|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_ ***ИУ-КФ «Информатика и управление»\_\_\_\_\_\_\_\_\_***

**КАФЕДРА** \_\_ ***ИУ5-КФ «Системы обработки информации»***

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовой работе на тему:**

***Приложение для обмена или продажи книг «BookSharing»***

по дисциплине **Архитектура автоматизированных систем обработки информации и управления**

Студент гр.\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_Продувнов А.Г.\_\_)

(подпись) (Ф.И.О.)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_Смирнов М.Е.\_\_\_\_\_)

(подпись) (Ф.И.О.)

Оценка руководителя \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка защиты \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_

30-50 (дата)

Оценка работы \_\_\_\_\_ баллов \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(оценка по пятибалльной шкале)

Комиссия: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

(подпись) (Ф.И.О.)

Калуга, 2020

Калужский филиал   
федерального государственного бюджетного образовательного учреждения высшего образования

***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»   
(КФ МГТУ им. Н.Э. Баумана)***

УТВЕРЖДАЮ

Заведующий кафедрой **\_\_ИУК5-КФ \_\_**

\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине **Архитектура автоматизированных систем обработки информации и управления**

Студент Продувнов А.Г., ИУК5-51Б\_\_

(фамилия, инициалы, индекс группы)

Руководитель \_Смирнов М.Е.\_\_\_\_\_

(фамилия, инициалы)

График выполнения работы: 25% к\_4\_нед., 50% к\_7\_нед., 75% к\_10\_нед., 100% к\_14\_нед.

***1. Тема курсовой работы***

***Приложение для обмена или продажи книг «BookSharing»***

***2. Техническое задание***

*Разработать приложение для обмена или продажи книг «BookSharing»*

***3. Оформление курсовой работы***

3.1. Расчетно-пояснительная записка на\_\_\_\_\_\_\_\_ листах формата А4.

3.2. Перечень графического материала КП (плакаты, схемы, чертежи и т.п.)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

Руководитель курсовой работы\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/

(подпись) (Ф.И.О.)

Задание получил\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_20\_\_г.

(подпись) (Ф.И.О.)

Примечание:

Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

Оглавление

[1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ 5](#_Toc58961045)

[1.1. Наименование. 5](#_Toc58961046)

[1.2. Основание для разработки. 5](#_Toc58961047)

[1.3. Исполнитель. 5](#_Toc58961048)

[1.4. Цель разработки. 5](#_Toc58961049)

[1.5. Содержание работы: 5](#_Toc58961050)

[1.5.1. Задачи, подлежащие решению. 5](#_Toc58961051)

[1.5.2. Требования к архитектуре АСОИ. 6](#_Toc58961052)

[1.5.3. Требования к составу программных компонентов. 6](#_Toc58961053)

[1.5.4. Требования к прикладным программам. 6](#_Toc58961054)

[1.5.5. Требования к входным/выходным данным. 6](#_Toc58961055)

[1.5.6. Требования к временным характеристикам. 7](#_Toc58961056)

[1.5.7. Требования к составу технических средств. 7](#_Toc58961057)

[1.6. Этапы разработки. 7](#_Toc58961058)

[1.7. Техническая документация, предъявляемая по окончании работы. 7](#_Toc58961059)

[1.8. Дополнительные условия. 8](#_Toc58961060)

[2.НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ 9](#_Toc58961061)

[2.1. Постановка задачи 9](#_Toc58961062)

[2.2. Перечень задач, подлежащих решению в процессе разработки 9](#_Toc58961063)

[2.3 Анализ существующих аналогов 10](#_Toc58961064)

[2.4. Описание функционала 12](#_Toc58961065)

[2.5. Обоснование выбора языка и среды разработки 13](#_Toc58961066)

[2.5.1. Язык разработки 13](#_Toc58961067)

[2.5.2. Среда разработки 14](#_Toc58961068)

[2.6. Перечень задач, подлежащих решению в процессе разработки 15](#_Toc58961069)

[2.7. Вывод 15](#_Toc58961070)

[3.ПРОЕКТНО-КОНСТРУКТОРСКАЯ ЧАСТЬ 16](#_Toc58961071)

[3.1. Общие сведения 16](#_Toc58961072)

[3.2. Структура приложения 16](#_Toc58961073)

[3.3. Архитектура приложения 16](#_Toc58961074)

[3.4. Разработка интерфейса 22](#_Toc58961075)

[3.4.1. Планирование 22](#_Toc58961076)

[3.4.2. Реализация интерфейса 26](#_Toc58961077)

[3.5 Вывод 27](#_Toc58961078)

[4.ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 28](#_Toc58961079)

[4.1 Требования к аппаратной платформе. 28](#_Toc58961080)

[4.2. Пользовательские сценарии 28](#_Toc58961081)

[4.3. Тестирование приложения 30](#_Toc58961082)

[3.5 Вывод 37](#_Toc58961083)

[ЗАКЛЮЧЕНИЕ 38](#_Toc58961084)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 39](#_Toc58961085)

[ПРИЛОЖЕНИЕ А 41](#_Toc58961086)

# 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ

## 1.1. Наименование.

Приложение для обмена или продажи книг «BookSharing».

## 1.2. Основание для разработки.

Основанием для разработки сервиса является задание по курсовой работе в соответствии с учебным планом дисциплины «Базы данных».

## 1.3. Исполнитель.

Студент группы СОИ.Б-51 Продувнов А.Г.

## 1.4. Цель разработки.

Целью работы является предоставление людям возможности обмена и продажи книг.

Задачи проектирования:

1. Создание приложения, позволяющего пользователям выставлять книги для обмена или продажи книг.

2. Реализация функционала для частичной автоматизации обмена книгами.

## 1.5. Содержание работы:

### 1.5.1. Задачи, подлежащие решению.

1. разработать модель данных;

2. разработать интерфейс;

3. реализовать поиск книг по критериям;

4. произвести тестирования и выявить недостатки для дальнейшего исправления.

### 1.5.2. Требования к архитектуре АСОИ.

К архитектуре предъявляются следующие требования:

1. Модульная архитектура программного обеспечения, в которой каждый модуль представляет собой один из видов функционала приложения.

2. Каждая функция приложения должна быть представлен в виде отдельного класса.

3. Классы должны быть реализованы в виде иерархии классов.

### 1.5.3. Требования к составу программных компонентов.

Программный комплекс должен состоять из следующих программных компонентов:

1. программная оболочка для работы с данными.

2. база данных.

### 1.5.4. Требования к прикладным программам.

Для работы программного комплекса необходимы: Microsoft Windows 8/10 и минимальный набор драйверов, обеспечивающих работоспособность ПК.

### 1.5.5. Требования к входным/выходным данным.

Входные данные:

Пользовательские данные (логин, пароль, email)

Информация о книгах

Данные для поиска книг

Выходные данные:

Результат поиска книг

### 1.5.6. Требования к временным характеристикам.

Требования к временным характеристикам программы не предъявляются.

### 1.5.7. Требования к составу технических средств.

Для функционирования системы необходимы:

* процессор: 1 ГГц и быстрее с поддержкой PAE, NX и SSE2;
* RAM: 1 Гбайт (32 бит) или 2 Гбайт (64 бит);
* HDD: 16 Гбайт (32 бит) или 20 Гбайт (64 бит);
* видеокарта: поддержка Microsoft DirectX 9 с драйвером WDDM;
* монитор;
* клавиатура;
* мышь.

## 1.6. Этапы разработки.

1. Поиск и исследование аналогов.
2. Разработка модели данных для приложения
3. Создание базы данных
4. Разработка приложения для взаимодействия с базой данных

## 1.7. Техническая документация, предъявляемая по окончании работы.

По окончанию работы предъявлена расчетно-пояснительная записка в состав которой входят:

* техническое задание;
* научно-исследовательская часть;
* проектно-конструкторская часть;
* проектно-технологическая часть.

Также должна быть предоставлена графическая часть работы, выполненная формате А1 на 2 листах, в которую входят:

* демонстрационные чертежи;
* алгоритмические схемы.

## 1.8. Дополнительные условия.

1. Язык программирования С#

2. Использование базы данных для хранения информации о пользователях и книгах.

3. Среда разработки Visual Studio 2019

# 2.НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

## 2.1. Постановка задачи

Множество людей, читающих бумажные книги, читают их лишь однажды, а затем они занимают место и пылятся на полках. В настоящее время покупка книг является не дешевым удовольствием, а обмен ими задаче не из простых так как сейчас не многие люди читают именно бумажные книги. С решением этим проблем может помочь приложение позволяющее обмениваться книгами с незнакомыми людьми или же продавать их по более выгодной для людей цене нежели покупать новые.

Главной задачей курсовой работы является проектирование и создание приложения с использованием базы данных для реализации возможностей обмена и продажи книг пользователями.

## 2.2. Перечень задач, подлежащих решению в процессе разработки

В процессе выполнения курсовой работы и разработки приложения требуется решить следующие задачи:

исследовать существующие аналоги приложений для обмена книгами для поиска проблем, которым можно будет избежать в проектирование приложения;

разработать базу данных для хранения информации пользователях и данных о книгах;

спроектировать и разработать приложение;

* провести тестирование приложения.

## 2.3 Анализ существующих аналогов

Приложений для обмена именно книгами не так уж и много поэтому стоит рассматривать не только их, но и случаи, когда происходит обмен любыми, вещали.

1) BEE - Book Exchange Easy

Book Exchange Easy — это онлайн-платформа приложений для обмена книгами.

Это приложение позволяет вам делиться книгами в Интернете с читателями, которых вы не обязательно знаете.

Мобильное приложение, обладающее функциями чата и оценки как книг, так и человека, с которым был произведен обмен. Данное приложение не столько платформа для обмена книгами, сколько социальная сеть для знакомства людей с одинаковым предпочтением в книгах для дальнейшего обмена. Главным недостатком этого приложения является региональная ограниченность, оно доступно лишь на территории германии.

2) BasNaBash

«Баш на Баш» — сервис для обмена машинами, личными вещами и бытовой техникой.

