

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждения образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий  
Кафедра программной инженерии  
Специальность 1-40 01 01 Программное обеспечение информационных технологий  
Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема Программное средство для электронной библиотеки «Бумажный город»

Исполнитель  
студент (ка) 2 курса группы 4 Тараканов Никита Сергеевич  
(Ф.И.О.)

Руководитель работы ассистент Чистякова Ю. А.  
(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой \_\_\_\_\_  
Председатель Пацей Н.В  
(подпись)

Минск 2023

## Содержание

ВВЕДЕНИЕ .....	4
1 Аналитический обзор литературы и формирование требований .....	5
1.1 Анализ прототипов .....	5
1.1.1 ЛитРес.....	5
1.1.2 Google Play Книги .....	6
1.1.3 LiveLib .....	6
1.1.4 Twirpx .....	7
1.1.5 Библиотека .....	8
1.2 Аналитический обзор литературных источников .....	9
1.3 Требования к проекту .....	9
2 Анализ требований к программному средству и разработка функциональных требований .....	10
2.1 Описание средств разработки.....	10
2.1.1 Microsoft Visual Studio 2022 .....	10
2.1.2 Программная платформа .NET 7.0 .....	10
2.1.3 Язык программирования C#.....	11
2.1.4 Расширяемый язык разметки XAML .....	11
2.1.5 Технология WPF.....	11
2.1.6 Технология Entity Framework Core.....	11
2.1.7 Microsoft SQL Server .....	12
2.1.8 Протокол SMTP .....	12
2.1.9 MD5 хеширование.....	13
2.1.10 Паттерн MVVM.....	13
2.2 Спецификация функциональных требований к программному средству .....	14
2.3 Спецификация функциональных требований .....	15
3 Проектирование программного средства .....	16
3.1 Общая структура .....	16
3.2 Взаимоотношения между классами .....	24
3.3 Модель базы данных.....	24
3.4 Проектирование архитектуры приложения .....	28
3.5 Проектирование последовательностей проекта.....	28
4 Реализация программного средства.....	30
4.1 Основные классы программного средства .....	30

	3
4.2 Описание классов и методов программного средства .....	30
4.2.1 Авторизация.....	30
4.2.2 Регистрация.....	31
4.2.3 Просмотр книг .....	32
4.2.4 Скачивание книги.....	32
4.2.5 Оплата книги.....	32
4.2.6 Добавление отзыва к книге .....	32
4.2.7 Просмотр профиля .....	33
4.2.8 Приложение администратора .....	33
5 Тестирование, проверка работоспособности и анализ полученных результатов .	34
5.1 Тестирование авторизации и регистрации .....	34
5.2 Тестирование оплаты книги.....	38
5.3 Тестирование приложения администратора .....	41
6 Руководство по установке и использованию .....	43
ЗАКЛЮЧЕНИЕ .....	49
Список использованных источников .....	50
ПРИЛОЖЕНИЕ А .....	51
ПРИЛОЖЕНИЕ Б .....	52
ПРИЛОЖЕНИЕ В .....	53

## ВВЕДЕНИЕ

Электронная библиотека – это программное средство для ПК, которое позволяет хранить разного рода документы и использовать их в электронном виде. Электронные библиотеки позволяют сохранять и использовать разнообразные коллекции электронных книг, а также совершать последующую их передачу пользователю в текстовом виде. Вне зависимости от того, является человек студентом или бизнесменом, ему наверняка приходилось сталкиваться с проблемой поиска нужной информации, для данных целей и предназначены программы, которые хранят нужную информацию. В последнее время читатели все больше отдают предпочтение электронным ресурсам из-за их доступности. Людям больше не нужно ходить в библиотеку, общаться с регистраторами и следить за тем, чтобы не просрочить время на чтение, стоит всего скачать нужное программное средство и иметь доступ к глобальной сети интернет, и вся библиотека будет буквально в электронном устройстве пользователя. Также к приятным моментам электронных библиотек можно отнести то, что пользователь с легкостью может найти те книги, которых нет практически ни в одном магазине. Использование электронных библиотек позволяет людям сэкономить большое количество средств и времени, которое бы они тратили на поиск и покупку определенной книги.

Цель данного курсового проекта - разработать программное средство с реализацией эффективного поиска и навигации по коллекции книг для взаимодействия пользователя и приложения электронной библиотеки, дать пользователю возможность скачивать различные книги. Язык разработки проекта – C#. При выполнении курсового проекта будут использованы принципы и приемы объектно-ориентированного программирования. Также будут использоваться технологии Windows Presentation Foundation(WPF) и база данных Microsoft SQL Server.

В рамках курсового проекта будут рассматривать и реализовываться такие функциональные возможности, как поиск и фильтрация книг, возможность оставления отзывов, удобное управление вкладками, а также возможность скачивания интересных пользователю книг.

# 1 Аналитический обзор литературы и формирование требований

## 1.1 Анализ прототипов

Был проведен анализ целей и задач, поставленных в рамках данного курсового проекта, а также изучены примеры аналогичных решений. Исходя из этого анализа были определены требования к разрабатываемому программному средству, учитывающие как все его преимущества, так и недостатки рассмотренных альтернативных решений.

### 1.1.1 ЛитРес

Наиболее популярное альтернативное решение — интернет-ресурс «ЛитРес».

Интернет-ресурс «ЛитРес» представляет собой платформу для чтения электронных книг, предоставляющую пользователям доступ к широкому выбору литературы различных жанров. Анализируя данный ресурс, можно выделить следующие преимущества: удобный интерфейс, широкий выбор книг, возможность читать в автономном режиме. Также можно отнести к преимуществам возможность пользователей скачивать определенные книги в разных форматах, таких как EPUB, PDF или FB2. Пользователи данного ресурса могут свободно обсуждать книги в отзывах, либо же просто ознакомиться с рецензиями критиков.

К недостаткам можно отнести следующие моменты: ограниченный выбор бесплатных книг, зависимость от интернет-соединения, отсутствие возможности загрузки собственных книг.

Интерфейс интернет-ресурса представлен на рисунке 1.1.

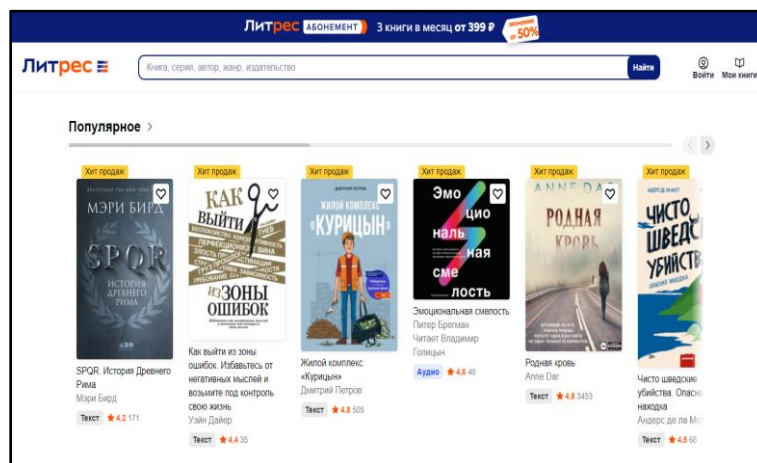


Рисунок 1.1 – Интернет-ресурс «ЛитРес»

Дизайн ЛитРес отличается современностью и элегантностью, создавая приятную и привлекательную атмосферу для пользователей. ЛитРес использует минималистичный дизайн, где основной акцент делается на контенте. Минималистический подход позволяет пользователям сосредоточиться на книгах и информации, не отвлекаясь на излишние элементы интерфейса.

### 1.1.2 Google Play Книги

К наиболее популярным альтернативным решениям также можно отнести приложение от Google Inc под названием «Google Play Книги».

Google Play Книги (Google Play Books) - это платформа электронных книг, разработанная Google. Она предоставляет пользователям возможность покупать, скачивать и читать электронные книги на различных устройствах, включая смартфоны, планшеты и компьютеры. В ходе анализа данного приложения были выявлены следующие преимущества: широкий выбор книг, автономное чтение, синхронизация прогресса. Также наиболее важным преимуществом данного приложения является мультиплатформенность, т. е. приложение Google Play Книги доступно на различных платформах, включая Android, IOS и веб-версия. В отличие от интернет-ресурса «ЛитРес» в данном приложении есть возможность добавления собственных книг.

Но, не смотря на все преимущества данного приложения, в нем также можно выделить следующие недостатки: ограниченный доступ, т. е. некоторые книги могут быть доступны только в отдельных регионах, ограниченная совместимость форматов и отсутствие поддержки аудиокниг.

Интерфейс приложения представлен на рисунке 1.2.

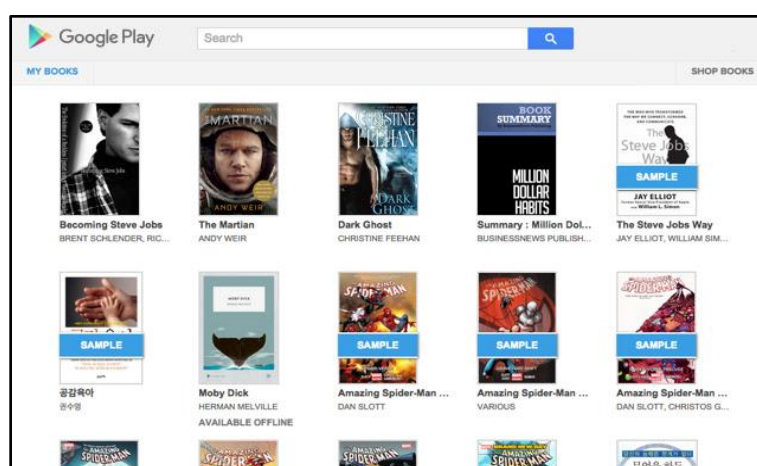


Рисунок 1.2 – Приложение «Google Play Книги»

Дизайн Google Play Книги отличается современностью, функциональностью и простотой использования. Google Play Книги использует принципы материального дизайна, которые придают интерфейсу современный и стильный вид. Он включает плавные анимации, яркие цвета и интуитивные элементы управления.

### 1.1.3 LiveLib

Ещё одной альтернативой решения поставленных задач является интернет-ресурс «LiveLib».

LiveLib - это онлайн-платформа для чтения и обмена электронными книгами. Она предоставляет доступ к широкому выбору книг различных жанров и тематик, которые можно читать онлайн или скачивать для офлайн-чтения. К преимуществам данной платформы можно отнести: большой выбор книг, бесплатный доступ к книгам, возможность делиться отзывами и рекомендациями, удобный интерфейс. Отдельно можно выделить следующее преимущество: возможность создания собственной библиотеки, что позволяет пользователям добавлять книги в списки «Читаю», «Прочитал» и «Хочу прочитать».

Также были выделены следующие недостатки: ограниченный выбор книг, доступность некоторых книг только в определенных регионах, ограниченная пользовательская база.

Интерфейс интернет-ресурса «LiveLib» представлен на рисунке 1.3.

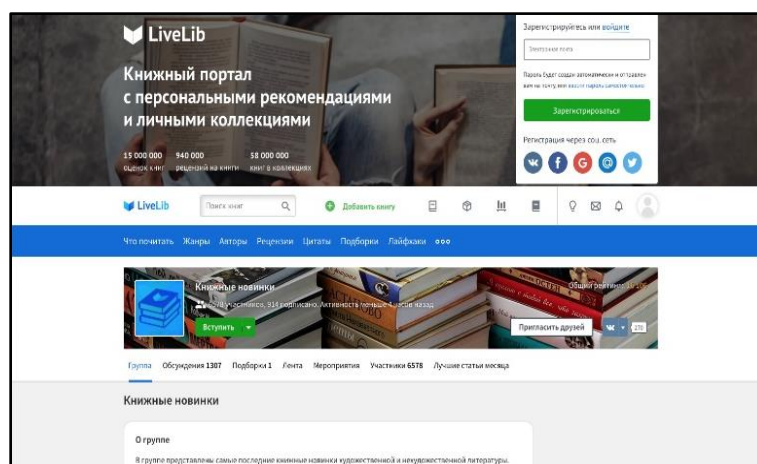


Рисунок 1.3 – Интернет-ресурс «LiveLib»

Дизайн LiveLib представляет собой современную и привлекательную визуальную концепцию, созданную с учетом потребностей пользователей в удобстве и функциональности. LiveLib использует яркую цветовую палитру, которая создает энергичную и динамичную атмосферу на сайте. Цвета акцентов помогают выделить важные элементы интерфейса и привлекают внимание пользователей.

#### 1.1.4 Twirpx

В список рассматриваемых альтернатив решения поставленных задач также можно включить интернет-ресурс «Twirpx».

Twirpx (или TWIRPX) — это российская онлайн-платформа, предназначенная для обмена и публикации электронных книг, журналов и других текстовых материалов. В ходе анализа данного ресурса были выделены следующие преимущества: большой выбор книг, бесплатный доступ к книгам, наличие сообщества пользователей, возможность публикации книг или текстовых материалов.





## 1.2 Аналитический обзор литературных источников

В процессе разработки программного средства для электронной библиотеки была проведена обширная литературная работа, включающая изучение специализированной технической литературы, учебно-методических пособий и справочных материалов, а также анализ статей и публикаций, доступных в интернете.

Можно выделить следующий источник: официальная документация Microsoft, которая предоставляет подробную информацию о различных аспектах WPF, также предоставляет различную информацию о различных аспектах работы с базой данных. Дополнительная информация о принципах работы WPF и баз данных была получена из различных статей интернет-источников, таких как Metanit.

## 1.3 Требования к проекту

В ходе анализа вышеперечисленных аналогов и ознакомления с различными литературными источниками были выделены преимущества и недостатки альтернативных решений, что позволяет корректно сформулировать список требований, которые должны быть соблюдены при разработке программного средства. Эти требования служат основой для проектирования, разработки и тестирования системы.

Программное средство должно обеспечивать выполнение следующих функциональных требований:

- регистрация и аутентификация: пользователь должен иметь возможность создать учетную запись в системе и аутентифицироваться для доступа к функционалу библиотеки;
- поиск и фильтрация: пользователь должен иметь возможность осуществлять поиск и фильтрацию книг по различным критериям;
- скачивание книг: пользователь должен иметь возможность скачивать книги из приложения;
- отзывы: пользователь должен иметь возможность оставлять и читать отзывы к книгам;
- управление пользовательским аккаунтом: пользователь должен иметь возможность изменять телефон и фото в профиле;
- обратная связь: пользователь должен иметь возможность обращаться с проблемой в техническую поддержку приложения;
- производство оплаты: пользователь должен иметь возможность производить оплату книги и ожидать подтверждения оплаты администратором.

Нефункциональные требования:

- интерфейс пользователя: интерфейс должен быть интуитивно понятным и удобным для использования, с хорошим пользовательским опытом;
- безопасность: система должна обеспечивать безопасность пользовательских данных, включая защиту от несанкционированного доступа и хранение паролей в безопасной форме.

## **2 Анализ требований к программному средству и разработка функциональных требований**

### **2.1 Описание средств разработки**

При разработке приложения были использованы:

- интегрированная среда разработки Microsoft Visual Studio 2022;
- программная платформа .NET 7.0;
- язык программирования C#;
- расширяемый язык разметки XAML;
- технология WPF;
- технология Entity Framework Core;
- MS SQL Server;
- протокол SMTP;
- MD5 хеширование.

Все перечисленные выше инструменты помогут разработать приложения опираясь на все поставленные требования к приложению.

#### **2.1.1 Microsoft Visual Studio 2022**

Microsoft Visual Studio - это интегрированная среда разработки (IDE), разработанная компанией Microsoft. Она предоставляет разработчикам широкий спектр инструментов и функциональности для создания программного обеспечения под различные платформы, включая Windows, macOS, iOS, Android и веб-приложения.

Visual Studio также предлагает возможности для создания графического интерфейса приложений с использованием Windows Presentation Foundation (WPF), Windows Forms и других технологий. Она поддерживает создание веб-приложений с использованием ASP.NET, разработку мобильных приложений для платформы Xamarin, а также разработку облачных приложений на базе Microsoft Azure.

#### **2.1.2 Программная платформа .NET 7.0**

Программная платформа .NET 7 (или .NET Core 7) является одной из версий платформы .NET, разработанной компанией Microsoft. .NET является средой выполнения и набором инструментов для разработки и выполнения приложений под различные операционные системы, включая Windows, macOS и Linux.

