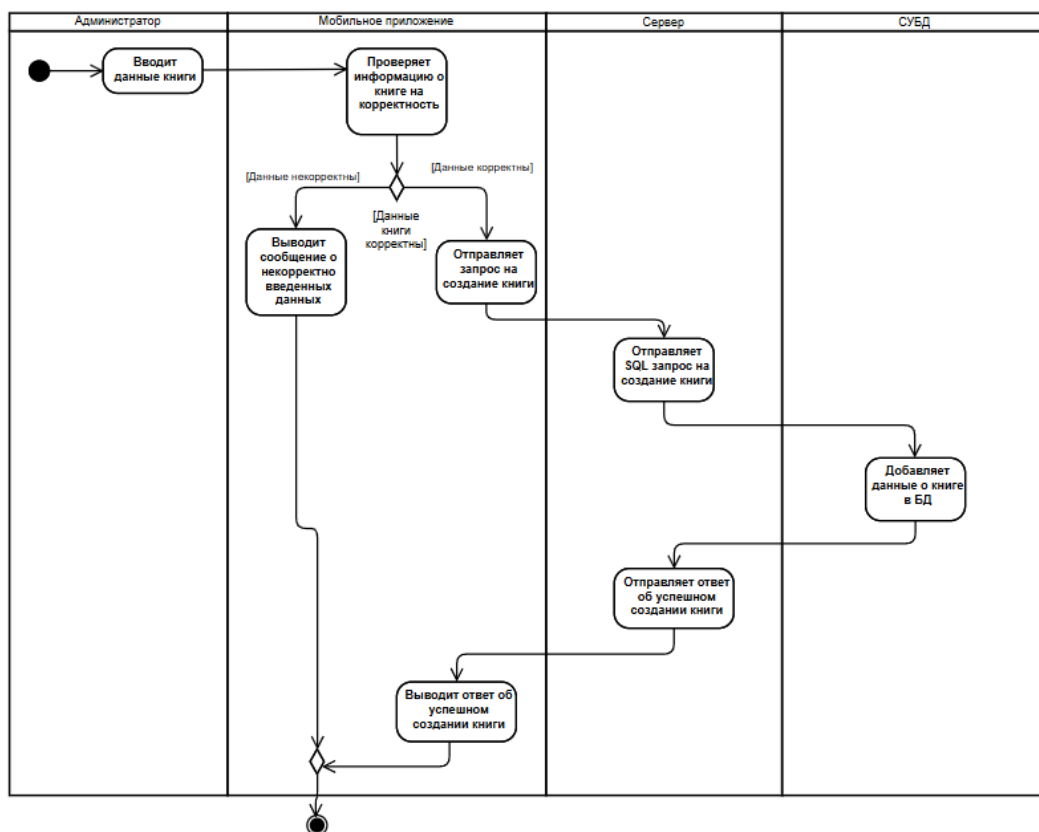


Рисунок 13 – Диаграмма деятельности варианта использования «Добавить книгу в каталог»

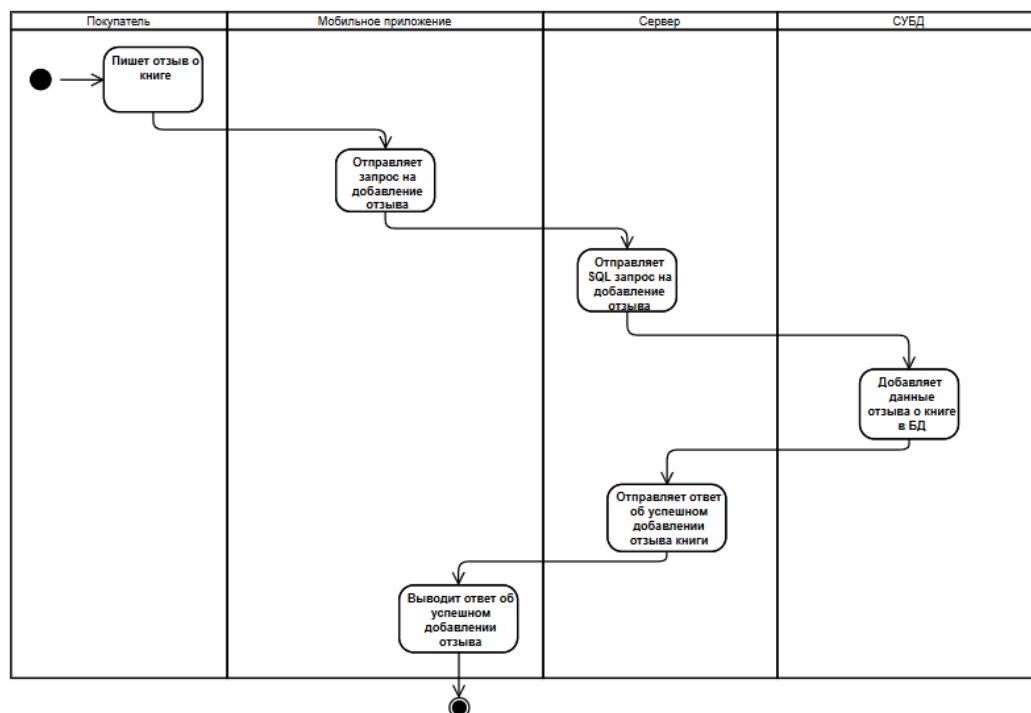


Рисунок 14 – Диаграмма деятельности варианта использования «Написать ОТЗЫВ»

Также для варианта использования «Добавить книгу в корзину» и для варианта использования «Проанализировать продажи» были составлены диаграммы последовательности. Диаграмма последовательности добавления книги в корзину была представлена на рисунке 15, диаграмма последовательности анализа продаж представлена на рисунке 16.

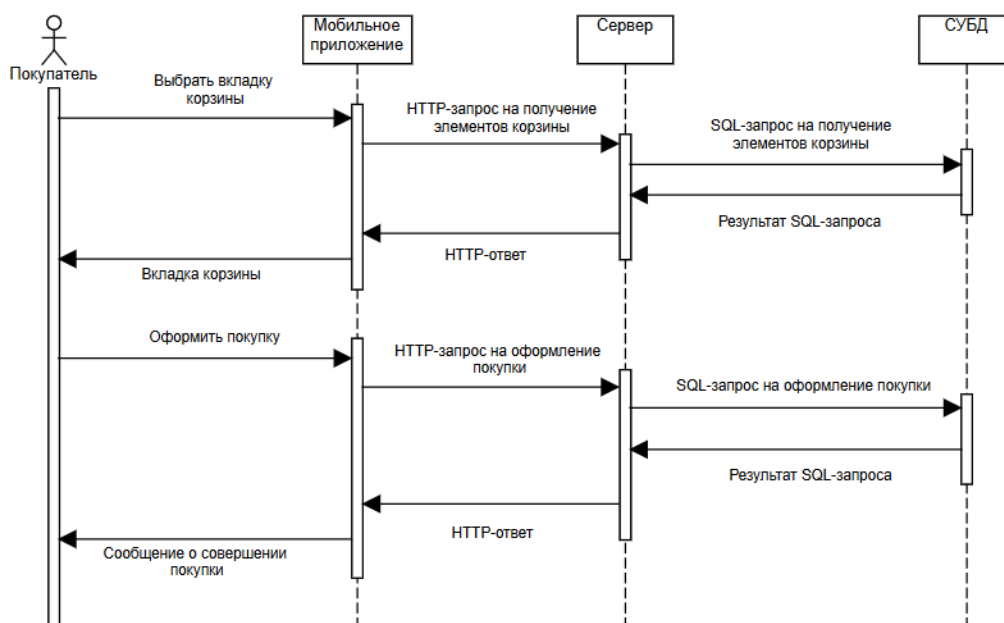


Рисунок 15 – Диаграмма последовательности варианта использования «Добавить книгу в корзину»

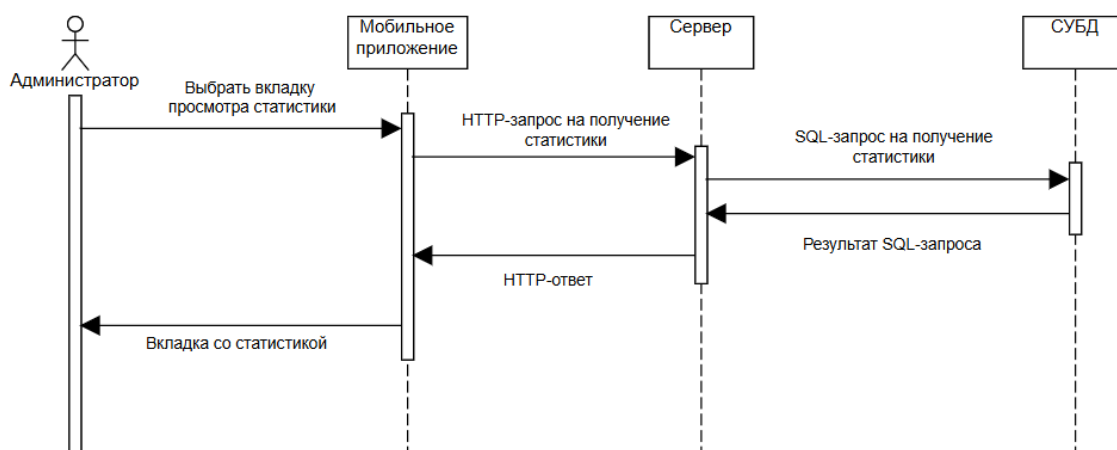


Рисунок 16 – Диаграмма последовательности варианта использования «Проанализировать продажи»

На текущем этапе были определены ключевые функции разрабатываемого мобильного приложения. В дополнение к этому были построены диаграммы вариантов использования, деятельности и последовательности, а также подготовлены спецификации для отдельных вариантов использования.

1.5 Анализ нефункциональных требований

На данном этапе были определены нефункциональные требования к разрабатываемому программному обеспечению. В качестве методологии разработки использована технология – RUP. В качестве шаблона проектирования, использован шаблон – MVC [14, 15].

Серверная часть приложения будет разработана с использованием языка C#, в среде выполнения Visual Studio 2022, с использованием платформы ASP.Net [16]. Данные на сервере должны сохраняться в базе данных, созданной с использованием СУБД MS SQL Server 2019. Взаимодействие с базой данных должно осуществляться с помощью SQL-запросов через фреймворк Entity Framework Core [17]. Серверная часть приложения должна функционировать под управлением операционной системы Windows 10 или более новой версии.

Клиентская часть приложения должна быть разработана на языке программирования C# с использованием платформы .NET MAUI в среде Microsoft Visual Studio 2022 [18]. Она должна представлять собой мобильное решение с удобным и интуитивно понятным пользовательским интерфейсом. Клиентская часть должна быть тщательно протестирована и корректно функционировать на устройствах под управлением операционной системы Android версии 9.0 и выше.

1.6 Планирование разработки и оценка бюджета

Эффективная реализация любого проекта, включая проекты в сфере информационных технологий, невозможна без тщательного и продуманного планирования. Именно оно служит основой для организации всех этапов работы таким образом, чтобы достичь поставленных целей в оптимальные сроки, с

минимальными затратами и на должном уровне качества [19–23]. Кроме того, грамотное планирование способствует снижению рисков, повышению управляемости проекта и позволяет более гибко реагировать на возникающие изменения или непредвиденные обстоятельства в ходе его реализации.

Задачи, представляющие собой этапы работы над проектом, представлены в таблице 2.

Таблица 2 – Перечень задач проекта

№	Название задачи	Длительность	Предшественники
1	Начало проекта	–	–
2	Анализ требований	–	–
3	Анализ предметной области и постановка задачи	2 дня	1
4	Анализ программных продуктов аналогичного назначения	3 дня	1
5	Анализ функциональных требований	2 дня	3;4
6	Анализ требований к интерфейсу	2 дня	5
7	Анализ целевой аудитории	1 день	6
8	Анализ требований завершен	-	7
9	Выбор технологий разработки ПО	-	-
10	Анализ существующих технологий разработки ПО	3 дня	8
11	Выбор технологии разработки ПО	2 дня	10
12	Выбор решения завершен	-	11
13	Проектирование программных средств	-	-
14	Проектирование базы данных	5 дней	8;12
15	Проектирование архитектуры серверной части приложения	5 дней	14
16	Проектирование архитектуры клиентской части приложения	5 дней	12
17	Проектирование приложения завершенно	-	16
18	Реализация программных средств	-	-
19	Реализация базы данных	5 дней	14
20	Реализация серверной части приложения	22 дня	15;19
21	Реализация клиентской части приложения	15 дней	17
22	Реализация завершена	-	20;21
23	Тестирование	-	-
24	Функциональное тестирование	2 дня	22
25	Модульное тестирование	2 дня	22
26	Тестирование завершенно	-	24;25
27	Составление и оформление документации	7	26

Продолжение таблицы 2

№	Название задачи	Длительность	Предшественники
28	Конец проекта	-	27

Таким образом, для реализации проекта нужно выполнить 28 задач.

В таблице 3 представлено описание затрачиваемых ресурсов.

Таблица 3 – Список ресурсов проекта

Название ресурса	Тип	Стандартная ставка	Ставка сверхурочных
Программист 1	Трудовой	400 Р/ч	500 Р/ч
Программист 2	Трудовой	400 Р/ч	500 Р/ч
Аналитик	Трудовой	400 Р/ч	500 Р/ч
Дизайнер	Трудовой	400 Р/ч	500 Р/ч
Тестировщик	Трудовой	300 Р/ч	350 Р/ч
Интернет	Затраты	2500 Р/месяц	
Компьютер	Материальный	500 Р	
Мобильное устройство	Материальный	300 Р	
Бумага	Материальный	500 Р	

Таким образом, в общей сложности, потребуется 10 видов ресурсов.

Распределение ресурсов по задачам представлено в таблице 4.

Таблица 4 – Назначение ресурсов

№	Название задачи	Ресурс	Единицы
1	Начало проекта	—	—
2	Анализ требований	—	—
3	Анализ предметной области и постановка задачи	Аналитик Компьютер Интернет	50% 1 шт.
4	Анализ программных продуктов аналогичного назначения	Аналитик Компьютер Интернет Мобильное устройство	50% 1 шт. 1 шт.
5	Анализ функциональных требований	Аналитик Компьютер Интернет	70% 1 шт.

Продолжение таблицы 4

№	Название задачи	Ресурс	Единицы
5	Анализ функциональных требований	Аналитик Компьютер Интернет	70% 1 шт.
6	Анализ требований к интерфейсу	Дизайнер Аналитик Компьютер Интернет	50% 50% 1 шт.
7	Анализ целевой аудитории	Аналитик Компьютер Интернет	50% 1 шт.
8	Анализ требований завершён	—	—
9	Выбор технологий разработки ПО	—	—
10	Анализ существующих технологий разработки ПО	Программист 1 Программист 2 Аналитик Компьютер Интернет	50% 50% 25% 3 шт.
11	Выбор технологий разработки ПО	Программист 1 Программист 2 Аналитик Компьютер Интернет	50% 50% 30% 3 шт.
12	Выбор средств разработки ПО завершён	—	—
13	Проектирование программных средств	—	—
14	Проектирование базы данных	Программист 1 Программист 2 Компьютер Интернет	70% 20% 2 шт.
15	Проектирование архитектуры серверной части приложения	Программист 1 Программист 2 Компьютер Интернет	50% 50% 2 шт.
16	Проектирование архитектуры клиентской части приложения	Программист 1 Программист 2 Дизайнер Компьютер Интернет	50% 50% 100% 3 шт.
17	Проектирование приложения завершено	—	—
18	Реализация программных средств	—	—

Окончание таблицы 4

№	Название задачи	Ресурс	Единицы
19	Реализация базы данных	Программист 1 Программист 2 Компьютер Интернет	100% 50% 1 шт.
20	Реализация серверной части приложения	Программист 1 Компьютер Интернет	100% 2 шт.
21	Реализация мобильного приложения	Программист 2 Дизайнер Компьютер Интернет	100% 100% 2 шт.
22	Реализация завершена	—	—
23	Тестирование	—	—
24	Функциональное тестирование	Программист 1 Тестировщик Компьютер Интернет Мобильное устройство	20% 100% 2 шт. 2 шт.
25	Модульное тестирование	Программист 2 Тестировщик Компьютер Интернет	20% 100% 2 шт.
26	Тестирование завершено	—	—
27	Составление и оформление документации	Программист 1 Программист 2 Тестировщик Аналитик Компьютер Интернет	100% 100% 100% 100% 4 шт.
28	Конец проекта	—	—

Таким образом, реализацией проекта будут заниматься два программиста, тестировщик, аналитик и один дизайнер, которым для работы нужны четыре компьютера, два мобильных устройства, доступ к интернету и бумага.

Для вычисления длительности реализации проекта в днях и его критического пути построен сетевой график, представленный ниже на рисунке 17.

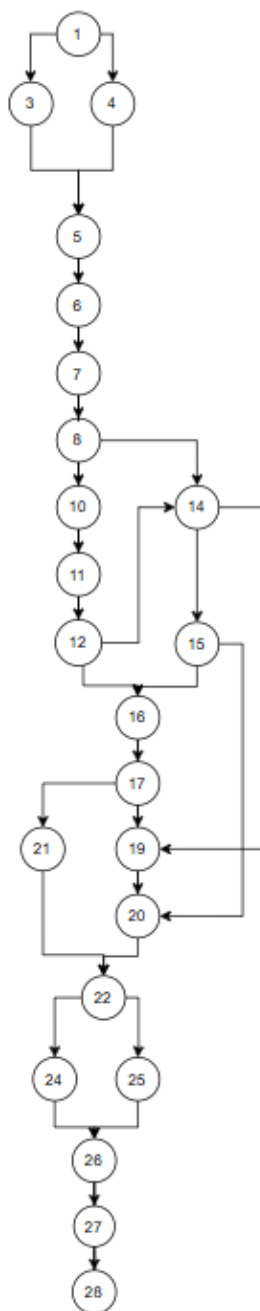


Рисунок 17 – Сетевой график проекта

В таблице 5 представлены расчеты, сделанные вручную, раннего времени начала работ для задач проекта.

Таблица 5 – Расчеты раннего времени начала работ

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
1	-	-	-	-	0

Продолжение таблицы 5

Работа	Предшественники			Вычисление	Раннее время начала
	№	Раннее время начала	Длительность		
3	1	0	0	0+0	0
4	1	0	0	0+0	0
5	3	0	2	Max(0+2;0+3)	3
	4	0	3		
6	5	3	2	3+2	5
7	6	5	2	5+2	7
8	7	7	1	7+1	8
10	8	8	3	8+3	11
11	10	11	2	11+2	13
12	11	11	0	11+0	11
14	8	8	0	Max(6+0;11+0)	11
	12	11	0		
15	14	11	5	11+5	16
16	12	11	0	Max(11+0;16+5)	21
	15	16	5		
17	16	21	5	21+5	26
19	14	11	5	Max(11+5;26+0)	26
	17	26	0		
20	15	15	5	Max(15+5;26+5)	31
	19	26	5		
21	17	25	0	25+0	25
22	20	31	22	Max(31+22;25+15)	53
	21	25	15		
24	22	53	0	53+0	53
25	22	53	0	53+0	53
26	24	53	2	Max(53+2;53+2)	55
	25	53	2		
27	26	55	7	55+7	62
28	27	62	0	62+0	62

Также было рассчитано позднее время начала работ для задач проекта.

Расчеты представлены в таблице 6.

Таблица 6 – Расчеты позднего времени начала работ

Работа	Последователи			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность		
28	-	-	-		62

Продолжение таблицы 6

Работа	Последователи			Вычисление	Позднее время начала
	№	Позднее время начала	Длительность		
27	28	62	0	62-0	62
26	27	62	7	62-7	55
25	26	55	0	55-0	55
24	26	55	0	55-0	55
22	24	55	2	Min(55-2;55-2)	53
	25	55	2		
21	22	53	0	53-0	53
20	22	53	0	53-0	53
19	20	53	22	53-22	31
17	19	31	5	Min(31-5;53-15)	26
	21	53	15		
16	17	26	0	26-0	26
15	16	26	0	Min(26-0;53-22)	26
	20	53	22		
14	15	26	5	Min(26-5;31-5)	21
	19	31	5		
12	14	21	5	Min(21-5;26-5)	16
	16	26	5		
11	12	16	0	16-0	16
10	11	16	2	16-2	14
8	10	14	3	Min(14-3;12-5)	7
	14	12	5		
7	8	7	0	7-0	7
6	7	7	1	7-1	6
5	6	6	1	6-1	5
4	5	5	2	5-2	3
3	5	4	2	4-2	2
1	3	2	2	Min(2-2;3-3)	0
	4	3	3		

По результатам нахождения раннего и позднего времени начала работ можно найти критический путь проекта. Критический путь проекта – это последовательность задач, которая определяет расчётную дату окончания проекта. Завершение последней задачи критического пути ведёт к завершению всего проекта. Вычисление критического пути представлено в таблице 7.

Таблица 7 – Вычисление критического пути проекта

Номер работы	Раннее время начала	Позднее время начала	Резерв времени
1	0	0	0
3	0	2	2
4	0	3	3
5	3	5	2
6	5	6	1
7	7	7	0
8	8	8	0
10	11	14	3
11	13	16	3
12	11	16	5
14	11	21	10
15	16	26	10
16	21	26	5
17	26	26	0
19	26	31	5
20	31	53	22
21	25	53	22
22	53	53	0
24	53	55	2
25	53	55	2
26	55	55	0
27	62	62	0
28	62	62	0

На основе полученных данных из расчета, можно сделать важный и небезосновательный вывод о том, что критический путь проекта проходит через узлы графа, с номерами работ 1, 3, 5, 6, 7, 8, 10, 11, 12, 16, 17, 21, 22, 24, 26, 27, 28. Общее время, затрачиваемое на выполнение всего проекта, составляет 62 рабочих дня. Граф проекта, где пунктирными линиями указан путь, показывающий за какое минимальное время, можно завершить всю работу над проектом, приведен на рисунке 18. Задачи из этой последовательности называют критическими: любые задержки или проблемы, возникающие во время их выполнения могут сдвинуть не только сроки следующей задачи, но и сроки всего проекта в целом.

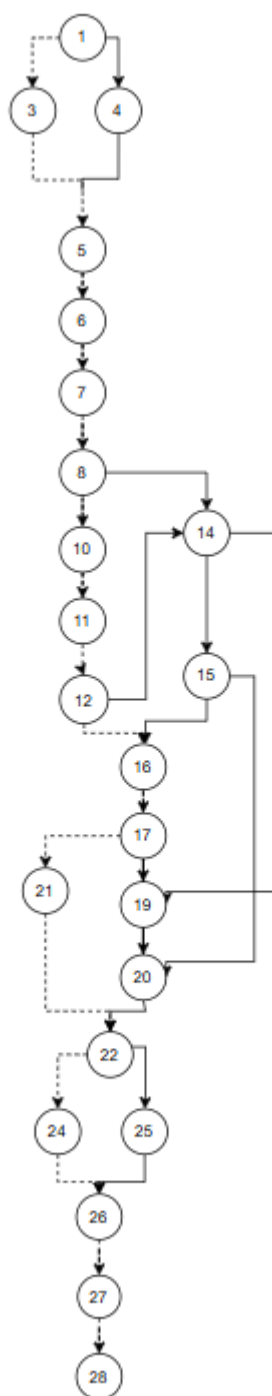


Рисунок 18 – Критический путь проекта

Для проверки корректности расчетов продолжительности проекта, а также для вычисления бюджета, необходимого для создания приложения, воспользуемся системой управления проектами Microsoft Project. Данная программа позволяет автоматически производить все вычисления, основываясь на данных пользователя.

Список задач из таблицы 4, внесённый в систему Microsoft Project, представлен на рисунке 19. В качестве времени начала проекта была принята дата 24.02.2025.

	📁	Название задачи	Длительность	Начало	Окончание	Предшественники
1	📁	Начало проекта	0 дней	Пн 24.02.25	Пн 24.02.25	
2	📁	Анализ требований	8 дней	Пн 24.02.25	Пн 03.03.25	1
3	📁	Анализ предметной области и постановка задачи	2 дней	Пн 24.02.25	Вт 25.02.25	
4	📁	Анализ программных продуктов аналогичного на	3 дней	Пн 24.02.25	Ср 26.02.25	
5	📁	Анализ функциональных требований	2 дней	Чт 27.02.25	Пт 28.02.25	3;4
6	📁	Анализ требований к интерфейсу	2 дней	Сб 01.03.25	Вс 02.03.25	5
7	📁	Анализ целевой аудитории	1 день	Пн 03.03.25	Пн 03.03.25	6
8	📁	Анализ требований завершен	0 дней	Пн 03.03.25	Пн 03.03.25	7
9	📁	Выбор технологий разработки ПО	5 дней	Вт 04.03.25	Сб 08.03.25	
10	📁	Анализ существующих технологий разработки П	3 дней	Вт 04.03.25	Чт 06.03.25	
11	📁	Выбор технологии разработки ПО	2 дней	Пт 07.03.25	Сб 08.03.25	10
12	📁	Выбор решения завершен	0 дней	Сб 08.03.25	Сб 08.03.25	11
13	📁	Проектирование программных средств	10 дней	Вс 09.03.25	Вт 18.03.25	
14	📁	Проектирование базы данных	5 дней	Вс 09.03.25	Чт 13.03.25	8;12
15	📁	Проектирование архитектуры серверной части п	5 дней	Пт 14.03.25	Вт 18.03.25	14
16	📁	Проектирование архитектуры клиентской части п	5 дней	Вс 09.03.25	Чт 13.03.25	12
17	📁	Проектирование приложения завершенно	0 дней	Вт 18.03.25	Вт 18.03.25	16;15;14
18	📁	Реализация программных средств	27 дней	Чт 20.03.25	Вт 15.04.25	
19	📁	Реализация базы данных	5 дней	Чт 20.03.25	Пн 24.03.25	14
20	📁	Реализация серверной части приложения	22 дней	Вт 25.03.25	Вт 15.04.25	15;19
21	📁	Реализация клиентской части приложения	15 дней	Чт 20.03.25	Чт 03.04.25	16
22	📁	Реализация завершен	0 дней	Вт 15.04.25	Вт 15.04.25	20;21
23	📁	Тестирование	11 дней	Ср 16.04.25	Сб 26.04.25	
24	📁	Функциональное тестирование	4 дней	Ср 16.04.25	Сб 19.04.25	22
25	📁	Модульное тестирование	4 дней	Ср 16.04.25	Сб 19.04.25	22
26	📁	Тестирование завершенно	0 дней	Сб 19.04.25	Сб 19.04.25	24;25
27	📁	Составление и оформление документации	7 дней	Вс 20.04.25	Сб 26.04.25	26
28	📁	Конец проекта	0 дней	Сб 26.04.25	Сб 26.04.25	27

Рисунок 19 – Задачи проекта в среде Microsoft Project

Построенная системой Microsoft Project диаграмма Ганта приведена на рисунке 20.

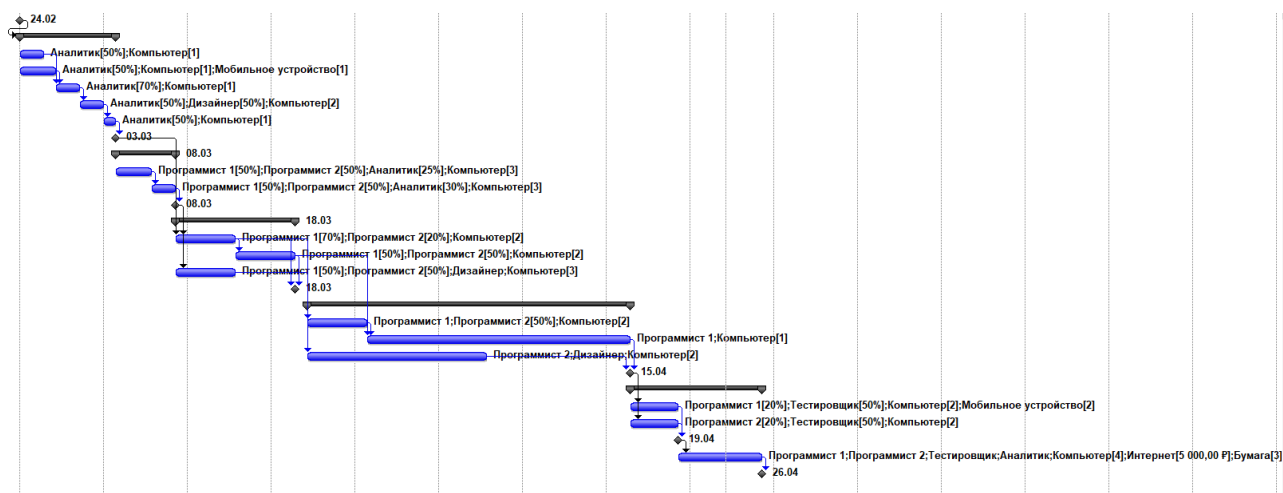


Рисунок 20 – Диаграмма Ганта

На рисунке 21 представлен график трудового ресурса «Аналитик».

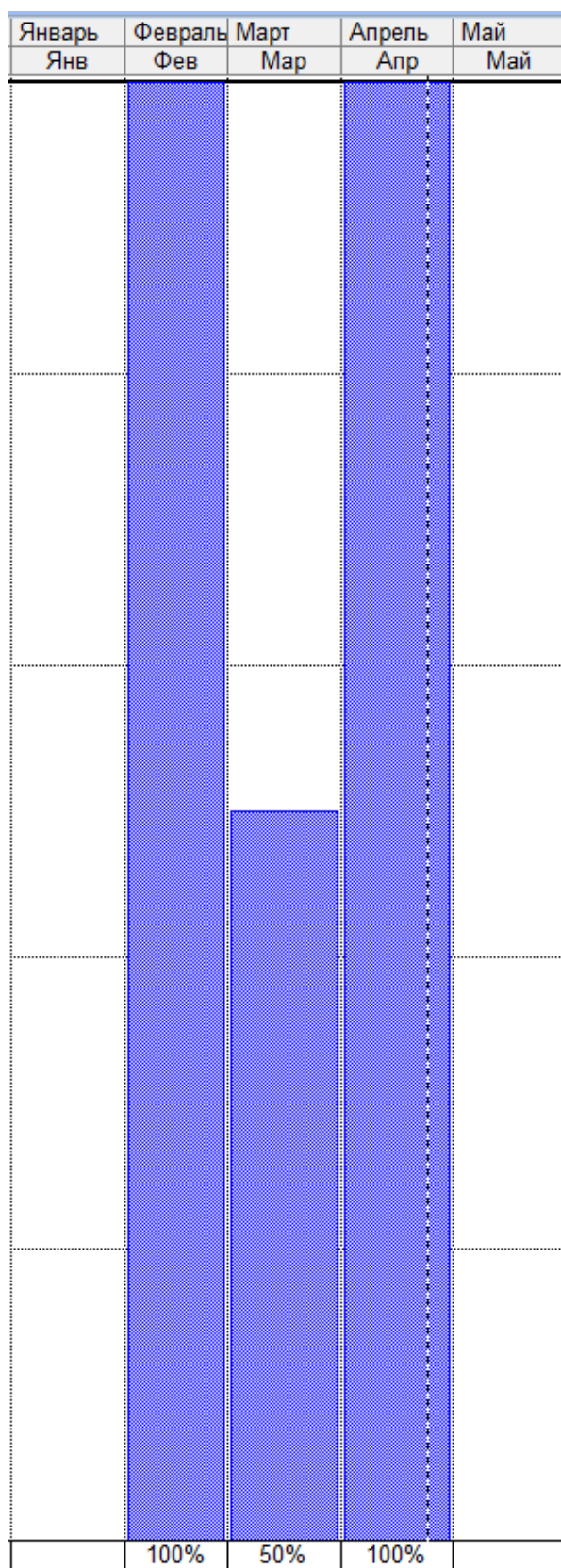


Рисунок 21 – График ресурсов для ресурса «Аналитик»

На рисунке 22 представлен график трудового ресурса «Тестировщик».

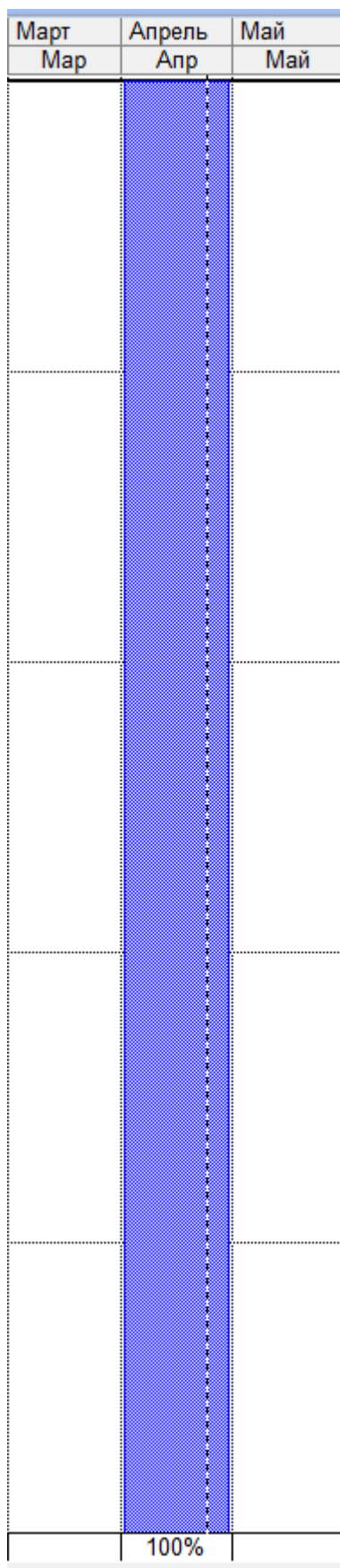


Рисунок 22 – График ресурсов для ресурса «Тестировщик»

На рисунке 23 представлен график трудового ресурса «Дизайнер».