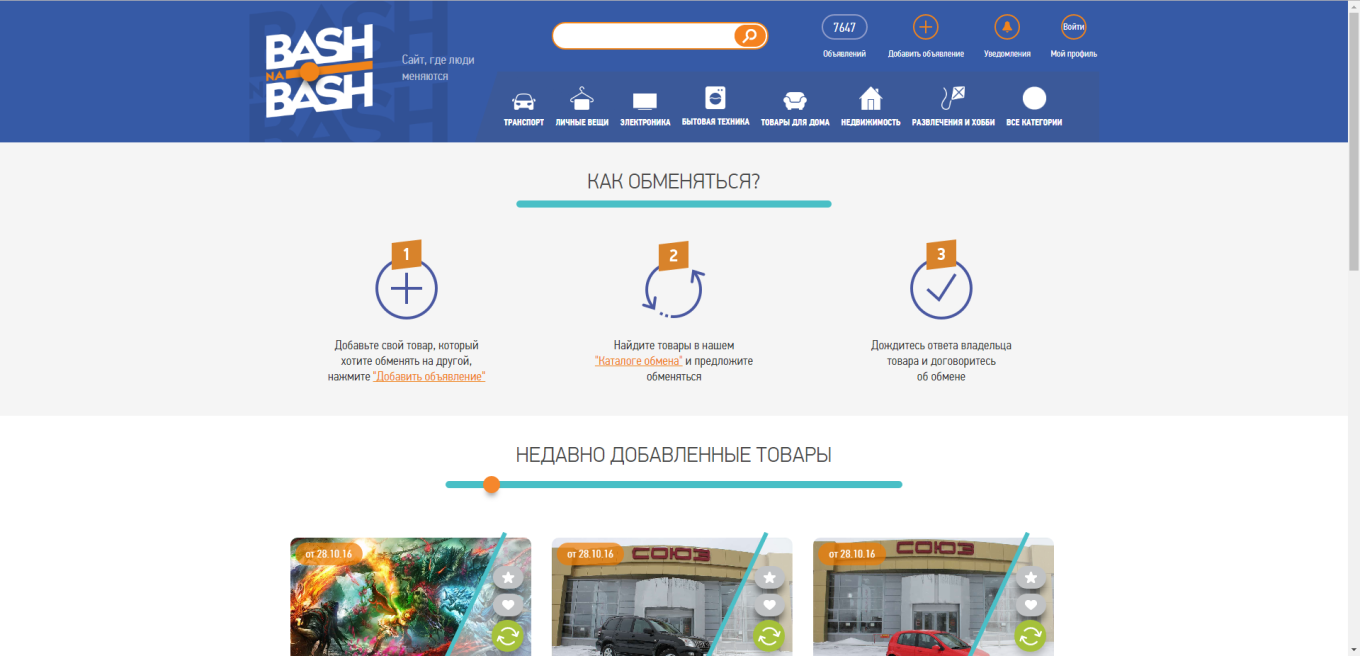


Рисунок 1 – Главная страница «Баш на Баш»

Хоть создатели сервиса и говорят, что он создан для обмена вещами, если деньги считать вещами, то это действительно так, пользователи выставляют вещи и назначают им цену именно обмена не происходит. Интересным решение является вывешивание недавно добавленных вещей на главную страницу, что позволяет заинтересовать пользователей, которые скорее всего и не думали, что подобная вещь им понадобится.

3) Swaper – сервис, позволяющий пользователям размещать вещи, которые они хотели бы обменять и указывать интересные для себя вещи, обмен на которые они готовы произвести.

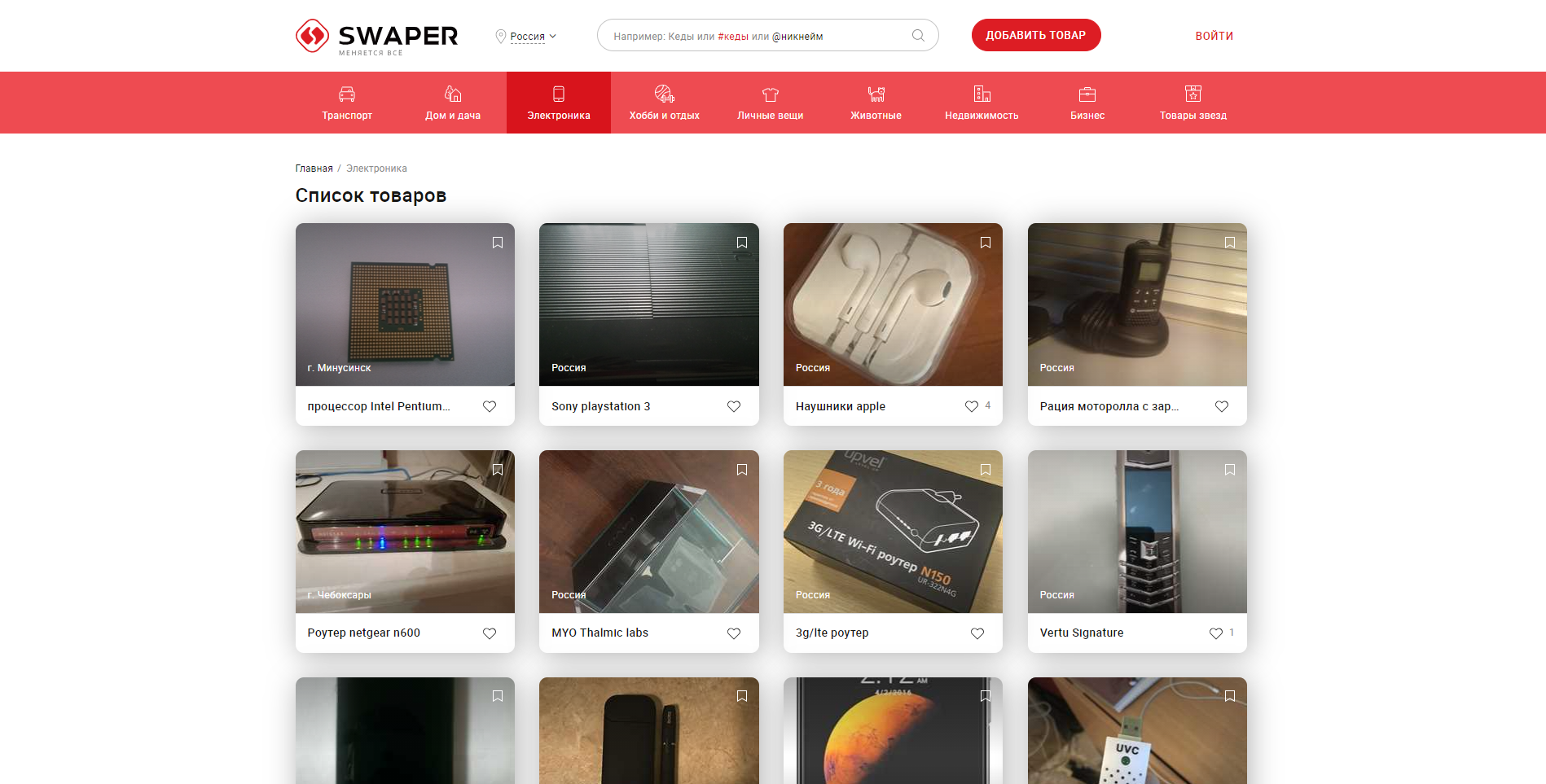
****

Рисунок 2 – Сайт «Swaper»

Сервис позволяет размещать товары, прикладывать фото и описание. Так же существует внутренняя валюта сервиса, упрощающая обмен, но непонятно почему бы не использовать просто деньги для этих целей.

## 2.4. Описание функционала

На основании рассмотренных аналогов можно предположить действие пользователей в системе и реализовать функционал для повышения удобства пользователя добавляя элементы и функции для ускорения или упрощения этих действий.

Для облегчения поиска нужно пользователю книги нужно добавить поиск по названиям, жанрам, авторам.

Реализация ветрины недавно добавленных книг поможет пользователям заинтересоваться книгами, которые до этого им возможно даже не были известны.

Так же для облегчения поиска необходимо реализовать функцию поиска по нажатию на название книги, её жанр или автора.

При выставлении книги на обмен пользователь сможет ввести ISBN книги и данные о ней заполнятся автоматически, если такой книги в базе нет пользователь сможет её добавить.

Администраторы смогут просматривать новые книги, добавленные в базу и исправлять их, с редактирование всех пользовательских книг или их удалением.

## 2.5. Обоснование выбора языка и среды разработки

### 2.5.1. Язык разработки

C# — компилируемый, статически типизированный язык программирования общего назначения.

Поддерживает такие парадигмы программирования, как процедурное программирование, объектно-ориентированное программирование, обобщённое программирование. Язык имеет богатую стандартную библиотеку, которая включает в себя распространённые контейнеры и алгоритмы, ввод-вывод, регулярные выражения, поддержку многопоточности и другие возможности. C# сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

C# широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также игр. Существует множество реализаций языка C#, как бесплатных, так и коммерческих и для различных платформ.

### 2.5.2. Среда разработки

Microsoft Visual Studio - полнофункциональная интегрированная среда разработки (IDE) с поддержкой популярных языков программирования, среди которых С, C++, VB.NET, C#, F#, JavaScript, Python.

Функциональность Visual Studio охватывает все этапы разработки программного обеспечения, предоставляя современные инструменты для написания кода, проектирования графических интерфейсов, сборки, отладки и тестирования приложений. Возможности Visual Studio могут быть дополнены путем подключения необходимых расширений.

Редактор кода Visual Studio поддерживает подсветку синтаксиса, вставку фрагментов кода, отображение структуры и связанных функций. Существенно ускорить работу помогает технология IntelliSense - автозавершение кода по мере ввода.

Visual Studio может быть использована для создания и обновления файла базы данных в SQL Server Express, что позволяет упростить работу с базой данных для создания приложения.

Встроенный отладчик Visual Studio используется для поиска и исправления ошибок в исходном коде, в том числе на низком аппаратном уровне. Инструменты диагностики позволяют оценить качество кода с точки зрения производительности и использования памяти.

Дизайнер форм Visual Studio незаменим при разработке программ с графическим интерфейсом, помогая спроектировать внешний вид будущего приложения и работу каждого элемента интерфейса.

Visual Studio предоставляет комплекс инструментов для автоматизации тестирования приложений в части проверки работы интерфейсов, модульного и нагрузочного тестирования.

## 2.6. Перечень задач, подлежащих решению в процессе разработки

Для выполнения поставленной задачи необходимо определить оптимальную архитектуру для построения системы и средства реализации, соответствующего данной архитектуре. В задачи разрабатываемой системы входят разработка базы данных для хранения информации о пользователях и книгах, разработка приложения и его последующее тестирование.

## 2.7. Вывод

На основании анализа аналогов, был выявлен функционал, предстоящий к решению. Было принято решение использовать следующие инструменты: в качестве языка программирования был выбран С#. Для реализации сервиса была выбрана интегрированная среда разработки Microsoft Visual Studio.

# 3.ПРОЕКТНО-КОНСТРУКТОРСКАЯ ЧАСТЬ

## 3.1. Общие сведения

Данная курсовая работа представляет собой приложение «BookSharing» для обмена и продажи книг.

## 3.2. Структура приложения

Приложение «BookSharing», представляющее собой оконное приложение с возможностью выставить собственные книги для обмена или продажи, а также найти книги для других пользователей использую поиск или просматривая последние добавленные в приложения книги.

## 3.3. Архитектура приложения

При разработке приложения была реализована иерархическая архитектура классов.

