**Министерство образования Московской области**

**ГОУ ВО МО «Государственный гуманитарно-технологический университет» (ГГТУ) Ликино-Дулевский политехнический колледж- филиал ГГТУ**

Наименование колледжа/техникума

**О Т Ч Ё Т**

**ПО УЧЕБНОЙ ПРАКТИКЕ**

ПМ.01 Разработка модулей программного обеспечения для компьютерных систем

(код и наименование ПМ)

Обучающейся\_\_\_\_Федяевой Ольги Сергеевны \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ф.И.О.

Курс \_\_\_\_3\_\_\_\_\_ группа \_\_\_ИСП.22А\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Специальность 09.02.07 Информационные системы и программирование \_\_\_

Место практики Ликино-Дулевский политехнический колледж – филиал ГГТУ

Период практики 25.11.2024 г. – 07.12.2024 г., 07.04.2025 г. - 19.04.2025 г.

Руководители практики

от колледжа \_\_\_\_\_\_\_\_ Кузьмина Елена Евгеньевна \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Гжегожевский Сергей Владимирович\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ветрова Карина Валерьевна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тишкин Валентин Васильевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ф.И.О.

г. Ликино-Дулево

2025 г.

Оглавление

[Введение 3](#_Toc184291046)

[1.Разработка технического задания 4](#_Toc184291047)

[2. Проектирование и создание базы данных 9](#_Toc184291048)

[2.1 Таблицы с данными 10](#_Toc184291049)

[3. Разработка приложения 14](#_Toc184291050)

[4. Разработка программных модулей 21](#_Toc184291051)

[4.1 Использование библиотек 51](#_Toc184291052)

[4.2 Использование функций и методов 52](#_Toc184291053)

[4.3 Таблицы базы данных 53](#_Toc184291054)

[4.4 Основные кнопки приложения 53](#_Toc184291055)

[5. Разработка API 55](#_Toc184291056)

[6. Библиотека классов 60](#_Toc184291057)

[Заключение 64](#_Toc184291058)

# Введение

Целью учебной практики является закрепление и углубление знаний, полученных обучающимися в процессе теоретического обучения по профессиональному модулю ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем».

Задачи практики:

1. Получение первичных профессиональных умений и навыков;

2. Подготовка студентов к осознанному и глубокому изучению общепрофессиональных и специальных дисциплин;

3. Получение практических профессиональных умений и навыков по специальности 09.02.07 Информационные системы и программирование.

С целью овладения видом деятельности «Разработка модулей программного обеспечения для компьютерных систем» обучающийся должен обладать соответствующими профессиональными компетенциями:

ПК 1.1. Формировать алгоритмы разработки программных модулей в соответствии с техническим заданием;

ПК 1.2. Разрабатывать программные модули в соответствии с техническим заданием;

ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств;

ПК 1.4. Выполнять тестирование программных модулей;

# Разработка технического задания

Задание:

Разработать социальную сеть для любителей книг – сайт для обсуждения книг, рекомендаций и ведения личных библиотек.

*1. Ключевые сущности выбранной предметной области:*

Сущность: Пользователи.

Атрибуты: ФИО, Email, пароль, дата, фото.

Сущность: Книги.

Атрибуты: Название, год публикации, количество страниц, описание, фотография.

Сущность: Авторы.

Атрибуты: ФИО, дата рождения.

Сущность: Отзывы.

Атрибуты: Рейтинг, текст отзыва, дата отзыва.

Сущность: Полка для книг.

Атрибуты: Дата добавления, статус.

Сущность: Жанр.

Атрибуты: Название.

Сущность: Друзья.

Процессы:

- просмотр аккаунта и редактирование;

- просмотр и добавление книг;

- добавление книг на свою полку;

- добавление и удаление из друзей;

- добавление отзыва, просмотр всех отзывов.

*2. Диаграмма сущностей и связей*

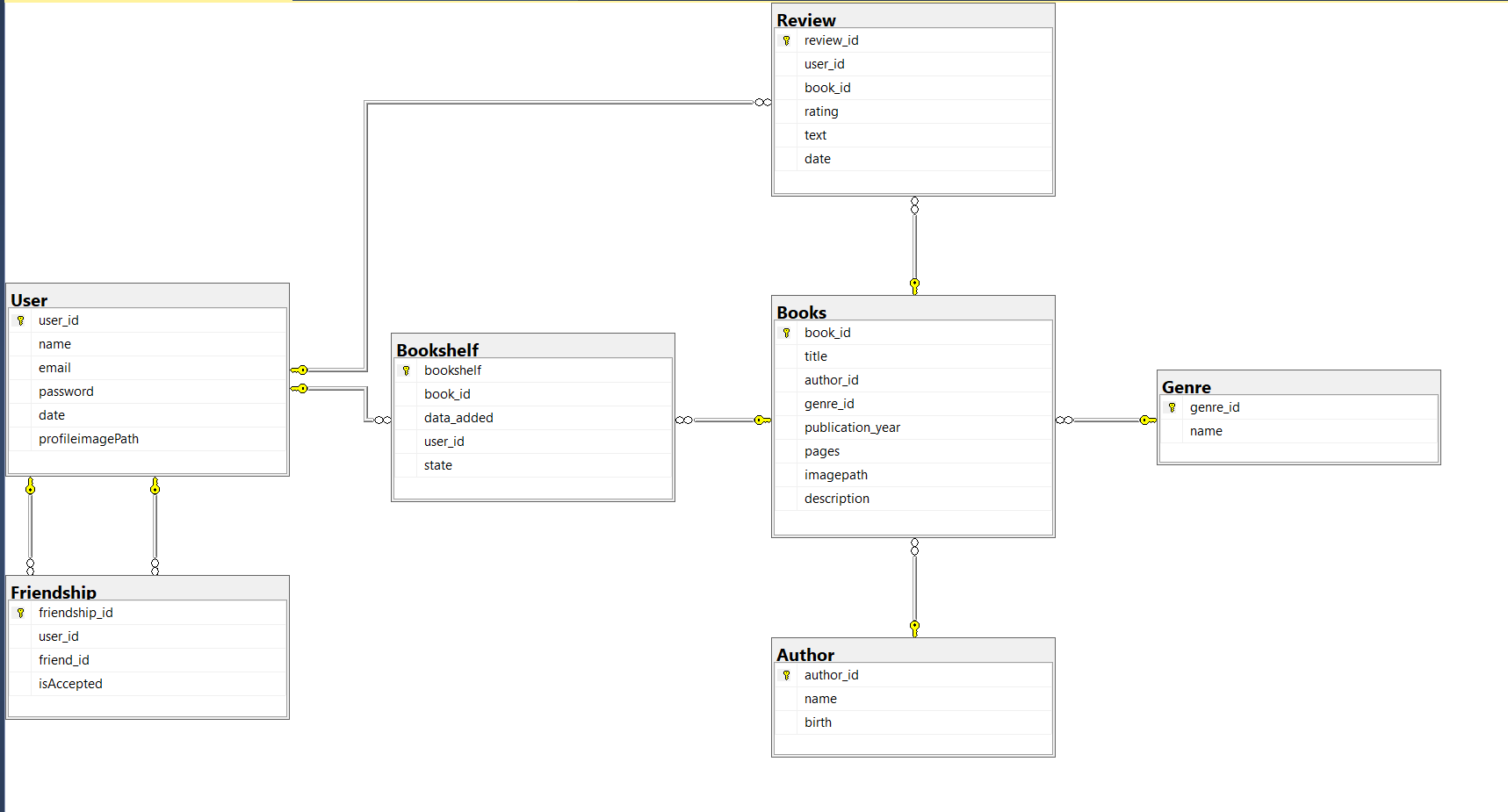


Рис. 1 «ER- диаграмма»

*3. Разработка блок схемы и бизнес- процессов, иллюстрирующие работу в системе*

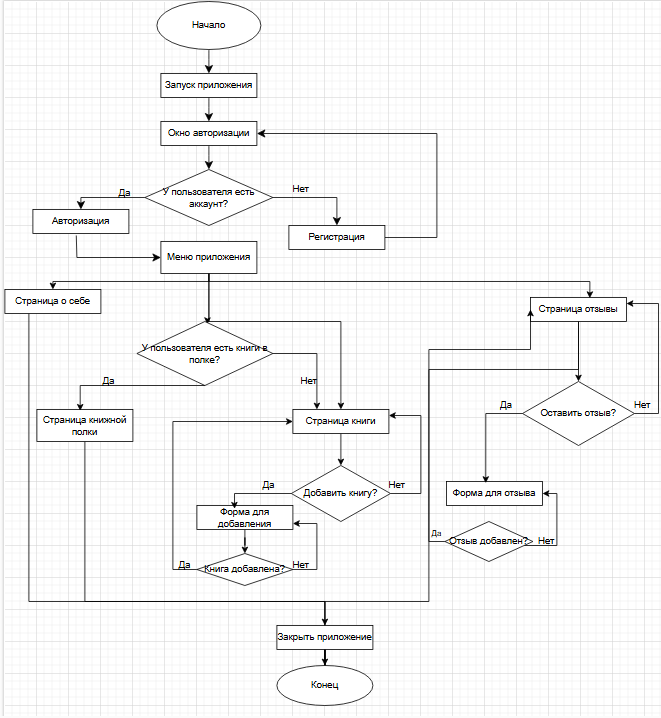


Рис. 2 «Блок-схема Бизнес-процесса»

*4. Основные функции и возможности приложения:*

**Регистрация и авторизация пользователей:**

Регистрация пользователей, пользователь вводит ФИО, почту, пароль;

Авторизация через email и пароль.

**Редактирование информации о себе:**

Пользователь может изменять информацию о себе: изменить ФИО и почту;

Загрузка фотографии профиля.

**Работа с личной полкой пользователя:**

Пользователь может добавлять книги и указывать их статус «Прочитал», «Читаю», «Хочу прочитать»;

Пользователь может сортировать книги по их статусу;

Пользователь может удалять книги с полки.

**Работа с книгами:**

Просмотр книг;

Добавляет книги к себе на полку;

Добавление новой книги.

**Добавление и удаление друзей:**

Пользователь может с помощью поиска искать других и отправлять запрос на дружбу;

Пользователь может добавлять друзей, которые отправили ему заявку;

Пользователь может удалять друзей.

**Работа с отзывами:**

Пользователь может просматривать отзывы других;

Пользователь может оставлять отзыв на книгу и редактировать его.

**Выход из аккаунта:**

Пользователь может выйти из аккаунта с помощью кнопки.

*5. Описание нефункциональных требований*

**Производительность:**

Приложение должно быстро реагировать и обрабатывать запросы

**Безопасность:**

Все данные пользователей должны быть зашифрованы;

Аутентификация и авторизация пользователей;

Хранение паролей.