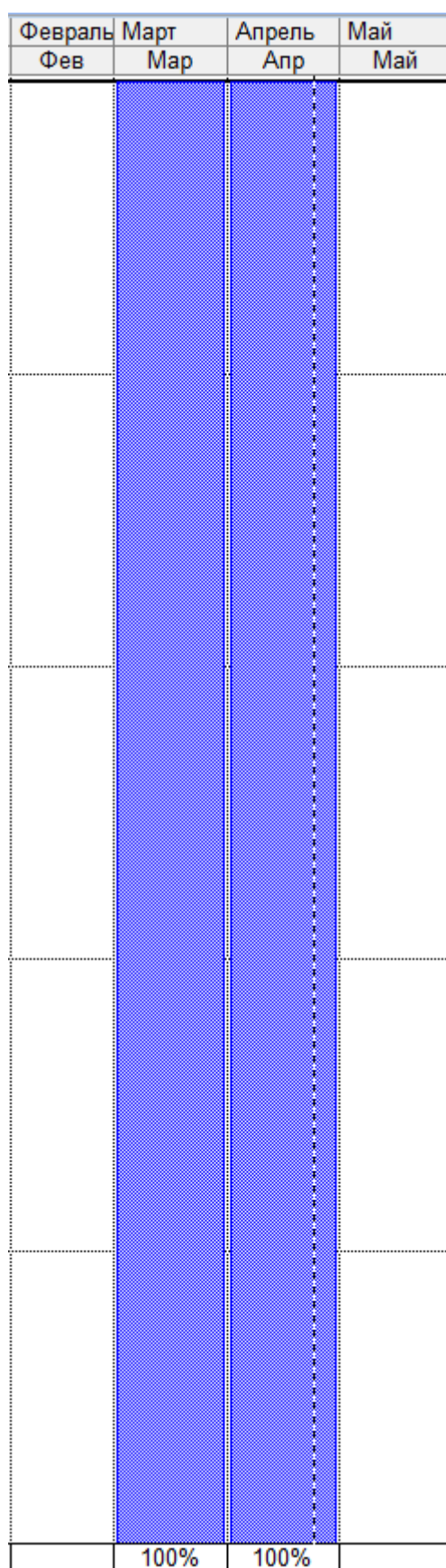


Рисунок 23 – График ресурсов для ресурса «Дизайнер»

На рисунке 26 представлен график трудового ресурса «Бумага».

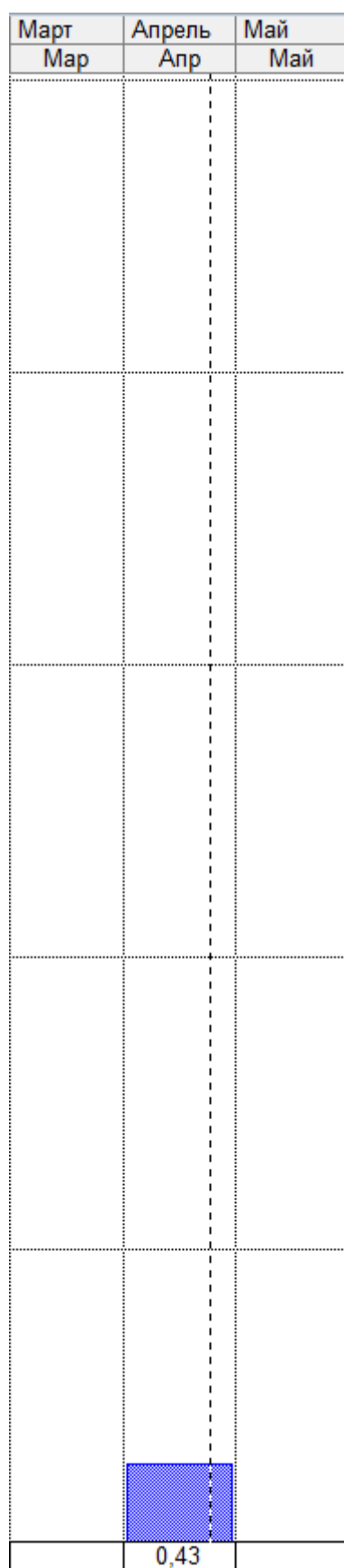


Рисунок 26 – График ресурсов для ресурса «Бумага»

На рисунке 27 представлен график трудового ресурса «Программист 1».

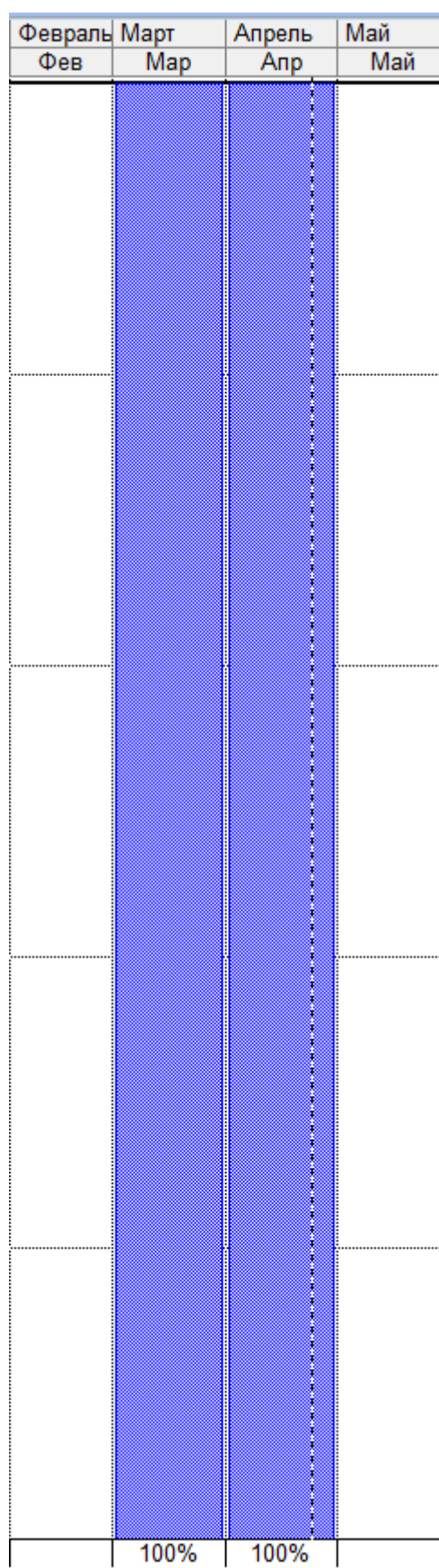


Рисунок 27 – График ресурсов для ресурса «Программист 1»

На рисунке 28 представлен график трудового ресурса «Программист 2».

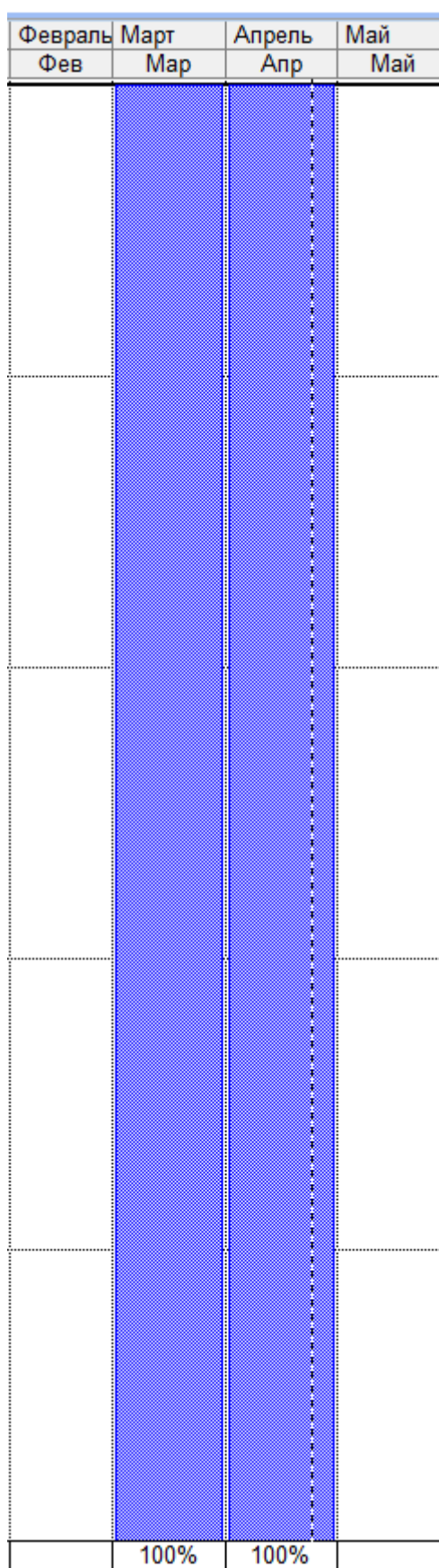


Рисунок 28 – График ресурсов для ресурса «Программист 2»

На рисунке 29 представлен обзор проекта. Из рисунка видно, что общая длительность проекта составляет 62 дней (что совпадает со значением, вычисленным вручную), а предполагаемые затраты – 427 860 руб.

Проект1.mpp

на дату Чт 08.05.25

Даты			
Начало:	Пн 24.02.25	Окончание:	Сб 26.04.25
Баз. начало:	НД	Баз. окончание:	НД
Факт. начало:	НД	Факт. окончание:	НД
Отклон. начала:	0 дней	Отклон. окончания:	0 дней
Длительность			
Текущий план:	62 дней	Оставшаяся:	62 дней
Базовый план:	0 дней?	Фактическая:	0 дней
Отклонение:	62 дней	Процент завершения:	0%
Трудозатраты			
Текущий план:	1 060,4 ч	Оставшиеся:	1 060,4 ч
Базовый план:	0 ч	Фактические:	0 ч
Отклонение:	1 060,4 ч	Процент завершения:	0%
Затраты			
Текущий план:	427 860,00 Р	Оставшиеся:	427 860,00 Р
Базовый план:	0,00 Р	Фактические:	0,00 Р
Отклонение:	427 860,00 Р		
Состояние задачи		Состояние ресурса	
Еще не начатые задачи	28	Трудовые ресурсы:	3
Выполняющиеся задачи	0	Трудовые ресурсы с прев. дос:	3
Завершенные задачи:	0	Материальные ресурсы:	3
Итого задач:	28	Итого ресурсов:	9

Рисунок 29 – Обзорный отчёт проекта

На основе произведённых оценок стоимости и сроков проекта можно сделать вывод о его экономической целесообразности.

2 Проектирование мобильного приложения

2.1 Выбор технологий и инструментальных средств

В результате проектирования программного обеспечения было решено использовать одну из самых известных методологий разработки Rational Unified Process (RUP). RUP – универсальная методология, для создания высококачественного программного обеспечения, особенностью которой, является четкое распределение задач и сфер ответственности при разработке программного обеспечения. Данная методология состоит из 5 основных фаз:

- начало – определяются требования клиентов, составляется план проекта, цель проекта, рассчитываются риски проекта, создается модель вариантов использования и описание проекта, определяется масштаб проекта на основе диаграммы вариантов использования, что позволяет рассчитать затрачиваемое время и ресурсы;
- разработка – на данном этапе выполняется планирование и моделирование, разрабатывается план развития;
- конструкция – создание проекта (кодирование), тестирование проекта;
- переходный период – проект публикуется для потребителя, обновляется документация проекта, устраняются дефекты, выявленные потребителями;
- производство – проект поддерживается и обновляется.

Каждая фаза завершается вехой, на которой команда разработчиков оценивает выполнение поставленных целей и принимает критически важные решения о ходе дальнейшей работы над продуктом.

Выбор методологии RUP был сделан на основе ее преимуществ. Главными достоинствами данной методологии является ее четкое планирование и распределение задач и сфер ответственности, что позволяет сократить время необходимое на разработку. Также, достоинством данной методологии является

гибкость разработки и возможность быстро вносить коррективы, для устранения проблем [14].

Для составления моделей и анализа требований, в соответствии с методологией RUP, было использован язык моделирования UML. Для построения диаграмм использовалось средство UML-моделирования Visual Paradigm.

Для планирования было использовано приложение Microsoft Project. Оно позволяет составить план разработки, четко распределить задачи, рассчитать сроки проекта, затраты проекта, выраженные в денежном, трудовом и материальном эквиваленте. Использовать графики загруженности и прочие функции для осуществления эффективного управления проектом, на всех стадиях его создания.

Для создания мобильного приложения было решено использовать язык программирования C#, в среде Microsoft Visual Studio 2022. В качестве используемой платформы, было принято решение использовать ASP.Net. Для реализации серверной части приложения, был использован шаблон проектирования MVC. Для взаимодействия с базой данных, был использован фреймворк Entity Framework Core, взаимодействие происходит через SQL-запросы. Для реализации клиентской части приложения была использована платформы .NET MAUI. Для хранения данных используется БД, работающая под управлением СУБД Microsoft SQL Server [24]. Проектирование базы данных выполнялось с использованием приложения Visual Paradigm.

2.2 Проектирование архитектуры программного обеспечения

Было принято решение использовать в приложении трехуровневую архитектуру «клиент-сервер».

В данной архитектуре приложение делится на 3 отдельных уровня:

- уровень представления (пользовательский интерфейс) – отвечает за ввод и вывод данных пользователю;

- уровень приложения (бизнес-логика) – отвечает за обработку запросов пользователей, выполняет вычисления и принимает решения, выступает в качестве посредника между уровнями представления и данных;
- уровень данных (управление данными) – отвечает за управление и хранение данных.

Данная архитектура была выбрана на основе ее преимуществ. Данная архитектура легко масштабируется независимо от другого уровня. Она удобна в обслуживании, обновление или изменение одного уровня, не затрагивает другие. Также, главными достоинствами данной архитектуры являются ее гибкость, производительность и безопасность [25]. Её схема представлена на рисунке 30.

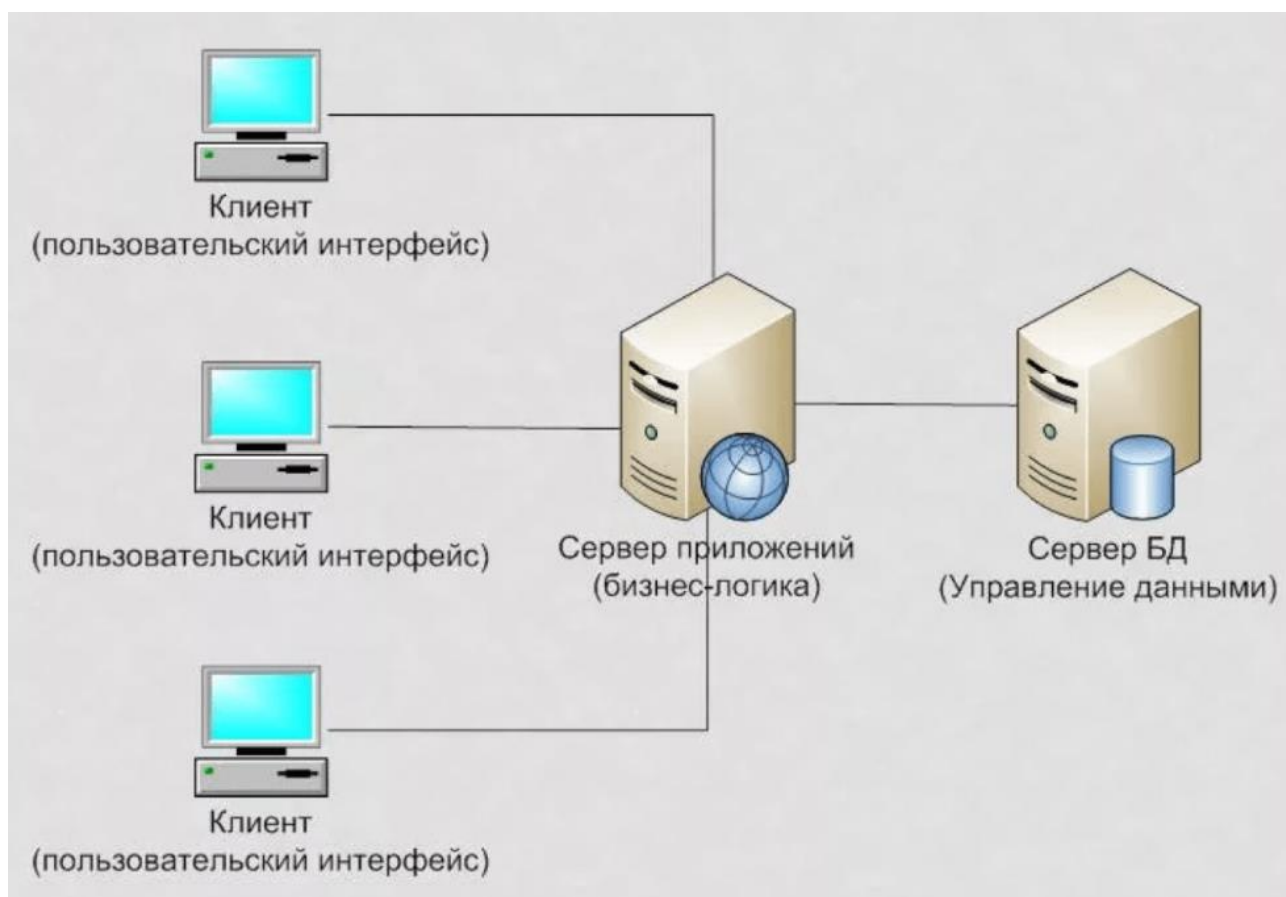


Рисунок 30 – Трехуровневая архитектура «Клиент-сервер»

Взаимодействие в данной архитектуре происходит следующим образом: клиент отправляет http-запрос серверу, сервер обрабатывает этот запрос, выполняет необходимые вычисления и при необходимости отправляет SQL-

запрос базе данных, СУБД обрабатывает SQL-запрос и возвращает его результаты серверу, сервер на основе полученных результатов от базы данных формирует ответ на запрос клиента и отправляет его ему, клиент получает ответ на запрос и представляет его в удобочитаемой форме пользователю приложения.

2.3 Проектирование базы данных

2.3.1 Концептуальное проектирование базы данных

На основе анализа предметной области и диаграммы вариантов использования была построена концептуальная модель базы данных. Графическое представление концептуальной модели БД в виде диаграммы «сущность-связь» представлено на рисунке 31.

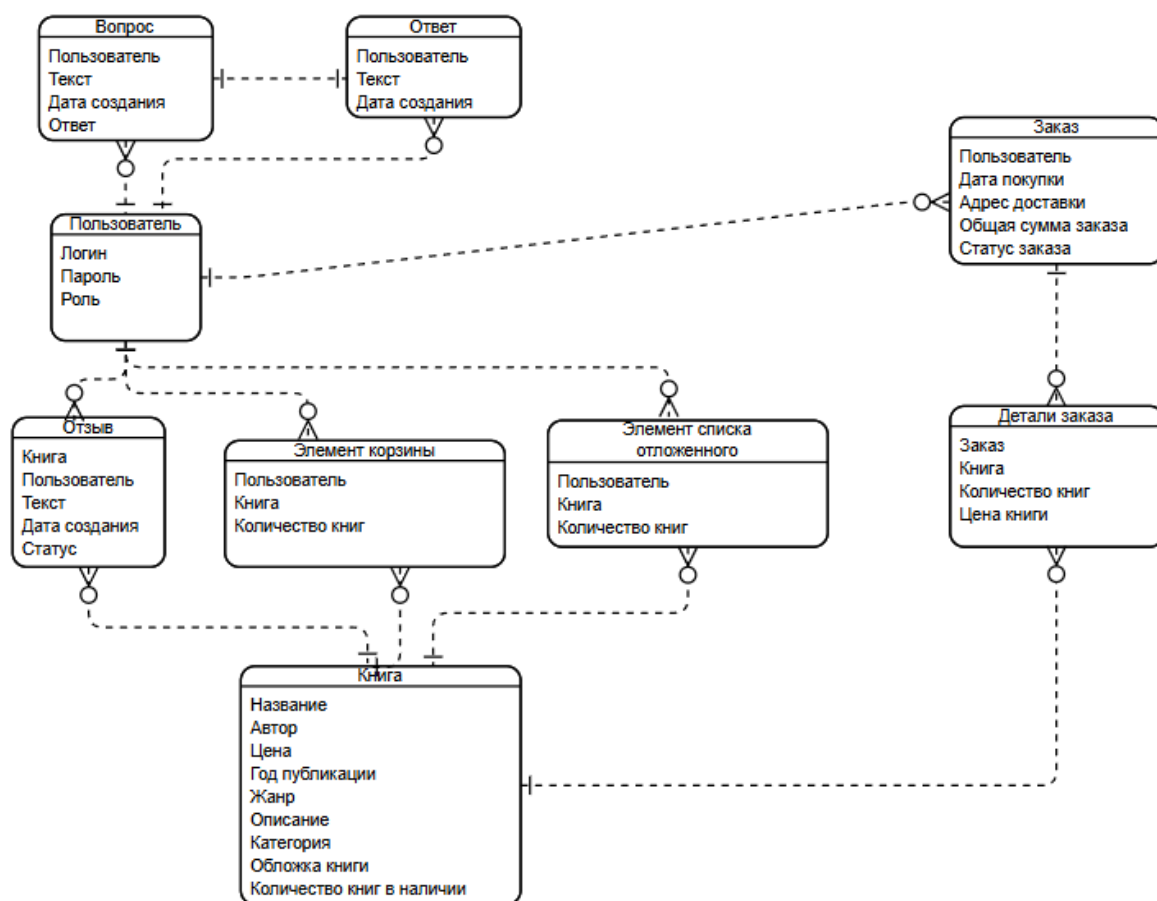


Рисунок 31 – Графическое представление концептуальной модели базы данных

Сущности концептуальной модели данных представлены в таблице 8.

Таблица 8 – Сущности концептуальной модели базы данных

№	Название	Описание	Количество экземпляров	Режим обновления
1	Пользователь	Список покупателей, менеджеров по работе с клиентами и администраторов	10000	При регистрации нового пользователя в системе
2	Отзыв	Список отзывов	300000	При добавлении нового отзыва или редактировании существующего
3	Элемент корзины	Список элементов корзины	300000	При добавлении, удалении элемента корзины
4	Элемент списка отложенного	Список элементов списка отложенного	300000	При добавлении, удалении элемента списка отложенного
5	Книга	Список книг каталога	30000	При добавлении, удалении, редактировании каталога книг
6	Заказ	Список заказов	30000	При оформлении и оплаты покупки, при отмене заказа
7	Детали заказа	Список деталей заказа	300000	При оформлении и оплаты покупки
8	Вопрос	Список вопросов пользователей	10000	При добавлении нового вопроса
9	Ответ	Список ответов пользователей	10000	При добавлении нового ответа

В таблице 9 представлено описание связей между сущностями.

Таблица 9 – Описание связей между сущностями концептуальной модели

№	Связь	Описание	Мощность	Обязательность
1	Пользователь – Отзыв	Определяет отзывы пользователя	1:M	Да
2	Пользователь – Элемент корзины	Определяет элементы корзины пользователя	1:M	Да
3	Пользователь – Заказ	Определяет заказы пользователя	1:M	Да
4	Пользователь – Элемент списка отложенного	Определяет отложенные книги пользователя	1:M	Да
5	Книга – Отзыв	Определяет отзывы книги	1:M	Да
6	Книга – Элемент корзины	Определяет книги в корзине	1:M	Да
7	Книга – Детали заказа	Определяет книги в заказе	1:M	Да
8	Книга – Элемент списка отложенного	Определяет книги в покупке	1:M	Да
9	Заказ – Детали заказа	Определяет дополнительную информацию о заказе	1:M	Да
10	Пользователь – Вопрос	Определяет вопросы пользователей	1:M	Да
11	Вопрос – Ответ	Определяет ответ на вопрос	1:M	Да

Таким образом, были спроектированы основные сущности с атрибутами и связями между ними, без включения атрибутов первичных и внешних ключей.

2.3.2 Логическое проектирование базы данных

На основании концептуальной модели была спроектирована логическая модель базы данных, представленная на рисунке 32.

На данной логической модели структура базы данных отображена уже более подробно. Здесь определены первичные ключи для уникальной идентификации записей, внешние ключи, обеспечивающие целостность данных и устанавливающие связи между таблицами, атрибуты сущностей, ограничения атрибутов и типы связей между сущностями.

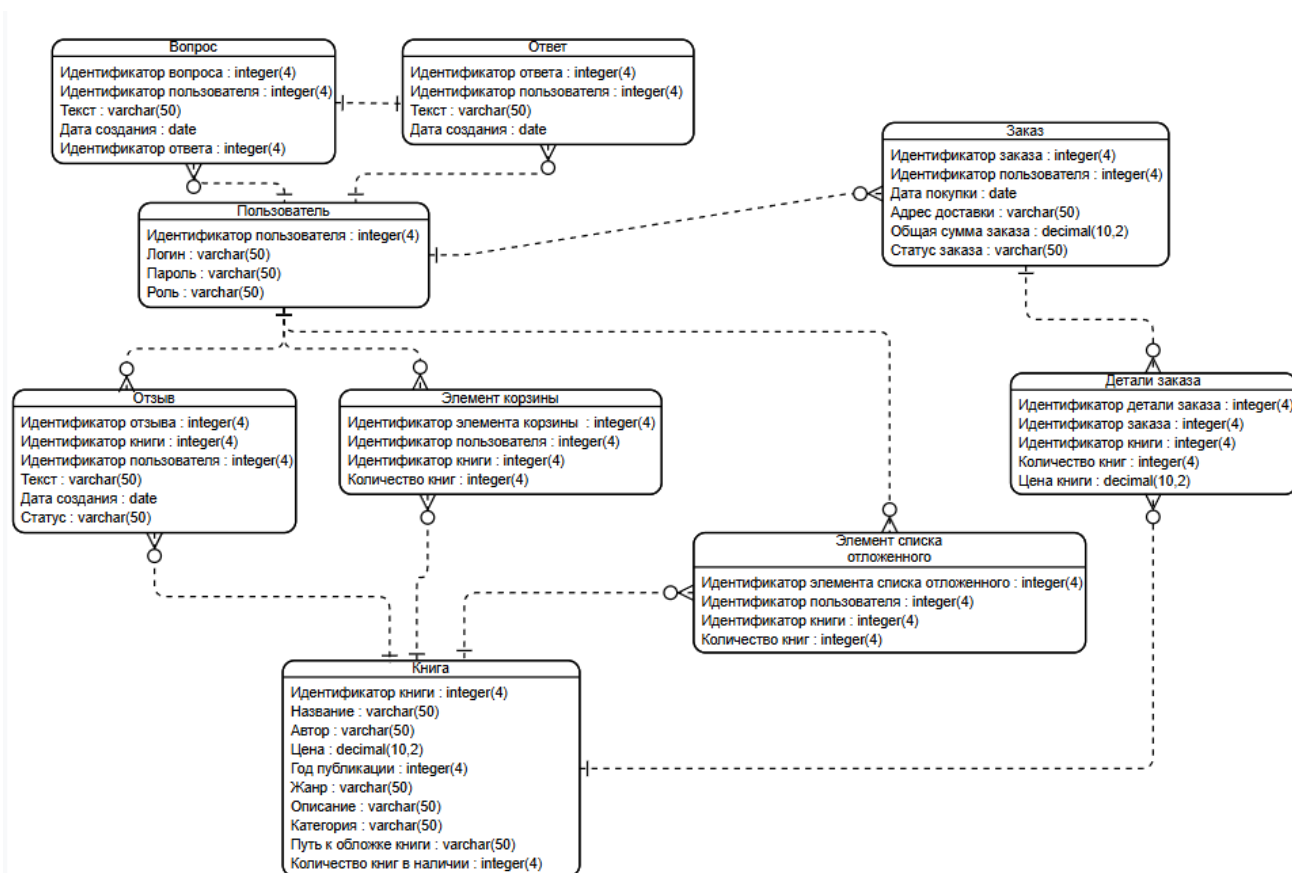


Рисунок 32 – Графическое представление логической модели базы данных

На данном представлении модели базы данных, сущности, связи и атрибуты представлены более подробно чем на концептуальной модели, однако на данный момент сущности представлены абстрактно, без привязки к конкретной реализации базы данных.

2.4 Проектирование серверной части приложения

Серверная часть разработана на основе архитектуры REST. REST – это архитектурный стиль взаимодействия клиента и сервера, основанный на HTTP-протоколе и использовании HTTP-методов, таких как GET – получить данные, POST – создать ресурс, PUT – обновить, DELETE – удалить и других. При этом обращение к ресурсам сервера происходит по их уникальным идентификаторам, называемым URI.

В приложении используется архитектура MVC. Ее схема представлена на рисунке 33.

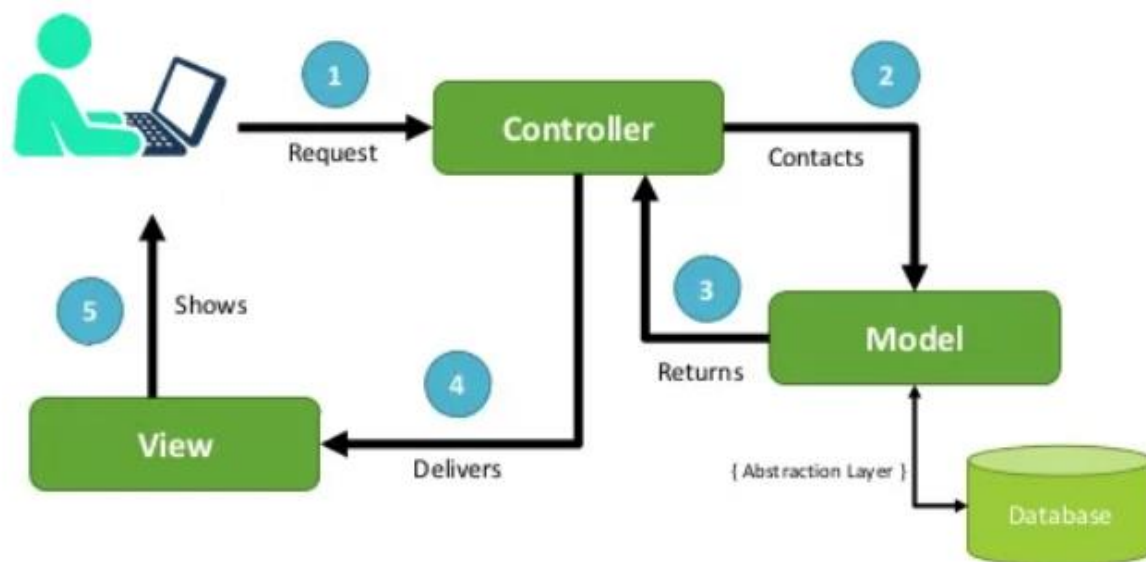


Рисунок 33 – Схема архитектуры MVC

Данная архитектура состоит из трех отдельных модулей:

- model (модель) – основная логика приложения, реагирует на команды из контроллера, выдает информацию и/или изменяет свое состояние;
- view (представление) – компонент визуализации информации, которую он получает от модели;
- controller (контроллер) – обеспечивает взаимодействие пользователя с системой, обрабатывает действия пользователя, проверяет полученную информацию и передает ее модели.

Главным преимуществами данной архитектуры являются разделение ответственности, простая поддержка, масштабируемость, гибкость и легкость тестирования [20].

2.5 Проектирование интерфейса пользователя

На основе выявленных функциональных требований был спроектирован интерфейс мобильного приложения. Он включает в себя страницы входа, регистрации, каталога, корзины, списка отложенного, страницы оплаты, страницы задания вопроса, страница просмотра заказов пользователя, страница просмотров вопросов, страница для ответов на вопросы, страницы для просмотра полной информации о книге, страница для написания отзыва, страница модерирования отзывов, профиля пользователя, страницы добавления новой книги, страницы редактирования книги, страницы для просмотра статистики, страница покупок.

Макет интерфейса страницы авторизации представлен на рисунке 34.



Макет интерфейса страницы авторизации представлен на рисунке 34. Он состоит из четырех элементов, пронумерованных с 1 по 4:

- 1 – поле для ввода логина;
- 2 – поле для ввода пароля;
- 3 – кнопка «Войти»;
- 4 – кнопка «Зарегистрироваться».

Рисунок 34 – Макет интерфейса страницы авторизации

На макете страницы авторизации имеются следующие элементы:

1 – поле, для ввода логина;

2 – поле, для ввода пароля;

3 – кнопка для осуществления входа;

4 – кнопка для перехода на страницу регистрации.

Макет интерфейса страницы регистрации представлен на рисунке 35.

The diagram shows a vertical rectangular container representing a registration form. Inside the container, there are five horizontal rectangular boxes stacked vertically. Each box contains text and is connected by a horizontal line to a number on the right. The boxes and their corresponding numbers are: 1. A box with the text 'Логин' (Login) connected to the number 1. 2. A box with the text 'Пароль' (Password) connected to the number 2. 3. A box with the text 'Подтверждение пароля' (Confirm password) connected to the number 3. 4. A box with the text 'Зарегистрироваться' (Register) connected to the number 4. 5. A box with the text 'Войти' (Login) connected to the number 5.

Рисунок 35 – Макет интерфейса страницы регистрации

На макете страницы регистрации имеются следующие элементы:

1 – поясняющий текст для пользователя;

2 – поле, для ввода логина;

3 – поле, для ввода пароля;

4 – поле, для подтверждения пароля;

5 – кнопка, для осуществления регистрации;

6 – поясняющий текст для пользователя;

7 – кнопка, для перехода на страницу авторизации.

Макет интерфейса каталога книг, для пользователя, представлен на рисунке 36.

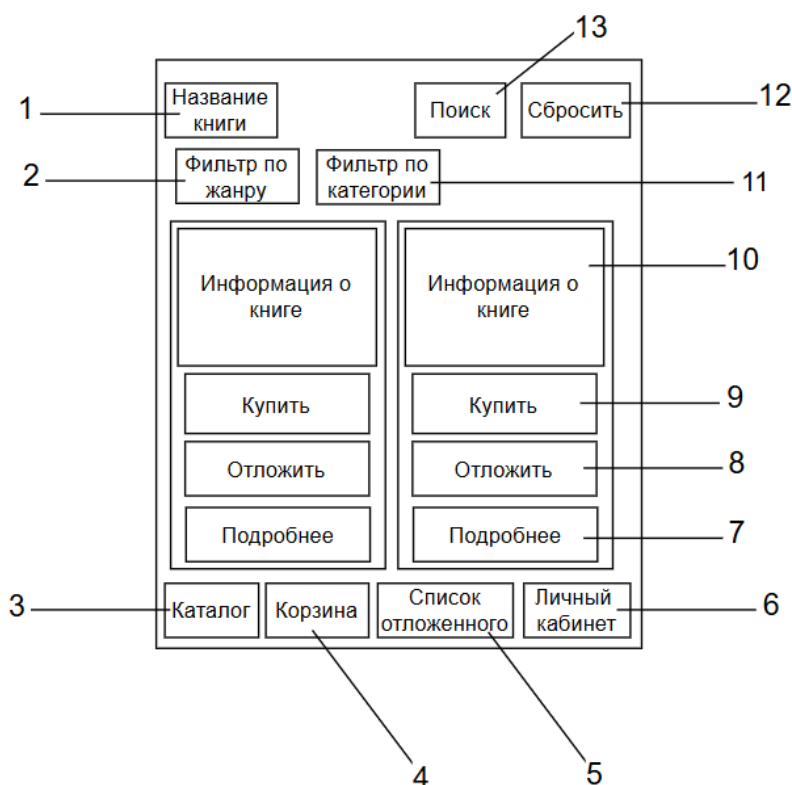


Рисунок 36 – Макет интерфейса страницы каталога книг, для пользователя

На макете страницы каталога имеются следующие элементы:

- 1 – поле, для ввода названия книги;
- 2 – поле, для выбора жанра;
- 3 – кнопка, для перехода к странице каталога;
- 4 – кнопка, для перехода к странице корзины;
- 5 – кнопка, для перехода к списку отложенного;
- 6 – кнопка, для перехода к странице личного кабинета;
- 7 – кнопка, для просмотра подробной информации о книге;
- 8 – кнопка, для перемещение указанной книги в список отложенного;
- 9 – кнопка, для перемещения указанной книги в корзину;
- 10 – поле, с информацией о книге;
- 11 – поле, для выбора категории;
- 12 – кнопка, для сброса фильтров;
- 13 – кнопка, для осуществления поиска.