.NET 7 представляет собой новую версию платформы, в которой внедрены различные улучшения и новые возможности. Она обеспечивает разработчикам возможность создавать масштабируемые, быстрые и безопасные приложения с использованием различных языков программирования, таких как C#, Visual Basic, F# и другие.

Одним из главных преимуществ платформы .NET 7 является ее инновационность и способность следовать современным трендам в разработке программного обеспечения.

### 2.1.3 Язык программирования C#

За основу языка программирования в приложении взят C# – это объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он является одним из основных языков разработки приложений для платформы .NET. C# сочетает в себе элементы языков C и C++, но имеет более простой синтаксис и включает удобные средства для разработки приложений.

C# широко используется для разработки приложений на платформе .NET, включая приложения для Windows, веб-приложения, игры, мобильные приложения и многое другое. Он остается одним из самых популярных языков программирования благодаря своей простоте, мощности и широкой поддержке.

### 2.1.4 Расширяемый язык разметки XAML

XAML (Extensible Application Markup Language) - это язык разметки, используемый в технологии WPF (Windows Presentation Foundation) для описания структуры и внешнего вида элементов пользовательского интерфейса (UI) в декларативной форме. XAML является основным инструментом для создания пользовательского интерфейса в WPF.

XAML используется не только в WPF, но также в других технологиях, таких как Universal Windows Platform (UWP) и Xamarin.Forms, для описания пользовательского интерфейса.

### 2.1.5 Технология WPF

WPF (Windows Presentation Foundation) - это технология разработки графического интерфейса пользователя (GUI) для приложений под управлением операционной системы Windows. Она была разработана компанией Microsoft и является одной из основных частей платформы .NET. WPF предоставляет разработчикам мощные инструменты и возможности для создания современных и привлекательных пользовательских интерфейсов.

WPF использует язык разметки XAML (Extensible Application Markup Language), который позволяет описывать структуру и внешний вид элементов интерфейса в декларативной форме. Это упрощает процесс разработки и позволяет разработчикам и дизайнерам эффективно сотрудничать.

WPF применяет паттерн MVVM (Model-View-ViewModel), который способствует разделению логики приложения от представления. Это позволяет создавать более гибкие и тестируемые приложения.

### 2.1.6 Технология Entity Framework Core

Entity Framework Core (EF Core) - это современная технология доступа к данным в .NET, предоставляющая ORM (Object-Relational Mapping) функциональность для работы с реляционными базами данных. EF Core является

легковесной и переносимой версией Entity Framework, оптимизированной для работы с платформой .NET Core и .NET 5+.

Entity Framework Core представляет собой мощный инструмент для работы с данными в .NET, упрощающий и ускоряющий разработку приложений, связанных с базами данных.

### **2.1.7 Microsoft SQL Server**

Microsoft SQL Server (MS SQL Server) - это реляционная система управления базами данных (СУБД), разработанная компанией Microsoft. Она предоставляет средства для хранения, организации и обработки структурированных данных, которые могут быть использованы в приложениях и веб-сайтах. MS SQL Server базируется на реляционной модели данных, где данные организованы в таблицы с определенными связями и ограничениями целостности. Это позволяет эффективно хранить и извлекать данные.

MS SQL Server является одной из наиболее популярных реляционных СУБД и широко применяется в различных сферах, включая предприятия, веб-разработку, аналитику данных и многое другое.

### **2.1.8 Протокол SMTP**

SMTP (Simple Mail Transfer Protocol) - это стандартный протокол, используемый для отправки и доставки электронной почты через сети TCP/IP. Он предоставляет набор правил и команд для передачи электронных писем между почтовыми серверами.

SMTP обеспечивает надежную доставку электронной почты, выполняя следующие функции:

- отправка почты: SMTP клиент отправляет электронное письмо на SMTP сервер. Он устанавливает соединение с сервером, а затем передает письмо в соответствии с протоколом SMTP;

- маршрутизация: SMTP серверы маршрутизируют письма к целевым серверам, основываясь на информации о доменах получателя и настройках DNS (системы доменных имен);

- проверка подлинности: SMTP серверы могут использовать различные методы проверки подлинности отправителя, чтобы предотвратить спам и несанкционированную отправку писем;

- очередь доставки: если целевой сервер недоступен или временно недоступен, SMTP сервер может сохранить письмо в очереди доставки и повторить попытку доставки позже.

SMTP является одним из основных протоколов, используемых для отправки и доставки электронной почты в Интернете. Он работает совместно с другими протоколами, такими как POP3 (Post Office Protocol version 3) и IMAP (Internet Message Access Protocol), которые используются для получения писем с почтового сервера.

### 2.1.9 MD5 хеширование

Для обеспечения безопасности пользователей, при регистрации в базу данных записывается не сам пароль, а его хеш, используя технологию хеширования MD5.

MD5 (Message Digest Algorithm 5) - это криптографический хеш-алгоритм, который используется для преобразования произвольного входного сообщения в фиксированный хеш-код фиксированной длины, обычно 128 бит (16 байт). Хеш-код, полученный с помощью MD5, является уникальным представлением входного сообщения, и даже небольшое изменение в исходном сообщении приведет к значительному изменению хеш-кода.

MD5 хеширование широко используется в информационной безопасности и приложениях, где требуется проверка целостности данных. Он может быть использован для хеширования паролей, проверки целостности файлов, аутентификации и других целей.

### 2.1.10 Паттерн MVVM

MVVM (Model-View-ViewModel) - это шаблон проектирования, который используется в разработке программного обеспечения, особенно в контексте пользовательского интерфейса. Он разделяет компоненты приложения на три основных уровня: модель (Model), представление (View) и модель представления (ViewModel). Каждый уровень выполняет свои функции и имеет свою ответственность:

- модель (Model): это слой данных, который представляет бизнес-логику, хранение данных и взаимодействие с базой данных или внешними источниками данных. Модель содержит объекты данных, методы для получения и сохранения данных, а также логику обработки данных;

- представление (View): это слой пользовательского интерфейса, который отображает данные пользователю и обрабатывает пользовательские взаимодействия. Представление предоставляет пользовательский интерфейс, который визуализирует данные из модели и позволяет пользователю взаимодействовать с ними;

- модель представления (ViewModel): это слой, который связывает модель и представление. Модель представления предоставляет данные и логику, необходимую для отображения данных в представлении и обработки пользовательских взаимодействий. Он предоставляет свойства и команды, которые связываются с элементами пользовательского интерфейса в представлении, и обновляет модель при необходимости.

Основная идея MVVM заключается в разделении логики приложения и пользовательского интерфейса. Модель предоставляет данные и функциональность, представление отображает эти данные пользователю, а модель представления обеспечивает связь между моделью и представлением. Это позволяет улучшить разделение обязанностей, повысить переиспользуемость кода и упростить тестирование.

## 2.2 Спецификация функциональных требований к программному средству

Спецификация функциональных требований к программному средству для электронной библиотеки «Бумажный город» описывает основные функции и возможности, которые предоставляются пользователям и администраторам системы.

Были поставлены следующие функциональные требования к программному средству:

Для пользователя:

- регистрация;
- авторизация;
- просмотр каталога книг;
- поиск и фильтрация книг по заданным критериям;
- просмотр отдельных книг;
- возможность ознакомления с отзывами от других пользователей;
- добавление книг в корзину и их удаление;
- возможность покупки книг;
- получение информации о статусе заказа на почту;
- обратная связь с технической поддержкой;
- просмотр собственного профиля;
- изменение профиля;
- возможность смены темы или языка приложения.

Для администратора:

- авторизация;
- ответ на сообщения в техническую поддержку путем отправки сообщений на почту;
- просмотр каталога книг;
- просмотр отдельных книг;
- просмотр всех пользователей;
- возможность отправки сообщений на почту определенным пользователям;
- изменение, удаление и добавление книг;
- удаление отзывов у книг;
- просмотр статистики книг;
- возможность подтверждения или отказа в оплате книги;
- возможность смены языка или темы приложения.

В целом, спецификация функциональных требований является важным этапом в разработке программного средства, поскольку она позволяет четко определить цели и задачи проекта, установить требования пользователей и обеспечить успешную реализацию программного продукта.

Таким образом были представлены основные функциональные требования к разрабатываемому программному средству для электронной библиотеки "Бумажный город".



### 3 Проектирование программного средства

#### 3.1 Общая структура

Программное средство «Бумажный город» имеет следующую структуру, представленную на рисунке 3.1.

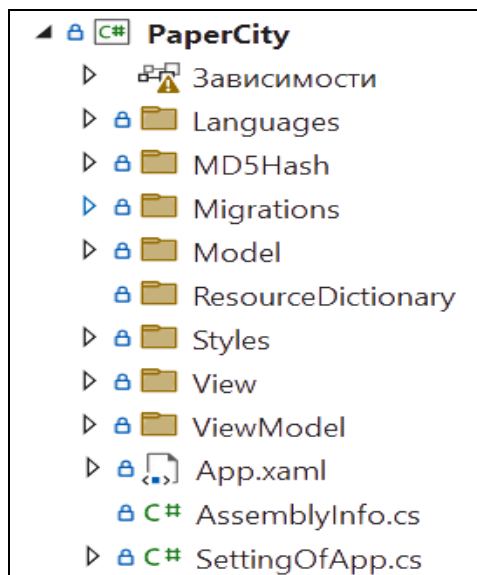


Рисунок 3.1 – Структура проекта

Описание структуры основных папок и файлов проекта представлено в таблице 3.1.

Таблица 3.1 – Описание структуры папок и файлов проекта

Имя пакета	Содержание
Папка Languages	В данной папке хранятся словари ресурсов для английской и русской версий приложения.
Папка MD5Hash	В данной папке хранится класс, предназначенный для хеширования пароля при регистрации и авторизации пользователя.
Папка Migrations	Папка, которая создается автоматически при миграции с базой данных.
Папка Model	Здесь описаны модели, с которыми происходит вся работа в приложении: <ul style="list-style-type: none"> <li>– клиент;</li> <li>– пользователь;</li> <li>– база данных;</li> <li>– объекты для работы с БД.</li> </ul>



Окончание таблицы 3.1

Имя пакета	Содержание
Папка View	В этой папке хранятся файлы с xaml-разметкой, то есть окна, страницы.
Папка ViewModel	В данной папке хранятся все классы для взаимодействия графической составляющей и бизнес логики приложения.
App.xaml	Файл, определяющий общие ресурсы приложения.
AssemblyInfo.cs	Файл содержит информацию о сборке (assembly), такую как версия, описание, автор и другие атрибуты.
SettingOfApp.cs	Файл содержит статический класс, который содержит в себе информацию о текущих настройках приложения, таких как текущая тема, язык и текущий пользователь.

Более подробная структура содержимого папки «Languages» программного средства представлена на рисунке 3.2.

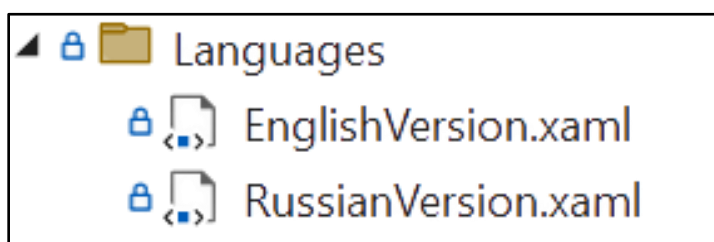


Рисунок 3.2 — Подробная структура папки «Languages»

Описание файлов данной папки представлено в таблице 3.2.

Таблица 3.2 – Описание файлов папки «Languages»

Имя файла/папки	Содержание
Файл EnglishVersion.xaml	Представляет собой словарь ресурсов, хранящий английскую версию приложения.
Файл RussianVersion.xaml	Представляет собой словарь ресурсов, хранящий русскую версию приложения.

При нажатии на кнопку смены языка и последующего выбора, т. е. английский или русский, в файле App.xaml подключается нужная пользователю версия приложения, что представляет собой подстановку нужного словаря ресурсов.

Подробная структура папки MD5Hash представлена на рисунке 3.3.

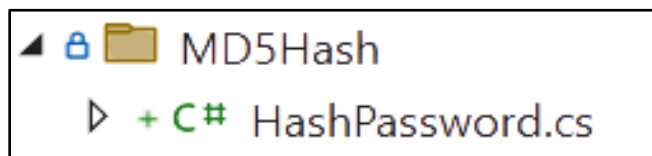


Рисунок 3.3 — Подробная структура папки «MD5Hash»

Описание файлов представлено в таблице 3.3.

Таблица 3.3 – Описание файлов папки «MD5Hash»

Имя файла	Содержание
Файл HashPassword.cs	Файл содержит класс, который имеет статический метод для получения хеша заданной строки по алгоритму MD5.

Подробная структура папки Migrations представлена на рисунке 3.4.

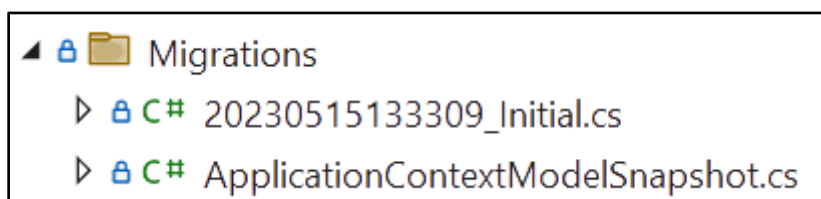


Рисунок 3.4 — Подробная структура папки «Migrations»

Описание файлов представлено в таблице 3.4.

Таблица 3.4 – Описание файлов папки «Migrations»

Имя файла	Содержание
Файл 20230515133309_Initial.cs	Файл миграции, который содержит код на языке программирования (обычно на C#), описывающий изменения, которые должны быть применены к базе данных для создания начальной схемы.
Файл ApplicationContextModelSnapshot.cs	Файл содержит информацию о текущей структуре базы данных, включая таблицы, поля и связи.

Миграция в Entity Framework Core является механизмом, который позволяет управлять изменениями в базе данных в процессе разработки приложения. Она позволяет автоматически создавать, обновлять и откатывать схему базы данных, чтобы она соответствовала моделям данных приложения.

Подробная структура папки Model представлена на рисунке 3.5.

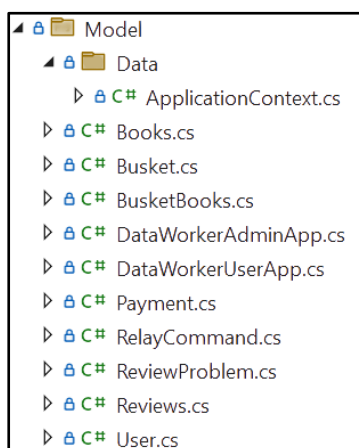


Рисунок 3.5 — Подробная структура папки «Model»

Описание файлов и внутренних папок папки представлено в таблице 3.5.

Таблица 3.5 – Описание файлов и внутренних папок папки «Model»

Имя файла/папки	Содержание
Папка Data	Папка хранит в себе класс для работы с базой данных.
Файл ApplicationContext.cs	Файл предназначен для подключения к базе данных и использования всех таблиц, который присутствуют в данной базе данных.
Файл Books.cs	Данный файл содержит класс Books, который используется для хранения данных о книгах.
Файл Busket.cs	Данный файл содержит класс Busket, который используется для хранения всех созданных корзин пользователей.
Файл BusketBooks.cs	Данный файл хранит класс BusketBooks, который используется для хранения книг, который есть у пользователей в корзинах.
Файл DataWorkerAdminApp.cs	Данный файл содержит класс DataWorkerAdminApp, который содержит функции для работы с базой данных в приложение администратора.
Файл DataWorkerUserApp.cs	Данный файл содержит класс DataWorkerUserApp, который содержит функции для работы с базой данных в приложение администратора.

## Окончание таблицы 3.5

Имя файла/папки	Содержание
Файл Payment.cs	Данный файл содержит класс Payment, который предназначен для хранения в себе информации о оплатах книг пользователями.
Файл RelayCommand.cs	Файл содержит класс RelayCommand, который реализует интерфейс ICommand, для реализации паттерна Command.
Файл ReviewProblem.cs	Файл содержит класс ReviewProblem, который предназначен для хранения сообщений от пользователей в техническую поддержку.
Файл Reviews.cs	Файл содержит класс Reviews, который предназначен для хранения всех отзывов к книгам.
Файл User.cs	Файл содержит класс User, который предназначен для хранения всех пользователей.