**Масштабируемость:**

Система должна поддерживать масштабирование для работы с большими объемами данных и увеличением числа пользователей.

*6. Анализ рисков*

Программа должна создавать резервные копии;

При высокой активности может привести к снижению производительности.

**Ограничения:**

Пользователь должен обладать компьютером с Windows 10 и выше.

Пользователь должен обладать компьютером с 4 или более гигабайтами оперативной памяти

**Тестирование функциональности:**

*Тест 1:* Проверка успешной авторизации нового пользователя.

Входные данные: email, пароль

Ожидаемый результат: Авторизация успешна.

*Тест 2:* Проверка добавления книги

Входные данные: Название книги, автор, жанр, год издания, количество страниц, описание книги, фотография книги

Ожидаемый результат: Книга успешно добавлена.

**Тестирование безопасности**

*Тест 1:* Проверка защиты пароля при регистрации.

Входные данные: email, пароль.

Ожидаемый результат: пароль должен быть зашифрован перед сохранением в базе данных.

**Тестирование производительности**

*Тест 1*: Проверка времени отклика при добавлении книги с большим количеством информации.

Входные данные: книга с большим объемом данных.

Ожидаемый результат: время отклика не должно превышать 3 секунд.

# 2. Проектирование и создание базы данных

*Цель:* Разработка структуры базы данных для хранения данных о пользователях, книгах, личных полках пользователя, отзывов, друзей

*Задание:* На основании описания предметной области, созданной ранее, разработать базу данных, выполнив следующие шаги:

СУБД: Microsoft SQL Server

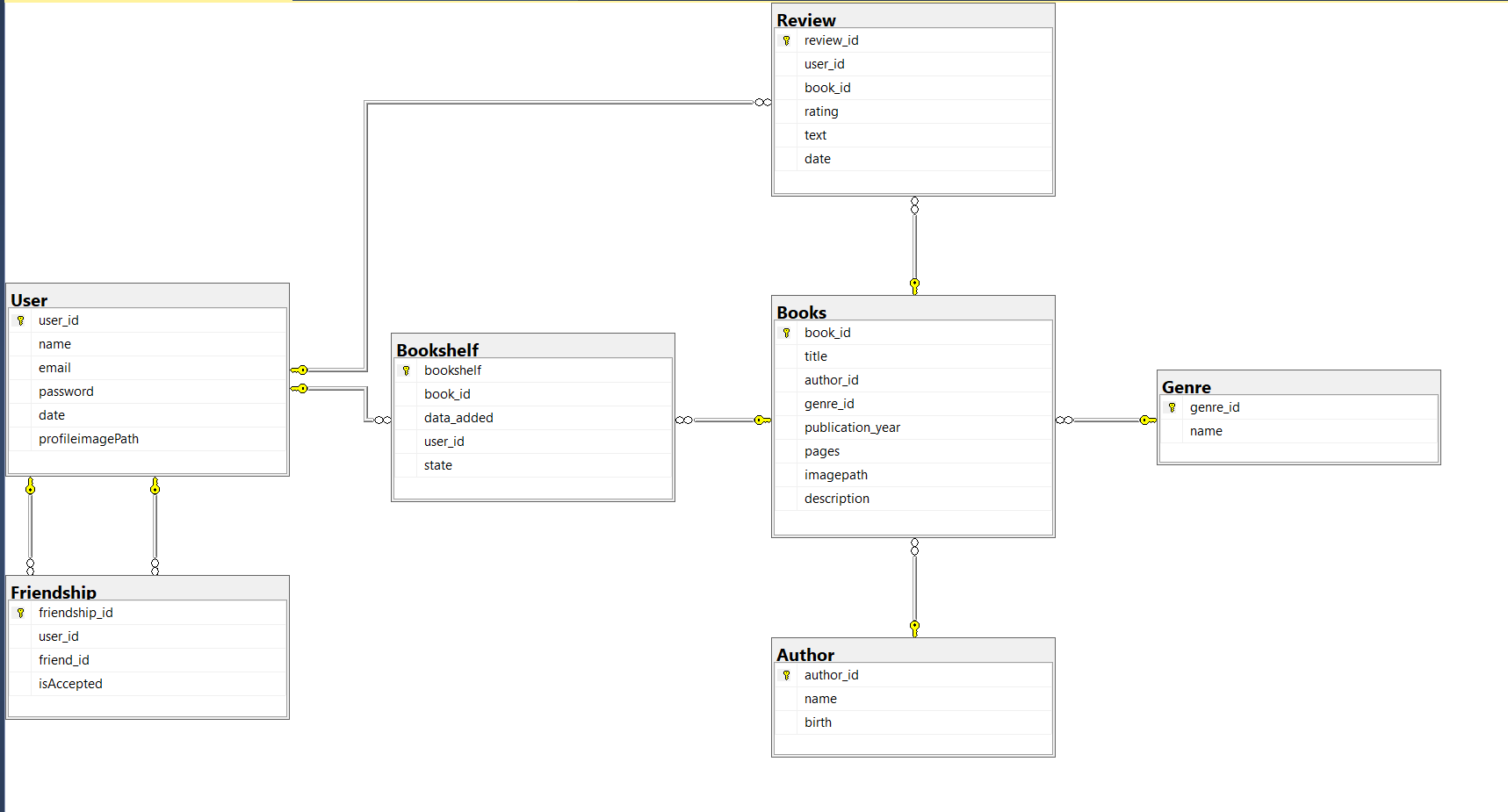


Рис. 3 «ER-диаграмма базы данных»

## 2.1 Таблицы с данными



Рис. 4 «Данные авторов»

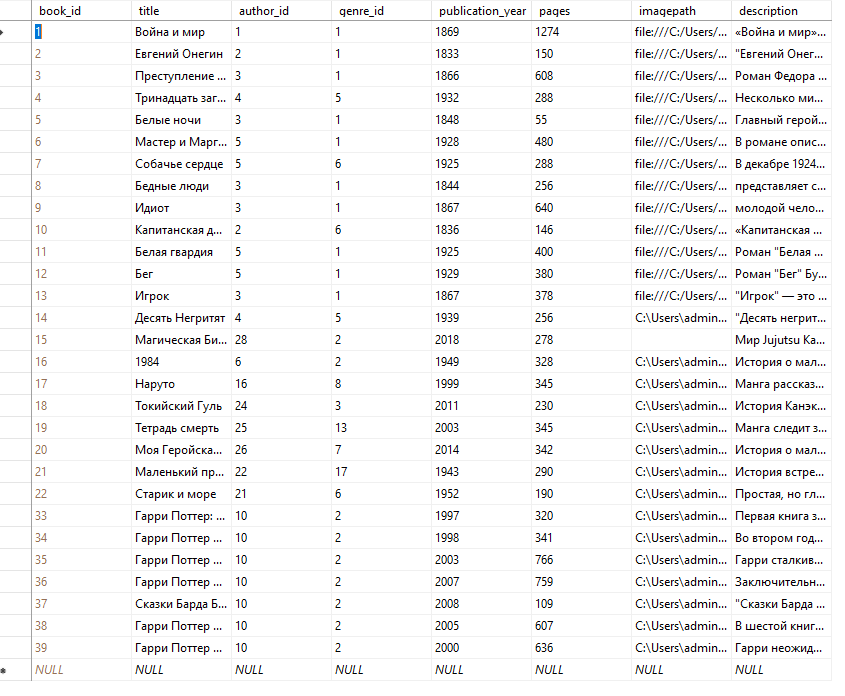


Рис. 5 «Данные таблицы книг»



Рис. 6 «Данные таблицы пользователей»

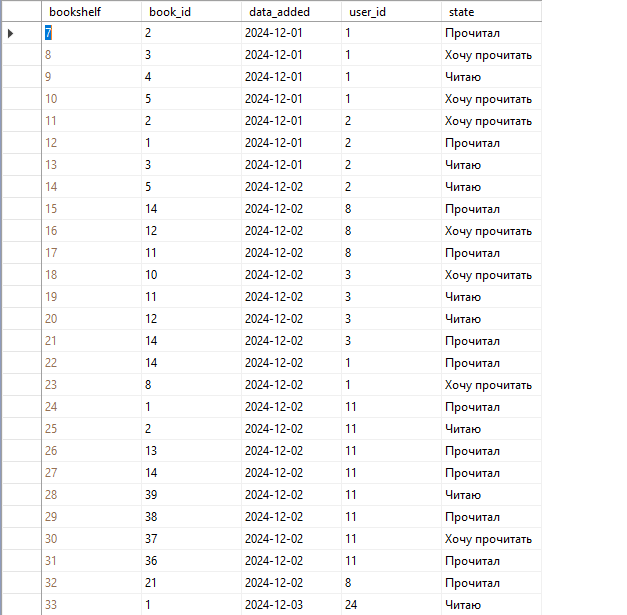


Рис. 7 «Данные таблицы полки»

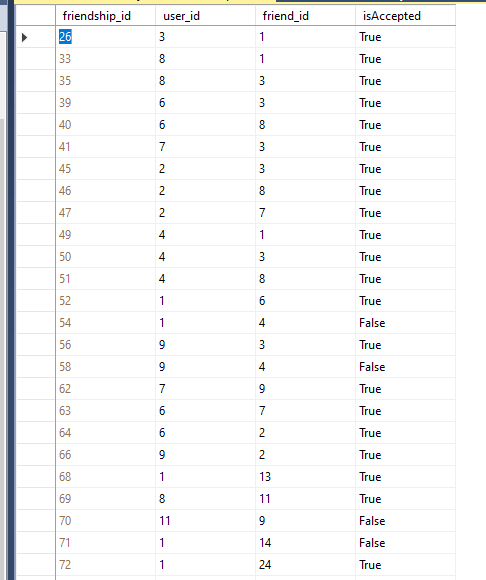


Рис. 8 «Данные таблицы друзей»

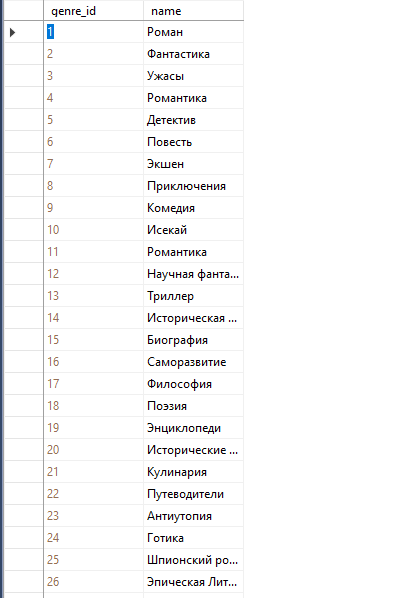


Рис. 9 «Данные таблицы жанров»