Макет интерфейса страницы корзины, представлен на рисунке 37.

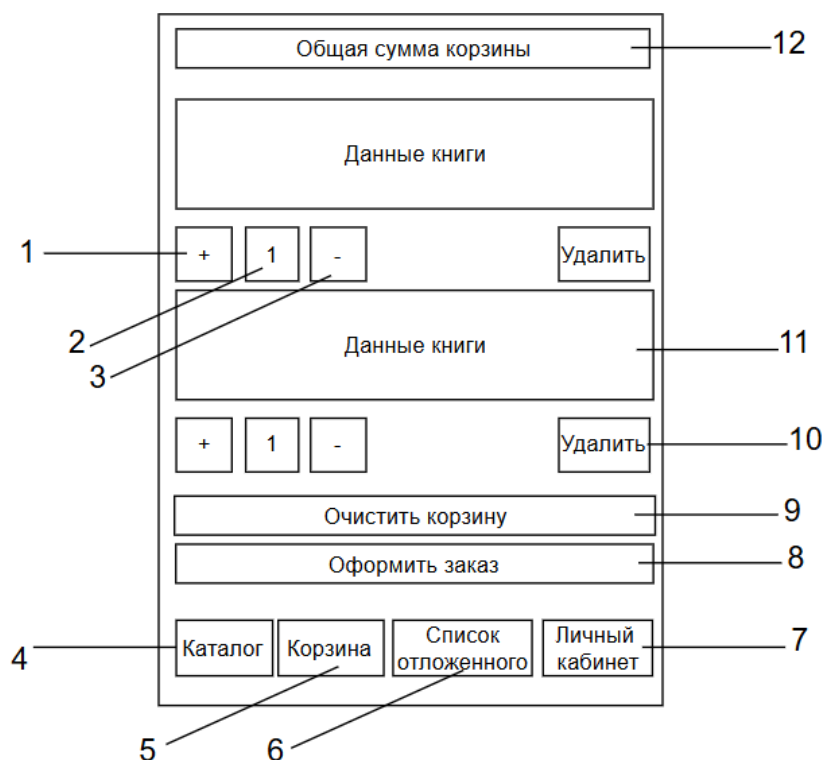


Рисунок 37 – Макет интерфейса страницы корзины

На макете страницы корзины имеются следующие элементы:

- 1 – кнопка, для увеличения количества книг;
- 2 – поле, показывающее количество книг для покупки;
- 3 – кнопка, для уменьшения количества книг;
- 4 – кнопка для перехода к странице каталога;
- 5 – кнопка, для перехода к странице корзины;
- 6 – кнопка, для перехода к списку отложенного;
- 7 – кнопка, для перехода к странице личного кабинета;
- 8 – кнопка, для оформления заказа;
- 9 – кнопка, для очистки корзины;
- 10 – кнопка, для удаления книги из корзины;
- 11 – поле, с информацией о книге;
- 12 – поле, с информацией об общей стоимости корзины.

Макет интерфейса страницы списка отложенного, представлен на рисунке 38.

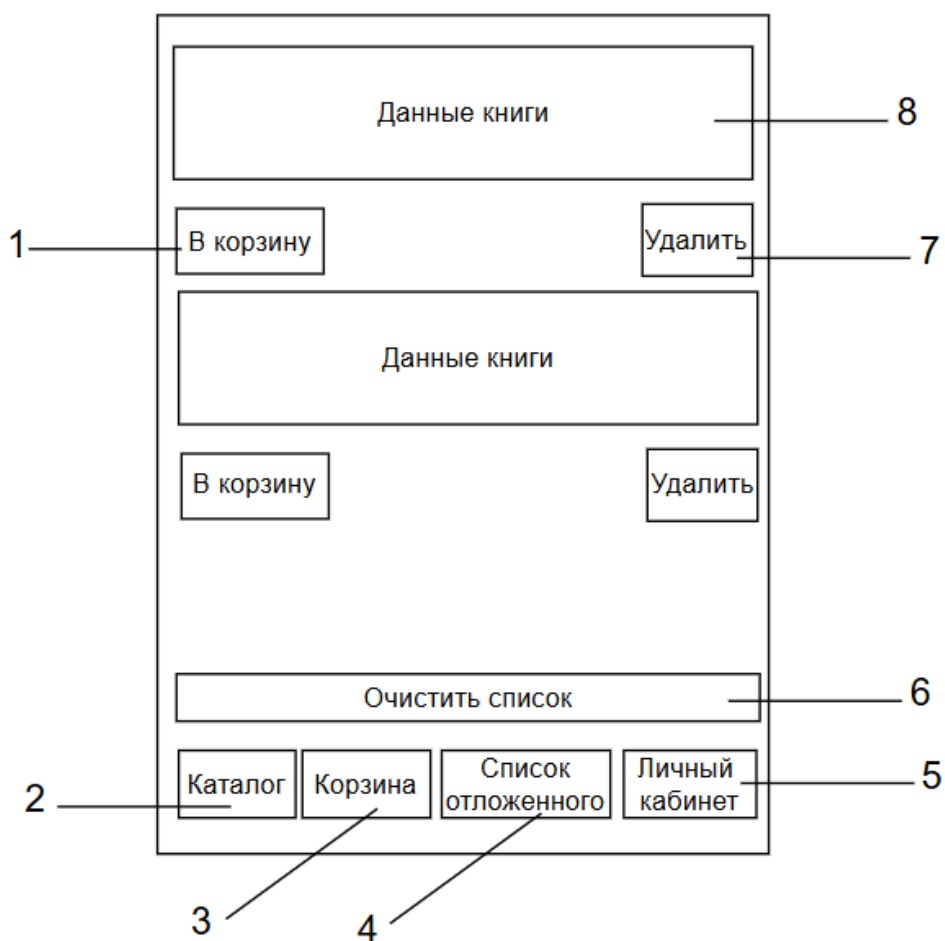


Рисунок 38 – Макет интерфейса страницы списка отложенного

На макете страницы списка отложенного имеются следующие элементы:

- 1 – кнопка, для перемещения книги в корзину;
- 2 – кнопка для перехода к странице каталога;
- 3 – кнопка, для перехода к странице корзины;
- 4 – кнопка, для перехода к списку отложенного;
- 5 – кнопка, для перехода к странице личного кабинета;
- 6 – кнопка, для очистки списка отложенного;
- 7 – кнопка, для удаления книги из списка отложенного;
- 8 – поле, с информацией о книге.

Макет интерфейса страницы личного кабинета покупателя, представлен на рисунке 39.

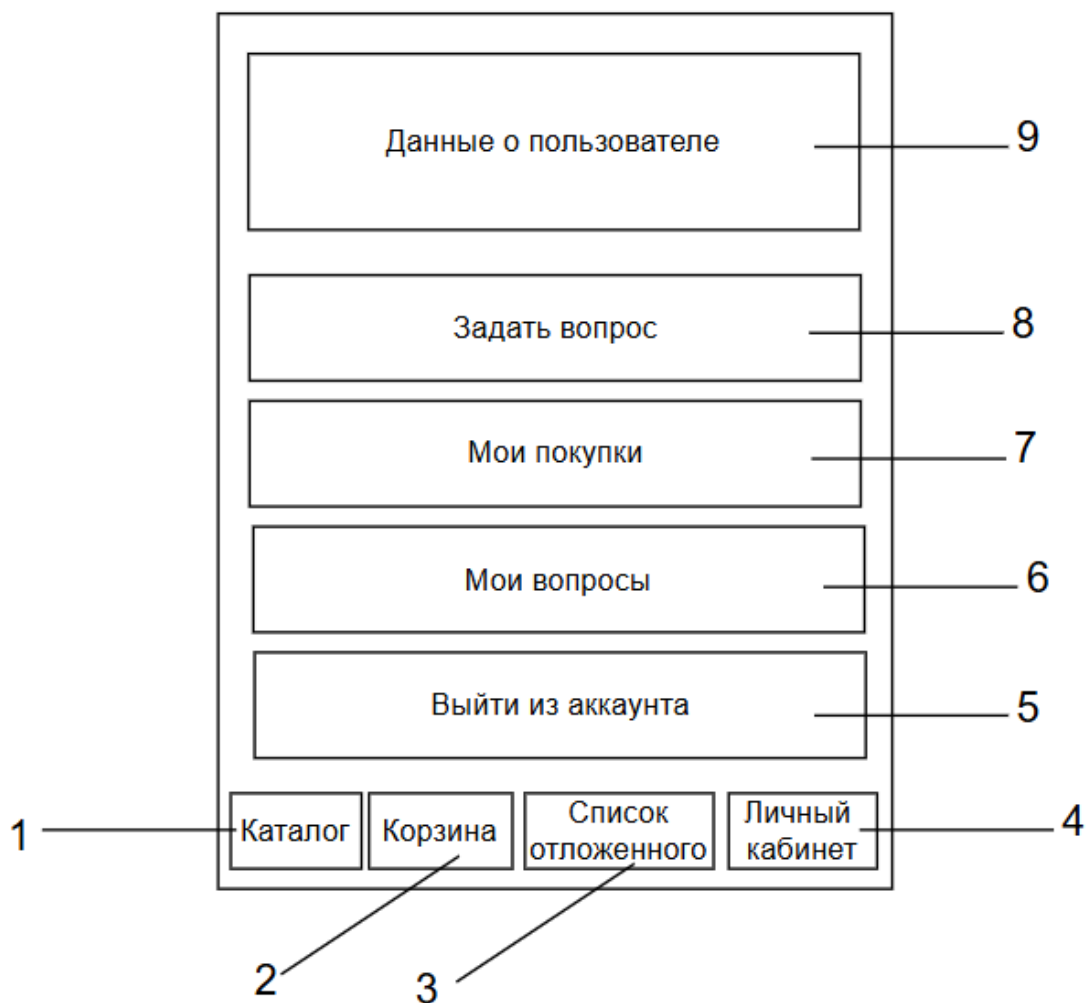


Рисунок 39 – Макет интерфейса страницы личного кабинета покупателя

На макете страницы личного кабинета покупателя имеются следующие элементы:

- 1 – кнопка для перехода к странице каталога;
- 2 – кнопка, для перехода к странице корзины;
- 3 – кнопка, для перехода к списку отложенного;
- 4 – кнопка, для перехода к странице личного кабинета;
- 5 – кнопка, для выхода из аккаунта;
- 6 – кнопка, для просмотра вопросов пользователя;
- 7 – кнопка, для просмотра покупок пользователя;
- 8 – кнопка, для задания вопроса;
- 9 – поле, с информацией о пользователе.

Макет интерфейса страницы оплата заказа, представлен на рисунке 40.

The wireframe shows a payment form layout. It consists of a central container with several input fields and two buttons. The fields are arranged vertically: 'Адрес доставки' (Delivery address), 'Номер карты' (Card number), 'MM/YY' (Month/Year), and 'CVV'. The buttons are 'Отмена' (Cancel) and 'Подтвердить' (Confirm). Numbered callouts point to specific elements: 1 points to the 'MM/YY' field, 2 points to the 'Отмена' button, 3 points to the 'Адрес доставки' field, 4 points to the 'Номер карты' field, 5 points to the 'CVV' field, and 6 points to the 'Подтвердить' button.

Рисунок 40 – Макет интерфейса страницы оплаты заказа

На макете страницы оплаты заказа имеются следующие элементы:

1 – поле, для ввода месяца и года, до которого действительна банковская карта;

2 – кнопка, для отмены оформления заказа;

3 – поле, для ввода адреса доставки;

4 – поле, для ввода номера банковской карты;

5 – поле, для ввода защитного кода для совершения операций с картой;

6 – кнопка, для подтверждения оформления заказа.

Макет интерфейса страницы для задания вопроса, представлен на рисунке 41.

The wireframe shows a question submission form layout. It consists of a central container with three elements: a text input field labeled 'Поле для вопроса' (Question field), a button labeled 'Отправить' (Send), and a button labeled 'Назад' (Back). Numbered callouts point to these elements: 1 points to the 'Поле для вопроса' field, 2 points to the 'Отправить' button, and 3 points to the 'Назад' button.

Рисунок 41 – Макет интерфейса страницы для задания вопроса

На макете страницы для задания вопроса имеются следующие элементы:

- 1 – поле, для ввода текста вопроса;
- 2 – кнопка, для отправки вопроса;
- 3 – кнопка, для возврата к странице личного кабинета.

Макет интерфейса страницы для просмотра заказов, представлен на рисунке 42.

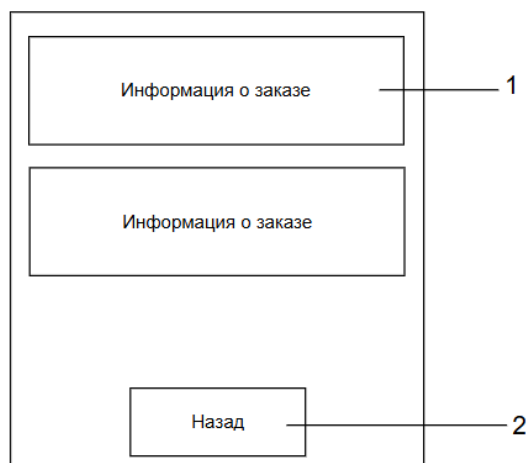


Рисунок 42 – Макет интерфейса страницы для просмотра заказов

На макете страницы для просмотра заказов имеются следующие элементы:

- 1 – поле с информацией о совершенном заказе;
- 2 – кнопка, для возврата к личному кабинету.

Макет интерфейса страницы для просмотра вопросов, представлен на рисунке 43.

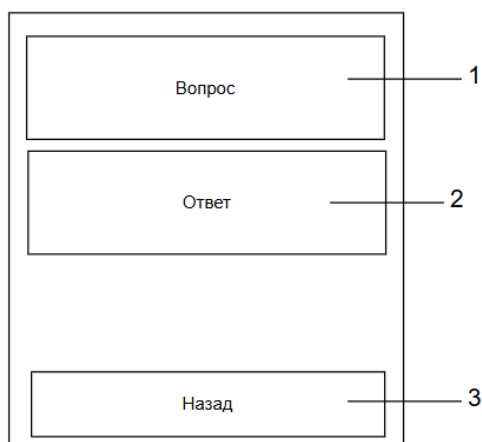


Рисунок 43 – Макет интерфейса страницы для просмотра вопросов

На макете страницы просмотра вопросов имеются следующие элементы:

- 1 – поле с информацией о вопросе;
- 2 – поле с информацией об ответе на вопрос;
- 3 – кнопка, для возврата к странице личного кабинета.

Макет интерфейса страницы личного кабинета менеджера, представлен на рисунке 44.

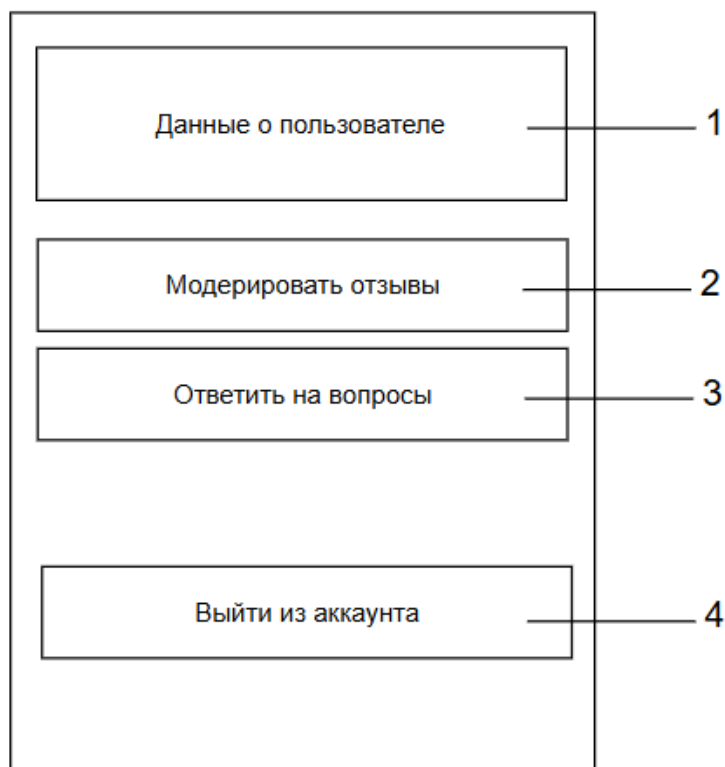


Рисунок 44 – Макет интерфейса страницы для личного кабинета менеджера

На макете страницы личного кабинета менеджера имеются следующие элементы:

- 1 – поле с информацией о текущем пользователе;
- 2 – кнопка, для перехода к странице модерирования отзывов;
- 3 – кнопка, для перехода к странице для ответов на вопросы;
- 4 – кнопка, для выхода из личного кабинета.

Макет интерфейса страницы ответа на вопросы, представлен на рисунке 45.

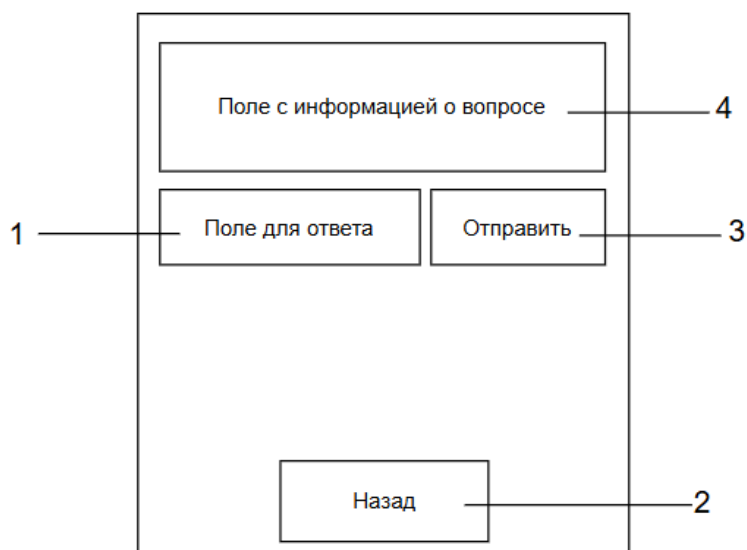


Рисунок 45 – Макет интерфейса страницы для ответов на вопросы

На макете страницы для ответов на вопросы имеются следующие элементы:

- 1 – поле, для ввода текста ответа на вопрос;
- 2 – кнопка, для перехода к странице личного кабинета;
- 3 – кнопка, для отправки ответа на вопрос;
- 4 – поле с информацией о вопросе.

Макет интерфейса страницы с деталями о книге, представлен на рисунке 46.



Рисунок 46 – Макет интерфейса страницы с информацией о книге

На макете страницы с информацией о книге имеются следующие элементы:

- 1 – поле с информацией о книге;
- 2 – кнопка, для перехода к странице для написания отзыва;
- 3 – кнопка, для просмотра всех отзывов о книге;
- 4 – кнопка, для перехода к станице каталога книг.

Макет интерфейса страницы для отправки отзыва, представлен на рисунке 47.

Рисунок 47 – Макет интерфейса страницы для написания отзыва о книге

На макете страницы для написания отзыва о книге имеются следующие элементы:

- 1 – поле, для ввода текста отзыва;
- 2 – кнопка, для отправки отзыва;
- 3 – кнопка, для перехода к странице с информацией о книге.

Макет интерфейса страницы для модерирования отзывов, представлен на рисунке 48.

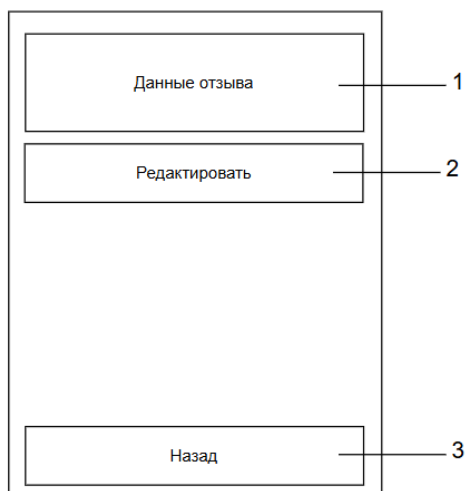


Рисунок 48 – Макет интерфейса страницы для модерирования отзывов

На макете страницы для модерирования отзывов имеются следующие элементы:

- 1 – поле с данными отзыва;
- 2 – кнопка, для модерирования отзыва;
- 3 – кнопка, для перехода к странице личного кабинета менеджера.

Макет интерфейса страницы каталога книг для администратора, представлен на рисунке 49.

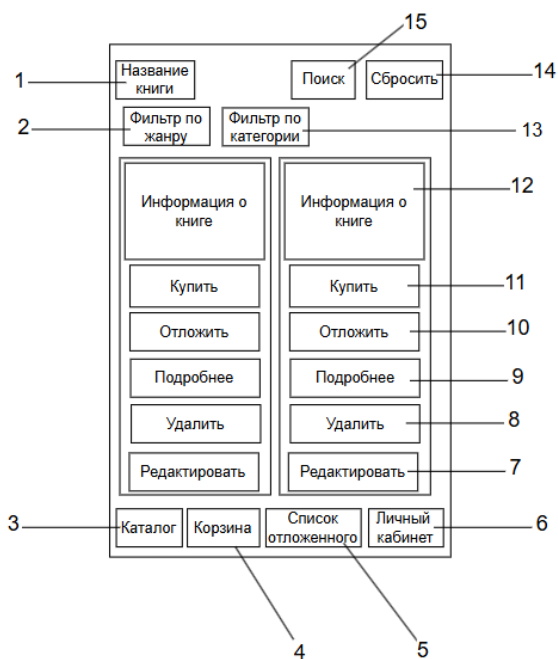


Рисунок 49 – Макет интерфейса страницы каталога для администратора

На макете страницы каталога книг для администратора имеются следующие элементы:

- 1 – поле, для ввода названия книги;
- 2 – поле, для выбора жанра;
- 3 – кнопка, для перехода к странице каталога;
- 4 – кнопка, для перехода к странице корзины;
- 5 – кнопка, для перехода к списку отложенного;
- 6 – кнопка, для перехода к странице личного кабинета;
- 7 – кнопка, для редактирования информации о книге;
- 8 – кнопка, для удаления книги из каталога;
- 9 – кнопка, для просмотра подробной информации о книге;
- 10 – кнопка, для добавления книги в список отложенного;
- 11 – кнопка, для добавления книги в корзину;
- 12 – поле с базовой информацией о книге;
- 13 – кнопка, для выбора категории для фильтрации;
- 14 – кнопка, для сброса параметров фильтрации;
- 15 – кнопка, для поиска книги по указанному названию.

Макет интерфейса страницы добавления книги, представлен на рисунке 50.

The diagram shows a vertical form for adding a book. It consists of the following elements from top to bottom:

- Input field: "Поле для ввода названия" (1)
- Input field: "Поле для ввода автора" (2)
- Input field: "Поле для ввода жанра" (3)
- Input field: "Поле для ввода категории" (4)
- Input field: "Поле для ввода года издания" (5)
- Input field: "Поле для ввода описания" (6)
- Input field: "Поле для ввода цены" (7)
- Input field: "Поле для ввода количества книг в наличии" (8)
- Button: "Загрузить изображение" (9)
- Buttons: "Сохранить" (10) and "Отмена" (11)

Рисунок 50 – Макет интерфейса страницы добавления книги

На макете страницы добавления книги имеются следующие элементы:

- 1 – поле, для ввода названия книги;
- 2 – поле, для ввода автора книги;
- 3 – поле, для ввода жанра книги;
- 4 – поле, для ввода категории книги;
- 5 – поле, для ввода года издания книги;
- 6 – поле, для ввода описания книги;
- 7 – поле, для ввода цены книги;
- 8 – поле, для ввода количества книг в наличии;
- 9 – кнопка, для загрузки изображения книги;
- 10 – кнопка, для перехода к каталогу книг;
- 11 – кнопка, для добавления книги в каталог.

Макет интерфейса страницы редактирования книги, представлен на рисунке 51.

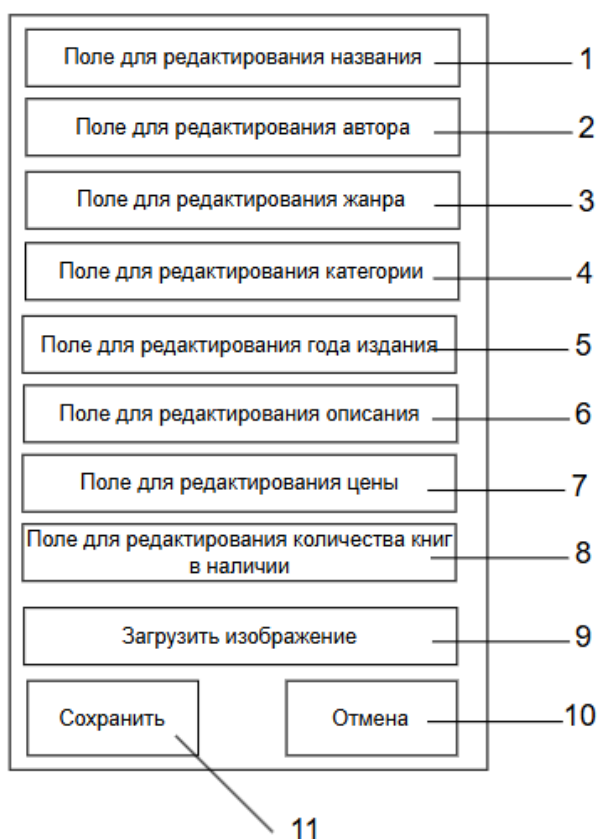


Рисунок 51 – Макет интерфейса страницы редактирования книги

На макете страницы редактирования книги имеются следующие элементы:

- 1 – поле, для редактирования названия книги;
- 2 – поле, для редактирования автора книги;
- 3 – поле, для редактирования жанра книги;
- 4 – поле, для редактирования категории книги;
- 5 – поле, для редактирования года издания книги;
- 6 – поле, для редактирования описания книги;
- 7 – поле, для редактирования цены книги;
- 8 – поле, для редактирования количества книг в наличии;
- 9 – кнопка, для загрузки изображения книги;
- 10 – кнопка, для перехода к каталогу книг;
- 11 – кнопка, для сохранения новых данных книги в каталог.

Макет интерфейса страницы статистики продаж книг, представлен на рисунке 52.

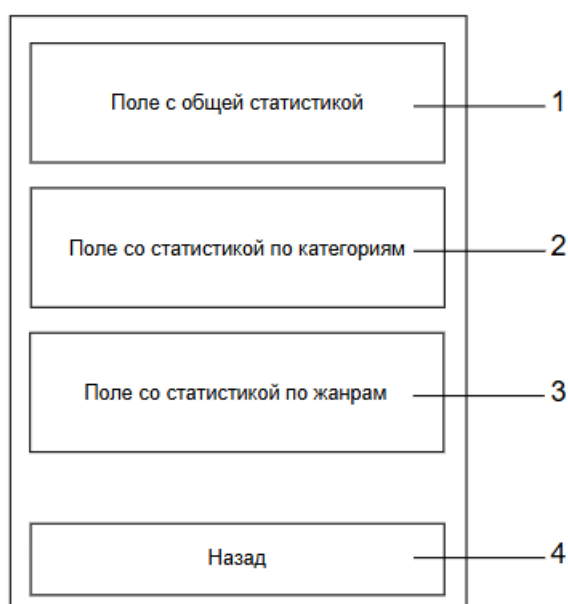


Рисунок 52 – Макет интерфейса страницы статистики продаж книг

На макете страницы статистики продаж книг имеются следующие элементы:

- 1 – поле с общей статистикой продаж;
- 2 – поле со статистикой продаж по категориям;

3 – поле со статистикой продаж по жанрам;

4 – кнопка, для перехода к странице личного кабинета администратора.

Макет интерфейса страницы личного кабинета администратора, представлен на рисунке 53.

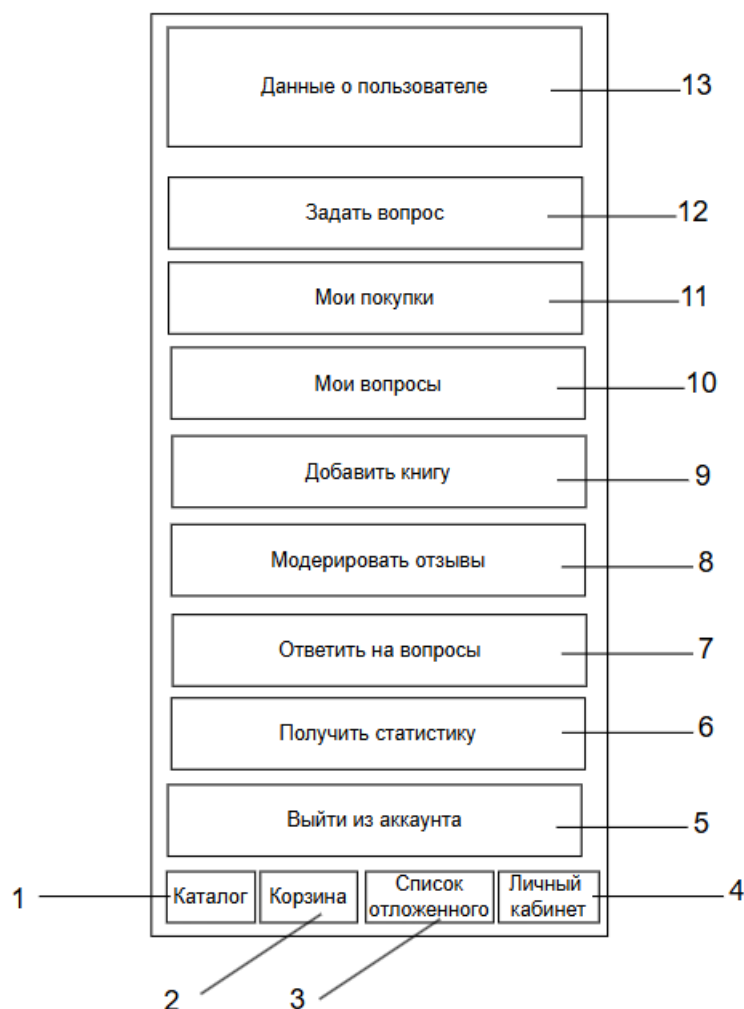


Рисунок 53 – Макет интерфейса страницы личного кабинета администратора

На макете страницы личного кабинета администратора имеются следующие элементы:

1 – кнопка, для перехода к каталогу книг;

2 – кнопка, для перехода к странице корзины;

3 – кнопка, для перехода с странице списка отложенного;

4 – кнопка, для перехода к странице личного кабинета;

5 – кнопка, для выхода из аккаунта;

- 6 – кнопка, для перехода к странице статистики продаж;
- 7 – кнопка, для перехода к странице для ответов на вопросы;
- 8 – кнопка, для перехода к странице модерации отзывов;
- 9 – кнопка, для перехода к странице добавления книги;
- 10 – кнопка, для перехода к странице просмотра вопросов;
- 11 – кнопка, для перехода к странице просмотра покупок;
- 12 – кнопка, для перехода к странице для задания вопроса;
- 13 – поле с информацией о текущем пользователе.

3 Реализация программного обеспечения

3.1 Реализация базы данных

На основании концептуальной и логической модели для СУБД Microsoft SQL Server 2019 была разработана физическая модель базы данных (рисунок 54).

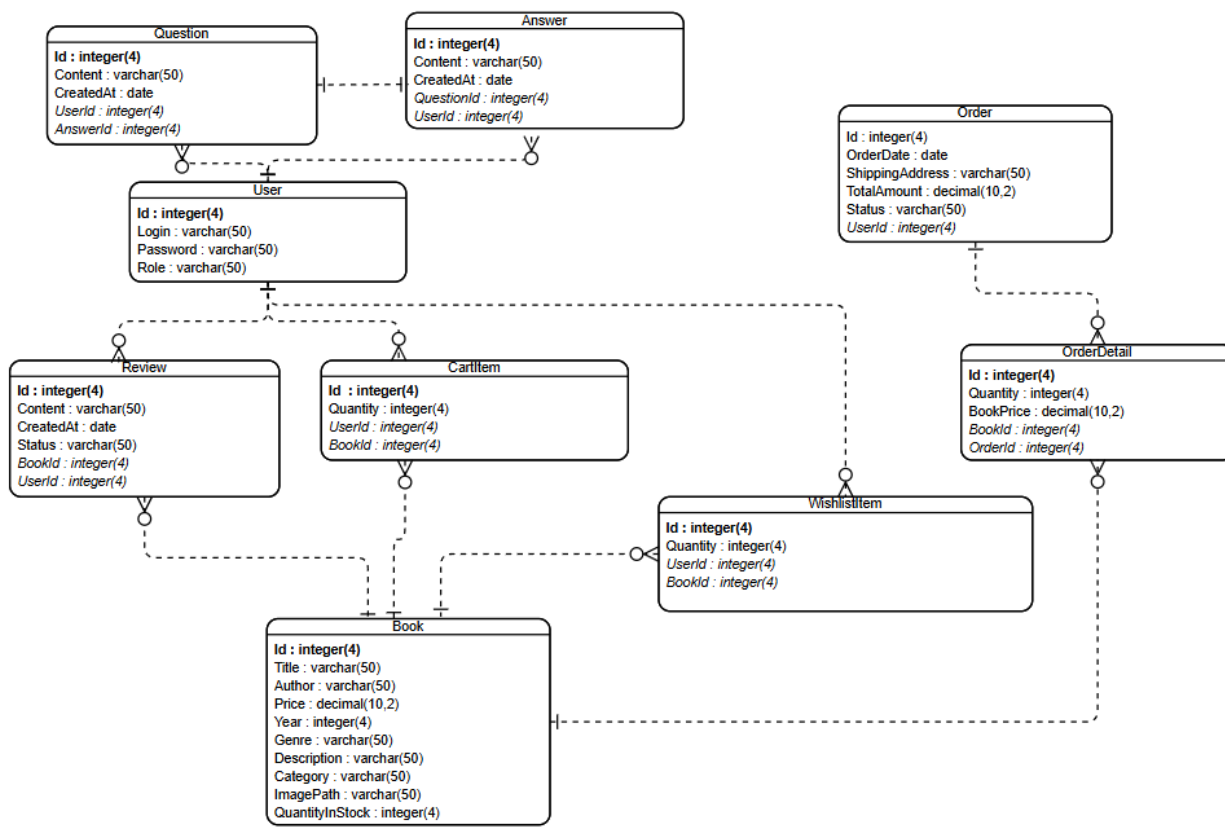


Рисунок 54 – Графическое представление физической модели базы данных

Описание атрибутов сущности «Книга» представлено в таблице 10.

Таблица 10 – Описание атрибутов сущности «Книга»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
Id	Идентификатор	integer(4)	Нет	Да	PK
Title	Название	varchar(50)	Нет	Нет	-
Author	Автор	varchar(50)	Нет	Нет	-
Price	Цена	decimal(10,2)	Нет	Нет	-
Year	Год издания	integer(4)	Нет	Нет	-
Genre	Жанр	varchar(50)	Нет	Нет	-
Description	Описание	varchar(50)	Нет	Нет	-
Category	Категория	varchar(50)	Нет	Нет	-

Продолжение таблицы 10

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
ImagePath	Путь к изображению книги	varchar(50)	Нет	Нет	-
QuantityStock	Количество книг в наличии	integer(4)	Нет	Нет	-

Описание атрибутов сущности «Отзыв» представлено в таблице 11.

Таблица 11 – Описание атрибутов сущности «Отзыв»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
ImagePath	Путь к изображению книги	varchar(50)	Нет	Нет	-
QuantityStock	Количество книг в наличии	integer(4)	Нет	Нет	-
Id	Идентификатор	integer(4)	Нет	Да	PK
Content	Текст	varchar(50)	Нет	Нет	-
CreatedAt	Дата создания	date	Нет	Нет	-
Status	Статус	varchar(50)	Нет	Нет	-
BookId	Идентификатор книги	integer(4)	Нет	Да	FK
UserId	Идентификатор пользователя	integer(4)	Нет	Да	FK

Описание атрибутов сущности «Элемент корзины» представлено в таблице 12.

Таблица 12 – Описание атрибутов сущности «Элемент корзины»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
Id	Идентификатор	integer(4)	Нет	Да	PK
Quantity	Количество книг	integer(4)	Нет	Нет	-
UserId	Идентификатор пользователя	integer(4)	Нет	Да	FK
BookId	Идентификатор книги	integer(4)	Нет	Да	FK

Описание атрибутов сущности «Элемент списка отложенного» представлено в таблице 13.

Таблица 13 – Описание атрибутов сущности «Элемент списка отложенного»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
Id	Идентификатор	integer(4)	Нет	Да	PK
Quantity	Количество книг	integer(4)	Нет	Нет	-
UserId	Идентификатор пользователя	integer(4)	Нет	Да	FK
BookId	Идентификатор книги	integer(4)	Нет	Да	FK

Описание атрибутов сущности «Детали заказа» представлено в таблице 14.

Таблица 14 – Описание атрибутов сущности «Детали заказа»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
Id	Идентификатор	integer(4)	Нет	Да	PK
Quantity	Количество книг	integer(4)	Нет	Нет	-
BookPrice	Цена книги	decimal(10,2)	Нет	Нет	-
BookId	Идентификатор книги	integer(4)	Нет	Да	FK
OrderId	Идентификатор отрыва	integer(4)	Нет	Да	FK

Описание атрибутов сущности «Пользователи» представлено в таблице 15.

Таблица 15 – Описание атрибутов сущности «Пользователь»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
Id	Идентификатор	integer(4)	Нет	Да	PK
Login	Логин	varchar(50)	Нет	Да	-
Password	Пароль	varchar(50)	Нет	Нет	-
Role	Роль	varchar(50)	Нет	Нет	-

Описание атрибутов сущности «Вопрос» представлено в таблице 16.

Таблица 16 – Описание атрибутов сущности «Вопрос»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
Id	Идентификатор	integer(4)	Нет	Да	PK
Content	Текст	varchar(50)	Нет	Нет	-
CreatedAt	Дата создания	date	Нет	Нет	-
UserId	Идентификатор пользователя	integer(4)	Нет	Да	FK

Продолжение таблицы 16

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
AnswerId	Идентификатор ответа	integer(4)	Нет	Да	FK

Описание атрибутов сущности «Ответ» представлено в таблице 17.

Таблица 17 – Описание атрибутов сущности «Ответ»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
Id	Идентификатор	integer(4)	Нет	Да	PK
Content	Текст	varchar(50)	Нет	Нет	-
CreatedAt	Дата создания	date	Нет	Нет	-
UserId	Идентификатор пользователя	integer(4)	Нет	Да	FK
QuestionId	Идентификатор вопроса	integer(4)	Нет	Да	FK

Описание атрибутов сущности «Заказ» представлено в таблице 18.

Таблица 18 – Описание атрибутов сущности «Заказ»

Наименование	Описание	Тип данных	Возможность значения NULL	Unique	Роль
Id	Идентификатор	integer(4)	Нет	Да	PK
OrderDate	Дата покупки	date	Нет	Нет	-
ShoppingAddress	Адрес доставки	varchar(50)	Нет	Нет	-
TotalAmount	Общая сумма заказа	decimal(10,2)	Нет	Нет	-
Status	Статус	varchar(50)	Нет	Нет	-
UserId	Идентификатор пользователя	integer(4)	Нет	Да	FK

На данной физической модели структура базы данных отображена с учетом спецификации типов СУБД Microsoft SQL Server 2019.

3.2 Разработка серверной части приложения

Разработка серверной части приложения производилась в среде программирования Microsoft Visual Studio 2022 на языке C# с применением технологии Web API [26]. Для взаимодействия с базой данных использовалась

технология Entity Framework [17]. Диаграмма классов серверной части представлена на рисунке 55.

В соответствии с архитектурным шаблоном MVC все классы серверной части можно разделить на две части: модели, предназначенные для описания данных предметной области, и контроллеры, для обработки HTTP-запросов к серверу.

Диаграмма классов моделей представлена на рисунке 56. Эти классы были написаны при разработке серверной части приложения и на основе их была автоматически сгенерирована база данных (подход Code First) [27].

«Code first» (Сначала код) – это подход к разработке программного обеспечения, в котором сначала создаётся код реализации системы, в частности описываются модели предметной области, а затем на его основе автоматически генерируется соответствующая схема базы данных. Преимущества данного подхода заключается в его простоте и скорости.

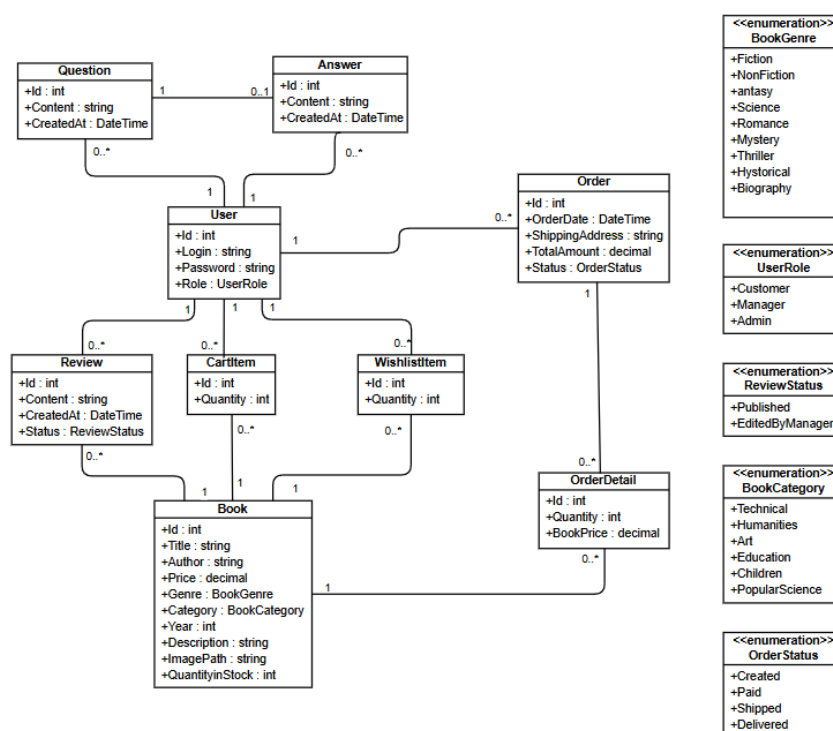


Рисунок 55 – Диаграмма классов серверной части (модели)

Диаграмма классов слоя контроллеров представлена на рисунке 56.

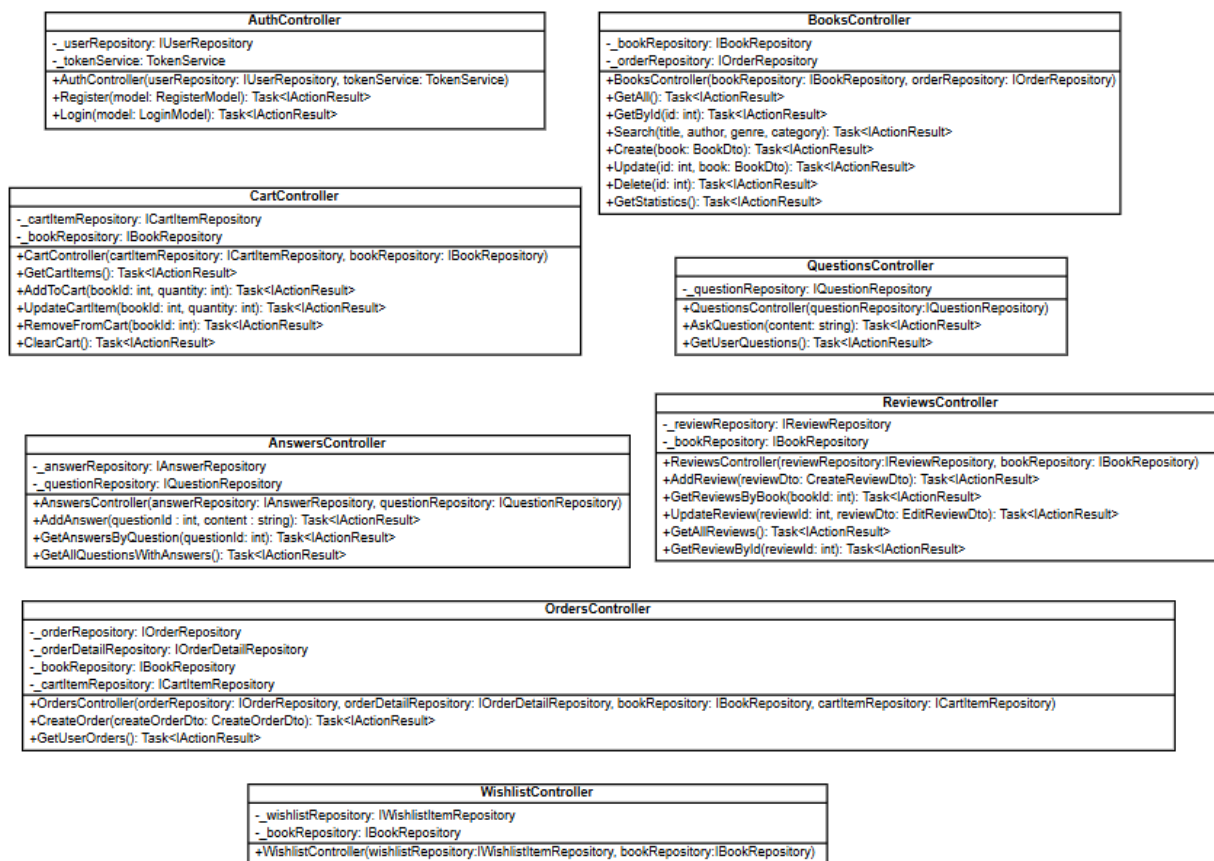


Рисунок 56 – Диаграмма классов серверной части (контроллеры)

Более подробное описание классов слоя контроллеров и их назначений представлено в таблице 19.

Таблица 19 – Описание классов контроллеров

Класс	Назначение
AuthController	Отвечает за обработку запросов по регистрации и аутентификации пользователей.
AnswersController	Отвечает за обработку запросов на добавление и получение ответов на вопросы.
BooksController	Отвечает за обработку запросов на добавление, получение, редактирование, удаление книг из каталога, а также получение статистики.
CartController	Отвечает за обработку запросов на получение, добавление, удаление и обновления книг в корзине.
OrdersController	Отвечает за обработку запросов на создание и получения заказов пользователей.
QuestionsController	Отвечает за обработку запросов на добавления и получения вопросов.
ReviewsController	Отвечает за обработку запросов на добавление, модерирования и получения отзывов о книгах.

Продолжение таблицы 19

Класс	Назначение
WishlistController	Отвечает за обработку запросов на получение, добавление, удаление и книг в списке отложенного.

Листинг серверной части представлен в приложении В.

3.3 Реализация клиентской части приложения

Разработка мобильного приложения производилась на языке C# с использованием среды программирования Visual Studio 2022. Диаграмма компонентов, отражающая структуру программы, представлена на рисунке 57.

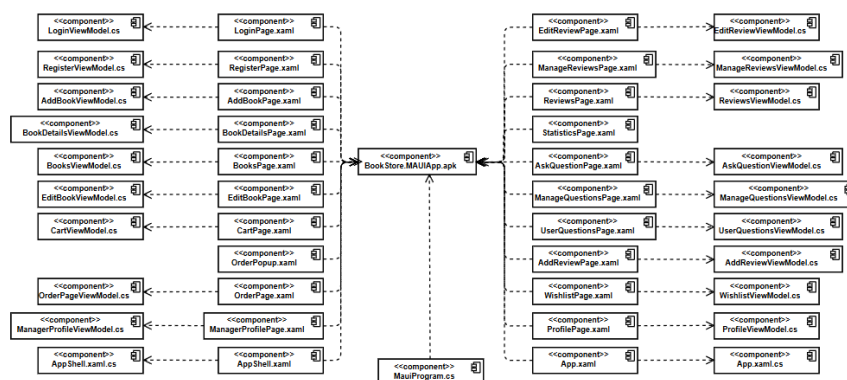


Рисунок 57 – Диаграмма компонентов мобильного приложения

Описание компонентов представлено в таблице 20.

Таблица 20 – Компоненты клиентского приложения

Компонент	Назначение
App.xaml	XAML-файл описания подключаемых ресурсов к приложению
App.xaml.cs	Является начальной точкой загрузки, закрытия приложения
AppShell.xaml	XAML-файл описания структуры интерфейса приложения
AppShell.xaml.cs	Регистрирует страницы приложения, не вошедшие в основную структуру
MauiProgram.cs	Отвечает за инициализацию и конфигурацию приложения.
LoginPage.xaml	XAML-файл описания интерфейса страницы авторизации
LoginViewModel.cs	Управляет логикой и данными страницы авторизации
RegisterPage.xaml	XAML-файл описания интерфейса страницы регистрации

Продолжение таблицы 20

Компонент	Назначение
RegisterViewModel.cs	Управляет логикой и данными страницы регистрации
AddBookPage.xaml	XAML-файл описания интерфейса страницы добавления книги
AddBookViewModel.cs	Управляет логикой и данными страницы добавления книги
BookDetailPage.xaml	XAML-файл описания интерфейса страницы просмотра подробной информации о книге
BookDetailViewModel.cs	Управляет логикой и данными страницы просмотра подробной информации о книге
BooksPage.xaml	XAML-файл описания интерфейса страницы каталога книг
BooksViewModel.cs	Управляет логикой и данными страницы каталога книг
EdirBookPage.xaml	XAML-файл описания интерфейса страницы редактирования информации о книге
EdirBookViewModel.cs	Управляет логикой и данными страницы редактирования информации о книге
CartPage.xaml	XAML-файл описания интерфейса страницы корзины
CartViewModel.cs	Управляет логикой и данными страницы корзины
OrderPage.xaml	XAML-файл описания интерфейса страницы заказа
OrderViewModel.cs	Управляет логикой и данными страницы заказа
EditReviewPage.xaml	XAML-файл описания интерфейса страницы модерирования отзыва
EditReviewViewModel.cs	Управляет логикой и данными страницы модерирования отзыва
ManageReviewsPage.xaml	XAML-файл описания интерфейса страницы просмотра отзывов для менеджера
ManageReviewsPage.xaml	XAML-файл описания интерфейса страницы просмотра отзывов для менеджера
ManageReviewsViewModel.cs	Управляет логикой и данными страницы просмотра отзывов для менеджера
ReviewsPage.xaml	XAML-файл описания интерфейса страницы просмотра отзывов книги
ReviewsViewModel.cs	Управляет логикой и данными страницы просмотра отзывов книги
AskQuestionPage.xaml	XAML-файл описания интерфейса страницы задания вопроса
AskQuestionViewModel.cs	Управляет логикой и данными страницы задания вопроса
ManageQuestionsPage.xaml	XAML-файл описания интерфейса страницы ответа на вопросы

Окончание таблицы 20

Компонент	Назначение
ManageQuestionsViewModel.cs	Управляет логикой и данными страницы ответа на вопросы
UserQuestionsPage.xaml	XAML-файл описания интерфейса страницы просмотра вопросов
UserQuestionsViewModel.cs	Управляет логикой и данными страницы просмотра вопросов
ProfilePage.xaml	XAML-файл описания интерфейса страницы профиля
ProfileViewModel.cs	Управляет логикой и данными страницы профиля
OrderPopur.xaml	XAML-файл описания интерфейса страницы оплаты заказа
StatisticPage.xaml	XAML-файл описания интерфейса страницы просмотра статистики

Диаграмма развертывания, на которой описывается физическое расположение компонентов системы, представлена на рисунке 58.

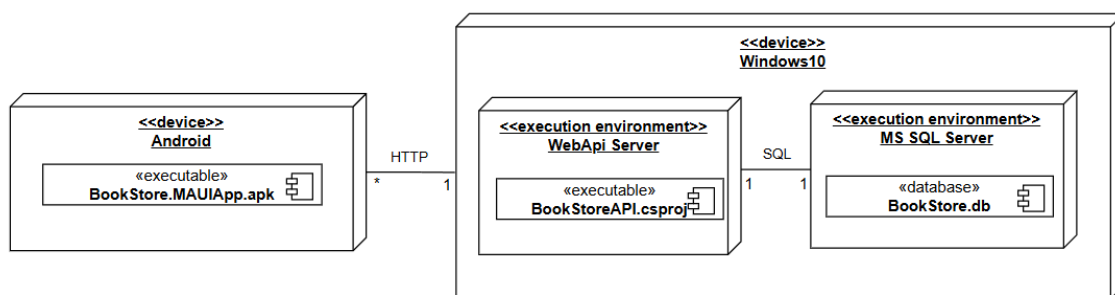


Рисунок 58 – Диаграмма развертывания мобильного приложения

Как можно увидеть на этой диаграмме, клиентское приложение работает на мобильном устройстве под управлением системы Android. API и база данных располагаются на сервере, работающем под управлением ОС Windows. Взаимодействие между серверной и клиентской частями приложения происходит с использованием протокола HTTP. Взаимодействие между сервером и БД происходит при помощи SQL-запросов.

Листинг клиентской части программы представлен в приложении Г.

Все тест-кейсы были пройдены успешно, что говорит о корректной работе программы в рамках проверки поля названия книги.

4.2 Модульное тестирование

Модульное тестирование – это тестирование при котором тестируются отдельные модули или компоненты программы. Тестирование проводится изолировано, автоматизировано с использованием заглушек. Цель модульного тестирования заключается в проверке работоспособности отдельного модуля на корректность работы [29].

В данном проекте было принято решение написать модульные тесты для серверной части приложения. Рассмотрим применение данного вида тестирования на примере класса BooksController, который отвечает за добавление, удаление, редактирование и просмотр книг.

Для проведения модульных тестов был разработан тестовый класс BooksControllerTests, код которого представлен в приложении Д.

Результат проведения модульных тестов представлен на рисунке 59.

Тестирование	Длительность	Признаки	Сообщение об ошибке
✔ BookStore.Tests (10)	671 мс		
✔ <Пустое пространство имен> (10)	671 мс		
✔ BooksControllerTests (10)	671 мс		
✔ Create_ValidBook_ReturnsCreat...	5 мс		
✔ Delete_BookExists_ReturnsNoC...	< 1 мс		
✔ Delete_BookNotFound_Returns...	649 мс		
✔ GetAll_ReturnsOkResult_WithLis...	2 мс		
✔ GetById_BookExists_ReturnsOk	3 мс		
✔ GetById_BookNotFound_Return...	< 1 мс		
✔ GetStatistics_ReturnsOk_WithSt...	6 мс		
✔ Search_NoBooksFound_Returns...	1 мс		
✔ Update_BookExists_ReturnsNoC...	5 мс		
✔ Update_BookNotFound_Return...	< 1 мс		

Рисунок 59 – Результаты модульного тестирования

Все тесты были успешно пройдены, что свидетельствует о корректной работе тестируемых методов.

4.3 Метрики кода

Метриками кода называются некоторые количественные величины, позволяющие определить характеристики программного обеспечения и определить сложность, качество и читаемость кода.

Рассмотрим метод «CreateOrder», отвечающий за получение всех книг в системе. Его код представлен в листинге 1.

Листинг 1 – Метод создания заказа

```
public async Task<IActionResult> CreateOrder([FromBody]
CreateOrderDto createOrderDto)
{
    int userId = int.Parse(User.Claims.FirstOrDefault(c => c.Type
== ClaimTypes.NameIdentifier).Value);

    if (createOrderDto == null || createOrderDto.CartItems == null
|| !createOrderDto.CartItems.Any())
        return BadRequest("Корзина пуста.");

    // Создаем новый заказ
    var order = new Order
    {
        UserId = userId,
        OrderDate = DateTime.UtcNow,
        ShippingAddress = createOrderDto.Address,
        TotalAmount = createOrderDto.CartItems.Sum(item =>
item.Price * item.Quantity),
        Status = OrderStatus.Created,
    };

    await _orderRepository.AddAsync(order);
    await _orderRepository.SaveChangesAsync(); // Сохраняем заказ
и получаем его Id

    // Создаем детали заказа, связывая их с заказом
    foreach (var item in createOrderDto.CartItems)
    {
        var book = await
_bookRepository.GetByIdAsync(item.BookId);
        if (book == null) continue;

        var orderDetail = new OrderDetail
        {
            OrderId = order.Id, // Связываем детали заказа с
созданным заказом
            BookId = item.BookId,
            Quantity = item.Quantity,
```

```

        BookPrice = book.Price
    };

    await _orderDetailRepository.AddAsync(orderDetail);
}

await _orderDetailRepository.SaveChangesAsync();

// Очищаем корзину после создания заказа
_cartItemRepository.ClearCart(userId);
await _cartItemRepository.SaveChangesAsync();

return Ok(new { message = "Заказ успешно создан." });
}

```

Данный метод находится в серверной части приложения и вызывается при оформлении заказа пользователем. Он получает данные корзины, формирует заказ и записывает данные заказа в базу данных.

Для данного метода будут вычислены две группы метрик: метрики размера программы и метрики Холстеда.

4.3.1 Метрики размера программы

Данные метрики представляют собой подсчет количества строк программного кода, а также числа строк с комментариями. Данный метод легок для оценки, а также хорошо подходит для программных продуктов, которые написаны на одном языке программирования. Результаты вычисления метрик представлены в таблице 23 [30].

Таблица 23 – Метрики размера программы

Метрика	Количество
Количество строк кода	44
Строки-комментарии	5

Также вычислим процент комментариев. Он равен $(5/44) \cdot 100 = 11,364$. Данный показатель указывает на небольшой размер метода, что говорит о том, что данный метод не имеет избыточный или плохо структурированный код.

4.3.2 Метрики Холстеда

Данный метод помогает оценить сложность программы. Полученные, благодаря подсчету, метрики помогают разработчикам понимать и контролировать сложность программного обеспечения, выявлять потенциальные области для оптимизации и повышать общее качество программного обеспечения. Для вычисления метрик Холстеда, подсчитаем необходимые характеристики программы [31]:

- число уникальных операторов $n1$;
- число уникальных операндов $n2$;
- общее число операторов $N1$;
- общее число операндов $N2$.

Составим словарь операторов и операндов для вычисления метрик. Словарь операторов представлен в таблице 24.

Таблица 24 – Словарь операторов

Операторы	Число вхождений
public	1
async	1
await	6
return	1
if	2
foreach	1
var	4
new	3
=	14
.	28
()	10
{}	4
==	4
	2
!	1
Sum	1
AddAsync	1
SaveChangesAsync	2
FirstOrDefault	1
int.Parse	1

Продолжение таблицы 24

Операторы	Число вхождений
ToList	1
ClearCart	1
Value	1
continue	1
Ok	1
Операторы	Число вхождений
BadRequest	1

Словарь операндов представлен в таблице 25.

Таблица 25 – Словарь операндов

Операнды	Число вхождений
createOrderDto	6
CartItem	4
User	1
Claims	1
ClaimTypes.NameIdentifier	1
userId	4
item	4
Address	1
TotalAmount	1
OrderStatus.Created	1
order	3
orderDetail	2
book	3
BookId	2
Quantity	1
Price	1
«Корзина пуста.»	1
«Заказ успешно создан.»	1

Таким образом, после сложения всех уникальных операторов $n1 = 27$, после сложения всех вхождений операторов $n2 = 18$, после сложения всех уникальных операндов $N1 = 93$, после сложения всех вхождений операндов $N2 = 38$.

На основе метрик Холстеда можно вычислить целый ряд характеристик программы. Рассмотрим некоторые из них.

Расчёт словаря программы производится по формуле:

$$n = n1 + n2 = 27 + 18 = 45,$$

где $n1$ – количество операторов в программе;

$n2$ – количество операндов программе.

Расчёт длины программы производится по формуле:

$$N = N1 + N2 = 93 + 38 = 131,$$

где $N1$ – число вхождений всех операторов в программе;

$N2$ – число вхождений всех операндов программе.

Расчёт объёма программы производится по формуле:

$$V = N * \log_2 n = 131 * \log_2 45 = 242,$$

где N – длина программы;

n – словарь программы.

Расчёт сложности программы производится по формуле:

$$D = (n1/2) * (N2/n2) = (27 / 2) * (38/ 18) = 28,5,$$

где $n1$ – количество операторов в программе;

$n2$ – количество операндов программе;

$N2$ – число вхождений всех операндов программе.

Расчёт оценки интеллектуальных усилий производится по формуле:

$$E = D * V = 28,5 * 242 = 6897,$$

где D – сложность программы;

V – объем программы.

На основе полученных характеристик можно сделать вывод, что код, который мы протестировали обладает маленьким объемом, легок для понимания, легко анализируется и требует меньше умственных затрат для поддержки и отладки.

Заключение

В ходе выполнения выпускной квалификационной работы было разработано мобильное программное приложение для автоматизированного управления книжным магазином.

На этапе анализа требований был проведен анализ предметной области и постановка задачи, были выделены основные функции приложения. Был проанализирован рынок аналогичных программных продуктов, в результате чего были выявлены плюсы и минусы, были выявлены требования к программному продукту и произведен анализ целевой аудитории.

Было осуществлено планирование и объем затрачиваемого бюджета. Общий срок разработки составляет 62 дня. Общая сумма разработки составляет 427860 рублей. Данные показатели говорят об экономической целесообразности разрабатываемого продукта.

На этапе проектирования были выбраны технологии и инструментальные средства разработки. Спроектированы модель базы данных, а также серверная и клиентская части приложения.

На этапе разработки была разработана база данных с использованием языка программирования SQL и развёрнута под управлением СУБД Microsoft SQL Server 2019. Серверная часть приложения была разработана на языке C# с использованием среды программирования Microsoft Visual Studio 2022. Клиентская часть была разработана на языке C# с использованием среды программирования Microsoft Visual Studio 2022.

В ходе тестирования приложения была проверена работоспособность и корректность разрабатываемого продукта. Были подсчитаны метрики для анализа кода. На данном этапе все ошибки и недочеты были устранены.

В дальнейшем планируется опубликовать приложение в онлайн маркете Google Play. Создать версию приложения для операционной системы iOS и опубликовать его серверную часть на облачном сервисе Azure Mobile Services.

Таким образом, в ходе выполнения выпускной квалификационной работы были успешно выполнены все поставленные задачи.

Список использованных источников

1. Демура, Н.А. Цифровизация: сущность и роль в развитии национальной экономики / Н.А. Демура, Н.П. Путинцева // Научный результат. Экономические исследования. – 2021. – №1. – С. 22 – 30.
2. Распоряжение Президента Российской Федерации от 9 мая 2017 г. № 203 «Стратегия развития информационного общества в Российской Федерации на 2017 – 2030 годы». [Электронный ресурс]. URL: http://zmedu.hostedu.ru/files/ykaz_7668.pdf (дата обращения: 06.05.2025).
3. Программа «Цифровая экономика Российской Федерации». Утверждена распоряжением Правительства Российской Федерации от 28 июля 2017 г. № 1632-р. [Электронный ресурс]. URL: <http://static.government.ru/media/files/9gFM4FHj4PsB79I5v7yLVuPgu4bvR7M0.pdf> (дата обращения: 06.05.2025).
4. Информация об интернет-магазине Books.ru. [Электронный ресурс]. URL: <https://www.books.ru/pages.php?id=3> (дата обращения: 04.04.2025).
5. Информация о компании Labirint.ru. [Электронный ресурс]. URL: <https://www.labirint.org/o-kompanii> (дата обращения: 04.04.2025).
6. Информация о компании «Читай-город». [Электронный ресурс]. URL: <https://www.chitai-gorod.ru/about?ysclid=mamlz1x92a833068829> (дата обращения: 04.04.2025).
7. Сохранение тренда на чтение печатных текстов. [Электронный ресурс]. URL: <https://www.kommersant.ru/doc/7624406> (дата обращения: 06.04.2025).
8. Книжный рынок в России. [Электронный ресурс]. URL: <https://www.kommersant.ru/doc/6097227> (дата обращения: 06.04.2025).
9. Глобальная статистика мобильных операционных систем. [Электронный ресурс]. URL: <https://gs.statcounter.com/os-market-share/mobile/worldwide> (дата обращения: 08.04.2025).
10. Доли рынка мобильных операционных систем в России, по годам. [Электронный ресурс]. URL: <https://topic.ru/statistics/technologies-and->

telecommunications/software/doli-rynka-mobilnykh-operatsionnykh-sistem-v-rossii-po-godam/ (дата обращения: 08.04.2025).

11. Популярные российские операционные системы. [Электронный ресурс]. URL: <https://ppt.ru/soft/rossiyskie-analogi-android> (дата обращения: 28.04.2025).

12. Статистика версий Android. [Электронный ресурс]. URL: <https://composables.com/android-distribution-chart> (дата обращения: 28.04.2025).

13. Рамбо, Дж. UML 2.0. Объектно-ориентированное моделирование и разработка, 2-е издание / Дж. Рамбо, М. Блаха // СПб.: Питер, 2007. – 544 с.: ил.

14. Крачтен, Ф. Введение в Rational Unified Process, 2-е издание / Ф. Крачтен // М.: Вильямс, 2002. – 240 с.

15. Шаблон MVC. [Электронный ресурс]. URL: <https://blog.skillfactory.ru/glossary/mvc/> (дата обращения: 19.04.2025).

16. Документация по ASP.NET. [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/aspnet/core/> (дата обращения: 19.04.2025).

17. Документация по Entity Framework. [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/ef> (дата обращения: 19.04.2025).

18. Документация по .Net MAUI. [Электронный ресурс]. URL: <https://learn.microsoft.com/en-us/dotnet/maui/?view=net-maui-9.0> (дата обращения: 19.04.2025).

19. Беркун, С. Искусство управления IT-проектами / С. Беркун // СПб.: Питер, 2007. – 400 с.: ил.

20. Разу, М.Л. Управление проектом. Основы проектного управления: учебник, 4-е издание / М.Л. Разу, Т.М. Бронникова, А.М. Лялин // М.: КНОРУС, 2015. – 756 с.

21. Гвоздев, В.Е. Управление программными проектами: учебное пособие для вузов / В.Е. Гвоздев // Москва: Издательство Юрайт, 2023. – 167 с.

22. Ехлаков, Ю.П. Управление программными проектами. Стандарты, модели: Учебное пособие / Ю.П. Ехлаков // СПб.: Лань, 2019. – 244 с.

23. Иванов, П.В. Управление проектами: Учебное пособие / П.В. Иванов // Рн/Д: Феникс, 2018. – 102 с.

24. Документация по Microsoft SQL. [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/sql/> (дата обращения: 16.05.2025).

25. Трехуровневая архитектура клиент-серверного приложения. [Электронный ресурс]. URL: <https://sky.pro/wiki/sql/trehurovnevaya-klient-servernaya-arhitektura/> (дата обращения: 16.05.2025).

26. Документация по MS Web API. [Электронный ресурс]. URL: <https://learn.microsoft.com/en-us/aspnet/web-api/> (дата обращения: 16.05.2025).

27. Документация по Entity Framework Code First. [Электронный ресурс]. URL: <https://learn.microsoft.com/ru-ru/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application> (дата обращения: 16.05.2025).

28. Метод тестирования «Черный ящик». [Электронный ресурс]. URL: <https://codeby.one/blog/pentesty/testirovanie-chernogo-i-belogo-yashhika-v-chem-raznitsa/> (дата обращения: 20.05.2025).

29. Модульное тестирование. [Электронный ресурс]. URL: https://logrocon.ru/news/unit_testing (дата обращения: 23.05.2025).

30. Метрики размера программы. [Электронный ресурс]. URL: https://translated.turbopages.org/proxy_u/en-ru.ru.b6f4eb03-6831fb87-e7ebf1f7-74722d776562/https/www.baeldung.com/cs/software-metrics-loc (дата обращения: 23.05.2025).

31. Метрики Холстеда. [Электронный ресурс]. URL: https://translated.turbopages.org/proxy_u/en-ru.ru.c9c99f13-6831fcd4-2f601953-74722d776562/https/www.geeksforgeeks.org/software-engineering-halsteads-software-metrics/ (дата обращения: 23.05.2025).

Приложение А

(справочное)

Глоссарий

Android – мобильная операционная система для смартфонов, планшетов и других мобильных устройств, разрабатываемая и выпускаемая американской компанией Google и используемая на подавляющем большинстве современных мобильных устройств.

API (Application Programming Interface) – набор готовых классов, функций, процедур, констант и структур, предоставляемых приложением или операционной системой для использования во внешних программных продуктах.

RUP (Rational Unified Process) – методология разработки программного обеспечения, в основе которой лежит итеративная модель разработки.

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания и модификации данных в реляционной базе данных, работающей под управлением соответствующей системой управления базами данных, а также для управления этими данными.

HTTP (HyperText Transfer Protocol) – протокол прикладного уровня передачи данных, основанный на технологии «клиент–сервер».

UML (Unified Modeling Language) – язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес–процессов, системного проектирования и отображения организационных структур.

.NET MAUI (.NET Multi–Platform App UI) – это кросс–платформенная платформа для создания собственных мобильных и классических приложений с помощью C# и XAML.

Entity Framework – представляет специальную объектно–ориентированную технологию на базе фреймворка .NET для работы с данными, обладает высоким уровнем абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища.

Приложение Б

(обязательное)

Исходный код серверной части

Листинг Б.1 – Файл Answer.cs

```
namespace BookStore.Domain.Entities;

/// <summary>
/// Модель ответа на вопрос.
/// </summary>
public class Answer
{
    /// <summary>
    /// Идентификатор ответа.
    /// </summary>
    public int Id { get; set; }

    /// <summary>
    /// Текст ответа.
    /// </summary>
    public string Content { get; set; } = string.Empty;

    /// <summary>
    /// Дата создания ответа.
    /// </summary>
    public DateTime CreatedAt { get; set; } = DateTime.UtcNow;

    /// <summary>
    /// Идентификатор вопроса.
    /// </summary>
    public int QuestionId { get; set; }

    /// <summary>
    /// Навигационной свойство вопроса.
    /// </summary>
    public Question Question { get; set; } = null!;

    /// <summary>
    /// Пользователь, ответивший на вопрос.
    /// </summary>
    public int UserId { get; set; }

    /// <summary>
    /// Навигационное свойство пользователя, ответившего на
    вопрос.
    /// </summary>
    public User User { get; set; } = null!;
}
```

Листинг Б.2 – Файл Book.cs

```
using BookStore.Domain.Enums;

namespace BookStore.Domain.Entities;

/// <summary>
```

```

/// Книга в каталоге.
/// </summary>
public class Book
{
    /// <summary>
    /// Идентификатор книги.
    /// </summary>
    public int Id { get; set; }

    /// <summary>
    /// Название книги.
    /// </summary>
    public string Title { get; set; } = null!;

    /// <summary>
    /// Автор книги.
    /// </summary>
    public string Author { get; set; } = null!;

    /// <summary>
    /// Цена книги.
    /// </summary>
    public decimal Price { get; set; }

    /// <summary>
    /// Жанр книги.
    /// </summary>
    public BookGenre Genre { get; set; }

    /// <summary>
    /// Категория книги.
    /// </summary>
    public BookCategory Category { get; set; }

    /// <summary>
    /// Год издания книги.
    /// </summary>
    public int Year { get; set; }

    /// <summary>
    /// Описание книги.
    /// </summary>
    public string Description { get; set; } = null!;

    /// <summary>
    /// Путь к изображению книги.
    /// </summary>
    public string ImagePath { get; set; } = null!;

    /// <summary>
    /// Количество книг в наличии.
    /// </summary>

```

```

        public int QuantityInStock { get; set; }

        /// <summary>
        /// ОТЗЫВЫ О КНИГЕ.
        /// </summary>
        public ICollection<Review> Reviews { get; set; } = new
List<Review>();
    }

```

Листинг Б.3 – Файл CartItem.cs

```

namespace BookStore.Domain.Entities;

/// <summary>
/// Элемент корзины.
/// </summary>
public class CartItem
{
    /// <summary>
    /// Идентификатор элемента корзины.
    /// </summary>
    public int Id { get; set; }

    /// <summary>
    /// Идентификатор книги, добавленной в корзину.
    /// </summary>
    public int BookId { get; set; }

    /// <summary>
    /// Книга, которая находится в корзине пользователя.
    /// </summary>
    /// <remarks>Навигационное свойство.</remarks>
    public Book Book { get; set; } = null!;

    /// <summary>
    /// Идентификатор пользователя, которому принадлежит корзина.
    /// </summary>
    public int UserId { get; set; }

    /// <summary>
    /// Пользователь, которому принадлежит корзина.
    /// </summary>
    /// <remarks>Навигационное свойство.</remarks>
    public User User { get; set; } = null!;

    /// <summary>
    /// Количество книг в корзине.
    /// </summary>
    public int Quantity { get; set; }
}

```

Листинг Б.4 – Файл Order.cs

```

using BookStore.Domain.Enums;

namespace BookStore.Domain.Entities;

/// <summary>
/// Заказ пользователя.
/// </summary>
public class Order
{
    /// <summary>
    /// Идентификатор заказа.
    /// </summary>
    public int Id { get; set; }

    /// <summary>
    /// Идентификатор пользователя, которому принадлежит заказ.
    /// </summary>
    public int UserId { get; set; }

    /// <summary>
    /// Пользователь, которому принадлежит заказ.
    /// </summary>
    /// <remarks>Навигационное свойство.</remarks>
    public User User { get; set; } = null!;

    /// <summary>
    /// Дата и время создания заказа.
    /// </summary>
    public DateTime OrderDate { get; set; }

    /// <summary>
    /// Адрес доставки заказа.
    /// </summary>
    public string ShippingAddress { get; set; } = null!;

    /// <summary>
    /// Общая сумма заказа.
    /// </summary>
    public decimal TotalAmount { get; set; }

    /// <summary>
    /// Статус заказа.
    /// </summary>
    public OrderStatus Status { get; set; }

    /// <summary>
    /// Элементы заказа (книги в заказе).
    /// </summary>
    public ICollection<OrderDetail> OrderDetails { get; set; } =
new List<OrderDetail>();
}

```

Листинг Б.5 – Файл OrderDetail.cs

```
namespace BookStore.Domain.Entities;

/// <summary>
/// Детали заказа.
/// </summary>
public class OrderDetail
{
    /// <summary>
    /// Идентификатор детали заказа.
    /// </summary>
    public int Id { get; set; }

    /// <summary>
    /// Идентификатор заказа, к которому относится деталь заказа.
    /// </summary>
    public int OrderId { get; set; }

    /// <summary>
    /// Заказ, к которому относится деталь заказа.
    /// </summary>
    /// <remarks>Навигационное свойство.</remarks>
    public Order Order { get; set; } = null!;

    /// <summary>
    /// Идентификатор книги, которая входит в деталь заказа.
    /// </summary>
    public int BookId { get; set; }

    /// <summary>
    /// Книга, которая входит в деталь заказа.
    /// </summary>
    /// <remarks>Навигационное свойство.</remarks>
    public Book Book { get; set; } = null!;

    /// <summary>
    /// Количество книг в детали заказа.
    /// </summary>
    public int Quantity { get; set; }

    /// <summary>
    /// Цена книги в момент заказа.
    /// </summary>
    public decimal BookPrice { get; set; }
}
```

Листинг Б.6 – Файл Question.cs

```
namespace BookStore.Domain.Entities;

/// <summary>
/// Модель вопроса, заданного пользователем.
```

```

/// </summary>
public class Question
{
    /// <summary>
    /// Идентификатор вопроса.
    /// </summary>
    public int Id { get; set; }

    /// <summary>
    /// Текст вопроса.
    /// </summary>
    public string Content { get; set; } = string.Empty;

    /// <summary>
    /// Дата создания вопроса.
    /// </summary>
    public DateTime CreatedAt { get; set; } = DateTime.UtcNow;

    /// <summary>
    /// Идентификатор пользователя, задавшего вопрос.
    /// </summary>
    public int UserId { get; set; }

    /// <summary>
    /// Навигационное свойство пользователя.
    /// </summary>
    public User User { get; set; } = null!;

    /// <summary>
    /// Ответ, данный на вопрос.
    /// </summary>
    public Answer? Answer { get; set; }
}

```

Листинг Б.7 – Файл Review.cs

```

using BookStore.Domain.Enums;

namespace BookStore.Domain.Entities;

/// <summary>
/// Отзыв пользователя на книгу.
/// </summary>
public class Review
{
    /// <summary>
    /// Идентификатор отзыва.
    /// </summary>
    public int Id { get; set; }

    /// <summary>
    /// Идентификатор книги, к которой относится отзыв.
    /// </summary>

```

```

public int BookId { get; set; }

/// <summary>
/// Книга, к которой относится отзыв.
/// </summary>
/// <remarks>Навигационное свойство.</remarks>
public Book Book { get; set; } = null!;

/// <summary>
/// Идентификатор пользователя, оставившего отзыв.
/// </summary>
public int UserId { get; set; }

/// <summary>
/// Пользователь, оставивший отзыв.
/// </summary>
/// <remarks>Навигационное свойство.</remarks>
public User User { get; set; } = null!;

/// <summary>
/// Содержание отзыва.
/// </summary>
public string Content { get; set; } = null!;

/// <summary>
/// Дата и время создания отзыва.
/// </summary>
public DateTime CreatedAt { get; set; }

/// <summary>
/// Статус отзыва.
/// </summary>
public ReviewStatus Status { get; set; }
}

```

Листинг Б.8 – Файл User.cs

```

using BookStore.Domain.Enums;

namespace BookStore.Domain.Entities;

/// <summary>
/// Пользователь системы.
/// </summary>
public class User
{
    /// <summary>
    /// Идентификатор пользователя.
    /// </summary>
    public int Id { get; set; }

    /// <summary>
    /// Имя пользователя.

```

```

    /// </summary>
    public string Login { get; set; } = null!;

    /// <summary>
    /// Пароль пользователя.
    /// </summary>
    public string Password { get; set; } = null!;

    /// <summary>
    /// Роль пользователя.
    /// </summary>
    public UserRole Role { get; set; }

    /// <summary>
    /// Отзывы пользователя.
    /// </summary>
    public ICollection<Review> Reviews { get; set; } = new
List<Review>();

    /// <summary>
    /// Корзина пользователя.
    /// </summary>
    public ICollection<CartItem> CartItems { get; set; } = new
List<CartItem>();

    /// <summary>
    /// Список желаемого пользователя.
    /// </summary>
    public ICollection<WishlistItem> WishlistItems { get; set; } =
new List<WishlistItem>();

    /// <summary>
    /// Заказы пользователя.
    /// </summary>
    public ICollection<Order> Orders { get; set; } = new
List<Order>();

    /// <summary>
    /// Вопросы пользователя.
    /// </summary>
    public ICollection<Question>? Questions { get; set; }

    /// <summary>
    /// Ответы пользователя.
    /// </summary>
    public ICollection<Answer>? Answers { get; set; }
}

```

Листинг Б.9 – Файл WishlistItem.cs

```

namespace BookStore.Domain.Entities;

/// <summary>

```

```

/// Элемент списка отложенного.
/// </summary>
public class WishListItem
{
    /// <summary>
    /// Идентификатор элемента списка отложенного.
    /// </summary>
    public int Id { get; set; }

    /// <summary>
    /// Идентификатор книги, добавленной в список отложенного.
    /// </summary>
    public int BookId { get; set; }

    /// <summary>
    /// Книга, которая находится в списке отложенного
пользователя.
    /// </summary>
    /// <remarks>Навигационное свойство.</remarks>
    public Book Book { get; set; } = null!;

    /// <summary>
    /// Идентификатор пользователя, которому принадлежит список
отложенного.
    /// </summary>
    public int UserId { get; set; }

    /// <summary>
    /// Пользователь, которому принадлежит список отложенного.
    /// </summary>
    /// <remarks>Навигационное свойство.</remarks>
    public User User { get; set; } = null!;

    /// <summary>
    /// Количество книг в списке отложенного.
    /// </summary>
    public int Quantity { get; set; }
}

```

Листинг Б.10 – Файл BookCategory.cs

```

namespace BookStore.Domain.Enums;

/// <summary>
/// Категория книги.
/// </summary>
public enum BookCategory
{
    /// <summary>
    /// Техническая литература.
    /// </summary>
    Technical,

```

```

    /// <summary>
    /// Гуманитарные науки.
    /// </summary>
    Humanities,

    /// <summary>
    /// Искусство.
    /// </summary>
    Art,

    /// <summary>
    /// Образование.
    /// </summary>
    Education,

    /// <summary>
    /// Детская.
    /// </summary>
    Children,

    /// <summary>
    /// Научно-популярная литература.
    /// </summary>
    PopularScience,
    Unknown
}

```

Листинг Б.11 – Файл BookGenre.cs

```

namespace BookStore.Domain.Enums;

/// <summary>
/// Жанр книги.
/// </summary>
public enum BookGenre
{
    /// <summary>
    /// Фантастика.
    /// </summary>
    Fiction,

    /// <summary>
    /// Нон-фикшн (научно-популярная литература).
    /// </summary>
    NonFiction,

    /// <summary>
    /// Фэнтези.
    /// </summary>
    Fantasy,

    /// <summary>
    /// Научная литература.

```

```

    /// </summary>
    Science,

    /// <summary>
    /// Роман.
    /// </summary>
    Romance,

    /// <summary>
    /// Детектив.
    /// </summary>
    Mystery,

    /// <summary>
    /// Ужасы.
    /// </summary>
    Thriller,

    /// <summary>
    /// Историческая литература.
    /// </summary>
    Historical,

    /// <summary>
    /// Биография.
    /// </summary>
    Biography,
    Unknown
}

```

Листинг Б.12 – Файл OrderStatus.cs

```

namespace BookStore.Domain.Enums;

/// <summary>
/// Статус оформления заказа.
/// </summary>
public enum OrderStatus
{
    /// <summary>
    /// Создан.
    /// </summary>
    Created = 0,

    /// <summary>
    /// Оплачен.
    /// </summary>
    Paid = 1,

    /// <summary>
    /// Отправлен.
    /// </summary>
    Shipped = 2,
}

```

```

    /// <summary>
    /// Доставлен.
    /// </summary>
    Delivered = 3
}

```

Листинг Б.13 – Файл ReviewStatus.cs

```

namespace BookStore.Domain.Enums;

/// <summary>
/// Статус отзыва.
/// </summary>
public enum ReviewStatus
{
    /// <summary>
    /// Опубликован.
    /// </summary>
    Published = 0,

    /// <summary>
    /// Изменен менеджером.
    /// </summary>
    EditedByManager = 1
}

```

Листинг Б.14 – Файл UserRole.cs

```

namespace BookStore.Domain.Enums;

/// <summary>
/// Роль пользователя в системе.
/// </summary>
public enum UserRole
{
    /// <summary>
    /// Покупатель.
    /// </summary>
    Customer = 0,

    /// <summary>
    /// Менеджер по работе с клиентами.
    /// </summary>
    Manager = 1,

    /// <summary>
    /// Администратор.
    /// </summary>
    Admin = 2
}

```

Листинг Б.15 – Файл BookStoreDbContext.cs

```

using BookStore.Domain.Entities;
using BookStore.Domain.Enums;
using Microsoft.EntityFrameworkCore;

namespace BookStore.Infrastructure.Db;

/// <summary>
/// Контекст базы данных для книжного магазина.
/// </summary>
public class BookStoreDbContext : DbContext
{
    /// <summary>
    /// Конструктор контекста базы данных.
    /// </summary>
    /// <param name="options">Параметры контекста БД.</param>
    public BookStoreDbContext(DbContextOptions<BookStoreDbContext>
options) : base(options)
    {
        Database.EnsureCreated();
    }

    /// <summary>
    /// Пользователи книжного магазина.
    /// </summary>
    public DbSet<User> Users => Set<User>();

    /// <summary>
    /// Книги в книжном магазине.
    /// </summary>
    public DbSet<Book> Books => Set<Book>();

    /// <summary>
    /// Отзывы на книги.
    /// </summary>
    public DbSet<Review> Reviews => Set<Review>();

    /// <summary>
    /// Корзина пользователей.
    /// </summary>
    public DbSet<CartItem> CartItems => Set<CartItem>();

    /// <summary>
    /// Список отложенного для пользователей.
    /// </summary>
    public DbSet<WishlistItem> WishlistItems =>
Set<WishlistItem>();

    /// <summary>
    /// Заказы пользователей.
    /// </summary>
    public DbSet<Order> Orders => Set<Order>();

```

```

    /// <summary>
    /// Детали заказов пользователей.
    /// </summary>
    public DbSet<OrderDetail> OrderDetails => Set<OrderDetail>();

    /// <summary>
    /// Вопросы пользователей.
    /// </summary>
    public DbSet<Question> Questions => Set<Question>();

    /// <summary>
    /// Ответы на вопросы.
    /// </summary>
    public DbSet<Answer> Answers => Set<Answer>();

    /// <summary>
    /// Конфигурация модели базы данных.
    /// </summary>
    protected override void OnModelCreating(ModelBuilder
modelBuilder)
    {
        base.OnModelCreating(modelBuilder);

        // Уникальный логин
        modelBuilder.Entity<User>()
            .HasIndex(u => u.Login)
            .IsUnique();

        // Отношения "пользователь - отзыв"
        modelBuilder.Entity<Review>()
            .HasOne(r => r.User)
            .WithMany(u => u.Reviews)
            .HasForeignKey(r => r.UserId);

        // Отношения "книга - отзыв"
        modelBuilder.Entity<Review>()
            .HasOne(r => r.Book)
            .WithMany(b => b.Reviews)
            .HasForeignKey(r => r.BookId);

        // Корзина
        modelBuilder.Entity<CartItem>()
            .HasOne(ci => ci.User)
            .WithMany(u => u.CartItems)
            .HasForeignKey(ci => ci.UserId);

        modelBuilder.Entity<CartItem>()
            .HasOne(ci => ci.Book)
            .WithMany()
            .HasForeignKey(ci => ci.BookId);

        // Список отложенного

```

```

modelBuilder.Entity<WishlistItem>()
    .HasOne(wi => wi.User)
    .WithMany(u => u.WishlistItems)
    .HasForeignKey(wi => wi.UserId);

modelBuilder.Entity<WishlistItem>()
    .HasOne(wi => wi.Book)
    .WithMany()
    .HasForeignKey(wi => wi.BookId);

// Заказы
modelBuilder.Entity<Order>()
    .HasOne(o => o.User)
    .WithMany(u => u.Orders)
    .HasForeignKey(o => o.UserId);

// Детали заказа
modelBuilder.Entity<OrderDetail>()
    .HasOne(od => od.Order)
    .WithMany(o => o.OrderDetails)
    .HasForeignKey(od => od.OrderId);

modelBuilder.Entity<OrderDetail>()
    .HasOne(od => od.Book)
    .WithMany()
    .HasForeignKey(od => od.BookId);

// Связь вопроса с пользователем
modelBuilder.Entity<Question>()
    .HasOne(q => q.User)
    .WithMany(u => u.Questions)
    .HasForeignKey(q => q.UserId)
    .onDelete(DeleteBehavior.Restrict);

// Связь ответа с вопросом и пользователем
modelBuilder.Entity<Answer>()
    .HasOne(a => a.Question)
    .WithOne(q => q.Answer)
    .HasForeignKey<Answer>(a => a.QuestionId)
    .onDelete(DeleteBehavior.Restrict);

modelBuilder.Entity<Answer>()
    .HasOne(a => a.User)
    .WithMany(u => u.Answers)
    .HasForeignKey(a => a.UserId)
    .onDelete(DeleteBehavior.Restrict);

modelBuilder.Entity<Book>().HasData(
    new Book[]
    {
        new Book
        {

```

```

        Id = 1,
        Title = "Война и мир",
        Author = "Лев Толстой",
        Price = 499.99m,
        Genre = BookGenre.Romance,
        Category = BookCategory.Education,
        Year = 1869,
        Description = "Эпическое произведение о жизни
России в начале XIX века.",
        ImagePath =
"Resources/Images/war_and_peace.jpg",
        QuantityInStock = 10
    },
    new Book
    {
        Id = 2,
        Title = "Преступление и наказание",
        Author = "Фёдор Достоевский",
        Price = 399.99m,
        Genre = BookGenre.Romance,
        Category = BookCategory.Education,
        Year = 1866,
        Description = "Психологический роман о борьбе
совести и искуплении.",
        ImagePath =
"Resources/Images/crime_and_punishment.jpg",
        QuantityInStock = 5
    },
    new Book
    {
        Id = 3,
        Title = "Мастер и Маргарита",
        Author = "Михаил Булгаков",
        Price = 459.99m,
        Genre = BookGenre.Romance,
        Category = BookCategory.Education,
        Year = 1967,
        Description = "Мистический роман о любви,
власти и человеческой природе.",
        ImagePath =
"Resources/Images/master_and_margarita.jpg",
        QuantityInStock = 8
    }
    });

modelBuilder.Entity<User>().HasData(
    new User[]
    {
        new User
        {
            Id = 1,
            Login = "admin",

```

```

        Role = UserRole.Admin,
        Password = "123456"
    },

    new User
    {
        Id = 2,
        Login = "manager",
        Role = UserRole.Manager,
        Password = "123456"
    },

    new User
    {
        Id = 3,
        Login = "customer",
        Role = UserRole.Customer,
        Password = "123456"
    },

    });
}
}

```

Листинг Б.16 – Файл AnswersController.cs

```

using BookStore.Application.Interfaces;
using BookStore.Domain.DTOs;
using BookStore.Domain.Entities;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;

namespace BookStoreAPI.Controllers;

/// <summary>
/// Контроллер для работы с ответами на вопросы.
/// </summary>
[ApiController]
[Route("api/answer")]
[Authorize]
public class AnswersController : ControllerBase
{
    private readonly IAnswerRepository _answerRepository;
    private readonly IQuestionRepository _questionRepository;

    /// <summary>
    /// Конструктор класса.
    /// </summary>
    /// <param name="answerRepository">Репозиторий
    ответов.</param>
    /// <param name="questionRepository">Репозиторий
    вопросов.</param>

```

```

    public AnswersController(IAnswerRepository answerRepository,
        IQuestionRepository questionRepository)
    {
        _answerRepository = answerRepository;
        _questionRepository = questionRepository;
    }

    /// <summary>
    /// Добавление ответа на вопрос.
    /// </summary>
    /// <param name="questionId">Идентификатор вопроса.</param>
    /// <param name="content">Текст ответа.</param>
    [HttpPost("{questionId}")]
    public async Task<IActionResult> AddAnswer(int questionId,
        [FromBody] string content)
    {
        if (string.IsNullOrEmpty(content))
            return BadRequest("Текст ответа не может быть
пустым.");

        // Проверка существования вопроса
        var question = await
            _questionRepository.GetQuestionByIdAsync(questionId);
        if (question == null)
            return NotFound("Вопрос не найден.");

        // Получение ID пользователя
        var userId = int.Parse(User.Claims.FirstOrDefault(c =>
            c.Type == ClaimTypes.NameIdentifier)?.Value ?? "0");

        if (userId == 0)
            return Unauthorized();

        var answer = new Answer
        {
            QuestionId = questionId,
            UserId = userId,
            Content = content,
            CreatedAt = DateTime.UtcNow
        };

        await _answerRepository.AddAnswerAsync(answer);
        return Ok("Ответ успешно добавлен.");
    }

    /// <summary>
    /// Получение всех ответов на вопрос.
    /// </summary>
    /// <param name="questionId">Идентификатор вопроса.</param>
    [HttpGet("{questionId}")]
    public async Task<IActionResult> GetAnswersByQuestion(int
        questionId)

```

```

        {
            var answers = await
_answerRepository.GetAnswersByQuestionIdAsync(questionId);
            return Ok(answers);
        }

        /// <summary>
        /// Получение всех вопросов и их ответов.
        /// </summary>
        [HttpGet("all")]
        [AllowAnonymous] // Сделаем публичным для удобства
        тестирования
        public async Task<IActionResult> GetAllQuestionsWithAnswers()
        {
            var questions = await
_questionRepository.GetQuestionsAsync();

            var questionsWithAnswers = questions.Select(q => new
            QuestionDto
            {
                Id = q.Id,
                Content = q.Content,
                CreatedAt = q.CreatedAt,
                Answer = q.Answer?.Content
            }).ToList();

            return Ok(questionsWithAnswers);
        }
    }
}

```

Листинг Б.17 – Файл AuthController.cs

```

using BookStore.Application.Interfaces;
using BookStore.Domain.DTOs.Jwt;
using BookStore.Domain.DTOs.User;
using BookStore.Domain.Utills.Converters;
using BookStoreAPI.Services;
using Microsoft.AspNetCore.Mvc;

namespace BookStoreAPI.Controllers;

/// <summary>
/// Контроллер, отвечающий за аутентификацию и регистрацию
пользователей.
/// </summary>
[ApiController]
[Route("api/auth")]
public class AuthController : ControllerBase
{
    private readonly IUserRepository _userRepository;
    private readonly TokenService _tokenService;

    /// <summary>

```

```

    /// Конструктор контроллера аутентификации.
    /// </summary>
    /// <param name="userRepository">Объект для доступа к
репозиторию пользователей.</param>
    /// <param name="tokenService">Объект для генерации JWT-
токена.</param>
    public AuthController(IUserRepository userRepository,
TokenService tokenService)
    {
        _userRepository = userRepository;
        _tokenService = tokenService;
    }

    /// <summary>
    /// Регистрация нового пользователя.
    /// </summary>
    /// <param name="model">Данные для регистрации нового
пользователя.</param>
    /// <returns>Результат выполнения регистрации.</returns>
    [HttpPost("register")]
    public async Task<IActionResult> Register([FromBody]
RegisterModel model)
    {
        if (model.Password != model.ConfirmPassword)
        {
            return BadRequest("Пароли не совпадают");
        }

        if (await _userRepository.IsLoginTakenAsync(model.Login))
        {
            return BadRequest("Данный логин уже занят");
        }

        var user = model.ToUser();

        await _userRepository.AddAsync(user);

        return Ok("Пользователь успешно зарегистрирован");
    }

    /// <summary>
    /// Аутентификация пользователя.
    /// </summary>
    /// <param name="model">Данные для аутентификации
пользователя.</param>
    /// <returns>Результат аутентификации пользователя.</returns>
    [HttpPost("login")]
    public async Task<IActionResult> Login([FromBody] LoginModel
model)
    {
        var user = await
_userRepository.GetByLoginAsync(model.Login);

```

```

        if (user == null || user.Password != model.Password)
        {
            return Unauthorized("Не удалось найти пользователя с такими данными");
        }

        var token = _tokenService.GenerateJwtToken(user);

        return Ok(new { Token = token });
    }
}

```

Листинг Б.18 – Файл BooksController.cs

```

using BookStore.Application.Interfaces;
using BookStore.Domain.Entities;
using BookStore.Domain.Enums;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using BookStore.Domain.DTOs;
using BookStore.Infrastructure.Repositories;

namespace BookStoreAPI.Controllers;

/// <summary>
/// Контроллер для работы с каталогом книг.
/// </summary>
[ApiController]
[Route("api/books")]
public class BooksController : ControllerBase
{
    private readonly IBookRepository _bookRepository;
    private readonly IOrderRepository _orderRepository;

    /// <summary>
    /// Конструктор контроллера книг.
    /// </summary>
    /// <param name="bookRepository">Репозиторий книг.</param>
    public BooksController(IBookRepository bookRepository,
        IOrderRepository orderRepository)
    {
        _bookRepository = bookRepository;
        _orderRepository = orderRepository;
    }

    /// <summary>
    /// Получить список всех книг.
    /// </summary>
    /// <returns>Список книг.</returns>
    [HttpGet]
    public async Task<IActionResult> GetAll()
    {

```

```

        var books = await _bookRepository.GetAllAsync();

        return Ok(books);
    }

    /// <summary>
    /// Получить книгу по идентификатору.
    /// </summary>
    /// <param name="id">Идентификатор книги.</param>
    /// <returns>Данные книги.</returns>
    [HttpGet("{id}")]
    public async Task<IActionResult> GetById(int id)
    {
        var book = await _bookRepository.GetByIdAsync(id);

        if (book == null)
        {
            return NotFound("Книга не найдена");
        }

        return Ok(book);
    }

    /// <summary>
    /// Поиск книг по различным параметрам.
    /// </summary>
    /// <param name="title">Название книги.</param>
    /// <param name="author">Автор книги.</param>
    /// <param name="genre">Жанр книги.</param>
    /// <param name="category">Категория книги.</param>
    /// <returns>Список найденных книг.</returns>
    [HttpGet("search")]
    public async Task<IActionResult> Search(
        [FromQuery] string? title = null,
        [FromQuery] string? author = null,
        [FromQuery] BookGenre? genre = null,
        [FromQuery] BookCategory? category = null)
    {
        var books = await _bookRepository.SearchAsync(title,
author, genre, category);

        if (!books.Any())
        {
            return NotFound("Книги не найдены");
        }

        return Ok(books);
    }

    /// <summary>
    /// Добавить новую книгу (доступно только для менеджеров и
администраторов).

```

```

    /// </summary>
    /// <param name="book">Данные книги.</param>
    /// <returns>Результат создания книги.</returns>
    [HttpPost]
    [Authorize(Roles = "Admin")]
    public async Task<IActionResult> Create([FromBody] BookDto
book)
    {
        if (book == null)
        {
            return BadRequest("Некорректные данные книги.");
        }

        var newBook = new Book
        {
            Title = book.Title!,
            Author = book.Author!,
            Genre = (BookGenre)book.Genre!,
            Category = (BookCategory)book.Category!,
            Price = (decimal)book.Price!,
            QuantityInStock = (int)book.QuantityInStock!,
            Year = (int)book.Year!,
            Description = book.Description!,
            ImagePath = book.ImagePath!
        };

        await _bookRepository.AddAsync(newBook);
        await _bookRepository.SaveChangesAsync();

        return CreatedAtAction(nameof(GetById), new { id =
newBook.Id }, newBook);
    }

    /// <summary>
    /// Обновить информацию о книге (доступно только для
менеджеров и администраторов).
    /// </summary>
    /// <param name="id">Идентификатор книги.</param>
    /// <param name="book">Обновленные данные книги.</param>
    /// <returns>Результат обновления книги.</returns>
    [HttpPut]
    [Authorize(Roles = "Admin")]
    public async Task<IActionResult> Update(int id, BookDto book)
    {
        if (book == null)
        {
            return BadRequest("Некорректные данные книги.");
        }

        var existingBook = await _bookRepository.GetByIdAsync(id);

        if (existingBook == null)

```

```

    {
        return NotFound("Книга не найдена.");
    }

    existingBook.Title = book.Title!;
    existingBook.Author = book.Author!;
    existingBook.Genre = (BookGenre)book.Genre!;
    existingBook.Category = (BookCategory)book.Category!;
    existingBook.Price = (decimal)book.Price!;
    existingBook.QuantityInStock = (int)book.QuantityInStock!;
    existingBook.ImagePath = book.ImagePath!;

    _bookRepository.Update(existingBook);
    await _bookRepository.SaveChangesAsync();

    return NoContent();
}

/// <summary>
/// Удалить книгу (доступно только для менеджеров и
администраторов).
/// </summary>
/// <param name="id">Идентификатор книги.</param>
/// <returns>Результат удаления.</returns>
[HttpDelete("{id}")]
[Authorize(Roles = "Admin")]
public async Task<IActionResult> Delete(int id)
{
    var book = await _bookRepository.GetByIdAsync(id);

    if (book == null)
    {
        return NotFound("Книга не найдена.");
    }

    _bookRepository.Remove(book);
    await _bookRepository.SaveChangesAsync();

    return NoContent();
}

[Authorize(Roles = "Admin")]
[HttpGet("statistic")]
public async Task<IActionResult> GetStatistics()
{
    var orders = await _orderRepository.GetAll();
    var books = await _bookRepository.GetAllAsync();
    var now = DateTime.UtcNow;

    // Общие данные по продажам
    var totalSold = orders.SelectMany(o =>
o.OrderDetails).Sum(oi => oi.Quantity);

```

```

var totalRevenue = orders.SelectMany(o =>
o.OrderDetails).Sum(oi => oi.BookPrice * oi.Quantity);

// Выручка за текущий месяц и год
var monthlyRevenue = orders
    .Where(o => o.OrderDate.Month == now.Month &&
o.OrderDate.Year == now.Year)
    .SelectMany(o => o.OrderDetails)
    .Sum(oi => oi.BookPrice * oi.Quantity);

var yearlyRevenue = orders
    .Where(o => o.OrderDate.Year == now.Year)
    .SelectMany(o => o.OrderDetails)
    .Sum(oi => oi.BookPrice * oi.Quantity);

// Продажи по категориям
var categoryStats = books
    .GroupBy(b => b.Category)
    .Select(g => new CategorySalesDto
    {
        Category = g.Key,
        TotalSold = orders.SelectMany(o => o.OrderDetails)
            .Where(oi => g.Any(b => b.Id == oi.BookId))
            .Sum(oi => oi.Quantity),
        TotalRevenue = orders.SelectMany(o =>
o.OrderDetails)
            .Where(oi => g.Any(b => b.Id == oi.BookId))
            .Sum(oi => oi.BookPrice * oi.Quantity)
    })
    .ToList();

// Продажи по жанрам
var genreStats = books
    .GroupBy(b => b.Genre)
    .Select(g => new GenreSalesDto
    {
        Genre = g.Key,
        TotalSold = orders.SelectMany(o => o.OrderDetails)
            .Where(oi => g.Any(b => b.Id == oi.BookId))
            .Sum(oi => oi.Quantity),
        TotalRevenue = orders.SelectMany(o =>
o.OrderDetails)
            .Where(oi => g.Any(b => b.Id == oi.BookId))
            .Sum(oi => oi.BookPrice * oi.Quantity)
    })
    .ToList();

// Создаем DTO объекта статистики
var statisticsDto = new BookSalesStatisticsDto
{
    TotalSold = totalSold,
    TotalRevenue = totalRevenue,

```

```

        MonthlyRevenue = monthlyRevenue,
        YearlyRevenue = yearlyRevenue,
        SalesByCategory = categoryStats,
        SalesByGenre = genreStats
    };

    return Ok(statisticsDto);
}
}

```

Листинг Б.19 – Файл CartController.cs

```

using BookStore.Application.Interfaces;
using BookStore.Domain.Entities;
using BookStore.Infrastructure.Repositories;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;

namespace BookStoreAPI.Controllers;

/// <summary>
/// Контроллер для работы с корзиной.
/// </summary>
[ApiController]
[Route("api/cart")]
[Authorize]
public class CartController : ControllerBase
{
    private readonly ICartItemRepository _cartItemRepository;
    private readonly IBookRepository _bookRepository;

    /// <summary>
    /// Конструктор контроллера корзины.
    /// </summary>
    /// <param name="cartItemRepository">Репозиторий для работы с
    элементами корзины.</param>
    /// <param name="bookRepository">Репозиторий для работы с
    книгами.</param>
    public CartController(ICartItemRepository cartItemRepository,
        IBookRepository bookRepository)
    {
        _cartItemRepository = cartItemRepository;
        _bookRepository = bookRepository;
    }

    /// <summary>
    /// Получить элементы корзины для текущего пользователя.
    /// </summary>
    /// <returns>Элементы корзины текущего пользователя.</returns>
    [HttpGet]
    public async Task<IActionResult> GetCartItems()

```

```

    {
        int userId = int.Parse(User.Claims.FirstOrDefault(c =>
c.Type == ClaimTypes.NameIdentifier).Value);

        var cartItems = await
_cartItemRepository.GetByUserIdAsync(userId);

        return Ok(cartItems);
    }

    /// <summary>
    /// Добавить книгу в корзину.
    /// </summary>
    /// <param name="bookId">Идентификатор добавляемой
книги.</param>
    /// <param name="quantity">Количество добавляемых
книг.</param>
    /// <returns></returns>
    [HttpPost]
    public async Task<IActionResult> AddToCart(int bookId, int
quantity)
    {
        int userId = int.Parse(User.Claims.FirstOrDefault(c =>
c.Type == ClaimTypes.NameIdentifier).Value);

        // Проверяем существование книги
        var book = await _bookRepository.GetByIdAsync(bookId);
        if (book == null)
        {
            return NotFound("Книга не найдена");
        }

        // Проверяем, есть ли уже такая книга в корзине
        var existingItem = await
_cartItemRepository.GetByUserAndBookAsync(userId, bookId);

        if (existingItem != null)
        {
            // Если книга уже в корзине, увеличиваем количество
            existingItem.Quantity += quantity;
            _cartItemRepository.Update(existingItem);
        }
        else
        {
            // Добавляем новый элемент в корзину
            var cartItem = new CartItem
            {
                UserId = userId,
                BookId = bookId,
                Quantity = quantity
            };

```

```

        await _cartItemRepository.AddAsync(cartItem);
    }

    await _cartItemRepository.SaveChangesAsync();

    return Ok();
}

/// <summary>
/// Обновить количество книг в корзине.
/// </summary>
/// <param name="bookId">Идентификатор обновляемой
книги.</param>
/// <param name="quantity">Количество книг.</param>
/// <returns></returns>
[HttpPut]
public async Task<IActionResult> UpdateCartItem(int bookId,
int quantity)
{
    int userId = int.Parse(User.Claims.FirstOrDefault(c =>
c.Type == ClaimTypes.NameIdentifier).Value);

    var cartItem = await
_cartItemRepository.GetByUserAndBookAsync(userId, bookId);

    if (cartItem == null)
    {
        return NotFound("Корзина не найдена");
    }

    cartItem.Quantity = quantity;

    _cartItemRepository.Update(cartItem);

    await _cartItemRepository.SaveChangesAsync();

    return Ok();
}

/// <summary>
/// Удаление книги из корзины.
/// </summary>
/// <param name="bookId">Идентификатор удаляемой книги.</param>
/// <returns></returns>
[HttpDelete]
public async Task<IActionResult> RemoveFromCart(int bookId)
{
    int userId = int.Parse(User.Claims.FirstOrDefault(c =>
c.Type == ClaimTypes.NameIdentifier).Value);

    var cartItem = await
_cartItemRepository.GetByUserAndBookAsync(userId, bookId);

```

```

        if (cartItem == null) return NotFound("Cart item not
found.");

        _cartItemRepository.Remove(cartItem);

        await _cartItemRepository.SaveChangesAsync();

        return Ok();
    }

    /// <summary>
    /// Очистить корзину полностью.
    /// </summary>
    /// <returns></returns>
    [HttpDelete("clear")]
    public async Task<IActionResult> ClearCart()
    {
        int userId = int.Parse(User.Claims.FirstOrDefault(c =>
c.Type == ClaimTypes.NameIdentifier).Value);

        _cartItemRepository.ClearCart(userId);

        return Ok();
    }
}

```

Листинг Б.20 – Файл OrdersController.cs

```

using BookStore.Application.Interfaces;
using BookStore.Domain.DTOs;
using BookStore.Domain.Entities;
using BookStore.Domain.Enums;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;

namespace BookStoreAPI.Controllers;

/// <summary>
/// Контроллер для работы с заказами.
/// </summary>
[ApiController]
[Route("api/orders")]
[Authorize]
public class OrdersController : ControllerBase
{
    private readonly IOrderRepository _orderRepository;
    private readonly IOrderDetailRepository
_orderDetailRepository;
    private readonly IBookRepository _bookRepository;
    private readonly ICartItemRepository _cartItemRepository;

```

```

public OrdersController(
    IOrderRepository orderRepository,
    IOrderDetailRepository orderDetailRepository,
    IBookRepository bookRepository,
    ICartItemRepository cartItemRepository)
{
    _orderRepository = orderRepository;
    _orderDetailRepository = orderDetailRepository;
    _bookRepository = bookRepository;
    _cartItemRepository = cartItemRepository;
}

/// <summary>
/// Создать новый заказ.
/// </summary>
[HttpPost("create")]
public async Task<IActionResult> CreateOrder([FromBody]
CreateOrderDto createOrderDto)
{
    int userId = int.Parse(User.Claims.FirstOrDefault(c =>
c.Type == ClaimTypes.NameIdentifier).Value);

    if (createOrderDto == null || createOrderDto.CartItems ==
null || !createOrderDto.CartItems.Any())
        return BadRequest("Корзина пуста.");

    // Создаем новый заказ
    var order = new Order
    {
        UserId = userId,
        OrderDate = DateTime.UtcNow,
        ShippingAddress = createOrderDto.Address,
        TotalAmount = createOrderDto.CartItems.Sum(item =>
item.Price * item.Quantity),
        Status = OrderStatus.Created,
    };

    await _orderRepository.AddAsync(order);
    await _orderRepository.SaveChangesAsync(); // Сохраняем
заказ и получаем его Id

    // Создаем детали заказа, связывая их с заказом
    foreach (var item in createOrderDto.CartItems)
    {
        var book = await
_bookRepository.GetByIdAsync(item.BookId);
        if (book == null) continue;

        var orderDetail = new OrderDetail
        {
            OrderId = order.Id, // Связываем детали заказа с
созданным заказом

```

```

        BookId = item.BookId,
        Quantity = item.Quantity,
        BookPrice = book.Price
    };

    await _orderDetailRepository.AddAsync(orderDetail);
}

await _orderDetailRepository.SaveChangesAsync();

// Очищаем корзину после создания заказа
_cartItemRepository.ClearCart(userId);
await _cartItemRepository.SaveChangesAsync();

return Ok(new { message = "Заказ успешно создан." });
}

[HttpGet]
public async Task<IActionResult> GetUserOrders()
{
    try
    {
        var userIdClaim =
User.FindFirst(ClaimTypes.NameIdentifier);

        if (userIdClaim == null)
            return Unauthorized();

        int userId = int.Parse(userIdClaim.Value);

        var orders = await
_orderRepository.GetByUserIdAsync(userId);

        return Ok(orders);
    }
    catch (Exception ex)
    {
        return UnprocessableEntity(ex.Message);
    }
}
}

```

Листинг Б.21 – Файл QuestionsController.cs

```

using BookStore.Application.Interfaces;
using BookStore.Domain.Entities;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;

namespace BookStoreAPI.Controllers;

/// <summary>

```

```

/// Контроллер для работы с вопросами.
/// </summary>
[ApiController]
[Route("api/question")]
[Authorize]
public class QuestionsController : ControllerBase
{
    private readonly IQuestionRepository _questionRepository;

    public QuestionsController(IQuestionRepository
questionRepository)
    {
        _questionRepository = questionRepository;
    }

    /// <summary>
    /// Создание нового вопроса
    /// </summary>
    [HttpPost("ask")]
    public async Task<IActionResult> AskQuestion([FromBody] string
content)
    {
        if (string.IsNullOrEmpty(content))
            return BadRequest("Текст вопроса не может быть
пустым.");

        var userId = int.Parse(User.Claims.FirstOrDefault(c =>
c.Type == ClaimTypes.NameIdentifier).Value);

        if (userId == null)
            return Unauthorized();

        var question = new Question
        {
            UserId = userId,
            Content = content,
            CreatedAt = DateTime.UtcNow
        };

        await _questionRepository.AddQuestionAsync(question);

        return Ok("Вопрос успешно добавлен");
    }

    /// <summary>
    /// Получение всех вопросов пользователя
    /// </summary>
    [HttpGet("user")]
    public async Task<IActionResult> GetUserQuestions()
    {
        var userId = int.Parse(User.Claims.FirstOrDefault(c =>
c.Type == ClaimTypes.NameIdentifier).Value);

```

```

        if (userId == null)
            return Unauthorized();

        var questions = await
            _questionRepository.GetQuestionsByUserIdAsync(userId);

        return Ok(questions);
    }
}

```

Листинг Б.22 – Файл ReviewsController.cs

```

using BookStore.Application.Interfaces;
using BookStore.Domain.DTOs;
using BookStore.Domain.Entities;
using BookStore.Domain.Enums;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Security.Claims;

namespace BookStoreAPI.Controllers;

/// <summary>
/// Контроллер для работы с отзывами.
/// </summary>
[ApiController]
[Route("api/review")]
[Authorize]
public class ReviewsController : ControllerBase
{
    private readonly IReviewRepository _reviewRepository;
    private readonly IBookRepository _bookRepository;

    public ReviewsController(IReviewRepository reviewRepository,
        IBookRepository bookRepository)
    {
        _reviewRepository = reviewRepository;
        _bookRepository = bookRepository;
    }

    /// <summary>
    /// Создать новый отзыв к книге.
    /// </summary>
    /// <param name="model">Данные отзыва.</param>
    /// <returns>Результат создания.</returns>
    [HttpPost]
    public async Task<ActionResult> AddReview([FromBody]
        CreateReviewDto reviewDto)
    {
        if (!ModelState.IsValid)
            return BadRequest(ModelState);
    }
}

```

```

        int userId = int.Parse(User.Claims.FirstOrDefault(c =>
c.Type == ClaimTypes.NameIdentifier).Value);

        var book = await
_bookRepository.GetByIdAsync(reviewDto.BookId);

        if (book == null)
            return NotFound("Книга не найдена.");

        var review = new Review
        {
            Content = reviewDto.Content,
            CreatedAt = DateTime.UtcNow,
            BookId = reviewDto.BookId,
            UserId = userId,
            Status = ReviewStatus.Published
        };

        await _reviewRepository.AddAsync(review);
        await _reviewRepository.SaveChangesAsync();

        return Ok(new { message = "Отзыв успешно добавлен." });
    }

    /// <summary>
    /// Получить все отзывы к книге.
    /// </summary>
    [HttpGet("book/{bookId}")]
    [AllowAnonymous]
    public async Task<IActionResult> GetReviewsByBook(int bookId)
    {
        var reviews = await
_reviewRepository.GetByBookIdAsync(bookId);

        if (!reviews.Any())
            return NotFound("Отзывы для этой книги не найдены.");

        var reviewDtos = reviews.Select(r => new
        {
            r.Content,
            r.CreatedAt
        });

        return Ok(reviewDtos);
    }

    /// <summary>
    /// Обновить отзыв.
    /// </summary>
    [HttpPut("{reviewId}")]
    public async Task<IActionResult> UpdateReview(int reviewId,
[FromBody] EditReviewDto reviewDto)

```

```

{
    if (!ModelState.IsValid)
        return BadRequest(ModelState);

    var review = await
_reviewRepository.GetByIdAsync(reviewId);

    if (review == null)
        return NotFound("Отзыв не найден.");

    int userId = int.Parse(User.Claims.FirstOrDefault(c =>
c.Type == ClaimTypes.NameIdentifier).Value);

    review.Content = reviewDto.Content;
    review.Status = ReviewStatus.EditedByManager;
    review.CreatedAt = DateTime.UtcNow;

    _reviewRepository.Update(review);
    await _reviewRepository.SaveChangesAsync();

    return Ok(new { message = "Отзыв успешно обновлен." });
}

/// <summary>
/// Получить все отзывы (только для менеджера).
/// </summary>
[HttpGet("all")]
[Authorize(Roles = "Manager")]
public async Task<IActionResult> GetAllReviews()
{
    var reviews = await _reviewRepository.GetAllAsync();
    if (!reviews.Any())
        return NotFound("Отзывы отсутствуют.");

    var reviewDtos = reviews.Select(r => new EditReviewDto
    {
        Id = r.Id,
        Content = r.Content,
        CreatedAt = r.CreatedAt
    });

    return Ok(reviewDtos);
}

/// <summary>
/// Получить отзыв по идентификатору.
/// </summary>
/// <param name="reviewId">Идентификатор отзыва.</param>
/// <returns>Данные отзыва.</returns>
[HttpGet("{reviewId}")]
[AllowAnonymous]
public async Task<IActionResult> GetReviewById(int reviewId)

```

```

    {
        var review = await
_reviewRepository.GetByIdAsync(reviewId);

        if (review == null)
            return NotFound("Отзыв не найден.");

        var reviewDto = new ReviewDto
        {
            Content = review.Content,
            CreatedAt = review.CreatedAt
        };

        return Ok(reviewDto);
    }
}

```

Листинг Б.23 – Файл WishlistController.cs

```

using BookStore.Application.Interfaces;
using BookStore.Domain.DTOs;
using BookStore.Domain.Entities;
using Microsoft.AspNetCore.Authorization;
using Microsoft.AspNetCore.Mvc;
using System.Data;
using System.Security.Claims;

namespace BookStoreAPI.Controllers;

/// <summary>
/// Контроллер для работы со списком отложенных книг.
/// </summary>
[ApiController]
[Route("api/wishlist")]
[Authorize]
public class WishlistController : ControllerBase
{
    private readonly IWishlistItemRepository _wishlistRepository;
    private readonly IBookRepository _bookRepository;

    /// <summary>
    /// Конструктор контроллера WishlistController.
    /// </summary>
    /// <param name="wishlistRepository">Репозиторий списка
отложенного.</param>
    /// <param name="bookRepository">Репозиторий книг.</param>
    public WishlistController(IWishlistItemRepository
wishlistRepository, IBookRepository bookRepository)
    {
        _wishlistRepository = wishlistRepository;
        _bookRepository = bookRepository;
    }
}

```

```

    /// <summary>
    /// Получить список отложенных книг пользователя.
    /// </summary>
    [HttpGet("user")]
    public async Task<IActionResult> GetUserWishlist()
    {
        int userId = int.Parse(User.Claims.FirstOrDefault(c =>
c.Type == ClaimTypes.NameIdentifier).Value);

        var wishlist = await
_wishlistRepository.GetByUserIdAsync(userId);
        return Ok(wishlist);
    }

    /// <summary>
    /// Добавить книгу в список отложенного.
    /// </summary>
    [HttpPost("add")]
    public async Task<IActionResult> AddToWishlist(int bookId)
    {
        int userId = int.Parse(User.Claims.FirstOrDefault(c =>
c.Type == ClaimTypes.NameIdentifier).Value);

        var book = await _bookRepository.GetByIdAsync(bookId);

        if (book == null)
        {
            return NotFound("Книга не найдена.");
        }

        var existingItem = await
_wishlistRepository.GetByUserAndBookAsync(
            userId, bookId);

        if (existingItem != null)
        {
            return BadRequest("Книга уже в списке отложенного.");
        }

        var wishlistItem = new WishlistItem
        {
            UserId = userId,
            BookId = bookId
        };

        await _wishlistRepository.AddAsync(wishlistItem);

        await _wishlistRepository.SaveChangesAsync();

        return Ok("Книга добавлена в список отложенного.");
    }
}

```

```

    /// <summary>
    /// Удалить книгу из списка отложенного.
    /// </summary>
    /// <param name="bookId">Идентификатор книги.</param>
    [HttpDelete("remove/{bookId}")]
    public async Task<IActionResult> RemoveFromWishlist(int
bookId)
    {
        int userId = int.Parse(User.Claims.FirstOrDefault(c =>
c.Type == ClaimTypes.NameIdentifier).Value);

        var wishlistItem = await
_wishlistRepository.GetByUserAndBookAsync(userId, bookId);

        if (wishlistItem == null)
        {
            return NotFound("Книга не найдена в списке
отложенного.");
        }

        _wishlistRepository.Remove(wishlistItem);

        await _wishlistRepository.SaveChangesAsync();

        return Ok("Книга удалена из списка отложенного.");
    }

    /// <summary>
    /// Очистка списка отложенного пользователя.
    /// </summary>
    /// <returns></returns>
    [HttpDelete("clear")]
    public async Task<IActionResult> Clear()
    {
        int userId = int.Parse(User.Claims.FirstOrDefault(c =>
c.Type == ClaimTypes.NameIdentifier).Value);

        _wishlistRepository.ClearWishlist(userId);

        await _wishlistRepository.SaveChangesAsync();

        return Ok();
    }
}

```

Приложение В

(обязательное)

Исходный код мобильного приложения

Листинг В.1 – Файл LoginViewModel.cs

```

using BookStore.Domain.DTOs.User;
using BookStore.MAUIApp.Services.AuthService;
using System.IdentityModel.Tokens.Jwt;
using System.Windows.Input;

namespace BookStore.MAUIApp.ViewModel.AuthViewModel;

/// <summary>
/// ViewModel для страницы аутентификации.
/// </summary>
public class LoginViewModel : BaseViewModel
{
    private readonly IAuthService _authService;

    /// <summary>
    /// Модель для аутентификации.
    /// </summary>

    private LoginModel model = new();
    public LoginModel Model
    {
        get => model;
        set
        {
            SetProperty(ref model, value);
        }
    }

    /// <summary>
    /// Команда для выполнения аутентификации.
    /// </summary>
    public ICommand LoginCommand { get; }

    /// <summary>
    /// Команда для перехода на страницу регистрации.
    /// </summary>
    public ICommand GoToRegisterCommand { get; }

    /// <summary>
    /// Конструктор ViewModel для страницы аутентификации.
    /// </summary>
    /// <param name="authService">Сервис аутентификации.</param>
    public LoginViewModel(IAuthService authService)
    {
        _authService = authService;
        LoginCommand = new Command(async () => await
LoginAsync());
        GoToRegisterCommand = new Command(async () => await
Shell.Current.GoToAsync("///RegisterPage"));
    }

```

```

/// <summary>
/// Метод для выполнения аутентификации.
/// </summary>
private async Task LoginAsync()
{
    var result = await _authService.LoginAsync(model);

    if (result.IsSuccess)
    {
        Preferences.Set("token", result.Token);

        // Проверяем роль пользователя из JWT-токена
        var role = GetUserRoleFromToken(result.Token);

        if (role == "Manager")
        {
            await
Shell.Current.GoToAsync(":///ManagerProfilePage");
        }
        else if (role == "Admin")
        {
            await Shell.Current.GoToAsync(":///BooksPage");
        }
        else
        {
            await Shell.Current.GoToAsync(":///BooksPage");
        }
    }
    else
    {
        await App.Current.MainPage.DisplayAlert("Ошибка",
result.Message, "Ок");
    }
}

/// <summary>
/// Метод для извлечения роли пользователя из JWT-токена.
/// </summary>
/// <param name="token">JWT-токен.</param>
/// <returns>Роль пользователя.</returns>
private string GetUserRoleFromToken(string token)
{
    var jwtHandler = new JwtSecurityTokenHandler();
    var jwtToken = jwtHandler.ReadJwtToken(token);

    var roleClaim = jwtToken.Claims.FirstOrDefault(c => c.Type
== "role" || c.Type ==
"http://schemas.microsoft.com/ws/2008/06/identity/claims/role");
    return roleClaim?.Value ?? string.Empty;
}
}

```

Листинг В.2 – Файл RegisterViewModel.cs

```

using BookStore.Domain.DTOs.Jwt;
using BookStore.MAUIApp.Services.AuthService;
using System.Windows.Input;

namespace BookStore.MAUIApp.ViewModel.AuthViewModel;

/// <summary>
/// ViewModel для страницы регистрации.
/// </summary>
public class RegisterViewModel : BaseViewModel
{
    private readonly IAuthService _authService;

    /// <summary>
    /// Регистрационные данные.
    /// </summary>
    public RegisterModel Model { get; set; } = new();

    /// <summary>
    /// Команда для регистрации пользователя.
    /// </summary>
    public ICommand RegisterCommand { get; }

    /// <summary>
    /// Команда для перехода на страницу аутентификации.
    /// </summary>
    public ICommand GoToLoginCommand { get; }

    /// <summary>
    /// Конструктор ViewModel для страницы регистрации.
    /// </summary>
    /// <param name="authService">Сервис аутентификации.</param>
    public RegisterViewModel(IAuthService authService)
    {
        _authService = authService;
        RegisterCommand = new Command(async () => await
RegisterAsync());
        GoToLoginCommand = new Command(async () => await
Shell.Current.GoToAsync("///LoginPage"));
    }

    /// <summary>
    /// Метод для выполнения регистрации пользователя.
    /// </summary>
    private async Task RegisterAsync()
    {
        if (Model.Password != Model.ConfirmPassword)
        {
            await App.Current.MainPage.DisplayAlert("Ошибка",
"Пароли не совпадают", "Ок");
            return;
        }
    }
}

```

```

    }

    var result = await _authService.RegisterAsync(Model);

    if (result.IsSuccess)
    {
        await Shell.Current.GoToAsync("//LoginPage");
    }
    else
    {
        await App.Current.MainPage.DisplayAlert("Ошибка",
result.Message, "Ок");
    }
}
}

```

Листинг В.3 – Файл AddBookViewModel.cs

```

using BookStore.Domain.Entities;
using BookStore.Domain.Enums;
using BookStore.MAUIApp.Services.BookService;
using System.Collections.ObjectModel;
using System.Windows.Input;
using BookStore.Domain.DTOs;

namespace BookStore.MAUIApp.ViewModel.BookCatalogViewModel;

public class AddBookViewModel : BaseViewModel
{
    private readonly IBookService _bookService;

    public BookDto Book { get; set; } = new();
    public ObservableCollection<BookGenre> Genres { get; set; } =
new
ObservableCollection<BookGenre>(Enum.GetValues(typeof(BookGenre)).
Cast<BookGenre>());
    public ObservableCollection<BookCategory> Categories { get;
set; } = new
ObservableCollection<BookCategory>(Enum.GetValues(typeof(BookCateg
ory)).Cast<BookCategory>());

    public ICommand AddBookCommand { get; }
    public ICommand CancelCommand { get; }
    public ICommand SelectImageCommand { get; }

    public AddBookViewModel(IBookService bookService)
    {
        _bookService = bookService;
        AddBookCommand = new Command(async () => await
AddBookAsync());
        CancelCommand = new Command(async () => await
Shell.Current.GoToAsync("///ProfilePage"));
    }
}

```

```

        SelectImageCommand = new Command(async () => await
SelectImageAsync());
    }

    private async Task SelectImageAsync()
    {
        var result = await FilePicker.Default.PickAsync(new
PickOptions
        {
            FileTypes = FilePickerFileType.Images,
            PickerTitle = "Выберите изображение книги"
        });

        if (result != null)
        {
            // Путь к папке для изображений в директории данных
приложения
            var imagePath =
Path.Combine(FileSystem.Current.AppDataDirectory, "Images");
            Directory.CreateDirectory(imagePath);

            // Путь к сохраненному файлу
            var filePath = Path.Combine(imagePath,
result.FileName);

            // Копируем файл из выбранного местоположения
            using var stream = await result.OpenReadAsync();
            using var fileStream = File.Create(filePath);
            await stream.CopyToAsync(fileStream);

            // Устанавливаем путь для изображения в DTO
            Book.ImagePath = filePath;
            OnPropertyChanged(nameof(Book.ImagePath));
        }
    }

    private async Task AddBookAsync()
    {
        if (string.IsNullOrEmpty(Book.Title) ||
            string.IsNullOrEmpty(Book.Author) ||
            Book.Price <= 0 ||
            Book.QuantityInStock < 0 ||
            string.IsNullOrEmpty(Book.ImagePath) ||
            string.IsNullOrEmpty(Book.Description))
        {
            await App.Current.MainPage.DisplayAlert("Ошибка",
"Заполните все поля корректно.", "Ок");
            return;
        }

        // Проверка длины и запрещённых символов в названии
        if (Book.Title.Length < 2 || Book.Title.Length > 100 ||

```

```

System.Text.RegularExpressions.Regex.IsMatch(Book.Title,
@"[^\w\s\-\a-яА-ЯЁЁ]") || // Разрешаем буквы, цифры, пробелы, дефис
    Book.Title.Contains("_")) // Явно запрещаем
подчёркивания
    {
        await App.Current.MainPage.DisplayAlert("Ошибка",
"Название книги должно быть от 2 до 100 символов и не содержать
специальных символов.", "Ок");
        return;
    }

    try
    {
        await _bookService.CreateBookAsync(Book);
        await App.Current.MainPage.DisplayAlert("Успех",
"Книга успешно добавлена.", "Ок");
        await Shell.Current.GoToAsync(":///ProfilePage");
    }
    catch (Exception ex)
    {
        await App.Current.MainPage.DisplayAlert("Ошибка", $"Не
удалось добавить книгу: {ex.Message}", "Ок");
    }
}
}

```

Листинг В.4 – Файл BookDetailsViewModel.cs

```

using BookStore.Domain.Enums;
using System.ComponentModel;
using System.Windows.Input;

namespace BookStore.MAUIApp.ViewModel.BookCatalogViewModel;

[QueryProperty(nameof(BookId), "BookId")]
[QueryProperty(nameof(Title), "Title")]
[QueryProperty(nameof(Author), "Author")]
[QueryProperty(nameof(Price), "Price")]
[QueryProperty(nameof(GenreString), "GenreString")]
[QueryProperty(nameof(CategoryString), "CategoryString")]
[QueryProperty(nameof(Year), "Year")]
[QueryProperty(nameof(Description), "Description")]
[QueryProperty(nameof(ImagePath), "ImagePath")]
[QueryProperty(nameof(QuantityInStock), "QuantityInStock")]
public class BookDetailsViewModel : BaseViewModel
{
    // Свойства для строковых значений (принимаемых как параметры)
    private string genreString = string.Empty;
    public string GenreString
    {
        get => genreString;
        set
    }
}

```

```

        {
            if (SetProperty(ref genreString, value))
                InitializeProperties();
        }
    }

private string categoryString = string.Empty;
public string CategoryString
{
    get => categoryString;
    set
    {
        if (SetProperty(ref categoryString, value))
            InitializeProperties();
    }
}

// ОСНОВНЫЕ СВОЙСТВА
private string title;
public string Title
{
    get => title;
    set => SetProperty(ref title, value);
}

private string author;
public string Author
{
    get => author;
    set => SetProperty(ref author, value);
}

private int bookId;

public int BookId
{
    get => bookId;
    set => SetProperty(ref bookId, value);
}

private decimal price;
public decimal Price
{
    get => price;
    set => SetProperty(ref price, value);
}

private BookGenre genre;
public BookGenre Genre
{
    get => genre;
    private set => SetProperty(ref genre, value);
}

```

```

    }

    private BookCategory category;
    public BookCategory Category
    {
        get => category;
        private set => SetProperty(ref category, value);
    }

    private int year;
    public int Year
    {
        get => year;
        set => SetProperty(ref year, value);
    }

    private string description;
    public string Description
    {
        get => description;
        set => SetProperty(ref description, value);
    }

    private string imagePath;
    public string ImagePath
    {
        get => imagePath;
        set => SetProperty(ref imagePath, value);
    }

    private int quantityInStock;
    public int QuantityInStock
    {
        get => quantityInStock;
        set => SetProperty(ref quantityInStock, value);
    }

    public ICommand AddReviewCommand { get; }
    public ICommand GoToBack { get; }
    public ICommand ViewReviewsCommand { get; }

    public BookDetailsViewModel()
    {
        AddReviewCommand = new Command(async () => await
NavigateToAddReviewPage());
        ViewReviewsCommand = new Command(async () => await
NavigateToReviewsPage());
        GoToBack = new Command(async () => await
Shell.Current.GoToAsync(":///BooksPage"));
    }

```

```

private async Task NavigateToAddReviewPage()
{
    await
Shell.Current.GoToAsync($"///AddReviewPage?BookId={bookId}&Title={
Uri.EscapeDataString(Title)}");
}

/// <summary>
/// Переход на страницу отзывов книги.
/// </summary>
private async Task NavigateToReviewsPage()
{
    await
Shell.Current.GoToAsync($"///ReviewsPage?BookId={bookId}&Title={Ur
i.EscapeDataString(Title)}");
}

/// <summary>
/// Метод инициализации после загрузки данных.
/// </summary>
public void InitializeProperties()
{
    // Преобразуем строки в перечисления безопасно
    if (!string.IsNullOrEmpty(GenreString) &&
Enum.TryParse(GenreString, true, out BookGenre parsedGenre))
    {
        Genre = parsedGenre;
    }
    else
    {
        Genre = BookGenre.Unknown;
    }

    if (!string.IsNullOrEmpty(CategoryString) &&
Enum.TryParse(CategoryString, true, out BookCategory
parsedCategory))
    {
        Category = parsedCategory;
    }
    else
    {
        Category = BookCategory.Unknown;
    }
}
}

```

Листинг В.5 – Файл BooksViewModel.cs

```

using BookStore.Domain.DTOs;
using BookStore.Domain.Entities;
using BookStore.Domain.Enums;
using BookStore.MAUIApp.Services.BookService;
using BookStore.MAUIApp.Services.CartService;

```

```

using BookStore.MAUIApp.Services.Wishlist;
using System.Collections.ObjectModel;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Windows.Input;

namespace BookStore.MAUIApp.ViewModel.BookCatalogViewModel
{
    public class BooksViewModel : BaseViewModel
    {
        private readonly IBookService _bookService;
        private readonly ICartService _cartService;
        private readonly IWishlistService _wishlistService;

        private List<Book> AllBooks { get; set; } = new();
        public ObservableCollection<Book> FilteredBooks { get;
set; } = new();

        private string _searchText;
        public string SearchText
        {
            get => _searchText;
            set
            {
                if (SetProperty(ref _searchText, value))
                    ApplyFilters();
            }
        }

        private BookGenre? _selectedGenre;
        public BookGenre? SelectedGenre
        {
            get => _selectedGenre;
            set
            {
                if (SetProperty(ref _selectedGenre, value))
                    ApplyFilters();
            }
        }

        private BookCategory? _selectedCategory;
        public BookCategory? SelectedCategory
        {
            get => _selectedCategory;
            set
            {
                if (SetProperty(ref _selectedCategory, value))
                    ApplyFilters();
            }
        }

        public bool IsFiltered =>

```

```

        !string.IsNullOrEmpty(SearchText) ||
        SelectedGenre.HasValue ||
        SelectedCategory.HasValue;

public bool IsListEmpty => FilteredBooks.Count == 0;

private bool _isAdmin;
public bool IsAdmin
{
    get => _isAdmin;
    set => SetProperty(ref _isAdmin, value);
}

private BookSalesStatisticsDto? _statistics;
public BookSalesStatisticsDto? Statistics
{
    get => _statistics;
    set => SetProperty(ref _statistics, value);
}

public ICommand NavigateToCatalogCommand { get; }
public ICommand NavigateToCartCommand { get; }
public ICommand NavigateToWishlistCommand { get; }
public ICommand NavigateToProfileCommand { get; }
public ICommand GetStatisticsCommand { get; }

public ICommand SearchBooksCommand { get; }
public ICommand ResetFiltersCommand { get; }
public ICommand AddToCartCommand { get; }
public ICommand AddToWishlistCommand { get; }

public ICommand ViewBookDetailsCommand { get; }

public ICommand DeleteBookCommand { get; }
public ICommand EditBookCommand { get; }

public List<BookGenre> Genres { get; } =
Enum.GetValues(typeof(BookGenre)).Cast<BookGenre>().ToList();
public List<BookCategory> Categories { get; } =
Enum.GetValues(typeof(BookCategory)).Cast<BookCategory>().ToList()
;

public BooksViewModel(IBookService bookService,
ICartService cartService, IWishlistService wishlistService)
{
    _bookService = bookService;
    _cartService = cartService;
    _wishlistService = wishlistService;

    ViewBookDetailsCommand = new Command<Book>(async
(book) => await ViewBookDetailsAsync(book));

```

```

        SearchBooksCommand = new Command(ApplyFilters);
        ResetFiltersCommand = new Command(ResetFilters);
        AddToCartCommand = new Command<Book>(async (book) =>
await AddToCartAsync(book));
        AddToWishlistCommand = new Command<Book>(async (book)
=> await AddToWishlistAsync(book));

        NavigateToCatalogCommand = new Command(async () =>
await Shell.Current.GoToAsync(":///BooksPage"));
        NavigateToCartCommand = new Command(async () => await
Shell.Current.GoToAsync(":///CartPage"));
        NavigateToWishlistCommand = new Command(async () =>
await Shell.Current.GoToAsync(":///WishlistPage"));
        NavigateToProfileCommand = new Command(async () =>
await Shell.Current.GoToAsync(":///ProfilePage"));

        DeleteBookCommand = new Command<Book>(async (book) =>
await DeleteBookAsync(book));
        EditBookCommand = new Command<Book>(async (book) =>
await EditBookAsync(book));
        GetStatisticsCommand = new Command(async () => await
LoadStatisticsAsync());

        LoadUserRole();
        LoadBooksAsync();
    }

    private async Task LoadStatisticsAsync()
    {
        try
        {
            Statistics = await
_bookService.GetBookSalesStatisticsAsync();
        }
        catch (Exception ex)
        {
            await App.Current.MainPage.DisplayAlert("Ошибка",
$"Не удалось загрузить статистику: {ex.Message}", "Ок");
        }
    }

    private async Task EditBookAsync(Book book)
    {
        if (book == null) return;
        await
Shell.Current.GoToAsync($":///EditBookPage?BookId={book.Id}");
    }

    public void LoadUserRole()
    {
        // Загружаем токен и проверяем роль пользователя

```

```

        var jwtToken =
SecureStorage.GetAsync("AuthToken").Result;
        if (!string.IsNullOrEmpty(jwtToken))
        {
            var handler = new JwtSecurityTokenHandler();
            var token = handler.ReadJwtToken(jwtToken);
            var userRole = token.Claims.FirstOrDefault(c =>
c.Type == ClaimTypes.Role)?.Value;
            IsAdmin = userRole == "Admin";
        }
    }

    public async Task LoadBooksAsync()
    {
        var books = await _bookService.GetAllBooksAsync();
        AllBooks = books.ToList();
        ApplyFilters();
    }

    private async Task DeleteBookAsync(Book book)
    {
        if (book == null) return;

        var confirmed = await
App.Current.MainPage.DisplayAlert("Подтверждение",
$"Вы уверены, что хотите удалить книгу \"{book.Title}\"?",
"Удалить", "Отмена");

        if (!confirmed) return;

        try
        {
            await _bookService.DeleteBookAsync(book.Id);
            await LoadBooksAsync();
            await App.Current.MainPage.DisplayAlert("Успех",
"Книга успешно удалена.", "Ок");
        }
        catch (Exception ex)
        {
            await App.Current.MainPage.DisplayAlert("Ошибка",
$"Не удалось удалить книгу: {ex.Message}", "Ок");
        }
    }

    private async Task ViewBookDetailsAsync(Book book)
    {
        await Shell.Current.GoToAsync($"///BookDetailsPage?" +
            $"BookId={book.Id}&" +
            $"Title={book.Title}&" +
            $"Author={book.Author}&" +

```

```

        $"Price={book.Price}&" +

$"GenreString={Uri.EscapeDataString(book.Genre.ToString())}" +

$"CategoryString={Uri.EscapeDataString(book.Category.ToString())}
"+
        $"Year={book.Year}&" +
        $"Description={book.Description}&" +
        $"ImagePath={book.ImagePath}&" +
        $"QuantityInStock={book.QuantityInStock}");
    }

    private void ApplyFilters()
    {
        var filtered = AllBooks.AsEnumerable();

        if (!string.IsNullOrEmpty(SearchText))
            filtered = filtered.Where(b =>
                b.Title.Contains(SearchText,
StringComparison.OrdinalIgnoreCase) ||
                b.Author.Contains(SearchText,
StringComparison.OrdinalIgnoreCase));

        if (SelectedGenre.HasValue)
            filtered = filtered.Where(b => b.Genre ==
SelectedGenre.Value);

        if (SelectedCategory.HasValue)
            filtered = filtered.Where(b => b.Category ==
SelectedCategory.Value);

        UpdateFilteredBooks(filtered);
    }

    private void ResetFilters()
    {
        SearchText = string.Empty;
        SelectedGenre = null;
        SelectedCategory = null;
        ApplyFilters();
    }

    private void UpdateFilteredBooks(IEnumerable<Book> books)
    {
        FilteredBooks.Clear();
        foreach (var book in books)
        {
            FilteredBooks.Add(book);
        }

        OnPropertyChanged(nameof(IsListEmpty));
        OnPropertyChanged(nameof(IsFiltered));
    }

```

```

    }

    private async Task AddToCartAsync(Book book)
    {
        await _cartService.AddToCartAsync(book.Id, 1);
    }

    private async Task AddToWishlistAsync(Book book)
    {
        await _wishlistService.AddToWishlistAsync(book.Id);
    }
}
}

```

Листинг В.6 – Файл EditBookViewModel.cs

```

using BookStore.Domain.DTOs;
using BookStore.Domain.Enums;
using BookStore.MAUIApp.Services.BookService;
using System.Collections.ObjectModel;
using System.Windows.Input;

namespace BookStore.MAUIApp.ViewModel.BookCatalogViewModel;

[QueryProperty(nameof(BookId), "BookId")]
public class EditBookViewModel : BaseViewModel
{
    private int bookId;

    public int BookId
    {
        get => bookId;
        set => SetProperty(ref bookId, value);
    }

    private readonly IBookService _bookService;
    public BookDto Book { get; set; } = new();
    public ObservableCollection<BookGenre> Genres { get; set; }
    public ObservableCollection<BookCategory> Categories { get;
set; }

    public ICommand SaveChangesCommand { get; }
    public ICommand CancelCommand { get; }
    public ICommand SelectImageCommand { get; }

    public EditBookViewModel(IBookService bookService)
    {
        _bookService = bookService;

        SaveChangesCommand = new Command(async () => await
SaveChangesAsync());
        CancelCommand = new Command(async () => await
Shell.Current.GoToAsync(":///BooksPage"));
    }
}

```

```

        SelectImageCommand = new Command(async () => await
SelectImageAsync());

        Genres = new
ObservableCollection<BookGenre>(Enum.GetValues(typeof(BookGenre)).
Cast<BookGenre>());
        Categories = new
ObservableCollection<BookCategory>(Enum.GetValues(typeof(BookCateg
ory)).Cast<BookCategory>());
    }

    public async Task LoadBookAsync(int bookId)
    {
        var book = await _bookService.GetBookByIdAsync(bookId);

        if (book != null)
        {
            Book = new BookDto
            {
                Title = book.Title,
                Author = book.Author,
                Genre = book.Genre,
                Category = book.Category,
                Year = book.Year,
                Description = book.Description,
                Price = book.Price,
                QuantityInStock = book.QuantityInStock,
                ImagePath = book.ImagePath
            };
        }
    }

    private async Task SaveChangesAsync()
    {
        var (status, Msg) = await
_bookService.UpdateBookAsync(Book, bookId);

        if (status)
        {
            await Shell.Current.DisplayAlert("Успех", "Книга
успешно обновлена", "OK");
        }
        else
        {
            await Shell.Current.DisplayAlert("Ошибка", Msg, "OK");
        }

        await Shell.Current.GoToAsync("///BooksPage");
    }

    private async Task SelectImageAsync()
    {

```

```

        var result = await FilePicker.Default.PickAsync(new
PickOptions
    {
        FileTypes = FilePickerFileType.Images,
        PickerTitle = "Выберите изображение книги"
    });

    if (result != null)
    {
        var imagePath =
Path.Combine(FileSystem.Current.AppDataDirectory, "Images");
        Directory.CreateDirectory(imagePath);

        var filePath = Path.Combine(imagePath,
result.FileName);

        using var stream = await result.OpenReadAsync();
        using var fileStream = File.Create(filePath);
        await stream.CopyToAsync(fileStream);

        //Book.ImagePath =
$"Resources/Images/{result.FileName}";
        Book.ImagePath = filePath;
        OnPropertyChanged(nameof(Book.ImagePath));
    }
}
}

```

Листинг В.7 – Файл CartViewModel.cs

```

using BookStore.Domain.Entities;
using BookStore.MAUIApp.Services.CartService;
using BookStore.MAUIApp.Services.OrderService;
using BookStore.MAUIApp.Views.CartView;
using CommunityToolkit.Maui.Views;
using System.Collections.ObjectModel;
using System.Windows.Input;

namespace BookStore.MAUIApp.ViewModel.CartViewModel;

/// <summary>
/// ViewModel для страницы корзины.
/// </summary>
public class CartViewModel : BaseViewModel
{
    private readonly ICartService _cartService;
    private readonly IOrderService _orderService;

    /// <summary>
    /// Коллекция элементов корзины.
    /// </summary>
    public ObservableCollection<CartItem> CartItems { get; set; }
    = new();
}

```

```

    public bool IsCartEmpty => CartItems == null ||
    CartItems.Count == 0;

    public string Address { get; set; } = string.Empty;

    /// <summary>
    /// Общая сумма корзины.
    /// </summary>
    public decimal TotalAmount => CartItems.Sum(item =>
    item.Book.Price * item.Quantity);

    private bool _isOrderPopupVisible;
    public bool IsOrderPopupVisible
    {
        get => _isOrderPopupVisible;
        set
        {
            _isOrderPopupVisible = value;
            OnPropertyChanged(nameof(IsOrderPopupVisible));
        }
    }

    /// <summary>
    /// Команды для управления корзиной.
    /// </summary>
    public ICommand LoadCartCommand { get; }
    public ICommand IncreaseQuantityCommand { get; }
    public ICommand DecreaseQuantityCommand { get; }
    public ICommand RemoveFromCartCommand { get; }
    public ICommand ClearCartCommand { get; }

    public ICommand NavigateToCatalogCommand { get; }
    public ICommand NavigateToCartCommand { get; }
    public ICommand NavigateToWishlistCommand { get; }
    public ICommand NavigateToProfileCommand { get; }

    public ICommand CreateOrderCommand { get; }
    public ICommand CancelOrderCommand { get; }

    public ICommand ShowOrderPopupCommand { get; }

    /// <summary>
    /// Конструктор ViewModel для корзины.
    /// </summary>
    public CartViewModel(ICartService cartService, IOrderService
    orderService)
    {
        _cartService = cartService;
        _orderService = orderService;

        ShowOrderPopupCommand = new Command(ShowOrderPopup);
    }

```

```

        LoadCartCommand = new Command(async () => await
LoadCartAsync());
        CreateOrderCommand = new Command(async () => await
CreateOrderAsync());
        CancelOrderCommand = new Command(HideOrderPopup);

        IncreaseQuantityCommand = new Command<CartItem>(async item
=> await ChangeQuantityAsync(item, item.Quantity + 1));
        DecreaseQuantityCommand = new Command<CartItem>(async item
=> await ChangeQuantityAsync(item, item.Quantity - 1));
        RemoveFromCartCommand = new Command<CartItem>(async item
=> await RemoveFromCartAsync(item));
        ClearCartCommand = new Command(async () => await
ClearCartAsync());

        NavigateToCatalogCommand = new Command(async () => await
Shell.Current.GoToAsync("///BooksPage"));
        NavigateToCartCommand = new Command(async () => await
Shell.Current.GoToAsync("///CartPage"));
        NavigateToWishlistCommand = new Command(async () => await
Shell.Current.GoToAsync("///WishlistPage"));
        NavigateToProfileCommand = new Command(async () => await
Shell.Current.GoToAsync("///ProfilePage"));

        LoadCartCommand.Execute(null);
    }

    private void HideOrderPopup()
    {
        IsOrderPopupVisible = false;
        Address = string.Empty;
        OnPropertyChanged(nameof(IsOrderPopupVisible));
        OnPropertyChanged(nameof(Address));
    }

    private async void ShowOrderPopup()
    {
        var page = Shell.Current.CurrentPage;
        if (page is ContentPage contentPage)
        {
            var popup = new OrderPopup();
            var result = await contentPage.ShowPopupAsync(popup);

            if (result is null)
            {
                return;
            }

            Address = result.ToString();
            await CreateOrderAsync();
        }
    }

```

```

    }

    public async Task LoadCartAsync()
    {
        var cartItems = await _cartService.GetCartItemsAsync();

        CartItems.Clear();

        foreach (var item in cartItems)
        {
            CartItems.Add(item);
        }
        OnPropertyChanged(nameof(TotalAmount));
    }

    private async Task ChangeQuantityAsync(CartItem item, int
delta)
    {
        await
        _cartService.UpdateCartItemQuantityAsync(item.BookId, delta);
        await LoadCartAsync();
    }

    private async Task RemoveFromCartAsync(CartItem item)
    {
        await _cartService.RemoveFromCartAsync(item.BookId);
        await LoadCartAsync();
    }

    private async Task ClearCartAsync()
    {
        await _cartService.ClearCartAsync();
        await LoadCartAsync();
    }

    /// <summary>
    /// Создание заказа.
    /// </summary>
    private async Task CreateOrderAsync()
    {
        if (string.IsNullOrEmpty(Address))
        {
            await Shell.Current.DisplayAlert("Ошибка", "Введите
адрес доставки.", "OK");
            return;
        }

        await _orderService.CreateOrderAsync(Address,
CartItems.ToList());
    }
}

```

Листинг В.8 – Файл OrderPageViewModel.cs

```

using BookStore.Domain.DTOs;
using BookStore.Domain.Entities;
using BookStore.MAUIApp.Services.OrderService;
using System.Collections.ObjectModel;
using System.Windows.Input;

namespace BookStore.MAUIApp.ViewModel.OrderViewModel;

public class OrderPageViewModel : BaseViewModel
{
    private readonly IOrderService _orderService;

    public ObservableCollection<Order> Orders { get; } = new();

    private bool _isBusy;
    public bool IsBusy
    {
        get => _isBusy;
        set => SetProperty(ref _isBusy, value);
    }

    private string _statusMessage;
    public string StatusMessage
    {
        get => _statusMessage;
        set => SetProperty(ref _statusMessage, value);
    }

    public ICommand LoadOrdersCommand { get; }

    public ICommand NavigateToProfileCommand { get; }

    public OrderPageViewModel(IOrderService orderService)
    {
        _orderService = orderService;
        LoadOrdersCommand = new Command(async () => await
LoadOrdersAsync());
        NavigateToProfileCommand = new Command(async () => await
Shell.Current.GoToAsync(":///ProfilePage"));
    }

    public async Task LoadOrdersAsync()
    {
        if (IsBusy)
            return;

        try
        {
            IsBusy = true;
            StatusMessage = string.Empty;

```

```

        var orders = await _orderService.GetUserOrdersAsync();

        Orders.Clear();

        foreach (var order in orders)
            Orders.Add(order);

        if (Orders.Count == 0)
            StatusMessage = "Заказы не найдены.";
    }
    catch (Exception ex)
    {
        StatusMessage = $"Ошибка загрузки заказов:
{ex.Message}";
    }
    finally
    {
        IsBusy = false;
    }
}
}

```

Листинг В.9 – Файл ManagerProfileViewModel.cs

```

using BookStore.Domain.Enums;
using BookStore.MAUIApp.Services.QuestionService;
using BookStore.MAUIApp.Services.ReviewService;
using Newtonsoft.Json.Linq;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Windows.Input;

namespace BookStore.MAUIApp.ViewModel.ProfileViewModel;

public class ManagerProfileViewModel : BaseViewModel
{
    private readonly IReviewService _reviewService;
    private readonly IQuestionService _questionService;

    private string _username;
    public string Username
    {
        get => _username;
        set => SetProperty(ref _username, value);
    }

    private string _userRole;
    public string UserRole
    {
        get => _userRole;
        set => SetProperty(ref _userRole, value);
    }
}

```

```

public ICommand NavigateToManageReviewsCommand { get; }
public ICommand NavigateToManageQuestionsCommand { get; }
public ICommand LogoutCommand { get; }

public ManagerProfileViewModel(IReviewService reviewService,
IQuestionService questionService)
{
    _reviewService = reviewService;
    _questionService = questionService;

    LoadUserCredentials();

    LogoutCommand = new Command(async () => await
LogoutAsync());
    NavigateToManageReviewsCommand = new Command(async () =>
await Shell.Current.GoToAsync("///ManageReviewsPage"));
    NavigateToManageQuestionsCommand = new Command(async () =>
await Shell.Current.GoToAsync("///ManageQuestionsPage"));
}

private async void LoadUserCredentials()
{
    var jwtToken = await SecureStorage.GetAsync("AuthToken");

    if (!string.IsNullOrEmpty(jwtToken))
    {
        var handler = new JwtSecurityTokenHandler();
        var token = handler.ReadJwtToken(jwtToken);

        Username = token.Claims.FirstOrDefault(c => c.Type ==
ClaimTypes.Name)?.Value ?? "Неизвестно";
        var userRole = token.Claims.FirstOrDefault(c => c.Type
== ClaimTypes.Role)?.Value;

        switch (userRole)
        {
            case "Admin":
                UserRole = "Администратор";
                break;
            case "Customer":
                UserRole = "Покупатель";
                break;
            case "Manager":
                UserRole = "Менеджер";
                break;
            default:
                UserRole = "Неизвестно";
                break;
        }
    }
    else
    {

```

```

        Username = "Неизвестно";
        UserRole = "Неизвестно";
    }
}

private async Task LogoutAsync()
{
    // Очищаем данные пользователя
    SecureStorage.Remove("UserLogin");
    SecureStorage.Remove("UserRole");

    // Перенаправляем на страницу входа
    await Shell.Current.GoToAsync("///LoginPage");
}
}

```

Листинг В.10 – Файл ProfileViewModel.cs

```

using BookStore.MAUIApp.Services.BookService;
using BookStore.MAUIApp.Services.QuestionService;
using BookStore.MAUIApp.Views.StatisticView;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Windows.Input;

namespace BookStore.MAUIApp.ViewModel.ProfileViewModel;

public class ProfileViewModel : BaseViewModel
{
    private string _username;
    public string Username
    {
        get => _username;
        set => SetProperty(ref _username, value);
    }

    private string _userRole = "Customer";
    public string UserRole
    {
        get => _userRole;
        set => SetProperty(ref _userRole, value);
    }

    private bool _isAdmin = false;
    public bool IsAdmin
    {
        get => _isAdmin;
        set => SetProperty(ref _isAdmin, value);
    }

    public ICommand LogoutCommand { get; }

    public ICommand NavigateToAskQuestionCommand { get; }
}

```

```

public ICommand NavigateToOrdersCommand { get; }
public ICommand NavigateToQuestionsCommand { get; }

public ICommand NavigateToCatalogCommand { get; }
public ICommand NavigateToCartCommand { get; }
public ICommand NavigateToWishlistCommand { get; }
public ICommand NavigateToProfileCommand { get; }

public ICommand NavigateToAddBookCommand { get; }

public ICommand NavigateToManageReviewsCommand { get; }
public ICommand NavigateToManageQuestionsCommand { get; }

public ICommand GetStatisticsCommand { get; }

private readonly IBookService _bookService;

public ProfileViewModel(IBookService bookService)
{
    _bookService = bookService;

    LoadUserCredentials();

    // Команды навигации
    NavigateToAskQuestionCommand = new Command(async () =>
await GoToAskQuestionPage());
    NavigateToOrdersCommand = new Command(async () => await
GoToOrdersPage());
    NavigateToQuestionsCommand = new Command(async () => await
GoToQuestionsPage());
    LogoutCommand = new Command(async () => await
LogoutAsync());

    NavigateToCatalogCommand = new Command(async () => await
Shell.Current.GoToAsync("///BooksPage"));
    NavigateToCartCommand = new Command(async () => await
Shell.Current.GoToAsync("///CartPage"));
    NavigateToWishlistCommand = new Command(async () => await
Shell.Current.GoToAsync("///WishlistPage"));
    NavigateToProfileCommand = new Command(async () => await
Shell.Current.GoToAsync("///ProfilePage"));
    NavigateToManageReviewsCommand = new Command(async () =>
await Shell.Current.GoToAsync("///ManageReviewsPage"));
    NavigateToManageQuestionsCommand = new Command(async () =>
await Shell.Current.GoToAsync("///ManageQuestionsPage"));

    NavigateToAddBookCommand = new Command(async () => await
GoToAddBookPage());
    GetStatisticsCommand = new Command(async () => await
GoToStatisticsPage());
}

```

```

private async Task GoToStatisticsPage()
{
    // Переходим на страницу статистики
    await Shell.Current.GoToAsync("///StatisticsPage");
}

private async Task GoToAddBookPage()
{
    await Shell.Current.GoToAsync("///AddBookPage");
}

private async Task GoToAskQuestionPage()
{
    await Shell.Current.GoToAsync("///AskQuestionPage");
}

private async Task GoToOrdersPage()
{
    await Shell.Current.GoToAsync("///OrdersPage");
}

private async Task GoToQuestionsPage()
{
    await Shell.Current.GoToAsync("///UserQuestionsPage");
}

public async void LoadUserCredentials()
{
    var jwtToken = await SecureStorage.GetAsync("AuthToken");

    if (!string.IsNullOrEmpty(jwtToken))
    {
        var handler = new JwtSecurityTokenHandler();
        var token = handler.ReadJwtToken(jwtToken);

        Username = token.Claims.FirstOrDefault(c => c.Type ==
ClaimTypes.Name)?.Value ?? "Неизвестно";
        var userRole = token.Claims.FirstOrDefault(c => c.Type
== ClaimTypes.Role)?.Value;

        IsAdmin = userRole == "Admin";
        UserRole = userRole switch
        {
            "Admin" => "Администратор",
            "Customer" => "Покупатель",
            "Manager" => "Менеджер",
            _ => "Неизвестно"
        };
    }
    else
    {

```

```

        Username = "Неизвестно";
        UserRole = "Неизвестно";
        IsAdmin = false;
    }
}

private async Task LogoutAsync()
{
    // Очищаем данные пользователя
    SecureStorage.Remove("UserLogin");
    SecureStorage.Remove("UserRole");
    SecureStorage.Remove("token");

    // Перенаправляем на страницу входа
    await Shell.Current.GoToAsync("///LoginPage");
}
}

```

Листинг В.11 – Файл AskQuestionViewModel.cs

```

using BookStore.MAUIApp.Services.QuestionService;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Windows.Input;

namespace BookStore.MAUIApp.ViewModel.AskQuestionsViewModel;

public class AskQuestionViewModel : BaseViewModel
{
    private readonly IQuestionService _questionService;

    private string _questionContent;
    public string QuestionContent
    {
        get => _questionContent;
        set => SetProperty(ref _questionContent, value);
    }

    private string _statusMessage;
    public string StatusMessage
    {
        get => _statusMessage;
        set => SetProperty(ref _statusMessage, value);
    }

    private bool _isBusy;
    public bool IsBusy
    {
        get => _isBusy;
        set => SetProperty(ref _isBusy, value);
    }

    public ICommand SubmitQuestionCommand { get; }
}

```

```

public ICommand GoToBack { get; }

public AskQuestionViewModel(IQuestionService questionService)
{
    _questionService = questionService;
    SubmitQuestionCommand = new Command(async () => await
SubmitQuestionAsync(), () => !IsBusy);
    GoToBack = new Command(async () => await GoToBackAsync());
}

private async Task GoToBackAsync()
{
    var jwtToken = await SecureStorage.GetAsync("AuthToken");

    var handler = new JwtSecurityTokenHandler();
    var token = handler.ReadJwtToken(jwtToken);

    var userRole = token.Claims.FirstOrDefault(c => c.Type ==
ClaimTypes.Role)?.Value;

    switch (userRole)
    {
        case "Admin":
            await Shell.Current.GoToAsync(":///ProfilePage");
            break;
        case "Manager":
            await
Shell.Current.GoToAsync(":///ManagerProfilePage");
            break;
        default:
            await Shell.Current.GoToAsync(":///ProfilePage");
            break;
    }
}

private async Task SubmitQuestionAsync()
{
    if (string.IsNullOrEmpty(QuestionContent))
    {
        StatusMessage = "Введите текст вопроса.";
        return;
    }

    try
    {
        IsBusy = true;
        StatusMessage = string.Empty;

        var result = await
_questionService.AskQuestionAsync(QuestionContent);
    }
}

```

```

        StatusMessage = result;

        if (result.Contains("успешно"))
        {
            QuestionContent = string.Empty; // очистить поле
            после успешной отправки
        }
    }
    catch (Exception ex)
    {
        StatusMessage = $"Ошибка: {ex.Message}";
    }
    finally
    {
        IsBusy = false;
        ((Command) SubmitQuestionCommand).ChangeCanExecute();
    }
}
}

```

Листинг В.12 – Файл ManageQuestionsViewModel.cs

```

using BookStore.Domain.DTOs;
using BookStore.MAUIApp.Services.QuestionService;
using System.Collections.ObjectModel;
using System.IdentityModel.Tokens.Jwt;
using System.Security.Claims;
using System.Windows.Input;

namespace BookStore.MAUIApp.ViewModel.QuestionsViewModel;

public class ManageQuestionsViewModel : BaseViewModel
{
    private readonly IQuestionService _questionService;
    public ObservableCollection<QuestionDto> Questions { get; set; }
    = new();

    public ICommand AnswerQuestionCommand { get; }
    public ICommand GoToBack { get; }

    public ManageQuestionsViewModel(IQuestionService
questionService)
    {
        _questionService = questionService;
        GoToBack = new Command(async () => await GoToBackAsync());
        AnswerQuestionCommand = new Command<QuestionDto>(async
(question) => await AnswerQuestionAsync(question));
        LoadQuestionsAsync();
    }

    private async Task GoToBackAsync()
    {
        var jwtToken = await SecureStorage.GetAsync("AuthToken");
    }
}

```

```

var handler = new JwtSecurityTokenHandler();
var token = handler.ReadJwtToken(jwtToken);

var userRole = token.Claims.FirstOrDefault(c => c.Type ==
ClaimTypes.Role)?.Value;

switch (userRole)
{
    case "Admin":
        await Shell.Current.GoToAsync(":///ProfilePage");
        break;
    case "Manager":
        await
Shell.Current.GoToAsync(":///ManagerProfilePage");
        break;
    default:
        await Shell.Current.GoToAsync(":///ProfilePage");
        break;
}
}

private async void LoadQuestionsAsync()
{
    var questions = await
_questionService.GetAllQuestionsAsync();

    Questions.Clear();

    foreach (var question in questions)
    {
        question.AnswerText = string.Empty; // Поле для ввода
ответа
        Questions.Add(question);
    }
}

private async Task AnswerQuestionAsync(QuestionDto question)
{
    if (question == null) return;

    if (string.IsNullOrEmpty(question.AnswerText))
    {
        await App.Current.MainPage.DisplayAlert("Ошибка",
"Ответ не может быть пустым.", "Ок");
        return;
    }

    await _questionService.AddAnswerAsync(question.Id,
question.AnswerText);
    await App.Current.MainPage.DisplayAlert("Успех", "Ответ
добавлен.", "Ок");
}

```

```

        // Обновляем список вопросов
        LoadQuestionsAsync();
    }
}

```

Листинг В.13 – Файл UserQuestionsViewModel.cs

```

using BookStore.Domain.Entities;
using BookStore.MAUIApp.Services.QuestionService;
using System.Collections.ObjectModel;
using System.Windows.Input;

namespace BookStore.MAUIApp.ViewModel.UserQuestionsViewModel;

public class UserQuestionsViewModel : BaseViewModel
{
    private readonly IQuestionService _questionService;

    public ObservableCollection<Question> Questions { get; set; }
    = new();
    public ICommand LoadQuestionsCommand { get; }
    public ICommand GoToBack { get; }

    public UserQuestionsViewModel(IQuestionService
questionService)
    {
        _questionService = questionService;
        LoadQuestionsCommand = new Command(async () => await
LoadQuestionsAsync());
        GoToBack = new Command(async () => await
Shell.Current.GoToAsync(":///ProfilePage"));
        LoadQuestionsAsync().ConfigureAwait(false);
    }

    private bool _isBusy;
    public bool IsBusy
    {
        get => _isBusy;
        set => SetProperty(ref _isBusy, value);
    }

    private async Task LoadQuestionsAsync()
    {
        IsBusy = true;
        Questions.Clear();

        var questions = await
_questionService.GetUserQuestionsAsync();
        foreach (var question in questions)
        {
            Questions.Add(question);
        }
    }
}

```

```

        IsBusy = false;
    }
}

```

Листинг В.14 – Файл AddReviewViewModel.cs

```

using BookStore.Domain.DTOs;
using BookStore.MAUIApp.Services.ReviewService;
using System.Windows.Input;

namespace BookStore.MAUIApp.ViewModel.ReviewViewModel;

[QueryProperty(nameof(Title), "Title")]
[QueryProperty(nameof(BookIdString), "BookId")]
public class AddReviewViewModel : BaseViewModel
{
    private readonly IReviewService _reviewService;

    private int bookId;

    public int BookId
    {
        get => bookId;
        set => SetProperty(ref bookId, value);
    }

    private string bookIdString;
    public string BookIdString
    {
        get => bookIdString;
        set
        {
            bookIdString = value;
            if (int.TryParse(bookIdString, out int parsedId))
            {
                BookId = parsedId;
            }
        }
    }

    private string title;
    public string Title
    {
        get => title;
        set => SetProperty(ref title, value);
    }

    private string reviewText;
    public string ReviewText
    {
        get => reviewText;
        set => SetProperty(ref reviewText, value);
    }
}

```

```

    }

    public ICommand SubmitReviewCommand { get; }
    public ICommand GoToBack { get; }

    public AddReviewViewModel(IReviewService reviewService)
    {
        _reviewService = reviewService;
        SubmitReviewCommand = new Command(async () => await
SubmitReview());
        GoToBack = new Command(async () => await
Shell.Current.GoToAsync(":///BookDetailsPage"));
    }

    private async Task SubmitReview()
    {
        if (string.IsNullOrEmpty(ReviewText))
        {
            await
Application.Current.MainPage.DisplayAlert("Ошибка", "Отзыв не
может быть пустым.", "OK");
            return;
        }

        var reviewDto = new CreateReviewDto
        {
            Content = ReviewText,
            BookId = BookId
        };

        await _reviewService.SubmitReviewAsync(reviewDto);
    }
}

```

Листинг В.15 – Файл EditReviewViewModel.cs

```

using BookStore.Domain.DTOs;
using BookStore.MAUIApp.Services.ReviewService;
using System.Threading.Tasks;
using System.Windows.Input;

namespace BookStore.MAUIApp.ViewModel.ReviewViewModel;

[QueryProperty(nameof(ReviewId), "ReviewId")]
public class EditReviewViewModel : BaseViewModel
{
    private readonly IReviewService _reviewService;
    private int _reviewId;
    public int ReviewId
    {
        get => _reviewId;
        set
        {

```

```

        _reviewId = value;
        _ = LoadReviewAsync();
    }
}

private string _reviewContent;
public string ReviewContent
{
    get => _reviewContent;
    set => SetProperty(ref _reviewContent, value);
}

public ICommand SaveReviewCommand { get; }
public ICommand CancelCommand { get; }

public EditReviewViewModel(IReviewService reviewService)
{
    _reviewService = reviewService;
    SaveReviewCommand = new Command(async () => await
SaveReviewAsync());
    CancelCommand = new Command(async () => await
Shell.Current.GoToAsync(":///ManageReviewsPage"));
}

public async Task LoadReviewAsync()
{
    var review = await
_reviewService.GetReviewByIdAsync(ReviewId);
    if (review != null)
    {
        ReviewContent = review.Content;
    }
    else
    {
        await App.Current.MainPage.DisplayAlert("Ошибка",
"Отзыв не найден", "Ок");
        await Shell.Current.GoToAsync(":///ManageReviewsPage");
    }
}

private async Task SaveReviewAsync()
{
    if (string.IsNullOrEmpty(ReviewContent))
    {
        await App.Current.MainPage.DisplayAlert("Ошибка",
"Текст отзыва не может быть пустым.", "Ок");
        return;
    }

    var editDto = new EditReviewDto { Content = ReviewContent
};
    await _reviewService.UpdateReviewAsync(ReviewId, editDto);
}

```

```

        // Обновляем список отзывов
        await Shell.Current.GoToAsync("///ManageReviewsPage");
        if (Shell.Current.CurrentPage.BindingContext is
ManageReviewsViewModel viewModel)
        {
            await viewModel.RefreshReviewsAsync();
        }
    }
}

```

Листинг В.16 – Файл ManageReviewsViewModel.cs

```

using BookStore.Domain.DTOs;
using BookStore.MAUIApp.Services.ReviewService;
using System.Collections.ObjectModel;
using System.Windows.Input;

namespace BookStore.MAUIApp.ViewModel.ReviewViewModel;

public class ManageReviewsViewModel : BaseViewModel
{
    private readonly IReviewService _reviewService;

    public ObservableCollection<EditReviewDto> Reviews { get; set; } = new();
    public ICommand EditReviewCommand { get; }
    public ICommand GoToBack { get; }

    public ManageReviewsViewModel(IReviewService reviewService)
    {
        _reviewService = reviewService;
        EditReviewCommand = new Command<EditReviewDto>(async
(reviewDto) => await EditReviewAsync(reviewDto));
        GoToBack = new Command(async () => await GoToBackAsync());
        LoadReviewsAsync();
    }

    private async Task GoToBackAsync()
    {
        await Shell.Current.GoToAsync("///ManagerProfilePage");
    }

    private async void LoadReviewsAsync()
    {
        var reviews = await _reviewService.GetAllReviewsAsync();

        Reviews.Clear();

        foreach (var review in reviews)
        {
            Reviews.Add(review);
        }
    }
}

```

```

    }

    private async Task EditReviewAsync(EditReviewDto reviewDto)
    {
        if (reviewDto.Id <= 0)
        {
            await App.Current.MainPage.DisplayAlert("Ошибка",
"Некорректный идентификатор отзыва.", "Ок");
            return;
        }

        await
Shell.Current.GoToAsync($"///EditReviewPage?ReviewId={reviewDto.Id
}");
    }

    public async Task RefreshReviewsAsync()
    {
        LoadReviewsAsync();
    }
}

```

Листинг В.17 – Файл ReviewsViewModel.cs

```

using BookStore.Domain.DTOs;
using BookStore.MAUIApp.Utils;
using System.Collections.ObjectModel;
using System.Net.Http.Json;
using System.Windows.Input;

namespace BookStore.MAUIApp.ViewModel.ReviewViewModel;

[QueryProperty(nameof(BookId), "BookId")]
[QueryProperty(nameof(Title), "Title")]
public class ReviewsViewModel : BaseViewModel
{
    private readonly HttpClient _httpClient;

    public ObservableCollection<ReviewDto> Reviews { get; } =
new();

    private int bookId;
    public int BookId
    {
        get => bookId;
        set => SetProperty(ref bookId, value);
    }

    private string title;
    public string Title
    {
        get => title;
        set => SetProperty(ref title, value);
    }
}

```

```

    }

    public ICommand GoToBackCommand { get; }

    public ReviewsViewModel()
    {
        _httpClient = new HttpClient();
        GoToBackCommand = new Command(async () => await
        Shell.Current.GoToAsync(":///BooksPage"));
        LoadReviewsCommand = new Command(async () => await
        LoadReviewsAsync());

        // Автоматически загружаем отзывы при инициализации
        Task.Run(async () => await LoadReviewsAsync());
    }

    public ICommand LoadReviewsCommand { get; }

    public async Task LoadReviewsAsync()
    {
        Reviews.Clear();

        await _httpClient.AddAuthorizationHeaderAsync();

        var response = await
        _httpClient.GetAsync($"http://10.0.2.2:5073/api/review/book/{BookId}");

        if (response.IsSuccessStatusCode)
        {
            var reviews = await
            response.Content.ReadFromJsonAsync<IEnumerable<ReviewDto>>();

            if (reviews != null)
            {
                foreach (var review in reviews)
                    Reviews.Add(review);
            }
        }
        else
        {
            await App.Current.MainPage.DisplayAlert("Отзывы",
            "Отзывы не найдены", "Ок");
        }
    }
}

```

Листинг В.18 – Файл WishlistViewModel.cs

```

using BookStore.Domain.Entities;
using BookStore.MAUIApp.Services.CartService;
using BookStore.MAUIApp.Services.Wishlist;
using System.Collections.ObjectModel;

```

```

using System.Windows.Input;

namespace BookStore.MAUIApp.ViewModel.WishlistViewModel;

public class WishlistViewModel : BaseViewModel
{
    private readonly IWishlistService _wishlistService;
    private readonly ICartService _cartService;

    public ObservableCollection<WishlistItem> WishlistItems { get;
set; } = new();

    public ICommand LoadWishlistCommand { get; }
    public ICommand MoveToCartCommand { get; }
    public ICommand RemoveFromWishlistCommand { get; }
    public ICommand ClearWishlistCommand { get; }

    public ICommand NavigateToCatalogCommand { get; }
    public ICommand NavigateToCartCommand { get; }
    public ICommand NavigateToWishlistCommand { get; }
    public ICommand NavigateToProfileCommand { get; }

    public WishlistViewModel(IWishlistService wishlistService,
ICartService cartService)
    {
        _wishlistService = wishlistService;
        _cartService = cartService;

        LoadWishlistCommand = new Command(async () => await
LoadWishlistAsync());
        MoveToCartCommand = new Command<WishlistItem>(async item
=> await MoveToCartAsync(item));
        RemoveFromWishlistCommand = new
Command<WishlistItem>(async item => await
RemoveFromWishlistAsync(item.BookId));
        ClearWishlistCommand = new Command(async () => await
ClearWishlistAsync());

        NavigateToCatalogCommand = new Command(async () => await
Shell.Current.GoToAsync(":///BooksPage"));
        NavigateToCartCommand = new Command(async () => await
Shell.Current.GoToAsync(":///CartPage"));
        NavigateToWishlistCommand = new Command(async () => await
Shell.Current.GoToAsync(":///WishlistPage"));
        NavigateToProfileCommand = new Command(async () => await
Shell.Current.GoToAsync(":///ProfilePage"));

        // Загрузка при инициализации
        LoadWishlistCommand.Execute(null);
    }

    public async Task LoadWishlistAsync()

```

```

{
    var wishlist = await _wishlistService.GetWishlistAsync();

    WishlistItems.Clear();

    foreach (var book in wishlist)
    {
        WishlistItems.Add(book);
    }
}

private async Task MoveToCartAsync(WishlistItem item)
{
    await
_wishlistService.RemoveFromWishlistAsync(item.BookId);
    await _cartService.AddToCartAsync(item.BookId, 1);
    await LoadWishlistAsync();
}

private async Task RemoveFromWishlistAsync(int bookId)
{
    await _wishlistService.RemoveFromWishlistAsync(bookId);

    var itemToRemove = WishlistItems.FirstOrDefault(b => b.Id
== bookId);

    if (itemToRemove != null)
    {
        WishlistItems.Remove(itemToRemove);
    }

    await LoadWishlistAsync();
}

private async Task ClearWishlistAsync()
{
    await _wishlistService.ClearWishlistAsync();
    await LoadWishlistAsync();
}
}

```

Листинг В.19 – Файл LoginPage.xaml

```

<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
            x:Class="BookStore.MAUIApp.View.AuthView.LoginPage"
            Shell.NavBarIsVisible="False">

    <StackLayout Padding="20" Spacing="15">
        <Label Text="Введите данные" FontSize="32"
HorizontalOptions="Center" />

```

```

        <Label Text="для входа" FontSize="32"
HorizontalOptions="Center" />
        <Entry Text="{Binding Model.Login}" Placeholder="Введите
логин" />
        <Entry Text="{Binding Model.Password}"
Placeholder="Введите пароль" IsPassword="True" />
        <Button Text="Войти" Command="{Binding LoginCommand}" />
        <Label Text="Нет аккаунта?" HorizontalOptions="Center" />
        <Button Text="Зарегистрироваться" Command="{Binding
GoToRegisterCommand}" />
    </StackLayout>
</ContentPage>

```

Листинг В.20 – Файл RegisterPage.xaml

```

<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
x:Class="BookStore.MAUIApp.View.AuthView.LoginPage"
Shell.NavBarIsVisible="False">

    <StackLayout Padding="20" Spacing="15">
        <Label Text="Введите данные" FontSize="32"
HorizontalOptions="Center" />
        <Label Text="для входа" FontSize="32"
HorizontalOptions="Center" />
        <Entry Text="{Binding Model.Login}" Placeholder="Введите
логин" />
        <Entry Text="{Binding Model.Password}"
Placeholder="Введите пароль" IsPassword="True" />
        <Button Text="Войти" Command="{Binding LoginCommand}" />
        <Label Text="Нет аккаунта?" HorizontalOptions="Center" />
        <Button Text="Зарегистрироваться" Command="{Binding
GoToRegisterCommand}" />
    </StackLayout>
</ContentPage>

```

Листинг В.21 – Файл AddBookPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
x:Class="BookStore.MAUIApp.Views.BookView.AddBookPage"
Title="Добавить книгу"
BackgroundColor="#F4F4F9"
Padding="20">

    <ScrollView>
        <StackLayout Spacing="15">
            <Label Text="Добавление новой книги"
FontSize="24"

```

```

        FontAttributes="Bold"
        HorizontalOptions="Center"
        TextColor="#333" />

        <Frame BackgroundColor="White" CornerRadius="10"
        Padding="15" HasShadow="True">
            <StackLayout Spacing="10">
                <Entry Placeholder="Название книги"
                    Text="{Binding Book.Title}"
                    BackgroundColor="#F0F0F5"
                    TextColor="#333" />

                <Entry Placeholder="Автор"
                    Text="{Binding Book.Author}"
                    BackgroundColor="#F0F0F5"
                    TextColor="#333" />

                <Picker Title="Жанр"
                    ItemsSource="{Binding Genres}"
                    SelectedItem="{Binding Book.Genre}"
                    BackgroundColor="#F0F0F5"
                    TextColor="#333">
                    <Picker.ItemDisplayBinding>
                        <Binding Path="."
        Converter="{StaticResource GenreToRussianConverter}" />
                    </Picker.ItemDisplayBinding>
                </Picker>

                <Picker Title="Категория"
                    ItemsSource="{Binding Categories}"
                    SelectedItem="{Binding Book.Category}"
                    BackgroundColor="#F0F0F5"
                    TextColor="#333" >
                    <Picker.ItemDisplayBinding>
                        <Binding Path="."
        Converter="{StaticResource GenreToRussianConverter}" />
                    </Picker.ItemDisplayBinding>
                </Picker>

                <Entry Placeholder="Год издания"
                    Text="{Binding Book.Year}"
                    Keyboard="Numeric"
                    BackgroundColor="#F0F0F5"
                    TextColor="#333" />

                <Editor Placeholder="Описание книги"
                    Text="{Binding Book.Description}"
                    HeightRequest="100"
                    BackgroundColor="#F0F0F5"
                    TextColor="#333" />

                <Entry Placeholder="Цена"

```

```

        Text="{Binding Book.Price}"
        Keyboard="Numeric"
        BackgroundColor="#F0F0F5"
        TextColor="#333" />

        <Entry Placeholder="Количество в наличии"
            Text="{Binding Book.QuantityInStock}"
            Keyboard="Numeric"
            BackgroundColor="#F0F0F5"
            TextColor="#333" />

        <!-- Выбор изображения -->
        <Button Text="Загрузить изображение"
            TextColor="White"
            FontSize="Medium"
            Command="{Binding SelectImageCommand}"
        />

        <Image Source="{Binding Book.ImagePath}"
            HeightRequest="200"
            Aspect="AspectFit"
            Margin="0,10,0,0" />
    </StackLayout>
</Frame>

<!-- Кнопки управления -->
<StackLayout Orientation="Horizontal" Spacing="15">
    <Button Text="Добавить книгу"
        TextColor="White"
        FontSize="Medium"
        Command="{Binding AddBookCommand}"
        HorizontalOptions="FillAndExpand" />

    <Button Text="Отмена"
        TextColor="White"
        FontSize="Medium"
        Command="{Binding CancelCommand}"
        HorizontalOptions="FillAndExpand" />
</StackLayout>
</StackLayout>
</ScrollView>
</ContentPage>

```

Листинг В.22 – Файл BookDetailsPage.xaml

```

<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
    xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:viewModel="clr-namespace:BookStore.MAUIApp.ViewModel.BookCatalogViewModel"
    x:Class="BookStore.MAUIApp.Views.BookView.BookDetailsPage"
    Title="Детали книги"

```

```

        BackgroundColor="#F0F0F5"
        Padding="15">

    <ScrollView>
        <StackLayout Spacing="15">
            <!-- Изображение книги -->
            <Frame Padding="0" CornerRadius="10" HasShadow="True"
BackgroundColor="White">
                <Image Source="{Binding ImagePath}"
                    HeightRequest="300"
                    Aspect="AspectFill"
                    HorizontalOptions="Center"
                    VerticalOptions="Center" />
            </Frame>

            <!-- Основная информация о книге -->
            <Frame BackgroundColor="White" CornerRadius="10"
HasShadow="True" Padding="15">
                <StackLayout Spacing="10">
                    <Label Text="{Binding Title}"
                        FontSize="24"
                        FontAttributes="Bold"
                        TextColor="#333" />

                    <Label Text="Автор:" FontSize="Medium"
TextColor="#666" />
                    <Label Text="{Binding Author}"
FontSize="Medium" TextColor="#333" />

                    <Label Text="Жанр:" FontSize="Medium"
TextColor="#666" />
                    <Label Text="{Binding Genre,
Converter={StaticResource GenreToRussianConverter}}"
FontSize="Medium"
TextColor="#333" />

                    <Label Text="Категория:" FontSize="Medium"
TextColor="#666" />
                    <Label Text="{Binding Category,
Converter={StaticResource GenreToRussianConverter}}"
FontSize="Medium"
TextColor="#333" />
                </StackLayout>
            </Frame>

            <!-- Цена и описание -->
            <Frame BackgroundColor="White" CornerRadius="10"
HasShadow="True" Padding="15">
                <StackLayout Spacing="10">
                    <Label Text="Цена:" FontSize="Medium"
TextColor="#666" />

```

```

        <Label Text="{Binding Price,
StringFormat='{{0:N2}} ₺'"
        FontSize="Large"
        FontAttributes="Bold"
        TextColor="#2E8B57" />

        <Label Text="Описание:" FontSize="Medium"
TextColor="#666" />
        <Label Text="{Binding Description}"
        FontSize="Medium"
        TextColor="#333"
        LineBreakMode="WordWrap" />
    </StackLayout>
</Frame>

    <!-- Дополнительные сведения -->
    <Frame BackgroundColor="White" CornerRadius="10"
HasShadow="True" Padding="15">
        <StackLayout Spacing="10">
            <Label Text="Год издания:" FontSize="Medium"
TextColor="#666" />
            <Label Text="{Binding Year}" FontSize="Medium"
TextColor="#333" />

            <Label Text="Количество в наличии:"
FontSize="Medium" TextColor="#666" />
            <Label Text="{Binding QuantityInStock}"
FontSize="Medium" TextColor="#333" />
        </StackLayout>
    </Frame>

    <!-- Кнопка оставить отзыв -->
    <Button Text="Оставить отзыв"
        Command="{Binding AddReviewCommand}"
        CornerRadius="10"
        HeightRequest="45"
        FontSize="Medium" />

    <!-- Кнопка просмотра отзывов -->
    <Button Text="Посмотреть отзывы"
        Command="{Binding ViewReviewsCommand}"
        CornerRadius="10"
        HeightRequest="45"
        FontSize="Medium" />

    <Button Text="Назад"
        Command="{Binding GoToBack}"
        CornerRadius="10"
        HeightRequest="45"
        FontSize="Medium" />

</StackLayout>

```

```

    </ScrollView>
</ContentPage>

```

Листинг В.23 – Файл AskQuestionPage.xaml

```

<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
    xmlns:viewModel="clr-
namespace:BookStore.MAUIApp.ViewModel.AskQuestionsViewModel"
x:Class="BookStore.MAUIApp.Views.QuestionsView.AskQuestionPage"
    Title="Задать вопрос"
    BackgroundColor="#F0F0F5"
    Padding="15">

    <Grid RowDefinitions="*,Auto" VerticalOptions="FillAndExpand"
    >

        <StackLayout Spacing="20">

            <Label Text="Введите ваш вопрос:" FontSize="Medium"
            TextColor="#333" />

            <Editor Text="{Binding QuestionContent}"
            AutoSize="TextChanges"
            HeightRequest="150"
            Placeholder="Напишите ваш вопрос здесь..."
            BackgroundColor="White"
            TextColor="#000"
            />

            <Button Text="Отправить"
            Command="{Binding SubmitQuestionCommand}"
            IsEnabled="{Binding IsBusy,
Converter={StaticResource InverseBooleanConverter}}}"
            TextColor="White" />

            <Label Text="{Binding StatusMessage}"
            TextColor="Green"
            FontAttributes="Italic"
            HorizontalOptions="Center" />

            <ActivityIndicator IsRunning="{Binding IsBusy}"
            IsVisible="{Binding IsBusy}"
            HorizontalOptions="Center" />

        </StackLayout>

        <!-- Кнопка возврата, закрепленная снизу -->
        <Button Text="Назад"
            Command="{Binding GoToBack}"
            Grid.Row="1"

```

```

        TextColor="White"
        HeightRequest="50"/>

</Grid>
</ContentPage>

```

Листинг В.24 – Файл EditReviewPage.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"

x:Class="BookStore.MAUIApp.Views.ReviewView.EditReviewPage"
        Title="Редактировать отзыв"
        BackgroundColor="#F0F0F5"
        Padding="15">

    <StackLayout Spacing="20">
        <Label Text="Редактирование отзыва" FontSize="24"
FontAttributes="Bold" HorizontalOptions="Center" />

        <Editor Text="{Binding ReviewContent}"
        Placeholder="Введите текст отзыва..."
        AutoSize="TextChanges"
        BackgroundColor="White"
        HeightRequest="200" />

        <Button Text="Сохранить изменения"
        Command="{Binding SaveReviewCommand}" />

        <Button Text="Отмена"
        Command="{Binding CancelCommand}" />
    </StackLayout>

</ContentPage>

```

Листинг В.25 – Файл App.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<Application xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

x:Class="BookStore.MAUIApp.App"
        xmlns:converters="clr-
namespace:BookStore.MAUIApp.Utils">
    <Application.Resources>
        <ResourceDictionary>
            <Style TargetType="Label">
                <Setter Property="TextColor" Value="Black" />
            </Style>

```

```

        <converters:IsEmptyConverter x:Key="IsEmptyConverter"
/>
        <converters:IsNotEmptyConverter
x:Key="IsNotEmptyConverter" />
        <converters:InverseBooleanConverter
x:Key="InverseBooleanConverter" />
        <converters:GenreToRussianConverter
x:Key="GenreToRussianConverter" />

    </ResourceDictionary>
</Application.Resources>
</Application>

```

Листинг В.26 – Файл AppShell.xaml

```

<?xml version="1.0" encoding="utf-8" ?>
<Shell xmlns="http://schemas.microsoft.com/dotnet/2021/maui"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:auth="clr-namespace:BookStore.MAUIApp.View.AuthView"
        xmlns:books="clr-namespace:BookStore.MAUIApp.View.BookView"
        xmlns:booksAdd="clr-
namespace:BookStore.MAUIApp.Views.BookView"
        xmlns:cart="clr-namespace:BookStore.MAUIApp.Views.CartView"
        xmlns:wishlist="clr-namespace:BookStore.MAUIApp.Views"
        xmlns:bookDetail="clr-
namespace:BookStore.MAUIApp.Views.BookView"
        xmlns:profile="clr-
namespace:BookStore.MAUIApp.Views.ProfileView"
        xmlns:questions="clr-
namespace:BookStore.MAUIApp.Views.QuestionsView"
        xmlns:order="clr-
namespace:BookStore.MAUIApp.Views.OrderView"
        xmlns:review="clr-
namespace:BookStore.MAUIApp.Views.ReviewView"
        xmlns:statistic="clr-
namespace:BookStore.MAUIApp.Views.StatisticView"
        x:Class="BookStore.MAUIApp.AppShell"
        FlyoutBehavior="Disabled">

    <!-- Роутинг -->
    <ShellContent Route="LoginPage" ContentTemplate="{DataTemplate
auth:LoginPage}" />
    <ShellContent Route="RegisterPage"
ContentTemplate="{DataTemplate auth:RegisterPage}" />
    <ShellContent Route="BooksPage" ContentTemplate="{DataTemplate
books:BooksPage}" />
    <ShellContent Route="CartPage" ContentTemplate="{DataTemplate
cart:CartPage}" />
    <ShellContent Route="WishlistPage"
ContentTemplate="{DataTemplate wishlist:WishlistPage}" />
    <ShellContent Route="BookDetailsPage"
ContentTemplate="{DataTemplate bookDetail:BookDetailsPage}" />

```

```

        <ShellContent Route="ProfilePage"
ContentTemplate="{DataTemplate profile:ProfilePage}" />
        <ShellContent Route="UserQuestionsPage"
ContentTemplate="{DataTemplate questions:UserQuestionsPage}" />
        <ShellContent Route="AskQuestionPage"
ContentTemplate="{DataTemplate questions:AskQuestionPage}" />
        <ShellContent Route="OrdersPage"
ContentTemplate="{DataTemplate order:OrderPage}" />
        <ShellContent Route="AddReviewPage"
ContentTemplate="{DataTemplate review:AddReviewPage}" />
        <ShellContent Route="ReviewsPage"
ContentTemplate="{DataTemplate review:ReviewsPage}" />
        <ShellContent Route="ManagerProfilePage"
ContentTemplate="{DataTemplate profile:ManagerProfilePage}" />
        <ShellContent Route="ManageReviewsPage"
ContentTemplate="{DataTemplate review:ManageReviewsPage}" />
        <ShellContent Route="ManageQuestionsPage"
ContentTemplate="{DataTemplate questions:ManageQuestionsPage}" />
        <ShellContent Route="EditReviewPage"
ContentTemplate="{DataTemplate review:EditReviewPage}" />
        <ShellContent Route="AddBookPage"
ContentTemplate="{DataTemplate booksAdd:AddBookPage}" />
        <ShellContent Route="EditBookPage"
ContentTemplate="{DataTemplate booksAdd:EditBookPage}" />
        <ShellContent Route="StatisticsPage"
ContentTemplate="{DataTemplate statistic:StatisticsPage}" />

</Shell>

```

Листинг В.27 – Файл MauiProgram.cs

```

using BookStore.MAUIApp.Services.AuthService;
using BookStore.MAUIApp.Services.BookService;
using BookStore.MAUIApp.Services.CartService;
using BookStore.MAUIApp.Services.OrderService;
using BookStore.MAUIApp.Services.QuestionService;
using BookStore.MAUIApp.Services.ReviewService;
using BookStore.MAUIApp.Services.Wishlist;
using BookStore.MAUIApp.View.AuthView;
using BookStore.MAUIApp.View.BookView;
using BookStore.MAUIApp.ViewModel.AskQuestionsViewModel;
using BookStore.MAUIApp.ViewModel.AuthViewModel;
using BookStore.MAUIApp.ViewModel.BookCatalogViewModel;
using BookStore.MAUIApp.ViewModel.CartViewModel;
using BookStore.MAUIApp.ViewModel.OrderViewModel;
using BookStore.MAUIApp.ViewModel.ProfileViewModel;
using BookStore.MAUIApp.ViewModel.QuestionsViewModel;
using BookStore.MAUIApp.ViewModel.ReviewViewModel;
using BookStore.MAUIApp.ViewModel.UserQuestionsViewModel;
using BookStore.MAUIApp.ViewModel.WishListViewModel;
using BookStore.MAUIApp.Views;
using BookStore.MAUIApp.Views.BookView;

```

```

using BookStore.MAUIApp.Views.CartView;
using BookStore.MAUIApp.Views.OrderView;
using BookStore.MAUIApp.Views.ProfileView;
using BookStore.MAUIApp.Views.QuestionsView;
using BookStore.MAUIApp.Views.ReviewView;
using BookStore.MAUIApp.Views.StatisticView;
using CommunityToolkit.Maui;
using Microsoft.Extensions.Logging;

namespace BookStore.MAUIApp
{
    public static class MauiProgram
    {
        public static MauiApp CreateMauiApp()
        {
            var builder = MauiApp.CreateBuilder();
            builder
                .UseMauiApp<App>()
                .UseMauiCommunityToolkit()
                .ConfigureFonts(fonts =>
                {
                    fonts.AddFont("OpenSans-Regular.ttf",
"OpenSansRegular");
                    fonts.AddFont("OpenSans-Semibold.ttf",
"OpenSansSemibold");
                });

            builder.Services.AddSingleton<IAuthService,
AuthService>();
            builder.Services.AddSingleton<IBookService,
BookService>();
            builder.Services.AddSingleton<ICartService,
CartService>();
            builder.Services.AddSingleton<IWishlistService,
WishlistService>();
            builder.Services.AddSingleton<IOrderService,
OrderService>();
            builder.Services.AddSingleton<IQuestionService,
QuestionService>();
            builder.Services.AddSingleton<IReviewService,
ReviewService>();

            builder.Services.AddTransient<LoginViewModel>();
            builder.Services.AddTransient<RegisterViewModel>();
            builder.Services.AddTransient<BooksViewModel>();
            builder.Services.AddTransient<CartViewModel>();
            builder.Services.AddTransient<WishlistViewModel>();
            builder.Services.AddTransient<BookDetailsViewModel>();
            builder.Services.AddTransient<ProfileViewModel>();

            builder.Services.AddTransient<UserQuestionsViewModel>();

```

```

        builder.Services.AddTransient<AskQuestionViewModel>();
        builder.Services.AddTransient<OrderPageViewModel>();
        builder.Services.AddTransient<AddReviewViewModel>();
        builder.Services.AddTransient<ReviewsViewModel>();

builder.Services.AddTransient<ManageReviewsViewModel>();

builder.Services.AddTransient<ManagerProfileViewModel>();
        builder.Services.AddTransient<EditReviewViewModel>();

builder.Services.AddTransient<ManageQuestionsViewModel>();
        builder.Services.AddTransient<AddBookViewModel>();
        builder.Services.AddTransient<EditBookViewModel>();

        builder.Services.AddTransient<LoginPage>();
        builder.Services.AddTransient<RegisterPage>();
        builder.Services.AddTransient<BooksPage>();
        builder.Services.AddTransient<CartPage>();
        builder.Services.AddTransient<WishlistPage>();
        builder.Services.AddTransient<OrderPopup>();
        builder.Services.AddTransient<BookDetailsPage>();
        builder.Services.AddTransient<ProfilePage>();
        builder.Services.AddTransient<AskQuestionPage>();
        builder.Services.AddTransient<UserQuestionsPage>();
        builder.Services.AddTransient<OrderPage>();
        builder.Services.AddTransient<AddReviewPage>();
        builder.Services.AddTransient<ReviewsPage>();
        builder.Services.AddTransient<ManagerProfilePage>();
        builder.Services.AddTransient<ManageReviewsPage>();
        builder.Services.AddTransient<EditReviewPage>();
        builder.Services.AddTransient<ManageQuestionsPage>();
        builder.Services.AddTransient<AddBookPage>();
        builder.Services.AddTransient<EditBookPage>();
        builder.Services.AddTransient<StatisticsPage>();

        builder.Services.AddHttpClient<IAuthService,
AuthService>();
        builder.Services.AddHttpClient<IBookService,
BookService>();
        builder.Services.AddHttpClient<ICartService,
CartService>();
        builder.Services.AddHttpClient<IWishlistService,
WishlistService>();
        builder.Services.AddHttpClient<IQuestionService,
QuestionService>();
        builder.Services.AddHttpClient<IOrderService,
OrderService>();
        builder.Services.AddHttpClient<IReviewService,
ReviewService>();

#if DEBUG

```

```

        builder.Logging.AddDebug();
    #endif

    return builder.Build();
}
}
}

```

Листинг В.28 – CartPage.xaml

```

<ContentPage xmlns="http://schemas.microsoft.com/dotnet/2021/maui"

xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
        xmlns:viewModel="clr-
namespace:BookStore.MAUIApp.ViewModel.CartViewModel"
        x:Class="BookStore.MAUIApp.Views.CartView.CartPage"
        Title="Корзина">

    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="Auto" />
            <RowDefinition Height="60" />
        </Grid.RowDefinitions>

        <!-- Основной контент -->
        <StackLayout Padding="10" Grid.Row="0">
            <!-- Общая сумма, только если корзина не пуста -->
            <StackLayout IsVisible="{Binding CartItems.Count,
Converter={StaticResource IsNotEmptyConverter}}">
                <Label FontSize="Large" Margin="0,10,0,0">
                    <Label.FormattedText>
                        <FormattedString>
                            <Span Text="Общая сумма: " />
                            <Span Text="{Binding TotalAmount,
StringFormat='{0:C}}'" />
                        </FormattedString>
                    </Label.FormattedText>
                </Label>
            </StackLayout>

            <!-- Если корзина пуста -->
            <Label Text="Корзина пуста"
                FontSize="Large"
                HorizontalOptions="Center"
                VerticalOptions="Center"
                IsVisible="{Binding CartItems.Count,
Converter={StaticResource IsEmptyConverter}}" />

            <!-- Если корзина не пуста, показываем список -->
            <CollectionView ItemsSource="{Binding CartItems}"
                IsVisible="{Binding CartItems.Count,
Converter={StaticResource IsNotEmptyConverter}}">

```

```

        <CollectionView.ItemTemplate>
            <DataTemplate>
                <StackLayout Padding="10"
BackgroundColor="White" Margin="0,5">
                    <!-- Информация о книге -->
                    <Grid ColumnDefinitions="Auto,*"
Padding="5">
                        <Image Source="{Binding
Book.ImagePath}" HeightRequest="100" WidthRequest="70"
Aspect="AspectFit" />

                        <StackLayout Grid.Column="1"
Padding="10,0,0,0">
                            <Label Text="{Binding
Book.Title}" FontSize="Medium" FontAttributes="Bold" />
                            <Label Text="Автор: "
FontSize="Small" TextColor="Gray" />
                            <Label Text="{Binding
Book.Author}" FontSize="Small" />
                            <Label Text="Цена: "
FontSize="Small" TextColor="Gray" />
                            <Label Text="{Binding
Book.Price, StringFormat='{{0:N2}} ₸'" FontSize="Small" />
                        </StackLayout>
                    </Grid>

                    <!-- Управление количеством и
удалением -->
                    <StackLayout Orientation="Horizontal"
Spacing="10" HorizontalOptions="Start" Margin="10,5,0,0">
                        <Button Text="-"
Command="{Binding
Source={RelativeSource AncestorType={x:Type
viewModel:CartViewModel}}}, Path=DecreaseQuantityCommand}"
CommandParameter="{Binding}"
WidthRequest="40"
HeightRequest="40" />

                        <Label Text="{Binding Quantity}"
FontSize="Medium" VerticalOptions="Center" />

                        <Button Text="+"
Command="{Binding
Source={RelativeSource AncestorType={x:Type
viewModel:CartViewModel}}}, Path=IncreaseQuantityCommand}"
CommandParameter="{Binding}"
WidthRequest="40"
HeightRequest="40" />

                        <Button Text="Удалить"

```

```

Command="{Binding
Source={RelativeSource AncestorType={x:Type
viewModel:CartViewModel}}}, Path=RemoveFromCartCommand}"

CommandParameter="{Binding}"

BackgroundColor="Red"
TextColor="White"
HeightRequest="40"
WidthRequest="90" />
</StackLayout>
</StackLayout>
</DataTemplate>
</CollectionView.ItemTemplate>
</CollectionView>

</StackLayout>

<!-- Кнопки управления (Очистить корзину, Оформить заказ)
-->
<StackLayout Grid.Row="1"
Margin="10,0,10,10"
Spacing="10"
IsVisible="{Binding CartItems.Count,
Converter={StaticResource IsNotEmptyConverter}}">
<Button Text="Очистить корзину"
Command="{Binding ClearCartCommand}"
HeightRequest="50"
HorizontalOptions="FillAndExpand" />
<Button Text="Оформить заказ"
Command="{Binding ShowOrderPopupCommand}"
HeightRequest="50"
HorizontalOptions="FillAndExpand" />
</StackLayout>

<!-- Нижнее меню с иконками -->
<Grid Grid.Row="2" BackgroundColor="LightGray"
Padding="10">
<Grid.ColumnDefinitions>
<ColumnDefinition Width="*" />
<ColumnDefinition Width="*" />
<ColumnDefinition Width="*" />
<ColumnDefinition Width="*" />
</Grid.ColumnDefinitions>

<ImageButton Grid.Column="0" Source="Icons/home.png"
WidthRequest="30" HeightRequest="30" Command="{Binding
NavigateToCatalogCommand}" />
<ImageButton Grid.Column="1"
Source="Icons/shopping_cart.png" WidthRequest="30"
HeightRequest="30" Command="{Binding NavigateToCartCommand}" />

```

```
        <ImageButton Grid.Column="2"
Source="Icons/wishlist.png" WidthRequest="30" HeightRequest="30"
Command="{Binding NavigateToWishlistCommand}" />
        <ImageButton Grid.Column="3" Source="Icons/user.png"
WidthRequest="30" HeightRequest="30" Command="{Binding
NavigateToProfileCommand}" />
    </Grid>
</Grid>
</ContentPage>
```

Приложение Г

(рекомендуемое)

Результаты тестирования серверной части

The screenshot shows a mobile application interface for adding a new book. At the top, there is a dark header with the text "Добавить книгу". Below it, a section titled "Добавление новой книги" contains four input fields. The first field, labeled "Война_Мир_название", contains the text "Война_Мир_название". The second field contains "Толстой", the third contains "Роман", and the fourth contains "Детская литература". A white error dialog box is displayed in the center, with the title "Ошибка" and the message "Название книги не должно содержать специальных символов." and an "Ок" button. Below the input fields, there are two more fields containing "123", a purple button labeled "Загрузить изображение", and at the bottom, two purple buttons labeled "Добавить книгу" and "Отмена".

Рисунок Г.1 – Ввод название с недопустимыми символами

The screenshot shows the same mobile application interface as Figure G.1. The first input field, labeled "Война_Мир_название", now contains a long string of characters: "кАбвгдежзикАбвгдежзикАбвгдежзикАбвгдежзик1". The error dialog box is still present, but the message has changed to "Название книги должно не больше 100 символов." with an "Ок" button. The other elements of the interface, including the other input fields, the image upload button, and the bottom navigation buttons, remain the same.

Рисунок Г.2 – Ввод названия, превышающего 100 символов

The screenshot shows a mobile application interface for adding a new book. At the top, there is a dark blue header with the text "Добавить книгу". Below it, a light gray section is titled "Добавление новой книги". This section contains four input fields: the first has "А", the second "Толстой", the third "Роман", and the fourth "Детская литература". Below these fields is a white error dialog box with the title "Ошибка" and the message "Название книги должно не меньше 2 символов." (Book name must be at least 2 symbols). The dialog has an "Ок" button. Below the error box, there are two more input fields, both containing "123". Below these is a blue button labeled "Загрузить изображение". At the bottom of the form are two blue buttons: "Добавить книгу" and "Отмена".

Рисунок Г.3 – Ввод названия, длина которого меньше двух символов

This screenshot shows the same mobile application interface as Figure G.3, but with a successful outcome. The error dialog is replaced by a white success dialog box with the title "Успех" and the message "Книга успешно добавлена." (Book successfully added). The dialog has an "Ок" button. The input fields now contain "Ад", "Толстой", "Роман", and "Детская литература". The "123" fields and the "Загрузить изображение" button remain the same. The "Добавить книгу" and "Отмена" buttons are still at the bottom.

Рисунок Г.4 – Ввод названия минимально допустимого размера

Добавить книгу

Добавление новой книги

зикАбвгдежзикАбвгдежзикАбвгдежзикАбвгдежзик

Толстой

Роман

Детская литература

Успех

Книга успешно добавлена.

Ок

123

123

Загрузить изображение

Добавить книгу Отмена

Рисунок Г.5 – Ввод названия максимального допустимого размера

Приложение Д
Исходный код функций тестирования
(рекомендуемое)

Листинг Д.1 – Файл BooksControllerTests.cs

```

using BookStore.Application.Interfaces;
using BookStore.Domain.DTOs;
using BookStore.Domain.Entities;
using BookStore.Domain.Enums;
using BookStoreAPI.Controllers;
using Microsoft.AspNetCore.Mvc;
using Moq;

public class BooksControllerTests
{
    private readonly Mock<IBookRepository> _bookRepoMock = new();
    private readonly Mock<IOrderRepository> _orderRepoMock =
new();
    private readonly BooksController _controller;

    public BooksControllerTests()
    {
        _controller = new BooksController(_bookRepoMock.Object,
_orderRepoMock.Object);
    }

    [Fact]
    public async Task GetAll_ReturnsOkResult_WithListOfBooks()
    {
        var books = new List<Book> { new Book { Id = 1, Title =
"Test" } };
        _bookRepoMock.Setup(r =>
r.GetAllAsync()).ReturnsAsync(books);

        var result = await _controller.GetAll();

        var okResult = Assert.IsType<OkObjectResult>(result);
        var returnBooks =
Assert.IsAssignableFrom<IEnumerable<Book>>(okResult.Value);
        Assert.Single(returnBooks);
    }

    [Fact]
    public async Task GetById_BookExists_ReturnsOk()
    {
        var book = new Book { Id = 1, Title = "Test" };
        _bookRepoMock.Setup(r =>
r.GetByIdAsync(1)).ReturnsAsync(book);

        var result = await _controller.GetById(1);

        var okResult = Assert.IsType<OkObjectResult>(result);
        Assert.Equal(book, okResult.Value);
    }

    [Fact]

```

```

    public async Task GetById_BookNotFound_ReturnsNotFound()
    {
        _bookRepoMock.Setup(r =>
r.GetByIdAsync(99)).ReturnsAsync((Book?)null);

        var result = await _controller.GetById(99);

        Assert.IsType<NotFoundObjectResult>(result);
    }

    [Fact]
    public async Task Search_NoBooksFound_ReturnsNotFound()
    {
        _bookRepoMock.Setup(r => r.SearchAsync(null, null, null,
null)).ReturnsAsync(new List<Book>());

        var result = await _controller.Search();

        Assert.IsType<NotFoundObjectResult>(result);
    }

    [Fact]
    public async Task Create_ValidBook_ReturnsCreatedAtAction()
    {
        var dto = new BookDto
        {
            Title = "New",
            Author = "Author",
            Genre = BookGenre.Fantasy,
            Category = BookCategory.PopularScience,
            Price = 10,
            QuantityInStock = 5,
            Year = 2020,
            Description = "Test",
            ImagePath = "cover.jpg"
        };

        var result = await _controller.Create(dto);

        var createdResult =
Assert.IsType<CreatedAtActionResult>(result);
        var book = Assert.IsType<Book>(createdResult.Value);
        Assert.Equal("New", book.Title);
    }

    [Fact]
    public async Task Update_BookExists_ReturnsNoContent()
    {
        var existingBook = new Book { Id = 1 };
        _bookRepoMock.Setup(r =>
r.GetByIdAsync(1)).ReturnsAsync(existingBook);

```

```

var dto = new BookDto
{
    Title = "Updated",
    Author = "Author",
    Genre = BookGenre.Historical,
    Category = BookCategory.Art,
    Price = 15,
    QuantityInStock = 3,
    Year = 2021,
    Description = "Updated",
    ImagePath = "new.jpg"
};

var result = await _controller.Update(1, dto);

Assert.IsType<NoContentResult>(result);
}

[Fact]
public async Task Update_BookNotFound_ReturnsNotFound()
{
    _bookRepoMock.Setup(r =>
r.GetByIdAsync(1)).ReturnsAsync((Book?)null);
    var dto = new BookDto();

    var result = await _controller.Update(1, dto);

    Assert.IsType<NotFoundObjectResult>(result);
}

[Fact]
public async Task Delete_BookExists_ReturnsNoContent()
{
    var book = new Book { Id = 1 };
    _bookRepoMock.Setup(r =>
r.GetByIdAsync(1)).ReturnsAsync(book);

    var result = await _controller.Delete(1);

    Assert.IsType<NoContentResult>(result);
}

[Fact]
public async Task Delete_BookNotFound_ReturnsNotFound()
{
    _bookRepoMock.Setup(r =>
r.GetByIdAsync(1)).ReturnsAsync((Book?)null);

    var result = await _controller.Delete(1);

    Assert.IsType<NotFoundObjectResult>(result);
}

```

```
[Fact]
public async Task GetStatistics_ReturnsOk_WithStatisticsDto()
{
    _orderRepoMock.Setup(r => r.GetAll()).ReturnsAsync(new
List<Order>());
    _bookRepoMock.Setup(r => r.GetAllAsync()).ReturnsAsync(new
List<Book>());

    var result = await _controller.GetStatistics();

    var okResult = Assert.IsType<OkObjectResult>(result);
    Assert.IsType<BookSalesStatisticsDto>(okResult.Value);
}
}
```