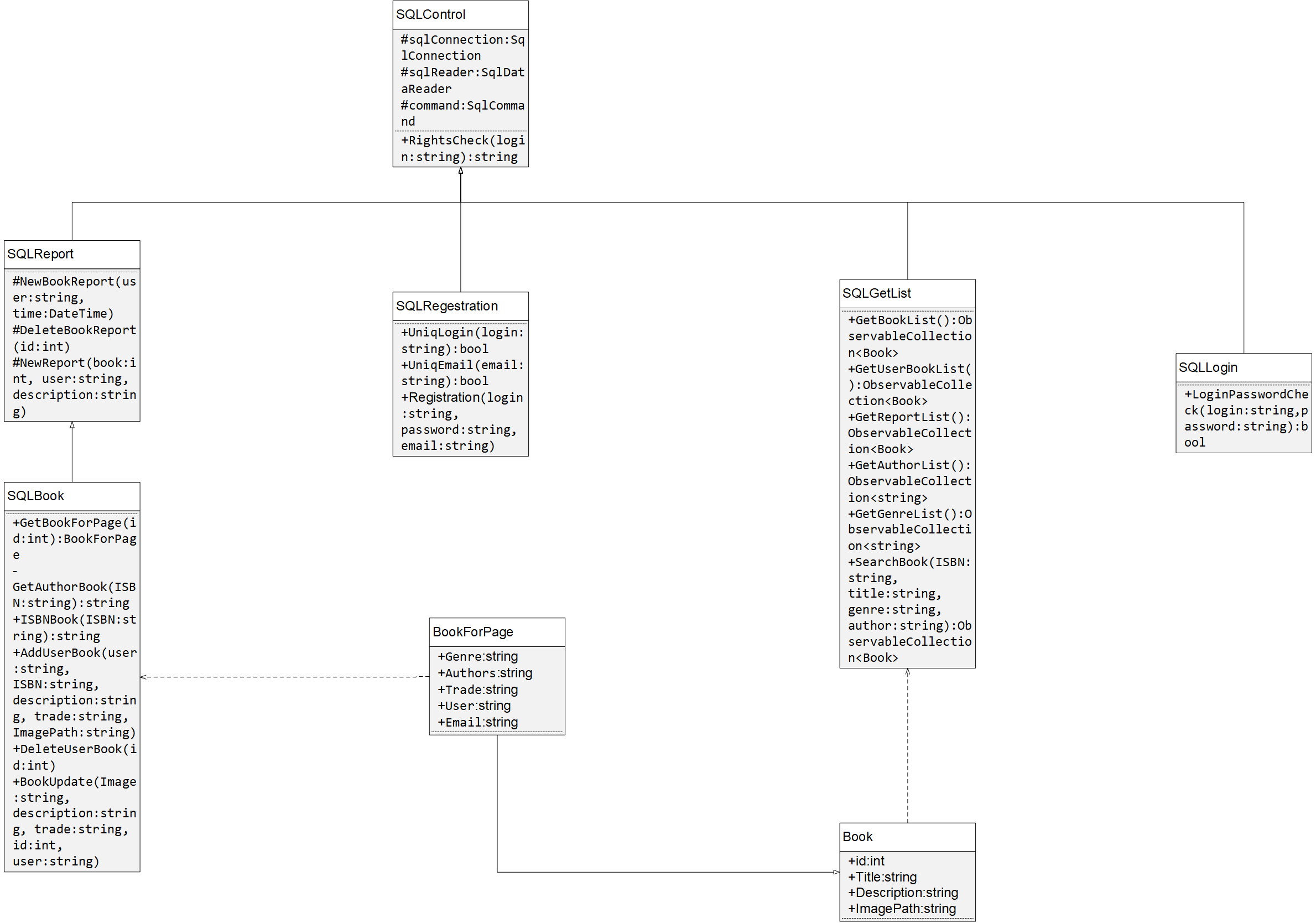


Рисунок 2 – Диаграмма классов

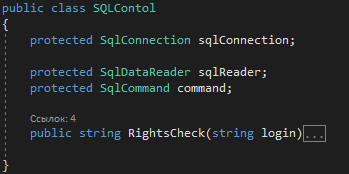


Рисунок 3 – Класс SQLControl

Класс SQLControl является основой для создания классов реализующий функционал. Атрибутами класса являются sqlConnection, sqlReader, command используемые для подключения базы данных и управления запросами.

Функция RightsCheck необходима для получения прав пользователя и используется в классах наследниках SQLControl.

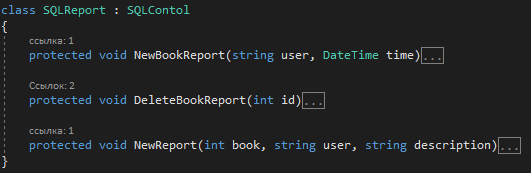


Рисунок 4 – Класс SQLReport

Класс SQLReport является наследником класса SQLControl и служит для создания отчетов, которые будут поступать администратору для проверки.

Функция NewBookReport служит для формирования отчета о том, что пользователь создал новую книгу.

Функция DeleteBookReport необходима для удаления проверенных или попросту не нужных отчетов.

Функция NewReport формирует новый отчет описание, которого поступает в аргументе description.

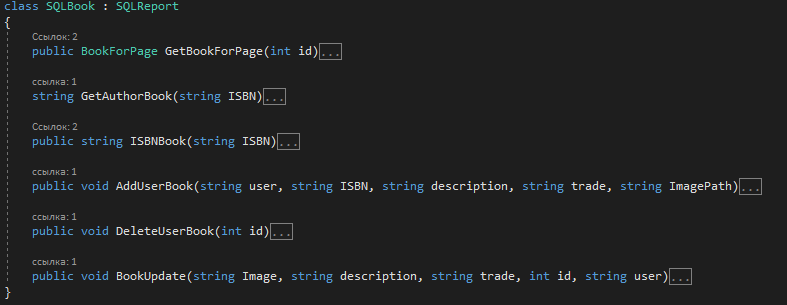


Рисунок 5 – Класс SQLBook

SQLBook – класс управления данными о книгах. Функции класса направлены на создание, удаление или изменение экземпляров книг, а также формированию отчета об этом благодаря функциям, наследуем от класса SQLReport.

Функция GetBookForPage служит для получения из базы данных экземпляра книги для заполнения страницы просмотра информации о книге.

GetAuthorBook функция для получения данных об авторах книги и формирования из них одной строки.

Функция ISBNBook необходима для нахождения книги по её ISBN или если она не найдена сообщать об этом.

AddUserBook функция для добавления в базу экземпляра пользовательской книги основываясь на книгах хранящийся в базе и данных введенным пользователем.

Функция DeleteUserBook удаляет определенный экземпляр пользовательской книги, а также удаляет все отчеты, с которым она была связана для сохранения целостности данных.

Функция BookUpdate служит для обновление данных о пользовательской книге и формирования о том, что изменения были внесены если их внес пользователь или удаление отчетов об этой книге если изменения внес администратор.

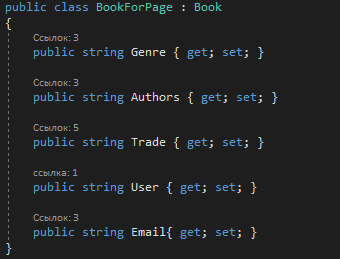


Рисунок 6– Класс BookForPage

Класс необходим для хранения данных получаемых из базы данных для заполнения страниц просмотра полной информации о конкретном экземпляре книги. Класс наследует часть своих атрибутов от класса Book.

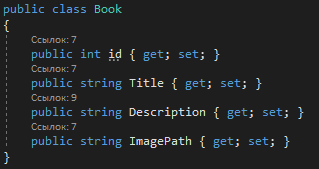


Рисунок 7– Класс Book

Класс для хранения данных получаемых из базы данных для заполнения страниц содержащих списки конкретных экземпляров книг.

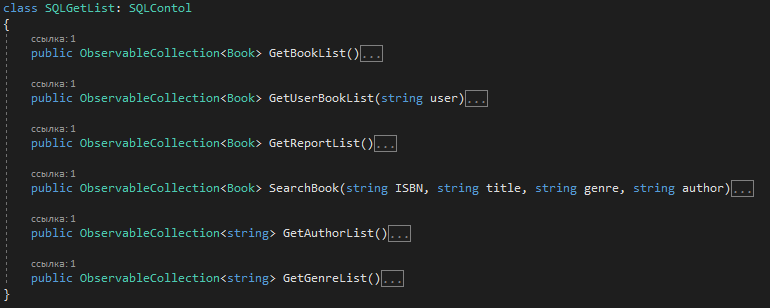


Рисунок 8– Класс SQLGetList

Класс SQLGetList служит для получения коллекций данных и базы данных для заполнения страниц и полей содержащих списки объектов.

GetBookList() функция получающая коллекцию тип Book для заполнения страницы с перечислением книг добавленных пользователями.

Функция GetUserBookList служит для получения данных для заполнения страницы книг конкретного пользователя.

GetReportList функция для получения списка отчетов, которые должен проверить администратор.

Функция SearchBook реализует поиск пользовательских книг по ISBN или названию, жанру и автору.

GetAuthorList возвращает список авторов в базе данных необходимый для реализации выпадающего списка на странице поиска.

GetGenreList возвращает список жанров в базе данных необходимый для реализации выпадающего списка на странице поиска.

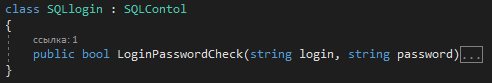


Рисунок 9– Класс SQLGetList

Класс SQLGetList необходим для реализации входа пользователя в приложении.

Функция LoginPasswordCheck проверяет введенные пользователем логин и пароль на соответствие с записанными в базе данных значениями и определяет верно введены данные или нет.

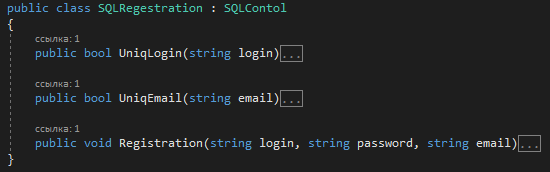


Рисунок 10– Класс SQLRegestration

SQLRegestration класс для реализации регистрации пользователя в системе.

Функция UniqLogin проверяет введенный пользователем логин на уникальность.

Функция UniqEmail проверяет введенный пользователем адрес электронной почты на уникальность.

Функция Registration производит регистрацию пользователя в системе занося его в базу данных.

## 3.4. Разработка интерфейса

### 3.4.1. Планирование

Для упрощения процесса разработки интерфейса необходимо сначала разработать его макет.