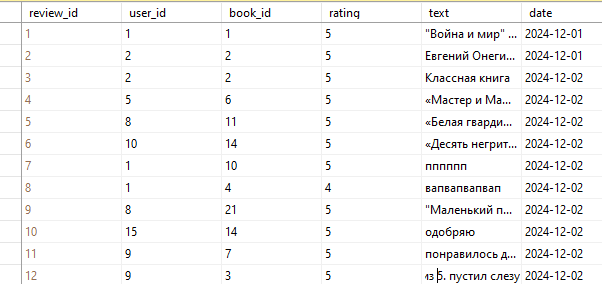


Рис. 10 «Данные таблицы Отзывы»

# 3. Разработка приложения

Для разработки приложения использовался язык программирования С#

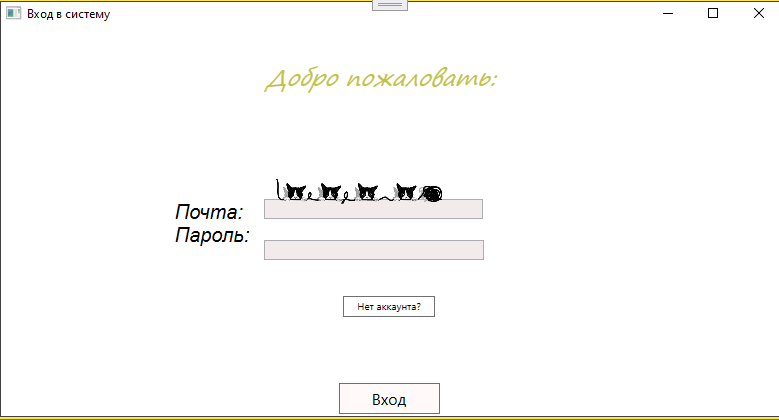


Рис. 11 «Окно авторизации»

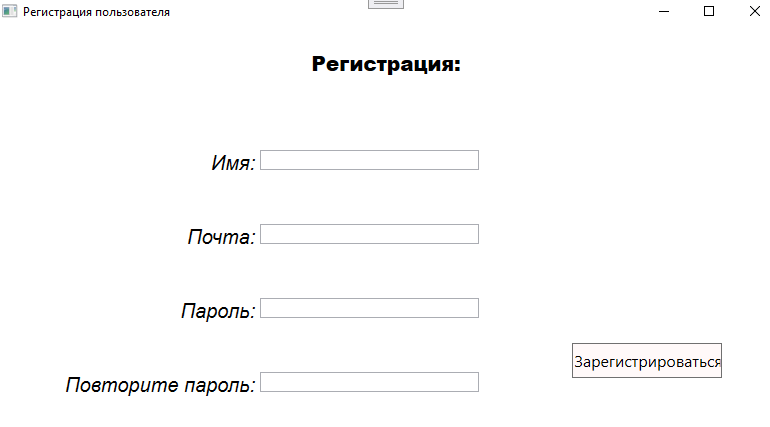


Рис. 12 «Окно регистрации пользователя»

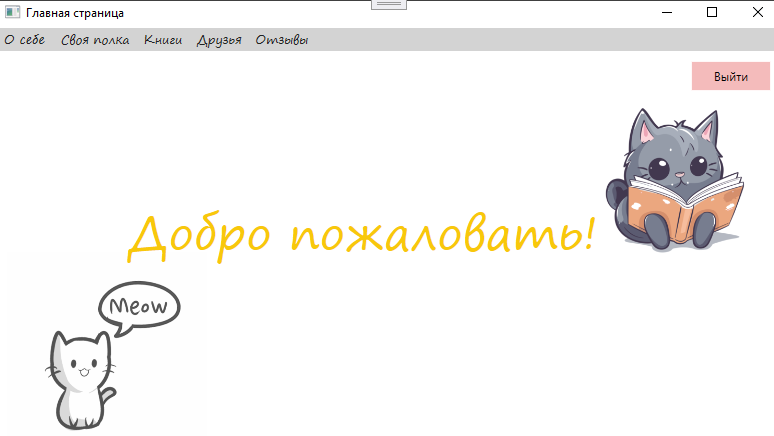


Рис. 13 «Главная страница»

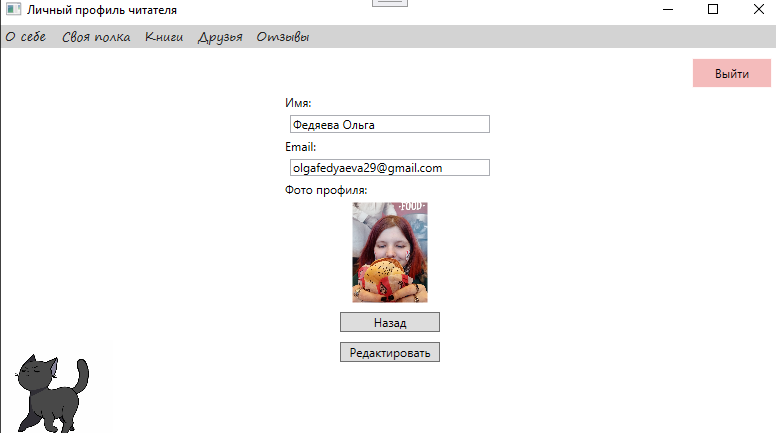


Рис. 14 «Страница о себе»

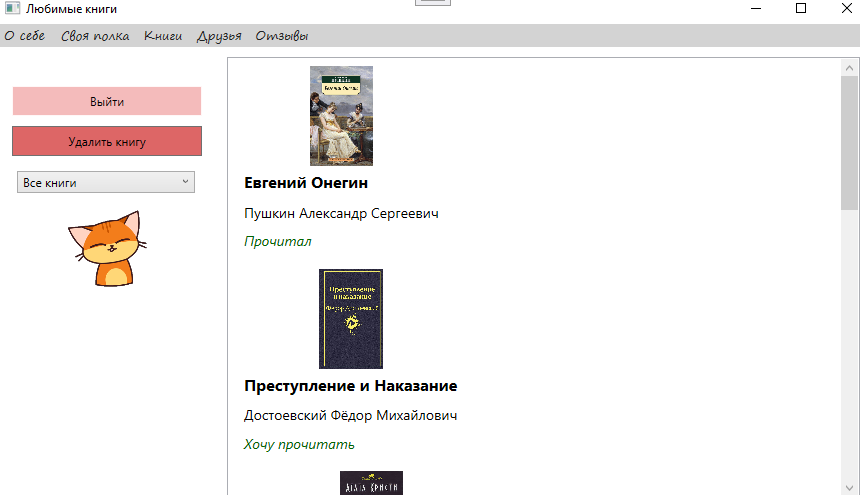


Рис. 15 «Страница своя полка»

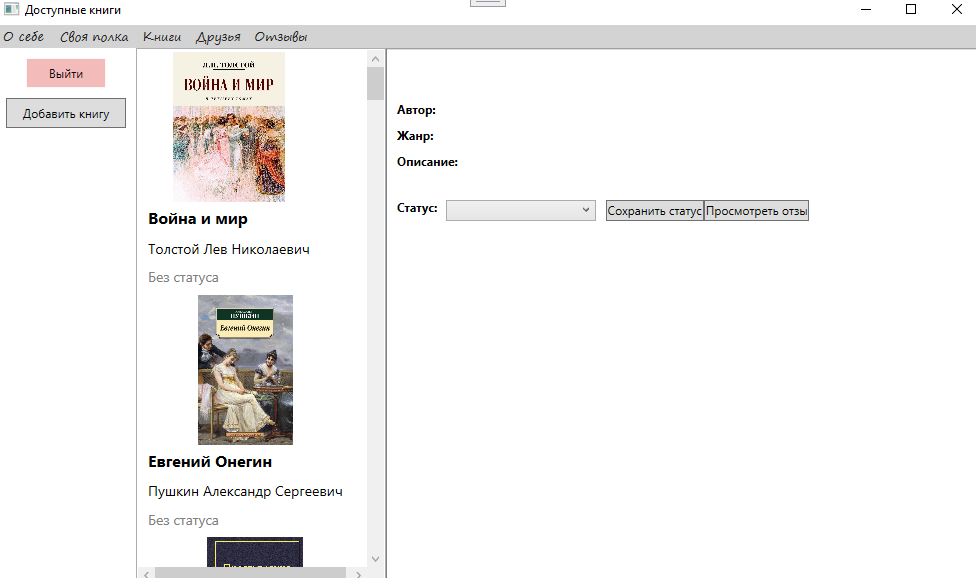


Рис. 16 «Страница книги»

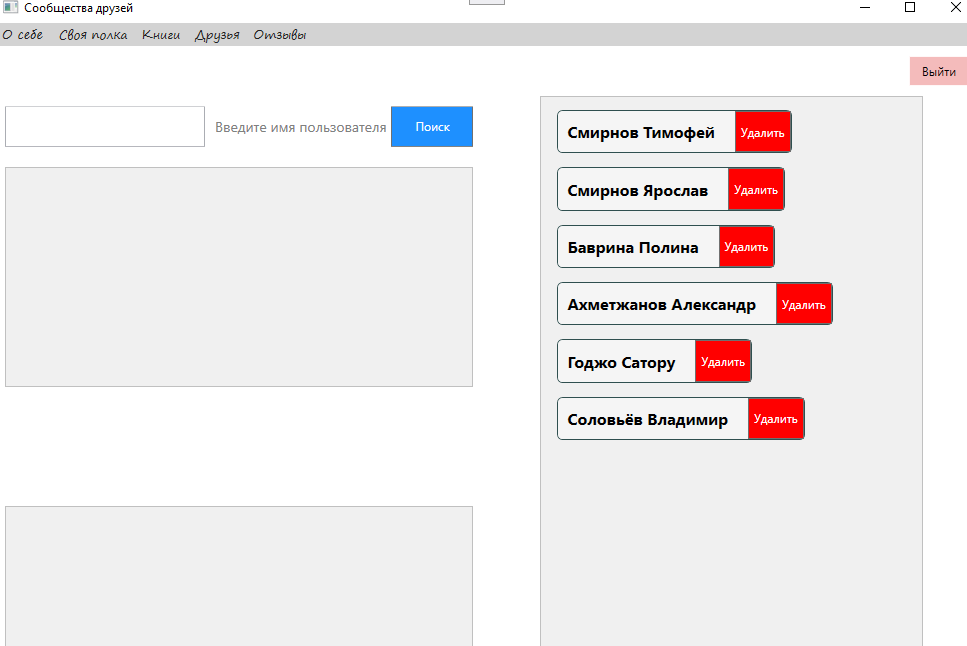


Рис. 17 «Страница друзья»

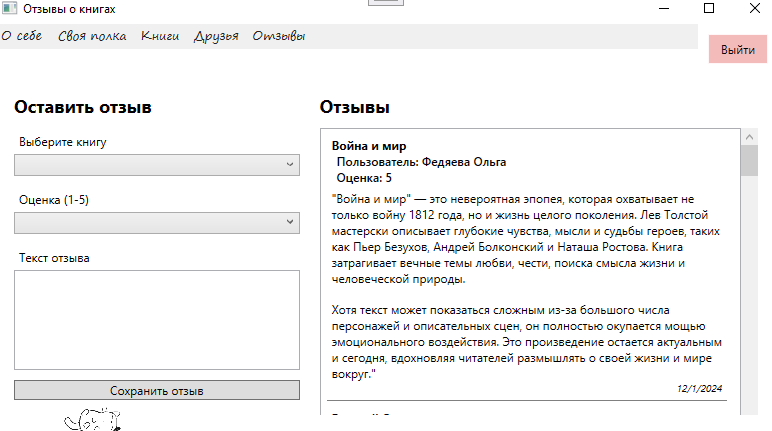
**

Рис. 18 «Страница отзывы»

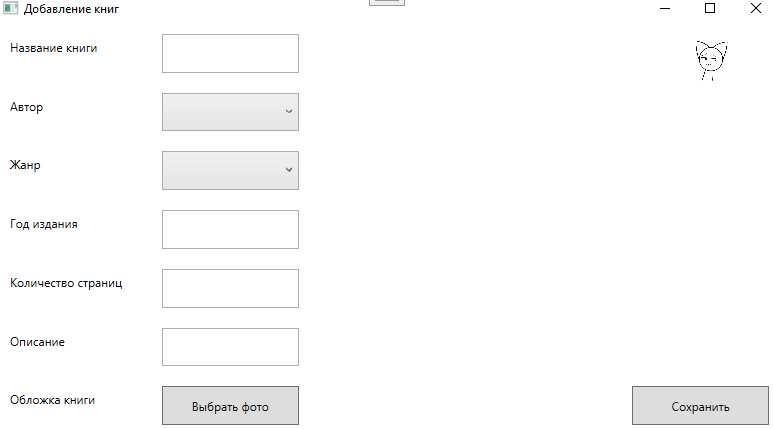


Рис. 19 «Страница добавления книги»

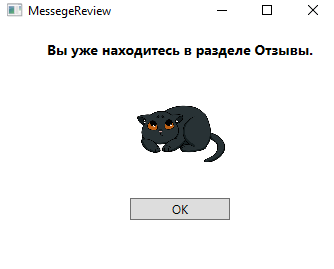


Рис. 20 «Сообщение при повторном нажатии на «Отзывы»»

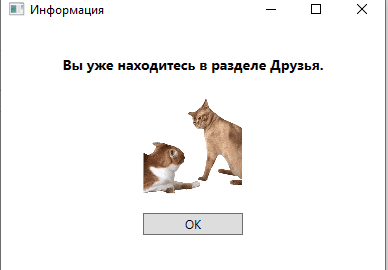


Рис. 21 «Сообщение при повторном нажатии на «Друзья»»

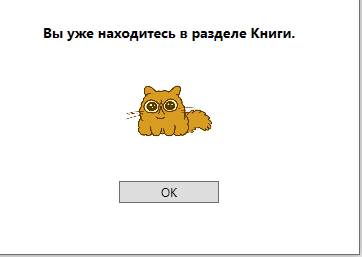


Рис. 22 «Сообщение при повторном нажатии на «Книги»»

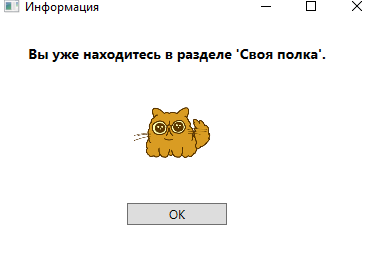


Рис. 23 «Сообщение при повторном нажатии на «Своя полка»»

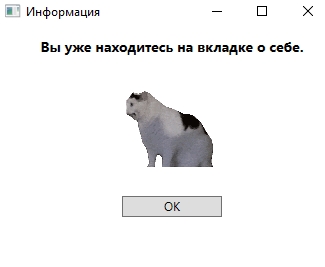


Рис. 24 «Сообщение при повторном нажатии на «О себе»»

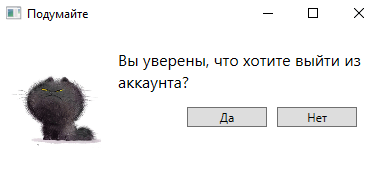


Рис. 25 «Сообщение при нажатии на кнопку Выйти»

# 4. Разработка программных модулей

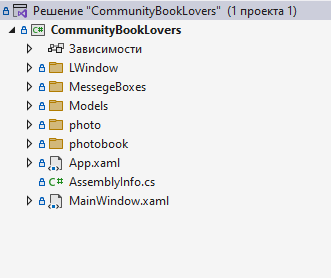


Рис. 26 «Структура проекта»



Рис. 27 «Код для интерфейса авторизации»



Рис. 28 «Логика для окна авторизации»



Рис. 29 «Код для интерфейса окна о себе»

*Логика для окна о себе:*

using CommunityBookLovers.LWindow;

using Microsoft.EntityFrameworkCore;

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Security;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using CommunityBookLovers.MessegeBoxes;

namespace CommunityBookLovers

{

/// <summary>

/// Логика взаимодействия для AboutMeWindow.xaml

/// </summary>

public partial class AboutMeWindow : Window

{

private User currentUser;

private bool isEditing = false;

private LibraryContext \_context;

public AboutMeWindow(User user)

{

InitializeComponent();

currentUser = user;

DisplayUserInfo();

}

private void DisplayUserInfo()

{

if (!isEditing)

{

txtName.Text = currentUser.Name;

txtEmail.Text = currentUser.Email;

if (!string.IsNullOrEmpty(currentUser.ProfileImagePath))

{

try

{

Uri imageUri = new Uri(currentUser.ProfileImagePath);

if (File.Exists(currentUser.ProfileImagePath))

{

BitmapImage bitmap = new BitmapImage(imageUri);

ProfileImage.Source = bitmap;

}

else

{

MessageBox.Show("Файл изображения не найден.");

}

}

catch (Exception ex) when (

ex is ArgumentException ||

ex is NotSupportedException ||

ex is SecurityException ||

ex is IOException)

{

MessageBox.Show($"Произошла ошибка при загрузке изображения: {ex.Message}");

}

}

btnEdit.Visibility = Visibility.Visible;

txtName.IsReadOnly = true;

txtEmail.IsReadOnly = true;

btnSave.Visibility = Visibility.Collapsed;

btnUploadImage.Visibility = Visibility.Collapsed;

}

else

{

txtName.IsReadOnly = false;

txtEmail.IsReadOnly = false;

btnSave.Visibility = Visibility.Visible;

btnUploadImage.Visibility = Visibility.Visible;

btnEdit.Visibility = Visibility.Collapsed;

}

}

private void btnEdit\_Click(object sender, RoutedEventArgs e)

{

isEditing = true;

DisplayUserInfo();

}

private void btnSave\_Click(object sender, RoutedEventArgs e)

{

currentUser.Name = txtName.Text;

currentUser.Email = txtEmail.Text;

SaveUserData();

isEditing = false;

DisplayUserInfo();

}

private void btnUploadImage\_Click(object sender, RoutedEventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog

{

Filter = "Image files|\*.jpg;\*.jpeg;\*.png;\*.bmp;\*.gif"

};

if (openFileDialog.ShowDialog() == true)

{

string imagePath = openFileDialog.FileName;

currentUser.ProfileImagePath = imagePath;

Uri imageUri = new Uri(imagePath);

BitmapImage bitmap = new BitmapImage(imageUri);

ProfileImage.Source = bitmap;

}

}

private void SaveUserData()

{

using (LibraryContext db = new LibraryContext())

{

db.Users.Update(currentUser);

db.SaveChanges();

}

}

private void btnBack\_Click(object sender, RoutedEventArgs e)

{

MainCommunityBook mainCommunityBook = new MainCommunityBook(currentUser);

mainCommunityBook.Show();

this.Close();

}

private void LogoutButton\_Click(object sender, RoutedEventArgs e)

{

var customMessageBox = new CustomWin();

customMessageBox.ShowDialog();

if (customMessageBox.Result)

{

foreach (var window in Application.Current.Windows)

{

if (window is AboutMeWindow || window is CustomWin)

{

continue;

}

(window as Window)?.Close();

}

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

}

private void MenuItem\_AboutMe\_Click(object sender, RoutedEventArgs e)

{

MessegeAboutme messegeAboutme = new MessegeAboutme();

messegeAboutme.Show();

}

private void MenuItem\_MyShelf\_Click(object sender, RoutedEventArgs e)

{

BookshelfWindow bookshelfWindow = new BookshelfWindow(currentUser);

bookshelfWindow.Show();

this.Close();

}

private void MenuItem\_Books\_Click(object sender, RoutedEventArgs e)

{

BookWindow bookWindow = new BookWindow(currentUser);

bookWindow.Show();

this.Close();

}

private void MenuItem\_Friends\_Click(object sender, RoutedEventArgs e)

{

var friendsWindow = new CommunityBook(currentUser);

friendsWindow.Show();

this.Close();

}

private void MenuItem\_Review\_Click(object sender, RoutedEventArgs e)

{

ReviewWindow reviewWindow = new ReviewWindow(currentUser, \_context);

reviewWindow.Show();

this.Close();

}

}

}



Рис. 30 «Код для интерфейса Добавления книг»

*Логика для окна добавления книг*

using Microsoft.Win32;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace CommunityBookLovers

{

/// <summary>

/// Логика взаимодействия для AddBookWindow.xaml

/// </summary>

public partial class AddBookWindow : Window

{

private readonly LibraryContext \_context;

public ObservableCollection<Author> Authors { get; set; }

public ObservableCollection<Genre> Genres { get; set; }

private string \_imagePath;

private User currentUser;

public AddBookWindow(LibraryContext context)

{

InitializeComponent();

DataContext = this;

\_context = context ?? throw new ArgumentNullException(nameof(context), "");

LoadAuthors();

LoadGenres();

}

private void LoadAuthors()

{

try

{

Authors = new ObservableCollection<Author>(\_context.Authors.ToList());

}

catch (Exception ex)

{

MessageBox.Show($"Не удалось загрузить авторов книги: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

AuthorComboBox.IsEnabled = false;

}

}

private void LoadGenres()

{

try

{

Genres = new ObservableCollection<Genre>(\_context.Genres.ToList());

}

catch (Exception ex)

{

MessageBox.Show($"Не удалось загрузить жанры: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

GenreComboBox.IsEnabled = false;

}

}

private void SelectImageButton\_Click(object sender, RoutedEventArgs e)

{

var openFileDialog = new OpenFileDialog

{

Filter = "Image Files|\*.jpg;\*.jpeg;\*.png;\*.bmp;\*.gif"

};

if (openFileDialog.ShowDialog() == true)

{

\_imagePath = openFileDialog.FileName;

var bitmap = new BitmapImage(new Uri(\_imagePath));

CoverImage.Source = bitmap;

}

}

private void SaveButton\_Click(object sender, RoutedEventArgs e)

{

if (!ValidateInput())

return;

try

{

var book = new Book

{

Title = TitleTextBox.Text,

AuthorId = (int?)AuthorComboBox.SelectedValue,

GenreId = (int?)GenreComboBox.SelectedValue,

PublicationYear = int.Parse(PublicationYearTextBox.Text),

Pages = int.Parse(PagesTextBox.Text),

Description = DescriptionTextBox.Text,

ImagePath = \_imagePath

};

\_context.Books.Add(book);

\_context.SaveChanges();

MessageBox.Show("Книга успешно сохранена!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

var bookWindow = new BookWindow(currentUser);

bookWindow.Show();

this.Close();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при сохранении книги: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private bool ValidateInput()

{

if (string.IsNullOrWhiteSpace(TitleTextBox.Text))

{

MessageBox.Show("Пожалуйста, введите название книги.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return false;

}

if (AuthorComboBox.SelectedItem == null)

{

MessageBox.Show("Пожалуйста, выберите автора книги.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return false;

}

if (GenreComboBox.SelectedItem == null)

{

MessageBox.Show("Пожалуйста, выберите жанр книги.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return false;

}

if (!int.TryParse(PublicationYearTextBox.Text, out int publicationYear) || publicationYear <= 0)

{

MessageBox.Show("Введите корректный год издания книги.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return false;

}

if (!int.TryParse(PagesTextBox.Text, out int pages) || pages <= 0)

{

MessageBox.Show("Введите корректное количество страниц в книге.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return false;

}

if (string.IsNullOrEmpty(\_imagePath))

{

MessageBox.Show("Пожалуйста, выберите обложку книги.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

return false;

}

return true;

}

}

}



Рис. 31 «Код для интерфейса своя полка»

*Логика окна своя полка:*

using CommunityBookLovers.LWindow;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Security.Policy;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace CommunityBookLovers

{

/// <summary>

/// Логика взаимодействия для BookshelfWindow.xaml

/// </summary>

public partial class BookshelfWindow : Window

{

private readonly User currentUser;

private readonly LibraryContext \_context;

private ObservableCollection<Bookshelf> bookshelf;

public BookshelfWindow(User user)

{

InitializeComponent();

currentUser = user;

\_context = new LibraryContext();

LoadBookshelf();

}

private void LoadBookshelf()

{

try

{

var bookshelfData = \_context.Bookshelves

.Include(bs => bs.Book)

.ThenInclude(b => b.Author)

.Where(bs => bs.UserId == currentUser.UserId)

.ToList();

if (bookshelfData == null || bookshelfData.Count == 0)

{

MessageBox.Show("На вашей полке пока нет книг.", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

return; // если книг нет, выходим

}

bookshelf = new ObservableCollection<Bookshelf>(bookshelfData);

FilterBooksByStatus("Все книги");

lstBookshelf.ItemsSource = bookshelf;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке полки: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void StatusSortComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

ComboBox comboBox = (ComboBox)sender;

string selectedStatus = ((ComboBoxItem)comboBox.SelectedItem).Content.ToString();

FilterBooksByStatus(selectedStatus);

}

private void FilterBooksByStatus(string status)

{

if (bookshelf == null)

{

MessageBox.Show("Полка не была загружена или пустая.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

ObservableCollection<Bookshelf> filteredBookshelf;

switch (status)

{

case "Все книги":

filteredBookshelf = new ObservableCollection<Bookshelf>(bookshelf);

break;

case "Прочитал":

filteredBookshelf = new ObservableCollection<Bookshelf>(bookshelf.Where(b => b.State == "Прочитал"));

break;

case "Читаю":

filteredBookshelf = new ObservableCollection<Bookshelf>(bookshelf.Where(b => b.State == "Читаю"));

break;

case "Хочу прочитать":

filteredBookshelf = new ObservableCollection<Bookshelf>(bookshelf.Where(b => b.State == "Хочу прочитать"));

break;

default:

filteredBookshelf = new ObservableCollection<Bookshelf>(bookshelf);

break;

}

lstBookshelf.ItemsSource = filteredBookshelf;

}

private void LstBookshelf\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (lstBookshelf.SelectedItem is Bookshelf selectedEntry)

{

MessageBox.Show($"Вы выбрали книгу: {selectedEntry.Book.Title}\nСтатус: {selectedEntry.State}",

"Информация о книге",

MessageBoxButton.OK,

MessageBoxImage.Information);

}

}

private void MenuItem\_AboutMe\_Click(object sender, RoutedEventArgs e)

{

AboutMeWindow aboutMeWindow = new AboutMeWindow(currentUser);

aboutMeWindow.Show();

Close();

}

private void MenuItem\_MyShelf\_Click(object sender, RoutedEventArgs e)

{

BookshelfMessageBox bookshelfMessageBox = new BookshelfMessageBox();

bookshelfMessageBox.Show();

}

private void MenuItem\_Books\_Click(object sender, RoutedEventArgs e)

{

BookWindow bookWindow = new BookWindow(currentUser);

bookWindow.Show();

Close();

}

private void MenuItem\_Friends\_Click(object sender, RoutedEventArgs e)

{

var friendsWindow = new CommunityBook(currentUser);

friendsWindow.Show();

this.Close();

}

private void LogoutButton\_Click(object sender, RoutedEventArgs e)

{

var customMessageBox = new CustomWin();

customMessageBox.ShowDialog();

if (customMessageBox.Result)

{

foreach (var window in Application.Current.Windows)

{

if (window is BookshelfWindow|| window is CustomWin)

{

continue;

}

(window as Window)?.Close();

}

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

}

private void DeleteBookButton\_Click(object sender, RoutedEventArgs e)

{

if (lstBookshelf.SelectedItem is Bookshelf selectedBookshelf)

{

try

{

\_context.Bookshelves.Remove(selectedBookshelf);

\_context.SaveChanges();

LoadBookshelf();

MessageBox.Show("Книга успешно удалена с полки!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при удалении книги: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Пожалуйста, выберите книгу для удаления.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

private void MenuItem\_Review\_Click(object sender, RoutedEventArgs e)

{

ReviewWindow reviewWindow = new ReviewWindow(currentUser, new LibraryContext());

reviewWindow.Show();

this.Close();

}

}

}

*Код для интерфейса окна Книги:*

<Window x:Class="CommunityBookLovers.BookWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:CommunityBookLovers"

xmlns:wpfAnimatedGif="http://wpfanimatedgif.codeplex.com"

mc:Ignorable="d"

Title="Доступные книги" MaxHeight="600" MaxWidth="1000" Height="600" Width="1000">

<DockPanel>

<Menu DockPanel.Dock="Top" Background="LightGray">

<MenuItem Header="О себе" FontFamily="Segoe Print" Click="MenuItem\_AboutMe\_Click" Width="57" />

<MenuItem Header="Своя полка" FontFamily="Segoe Print" Click="MenuItem\_MyShelf\_Click" />

<MenuItem Header="Книги" FontFamily="Segoe Print" Click="MenuItem\_Books\_Click" />

<MenuItem Header="Друзья" FontFamily="Segoe Print" Click="MenuItem\_Friends\_Click" />

<MenuItem Header="Отзывы" FontFamily="Segoe Print" Click="MenuItem\_Review\_Click" />

</Menu>

<StackPanel Orientation="Vertical" HorizontalAlignment="Right" VerticalAlignment="Top" Margin="10">

<Button Content="Выйти" Width="80" Height="30" Click="LogoutButton\_Click" BorderBrush="#FFFFF7F7" Background="#FFF4BBBB"/>

<Button Content="Добавить книгу" Width="120" Height="30" Margin="0,10,0,0" Click="AddBookButton\_Click" />

</StackPanel>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="250"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<ListBox Name="lstBooks" Grid.Column="0" SelectionChanged="LstBooks\_SelectionChanged">

<ListBox.ItemTemplate>

<DataTemplate>

<StackPanel Orientation="Vertical">

<Image Source="{Binding ImagePath}" MaxHeight="150" MaxWidth="150" Stretch="Uniform"/>

<TextBlock Text="{Binding Title}" FontWeight="Bold" FontSize="16" Margin="5"/>

<TextBlock Text="{Binding Author.Name}" FontSize="14" Margin="5"/>

<TextBlock Text="{Binding State, FallbackValue='Без статуса'}" FontSize="14" Foreground="Gray" Margin="5"/>

</StackPanel>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

<Border Grid.Column="1" BorderThickness="1" BorderBrush="Gray" Padding="10">

<StackPanel>

<TextBlock x:Name="txtBookTitle" FontSize="24" FontWeight="Bold" Margin="0,0,0,10"/>

<StackPanel Orientation="Horizontal" Margin="0,0,0,10">

<TextBlock Text="Автор: " FontWeight="Bold"/>

<TextBlock x:Name="txtBookAuthor"/>

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="0,0,0,10">

<TextBlock Text="Жанр: " FontWeight="Bold"/>

<TextBlock x:Name="txtBookGenre"/>

</StackPanel>

<StackPanel>

<TextBlock Text="Описание: " FontWeight="Bold" />

<TextBlock x:Name="txtBookDescription" TextWrapping="Wrap" Margin="5,0,0,0" FontFamily="Blackadder ITC" FontStyle="Italic" FontSize="16" />

</StackPanel>

<StackPanel Orientation="Horizontal" Margin="0,10,0,0">

<TextBlock Text="Статус: " FontWeight="Bold" Margin="0,0,5,0"/>

<ComboBox x:Name="cbBookStatus" Width="150" Margin="0,0,10,0">

<ComboBoxItem Content="Прочитал" />

<ComboBoxItem Content="Читаю" />

<ComboBoxItem Content="Хочу прочитать" />

</ComboBox>

<Button Content="Сохранить статус" Click="SaveStatusButton\_Click" />

<Button Content="Просмотреть отзыв на книгу" Click="ViewingReviewsButton\_Click" Width="105" />

</StackPanel>

</StackPanel>

</Border>

</Grid>

</DockPanel>

</Window>

*Логика для окна книги:*

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using CommunityBookLovers.LWindow;

using Microsoft.EntityFrameworkCore;

using CommunityBookLovers.MessegeBoxes;

namespace CommunityBookLovers

{

/// <summary>

/// Логика взаимодействия для BookWindow.xaml

/// </summary>

public partial class BookWindow : Window

{

private readonly User currentUser;

private readonly LibraryContext \_context;

private LibraryContext? context;

public BookWindow(User user)

{

InitializeComponent();

currentUser = user;

\_context = new LibraryContext();

LoadBooks();

}

private void LoadBooks()

{

try

{

var books = \_context.Books

.Include(b => b.Author)

.Include(b => b.Genre)

.ToList();

lstBooks.ItemsSource = books;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке книг: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void LstBooks\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (lstBooks.SelectedItem is Book selectedBook)

{

DisplayBookDetails(selectedBook);

}

}

private void DisplayBookDetails(Book book)

{

if (book == null) return;

txtBookTitle.Text = book.Title;

txtBookAuthor.Text = book.Author?.Name ?? "Автор не указан";

txtBookGenre.Text = book.Genre?.Name ?? "Жанр не указан";

txtBookDescription.Text = book.Description ?? "Описание отсутствует";

var bookshelfEntry = \_context.Bookshelves

.FirstOrDefault(bs => bs.BookId == book.BookId && bs.UserId == currentUser.UserId);

if (bookshelfEntry != null)

{

cbBookStatus.SelectedItem = cbBookStatus.Items

.Cast<ComboBoxItem>()

.FirstOrDefault(item => item.Content.ToString() == bookshelfEntry.State);

}

else

{

cbBookStatus.SelectedIndex = -1;

}

}

private void SaveStatusButton\_Click(object sender, RoutedEventArgs e)

{

if (lstBooks.SelectedItem is Book selectedBook)

{

var selectedStatus = (cbBookStatus.SelectedItem as ComboBoxItem)?.Content.ToString();

if (string.IsNullOrEmpty(selectedStatus))

{

MessageBox.Show("Выберите статус книги!", "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

return;

}

try

{

var bookshelfEntry = \_context.Bookshelves

.FirstOrDefault(bs => bs.BookId == selectedBook.BookId && bs.UserId == currentUser.UserId);

if (bookshelfEntry == null)

{

bookshelfEntry = new Bookshelf

{

BookId = selectedBook.BookId,

UserId = currentUser.UserId,

DataAdded = DateOnly.FromDateTime(DateTime.Now),

State = selectedStatus

};

\_context.Bookshelves.Add(bookshelfEntry);

}

else

{

bookshelfEntry.State = selectedStatus;

}

\_context.SaveChanges();

MessageBox.Show("Статус книги обновлен!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при обновлении статуса книги: {ex.Message}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

else

{

MessageBox.Show("Выберите книгу!", "Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

private void MenuItem\_AboutMe\_Click(object sender, RoutedEventArgs e)

{

AboutMeWindow aboutMeWindow = new AboutMeWindow(currentUser);

aboutMeWindow.Show();

this.Close();

}

private void MenuItem\_MyShelf\_Click(object sender, RoutedEventArgs e)

{

BookshelfWindow bookshelfWindow = new BookshelfWindow(currentUser);

bookshelfWindow.Show();

this.Close();

}

private void MenuItem\_Books\_Click(object sender, RoutedEventArgs e)

{

MessegeBook messegeBook = new MessegeBook();

messegeBook.Show();

}

private void MenuItem\_Friends\_Click(object sender, RoutedEventArgs e)

{

var friendsWindow = new CommunityBook(currentUser);

friendsWindow.Show();

this.Close();

}

private void MenuItem\_Review\_Click(object sender, RoutedEventArgs e)

{

ReviewWindow reviewWindow = new ReviewWindow(currentUser, new LibraryContext());

reviewWindow.Show();

this.Close();

}

private void LogoutButton\_Click(object sender, RoutedEventArgs e)

{

var customMessageBox = new CustomWin();

customMessageBox.ShowDialog();

if (customMessageBox.Result)

{

foreach (var window in Application.Current.Windows)

{

if (window is BookWindow || window is CustomWin)

{

continue;

}

(window as Window)?.Close();

}

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

}

private void AddBookButton\_Click(object sender, RoutedEventArgs e)

{

LibraryContext context = new LibraryContext();

AddBookWindow addBookWindow = new AddBookWindow(context);

addBookWindow.Show();

this.Close();

}

private void ViewingReviewsButton\_Click(object sender, RoutedEventArgs e)

{

}

}

}

*Код для интерфейса Друзья:*

<Window x:Class="CommunityBookLovers.LWindow.CommunityBook"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:CommunityBookLovers.LWindow"

mc:Ignorable="d"

Title="Сообщества друзей" Height="700" Width="1000">

<DockPanel>

<Menu DockPanel.Dock="Top" Background="LightGray">

<MenuItem Header="О себе" FontFamily="Segoe Print" Click="MenuItem\_AboutMe\_Click" Width="57" />

<MenuItem Header="Своя полка" FontFamily="Segoe Print" Click="MenuItem\_MyShelf\_Click" />

<MenuItem Header="Книги" FontFamily="Segoe Print" Click="MenuItem\_Books\_Click" />

<MenuItem Header="Друзья" FontFamily="Segoe Print" Click="MenuItem\_Friends\_Click" />

<MenuItem Header="Отзывы" FontFamily="Segoe Print" Click="MenuItem\_Review\_Click" />

</Menu>

<Button Content="Выйти" Width="60" Height="30" Click="LogoutButton\_Click" BorderBrush="#FFFFF7F7" Background="#FFF4BBBB" HorizontalAlignment="Right" VerticalAlignment="Top" Margin="10" DockPanel.Dock="Top"/>

<Grid Margin="10" DockPanel.Dock="Left" Width="468" Height="608">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="195\*"/>

<ColumnDefinition Width="71\*"/>

<ColumnDefinition Width="176\*"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto" />

<RowDefinition Height="\*" />

</Grid.RowDefinitions>

<ListBox x:Name="FriendshipRequestsList" Grid.Row="1" Margin="0,10,0,327" Background="#f0f0f0" BorderBrush="#c0c0c0" BorderThickness="1" Padding="5" Grid.ColumnSpan="3">

<ListBox.ItemTemplate>

<DataTemplate>

<Border BorderBrush="DarkSlateGray" BorderThickness="1" Margin="5" Background="WhiteSmoke" CornerRadius="5">

<StackPanel>

<TextBlock Text="{Binding User.Name}" Margin="5" FontSize="16" FontWeight="Bold"/>

<Button Content="Принять запрос" Tag="{Binding FriendshipId}" Margin="5" Padding="10" Background="Green" Foreground="White" Click="AcceptRequestButton\_Click" />

</StackPanel>

</Border>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

<StackPanel Grid.Row="0" Orientation="Horizontal" Margin="0,0,0,10" Grid.ColumnSpan="3">

<TextBox x:Name="SearchBox" Width="200" Margin="0,0,10,0" TextChanged="SearchBox\_TextChanged" Padding="10" FontSize="14"/>

<TextBlock x:Name="SearchPlaceholder" Text="Введите имя пользователя" Foreground="Gray" VerticalAlignment="Center" FontSize="14" />

<Button Content="Поиск" Padding="10" Margin="5,0" Background="DodgerBlue" Foreground="White" Click="SearchButton\_Click" Width="82" />

</StackPanel>

<ListBox x:Name="SearchResultsList" Grid.Row="1" Margin="0,349,0,20" Background="#f0f0f0" BorderBrush="#c0c0c0" BorderThickness="1" Padding="5" Grid.ColumnSpan="3">

<ListBox.ItemTemplate>

<DataTemplate>

<Border BorderBrush="DarkSlateGray" BorderThickness="1" Margin="5" Background="WhiteSmoke" CornerRadius="5">

<StackPanel Orientation="Horizontal" Margin="5">

<TextBlock Text="{Binding Name}" VerticalAlignment="Center" FontSize="16"/>

<Button Content="Добавить в друзья" Margin="10,0,0,0" Tag="{Binding UserId}" Padding="10" Background="MediumSeaGreen" Foreground="White" Click="SendFriendRequestButton\_Click" />

</StackPanel>

</Border>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</Grid>

<ListBox x:Name="FriendsList" DockPanel.Dock="Right" Width="383" Margin="10,0,10,10" Background="#f0f0f0" BorderBrush="#c0c0c0" BorderThickness="1" Padding="5">

<ListBox.ItemTemplate>

<DataTemplate>

<Border BorderBrush="DarkSlateGray" BorderThickness="1" Margin="5" Background="WhiteSmoke" CornerRadius="5">

<StackPanel Orientation="Horizontal">

<TextBlock Text="{Binding Name}" Margin="10" FontSize="16" FontWeight="Bold" VerticalAlignment="Center"/>

<Button Content="Удалить" Margin="10,0,0,0" Padding="5" Background="Red" Foreground="White" Tag="{Binding UserId}" Click="RemoveFriendButton\_Click" />

</StackPanel>

</Border>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</DockPanel>

</Window>

*Логика для окна Друзья:*

using CommunityBookLovers.MessegeBoxes;

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace CommunityBookLovers.LWindow

{

/// <summary>

/// Логика взаимодействия для CommunityBook.xaml

/// </summary>

public partial class CommunityBook : Window

{

private User currentUser;

private LibraryContext dbContext;

public CommunityBook(User user)

{

InitializeComponent();

currentUser = user;

dbContext = new LibraryContext();

LoadFriendshipRequests();

LoadFriendsList();

}

private void LoadFriendsList()

{

var friends = dbContext.Friendships

.Where(f =>

(f.UserId == currentUser.UserId || f.FriendId == currentUser.UserId) &&

f.IsAccepted)

.Select(f => f.UserId == currentUser.UserId ? f.Friend : f.User)

.Distinct()

.ToList();

FriendsList.ItemsSource = friends;

}

private void LoadFriendshipRequests()

{

var pendingRequests = dbContext.Friendships

.Where(f => f.FriendId == currentUser.UserId && !f.IsAccepted)

.Include(f => f.User)

.ToList();

FriendshipRequestsList.ItemsSource = pendingRequests;

}

private void AcceptRequestButton\_Click(object sender, RoutedEventArgs e)

{

Button button = (Button)sender;

int friendshipId = (int)button.Tag;

var friendshipRequest = dbContext.Friendships.FirstOrDefault(f => f.FriendshipId == friendshipId);

if (friendshipRequest != null)

{

friendshipRequest.IsAccepted = true;

dbContext.SaveChanges();

LoadFriendshipRequests();

LoadFriendsList();

}

}

private void RemoveFriendButton\_Click(object sender, RoutedEventArgs e)

{

Button button = (Button)sender;

int friendId = (int)button.Tag;

var friendship = dbContext.Friendships.FirstOrDefault(f =>

(f.UserId == currentUser.UserId && f.FriendId == friendId) ||

(f.UserId == friendId && f.FriendId == currentUser.UserId));

if (friendship != null)

{

dbContext.Friendships.Remove(friendship);

dbContext.SaveChanges();

LoadFriendsList();

MessageBox.Show("Друг удалён.", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

private void SearchBox\_TextChanged(object sender, TextChangedEventArgs e)

{

string query = SearchBox.Text.Trim().ToLower();

if (!string.IsNullOrEmpty(query))

{

var results = dbContext.Users

.Where(u => u.Name.ToLower().Contains(query) && u.UserId != currentUser.UserId)

.ToList();

SearchResultsList.ItemsSource = results;

}

}

private void SendFriendRequestButton\_Click(object sender, RoutedEventArgs e)

{

Button button = (Button)sender;

int friendId = (int)button.Tag;

var existingFriendship = dbContext.Friendships

.Any(f =>

(f.UserId == currentUser.UserId && f.FriendId == friendId && f.IsAccepted) ||

(f.UserId == friendId && f.FriendId == currentUser.UserId && f.IsAccepted));

if (existingFriendship)

{

MessageBox.Show("Вы уже друзья с этим пользователем.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Information);

return;

}

var existingRequest = dbContext.Friendships

.Any(f =>

(f.UserId == currentUser.UserId && f.FriendId == friendId && !f.IsAccepted) ||

(f.UserId == friendId && f.FriendId == currentUser.UserId && !f.IsAccepted));

if (existingRequest)

{

MessageBox.Show("Запрос на дружбу уже отправлен или принят.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Information);

return;

}

var newFriendship = new Friendship

{

UserId = currentUser.UserId,

FriendId = friendId,

IsAccepted = false

};

dbContext.Friendships.Add(newFriendship);

dbContext.SaveChanges();

LoadFriendshipRequests();

MessageBox.Show("Заявка на дружбу успешно отправлена!", "Заявка на дружбу", MessageBoxButton.OK, MessageBoxImage.Information);

}

private void MenuItem\_AboutMe\_Click(object sender, RoutedEventArgs e)

{

AboutMeWindow aboutMeWindow = new AboutMeWindow(currentUser);

aboutMeWindow.Show();

this.Close();

}

private void MenuItem\_MyShelf\_Click(object sender, RoutedEventArgs e)

{

BookshelfWindow bookshelfWindow = new BookshelfWindow(currentUser);

bookshelfWindow.Show();

this.Close();

}

private void MenuItem\_Books\_Click(object sender, RoutedEventArgs e)

{

BookWindow bookWindow = new BookWindow(currentUser);

bookWindow.Show();

this.Close();

}

private void MenuItem\_Friends\_Click(object sender, RoutedEventArgs e)

{

MessegeBoxFriend messegeBoxFriend = new MessegeBoxFriend();

messegeBoxFriend.Show();

}

private void MenuItem\_Review\_Click(object sender, RoutedEventArgs e)

{

ReviewWindow reviewWindow = new ReviewWindow(currentUser, new LibraryContext());

reviewWindow.Show();

this.Close();

}

private void LogoutButton\_Click(object sender, RoutedEventArgs e)

{

var customMessageBox = new CustomWin();

customMessageBox.ShowDialog();

if (customMessageBox.Result)

{

foreach (var window in Application.Current.Windows)

{

if (window is CommunityBook || window is CustomWin)

{

continue;

}

(window as Window)?.Close();

}

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

}

private void SearchButton\_Click(object sender, RoutedEventArgs e)

{

string query = SearchBox.Text.Trim().ToLower();

if (!string.IsNullOrEmpty(query))

{

var results = dbContext.Users

.Where(u => u.Name.ToLower().Contains(query) && u.UserId != currentUser.UserId)

.ToList();

SearchResultsList.ItemsSource = results;

if (results.Count == 0)

{

MessageBox.Show("Пользователи не найдены.", "Результаты поиска", MessageBoxButton.OK, MessageBoxImage.Information);

}

}

else

{

MessageBox.Show("Введите текст для поиска.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

}

}

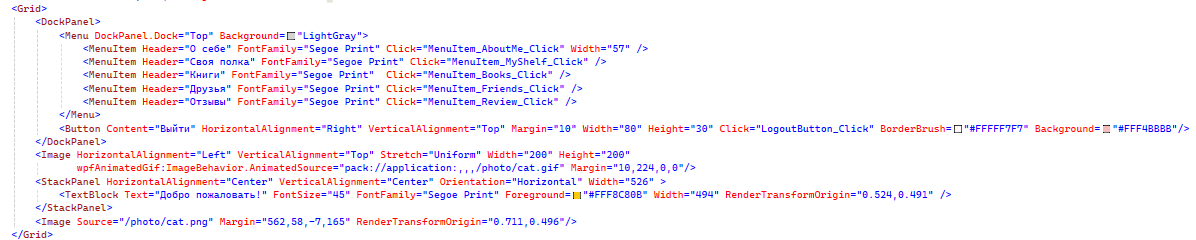


Рис. 32 «Код для интерфейса Главной страницы»

*Логика для главной страницы:*

using CommunityBookLovers.LWindow;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace CommunityBookLovers

{

/// <summary>

/// Логика взаимодействия для MainCommunityBook.xaml

/// </summary>

public partial class MainCommunityBook : Window

{

private User currentUser;

public MainCommunityBook(User user)

{

InitializeComponent();

currentUser = user;

}

private void MenuItem\_AboutMe\_Click(object sender, RoutedEventArgs e)

{

AboutMeWindow aboutMeWindow = new AboutMeWindow(currentUser);

aboutMeWindow.Show();

this.Close();

}

private void MenuItem\_MyShelf\_Click(object sender, RoutedEventArgs e)

{

BookshelfWindow bookshelfWindow = new BookshelfWindow(currentUser);

bookshelfWindow.Show();

this.Close();

}

private void MenuItem\_Books\_Click(object sender, RoutedEventArgs e)

{

BookWindow bookWindow = new BookWindow(currentUser);

bookWindow.Show();

this.Close();

}

private void MenuItem\_Friends\_Click(object sender, RoutedEventArgs e)

{

var friendsWindow = new CommunityBook(currentUser);

friendsWindow.Show();

this.Close();

}

private void LogoutButton\_Click(object sender, RoutedEventArgs e)

{

var customMessageBox = new CustomWin();

customMessageBox.ShowDialog();

if (customMessageBox.Result)

{

foreach (var window in Application.Current.Windows)

{

if (window is MainCommunityBook || window is CustomWin)

{

continue;

}

(window as Window)?.Close();

}

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

}

private void MenuItem\_Review\_Click(object sender, RoutedEventArgs e)

{

ReviewWindow reviewWindow = new ReviewWindow(currentUser, new LibraryContext());

reviewWindow.Show();

this.Close();

}

}

}

*Код для интерфейса окна Регистрации:*

<Window x:Class="CommunityBookLovers.RegistrationWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:CommunityBookLovers"

mc:Ignorable="d"

Title="Регистрация пользователя" Height="450" Width="800">

<Grid Margin="0,0,0,41">

<Grid.RowDefinitions>

<RowDefinition Height="1\*" />

<RowDefinition Height="1\*" />

<RowDefinition Height="1\*" />

<RowDefinition Height="1\*" />

<RowDefinition Height="1\*" />

</Grid.RowDefinitions>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*" />

<ColumnDefinition Width="1\*" />

<ColumnDefinition Width="1\*" />

</Grid.ColumnDefinitions>

<StackPanel Grid.Row="0" Grid.Column="1" HorizontalAlignment="Center" VerticalAlignment="Center">

<TextBlock Text="Регистрация:" FontSize="20" FontFamily="Arial Black" />

</StackPanel>

<StackPanel Grid.Row="1" Grid.Column="0" HorizontalAlignment="Right" VerticalAlignment="Bottom">

<TextBlock Text="Имя:" FontSize="20" FontFamily="Arial" FontStyle="Italic" />

</StackPanel>

<StackPanel Grid.Row="1" Grid.Column="1" HorizontalAlignment="Left" VerticalAlignment="Bottom">

<TextBox Name="txtName" Width="219" Height="20" Margin="5" />

</StackPanel>

<StackPanel Grid.Row="2" Grid.Column="0" HorizontalAlignment="Right" VerticalAlignment="Bottom">

<TextBlock Text="Почта:" FontSize="20" FontFamily="Arial" FontStyle="Italic" />

</StackPanel>

<StackPanel Grid.Row="2" Grid.Column="1" HorizontalAlignment="Left" VerticalAlignment="Bottom">

<TextBox Name="txtEmail" Width="219" Height="20" Margin="5" />

</StackPanel>

<StackPanel Grid.Row="3" Grid.Column="0" HorizontalAlignment="Right" VerticalAlignment="Bottom">

<TextBlock Text="Пароль:" FontSize="20" FontFamily="Arial" FontStyle="Italic" />

</StackPanel>

<StackPanel Grid.Row="3" Grid.Column="1" HorizontalAlignment="Left" VerticalAlignment="Bottom">

<PasswordBox Name="txtPassword" Width="219" Height="20" Margin="5" />

</StackPanel>

<StackPanel Grid.Row="4" Grid.Column="0" HorizontalAlignment="Right" VerticalAlignment="Bottom">

<TextBlock Text="Повторите пароль:" FontSize="20" FontFamily="Arial" FontStyle="Italic" />

</StackPanel>

<StackPanel Grid.Row="4" Grid.Column="1" HorizontalAlignment="Left" VerticalAlignment="Bottom">

<PasswordBox Name="txtConfirmPassword" Width="219" Height="20" Margin="5" />

</StackPanel>

<StackPanel Grid.Row="5" Grid.Column="2" HorizontalAlignment="Center" VerticalAlignment="Center">

<Button x:Name="btnRegister" Content="Зарегистрироваться" Click="btnRegister\_Click" Width="150" Height="35" FontSize="16" Background="#FFFFF9F9" />

</StackPanel>

</Grid>

</Window>

*Логика для окна Регистрации:*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace CommunityBookLovers

{

/// <summary>

/// Логика взаимодействия для RegistrationWindow.xaml

/// </summary>

public partial class RegistrationWindow : Window

{

public RegistrationWindow()

{

InitializeComponent();

}

private void btnRegister\_Click(object sender, RoutedEventArgs e)

{

string name = txtName.Text;

string email = txtEmail.Text;

string password = txtPassword.Password;

string confirmPassword = txtConfirmPassword.Password;

if (password != confirmPassword)

{

MessageBox.Show("Пароли не совпадают!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

using (LibraryContext db = new LibraryContext())

{

if (db.Users.Any(u => u.Email == email))

{

MessageBox.Show("Пользователь с такой почтой уже существует!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

User newUser = new User

{

Name = name,

Email = email,

Password = password,

Date = DateOnly.FromDateTime(DateTime.Now)

};

db.Users.Add(newUser);

db.SaveChanges();

MessageBox.Show("Регистрация прошла успешно!", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

MainWindow loginWindow = new MainWindow();

loginWindow.Show();

this.Close();

}

}

}

}

*Код интерфейса для окна Отзывы:*

<Window x:Class="CommunityBookLovers.LWindow.ReviewWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:CommunityBookLovers.LWindow"

xmlns:wpfAnimatedGif="http://wpfanimatedgif.codeplex.com"

mc:Ignorable="d"

Title="Отзывы о книгах" Height="450" Width="800">

<DockPanel>

<Grid DockPanel.Dock="Top" Background="White">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*" />

<ColumnDefinition Width="Auto" />

</Grid.ColumnDefinitions>

<Menu Grid.Column="0" Margin="0,0,0,25">

<MenuItem Header="О себе" FontFamily="Segoe Print" Click="MenuItem\_AboutMe\_Click" Width="57" />

<MenuItem Header="Своя полка" FontFamily="Segoe Print" Click="MenuItem\_MyShelf\_Click" />

<MenuItem Header="Книги" FontFamily="Segoe Print" Click="MenuItem\_Books\_Click" />

<MenuItem Header="Друзья" FontFamily="Segoe Print" Click="MenuItem\_Friends\_Click" />

<MenuItem Header="Отзывы" FontFamily="Segoe Print" Click="MenuItem\_Review\_Click" />

</Menu>

<Button Grid.Column="1" Content="Выйти" Width="60" Height="30" Click="LogoutButton\_Click" BorderBrush="#FFFFF7F7" Background="#FFF4BBBB" HorizontalAlignment="Right" VerticalAlignment="Top" Margin="10"/>

</Grid>

<Grid Margin="10">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="2\*" />

<ColumnDefinition Width="3\*" />

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="0" Margin="10">

<TextBlock Text="Оставить отзыв" FontSize="18" FontWeight="Bold" Margin="0,0,0,10" />

<Label Content="Выберите книгу" />

<ComboBox x:Name="BookComboBox" DisplayMemberPath="Title" SelectedValuePath="BookId" />

<Label Content="Оценка (1-5)" Margin="0,10,0,0" />

<ComboBox x:Name="RatingComboBox">

<ComboBoxItem Content="1" />

<ComboBoxItem Content="2" />

<ComboBoxItem Content="3" />

<ComboBoxItem Content="4" />

<ComboBoxItem Content="5" />

</ComboBox>

<Label Content="Текст отзыва" Margin="0,10,0,0" />

<TextBox x:Name="ReviewTextBox" AcceptsReturn="True" Height="100" TextWrapping="Wrap" />

<Button Content="Сохранить отзыв" Margin="0,10,0,0" Click="SaveReviewButton\_Click" />

</StackPanel>

<Image HorizontalAlignment="Left" Stretch="Uniform" Width="168"

wpfAnimatedGif:ImageBehavior.AnimatedSource="pack://application:,,,/photo/1V7a.gif" Margin="3,283,0,-13" OpacityMask="White" RenderTransformOrigin="-0.872,0.35"/>

<StackPanel Grid.Column="1" Margin="10">

<TextBlock Text="Отзывы" FontSize="18" FontWeight="Bold" Margin="0,0,0,10" />

<ScrollViewer VerticalScrollBarVisibility="Auto" HorizontalScrollBarVisibility="Disabled" MaxHeight="300">

<ListBox x:Name="ReviewListBox" SelectionChanged="ReviewListBox\_SelectionChanged" MinHeight="300">

<ListBox.ItemTemplate>

<DataTemplate>

<Border BorderThickness="0,0,0,1" BorderBrush="Gray" Padding="5">

<StackPanel>

<TextBlock FontWeight="Bold" Text="{Binding Book.Title}" />

<TextBlock Text="Пользователь: " FontWeight="SemiBold" Margin="5,0,0,0">

<Run Text="{Binding User.Name}" />

</TextBlock>

<TextBlock Text="Оценка: " FontWeight="SemiBold" Margin="5,0,0,0">

<Run Text="{Binding Rating}" />

</TextBlock>

<TextBlock TextWrapping="Wrap" Text="{Binding Text}" MaxWidth="400" Margin="0,5,0,0" />

<TextBlock FontStyle="Italic" FontSize="10" Text="{Binding Date}" HorizontalAlignment="Right" />

</StackPanel>

</Border>

</DataTemplate>

</ListBox.ItemTemplate>

</ListBox>

</ScrollViewer>

</StackPanel>

</Grid>

</DockPanel>

</Window>

*Логика для окна Отзывы:*

using Microsoft.EntityFrameworkCore;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

using CommunityBookLovers.MessegeBoxes;

namespace CommunityBookLovers.LWindow

{

/// <summary>

/// Логика взаимодействия для ReviewWindow.xaml

/// </summary>

public partial class ReviewWindow : Window

{

private readonly LibraryContext \_context;

private readonly User currentUser;

public ReviewWindow(User user, LibraryContext context)

{

InitializeComponent();

\_context = context;

currentUser = user;

LoadBooks();

LoadReviews();

}

private void LoadBooks()

{

try

{

var books = \_context.Books.ToList();

BookComboBox.ItemsSource = books;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке книг: {ex.Message}");

}

}

private void LoadReviews()

{

if (\_context == null)

{

MessageBox.Show("");

return;

}

var reviews = \_context.Reviews

.Include(r => r.Book)

.Include(r => r.User)

.ToList();

ReviewListBox.ItemsSource = reviews;

}

private void ReviewListBox\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

if (ReviewListBox.SelectedItem is Review selectedReview)

{

DisplayReviewDetails(selectedReview);

}

}

private void DisplayReviewDetails(Review review)

{

if (review == null) return;

BookComboBox.Text = $"Книга: {review.Book?.Title ?? "Неизвестно"}";

RatingComboBox.Text = $"Рейтинг: {review.Rating}/5";

ReviewTextBox.Text = review.Text ?? "Отзыв отсутствует";

}

private void SaveReviewButton\_Click(object sender, RoutedEventArgs e)

{

if (BookComboBox.SelectedItem == null || RatingComboBox.SelectedIndex == -1 || string.IsNullOrWhiteSpace(ReviewTextBox.Text))

{

MessageBox.Show("Пожалуйста, заполните все поля перед сохранением.");

return;

}

var selectedBook = (Book)BookComboBox.SelectedItem;

var rating = RatingComboBox.SelectedIndex + 1;

var text = ReviewTextBox.Text;

var newReview = new Review

{

UserId = currentUser.UserId,

BookId = selectedBook.BookId,

Rating = rating,

Text = text,

Date = DateOnly.FromDateTime(DateTime.Now)

};

\_context.Reviews.Add(newReview);

\_context.SaveChanges();

MessageBox.Show("Ваш отзыв был успешно сохранён!");

LoadReviews();

ReviewListBox.ScrollIntoView(ReviewListBox.Items[ReviewListBox.Items.Count - 1]);

ReviewTextBox.Clear();

RatingComboBox.SelectedIndex = -1;

BookComboBox.SelectedIndex = -1;

}

private void MenuItem\_AboutMe\_Click(object sender, RoutedEventArgs e)

{

var aboutMeWindow = new AboutMeWindow(currentUser);

aboutMeWindow.Show();

this.Close();

}

private void MenuItem\_MyShelf\_Click(object sender, RoutedEventArgs e)

{

var bookshelfWindow = new BookshelfWindow(currentUser);

bookshelfWindow.Show();

this.Close();

}

private void MenuItem\_Books\_Click(object sender, RoutedEventArgs e)

{

var bookWindow = new BookWindow(currentUser);

bookWindow.Show();

this.Close();

}

private void MenuItem\_Friends\_Click(object sender, RoutedEventArgs e)

{

var friendsWindow = new CommunityBook(currentUser);

friendsWindow.Show();

this.Close();

}

private void MenuItem\_Review\_Click(object sender, RoutedEventArgs e)

{

MessegeReview messegeReview = new MessegeReview();

messegeReview.Show();

}

private void LogoutButton\_Click(object sender, RoutedEventArgs e)

{

var customMessageBox = new CustomWin();

customMessageBox.ShowDialog();

if (customMessageBox.Result)

{

foreach (var window in Application.Current.Windows)

{

if (window is ReviewWindow || window is CustomWin)

{

continue;

}

(window as Window)?.Close();

}

MainWindow mainWindow = new MainWindow();

mainWindow.Show();

this.Close();

}

}

}

}

## 4.1 Использование библиотек

При создании проекта были использованы такие библиотеки, как:

Microsoft.EntityFrameworkCore – Он содержит все необходимые классы и методы для работы с сущностями, контекстами базы данных, миграциями и запросами к базе данных. Этот пакет является обязательным при использовании EF Core.

Microsoft.EntityFrameworkCore.Design – инструменты для разработки и миграции.

Microsoft.EntityFrameworkCore.SqlServer – Пакет предназначен для работы с базой данных SQL Server. Он добавляет поддержку специфичных для SQL Server функций, таких как работа с хранимыми процедурами, оптимизация запросов под особенности этой СУБД и другие возможности.

Microsoft.EntityFrameworkCore.Tools – командлетные утилиты для работы с EF Core через командную строку.

## 4.2 Использование функций и методов

Метод DisplayUserInfo – предназначен для загрузки информации о пользователе

Если пользователь не редактирует данные isEditing = false, поля txtName и txtEmail отображаются, как только для чтения, а кнопки скрыты.

Если включен режим редактирования isEditing = true, пользователь может изменять поля, а кнопки становятся видимыми.

Метод SaveUserData – обновляет информацию о пользователе в базе данных через контекст LibraryContext.

LoadAuthors – Загружает список авторов из базы данных и сохраняет его в коллекции Authors для привязки данных к элементу AuthorComboBox.

Если возникает ошибка при загрузке, выводит сообщение об ошибке, а элемент выбора авторов отключается.

Метод LoadGenres – аналогично загружает список жанров его коллекции Genres для привязки к GenreComboBox

Метод ValidateInput – данный метод проверяет корректно ли введены пользователем данные.

Метод LoadBookshelf – данный метод загружает данные о книгах и пользователя из таблицы Bookshelf, включая данные из таблицы. