ТУТ ТИТУЛ

# Содержание

[ВВЕДЕНИЕ 4](#_Toc195469743)

[1 Аналитический обзор литературы 5](#_Toc195469744)

[1.1 Обзор существующих решений 5](#_Toc195469745)

[1.1.1 Hello 5](#_Toc195469746)

[1.1.2 Rentacars 6](#_Toc195469747)

[1.1.3 To-car 7](#_Toc195469748)

[1.1.4 Multimotors 8](#_Toc195469749)

[1.2 Выводы по разделу 8](#_Toc195469750)

[2 Анализ требований к программному средству и разработка функциональных требований 9](#_Toc195469751)

[2.1 Описание функциональности программного средства 9](#_Toc195469752)

[2.2. Спецификация функциональных требований 9](#_Toc195469753)

[2.2.1 Регистрация 9](#_Toc195469754)

[2.2.2 Авторизация 9](#_Toc195469755)

[2.2.3 Просмотр списка автомобилей 10](#_Toc195469756)

[2.2.4 Сравнение автомобилей 10](#_Toc195469757)

[2.2.5 Оформление автомобиля в прокат 10](#_Toc195469758)

[2.2.6 Расчёт стоимости аренды 10](#_Toc195469759)

[2.2.7 Просмотр истории аренды 10](#_Toc195469760)

[2.2.8 Получение уведомлений о состоянии аренды 11](#_Toc195469761)

[2.2.9 Добавление и удаление отзывов 11](#_Toc195469762)

[2.2.10 Добавление, удаление, изменение автомобилей 11](#_Toc195469763)

[2.2.11 Добавление, удаление, изменение тарифов 11](#_Toc195469764)

[2.2.12 Просмотр статистики по прокату автомобилей 12](#_Toc195469765)

[2.2.13 Модерация списка отзывов 12](#_Toc195469766)

[2.2.14 Управление клиентами 12](#_Toc195469767)

[2.2.15 Управление заказами 12](#_Toc195469768)

[2.3 Выводу по главе 12](#_Toc195469769)

[3 Проектирование программного средства 13](#_Toc195469770)

[3.1 Общая структура проекта 13](#_Toc195469771)

[3.1.1 Работа с данными 13](#_Toc195469772)

[3.1.2 Бизнес-логика 14](#_Toc195469773)

[3.1.3 Пользовательский интерфейс 14](#_Toc195469774)

[3.2 Сущности 15](#_Toc195469775)

[3.2.1 Пользователь 16](#_Toc195469776)

[3.2.2 Автомобиль 16](#_Toc195469777)

[3.2.3 Тариф 17](#_Toc195469778)

[3.2.4 Аренда 17](#_Toc195469779)

[3.2.5 Отзыв 18](#_Toc195469780)

[3.3 Выводы по главе 18](#_Toc195469781)

[4 Реализация программного средства 19](#_Toc195469782)

[4.1 Связи между слоями 19](#_Toc195469783)

[4.2 Подключение к базе данных 20](#_Toc195469784)

[4.3 Ресурсы, используемые в программном средстве 20](#_Toc195469785)

[4.4 Преобразование типов данных 21](#_Toc195469786)

[4.5 Обновление данных на страницах 22](#_Toc195469787)

[4.6 Реализация команд 23](#_Toc195469788)

[4.7 Страницы 23](#_Toc195469789)

[4.8 Обмен событиями, между страницами 23](#_Toc195469790)

[4.9 Проверка завершения аренд 24](#_Toc195469791)

[4.10 Выводы по главе 25](#_Toc195469792)

[5 Тестирование, проверка работоспособности и анализ полученных результатов 26](#_Toc195469793)

[5.1 Методика тестирования 26](#_Toc195469794)

[5.2 Тестирование форм ввода данных 26](#_Toc195469795)

[5.2.1 Регистрация 26](#_Toc195469796)

[5.2.2 Создание автомобиля 27](#_Toc195469797)

[5.3 Аренда 29](#_Toc195469798)

[5.4 Выводы по главе 30](#_Toc195469799)

[6 Руководство по установке и использованию 31](#_Toc195469800)

[6.1 Установка программного средства 31](#_Toc195469801)

[6.2 Использование программного средства 33](#_Toc195469802)

[6.2.1 Регистрация 33](#_Toc195469803)

[6.2.2 Авторизация 34](#_Toc195469804)

[6.2.3 Список автомобилей 34](#_Toc195469805)

[6.2.4 Страница клиентов 36](#_Toc195469806)

[6.2.5 Страница аренд 36](#_Toc195469807)

[6.2.6 Страница отзывов 37](#_Toc195469808)

[6.2.7 Страница сравнения 38](#_Toc195469809)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 39](#_Toc195469810)

[Приложение А 40](#_Toc195469811)

[Приложение Б 41](#_Toc195469812)

[Приложение В 42](#_Toc195469813)

[Приложение Г 43](#_Toc195469814)

[Приложение Д 44](#_Toc195469815)

# ВВЕДЕНИЕ

В современном мире онлайн-покупки книг становятся все более популярными, и многие пользователи стремятся найти удобные способы поиска и бронирования книг, чтобы не тратить время на походы в магазин или библиотеку. Однако не каждый человек может выбрать книгу, которая подходит именно ему, не имея подробной информации или рекомендаций. Также возникает проблема отсутствия интуитивного интерфейса для самостоятельного бронирования.

Целью курсового проектирования является разработка программного средства, которое позволяет пользователю выбрать книгу, ознакомиться с ее подробным описанием, прочитать рецензии и заказать книгу на удобную дату для получения. Этот процесс значительно упрощает поиск и покупку книг, а также способствует лучшему пониманию содержания произведений, не выходя из дома.

В основе проектирования сервиса лежат: использование паттерна MVVM, разделения программного средства на слои, использования языка разметки «XAML».

Основная часть пояснительной записки разбита на следующие главы:

– аналитический обзор литературы: выделить ключевые аспекты, которым должен обладать сервис для аренды автомобилей;

– анализ требований к программному средству и разработка функциональных требований: сформировать и описать требования к программному средству;

– проектирование программного средства: описать структуру программного средства, механизмы взаимодействий различных её частей;

– создание программного средства: описать процесс разработки программного средства, описать реализацию;

– тестирование, проверка работоспособности и анализ полученных результатов: сверить функционал программного средства с заданным ранее, протестировать его;

– руководство по установке и использованию: описать процесс установки, процесс взаимодействия пользователя с ним.

# 1 Аналитический обзор литературы

Одним из первых этапов в создании программного продукта является анализ прототипов и литературных источников. При изучении сайтов-прототипов было выявлено несколько экземпляров схожих по функциональности. Результат анализа нескольких из них представлен ниже.

## Аналог №1: «oz.by»

oz.by – ведущий белорусский интернет-магазин, предлагающий широкий ассортимент товаров, в том числе книги различных жанров и категорий. Сайт выделяется удобным и привлекательным интерфейсом с гармоничным сочетанием цветов.

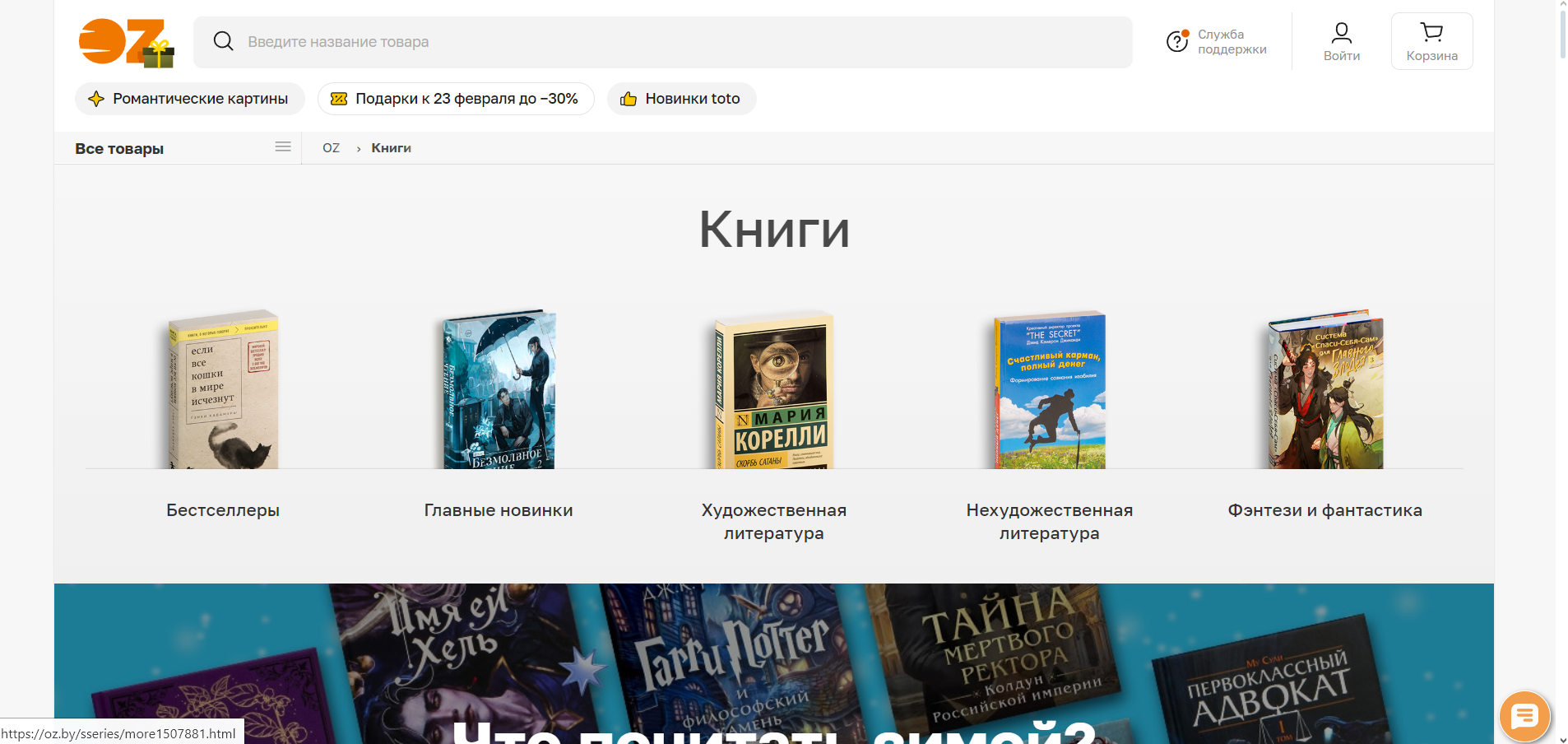


Рисунок 1.1 – Аналог №1 «oz.by»

В целом, oz.by эффективно реализует ключевые функции интернет-магазина книг:

1. Обширный ассортимент: Представлены как популярные бестселлеры, так и редкие издания. Доступна функция предзаказа, позволяющая приобрести книги, еще не поступившие в продажу.

2. Удобный поиск: Обеспечен быстрый и эффективный поиск по автору, названию, жанру и другим критериям. Результаты поиска можно сортировать по цене, рейтингу и другим параметрам.

3. Подробная информация о товаре: Каждая книга сопровождается детальным описанием, качественным изображением обложки и отзывами покупателей. Предоставляется возможность предварительного ознакомления с фрагментами книги.

4. Простое оформление заказа: Предлагаются различные варианты доставки и оплаты, включая курьерскую доставку, почтовую отправку и самовывоз из пунктов выдачи. Поддерживается использование промокодов, скидок и бонусных программ.

5. Служба поддержки: Предоставлен раздел с часто задаваемыми вопросами (FAQ) и возможность связи со службой поддержки по электронной почте или телефону.