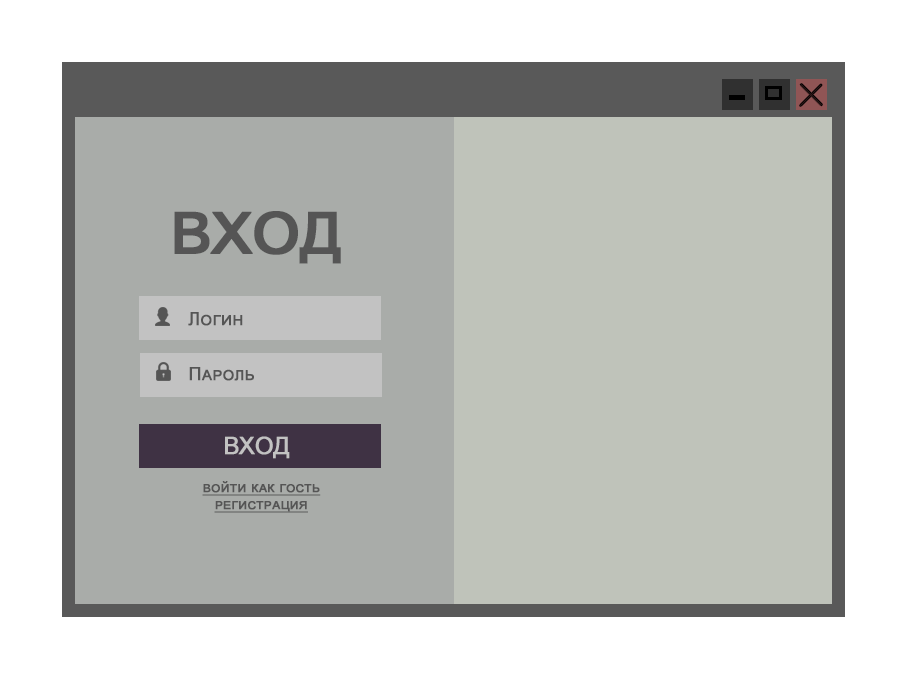


Рисунок 11– Макет страницы входа в приложение

Страница входа нужна для авторизации администратора или пользователя в приложении. Так же с данной страницы можно перейти на страницу регистрации.



Рисунок 12– Макет страницы регистрации

Назначение таблицы регистрации заключается в регистрации пользователя в приложении, после регистрации пользователь вернется на страницу входа.

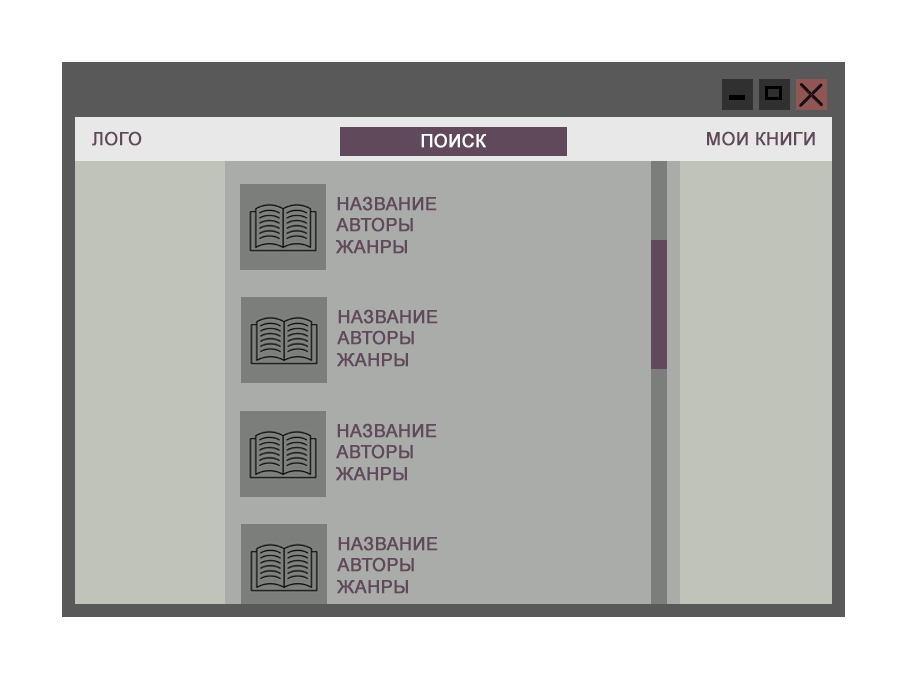


Рисунок 13– Макет страницы пользователя

На странице пользователя отображаются книги, загруженные другими пользователями. С этой страницы возможны переходы на страницу поиска, книг пользователя или просмотра подробной информации о книге.

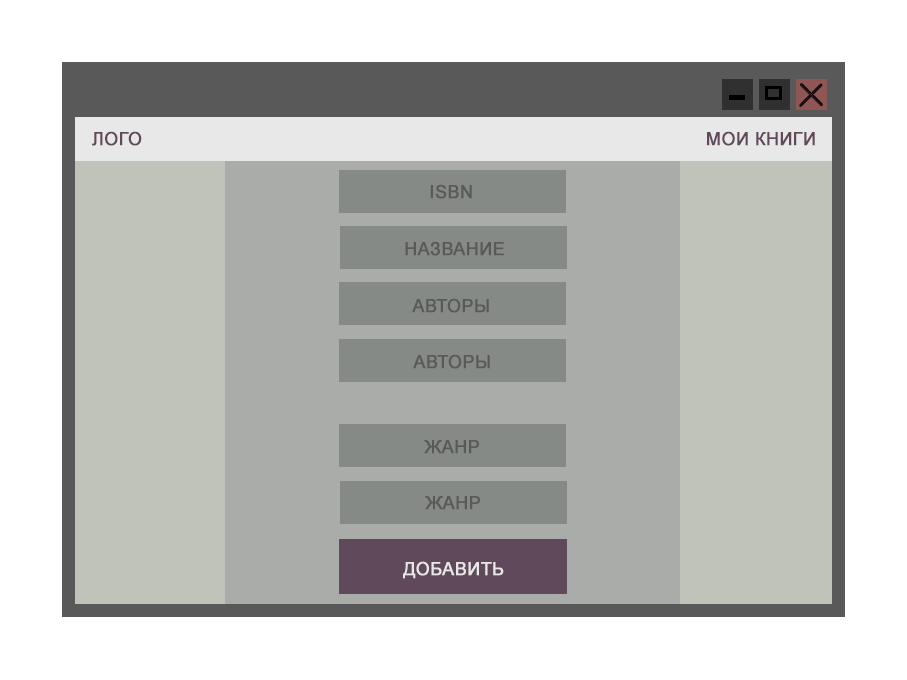


Рисунок 14– Макет страницы поиска

На данную страницу пользователь может использовать для поиска интересующий его книг, используя ISBN если он ищет конкретную книгу или поля название, автор и жанр если он не ищет конкретную книгу.

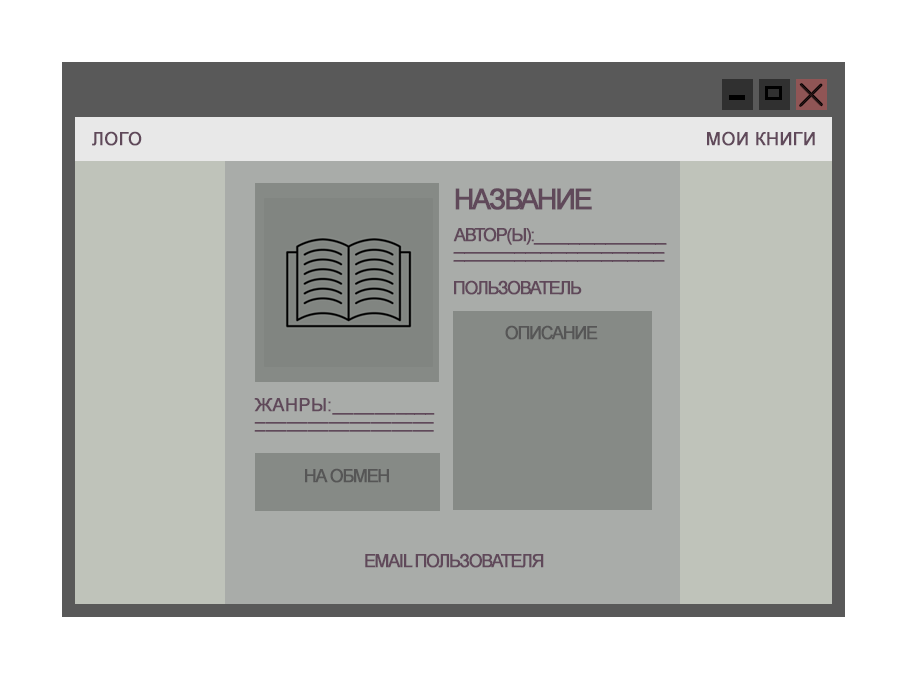


Рисунок 15– Макет страницы просмотра книги

Страница просмотра книги отображает информацию о пользовательской книге и общею информацию о экземпляре книги.

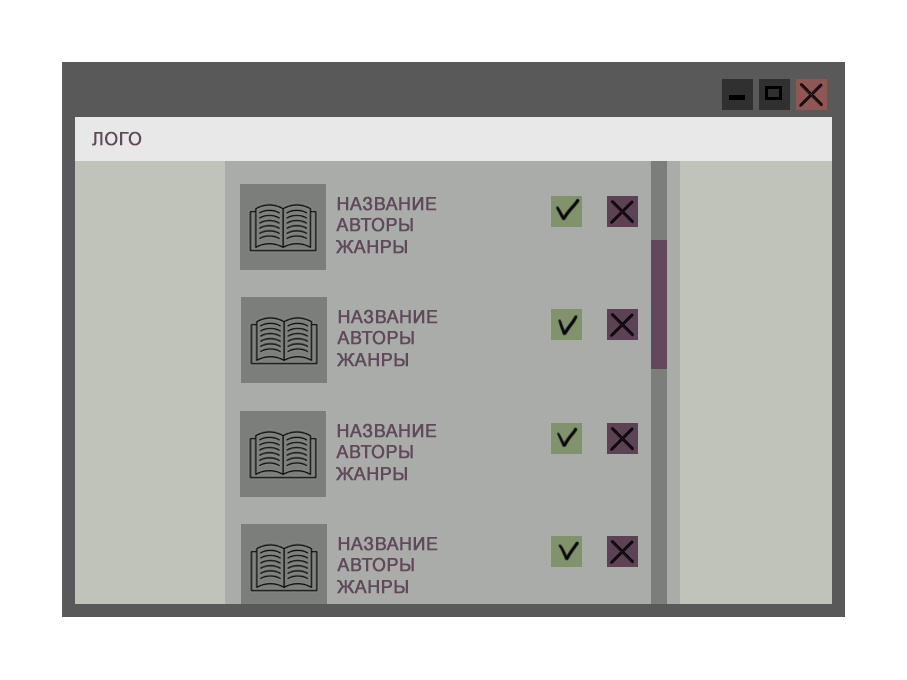


Рисунок 16– Макет страницы администратора

На странице администратора отображаются отчеты, формирующиеся при создании и изменении пользовательских книг, с нее администратор может просматривать пользовательские книги используя отчеты и редактировать или удалять их.

### 3.4.2. Реализация интерфейса

Для разработки интерфейса приложения была использована система для построения клиентских приложений Windows с визуальными возможностями взаимодействия с пользователем, графическая подсистема в составе .NET Framework, использующая язык XAML, WPF.

Приложение использует одно основное окно, в котором размещен фрейм, через который открываются различные страницы. Переход между страницами реализуется стандартными средствами навигации.

Так же на некоторых страницах есть свои фреймы для загрузки на них отдельных частей страниц, а не повторять одни и те же элементы интерфейса на всех страницах.

Для разметки страниц и облегчения расположения элементов на них используется Grid.

Для отображения списков элементов на страницах используется элемент listbox принимающий определенный тип данных из приложения и располагающий эти данные в отдельных элементах этой структуры.

## 3.5 Вывод

В ходе конструкторской части курсовой работы была определена структура приложения, которое будет выполнено в виде оконного приложения с возможностью выставить собственные книги для обмена или продажи, а также найти книги для других пользователей использую поиск или просматривая последние добавленные в приложения книги.

Так же в ходе работы были разработаны классы для реализации функциональной части приложения.

Был разработан интерфейс приложения с использованием WPF. Перед реализацией интерфейса был выполнен его макет для облегчения разработки.

# 4.ПРОЕКТНО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

## 4.1 Требования к аппаратной платформе.

Исходя из характеристик разработанного приложения и ресурсов, требуемых для его исправной работы необходимо, чтобы компьютер обладал следующими минимальными характеристиками:

* процессор: 1 ГГц и быстрее с поддержкой PAE, NX и SSE2;
* RAM: 1 Гбайт (32 бит) или 2 Гбайт (64 бит);
* HDD: 16 Гбайт (32 бит) или 20 Гбайт (64 бит);
* видеокарта: поддержка Microsoft DirectX 9 с драйвером WDDM;
* монитор;
* клавиатура;
* мышь.

## 4.2. Пользовательские сценарии

Разработав приложение можно взглянуть на пользовательские сценарии что, может помочь с дальнейшем тестирование приложения.

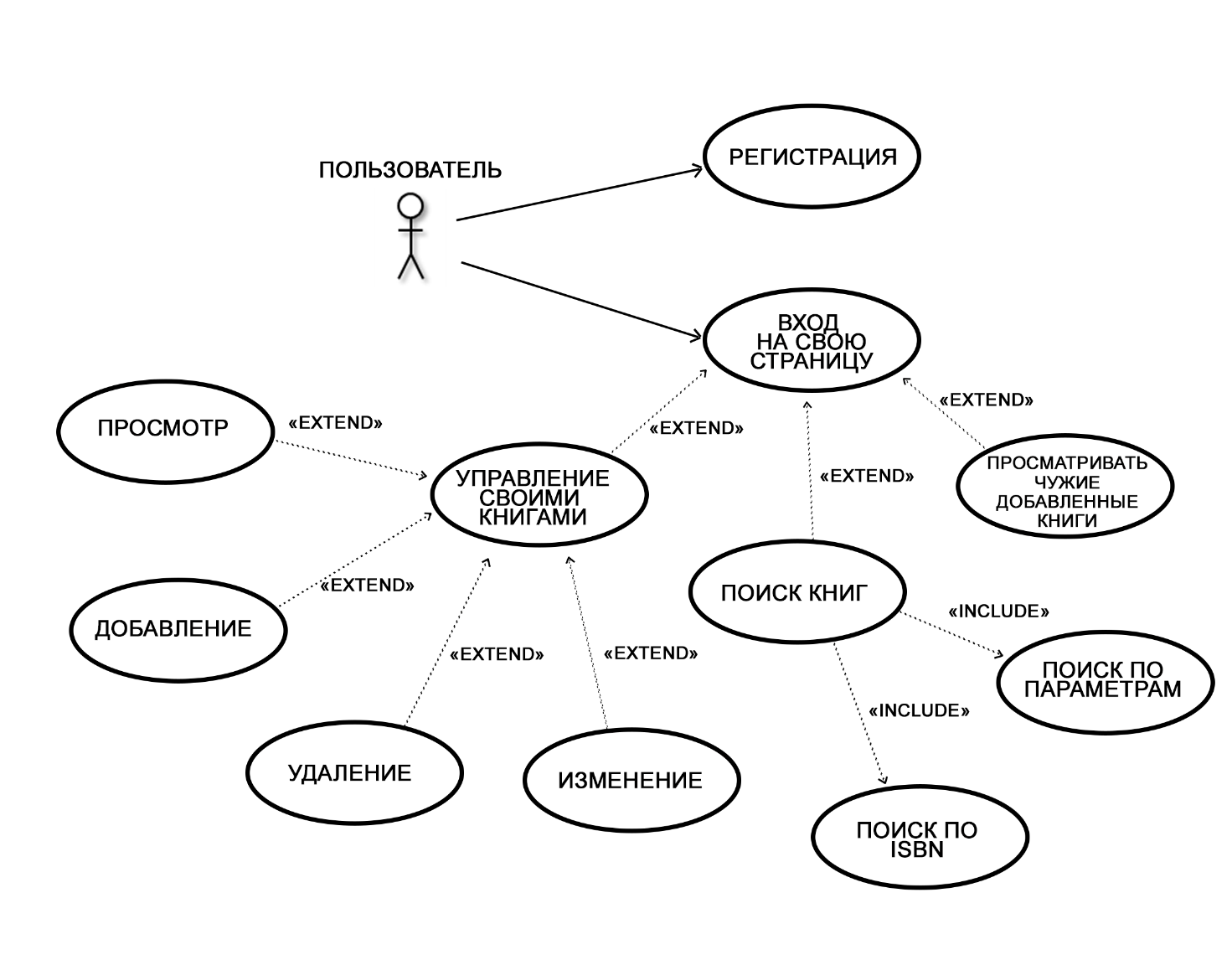


Рисунок 17 – Сценарий пользователя

Если пользователь запускает приложение впервые он должен зарегистрироваться. Если пользователь уже зарегистрирован, он вводит свои данные для входа и попадает на свою страницу. Зайдя на свою страницу, пользователь может просматривать свои книги, просматривать книги других пользователей и искать книги воспользовавшись поиском. Для поиска пользователь может использовать ISBN, либо параметры книги если он не ищет конкретный экземпляр книги. Просматривая свои книги, пользователь может просматривать, изменять, добавлять и удалять их.

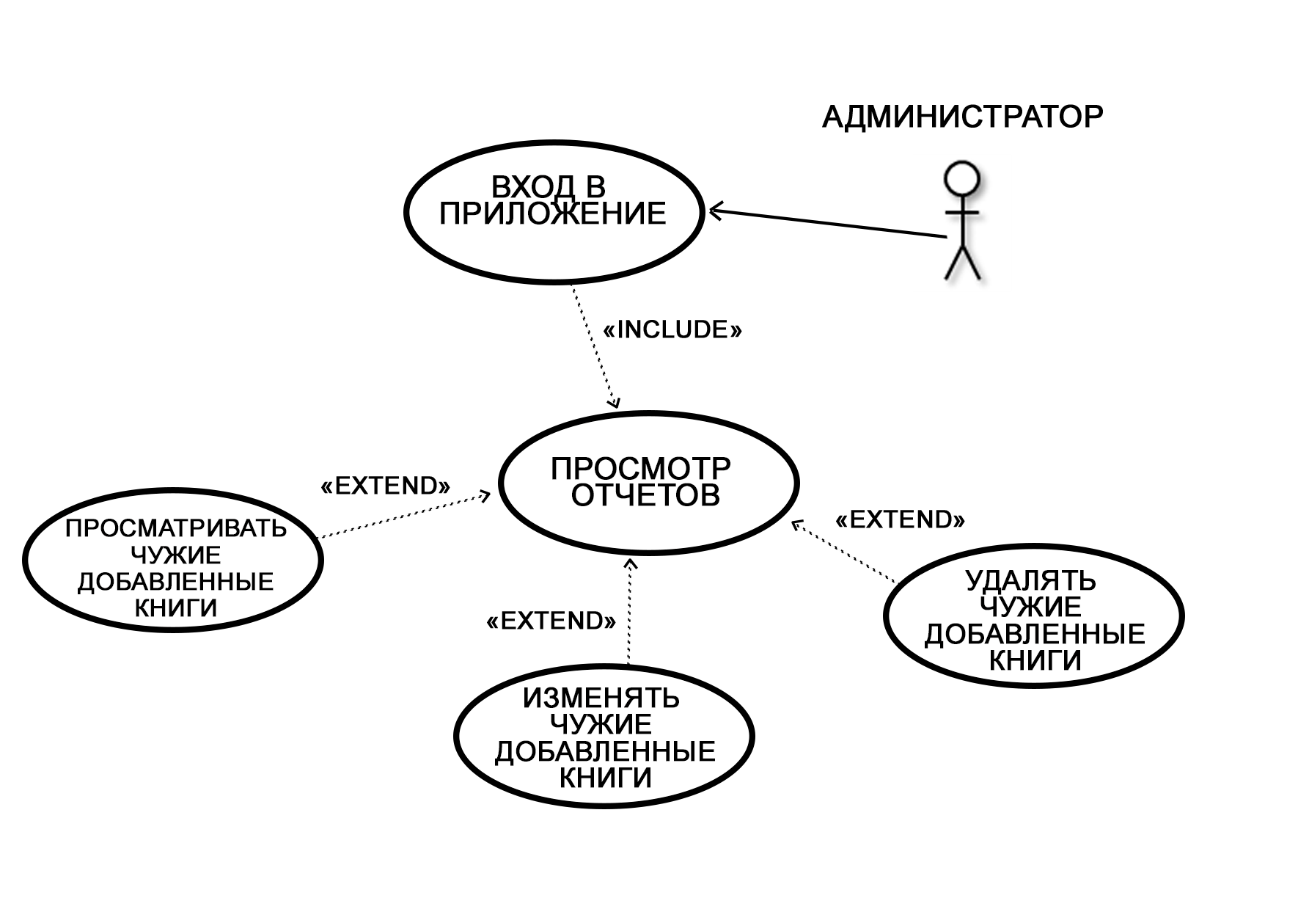


Рисунок 18 – Сценарий администратора

Администратор, входя в приложения должен ввести свои данные после чего попадет на страницу отчетов, используя которую он сможет просматривать книги пользователей, а также изменять или удалять их.

## 4.3. Тестирование приложения

Для тестирования приложения использовался персональная компьютер, соответствующий требованиям к аппаратной платформе.

В ходе тестирования были проверены различные функции приложения и по его окончании были исправлены обнаруженные ошибки.

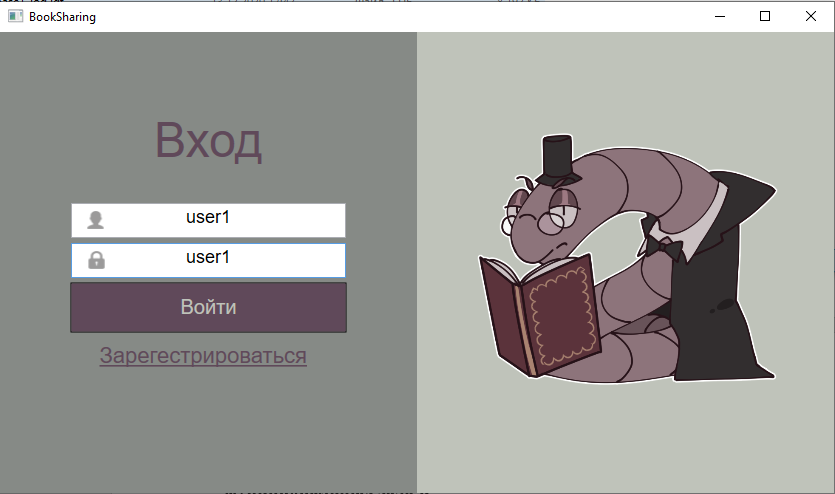


Рисунок 11 – Начальная страница

В ходе тестирования и исправления ошибок начальной страницы не было обнаружено критических ошибок, но было добавлено оповещения для пользователя о не верно введенном логине или пароле.

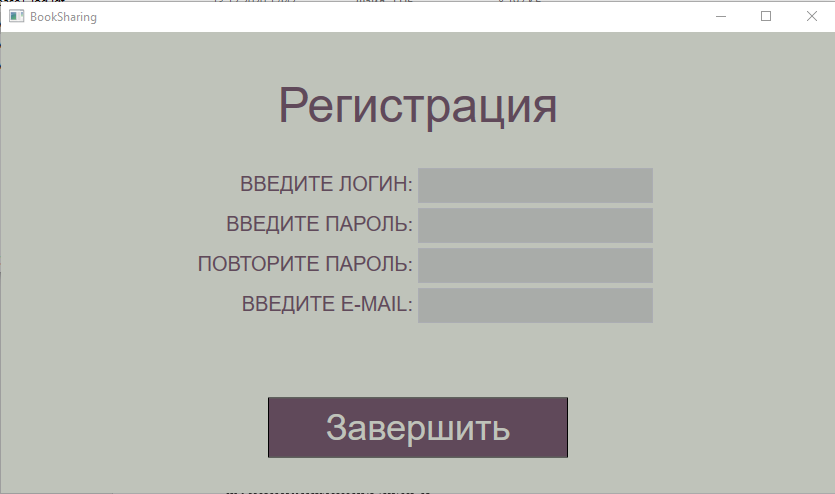


Рисунок 13 – Страница регистрации

При тестировании страницы регистрации была обнаружена возможность использовать при регистрации уже занятый другим пользователем email, для решения этой проблемы была разработана функция проверки email на уникальность и добавлено оповещение для пользователя что данная электронная почта уже используется.

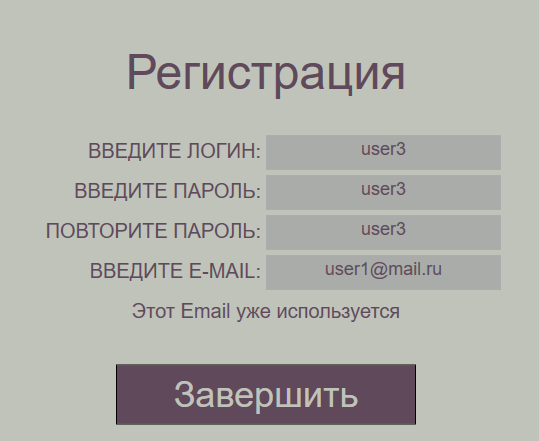


Рисунок 14 – Оповещении о занятой электронной почте

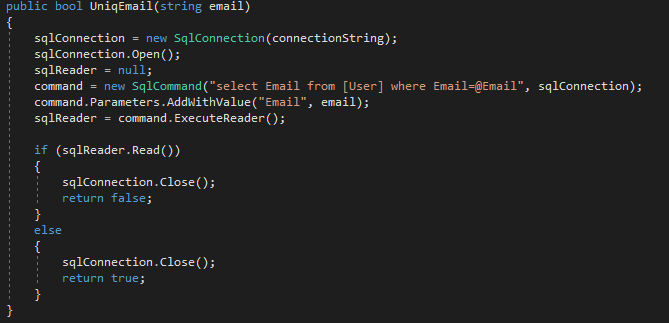


Рисунок 15 – Функция проверки email на уникальность

Также в ходе тестирования страницы регистрации было добавлено оповещение о необходимости заполнения пустых полей.

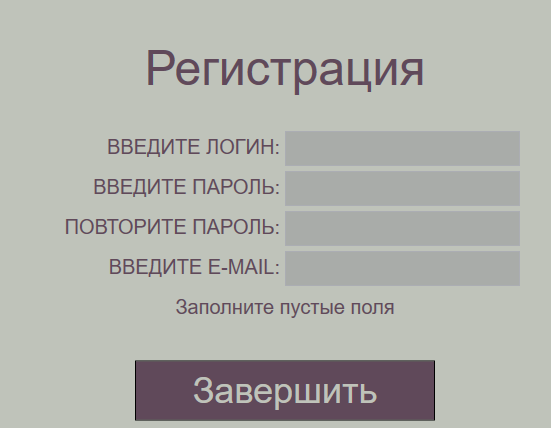


Рисунок 16 – Оповещении необходимости заполнения пустых полей

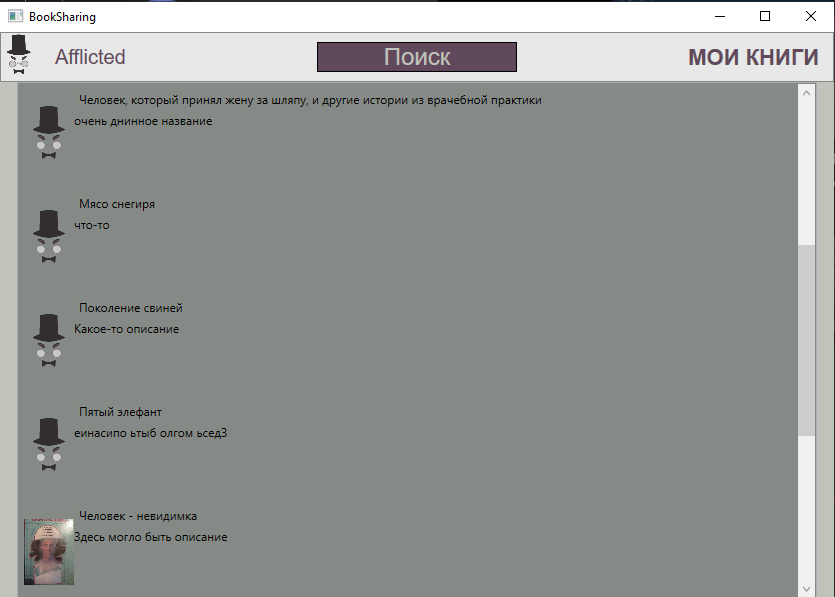


Рисунок 17 – Пользовательская страница

Проведя тестирование пользовательской страницы, на которой отображаются новые книги, добавленные пользователями ошибок обнаружено не было, но была добавлена сортировка списка книг по дате добавления их в базу.

При тестировании страницы добавления пользовательских книг была выявлена необходимость отображения есть ли книга с введенным ISBN в базе и какая эта книга, для этого была разработана функция и добавлено оповещение для пользователя с названием найденной книги или о том, что книга с введенным ISBN не была найдена в базе.

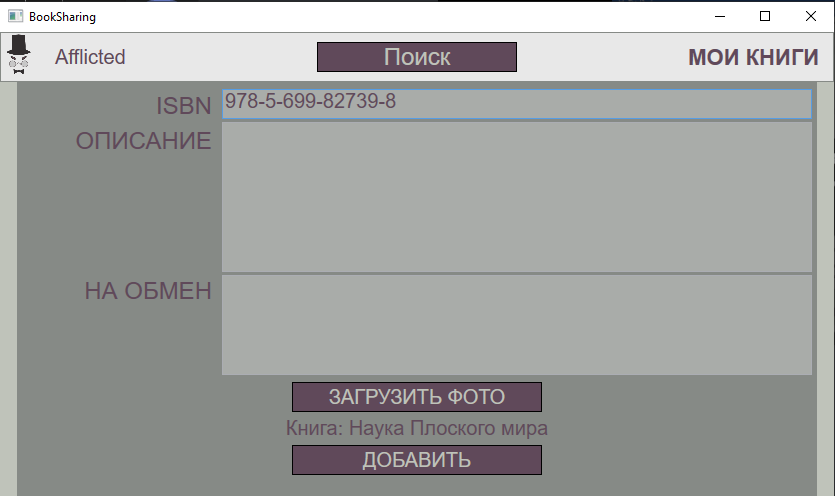


Рисунок 18 – Оповещении о найденной в базе книге

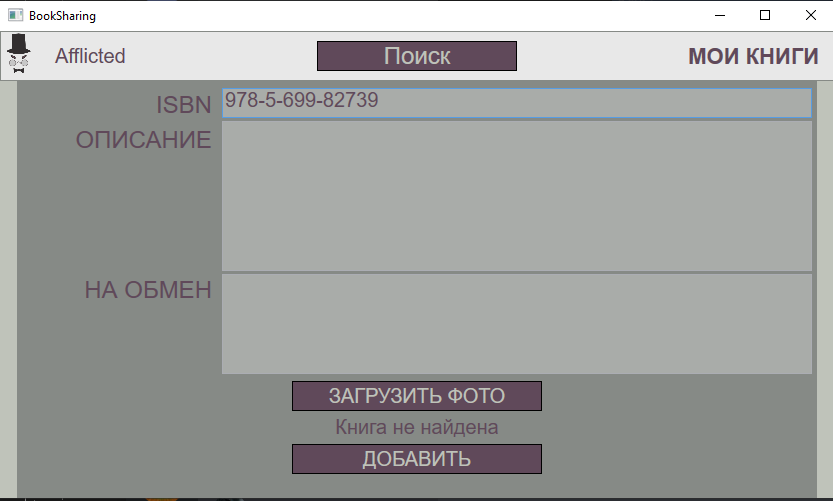


Рисунок 19 – Оповещении о том, что книга не найдена

При тестировании страницы поиска и результатов этого поиска была найдена критическая ошибка при поиске книг не использую поле автор, если у книги более одного автора она отображалась в результате поиска несколько раз. Данная ошибка была исправлена корректировкой SQL запроса.

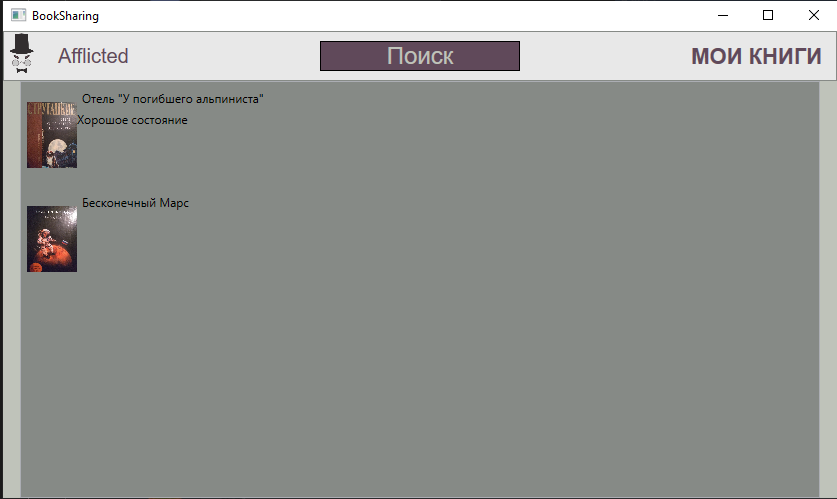


Рисунок 20 – Исправленное множественное появление одной и той же книги

Тестируя страницу редактирования пользовательской книги, была обнаружена ошибка нарушающая целостность базы данных при удалении книги пользователем, книга удалялась, но оставались отчеты, сформированные на основе этой книги. Ошибка была исправлена добавлением удаления отчетов о книге вместе с экземпляром самой книги.

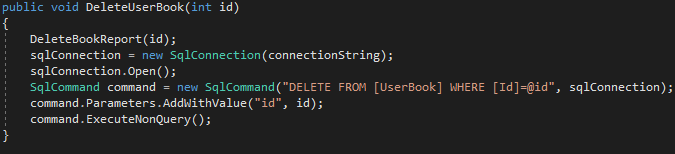


Рисунок 21 – Исправленная функция удаления книги

Проводя тестирование страницы просмотре книг других пользователей, была обнаружена ошибка отображения автора книги, если их было больше одного. Данная ошибка была исправлена внесением изменений в функцию получения авторов книги.

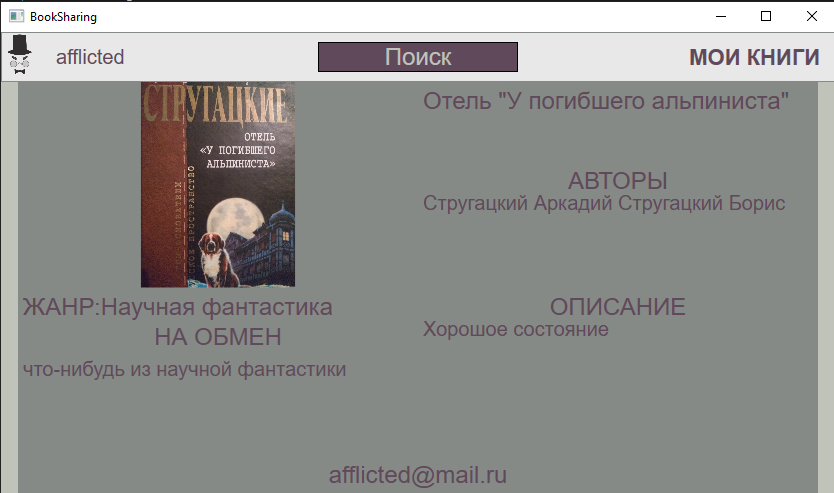


Рисунок 22 – Исправленная страница просмотра пользовательских книг

При тестировании функции администратора была обнаружено, что при редактировании книг пользователей, так же создавался отчет изменении книги, это приводило к генерации бесконечного количества отчетов. Проблема была решена изменением функции изменения книг.

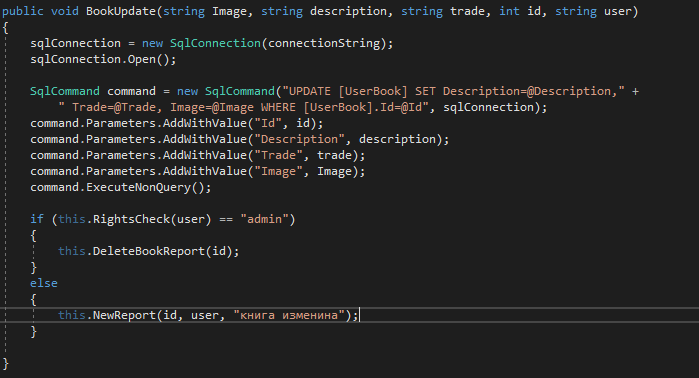


Рисунок 23 – Исправленная функция редактирования книг

## 3.5 Вывод

В ходе проектно-технологическая часть работы были определены системные требования для разработанного приложения.

Была разработана физическая модель базы данных на основании логической модели данных, а также было проведено тестирование приложения и исправление найденных ошибок.

# ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы было разработано оконное приложение «BookSharing» с возможностью выставить собственные книги для обмена или продажи, а также найти книги для других пользователей использую поиск или просматривая последние добавленные в приложения книги.

В ходе работы были проанализированы существующие аналоги, и на основе этого анализа был определен функционал необходимый приложению. После определения функционала были выбраны инструменты для решения поставленной задачи.

Для реализации функционала приложения были разработаны классы реализующие различные функции. Так же был разработан интерфейс приложения, перед реализацией которого был спроектирован его макет.

В ходе работы были достигнуты все поставленные задачи, а именно:

исследовать существующие аналоги приложений для обмена книгами для поиска проблем, которым можно будет избежать в проектирование приложения;

разработать базу данных для хранения информации пользователях и данных о книгах;

спроектировать и разработать приложение;

* провести тестирование приложения.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Абрамян, А.В. Разработка пользовательского интерфейса на основе технологии Windows Presentation Foundation: учебник по курсу «Основы разработки пользовательского интерфейса» для студентов направления 02.03.02 «Фундаментальная информатика и информационные технологии» (бакалавриат) / А.В. Абрамян, М.Э. Абрамян ; Южный федеральный университет. – Ростов-на-Дону ; Таганрог : Южный федеральный университет, 2018. – 302 с. : ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=499453 (дата обращения: 15.12.2020).
2. Влиссидес Д. Приемы объектно-ориентированного проектирования. Паттерны проектирования/ Э. Гамма, Р. Джонсон, Р. Хелм – Спб.: Питер, 2019. – 368 с.
3. Горелов, С.В. Современные технологии программирования: разработка Windows-приложений на языке С#: учебник для студентов, обучающихся по дисциплине «Современные технологии программирования», направление «Прикладная информатика» (09.03.03 — для бакалавров, 09.04.03 — для магистров) : в 2 томах : [16+] / С.В. Горелов ; под науч. ред. П.Б. Лукьянова ; Финансовый университет при Правительстве Российской Федерации. – Москва : Прометей, 2019. – Том 1. – 363 с. : ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=576037 (дата обращения: 15.12.2020).
4. Документация по С# [Электронный ресурс]/ Microsoft – URL: https://docs.microsoft.com/ru-ru/dotnet/csharp//(дата обращения 15.12.2020).
5. Зайцев, М.Г. Объектно-ориентированный анализ и программирование : учебное пособие : [16+] / М.Г. Зайцев ; Новосибирский государственный технический университет. – Новосибирск : Новосибирский государственный технический университет, 2017. – 84 с. : ил., табл. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=576800 (дата обращения: 15.12.2020).
6. Кудрявцев, К. Я. Создание баз данных : учебное пособие / К. Я. Кудрявцев. — Москва : НИЯУ МИФИ, 2010. — 155 с. — ISBN 978-5-7262-1302-6. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/75822 (дата обращения: 15.12.2020).
7. Рояк, М.Э. Программирование под Windows графических интерфейсов пользователя : учебное пособие : [16+] / М.Э. Рояк, И.М. Ступаков ; Новосибирский государственный технический университет. – Новосибирск : Новосибирский государственный технический университет, 2018. – 72 с. : ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=575018 (дата обращения: 15.12.2020).
8. Тарасов, С. В. СУБД для программиста. Базы данных изнутри / С. В. Тарасов. — Москва : СОЛОН-Пресс, 2015. — 320 с. — ISBN 978-2-7466-7383-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/64959 (дата обращения: 15.12.2020)
9. Шнырёв, С. Л. Базы данных : учебное пособие / С. Л. Шнырёв. — Москва : НИЯУ МИФИ, 2011. — 224 с. — ISBN 978-5-7262-1483-2. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/75809 (дата обращения: 15.12.2020).
10. Microsoft Visual Studio [Электронный ресурс]/ Википедия – Режим доступа: https://ru.wikipedia.org/wiki/Microsoft\_Visual\_Studio(дата обращения 15.12.2020).

# ПРИЛОЖЕНИЕ А

SQLControl:

using System;

using System.Data.SqlClient;

namespace BookSharing

{

public class SQLContol

{

protected SqlConnection sqlConnection;

protected string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\Afflicted\Desktop\BookSharing\WpfApp1\Database1.mdf;Integrated Security=True";

protected SqlDataReader sqlReader;

protected SqlCommand command;

public string RightsCheck(string login)

{

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

sqlReader = null;

command = new SqlCommand("select Rights from [User] where Login=@Login", sqlConnection);

command.Parameters.AddWithValue("Login", login);

sqlReader = command.ExecuteReader();

sqlReader.Read();

return Convert.ToString(sqlReader["Rights"]);

}

}

}

SQLReport:

using System;

using System.Data.SqlClient;

namespace BookSharing

{

class SQLReport : SQLContol

{

protected void NewBookReport(string user, DateTime time)

{

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

sqlReader = null;

command = new SqlCommand("select Id from [UserBook] where [UserBook].[User]=@user and [UserBook].[Date]=@date", sqlConnection);

command.Parameters.AddWithValue("user", user);

command.Parameters.AddWithValue("date", time);

sqlReader = command.ExecuteReader();

sqlReader.Read();

int id = Convert.ToInt32(sqlReader["id"]);

sqlConnection.Close();

sqlConnection.Open();

command = new SqlCommand("INSERT INTO [Report] ([Report].[User],Book, Description, Date) Values (@User,@Book, @Description, @Date)", sqlConnection);

command.Parameters.AddWithValue("User", user);

command.Parameters.AddWithValue("Book", id);

command.Parameters.AddWithValue("Description", "книга добавлена");

command.Parameters.AddWithValue("Date", time);

command.ExecuteNonQuery();

}

protected void DeleteBookReport(int id)

{

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlCommand command = new SqlCommand("DELETE FROM [Report] WHERE [Book]=@Book", sqlConnection);

command.Parameters.AddWithValue("Book", id);

command.ExecuteNonQuery();

}

protected void NewReport(int book, string user, string description)

{

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

command = new SqlCommand("INSERT INTO [Report] ([Report].[User],Book, Description, Date) Values (@User,@Book, @Description, @Date)", sqlConnection);

command.Parameters.AddWithValue("User", user);

command.Parameters.AddWithValue("Book", book);

command.Parameters.AddWithValue("Description", description);

command.Parameters.AddWithValue("Date", DateTime.Now);

command.ExecuteNonQuery();

}

}

}

SQLLogin:

using System;

using System.Data.SqlClient;

namespace BookSharing

{

class SQLlogin : SQLContol

{

public bool LoginPasswordCheck(string login, string password)

{

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

sqlReader = null;

command = new SqlCommand("select \* from [User] where Login=@Login and Password=@Password", sqlConnection);

command.Parameters.AddWithValue("Login", login);

command.Parameters.AddWithValue("Password", password);

sqlReader = command.ExecuteReader();

if (sqlReader.Read())

{

sqlConnection.Close();

return true;

}

else

{

sqlConnection.Close();

return false;

}

}

}

}

SQLGetList:

using System;

using System.Data.SqlClient;

using System.Collections.ObjectModel;

namespace BookSharing

{

class SQLGetList: SQLContol

{

public ObservableCollection<Book> GetBookList()

{

ObservableCollection<Book> Books = new ObservableCollection<Book>();

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlCommand command = new SqlCommand("select \* from [UserBook] INNER JOIN [Book] on [Book].ISBN = [UserBook].ISBN order by Date DESC", sqlConnection);

sqlReader = command.ExecuteReader();

while (sqlReader.Read())

{

Books.Add(new Book { id = Convert.ToInt32(sqlReader["Id"]), ImagePath = Convert.ToString(sqlReader["Image"]), Title = Convert.ToString(sqlReader["Name"]), Description = Convert.ToString(sqlReader["Description"]) });

}

return Books;

}

public ObservableCollection<Book> GetUserBookList(string user)

{

ObservableCollection<Book> Books = new ObservableCollection<Book>();

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlCommand command = new SqlCommand("select \* from [UserBook] INNER JOIN [Book] on [Book].ISBN = [UserBook].ISBN where [UserBook].[User] = @User order by Date DESC", sqlConnection);

command.Parameters.AddWithValue("User", user);

sqlReader = command.ExecuteReader();

while (sqlReader.Read())

{

Books.Add(new Book { id = Convert.ToInt32(sqlReader["Id"]), ImagePath = Convert.ToString(sqlReader["Image"]), Title = Convert.ToString(sqlReader["Name"]), Description = Convert.ToString(sqlReader["Description"]) });

}

return Books;

}

public ObservableCollection<Book> GetReportList()

{

ObservableCollection<Book> Books = new ObservableCollection<Book>();

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlCommand command = new SqlCommand("select [UserBook].Id as id, Image, Name, [Report].Description as Description from [UserBook] INNER JOIN [Book] on [Book].ISBN = [UserBook].ISBN INNER JOIN [Report] on [Report].[Book] = [UserBook].[Id] order by [Report].Date", sqlConnection);

sqlReader = command.ExecuteReader();

while (sqlReader.Read())

{

Books.Add(new Book { id = Convert.ToInt32(sqlReader["id"]), ImagePath = Convert.ToString(sqlReader["Image"]), Title = Convert.ToString(sqlReader["Name"]), Description = Convert.ToString(sqlReader["Description"]) });

}

return Books;

}

public ObservableCollection<Book> SearchBook(string ISBN, string title, string genre, string author)

{

ObservableCollection<Book> Books = new ObservableCollection<Book>();

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

string request = "SELECT [UserBook].[Id], Image, Description, Name FROM [UserBook] INNER JOIN [Book] on [Book].ISBN = [UserBook].ISBN INNER JOIN [BookAuthor] on [Book].ISBN = [BookAuthor].[Book]";

if (string.IsNullOrEmpty(ISBN))

{

if (!string.IsNullOrEmpty(title))

{

request += "WHERE [Book].[Name] like '%'+ @Title +'%' ";

}

else

{

if (!string.IsNullOrEmpty(genre))

{

request += "WHERE Genre=@Genre";

if (!string.IsNullOrEmpty(author))

{

request += " AND Author=@Author";

}

}

else

{

request += "WHERE Author=@Author";

}

}

}

else

{

request += "WHERE [UserBook].ISBN=@ISBN";

}

request += " GROUP BY [UserBook].[Id], Image, Description, Name ";

SqlCommand command = new SqlCommand(request, sqlConnection);

command.Parameters.AddWithValue("ISBN", ISBN);

command.Parameters.AddWithValue("Title", title);

command.Parameters.AddWithValue("Genre", genre);

command.Parameters.AddWithValue("Author", author);

sqlReader = command.ExecuteReader();

while (sqlReader.Read())

{

Books.Add(new Book { id = Convert.ToInt32(sqlReader["Id"]), ImagePath = Convert.ToString(sqlReader["Image"]), Title = Convert.ToString(sqlReader["Name"]), Description = Convert.ToString(sqlReader["Description"]) });

}

return Books;

}

public ObservableCollection<string> GetAuthorList()

{

ObservableCollection<string> Authors = new ObservableCollection<string>();

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlCommand command = new SqlCommand("select \* from [Author]", sqlConnection);

sqlReader = command.ExecuteReader();

while (sqlReader.Read())

{

Authors.Add(Convert.ToString(sqlReader["Name"]));

}

return Authors;

}

public ObservableCollection<string> GetGenreList()

{

ObservableCollection<string> Genre = new ObservableCollection<string>();

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlCommand command = new SqlCommand("select \* from [Genre]", sqlConnection);

sqlReader = command.ExecuteReader();

while (sqlReader.Read())

{

Genre.Add(Convert.ToString(sqlReader["Title"]));

}

return Genre;

}

}

}

SQLGetList:

using System;

using System.Data.SqlClient;

namespace BookSharing

{

class SQLBook : SQLReport

{

public BookForPage GetBookForPage(int id)

{

BookForPage Book = new BookForPage();

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlCommand command = new SqlCommand("select \* from [UserBook] INNER JOIN [Book] on [Book].ISBN = [UserBook].ISBN INNER JOIN [User] on [UserBook].[User] = [User].Login where [UserBook].[Id] = @id", sqlConnection);

command.Parameters.AddWithValue("id", id);

sqlReader = command.ExecuteReader();

sqlReader.Read();

Book.Title = Convert.ToString(sqlReader["Name"]);

Book.Genre = Convert.ToString(sqlReader["Genre"]);

Book.ImagePath = Convert.ToString(sqlReader["Image"]);

Book.Description = Convert.ToString(sqlReader["Description"]);

Book.User = Convert.ToString(sqlReader["User"]);

Book.Email = Convert.ToString(sqlReader["Email"]);

Book.Trade = Convert.ToString(sqlReader["Trade"]);

Book.Authors = GetAuthorBook(Convert.ToString(sqlReader["ISBN"]));

return Book;

}

string GetAuthorBook(string ISBN)

{

string Author = "";

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlCommand command = new SqlCommand("select Author from [BookAuthor] where [Book] = @Book", sqlConnection);

command.Parameters.AddWithValue("Book", ISBN);

sqlReader = command.ExecuteReader();

while (sqlReader.Read())

{

Author += Convert.ToString(sqlReader["Author"]) + " ";

}

return Author;

}

public string ISBNBook(string ISBN)

{

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

sqlReader = null;

command = new SqlCommand("select Name from [Book] where ISBN=@ISBN", sqlConnection);

command.Parameters.AddWithValue("ISBN", ISBN);

sqlReader = command.ExecuteReader();

if (sqlReader.Read())

{

return "Книга: " + Convert.ToString(sqlReader["Name"]);

}

else

{

sqlConnection.Close();

return "Книга не найдена";

}

}

public void AddUserBook(string user, string ISBN, string description, string trade, string ImagePath)

{

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlCommand command = new SqlCommand("INSERT INTO [UserBook] ([UserBook].[User], Description, Trade, ISBN, Image, Date) Values (@User, @Description, @Trade, @ISBN, @Image, @Date)", sqlConnection);

command.Parameters.AddWithValue("User", user);

command.Parameters.AddWithValue("Description", description);

command.Parameters.AddWithValue("Trade", trade);

command.Parameters.AddWithValue("ISBN", ISBN);

if (ImagePath == "")

{

ImagePath = @"C:\Users\Afflicted\Desktop\BookSharing\WpfApp1\logo.png";

}

command.Parameters.AddWithValue("Image", ImagePath);

DateTime time = DateTime.Now;

command.Parameters.AddWithValue("Date", time);

command.ExecuteNonQuery();

this.NewBookReport(user, time);

}

public void DeleteUserBook(int id)

{

DeleteBookReport(id);

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlCommand command = new SqlCommand("DELETE FROM [UserBook] WHERE [Id]=@id", sqlConnection);

command.Parameters.AddWithValue("id", id);

command.ExecuteNonQuery();

}

public void BookUpdate(string Image, string description, string trade, int id, string user)

{

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlCommand command = new SqlCommand("UPDATE [UserBook] SET Description=@Description, Trade=@Trade, Image=@Image WHERE [UserBook].Id=@Id", sqlConnection);

command.Parameters.AddWithValue("Id", id);

command.Parameters.AddWithValue("Description", description);

command.Parameters.AddWithValue("Trade", trade);

command.Parameters.AddWithValue("Image", Image);

command.ExecuteNonQuery();

if (this.RightsCheck(user) == "admin")

{

this.DeleteBookReport(id);

}

else

{

this.NewReport(id, user, "книга изменина");

}

}

}

}

SQLRegistration:

using System;

using System.Data.SqlClient;

namespace BookSharing

{

public class SQLRegestration : SQLContol

{

public bool UniqLogin(string login)

{

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

sqlReader = null;

command = new SqlCommand("select Login from [User] where Login=@Login", sqlConnection);

command.Parameters.AddWithValue("Login", login);

sqlReader = command.ExecuteReader();

if (sqlReader.Read())

{

sqlConnection.Close();

return false;

}

else

{

sqlConnection.Close();

return true;

}

}

public bool UniqEmail(string email)

{

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

sqlReader = null;

command = new SqlCommand("select Email from [User] where Email=@Email", sqlConnection);

command.Parameters.AddWithValue("Email", email);

sqlReader = command.ExecuteReader();

if (sqlReader.Read())

{

sqlConnection.Close();

return false;

}

else

{

sqlConnection.Close();

return true;

}

}

public void Registration(string login, string password, string email)

{

sqlConnection = new SqlConnection(connectionString);

sqlConnection.Open();

SqlCommand command = new SqlCommand("INSERT INTO [User] (Login, Password, Email, Rights) Values (@Login, @Password, @Email, @Rights)", sqlConnection);

command.Parameters.AddWithValue("Login", login);

command.Parameters.AddWithValue("Password", password);

command.Parameters.AddWithValue("Email", email);

command.Parameters.AddWithValue("Rights", "user");

command.ExecuteNonQuery();

}

}

}

Book:

namespace BookSharing

{

public class Book

{

public int id { get; set; }

public string Title { get; set; }

public string Description { get; set; }

public string ImagePath { get; set; }

}

}

BookForPage:

namespace BookSharing

{

public class BookForPage : Book

{

public string Genre { get; set; }

public string Authors { get; set; }

public string Trade { get; set; }

public string User { get; set; }

public string Email{ get; set; }

}

}