Подробная структура папки Styles представлена на рисунке 3.6.



Рисунок 3.6 — Подробная структура папки «Styles»

Описание файлов представлено в таблице 3.6.

Таблица 3.6 – Описание файлов папки «Styles»

Имя файла/папки	Содержание
Файл DartThemeLog.xaml	Файл представляет собой словарь ресурсов, в котором хранятся стили для темной темы приложения.
Файл LightThemeLog.xaml	Файл представляет собой словарь ресурсов, в котором хранятся стили для светлой темы приложения.

При нажатии на кнопку смены темы и последующего выбора, в файле App.xaml подключается нужная пользователю версия приложения, что представляет собой подстановку нужного словаря ресурсов.

Подробная структура папки View представлена на рисунке 3.7.

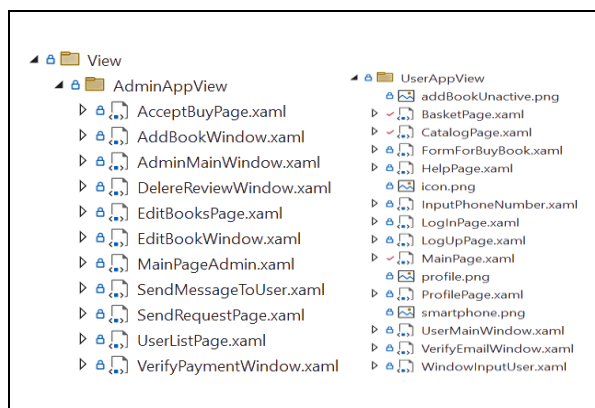


Рисунок 3.7 — Подробная структура папки «View»

Описание файлов и внутренних папок представлено в таблице 3.7.

Таблица 3.7 – Описание файлов и внутренних папок папки «View»

Имя файла/папки	Содержание
Папка AdminAppView	Папка хранит в себе окна и страницы, предназначенные для приложения администратора.
Файл AcceptBuyPage.xaml	Файл содержит в себе описание страницы, в которой идет перечисление всех оплат книг.
Файл AddBookWindow.xaml	Файл содержит в себе описание окна, в котором реализована форма для добавления книги.
Файл AdminMainWindow.xaml	Файл содержит в себе описание главного окна приложения администратора.
Файл DeleteReviewWindow.xaml	Файл содержит в себе определение окна, в котором удаляются отзывы.
Файл EditBooksPage.xaml	Файл содержит в себе определение для окна, в котором перечисляются все книги.
Файл EditBookWindow.xaml	Файл содержит в себе определение для окна, в котором реализована форма для редактирования книги.
Файл MainPageAdmin.xaml	Файл содержит в себе описание для главной страницы приложения администратора.
Файл SendMessageToUser.xaml	Файл содержит в себе описание для окна, в котором реализована форма для отправки сообщения пользователю.

Окончание таблицы 3.7

Имя файла/папки	Содержание
Файл SendRequestPage.xaml	Файл содержит в себе описание для окна, в котором реализована форма для отправки ответа на почту.
Файл UserListPage.xaml	Файл содержит описание страницы в которой содержится список пользователей.
Папка UserAppView	Папка содержит файлы для определения страниц и окон в приложении пользователя.
Файл BasketPage.xaml	Файл содержит определение страницы для корзины книг.
Файл CatalogPage.xaml	Файл содержит определение страницы для каталога книг.
Файл FormForBuyBook.xaml	Файл содержит определение окна, в котором описана форма для покупки книги.
Файл HelpPage.xaml	Файл содержит определение страницы для отправки письма в техническую поддержку.
Файл InputPhoneNumber.xaml	Файл содержит определение окна, в котором описана форма для изменения номера телефона.
Файл LogInPage.xaml	Файл содержит определение страницы для авторизации пользователя.
Файл LogUpPage.xaml	Файл содержит определение страницы для регистрации пользователя.
Файл MainPage.xaml	Файл содержит определение главной страницы у приложения пользователя.
Файл ProfilePage.xaml	Файл содержит определение страницы для отображения профиля пользователя.
Файл UserMainWindow.xaml	Файл содержит определение окна пользовательского приложения.
Файл VerifyEmailWindow.xaml	Файл содержит определение окна, в котором реализована форма для подтверждения электронной почты.
Файл WindowInputUser.xaml	Файл содержит определение главного окна, в котором пользователь может зарегистрироваться или авторизоваться.

В зависимости от того, под какой ролью зайдет пользователь, администратора или обычного пользователя, для него откроется определенное приложение, т. е. пользовательское приложение или же приложение администратора.

Подробная структура папки ViewModel представлена на рисунке 3.8.

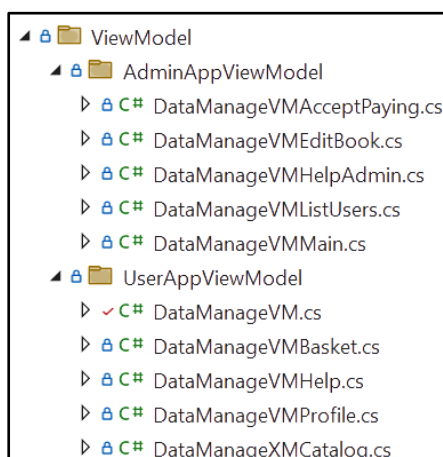


Рисунок 3.8 — Подробная структура папки «ViewModel»

Описание файлов и внутренних папок представлено в таблице 3.8.

Таблица 3.8 – Описание файлов и внутренних папок папки «ViewModel»

Имя файла/папки	Содержание
Папка AdminAppViewModel	Папка хранит в себе классы, которые описывают взаимодействие пользователя с графической частью приложения администратора и бизнес логикой.
Файл DataManageVMAcceptPaying.cs	Файл содержит класс, который описывает взаимодействие со страницей AcceptBuyPage.
Файл DataManageVMEditBook.cs	Файл содержит класс, который описывает взаимодействие со страницей EditBookPage.
Файл DataManageVMHelpAdmin.cs	Файл содержит класс, который описывает взаимодействие со страницей SendRequestPage.
Файл DataManageVMListUsers.cs	Файл содержит класс, который описывает взаимодействие со страницей UserListPage.
Файл DataManageVMMain.cs	Файл содержит класс, который описывает взаимодействие со страницей MainPageAdmin.
Папка UserAppViewModel	Папка хранит в себе классы, которые описывают взаимодействие пользователя с графической частью приложения пользователя и бизнес логикой.

## Окончание таблицы 3.8

Имя файла/папки	Содержание
Файл DataManageVM.cs	Файл содержит класс, который описывает взаимодействие с окном UserMainWindow.
Файл DataManageVMBasket.cs	Файл содержит класс, который описывает взаимодействие со страницей BasketPage.
Файл DataManageVMHelp.cs	Файл содержит класс, который описывает взаимодействие со страницей HelpPage.
Файл DataManageVMProfile.cs	Файл содержит класс, который описывает взаимодействие со страницей ProfilePage.
Файл DataManageVMCatalog.cs	Файл содержит класс, который описывает взаимодействие со страницей CatalogPage.

В целом, описание структуры проекта позволяет лучше понимать, как устроено программное средство и какие компоненты в нем присутствуют.

### 3.2 Взаимоотношения между классами

Для визуализации взаимосвязей между классами используется диаграмма UML.

UML (Unified Modeling Language) диаграмма классов - это графическое представление структуры и отношений между классами в системе. Она позволяет визуализировать классы, их атрибуты, методы и связи между ними. UML диаграмма классов является мощным инструментом для анализа, проектирования и документирования структуры программных систем. Она помогает разработчикам и аналитикам лучше понять взаимодействие классов в системе и способствует созданию более эффективного и модульного кода.

Для представления внутренней структуры программы в виде классов и связей между ними используется диаграмма классов. Приложение спроектировано таким образом, что каждый класс выполняет свои функции и практически не зависит от других. Диаграмма классов представлена в приложении А.

### 3.3 Модель базы данных

Для реализации поставленной задачи была создана база данных PAPERDB. Для ее создания использовалась система управления реляционными базами данных MS SQL Server. База данных состоит из таблиц, представленных на рисунке 3.9. Все таблицы выполняют определенную роль в приложении, и соответствуют требованиям, поставленным выше.

База данных разработана с учетом возможных изменений и расширений в будущем. Структура базы данных позволяет легко добавлять новые таблицы, поля или связи, а также вносить изменения в существующую структуру без необходимости значительных изменений в коде приложения.



В базе данных были определены ограничения целостности, которые обеспечивают правильность и согласованность данных. Это включает ограничения на значения полей, связи между таблицами и другие правила, которые гарантируют целостность данных в базе данных.

Скрипты для создания базы данных и всех таблиц представлен в приложении В.

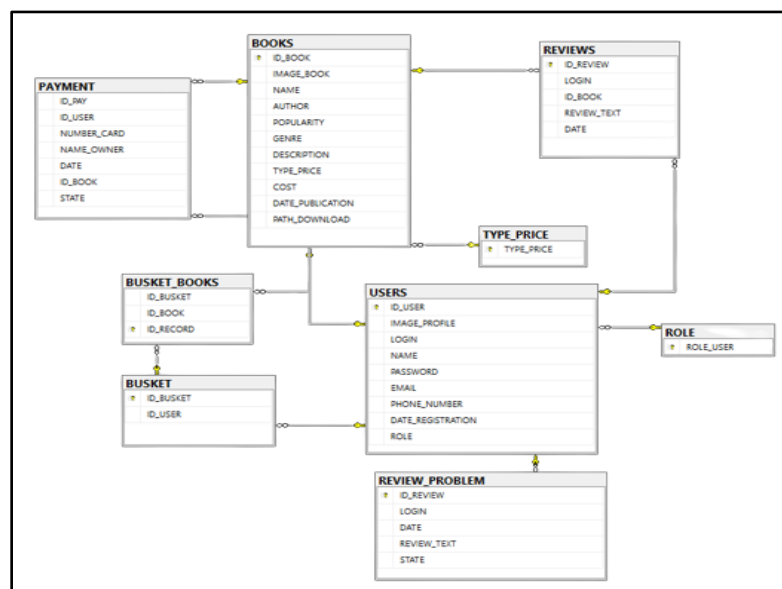


Рисунок 3.9 — База данных приложения «Бумажный город»

Таблица **USERS** изображена на рисунке 3.10. Данная таблица содержит в себе информацию о пользователе, которая представлена следующими столбцами: **ID\_USER**: уникальный номер пользователя, **IMAGE\_PROFILE**: путь к изображению пользователя, **LOGIN**: логин пользователя, **NAME**: личное имя пользователя, **PASSWORD**: хранит захешированный пароль пользователя, **EMAIL**: электронная почта, **PHONE\_NUMBER**: мобильный телефон, **DATE\_REGISTRATION**: дата регистрации, **ROLE**: роль текущего пользователя (обычный или администратор).

ID_USER	int
IMAGE_PROFILE	nvarchar(100)
LOGIN	nvarchar(40)
NAME	nvarchar(100)
PASSWORD	nvarchar(MAX)
EMAIL	nvarchar(70)
PHONE_NUMBER	nvarchar(14)
DATE_REGISTRATION	date
ROLE	nvarchar(20)

Рисунок 3.10 — Структура таблицы **USERS**

Таблица **BOOKS** изображена на рисунке 3.11. Данная таблица содержит в себе информацию о книгах, которая представлена следующими столбцами: **ID\_BOOK**: уникальный номер книги, **IMAGE\_BOOK**: содержит путь к книге, **NAME**: содержит

имя книги, AUTHOR: автор книги, POPULARITY: коэффициент популярности, GENRE: жанр книги, DESCRIPTION: описание книги, TYPE\_PRICE: тип книги (бесплатная, платная), COST: цена книги, DATE\_PUBLICATION: дата публикации книги в приложении, PATH\_DOWNLOAD: путь для скачивания книги.

🔑	ID_BOOK	int
	IMAGE_BOOK	nvarchar(100)
	NAME	nvarchar(100)
	AUTHOR	nvarchar(100)
	POPULARITY	int
	GENRE	nvarchar(100)
	DESCRIPTION	nvarchar(MAX)
	TYPE_PRICE	nvarchar(10)
	COST	real
	DATE_PUBLICATION	date
	PATH_DOWNLOAD	nvarchar(MAX)

Рисунок 3.11 — Структура таблицы BOOKS

Таблица BUSKET изображена на рисунке 3.12. Данная таблица содержит в себе перечисление соотношения корзины книг к пользователям и состоит из следующих столбцов: ID\_BUSKET: уникальный номер корзины, ID\_USER: уникальный номер пользователя.

🔑	ID_BUSKET	int
	ID_USER	int

Рисунок 3.12 — Структура таблицы BUSKET

Таблица BUSKET\_BOOKS изображена на рисунке 3.13. Данная таблица содержит в себе перечисление соотношения корзины и книг и состоит из следующих столбцов: ID\_BUSKET: уникальный номер корзины, ID\_BOOK: уникальный номер книги, ID\_RECORD: номер записи.

	ID_BUSKET	int
	ID_BOOK	int
🔑	ID_RECORD	int

Рисунок 3.13 — Структура таблицы BUSKET\_BOOKS

Таблица PAYMENT изображена на рисунке 3.14. Данная таблица содержит в себе список оплат книг и состоит из следующих столбцов: ID\_PAY: уникальный номер оплаты, ID\_USER: уникальный номер пользователя, NUMBER\_CARD: номер банковской карты, NAME\_OWNER: имя владельца карты, DATE: дата действия

карты, ID\_BOOK: уникальный номер книги, STATE: состояние платежа (подтверждено, отказано, в ожидании).

ID_PAY	int
ID_USER	int
NUMBER_CARD	nvarchar(MAX)
NAME_OWNER	nvarchar(MAX)
DATE	nvarchar(MAX)
ID_BOOK	int
STATE	nvarchar(MAX)

Рисунок 3.14 — Структура таблицы PAYMENT

Таблица REVIEW\_PROBLEM изображена на рисунке 3.15. Данная таблица содержит в себе список сообщений, отправленных в техническую поддержку и состоит из следующих столбцов: ID\_REVIEW: уникальный номер сообщения, LOGIN: логин пользователя, DATE: дата отзыва, REVIEW\_TEXT: само сообщение, STATE: состояние ответа (в ожидании, отвечено).

ID_REVIEW	int
LOGIN	nvarchar(40)
DATE	date
REVIEW_TEXT	nvarchar(MAX)
STATE	nvarchar(MAX)

Рисунок 3.15 — Структура таблицы REVIEW\_PROBLEM

Таблица REVIEWS изображена на рисунке 3.16. Данная таблица содержит в себе список отзывов к книгам и состоит из следующих столбцов: ID\_REVIEW: уникальный номер отзыва, LOGIN: логин пользователя, ID\_BOOK: уникальный номер книги, REVIEW\_TEXT: сам отзыв, DATE: дата отзыва.

ID_REVIEW	int
LOGIN	nvarchar(40)
ID_BOOK	int
REVIEW_TEXT	nvarchar(MAX)
DATE	datetime

Рисунок 3.16 — Структура таблицы REVIEWS

Таблица ROLE изображена на рисунке 3.17. Данная таблица представляет собой перечисление статусов пользователей и состоит из одного столбца: ROLE\_USER: статус пользователя (обычный, администратор).

ROLE_USER	nvarchar(20)
-----------	--------------

Рисунок 3.17 — Структура таблицы ROLE

Таблица TYPE\_PRICE изображена на рисунке 3.18. Данная таблица представляет собой перечисление типов книг по цене и состоит из одного столбца:

TYPE\_PRICE: тип книги по цене (бесплатная, платная).


	TYPE_PRICE	nvarchar(10)
---	------------	--------------

Рисунок 3.17 — Структура таблицы TYPE\_PRICE

База данных была спроектирована с учетом принципов нормализации данных, что позволяет избежать избыточности и несогласованности данных. Каждая таблица имеет четко определенные поля и связи с другими таблицами для эффективного хранения и управления информацией.

Таким образом, структура базы данных «PAPERDB» позволяет разработать приложение таким образом, чтобы все требования, установленные выше, были соблюдены. Данная база данных позволит хранить информацию о пользователях, книгах, отзывах и прочем, что необходимо в данном приложении.

### 3.4 Проектирование архитектуры приложения

Для описания функциональности системы и того, как она используется различными группами пользователей, применяется диаграмма использования. Эта диаграмма представляет информацию о том, какие возможности и функции доступны каждой группе пользователей. В диаграмме использования используются два основных типа элементов: варианты использования и актёры.