Дополнительные полезные функции oz.by включают разделы с новинками и персональными рекомендациями, возможность подписки на новостную рассылку для получения информации о скидках и акциях, а также программы лояльности для постоянных клиентов.

### 1.2 Аналог №2: «chitatel.by»

Chitatel.by – белорусский интернет-магазин, специализирующийся на продаже книг различных жанров, включая научную, художественную, учебную и другие категории. Основная цель сайта – предоставление пользователям доступа к широкому выбору литературы.

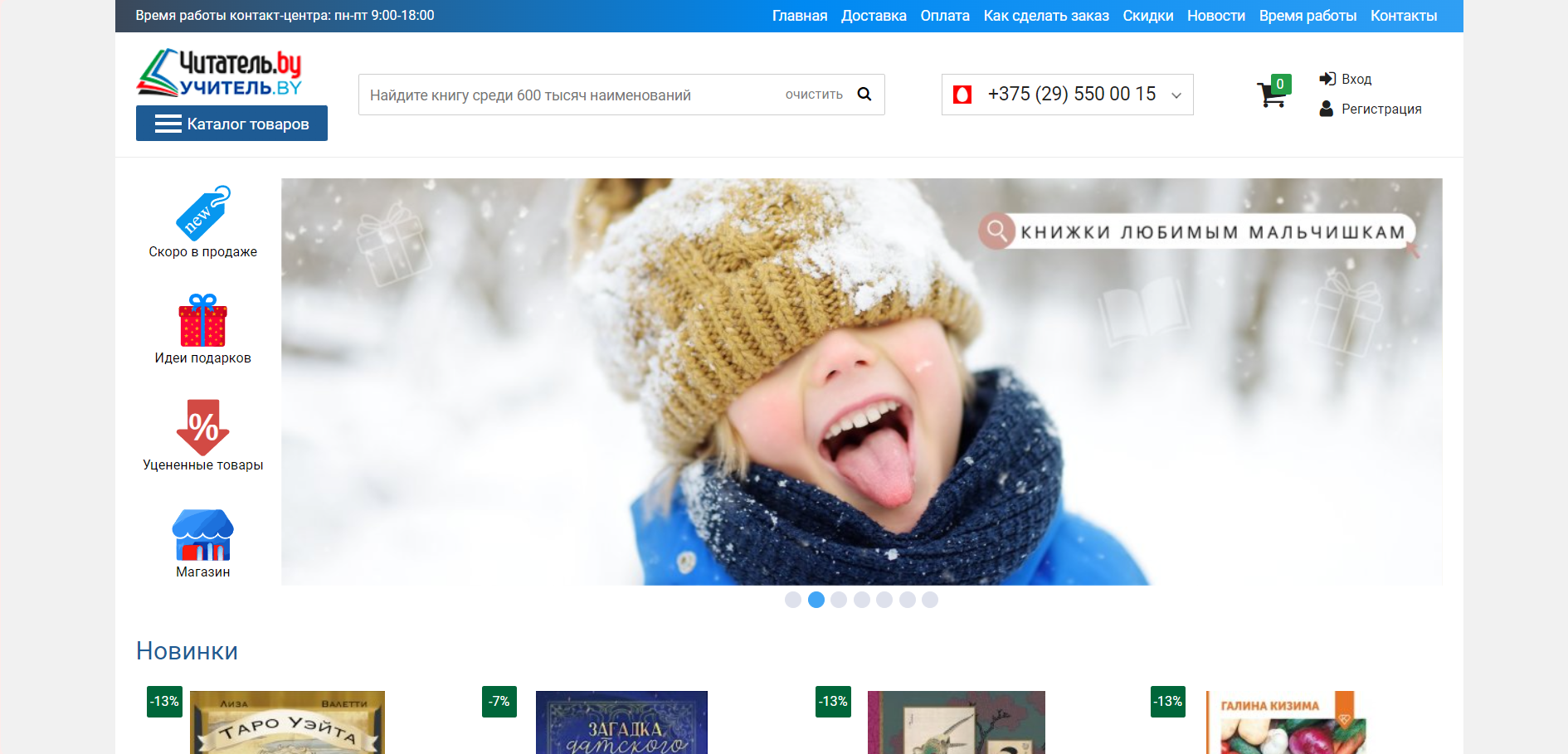


Рисунок 1.2 – Аналог №2 «chitatel.by»

На главной странице представлены основные категории книг, что облегчает навигацию по сайту. Пользователи могут также воспользоваться поиском для быстрого нахождения конкретных изданий.

При выборе книги предоставляется основная информация: название, автор, издательство и цена. Однако, в отличие от некоторых конкурентов, детальное описание и отзывы покупателей отсутствуют, что может затруднить процесс принятия решения о покупке. Тем не менее, отсутствие отвлекающих элементов делает страницу книги лаконичной и быстрой для просмотра.

Оформление заказа осуществляется через корзину. Доступны несколько способов доставки и оплаты, хотя их выбор может быть не столь обширным, как в других крупных интернет-магазинах. Стоит отметить, что процесс оформления заказа достаточно прост и интуитивно понятен. На данный момент на сайте не реализована поддержка промокодов и скидок.

Отсутствие программы лояльности для постоянных клиентов также является заметным упущением. Это может повлиять на удержание клиентов и стимулирование повторных покупок.

В целом, chitatel.by предоставляет базовый функционал для покупки книг онлайн. Хотя сайт и уступает некоторым конкурентам в плане детализации информации о товарах и расширенных возможностях оформления заказа, он обладает простой и понятной структурой. Для пользователей, ценящих скорость и простоту, chitatel.by может быть вполне подходящим вариантом. Развитие функциональности, особенно в части предоставления расширенной информации о книгах и внедрения системы лояльности, могло бы значительно повысить привлекательность данного интернет-магазина.

## Выводы по разделу

В результате анализа аналогичных онлайн-книжных магазинов были выделены ключевые функциональные возможности, неодходимые для обеспечения конкурентоспособности платформы. К ним относятся:

* Система фильтрации- наличие гибких фильтров(по жанру, наличию, цене);
* Функция добавления в корзину-возможность сохранить выбранные книги в корзину до момента оформления заказа обеспечивает гибкость в принятии решений и улучшает пользовательский опыт;
* Оформление заказа- простой и логический процесс покупки;

Ни на одном из сервисов не было возможности записаться на мероприятие, следует это добавить, чтобы выделиться на фоне конкурентов.

# 2 Анализ требований к программному средству и разработка функциональных требований

## 2.1 Описание функциональности программного средства

Функционал программного средства разделён на 2 роли: «Клиент» и «Администратор».

Клиент является конечным пользователем программного средства. Его основная цель — найти и заказать интересующую книгу. Для этого реализован функционал удобного поиска с использованием фильтров (по автору, жанру, году издания и т.д.), что позволяет быстро сориентироваться в каталоге.  
Пользователь может добавлять книги в корзину и оформлять заказы. Также доступна возможность записи на мероприятия (например, встречи с авторами или презентации книг), проводимые книжным магазином.

Администратор управляет всей системой. Его задача — поддерживать актуальность каталога книг, контролировать работу с клиентами, планировать и регистрировать мероприятия, а также администрировать заказы.

## 2.2. Спецификация функциональных требований

### 2.2.1 Поиск товаров

Пользователь должен иметь возможность выполнять поиск книг по различным критериям

### 2.2.2 Фильтрация и сортировка

Система должна позволять фильтровать список книг и сортировать по параметрам

### 2.2.4 Добавление книги в корзину

Пользователь может добавлять выбранные книги в корзину для последующего оформления заказа.

### 2.2.5 Оформление заказа

Пользователь может заказать выбранный товар, который находится в корзине.

### 2.2.6 Запись на мероприятия

Пользователь может записаться на доступные мероприятия, организуемые книжным магазином.

### 2.2.7 Управление каталогом

Администратор может добавлять, редактировать и удалять книги из каталога.

### 2.2.8 Управление заказами

Администратор просматривает, удаляет и меняет состояние заказа.

### 2.2.9 Управление клиентами

Доступ к просмотру информации о клиентах, возможность удалить или добавить клиента.

### 2.2.10 Планирование мероприятий

Администратор создает и редактирует мероприятия (имя, описание, место, дата), а также может удалить или добавить.

### 2.2.11 Вход в систему

Пользователь или администратор может войти в свой личный кабинет.

## 2.3 Выводу по главе

В данной главе были сформулированы основные функциональные требования к программному обеспечению, отражающие ключевые задачи, которые должны выполнять.

Всего было выделено 2 роли и 11 основных функциональных компонент.

UML-диаграмма вариантов использования, где описаны функциональные возможности и роли программного средства, представлена в приложении А.

# 3 Проектирование программного средства

## 3.1 Общая структура проекта

Проект состоит из двух основных компонентов, разделенных по принципу многоуровневой архитектуры (N-Layer Architecture), что обеспечивает удобство сопровождения и масштабирование приложения.



Рисунок 3.1 – Общая структура проекта

Проект «BookShop» представляет собой пользовательский интерфейс.

### 3.1.1 Работа с данными

Работа с данными в проекте реализована на уровне слоя доступа к данными(DAL) с применением паттерна «Repository», обеспечивающий разделение логики взаимодействия с базой данных от остального приложения. В качестве технологии доступа к данным используется ADO.NET c использованием классов SqlConnection, SqlCommand, SqlDataReader из пространства имен Microsoft.Data.SqlClient.

Для работы с сущностями, которые находятся в папке Models, реализованы репозитории, которые обеспечивает выполнение операций. Каждая операция выполняется посредством прямых SQL-запросов, что позволяет добиться высокой степени контроля над взаимодействием с базой данных. Данный подход позволяет добиться гибкости и высокой производительности, сохраняя при этом чёткую архитектурную структуру приложения.

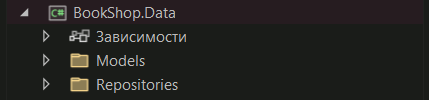


Рисунок 3.2 – Структура проекта «BookShop.Data»

### 3.1.2 Бизнес-логика

Бизнес-логика в проекте реализует поведение приложения, обеспечивая выполнение основных пользовательских сценариев: просмотр и фильтрация книг, оформление заказа, управление корзиной, регистрация и авторизация пользователей, а также участие в мероприятиях.

В проекте структура бизнес-логики организована по принципу MVVM (Model-View-ViewModel). Основной функционал сосредоточен в папке ViewModels, где каждый модуль (например, работа с книгами, заказами, пользователями) представлен отдельной моделью представления.

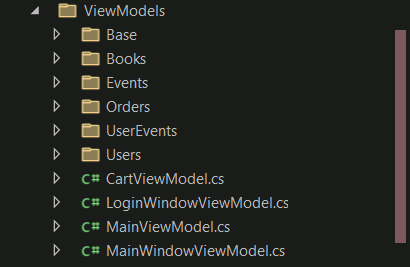


Рисунок 3.3 – Структура папки «ViewModels»

## 3.1.3 Пользовательский интерфейс

Пользовательский интерфейс (UI) является важной составляющей проекта, обеспечивая удобное и интуитивно понятное взаимодействие пользователя с программным средством. В данном проекте реализация UI выполнена с использованием технологии **WPF (Windows Presentation Foundation)** и архитектурного паттерна **MVVM (Model-View-ViewModel)**, что способствует разделению логики отображения, данных и бизнес-процессов.

Основные визуальные компоненты располагаются в папке Views, где каждая форма или окно соответствует определенной задаче пользователя:

* отображение списка книг;
* оформление заказа;
* работа с корзиной и заказами;
* управление профилем и просмотр мероприятий.

Визуальная часть интерфейса стилизована при помощи ресурсов, хранящихся в папках «Themes» и «Resources». Здесь размещены словари стилей, шаблоны элементов управления и другие графические элементы, обеспечивающие единообразие и эстетичность интерфейса. Для представления обложек книг и других изображений используется папка «Images».

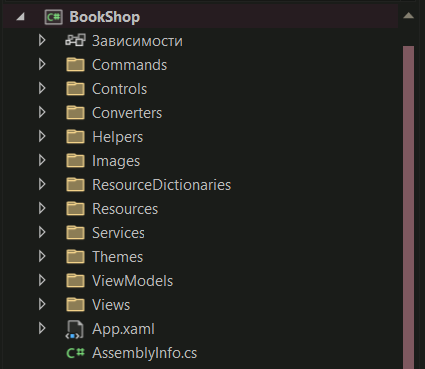


Рисунок 3.4 – Структура проекта «BookShop»

Файл «App.xaml» («App.xaml.cs») является входной точкой всего решения.

## 3.2 Сущности

Сущности проекта отражают основные объекты предметной области интернет-магазина книг. Каждая сущность представляет собой C#-класс, расположенный в папке «Models». Они служат для хранения и передачи данных между слоями архитектуры: пользовательским интерфейсом, бизнес-логикой и уровнем доступа к данным.

Все сущности наследуются от абстрактного класса «BaseEntity», расположенного в папке «Base». Он содержит общее свойство: int Id — уникальный идентификатор сущности

### 3.2.1 Пользователь

Пользователь – основная сущность в любом программном средстве.

Таблица 3.1 – Описание сущности пользователя

|  |  |  |
| --- | --- | --- |
| Поле | Ограничения целостности | Описание |
| Id | PRIMARY KEY  AUTOINCREMENT | Идентификатор пользователя |
| Login | TEXT NOT NULL | Логин пользователя |
| PasswordHash | TEXT NOT NULL | Хэш пароля |
| FirstName | TEXT NOT NULL | Имя пользователя |
| LastName | TEXT NOT NULL | Фамилия пользователя |
| Phone | TEXT NOT NULL | Номер телефона |
| Email | TEXT NOT NULL | Адрес электронной почты |
| Adress | TEXT NOT NULL | Почтовый адрес |
| IsAdmin | TEXT NOT NULL | Признак, является ли пользователь администратором |

### 3.2.2 Книга

Книга – вторая по важность сущность в программном средстве.

Таблица 3.2 – Описание сущности книги

|  |  |  |
| --- | --- | --- |
| Поле | Ограничения целостности | Описание |
| Id | PRIMARY KEY AUTOINCREMENT | Идентификатор книги |
| Name | TEXT NOT NULL | Название книги |
| Author | TEXT NOT NULL | Автор книги |
| Genre | INTEGER NOT NULL | Жанр книги |
| Price | TEXT NOT NULL | Стоимость книги |
| Image | BLOB NOT NULL | Фото автомобиля, массив байт |
| Available | INTEGER NOT NULL | Флаг, указывающий, доступна ли книга для заказа |

### 3.2.3 Order- заказ пользователя

Таблица 3.3 – Описание сущности заказа пользователя

|  |  |  |
| --- | --- | --- |
| Поле | Ограничения целостности | Описание |
| UserId | PRIMARY KEY AUTOINCREMENT | Идентификатор пользователя, оформившего заказ |
| User | Навигационное свойство (в коде), **в БД не хранится как отдельное поле** | Связанный объект пользователя |
| Date | DATETIME NOT NULL | Дата и время выполнения |
| TotalPrice | REAL NOT NULL | Общая сумма заказа |
| Status | TEXT NOT NULL | Текущий статус заказа |

### 3.2.4 OrderItem- элемент заказа

Таблица 3.4 – Описание сущности элемента заказа

|  |  |  |
| --- | --- | --- |
| Поле | Ограничения целостности | Описание |
| OrderId | INTEGER NOT NULL FORGEIGN KEY  REFERENCES | Идентификатор заказа, к которому относится позиция |
| Order | Навигационное свойство, **в БД не хранится напрямую** | Связанный объект заказа |
| BookId | INTEGER NOT NULL FOREIGN KEY REFERENCES Book(Id) | Идентификатор книги, включенной в заказ |
| Book | Навигационное свойство, **в БД не хранится напрямую** | Связанный объект книги |
| Quantity | INTEGER NOT NULL | Количество единиц книги в заказе |

### 3.2.5 Event

Таблица 3.5 – Описание сущности отзыва

|  |  |  |
| --- | --- | --- |
| Поле | Ограничения целостности | Описание |
| Name | TEXT NOT NULL | Название мероприятия |
| Description | TEXT | Описание мероприятия, необязательное поле |
| Date | DATETIME NOT NULL | Дата и время проведения мероприятия |
| Location | TEXT NOT NULL | Место проведения мероприятия |

### 3.2.6 UserEvent

Таблица 3.5 – Описание сущности отзыва

|  |  |  |
| --- | --- | --- |
| Поле | Ограничения целостности | Описание |
| UserId | INTEGER NOT NULL, FOREIGN KEY REFERENCES Users(Id) | Идентификатор пользователя |
| User | REFERENCE, NOT NULL (на уровне модели) | Объект пользователя, связанный с участием |
| EventId | INTEGER NOT NULL, FOREIGN KEY REFERENCES Events(Id) | Идентификатор мероприятия |
| Event | REFERENCE, NOT NULL (на уровне модели) | Объект мероприятия, в котором участвует пользователь |

# 3.3 Выводы по главе

В данной главе были:

* определена общая структура решения;
* определены структуры проектов;
* определён набор сущностей, представленных в программном средстве;
* определены поля сущностей, их типы данных и ограничения;
* определены связи между сущностями.

Всего в программном средстве будет 7 сущностей. Программное средство представлено двумя проектами.

Логическая схема базы данных программного средства представлена в приложении Б.

# 4 Реализация программного средства

## 4.1 Связи между слоями

В архитектуре проекта используется многоуровневый подход (N-Layer Architecture), обеспечивающий разделение ответственности и улучшенную сопровождаемость. Основные уровни включают:

* UI (пользовательский интерфейс) — отображение и взаимодействие с пользователем (папка Views, ViewModels).
* BLL (бизнес-логика) — обработка команд, навигация, принятие решений (например, Services, Commands).
* DAL (уровень доступа к данным) — взаимодействие с базой данных (папка Repositories, Models).

Структура DAL В проекте имеется множество репозиториев, реализующих интерфейсы доступа к конкретным сущностям:

* IBookRepository / BookRepository;
* IUserRepository / UserRepository;
* IEventRepository / EventRepository;
* IOrderRepository / OrderRepository;
* IOrderItemRepository / OrderItemRepository;
* IUserEventRepository / UserEventRepository;
* ReferencesRepository / IReferencesRepository;

Каждый репозиторий реализует интерфейс IRepository, обобщающий базовые операции.

using BookShop.Data.Models.Base;

namespace BookShop.Data.Repositories;

public interface IRepository<T> where T : BaseEntity

{

List<T> GetAll();

T? GetById(int id);

void Add(T entity);

void Update(T entity);

void Delete(T entity);

}

Листинг 4.1 – Реализация интерфейса базового репозитория

## 4.2 Подключение к базе данных

В проекте BookShop подключение к базе данных реализовано вручную через ADO.NET, без использования ORM-фреймворков вроде Entity Framework Core. Вся работа с данными сосредоточена в слое DAL (Data Access Layer), в частности — в классе BookRepository, находящемся в папке Repositories.

Подключение к базе данных происходит через строку подключения (\_connectionString), которая передаётся в BookRepository. На листинге 4.2 представлено подключение к базе данных.

private readonly string \_connectionString;

public BookRepository(string connectionString)

{

\_connectionString = connectionString;

}

private SqlConnection CreateConnection()

{

return new SqlConnection(\_connectionString);

}

Листинг 4.2 – Подключение к базе данных

Именно здесь задаётся строка подключения. Сама база данных расположена по пути «"C:\Program Files\Microsoft SQL Server\MSSQL16.SQLEXPRESS\MSSQL\DATA\BookShop.mdf"».

## 4.3 Ресурсы, используемые в программном средстве

В программном средстве BookShop используются различные ресурсы, обеспечивающие полноценную работу интерфейса, взаимодействие с пользователем и визуальное оформление. Ниже приведены основные категории ресурсов, задействованных в приложении:

* Графические ресурсы (Images);
* Ресурсы локализации (Resources);
* Темы и стили (Themes);
* Конвертеры и вспомогательные классы(Converters);
* Файлы конфигурации(App.xaml);

## 4.4 Преобразование типов данных

Преобразователи (Converters) реализованы с использованием интерфейса IValueConverter, предоставляемого платформой WPF. Эти классы служат для преобразования данных между моделью и представлением, обеспечивая, например:

* отображение логических значений в виде текста или изображения;
* форматирование даты;
* изменение цвета или стиля элементов на основе данных;
* отображение наличия товара.

На листинге 4.4 представлен конвертер, который преобразует логическое значение в значение перечисления Visiblity, применяемого в WPF-интерфейсе.

public class BooleanToVisibilityConverter : IValueConverter

{

public object? Convert(object? value, Type targetType, object? parameter, CultureInfo culture)

{

if (value is bool booleanValue)

{

return booleanValue ? Visibility.Visible : Visibility.Collapsed;

}

return Visibility.Collapsed;

}

public object? ConvertBack(object? value, Type targetType, object? parameter, CultureInfo culture)

{

throw new NotImplementedException();

}

}

Листинг 4.4 – Конвертер из логического значения в значение перечисления Visiblity

После создания конвертера необходимо его объявить, объявление конвертера представлено в листинге 4.5.

<Window xmlns:converters="clr-namespace:BookShop.Converters" ...> <Window.Resources> <converters:BoolToVisibilityConverter x:Key="BoolToVisibilityConverter" /> </Window.Resources>

<Grid>

Листинг 4.5 – Объявление конвертера в файле «MainView.xaml»

Затем, при необходимости, конвертер подключается при привязке.

<TextBlock Text="Текст"

Visibility="{Binding IsVisible, Converter={StaticResource BoolToVisibilityConverter}}" />

</Grid>

</Window>

Листинг 4.6 – Подключение конвертера

## 4.5 Обновление данных на страницах

Обновление данных на страницах приложения реализовано с использованием архитектуры MVVM и сервиса навигации INavigationService. Главное окно содержит привязку свойства CurrentViewModel к элементу ContentControl, что позволяет динамически отображать соответствующий View на основе текущей модели представления.

Переключение между страницами выполняется с помощью команд (RelayCommand), каждая из которых вызывает асинхронную навигацию к нужной ViewModel, представлено на листинге 4.7.

public RelayCommand OpenCartCommand => new RelayCommand(async obj => { if (CurrentViewModel is not CartViewModel) { var vm = await \_navigationService.NavigateToAsync<CartViewModel>(); CurrentViewModel = vm; } });

Листинг 4.7 – Переключение страниц в MainWindowViewModel

При успешной навигации сервис INavigationService возвращает экземпляр нужной ViewModel, который назначается в свойство CurrentViewModel. Благодаря этому, отображаемый контент в ContentControl автоматически обновляется, поскольку в ресурсах окна (Window.Resources) заранее настроены соответствующие DataTemplate для каждой ViewModel.

Для реализации механизма обновления данных и уведомления интерфейса об изменениях используется интерфейс INotifyPropertyChanged. Его реализация включена в базовый класс ViewModelBase, от которого наследуются все модели представления.

Реализация интерфейса INotifyPropertyChanged позволяет ViewModel уведомлять интерфейс о том, что значение свойства изменилось, и необходимо обновить привязанный элемент. Кроме того, в базовый класс также встроена валидация через INotifyDataErrorInfo, позволяющая отображать ошибки в пользовательском интерфейсе. Пример реализации приведён в листинге 4.8.

public abstract class ViewModelBase : INotifyPropertyChanged, INotifyDataErrorInfo, IAsyncDisposable

{

public ViewModelBase()

{

Validate();

}

private ValidationContext \_validationContext;

protected void Validate()

{

\_validationContext = new ValidationContext(this, null, null);

var validationResults = new List<ValidationResult>();

Validator.TryValidateObject(this, \_validationContext, validationResults, true);

foreach (var kv in \_errors.ToList()

.Where(kv => validationResults.All(r => r.MemberNames.All(m => m != kv.Key))))

{

\_errors.TryRemove(kv.Key, out \_);

OnErrorsChanged(kv.Key);

}

var q = from r in validationResults

from m in r.MemberNames

group r by m

into g

select g;

foreach (var prop in q)

{

var messages = prop.Select(r => r.ErrorMessage).ToList();

if (\_errors.ContainsKey(prop.Key))

{

\_ = \_errors.TryRemove(prop.Key, out \_);

}

if (\_errors.TryAdd(prop.Key, messages!))

{

OnErrorsChanged(prop.Key);

}

}

}

public event PropertyChangedEventHandler? PropertyChanged;

protected virtual void OnPropertyChanged([CallerMemberName] string? propertyName = null)

{

File.AppendAllText("log.txt", $"Свойство изменено: {propertyName}\n");

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs(propertyName));

}

protected bool Set<T>(ref T field, T value, [CallerMemberName] string? propertyName = null)

{

if (EqualityComparer<T>.Default.Equals(field, value)) return false;

field = value;

OnPropertyChanged(propertyName);

Validate();

return true;

}

public async virtual ValueTask DisposeAsync()

{

// TODO release managed resources here

}

private readonly ConcurrentDictionary<string, List<string>> \_errors = new();

public bool HasErrors => \_errors.Any(propErrors => propErrors.Value is { Count: > 0 });

public event EventHandler<DataErrorsChangedEventArgs>? ErrorsChanged;

public IEnumerable GetErrors(string? propertyName)

{

if (string.IsNullOrEmpty(propertyName)) return \_errors.SelectMany(err => err.Value.ToList());

if (!\_errors.ContainsKey(propertyName)) return null!;

if (\_errors.TryGetValue(propertyName, out var li) && li is { Count: > 0 }) return li;

return null!;

}

private void OnErrorsChanged(string propertyName)

{

ErrorsChanged?.Invoke(this, new DataErrorsChangedEventArgs(propertyName));

}

}

Листинг 4.8 – Реализация интерфейса «INotifyPropertyChanged» и базового ViewModel

## 4.6 Реализация команд

В WPF, при использовании паттерна «MVVM», для выполнения действий используются команды. Команда – обёртка над функцией, выполняющей действия.

Команды удобно привязываются к элементам управления (кнопкам). Они могут принимать параметры.

В программном средстве использовались 3 реализации команд: без параметра, обобщённая и команда, которая выбирает фото и возвращает его в виде массива байт. В приложении Г представлена реализация команды, которая выбирает фото.

## 4.7 Страницы

В приложении BookShop каждая страница реализована в виде представления (View), которое соответствует определённой модели представления (ViewModel). Использование шаблонов данных (DataTemplate) в главном окне позволяет автоматически отображать нужную страницу в зависимости от текущей ViewModel, привязанной к свойству CurrentViewModel. Пример соответствия View и ViewModel приведён в листинге 4.6.

<Window.Resources>

<DataTemplateDataType="{x:Type TypeName=viewModels:MainViewModel}">

<views:MainView/>

</DataTemplate>

<DataTemplateDataType="{x:Type TypeName=catalog:BookWrapViewModel}">

<books:BookWrapView/>

</DataTemplate>

<DataTemplateDataType="{x:Type TypeName=users1:UserWrapViewModel}">

<users:UserWrapView/>

</DataTemplate>

</Window.Resources>

Листинг 4.9 – Привязка View к ViewModel через DataTemplate

## 4.8 Выводы по главе

В данной главе были реализованы механизмы, обеспечивающие работоспособность программного средства. Были реализованы средства паттерна MVVM, реализована бизнес-логика.

Диаграммы классов программного средства представлена в приложении Д.

# 5 Тестирование, проверка работоспособности и анализ полученных результатов

## 5.1 Методика тестирования

Тестирование программного средства проводилась методом граничных значений, так как тестированию подвергались валидация вводных данных.

Для всех текстовых полей присутствует ограничение на максимальную длину, поэтому верхнюю границу + 1 проверять не предоставляется возможным.

Результаты тестирования отдельного поля подразумевают корректную валидацию остальных полей.

## 5.2 Тестирование форм ввода данных

### 5.2.1 Регистрация

Выполнить регистрацию, не указав все поля невозможно, так как кнопка «Зарегистрироваться» становится активной только после заполнения правильно всех полей.

Таблица 5.1 – Валидация регистрации

|  |  |  |
| --- | --- | --- |
| Поле | Ввод | Результат |
| Логин  Email  Фамилия  Имя  Телефон  Адрес  Пароль  Подтвердить пароль | Пустой | Невозможно выполнить регистрацию |
| Логин | Если существует в системе | Невозможно выполнить регистрацию (Такой пользователь уже зарегистрирован) |
| Пароль  Подтвердить пароль | 1  2 | Невозможно выполнить регистрацию (Пароли не совпадают) |

Окончание таблицы 5.1

|  |  |  |
| --- | --- | --- |
| Поле | Ввод | Результат |
| Пароль  Подтвердить пароль | 1  1 | Успех |
| Email |  | Неверный формат email |
| Email | @ | Неверный формат email |
| Email | test@mail | Неверный формат email |
| Email | test@mail.ru | Успех |
| Email | test@10.10 | Неверный формат email |

Поле email не поддерживает абсолютно все возможные варианты, так как это слишком трудозатратно и никто так не делает.

### 5.2.2 Создание книги

Создать книгу, не выбрав фото и не указав все поля, кроме описания невозможно, так как кнопка создания неактивна.

Таблица 5.2 – Валидация создания книги

|  |  |  |
| --- | --- | --- |
| Поле | Ввод | Результат |
| Имя  Автор  Жанр  Цена  Обложка | Пустой | Невозможно создать книгу |
| Цена | авыаыв | Не является числом |
| Цена | 0  -1  0.001 | Цена должна быть в диапазоне от 0.01 до 1000000.00 |
| Цена | 1 | Успех |
| Имя | Книга1 | Успех |
| Автор | Автор1 | Успех |
| Жанр | Художественная литература | Успех |
| Обложка | Выбрано изображение | Успех |
| Описание | Описание книги | Успех |

Для поля «Цена» действует ограничение ввода, принимаются только число от 0.01 до 1000000.00.

## 5.3 Оформление заказа

Для того, чтобы оформить заказ, необходимо чтобы книга была доступна и добавлена в корзину.

Таблица 5.3 – Добавление товара в корзину

|  |  |  |
| --- | --- | --- |
| Объект | Ввод | Результат |
| Книга1 | Выбрана книга | Книга добавлена в корзину |
| Книга1 | Не выбрана книга | Команда добавления в корзину не активна |

Таблица 5.4 – Оформление заказа

|  |  |  |
| --- | --- | --- |
| Объект | Ввод | Результат |
|  | Оформить заказ | Заказ оформлен |
| Книга1 | Удалить позицию | Книга1 удалена из корзины |
|  | Отменить | Корина очищается |

Оформление и отмена заказа невозможна, если корзина пуста. Чтобы удалить позицию необходимо выбрать позицию книги

## 5.4 Выводы по главе

В данной главе были выполнены проверки основных механизмов программного средства. Было проведено 4 теста на валидацию, где проверялись все поля. Проводились тесты оформления заказа, так как это ключевая задача программного средства.

В ходе тестов было подтверждены следующие моменты:

* все поля ввода текста имеют ограничение на максимальную длину;
* все формы, используемые в программном средстве, имеют валидацию;
* механизм проверки оформления заказа, а также отмены и удаления позиций работают корректно;
* программное средство работает как заявлено.

# 6 Руководство по установке и использованию

## 6.1 Установка программного средства

Для создания установщика программного средства использовалась программа «Inno Setup Compiler» [13], она позволяет быстро создать установщик, используя мастер, или написать скрипт на специальном языке.

Перед созданием установщика программное средство было собрано, использовались следующие параметры:

* конфигурация: Release | Any CPU;
* целевая платформа: net8.0-windows;
* режим развёртывания: автономное;
* целевая среда выполнения: win-x64;
* создание отдельного файла
* компиляция ReadyToRun.

Затем собранный проект был использован при создании установщика. Вместе с проектом, установщик включает в себя стандартную базу данных, содержащую небольшой набор данных. После установки база данных расположена по пути «%Пользователь%/Документы/CarRent/CarRent.db».

При запуске установщика мы видим окно, где нас спрашивают, требуется ли создавать ярлык на рабочем столе.

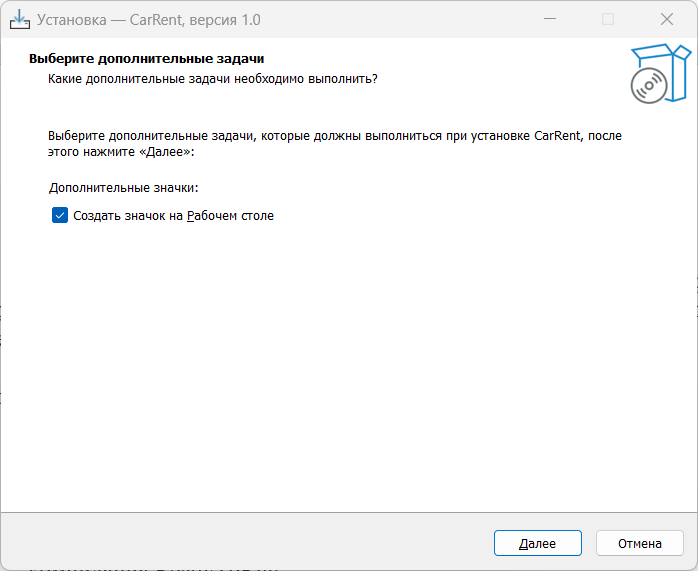


Рисунок 6.1 – Первый шаг установки

После нажатия на кнопку далее мы видим окно, где дублируются выбранные параметры (в данном случае он всего 1, создание ярлыка на рабочем столе).

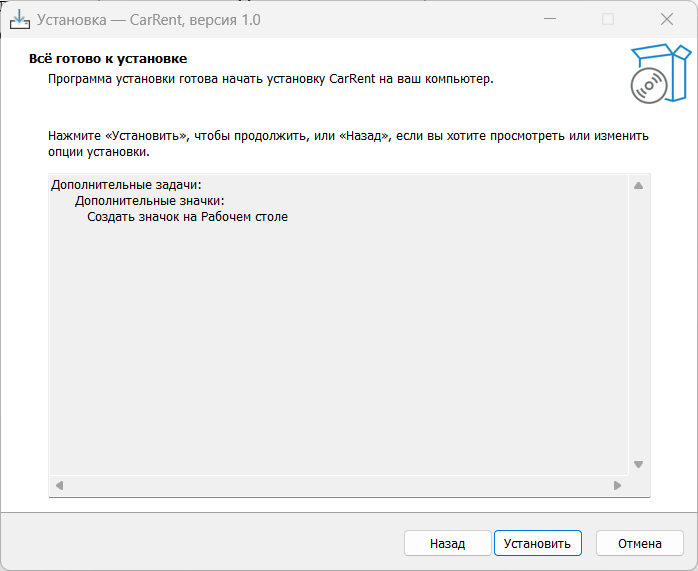


Рисунок 6.2 – Второй шаг установки

После нажатия на кнопку «Установить» запускается процесс установки.

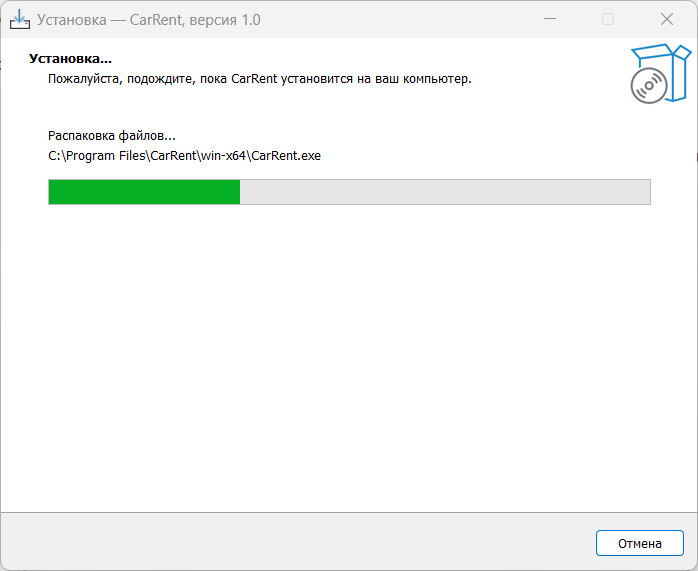


Рисунок 6.3 – Третий шаг установки

После завершения установки, пользователю будет предложено запустить программное средство либо просто завершить установку.

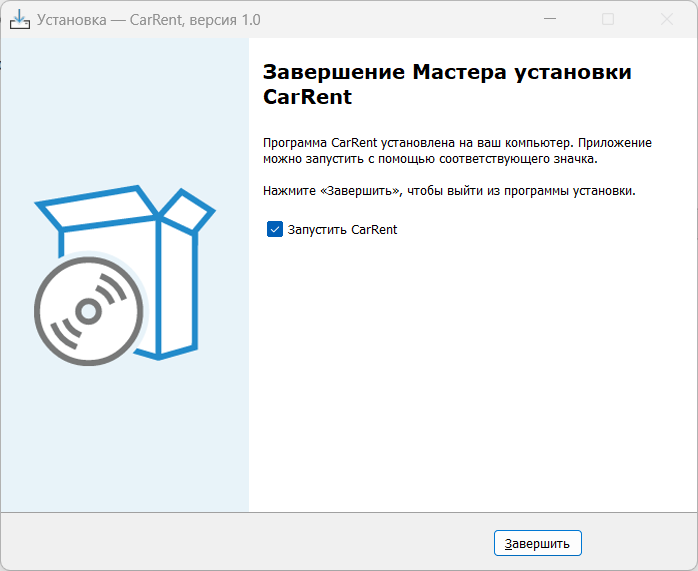


Рисунок 6.4 – Завершение установки

## 6.2 Использование программного средства

При использовании программного средства пользователь может иметь 2 роли: «Клиент» и «Администратор». В зависимости от роли, функционал программного средства отличается.

## 6.2.1 Регистрация

При первом запуске программного средства, пользователь не имеет своего аккаунта, ему необходимо перейти к регистрации, там он заполняет данные и, если данные заполнены корректно, то пользователь регистрируется, После ему необходимо произвести логин, потом ему становятся доступны функции оформления заказа и добавления товара в корзину.

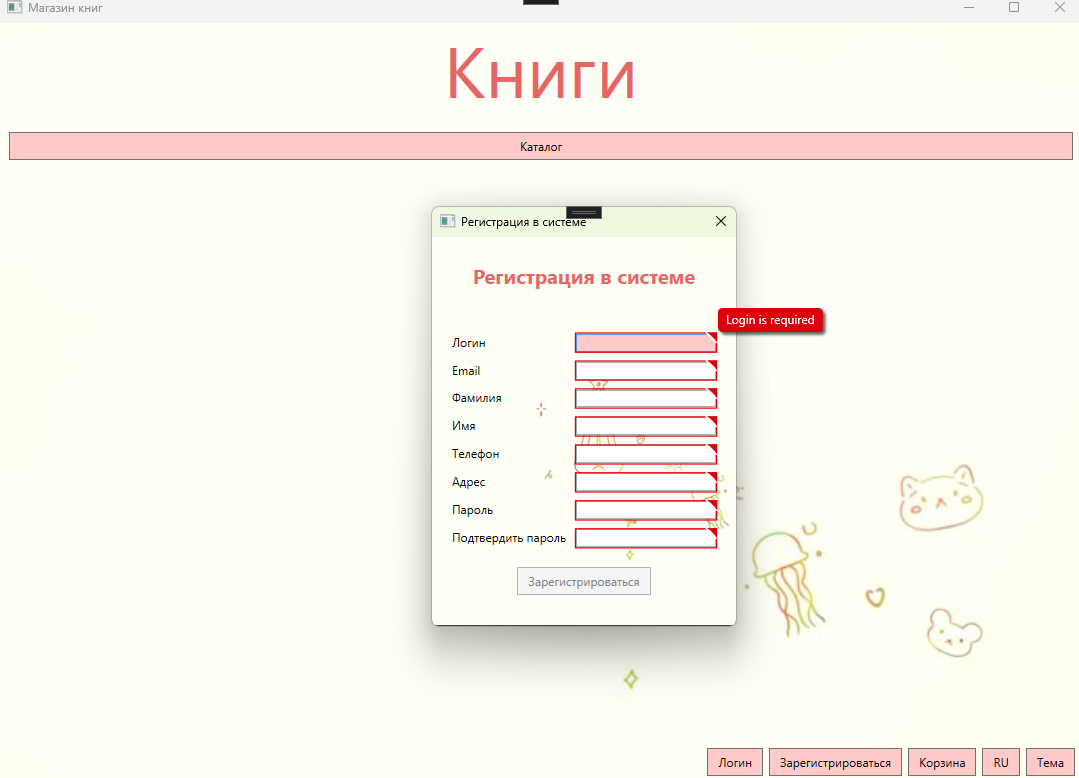


Рисунок 6.5 – Страница регистрации

## 6.2.2 Авторизация

В случае, если у пользователя имеется аккаунт, ему необходимо авторизоваться, страница авторизации открывается по кнопке логин.

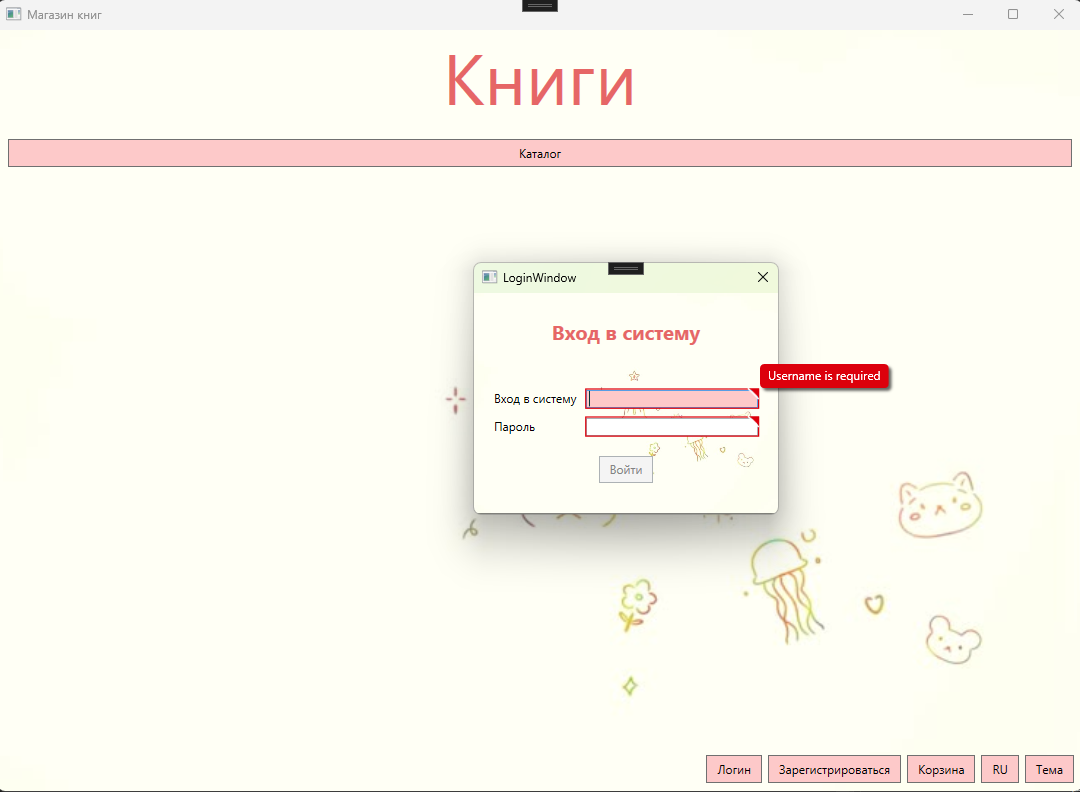


Рисунок 6.6 – Страница авторизации

### 6.2.3 Список книг

Главной страницей программного средства является список каталог книг. В зависимости от роли, функционал страницы отличается. Перейти к этой странице можно путём нажатия кнопки «Каталог» в навигационном меню.

Клиент может просмотреть список доступных книг. При выборе книги и нажатии на кнопку просмотр книги, показывается подробная информация о книге.

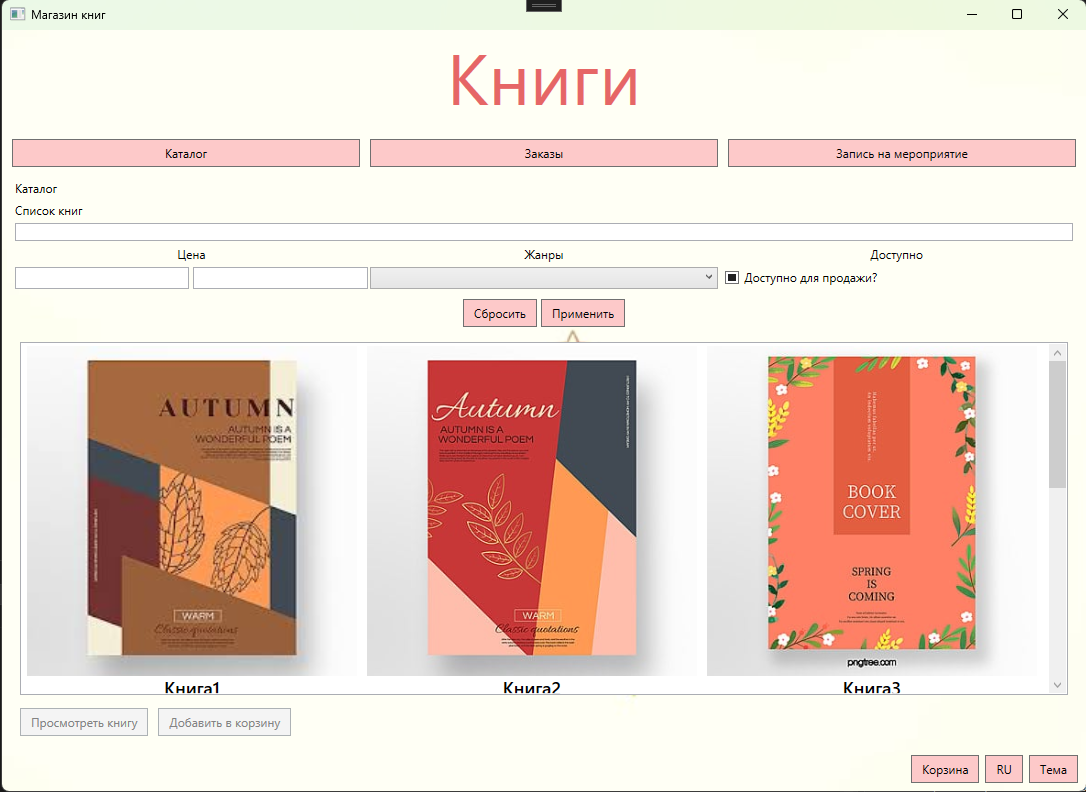


Рисунок 6.7 – Страница «Каталог» от лица клиента

Администратор, находясь на данной странице, может создавать, изменять, удалять книги просматривать информацию о книге.

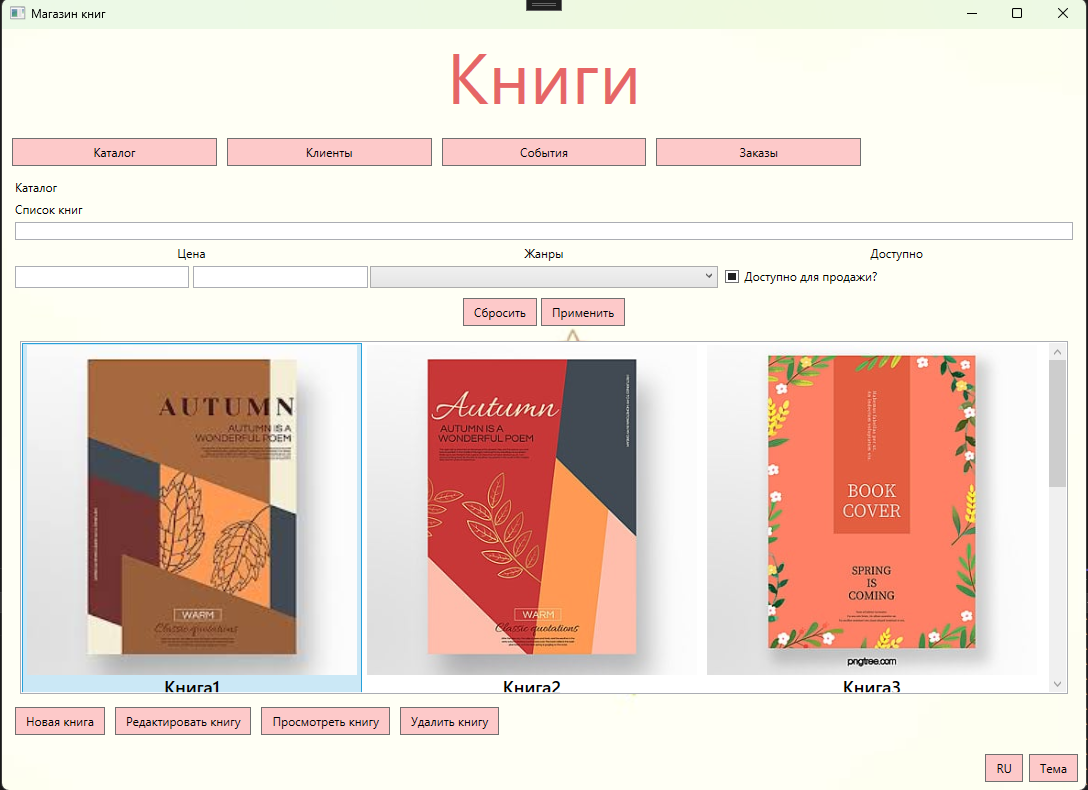


Рисунок 6.8 – Страница «Каталог» от лица администратора

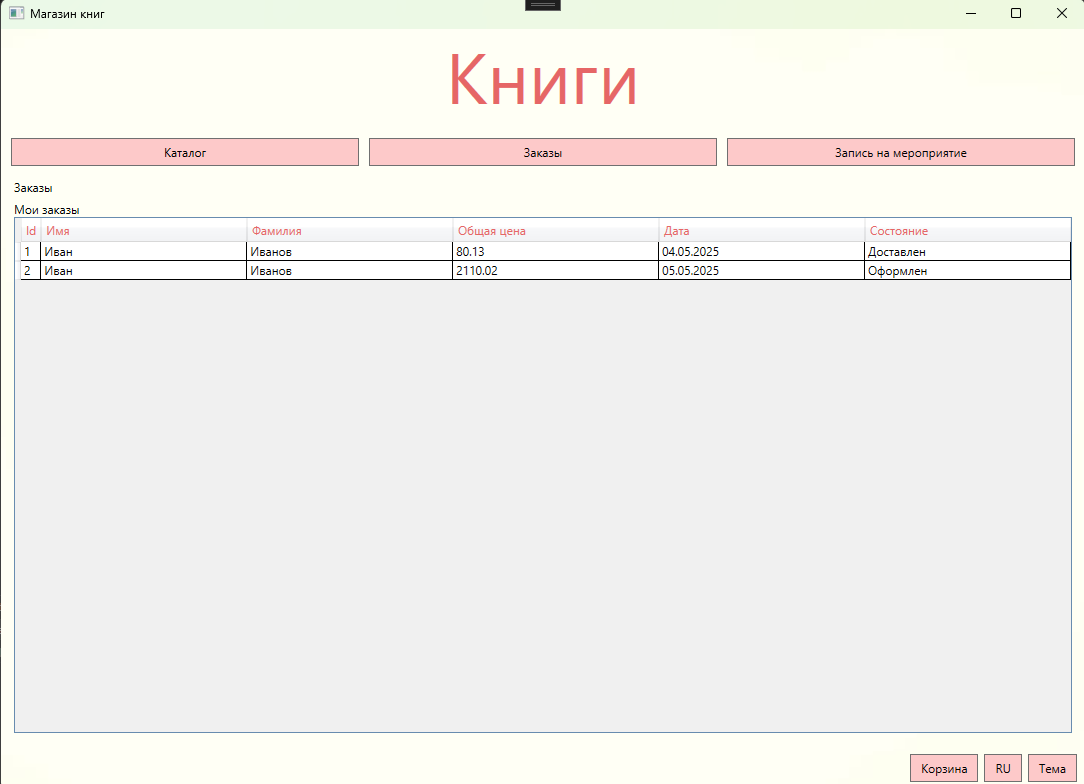


Рисунок 6.9 – Заказы

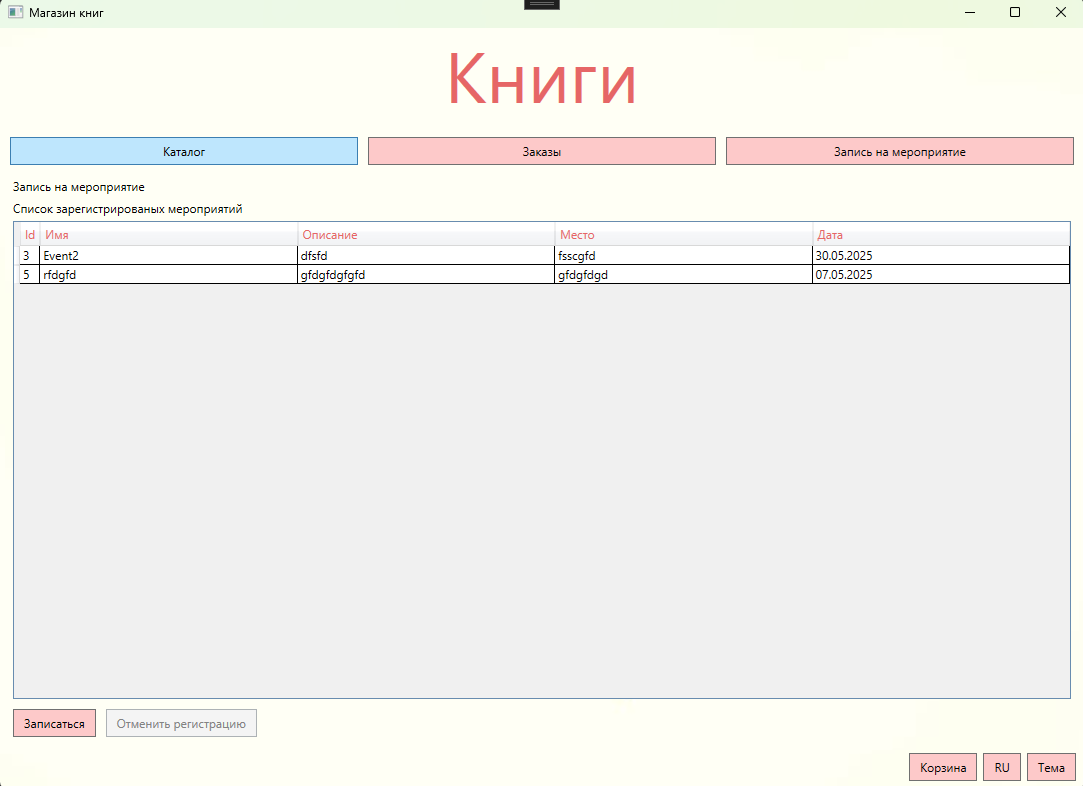


Рисунок 6.10 – Запись на мероприятия

### 6.2.4 Страница клиентов

Данная страница доступна только администратору, на ней он может добавлять, удалять, изменять клиентов, менять роли.

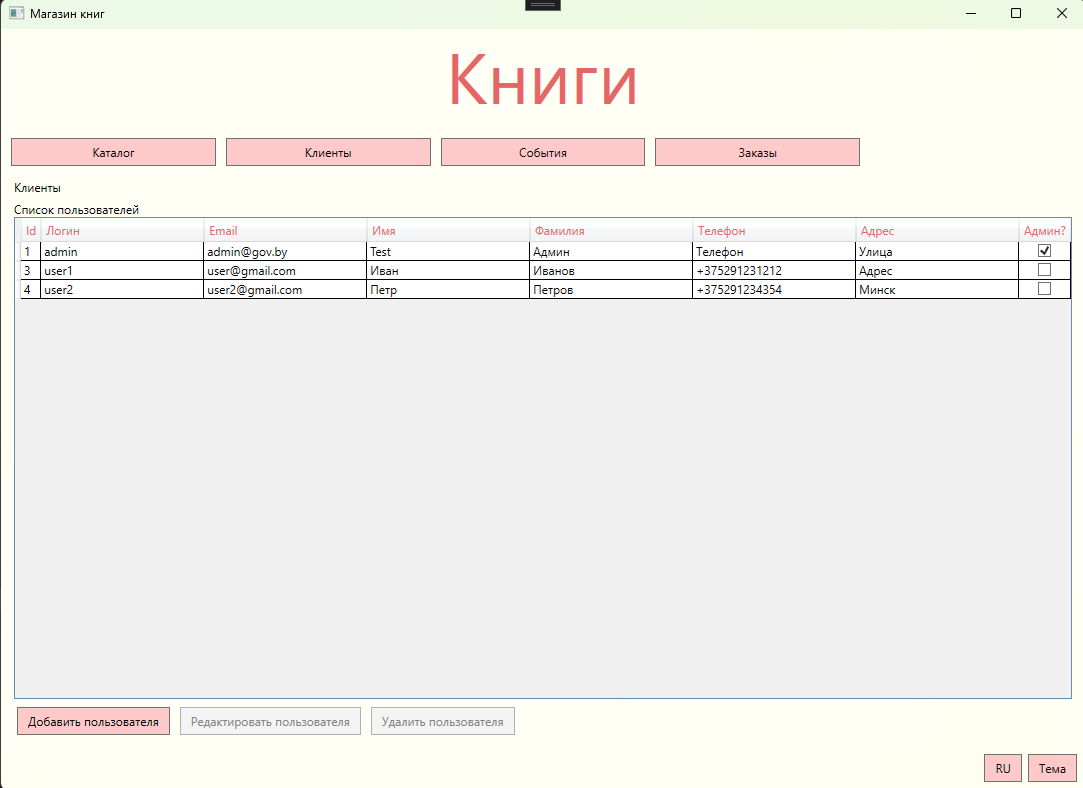


Рисунок 6.11 – Страница «Клиенты»

### 6.2.5 Страница заказы

Данная страница доступна как клиентам, так и администратору.

В случае, если пользователь является клиентом, то он видит все свои заказы.

В случае, если пользователь является администратором, то он видит все заказы и может изменить или удалить заказ.

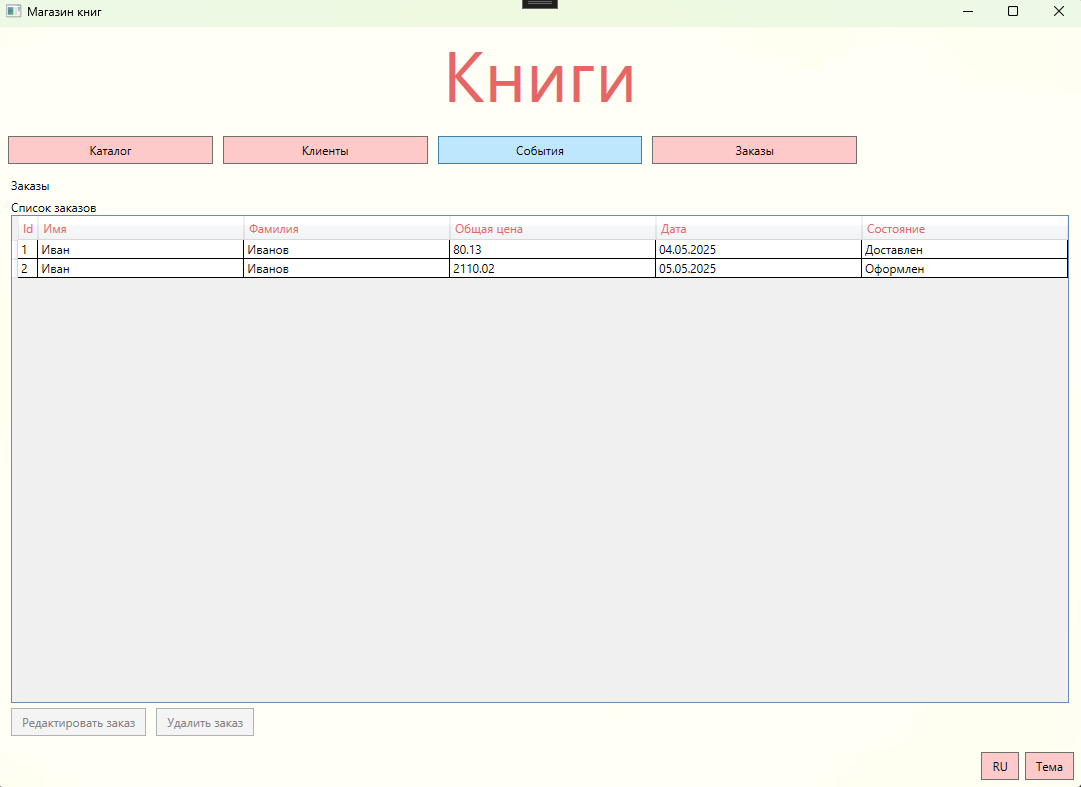


Рисунок 6.12 – Страница «Заказы» от лица администратора

### 6.2.6 Страница события (мероприятия)

Данная страница доступна администраторам, они могут добавлять, редактировать, удалять мероприятия.

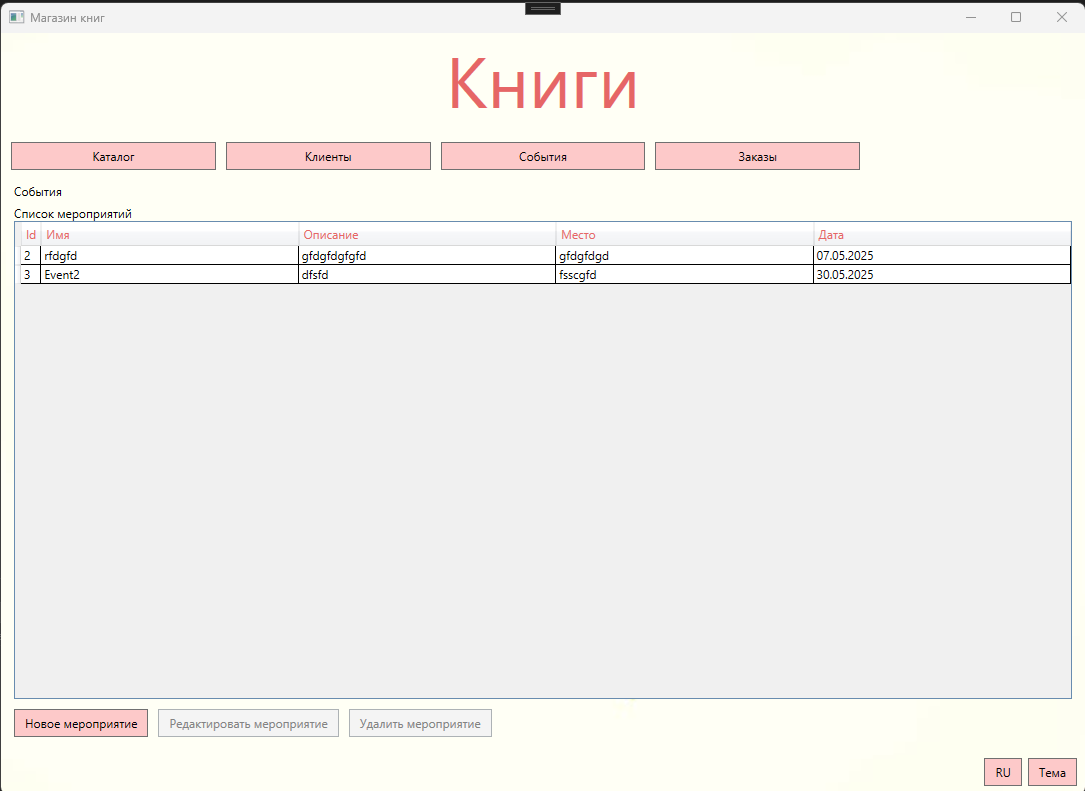


Рисунок 6.13 – Страница «События» от лица администратора

### 6.2.7 Корзина товаров

Данная страница доступна только клиентам. На странице клиент может управлять корзиной товаров, удалять, оформлять или отменять заказ

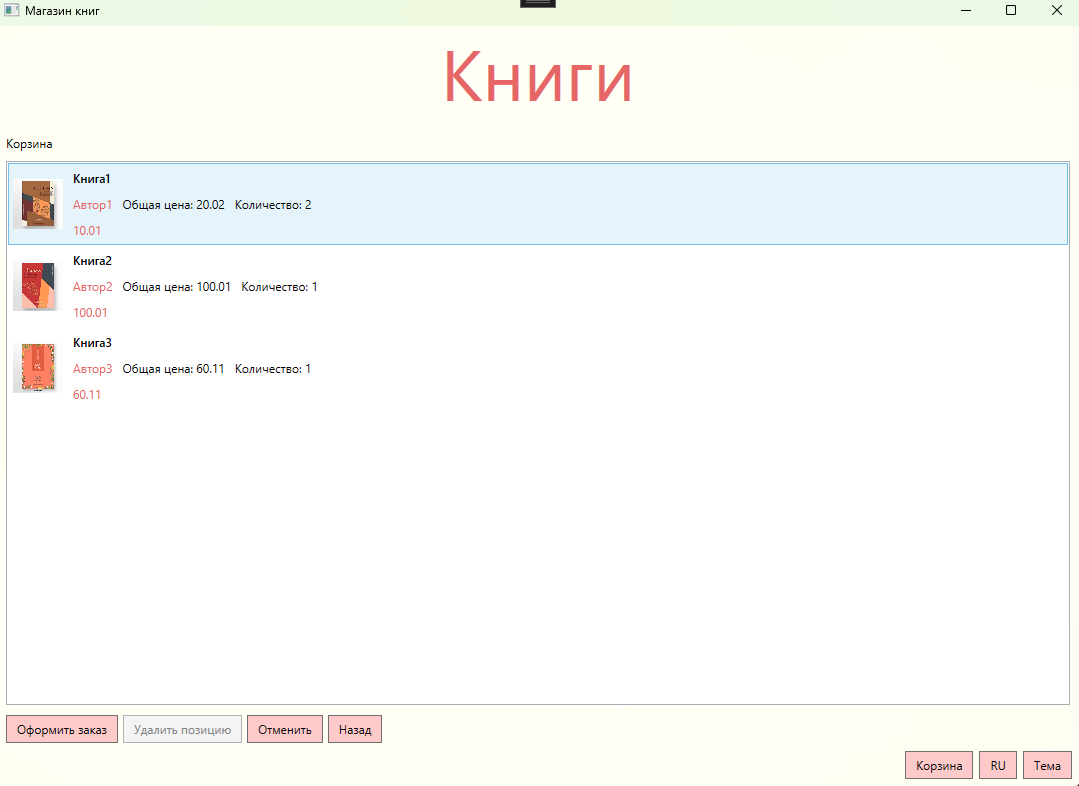


Рисунок 6.14 – Страница «Корзина товаров» от лица клиента

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Hello каршеринг - Поминутная аренда автомобилей в Минске [Электронный ресурс]. – 2025. – Режим доступа: <https://hello.by/> Дата доступа: 24.03.2025
2. Аренда авто в Алматы без водителя рядом | Rentacars - прокат по цене от 15 000 тенге/сутки [Электронный ресурс]. – 2025. – Режим доступа: <https://rentacars.kz/> Дата доступа: 24.03.2025
3. Аренда авто в Минске – прокат автомобилей с доставкой, стоимость [Электронный ресурс]. – 2025. – Режим доступа: <https://to-car.by/> Дата доступа: 24.03.2025
4. Каршеринг Мультимоторс [Электронный ресурс]. – 2025. – Режим доступа: <https://multik.by/> Дата доступа: 24.03.2025
5. WPF Login with N-Tier Architecture [Электронный ресурс]. – 2017. – Режим доступа: <https://www.c-sharpcorner.com/> Дата доступа: 30.03.2025
6. SQLite Home Page [Электронный ресурс]. – 2025. – Режим доступа: <https://sqlite.org/> Дата доступа: 30.03.2025
7. Entity Framework Core | Введение [Электронный ресурс]. – 2021. – Режим доступа: <https://metanit.com/sharp/entityframeworkcore/1.1.php> Дата доступа: 02.04.2025
8. Паттерн Репозиторий [Электронный ресурс]. – 2015. – Режим доступа: <https://metanit.com/sharp/articles/mvc/11.php> Дата доступа: 04.04.2025
9. Data Transfer Object (DTO) design pattern [Электронный ресурс]. – 2023. – Режим доступа: <https://colinchswift.github.io/2023-10-26/09-48-36-418884-data-transfer-object-dto-design-pattern/> Дата доступа: 06.04.2025
10. Model-View-ViewModel - .NET | Microsoft Learn [Электронный ресурс]. – 2024. – Режим доступа: <https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm> Дата доступа: 06.04.2025
11. Mediator [Электронный ресурс]. – 2025. – Режим доступа: <https://refactoring.guru/design-patterns/mediator> Дата доступа: 07.04.2025
12. Messenger - Community Toolkits for .NET | Microsoft Learn [Электронный ресурс]. – 2025. – Режим доступа: [https://learn.microsoft.com/en-us/dotnet/communitytoolkit/mvvm/messenger Дата доступа 10.04.2025](https://learn.microsoft.com/en-us/dotnet/communitytoolkit/mvvm/messenger%20Дата%20доступа%2010.04.2025)
13. Inno Setup [Электронный ресурс]. – 2025. – Режим доступа: <https://jrsoftware.org/isinfo.php> Дата доступа: 24.04.2025

# Приложение А

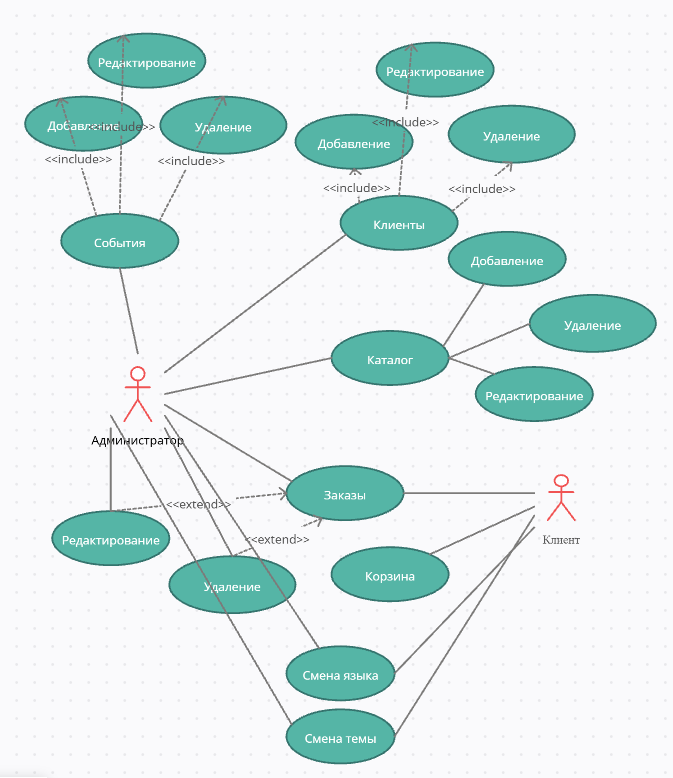


Рисунок А.1 – UML-диаграмма вариантов использования

# Приложение Б

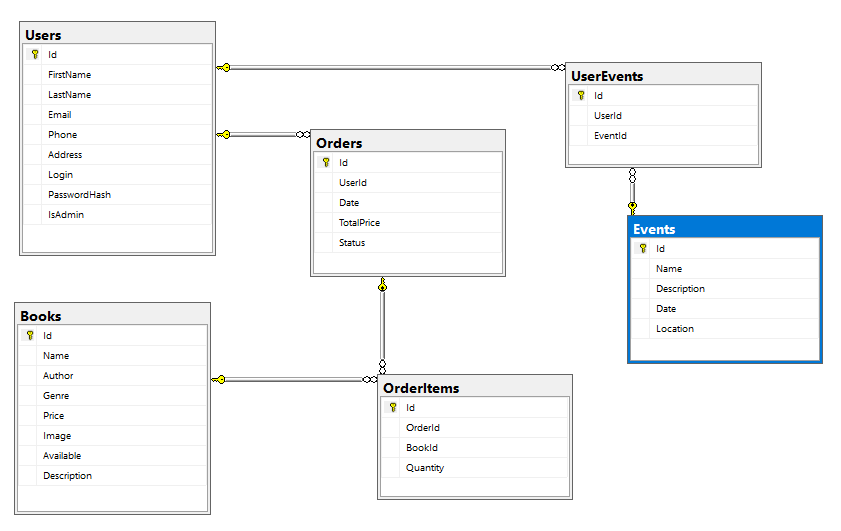


Рисунок Б.1 – UML логическая схема базы данных

# Приложение В

public RelayCommand ApplyFilterCommand => new(async obj =>

{

try

{

// var minPrice = decimal.Parse(MinPrice, CultureInfo.InvariantCulture);

// var maxPrice = decimal.Parse(MinPrice, CultureInfo.InvariantCulture);

FilterItems.Filter = o =>

{

var bookItem = o as Book;

if (bookItem == null)

{

return bookItem == null;

}

var result = string.IsNullOrEmpty(SearchText) || bookItem.Name.Contains(SearchText, StringComparison.CurrentCultureIgnoreCase);

if (!string.IsNullOrEmpty(MinPrice) && !string.IsNullOrEmpty(MaxPrice))

{

var minPrice = decimal.Parse(MinPrice, CultureInfo.InvariantCulture);

var maxPrice = decimal.Parse(MaxPrice, CultureInfo.InvariantCulture);

result = result && (bookItem.Price >= minPrice && bookItem.Price <= maxPrice);

}

else if(!string.IsNullOrEmpty(MinPrice) && string.IsNullOrEmpty(MaxPrice))

{

var minPrice = decimal.Parse(MinPrice, CultureInfo.InvariantCulture);

result = result && bookItem.Price >= minPrice;

}

else if (string.IsNullOrEmpty(MinPrice) && !string.IsNullOrEmpty(MaxPrice))

{

var maxPrice = decimal.Parse(MaxPrice, CultureInfo.InvariantCulture);

result = result && bookItem.Price <= maxPrice;

}

if (SelectedGenre == null)

{

result = result && SelectedGenre == null;

}

else

{

if (SelectedGenre.Id != 1)

{

result = result && bookItem.Genre.Equals(SelectedGenre.Name, StringComparison.CurrentCultureIgnoreCase);

}

}

if (IsAvailable == null)

{

result = result && IsAvailable == null;

}

else

{

result = result && bookItem.Available == IsAvailable;

}

return result;

};

}

catch (Exception e)

{

MessageBox.Show(e.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

});

Листинг В.1 – Реализация команды применения фильтра

# Приложение Г

public RelayCommand SelectImageCommand => new(async obj => { var openFileDialog = new OpenFileDialog(); openFileDialog.Filter = "JPG|.jpg|PNG|.png|GIF|\*.gif";

var result = openFileDialog.ShowDialog();  
  
 if (result == true)  
 {  
 var fileName = openFileDialog.FileName;  
   
 var appPath = AppDomain.CurrentDomain.BaseDirectory;  
 //var newFileName = Path.GetFileName(fileName);  
 var extension = Path.GetExtension(fileName);  
 var newFileName = $"{Guid.NewGuid().ToString()}{extension}";  
 var newImgPath = Path.Combine(appPath, "Images", newFileName);  
  
 if (File.Exists(fileName))  
 {  
 File.Copy(fileName, newImgPath, true);  
  
 Entity.Image = newFileName;  
 }  
 }

});

Листинг Г.1 – Реализация команды выбора фото

# Приложение Д

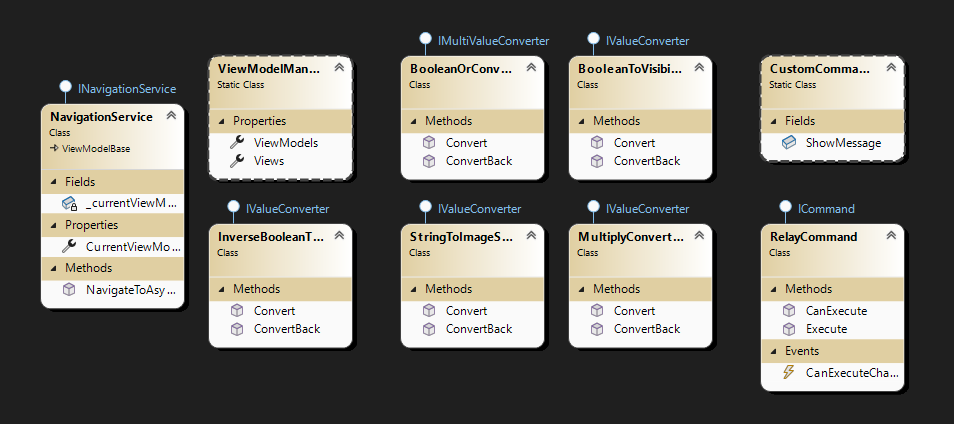


Рисунок Д.1 – UML-диаграмма классов сервисов, конвертеров и комманд

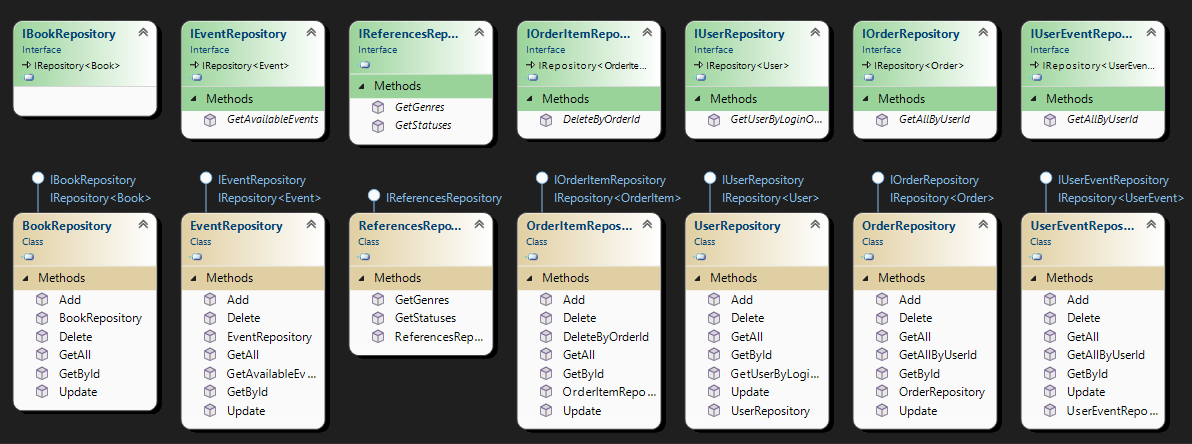


Рисунок Д.2 – UML-диаграмма классов репозиториев

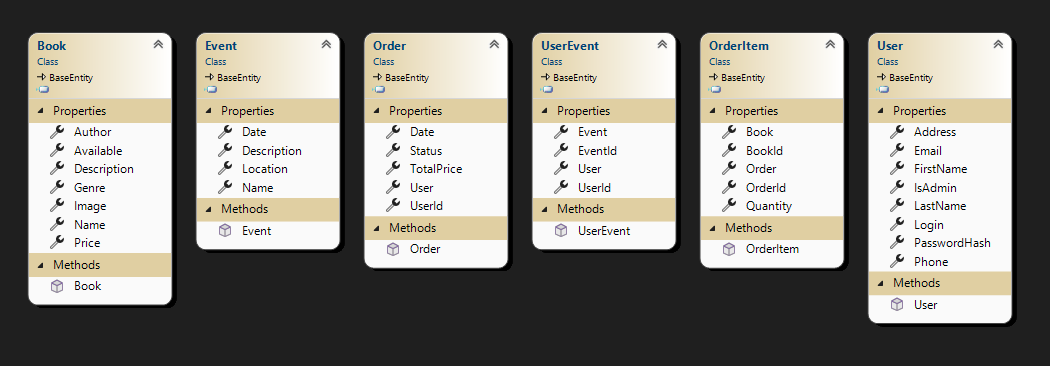


Рисунок Д.3 – UML-диаграмма классов-сущностей