Актёры представляют разные группы пользователей системы и могут быть любыми сущностями, которые взаимодействуют с системой. Варианты использования, с другой стороны, представляют собой функции или действия, которые пользователи могут выполнить в системе. Каждый вариант использования определяет набор шагов, которые актёры могут выполнить для взаимодействия с системой, и описывает, какие действия системы выполняются в ответ на эти действия пользователей.

Диаграмма использования является эффективным средством коммуникации между разработчиками, заказчиками и заинтересованными сторонами. Она помогает визуально представить функциональные требования системы и облегчает обсуждение и понимание функциональности между участниками проекта.

В целом, диаграмма использования является полезным инструментом для анализа требований, проектирования системы и коммуникации между участниками проекта. Она помогает лучше понять функциональность системы и ее взаимодействие с пользователями, что способствует успешной разработке программного продукта.

Диаграмма использования представлена в приложении Б.

### 3.5 Проектирование последовательностей проекта

Для наглядного представления взаимодействия между объектами системы в различные моменты времени, когда выполняется определенный сценарий

использования, применяется диаграмма последовательностей в языке моделирования UML. Эта диаграмма позволяет наглядно показать, как объекты системы обмениваются сообщениями и взаимодействуют друг с другом для достижения определенной цели. Она отображает порядок и временную последовательность этих взаимодействий, позволяя лучше понять, как система функционирует в рамках конкретного сценария использования.

Для наглядного отображения временной последовательности взаимодействия объектов в системе используется диаграмма последовательностей в языке моделирования UML. На этой диаграмме каждый объект представлен вертикальной линией, называемой "линией жизни", которая показывает его существование в системе в определенный период времени. Объекты изображаются в виде прямоугольников, а сообщения между ними представлены стрелками или линиями.

Диаграмма последовательностей помогает наглядно представить взаимодействие между объектами и последовательность выполняемых ими действий во времени. Она является полезным инструментом для анализа и проектирования системы, а также служит средством коммуникации между разработчиками и заинтересованными сторонами, позволяя лучше понять, как система функционирует в различных сценариях использования.

Пример диаграммы последовательности представлен на рисунке 3.18.

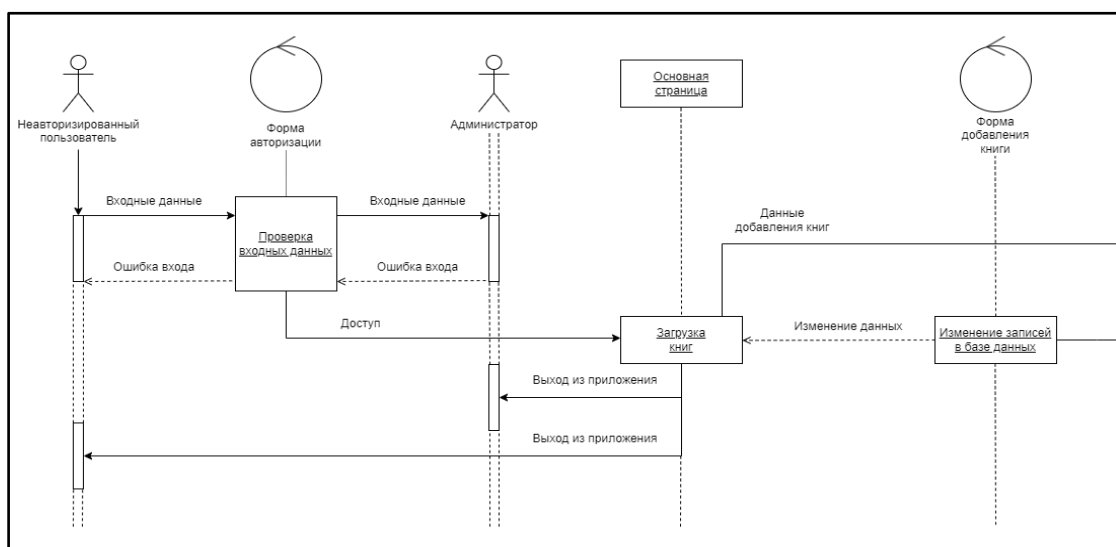


Рисунок 3.18 — Диаграмма последовательности

Пример диаграммы последовательности отображает процесс авторизации и загрузки книги из приложения.

## 4 Реализация программного средства

### 4.1 Основные классы программного средства

Для выполнения технических задач программного средства «Бумажный город» должны быть реализованы следующие функции:

- авторизация;
- регистрация;
- просмотр книг;
- скачивание книги;
- оплата книги;
- добавление отзыва к книге;
- просмотр профиля;
- заполнение формы заказа книги;
- редактирование профиля;
- возможность оставить отзыв к книге;
- функционирование приложения администратора.

Далее подробно рассмотрены каждые из необходимых для выполнения технических задач функции, а также созданные для их выполнения классы и методы и их функционал, и реализация.

### 4.2 Описание классов и методов программного средства

#### 4.2.1 Авторизация

Необходимый функционал для возможности входа пользователя в аккаунт, а также для проверки корректности данных, реализован в классе `DataManageVM`. После нажатия кнопки «Авторизоваться» сработает команда «`LogInUser`».

В классе `DataManageVM` команда `LogInUser` вызывает метод `LogInMethod`, который отправляет запрос к базе данных на совпадение комбинации логина и хеша пароля. Код метода `LogInMethod` изображен в листинге 4.1.

```
public void LogInMethod(LogInPage page)
{
    DataWorkerUserApp.LogInUserGet (page.LoginText.Text,
    HashPassword.GetHashPassword (page.PasswordText.Text) );

    Application.Current.Windows.OfType<WindowInputUser>().FirstOrDefault
    ().Close();
}
```

Листинг 4.1 — Метод `LogInMethod`

В случае успешного запроса к базе данных проверяется роль пользователя, т. е. если пользователь имеет роль «общий», то для него открывается пользовательское

приложение, иначе приложение администратора. Также, при авторизации пользователь может указать язык и тему приложения, под которым он хочет продолжать поиск книги. Разметка формы представлена в приложении В.

### 4.2.2 Регистрация

Для совершения пользователем регистрации также предназначен класс `DataManageVM`.

При нажатии на кнопку «Зарегистрироваться», сработает команда `RegisterNewUser`. Код данной команды отображен в листинге 4.2.

```
public RelayCommand RegisterNewUser
{
    get
    {
        return registerNewUser ?? new RelayCommand(async obj
=>
        {
            Random rand = new Random();
            int code = rand.Next(100000, 999999);
            await
PutMessageEmail(Convert.ToString(code), wind.EmailText.Text);
            VerifyEmailWindow windd = new
VerifyEmailWindow(code);
            if (windd.ShowDialog() == true)
            {
                DataWorkerUserApp.RegisterNewUser(wind.LoginText.Text,
                HashPassword.GetHashPassword(wind.PasswordTextStars.Password),
                wind.PNText.Text, wind.EmailText.Text);
                MessageBox.Show("Вы успешно
зарегистрированы!", "Registration type", MessageBoxButton.OK);
            }
            else
            {
                MessageBox.Show("Вы не подтвердили
Email! Вы не зарегистрированы!", "ErrorRegister",
MessageBoxButton.OK, MessageBoxImage.Error);
            }
        }
    });
}
```

Листинг 4.2 — Команда `LogUpUser`

В данной команде происходит генерация случайного 6-тизначного числа для отправки кода подтверждения электронной почты. Асинхронный метод `PutMessageEmail` отправляет сообщение на электронную почту пользователя в

другом потоке, после чего открывается окно подтверждения, куда пользователь должен ввести код. Если пользователь введет правильный код, то его регистрация будет успешной. Также, после регистрации пароль пользователя преобразуется в хеш по алгоритму MD5 хеширования, для обеспечения безопасности. После регистрации пользователь должен авторизоваться в приложении. Разметка формы, хеширование и метод отправки сообщения на почту представлена в приложении В.

### **4.2.3 Просмотр книг**

После успешной авторизации пользователю открывается окно с главной страницей, где выводятся книги, разделенные по разным критериям.

Просмотр книг должен быть доступен сразу на 3-ех страницах в пользовательском приложении, поэтому для них реализованы методы в предназначенных для этих страниц классов DataManageVM. При двойном клике на определенную книгу действие приложение переходит в команду ClickBook, которая в свою очередь вызывает метод ClickOnBook.

Метод ClickOnBook заполняет данными объект bookNow, для отображения книги, на которую нажал пользователь и после выводит ее на той же странице.

### **4.2.4 Скачивание книги**

Скачивание книги доступно только после ее добавления в корзину.

При нажатии на кнопку «Скачать», срабатывает команда DownloadBook, описанная в классе DataManageBasket. Данная команда вызывает метод DownloadBookMethod, в котором описан алгоритм скачивания определенной книги. Однако, если книга платная, то перед ее скачиванием придется заполнить форму оплаты и ждать подтверждения администратора, только после этого пользователь сможет скачать книгу.

### **4.2.5 Оплата книги**

Как было сказано ранее, если книга является платной, то перед ее скачиванием придется заполнить форму оплаты, после чего ждать подтверждения оплаты администратора приложения.

Графическая часть формы оплаты представлена в классе FormForBuyBook, где данные проверяются на корректность, и в случае успеха оплата книги сохраняется в базу данных. После подтверждения оплаты администратором, пользователь сможет войти и скачать данную книгу. В случае отказа, пользователь сможет увидеть уведомление на электронной почте, пользователь может заново произвести оплату.

### **4.2.6 Добавление отзыва к книге**

После добавления книги в корзину, пользователь может оставить отзыв. Для этого отображается текстовое поле для отзыва, и кнопка «Оставить отзыв». После написания отзыва в текстовое поле и нажатия кнопки, срабатывает команда



AddReview, которая в свою очередь вызывает метод AddReviewMethod. Данный метод описан в классе DataManageVMBasket. Если поле отзыва не пустое, то отзыв успешно записывается в базу данных и доступен всем пользователям приложения.

#### 4.2.7 Просмотр профиля

При нажатии на кнопку профиля в боковом меню приложения, пользователю открывается страница его профиля. На данной странице отображаются данные введенные пользователем при регистрации. Пользователю доступны две кнопки: «Изменить изображение» и «Изменить». При нажатии на первую кнопку срабатывает команда ChangeImage, описанный в классе DataManageVMProfile, которая описывает алгоритм изменения фотографии профиля и изменения данных в базе данных. Вторая кнопка предназначена для изменения номера телефона, при ее нажатии открывается форма InputPhoneNumber, куда вводится номер телефона, если номер телефона корректен, то данные пользователя перезапишутся.

#### 4.2.8 Приложение администратора

Для администраторов открывается собственное приложение, имеющее боковую панель с различными кнопками. Приложение отображает список всех книг, пользователей, сообщений в техническую поддержку и список оплат книг.

На странице EditBooksPage выводится список всех книг библиотеки. При двойном нажатии на книгу отображается заданная книга. Страница книги разделена на 2 составляющие: сама книга и отзывы к ней. В самой книге администратор может изменить книгу или удалить ее. Во вкладке отзывы администратор может удалить отзывы, которые не поддерживаются правилами приложения.

На странице UserListPage администратору выводится список всех обычных пользователей, которые зарегистрированы в приложении. При двойном нажатии на определенного пользователя, администратор может увидеть информацию о нем. Также ему отображается единственная кнопка «Отправить сообщению пользователю». При нажатии на эту кнопку, администратор может отправить письмо пользователю.

На странице AcceptBuyPage администратор видит список всех неподтвержденных оплат книг. При двойном нажатии на определенные данные, администратор может либо подтвердить оплату, либо отказать в ней.

На странице SendRequestPage администратор видит список всех сообщений в техническую поддержку. При двойном нажатии на сообщение открывается окно, в котором вводится заголовок сообщения и его текст и нажимает кнопку «Отправить». После этого сообщение помечается как отвеченное, и пропадает из списка.

Также страница AddBookPage имеет кнопку добавления книги, после нажатия которой открывается новое окно со всеми полями, которые необходимы для добавления книги. После корректного заполнения всех полей, и нажатия кнопки «Добавить» книга успешно добавляется в базу данных и становится доступной всем пользователям.

## 5 Тестирование, проверка работоспособности и анализ полученных результатов

### 5.1 Тестирование авторизации и регистрации

В момент авторизации и регистрации возможна ситуация, когда пользователь вводит некорректные данные, например, неверный пароль, незарегистрированный логин, некорректный адрес электронной почты или номер телефона. Также пользователь может вводить данные, которые уже были определены, например, логин, который есть уже у другого пользователя, или же электронную почту, которая уже используется в приложении. Такие исключения обрабатываются программным средством с помощью показа пользователю всплывающих сообщений с текстом ошибки.

В случаях неправильного ввода пароля, логина или в случае пустых полей, в форме отображается всплывающее окно с соответствующим текстом. Пример тестирования в случае пустых полей или неправильной комбинации логина и пароля изображен на рисунке 5.1.

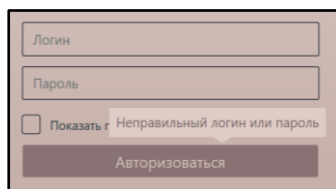


Рисунок 5.1 — Результат первого тестирования авторизации

При регистрации надо указывать следующие поля: логин, электронная почта, личное имя, пароль. Каждое поле данной страницы обрабатывает собственные ошибки.

Правило для формирования логина: можно использовать буквы как русского, так и английского алфавитом, также можно использовать цифры, минимальная длина 5 символов, не может быть 2-ух повторяющихся логинов.

Результат тестирования при вводе логина длиной менее 5-ти символов приведен на рисунке 5.2.

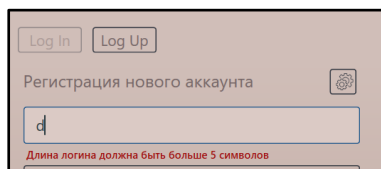


Рисунок 5.2 — Результат первого тестирования логина

В данных полях регистрации под самим полем будет выводиться сообщение об характерной ошибке.

Результат тестирования при вводе в поле логина недопустимых символов приведен на рисунке 5.3.

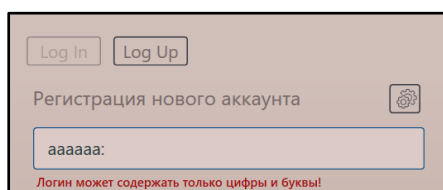


Рисунок 5.3 — Результат второго тестирования логина

Результат тестирования в случае пустого поля логина приведен на рисунке 5.4.

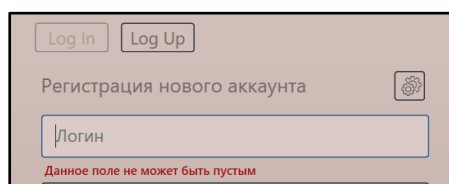


Рисунок 5.4 — Результат третьего тестирования логина

Результат тестирования в случае повторного логина приведен на рисунке 5.5.

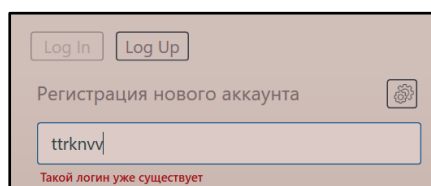


Рисунок 5.5 — Результат четвертого тестирования логина

Правило для записи электронной почты: имя почты может содержать буквы английского алфавита, цифры и различные символы, обязательно должен быть символ «@» после которого идет корректный домен почты, также не может быть повторяющихся почт.

Результат тестирования при вводе почты без символа «@» приведен на рисунке 5.6.

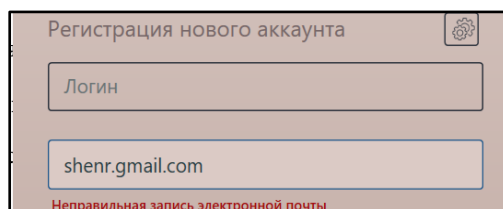
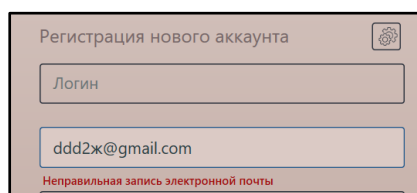


Рисунок 5.6 — Результат первого тестирования почты

Дополнительно в проверке валидности электронной почты идет проверка на существование данного домена.

Результат тестирования при вводе почты с недопустимыми символами приведен на рисунке 5.7.



Регистрация нового аккаунта

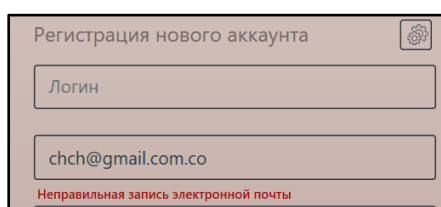
Логин

ddd2ж@gmail.com

Неправильная запись электронной почты

Рисунок 5.7 — Результат второго тестирования почты

Результат тестирования при вводе неправильного домена приведен на рисунке 5.8.



Регистрация нового аккаунта

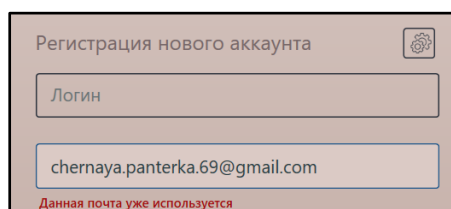
Логин

chch@gmail.com.co

Неправильная запись электронной почты

Рисунок 5.8 — Результат третьего тестирования почты

Результат тестирования при повторной почте приведен на рисунке 5.9.



Регистрация нового аккаунта

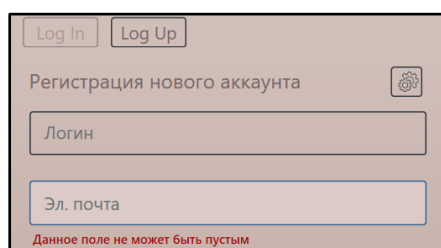
Логин

chernaya.panterka.69@gmail.com

Данная почта уже используется

Рисунок 5.9 — Результат четвертого тестирования почты

Результат тестирования при пустом поле для ввода почты приведен на рисунке 5.10.



Log In Log Up

Регистрация нового аккаунта

Логин

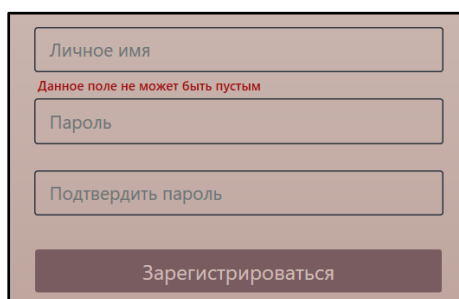
Эл. почта

Данное поле не может быть пустым

Рисунок 5.10 — Результат пятого тестирования почты

Правила для заполнения поля личного имени: может состоять только из букв английского и русского алфавитов.

Результат тестирования при пустом поле для ввода личного имени приведен на рисунке 5.11.



Личное имя

Данное поле не может быть пустым

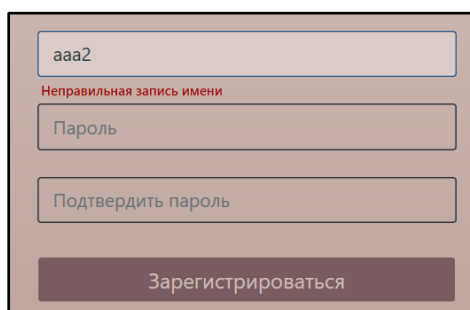
Пароль

Подтвердить пароль

Зарегистрироваться

Рисунок 5.11 — Результат первого тестирования личного имени

Результат тестирования при вводе недопустимых символов в поле личного имени приведен на рисунке 5.12.



aaa2

Неправильная запись имени

Пароль

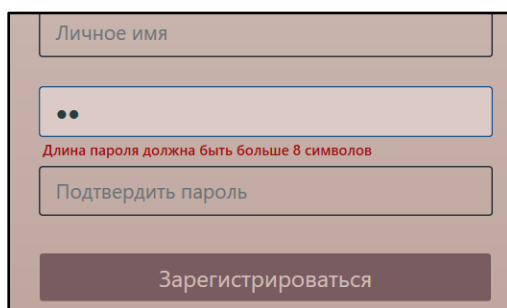
Подтвердить пароль

Зарегистрироваться

Рисунок 5.12 — Результат второго тестирования личного имени

Правила для заполнения пароля: пароль может содержать различные символы, но обязательна прописная и маленькая буква, также обязательно наличие цифр, минимальная длина пароля 8 символов.

Результат тестирования при вводе маленькой длины пароля приведен на рисунке 5.13.



Личное имя

••

Длина пароля должна быть больше 8 символов

Подтвердить пароль

Зарегистрироваться

Рисунок 5.13 — Результат первого тестирования пароля

Пользователь должен сам придумать пароль как можно надежнее, опираясь на правила составления пароля в приложении. Но также не стоит придумывать пароль слишком большой длины.

Результат тестирования при вводе пароля, который не соответствует правилам, приведен на рисунке 5.14.

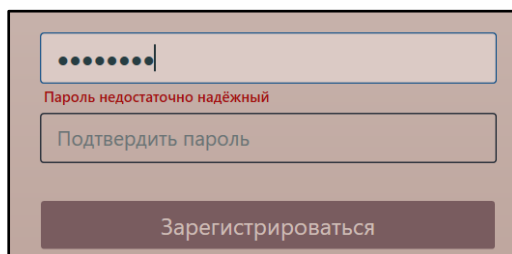
A screenshot of a web registration form. It features a password input field with a strength indicator (dots) and a red error message below it: "Пароль недостаточно надёжный". Below the password field is a "Подтвердить пароль" (Confirm password) field. At the bottom of the form is a large "Зарегистрироваться" (Register) button.

Рисунок 5.14 — Результат второго тестирования пароля

Результат тестирования при вводе пароля для подтверждения, который не соответствует первому паролю, приведен на рисунке 5.15.

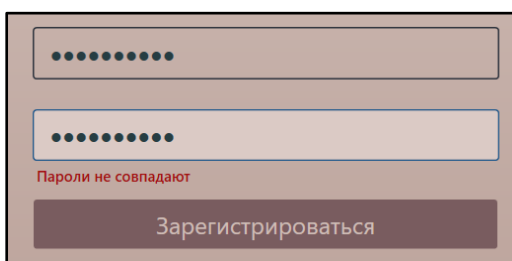
A screenshot of a web registration form. It features two password input fields. Below the second field is a red error message: "Пароли не совпадают" (Passwords do not match). At the bottom of the form is a large "Зарегистрироваться" (Register) button.

Рисунок 5.14 — Результат третьего тестирования пароля

Также, если пользователь не подтвердит свою электронную почту, то он также не сможет зарегистрироваться. Пример данного тестирования представлен на рисунке 5.15.

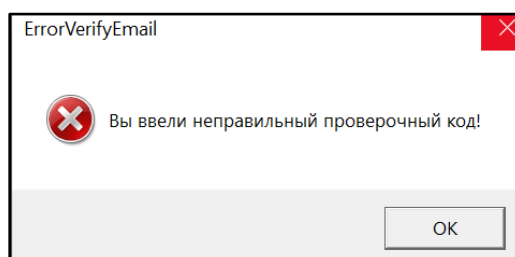


Рисунок 5.15 — Результат тестирования подтверждения почты

Таким образом было проведено успешное тестирование форм авторизации и регистрации.

## 5.2 Тестирование оплаты книги

При оплате книги пользователь должен вводить следующие поля: номер карты, срок действия карты и имя владельца.

Правила при заполнении номера карты: в приложении принимаются номера карт следующих типов: Visa, MasterCard, Maestro, Mir.

Результат тестирования при неправильном вводе карты приведен на рисунке 5.16.

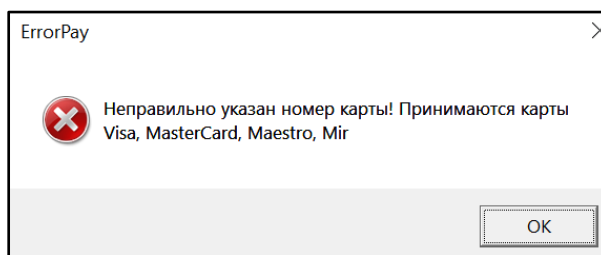


Рисунок 5.16 — Результат тестирования номера карты

Правила при заполнении даты: формат даты должен соответствовать формату MM/DD, где первая часть должны быть в промежутке от 1 до 12.

Результат тестирования при неправильном вводе даты приведен на рисунке 5.17.

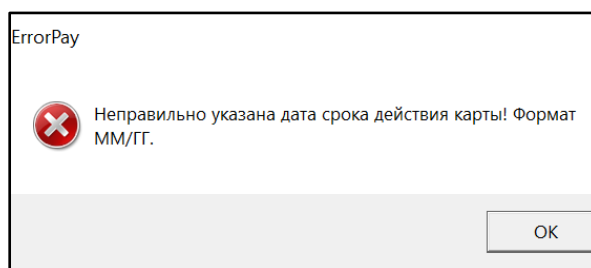


Рисунок 5.17 — Результат тестирования даты

Правила при заполнении имени владельца: допустимы только буквы английского алфавита.

Результат тестирования при неправильном вводе имени владельца приведен на рисунке 5.18.

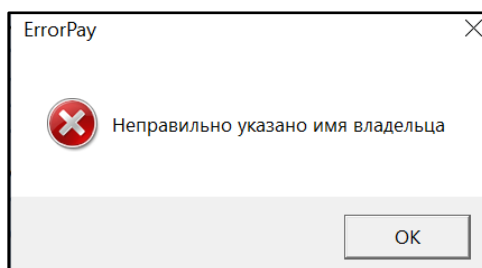


Рисунок 5.18 — Результат тестирования имени владельца

Также стоит рассмотреть различные случаи возникновения ошибок приложения, таких как повторное добавление книг, ввод неправильного номера и тому подобное.

Результат тестирования при повторном добавлении книги приведен на рисунке 5.19.

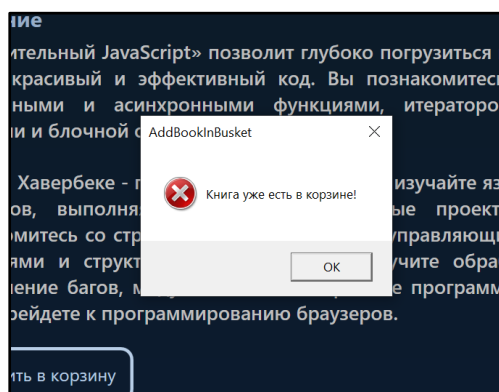


Рисунок 5.19 — Результат тестирования имени владельца

Результат тестирования при повторной оплате книги приведен на рисунке 5.20.

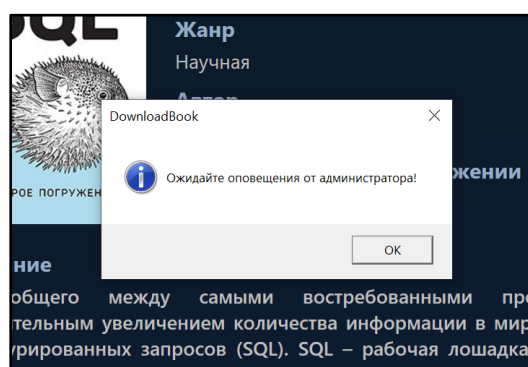


Рисунок 5.20 — Результат тестирования повторной оплаты

Результат тестирования при неправильном вводе мобильного телефона приведен на рисунке 5.21.

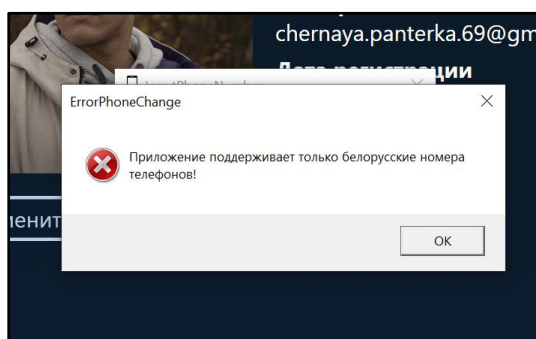


Рисунок 5.21 — Результат тестирования номера телефона

Так же следует проверить попытки отправки пустых отзывов, и попытки скачивания книг, которых нет в корзине.



Результат тестирования при попытке отправки пустого отзыва приведен на рисунке 5.22.

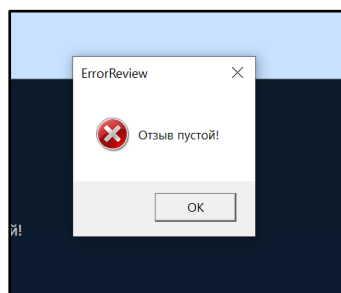


Рисунок 5.22 — Результат тестирования отправки отзыва

Результат тестирования при попытке сохранения книги, которая была удалена из корзины, приведен на рисунке 5.23.

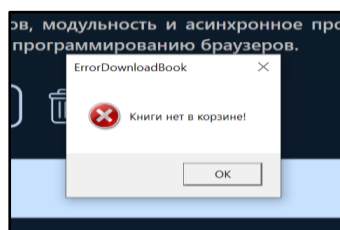


Рисунок 5.23 — Результат тестирования скачивания книги

Таким образом было проведено успешное тестирование пользовательского приложения.

### 5.3 Тестирование приложения администратора

В приложении администратора также возможны исключительные ситуации, которые нужно отловить и сообщить об ошибке. Результаты тестирования таких ситуаций представлены ниже.

Результат тестирования при попытке отправления сообщения без заголовка приведен на рисунке 5.24.

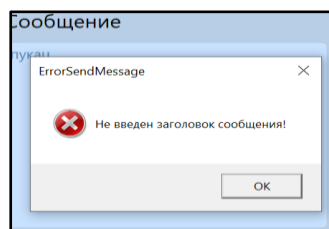


Рисунок 5.24 — Результат первого тестирования отправки сообщений

Также администратор может случайно нажать кнопку «Отправить» без указания сообщения, что может привести к ошибке.

Результат тестирования при попытке отправления сообщения без самого сообщения приведен на рисунке 5.25.

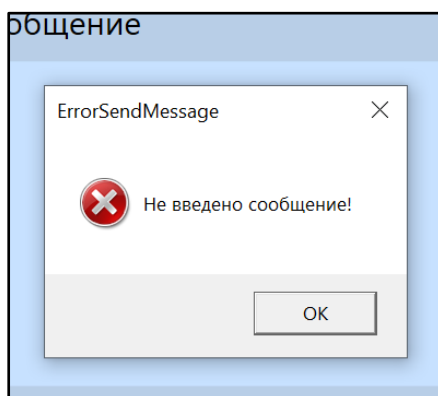


Рисунок 5.25 — Результат второго тестирования отправки сообщений

Результат тестирования при попытке добавления книги без изображения приведен на рисунке 5.26.

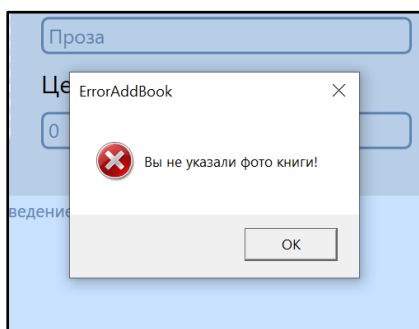


Рисунок 5.26 — Результат первого тестирования добавления книги

Результат тестирования при попытке добавления книги без пути к ее скачиванию приведен на рисунке 5.27.

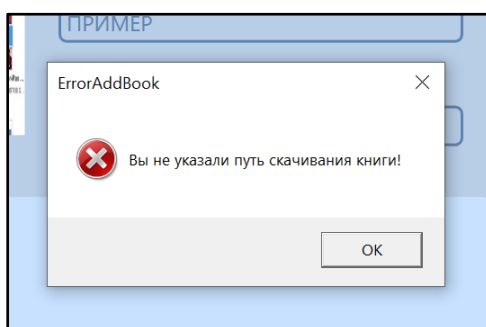


Рисунок 5.27 — Результат второго тестирования добавления книги

Все последующие возможные ошибки в приложении были успешно протестированы. По результатам тестирования, можно сказать, что приложения пользователя и администратора работают корректно.

## 6 Руководство по установке и использованию

При первом запуске приложения будет открыто окно авторизации, которое показано на рисунке 6.1.

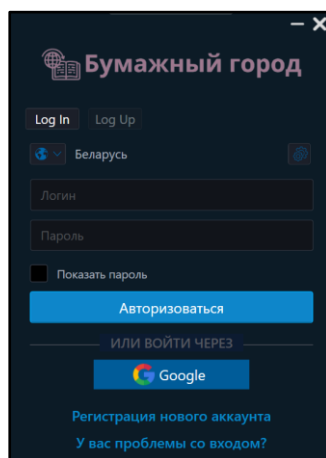


Рисунок 6.1 — Окно авторизации

В том случае, если у пользователя нет зарегистрированного аккаунта, он может нажать на кнопку «Log Up», и пройти там регистрацию.

После нажатия на кнопку «Log Up» откроется страница для регистрации, в котором все поля должны быть заполнены в соответствии с правилами: логин: буквы английского и русского алфавитов, а также цифры, электронная почта: корректный адрес электронной почты, на которую придёт проверочный код, персональное имя: только буквы английского и русского алфавитов, пароль: разные символы, но обязательно наличие прописных и строчных букв, а также цифр. После регистрации пароль пользователя захешируется, что дает дополнительную безопасность аккаунту пользователя. Окно регистрации изображено на рисунке 6.2.

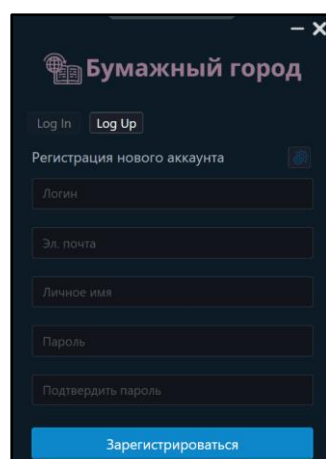


Рисунок 6.2 — Окно регистрации

После регистрации пользователь должен войти с свой аккаунт, нажав на кнопку «Log In». Нового пользователя сразу встречает главная страница

пользовательского приложения, с подборками книг по различным критериям. Слева располагается боковое меню, которое имеет кнопки перемещения по страницам и кнопку «Выйти».

Главная страница представлена на рисунке 6.3.

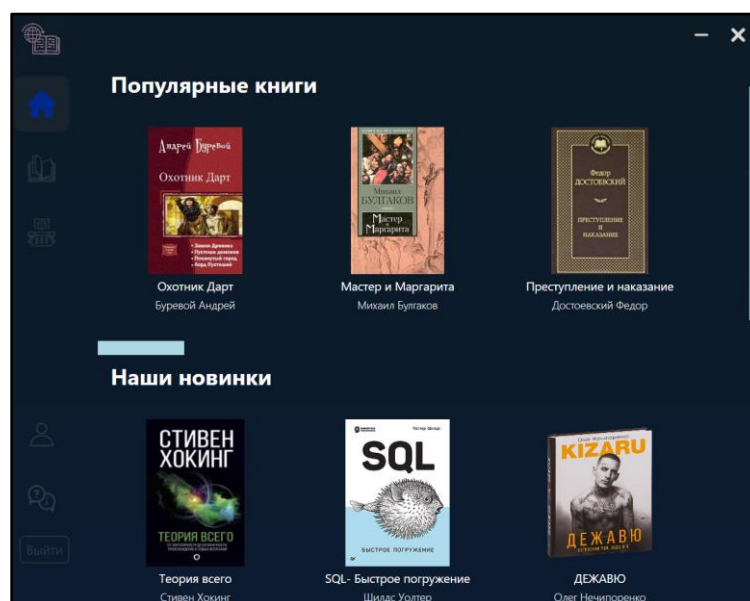


Рисунок 6.3 — Главная страница

Следующая кнопка под главной страницей переведет пользователя на новую страницу каталога книг. Каталог книг представлен на рисунке 6.4.

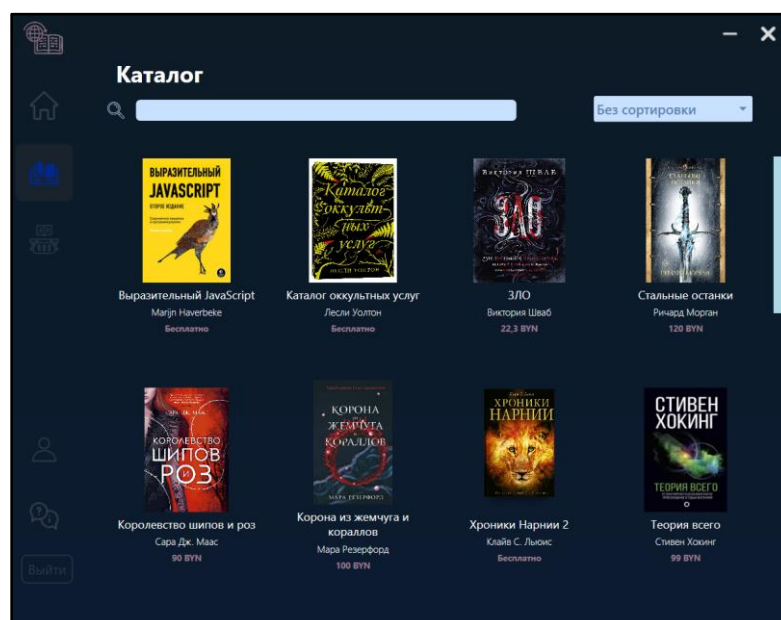


Рисунок 6.4 — Каталог книг

Каталог книг имеет при себе поисковую строку, вводя название книги, автора, или жанр будут выводиться определенные книги. Также есть выпадающий список,

который позволит отсортировать книги по определенным критериям. При нажатии на книгу на главной странице, или же на книгу в каталоге откроется страница с отображением информации о выбранной книге. Пример представлен на рисунке 6.5.

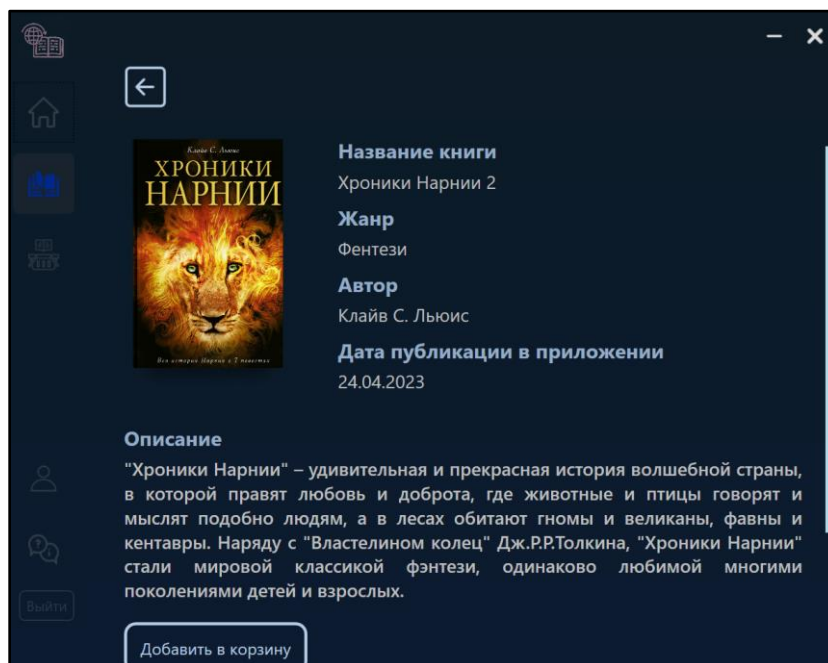


Рисунок 6.5 — Отображение текущей книги

Страница текущей книги содержит информацию о книге и кнопки «Добавить в корзину» и кнопка назад. При нажатии на кнопку «Добавить в корзину» книга добавляется в корзину пользователя, если у него нет такой книги. При нажатии на кнопку назад, пользователя перекидывает на предыдущую страницу. Также данная страница имеет список отзывов, который представлен на рисунке 6.6.

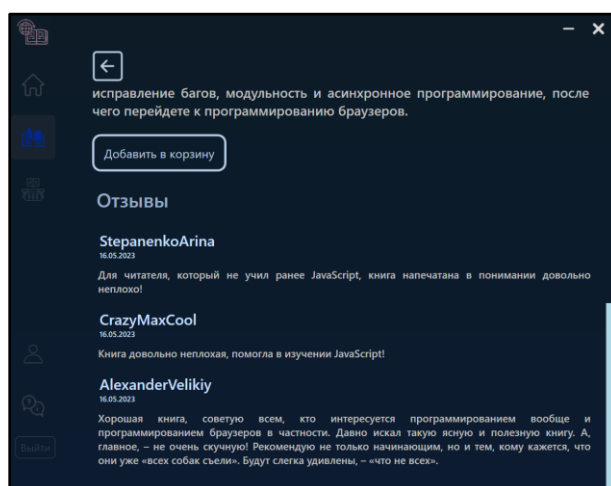


Рисунок 6.6 — Отзывы к книге

Отзыв содержит логин пользователя, который оставил его, дату отзыва и сам отзыв.

Следующая кнопка — это корзина книг, перейдя в которую пользователю может просмотреть все книги, которые он добавил к себе. Корзина изображена на рисунке 6.7.

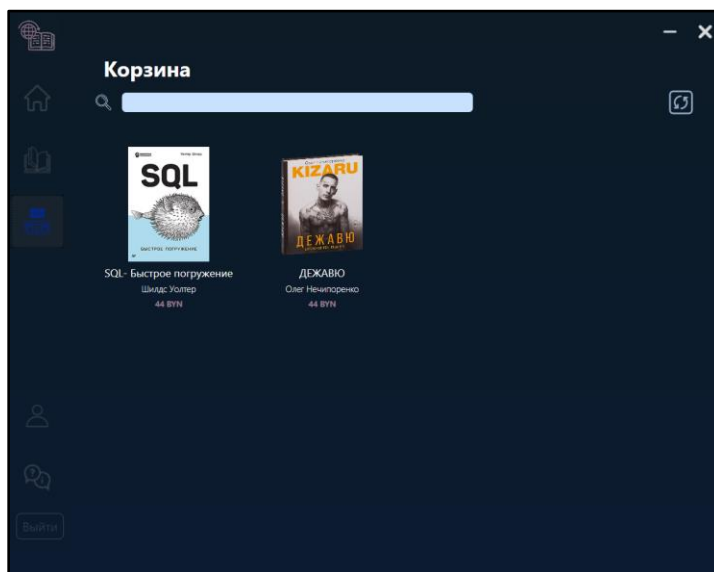


Рисунок 6.7 — Корзина

Корзина также имеет в себе поле для поиска определенных книг, а также кнопку обновления корзины. При выборе книги в корзине, открывается страница, представленная на рисунке 6.8.

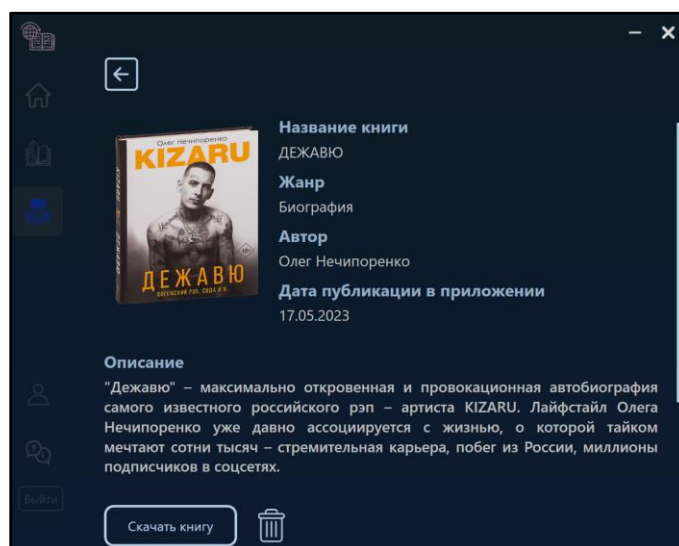


Рисунок 6.8 — Книга в корзине

На данной странице имеется 4 кнопки: вернуться на прошлую страницу, скачать книгу, удалить книгу из корзины и оставить отзыв. При нажатии на кнопку удалить книгу, она удалится из корзины, а при нажатии на кнопку скачать книгу, если книга платная, то откроется форма для произведения оплаты книги, иначе книга просто скачается по указанному пути.

Форма для отзыва представлена на рисунке 6.9.

A dark blue rectangular box containing a light blue horizontal input field on the left and a rounded rectangular button with the text "Оставить отзыв" (Leave review) on the right.

Рисунок 6.9 — Форма для отзыва

В случае, если книга платная, то при скачивании откроется форма для ее покупки, изображенная на рисунке 6.10.

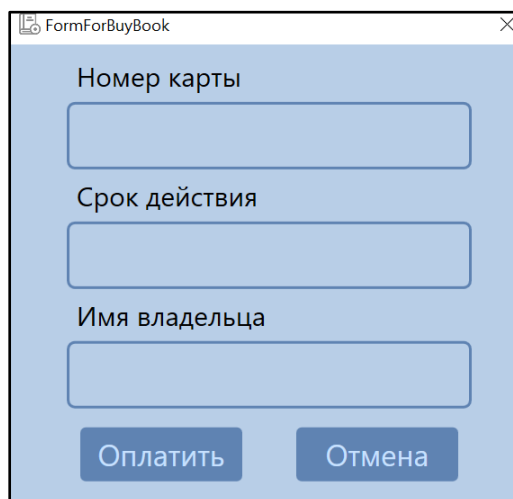
A light blue window titled "FormForBuyBook" with a close button. It contains three input fields labeled "Номер карты" (Card number), "Срок действия" (Expiration date), and "Имя владельца" (Owner's name). At the bottom are two buttons: "Оплатить" (Pay) and "Отмена" (Cancel).

Рисунок 6.10 — Форма для покупки книги

Страница профиля представлена на рисунке 6.11. На данной странице пользователь может изменить фото профиля и номер телефона.

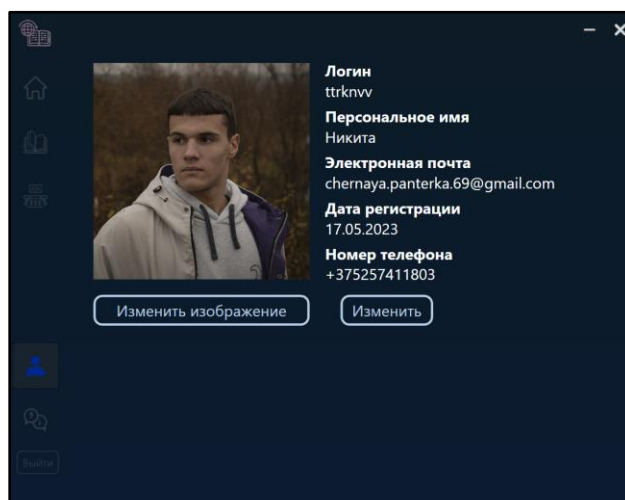
A dark blue window showing a user profile. On the left is a vertical sidebar with icons for home, books, and a profile. The main area features a profile picture of a man in a white jacket. To the right of the photo are the following details: "Логин" (Login) ttrknvv, "Персональное имя" (Personal name) Никита, "Электронная почта" (Email) chernaya.panterka.69@gmail.com, "Дата регистрации" (Registration date) 17.05.2023, and "Номер телефона" (Phone number) +375257411803. Below the photo are two buttons: "Изменить изображение" (Change image) and "Изменить" (Change).

Рисунок 6.11 — Страница профиля

Страница профиля отображает все данные, который были заполнены при регистрации пользователя.

Следующая кнопка переводит пользователя на страницу, предназначенной для отправки сообщения в техническую поддержку. Данная страница представлена на рисунке 6.12.

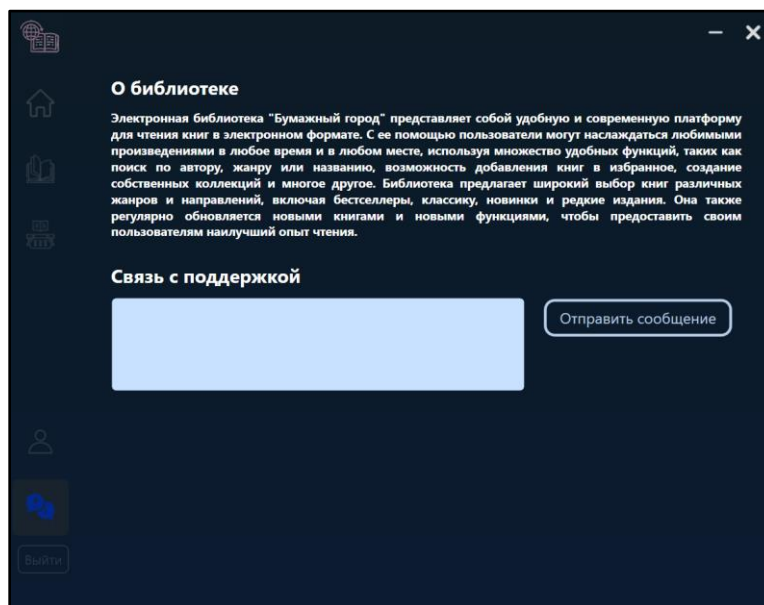


Рисунок 6.12 — Страница технической поддержки

Следующая кнопка позволяет пользователю выйти из приложения и вернуться к окну авторизации. В окне авторизации пользователь может сменить язык приложения, что изображено на рисунке 6.13.

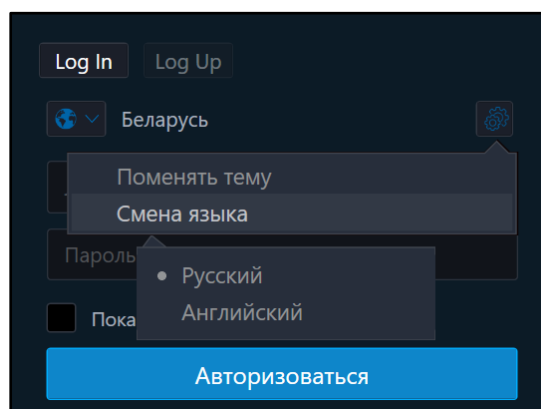


Рисунок 6.13 — Смена языка

Также пользователь может поменять тему, нажав на кнопку «Поменять тему». Важно помнить, для того, чтобы пользоваться данным приложением нужно иметь доступ в Интернет.



## ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта было успешно разработано программное средство для электронной библиотеки «Бумажный город». Это приложение предоставляет удобный и многофункциональный интерфейс для чтения и изучения книг, а также позволяет администраторам управлять контентом библиотеки.

В ходе разработки были рассмотрены и применены современные технологии, такие как WPF, Entity Framework Core и MS SQL, что позволило создать надежное и масштабируемое программное решение. Также применение архитектурного паттерна MVVM обеспечило разделение логики и пользовательского интерфейса, что повысило гибкость и переиспользуемость кода.

Программное средство предоставляет широкий набор функциональных возможностей для пользователей, включая регистрацию, авторизацию, поиск и фильтрацию книг, добавление книг в корзину, возможность их скачивания или покупки, получение информации о статусе заказа путем сообщения на почту, а также возможность оставить отзыв к заданной книге. Также приложение предоставляет дополнительную защиту для пользователей путем хеширования их паролей и проверки электронной почты путем отправления 6-тизначного кода.

В процессе разработки были учтены требования и пожелания пользователей, что способствовало созданию удобного и интуитивно понятного интерфейса. Также была предусмотрена возможность дальнейшего расширения и модификации программного средства для адаптации под изменяющиеся потребности пользователей.

Было проведено тестирование всех возможных функций. В ходе тестирования были исправлены некоторые моменты, что сделало программное средство для электронной библиотеки «Бумажный город» еще надежнее для пользователя.

Было разработано приложение для администратора библиотеки, которое включает в себя такие функции, как просмотр каталога книг, добавление книг, их изменение, просмотр списка пользователей, возможность отправления сообщения пользователям на почту, возможность подтверждения оплат пользователей и отвечать на сообщения в техническую поддержку.

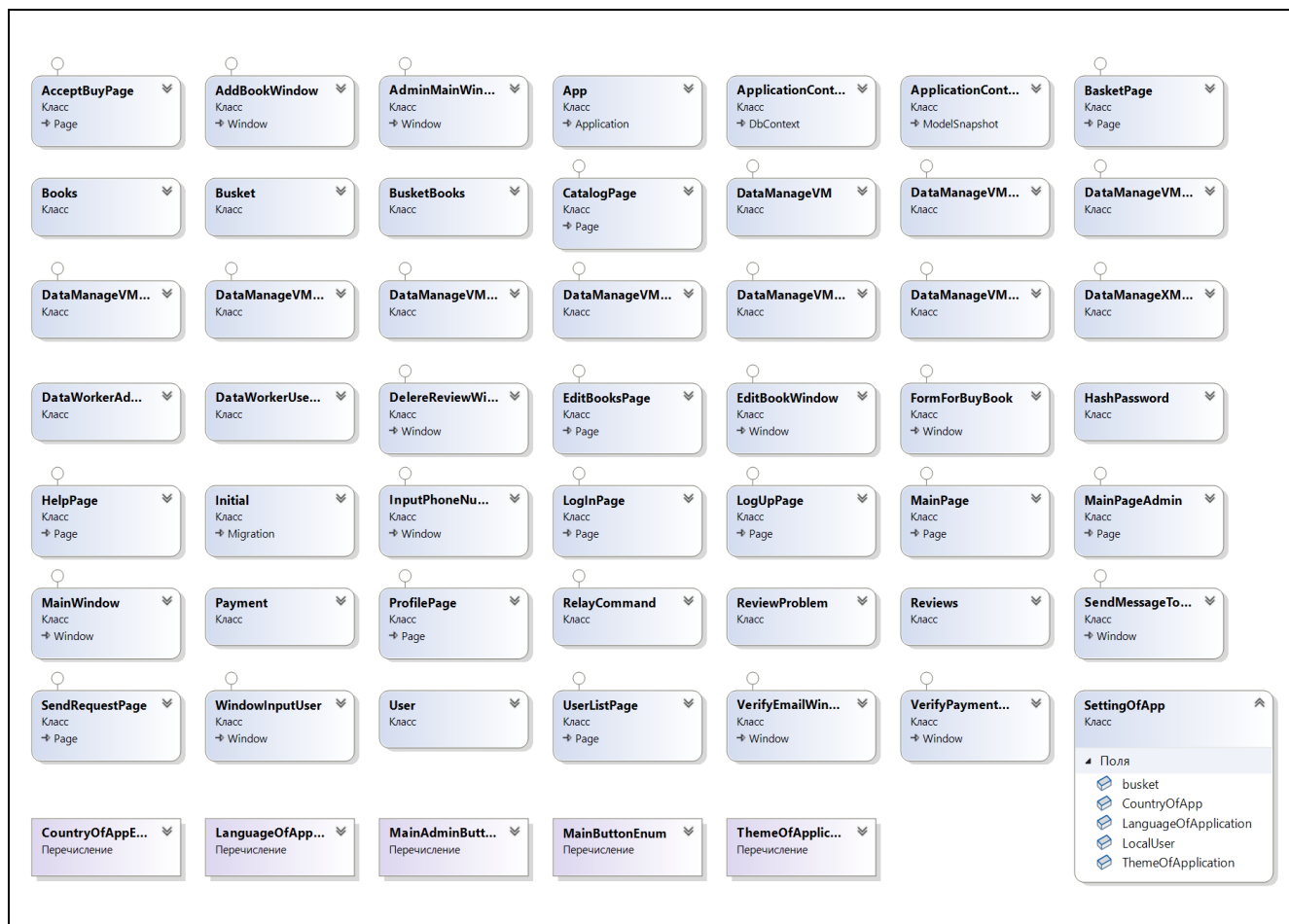
В дальнейшем рекомендуется продолжать развитие и поддержку программного средства, внедрять новые функции и улучшения на основе обратной связи от пользователей. Это поможет обеспечить бесперебойную работу системы и удовлетворение потребностей разных пользователей.

### Список использованных источников

1. Microsoft Visual Studio [Электронный ресурс]. – Режим доступа: [https://ru.wikipedia.org/wiki/Microsoft\\_Visual\\_Studio](https://ru.wikipedia.org/wiki/Microsoft_Visual_Studio) – Дата доступа 23.04.2023
2. Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7. Режим доступа: <https://metanit.com/sharp/tutorial/> – Дата доступа: 23.04.2023
3. Пацей, Н.В. Курс лекций по языку программирования C# / Н. В. Пацей. – Минск: БГТУ, 2018. – 175 с.
4. Руководство по WPF // [Электронный ресурс]. – Режим доступа: <https://metanit.com/sharp/wpf/> – Дата доступа: 25.04.2023
5. Руководство по XAML // [Электронный ресурс]. – Режим доступа: <https://www.tutorialspoint.com/xaml/index.htm>– Дата доступа: 25.04.2023
6. Блинова, Е.А. Курс лекций по Бадам данным / Е.А. Блинова. – Минск: БГТУ, 2019. – 175 с.
7. Работа с Entity Framework Core [Электронный ресурс]. – Режим доступа: <https://professorweb.ru/my/entity-framework/6/level1/> – Дата доступа 28.04.2023

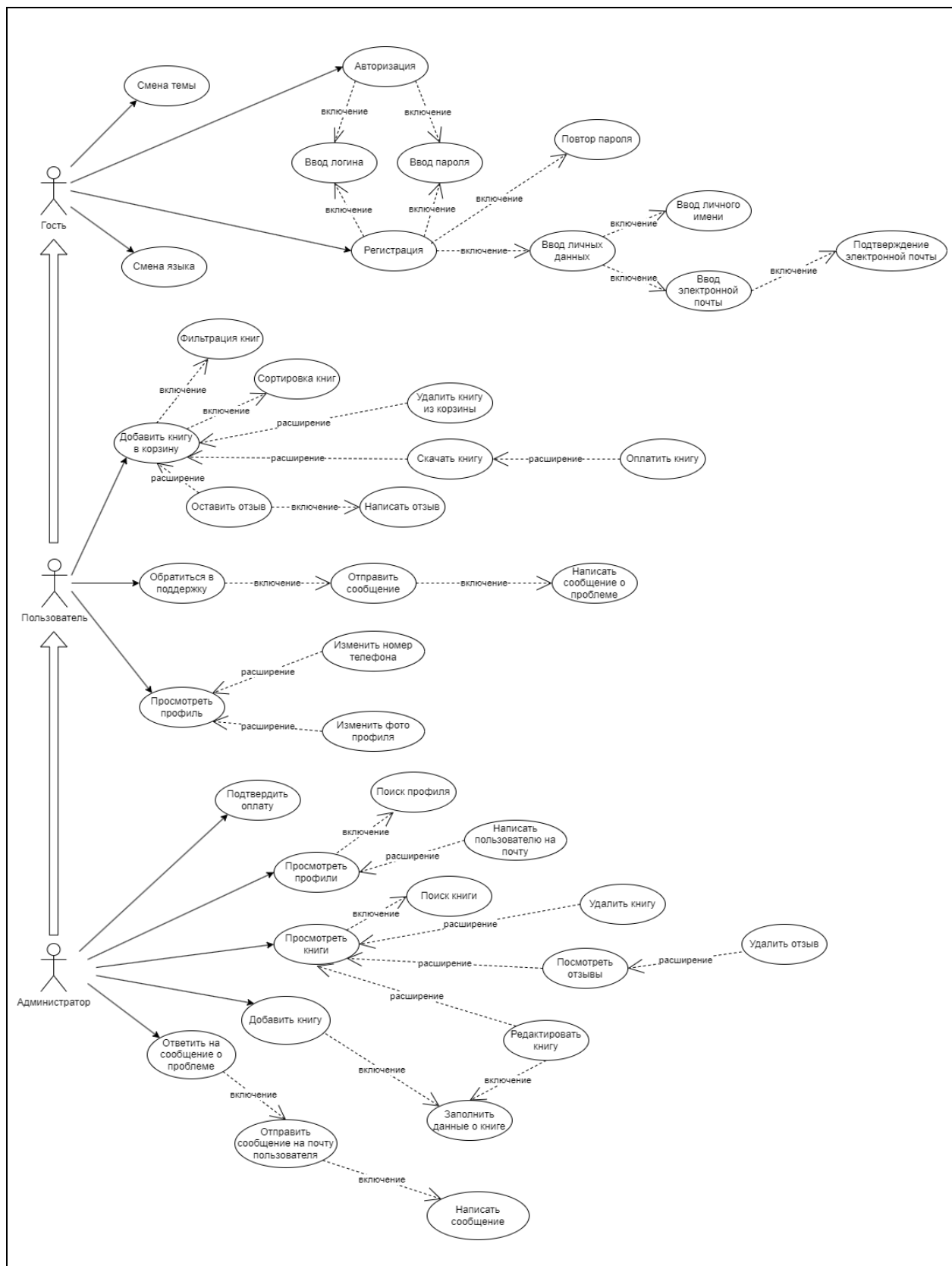
# ПРИЛОЖЕНИЕ А

## Диаграмма классов



# ПРИЛОЖЕНИЕ Б

## Диаграмма использования



## ПРИЛОЖЕНИЕ В

### Листинг 1 : создание базы данных

```

create database PAPERDB
on primary
(name = N'PAPERDB_MDF', filename =
N'D:\2k2s\COURSE_PROJECT\DB\MAINDB\PAPERDB.mdf',
size = 10240Kb, maxsize = UNLIMITED, filegrowth = 10240Kb),
(name = N'PAPERDB_NDF', filename =
N'D:\2k2s\COURSE_PROJECT\DB\MAINDB\PAPERDB.ndf',
size = 10240Kb, maxsize = UNLIMITED, filegrowth = 10240Kb),
filegroup forUser
(name = N'PAPERDB_USER', filename =
N'D:\2k2s\COURSE_PROJECT\DB\USERS\PAPERDB_USER_1.mdf',
size = 10240Kb, maxsize = UNLIMITED, filegrowth = 10240Kb),
(name = N'PAPERDB_USER_2', filename =
N'D:\2k2s\COURSE_PROJECT\DB\USERS\PAPERDB_USER_2.ndf',
size = 10240Kb, maxsize = UNLIMITED, filegrowth = 10240Kb),
filegroup forRole
(name = N'PAPERDB_ROLE', filename =
N'D:\2k2s\COURSE_PROJECT\DB\USERS\PAPERDB_ROLE_1.mdf',
size = 10240Kb, maxsize = UNLIMITED, filegrowth = 10240Kb),
(name = N'PAPERDB_ROLE_2', filename =
N'D:\2k2s\COURSE_PROJECT\DB\USERS\PAPERDB_ROLE_2.ndf',
size = 10240Kb, maxsize = UNLIMITED, filegrowth = 10240Kb),
filegroup forTypePrice
(name = N'PAPERDB_TYPEPRICE', filename =
N'D:\2k2s\COURSE_PROJECT\DB\USERS\PAPERDB_TYPEPRICE_1.mdf',
size = 10240Kb, maxsize = UNLIMITED, filegrowth = 10240Kb),
(name = N'PAPERDB_TYPEPRICE_2', filename =
N'D:\2k2s\COURSE_PROJECT\DB\USERS\PAPERDB_TYPEPRICE_2.ndf',
size = 10240Kb, maxsize = UNLIMITED, filegrowth = 10240Kb),
filegroup forBooks
(name = N'PAPERDB_BOOK', filename =
N'D:\2k2s\COURSE_PROJECT\DB\USERS\PAPERDB_BOOK_1.mdf',
size = 10240Kb, maxsize = UNLIMITED, filegrowth = 10240Kb),
(name = N'PAPERDB_BOOK_2', filename =
N'D:\2k2s\COURSE_PROJECT\DB\USERS\PAPERDB_BOOK_2.ndf',
size = 10240Kb, maxsize = UNLIMITED, filegrowth = 10240Kb),
filegroup forBusket
(name = N'PAPERDB_BUSKET', filename =
N'D:\2k2s\COURSE_PROJECT\DB\USERS\PAPERDB_BUSKET_1.mdf',
size = 10240Kb, maxsize = UNLIMITED, filegrowth = 10240Kb),
(name = N'PAPERDB_BUSKET_2', filename =
N'D:\2k2s\COURSE_PROJECT\DB\USERS\PAPERDB_BUSKET_2.ndf',
size = 10240Kb, maxsize = UNLIMITED, filegrowth = 10240Kb),
filegroup forBusketBooks
(name = N'PAPERDB_BUSKETBOOKS', filename =
N'D:\2k2s\COURSE_PROJECT\DB\USERS\PAPERDB_BUSKETBOOKS_1.mdf',
size = 10240Kb, maxsize = UNLIMITED, filegrowth = 10240Kb),

```

```
(name = N'PAPERDB_BUSKETBOOKS_2', filename =
N'D:\2k2s\COURSE_PROJECT\DB\USERS\PAPERDB_BUSKETBOOKS_2.ndf',
size = 10240Kb, maxsize = UNLIMITED, filegrowth = 10240Kb)
log on
(name = N'T_MyBase_log', filename =
N'D:\2k2s\COURSE_PROJECT\DB\LOG\T_MyBase_log.ldf',

size = 10240Kb, maxsize = 1Gb, filegrowth = 10%)
```

### Листинг 2: создание таблицы Role

```
create table ROLE
(
    ROLE_USER nvarchar(20) check(Role_user in ('admin', 'common'))
primary key
) on forRole;
```

### Листинг 3: создание таблицы USERS

```
create table USERS
(
    ID_USER int identity(1, 1) primary key,
    IMAGE_PROFILE nvarchar(100),
    LOGIN nvarchar(40) not null unique,
    NAME nvarchar(100) not null,
    PASSWORD nvarchar(50) not null,
    EMAIL nvarchar(70) not null,
    PHONE_NUMBER nvarchar(14),
    DATE_REGISTRATION date not null,
    ROLE nvarchar(20) foreign key references ROLE(ROLE_USER) not null
) on forUser;
```

### Листинг 4: создание таблицы TYPE\_PRICE

```
create table TYPE_PRICE
(
    TYPE_PRICE nvarchar(10) primary key check(TYPE_PRICE in('free',
'paying')) not null) on forTypePrice;
```

### Листинг 5: создание таблицы BOOKS

```
create table BOOKS
(
    ID_BOOK int identity(1, 1) primary key,
    IMAGE_BOOK nvarchar(100),
    NAME nvarchar(100) not null,
    AUTHOR nvarchar(100),
    POPULARITY int default(0),
    GENRE nvarchar(100),
    DESCRIPTION nvarchar(max),
    TYPE_PRICE nvarchar(10) foreign key references
TYPE_PRICE(TYPE_PRICE) not null,
    COST int,
    PATH_DOWNLOAD nvarchar(max),
    DATE_PUBLICATION date)
```

```
on forBooks;
```

#### Листинг 6: создание таблицы BUSKET

```
create table BUSKET
(
    ID_BUSKET int identity(1, 1) primary key,
    ID_USER int foreign key references USERS(ID_USER) not null) on
forBasket;
```

#### Листинг 7: создание таблицы BUSKET\_BOOKS

```
create table BUSKET_BOOKS
(
    ID_RECORD int primary key identity(1, 1),
    ID_BUSKET int foreign key references BUSKET(ID_BUSKET) not null,
    ID_BOOK int foreign key references BOOKS(ID_BOOK) not null) on
forBasketBooks;
```

#### Листинг 8: создание таблицы PAYMENT

```
create table PAYMENT
(
    ID_PAY int identity(1, 1),
    ID_USER int foreign key references USERS(ID_USER),
    NUMBER_CARD nvarchar(max),
    NAME_OWNER nvarchar(max),
    DATE nvarchar(max),
    ID_BOOK int foreign key references BOOKS(ID_BOOK),
    STATE nvarchar(max) check (STATE in ('waiting', 'confirmed',
'rejected')))
```

#### Листинг 9: создание таблицы REVIEWS

```
create table REVIEWS
(
    ID_REVIEW int identity(1, 1) primary key,
    LOGIN nvarchar(40) foreign key references USERS(LOGIN),
    ID_BOOK int foreign key references BOOKS(ID_BOOK),
    REVIEW_TEXT nvarchar(max),
    DATE datetime)
```

#### Листинг 10: создание таблицы REVIEW\_PROBLEM

```
create table REVIEW_PROBLEM
(
    ID_REVIEW int identity(1, 1) primary key,
    LOGIN nvarchar(40) foreign key references USERS(LOGIN),
    DATE date,
    REVIEW_TEXT nvarchar(max),
    STATE nvarchar(max) check (STATE in ('waiting', 'confirmed',
'rejected'))
)
```

# Листинг 11: форма авторизации

```

        <TextBox Name="LoginText" Grid.Row="0"
Style="{DynamicResource GetDataUser}"
        Margin="25, 55, 25, 0" VerticalAlignment="Top"/>
        <TextBlock x:Name="PlugOfLogin" IsHitTestVisible="False"
FontSize="14" Text="{DynamicResource PlugLogin}"
        Grid.Row="0" Margin="37, 63, 0, 0"
VerticalAlignment="Top"
        Style="{DynamicResource PlugText}"/>
        <PasswordBox x:Name="PasswordTextStars" Grid.Row="0"
Style="{DynamicResource ForPasswordSec}"
        Margin="25, 100, 25, 0" VerticalAlignment="Top"
PasswordChanged="PasswordTextStars_PasswordChanged"/>
        <TextBox x:Name="PasswordText" Grid.Row="0"
Style="{DynamicResource GetDataUser}"
        Margin="25, 100, 25, 0" VerticalAlignment="Top"
Visibility="Hidden" TextChanged="PasswordText_TextChanged"/>
        <TextBlock x:Name="PlugPassword" IsHitTestVisible="False"
FontSize="14" Text="{DynamicResource PlugPassword}"
        Grid.Row="0" Margin="37, 108, 0, 0"
VerticalAlignment="Top"
        Style="{DynamicResource PlugTextt}"/>
        <ToggleButton Name="ShowPassword" Grid.Row="0"
Height="30" Width="140"
        HorizontalAlignment="Left"
VerticalAlignment="Top"
        Margin="25, 145, 0, 0"
Content="{DynamicResource ShowPassword}"
        FontSize="13" Style="{DynamicResource
ShowPassColor}" Click="ShowPassword_Click"
MouseEnter="ShowPassword_MouseEnter">
        <ToggleButton.Template>
        <ControlTemplate TargetType="{x:Type ToggleButton}">
        <StackPanel Orientation="Horizontal">
        <Border Name="BorderButtonShowPass"
Height="20" Width="20"
        HorizontalAlignment="Left" Style="{DynamicResource
ShowPasswordBorder}"
        CornerRadius="2" BorderThickness="1">
        <Image x:Name="MarkCheck" Margin="4"
Style="{DynamicResource ShowPassMark}"/>
        </Border>
        <ContentPresenter Content="{TemplateBinding
Content}" Margin="10, 0, 0, 0" VerticalAlignment="Center"
HorizontalAlignment="Center"/>
        </StackPanel>
        </ControlTemplate>
        </ToggleButton.Template>
    </ToggleButton>

```



```

        <Grid Grid.Row="0" Background="Transparent"
Name="ErrorBlockBorder" Height="40" Width="230"
            VerticalAlignment="Center"
HorizontalAlignment="Center"
            Margin="100, 18, 0, 0" Opacity="1"
Visibility="Hidden">
            <Polygon Name="ErrorBlock" Style="{DynamicResource
NoCorrect}"
                VerticalAlignment="Bottom"
HorizontalAlignment="Center"
                Points="0, 20,
                230, 20,
                230, 50,
                110, 50,
                120, 56,
                130, 50,
                0, 50">
            </Polygon>
            <TextBlock FontSize="14" HorizontalAlignment="Center"
                VerticalAlignment="Center" Margin="0, 0, 0,
4"
                Text="{DynamicResource NotCorrect}"
Style="{DynamicResource StyleForButtonFrame}"/>
        </Grid>

        <Border Name="GetAuth"
            Grid.Row="0"
            Height="35"
            VerticalAlignment="Bottom"
            Margin="25, 0, 25, 85"
            Style="{DynamicResource AuthificBorder}"
            BorderThickness="1" CornerRadius="2">
            <Button x:Name="Auther" Content="{DynamicResource
LogIn}" FontSize="16"
                Style="{DynamicResource ButtonForAutherization}"
                MouseEnter="Auther_MouseEnter"
                Command="{Binding LogInUser}"
                CommandParameter="{Binding
ElementName=PageLogIn}"/>

            </Border>

```

## Листинг 12: форма регистрации

```

<TextBlock x:Name="registerNew" Grid.Row="0" Text="{DynamicResource
CreateNewAccount}"
    Style="{DynamicResource WordReg}"
    Margin="25, 15, 0, 0" HorizontalAlignment="Left"
    Width="300" VerticalAlignment="Top"
    Height="26"/>
    <TextBox Name="LoginText" Grid.Row="0"
    Style="{DynamicResource GetDataUser}"

```

```

        Margin="25, 50, 25, 0" VerticalAlignment="Top"
        TextChanged="LoginText_TextChanged"
MaxLength="20"/>
        <TextBlock x:Name="PlugOfLogin" IsHitTestVisible="False"
FontSize="14" Text="{DynamicResource PlugLogin}"
            Grid.Row="0" Margin="37, 57, 0, 0"
VerticalAlignment="Top"
            Style="{DynamicResource PlugText}"/>
        <TextBlock x:Name="ErrorLogin" FontSize="10"
            Grid.Row="0" Margin="28,88, 0, 0"
VerticalAlignment="Top"
            Style="{DynamicResource PlugTextError}"
Visibility="Hidden"/>
        <TextBox Name="EmailText" Grid.Row="0"
Style="{DynamicResource GetDataUser}"
            Margin="25, 105, 25, 0" VerticalAlignment="Top"
            TextChanged="EmailText_TextChanged"
MaxLength="256"/>
        <TextBlock x:Name="PlugOfEmail" IsHitTestVisible="False"
FontSize="14" Text="{DynamicResource PlugEmail}"
            Grid.Row="0" Margin="37, 113, 0, 0"
VerticalAlignment="Top"
            Style="{DynamicResource PlugTextt}"/>
        <TextBlock x:Name="ErrorEmail" FontSize="10"
            Grid.Row="0" Margin="28,144, 0, 0"
VerticalAlignment="Top"
            Style="{DynamicResource PlugTextError}"
Visibility="Hidden"/>
        <TextBox Name="PNText" Grid.Row="0" Style="{DynamicResource
GetDataUser}"
            Margin="25, 160, 25, 0" VerticalAlignment="Top"
            TextChanged="PNText_TextChanged" MaxLength="50"/>
        <TextBlock x:Name="PlugOfPN" IsHitTestVisible="False"
FontSize="14" Text="{DynamicResource PlugPersName}"
            Grid.Row="0" Margin="37, 168, 0, 0"
VerticalAlignment="Top"
            Style="{DynamicResource PlugTextt}"/>
        <TextBlock x:Name="ErrorPN" FontSize="10"
            Grid.Row="0" Margin="28,198, 0, 0"
VerticalAlignment="Top"
            Style="{DynamicResource PlugTextError}"
Visibility="Hidden"/>
        <PasswordBox x:Name="PasswordTextStars" Grid.Row="0"
Style="{DynamicResource ForPasswordSec}"
            Margin="25, 215, 25, 0" VerticalAlignment="Top"
            PasswordChanged="PasswordTextStars_PasswordChanged"/>
        <TextBlock x:Name="PlugOfPassword" IsHitTestVisible="False"
FontSize="14" Text="{DynamicResource PlugPassword}"
            Grid.Row="0" Margin="37, 223, 0, 0"
VerticalAlignment="Top"
            Style="{DynamicResource PlugTextt}"/>
        <TextBlock x:Name="ErrorPassword" FontSize="10"

```

```

Grid.Row="0" Margin="28,252, 0, 0"
VerticalAlignment="Top"
Style="{DynamicResource PlugTextError}"
Visibility="Hidden"/>
<PasswordBox x:Name="PasswordTextStarsConf" Grid.Row="0"
Style="{DynamicResource ForPasswordSec}"
Margin="25, 270, 25, 0" VerticalAlignment="Top"
PasswordChanged="PasswordTextStarsConf_PasswordChanged"/>
<TextBlock x:Name="PlugOfPasswordConfirm"
IsHitTestVisible="False" FontSize="14" Text="{DynamicResource
PlugConfirmPass}"
Grid.Row="0" Margin="37, 278, 0, 0"
VerticalAlignment="Top"
Style="{DynamicResource PlugTexttt}"/>
<TextBlock x:Name="ErrorPasswordConfirm" FontSize="10"
Grid.Row="0" Margin="28,308, 0, 0"
VerticalAlignment="Top"
Style="{DynamicResource PlugTextError}"
Visibility="Hidden"/>
<Border Name="GetAuth" Grid.Row="0" Height="35"
VerticalAlignment="Bottom"
Margin="25, 0, 25, 25" Style="{DynamicResource
AuthificBorder}"
BorderThickness="1" CornerRadius="2">
<Button x:Name="Auther" Content="{DynamicResource
ButtRegister}" FontSize="16"
Style="{DynamicResource ButtonForAutherization}"
Command="{Binding RegisterNewUser}"
CommandParameter="{Binding
ElementName=LogUpPageElem}"/>

</Border>

```

### Листинг 13: класс HashPassword

```

public class HashPassword
{
    public static string GetHashPassword(string password)
    {
        try
        {
            MD5 md5 = MD5.Create();

            byte[] b = Encoding.ASCII.GetBytes(password);
            byte[] hash = md5.ComputeHash(b);

            StringBuilder sb = new StringBuilder();
            foreach (var a in hash)
            {
                sb.Append(a.ToString("X2"));
            }
        }
    }
}

```

```

        return Convert.ToString(sb);
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "ErrorHashPassword",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        return "";
    }
}

}

```

#### Листинг 14: класс ApplicationContext

```

public class ApplicationContext : DbContext
{
    public DbSet<User> USERS { get; set; }
    public DbSet<Busket> BUSKET { get; set; }
    public DbSet<BusketBooks> BUSKET_BOOKS { get; set; }
    public DbSet<Books> BOOKS { get; set; }
    public DbSet<Payment> PAYMENT { get; set; }
    public DbSet<Reviews> REVIEWS { get; set; }
    public DbSet<ReviewProblem> REVIEW_PROBLEM { get; set; }
    public ApplicationContext()
    {
    }
    public
ApplicationContext(DbContextOptions<ApplicationContext> options) :
base(options)
    {
    }
    protected override void
OnConfiguring(DbContextOptionsBuilder optionsBuilder)
    {
optionsBuilder.UseSqlServer("Server=TARAKANOVNS;Database=PAPERDB;Tru
sted_Connection=True;TrustServerCertificate=True;");
    }
    protected override void OnModelCreating(ModelBuilder
modelBuilder)
    {
        modelBuilder.Entity<Books>().HasKey(b => b.ID_BOOK);
        modelBuilder.Entity<User>().HasKey(b => b.ID_USER);
        modelBuilder.Entity<Busket>().HasKey(b => b.ID_BUSKET);
        modelBuilder.Entity<BusketBooks>().HasKey(b =>
b.ID_RECORD);
        modelBuilder.Entity<Payment>().HasKey(b => b.ID_PAY);
    }
}

```

**Листинг 15: метод, отсылающий сообщение на почту**

```

public async Task PutMessageEmail()
{
    await Task.Run(() =>
    {
        SmtplibClient smtpClient = new
SmtplibClient("smtp.gmail.com", 587);
        smtpClient.EnableSsl = true;
        smtpClient.UseDefaultCredentials = false;
        smtpClient.Credentials = new
NetworkCredential("paperrcityy@gmail.com", "rkyymklkydinddnsz");

        MailMessage mailMessage = new MailMessage();
        mailMessage.From = new
MailAddress("paperrcityy@gmail.com");
        mailMessage.To.Add(SettingOfApp.LocalUser.EMAIL);
        mailMessage.Subject = $"Покупка книги
\"{bookNow.First().NAME}\"";
        var htmlBody = $"<html><body><p style='font-size:
18pt;'>Покупка книги {bookNow.First().NAME}</p>" +
            $"<br><img src=\"cid:bookImage\" style='width:
300px; height: 500px;'>" +
            $"<p style='font-size: 18pt;'>Здравствуйтесь,
{SettingOfApp.LocalUser.NAME}! Благодарим вас за покупку нашей
книги! Ждите подтверждения администратора в течение 24
часов!</p></body></html>";
        AlternateView alternateView =
AlternateView.CreateAlternateViewFromString(htmlBody, null,
MediaTypeNames.Text.Html);
        LinkedResource bookImage = new
LinkedResource(bookNow.First().IMAGE_BOOK,
MediaTypeNames.Image.Jpeg);
        bookImage.ContentId = "bookImage";
        alternateView.LinkedResources.Add(bookImage);
        mailMessage.AlternateViews.Add(alternateView);
        //mailMessage.Body = "This is a test email.";
        smtpClient.Send(mailMessage);
    });
}

```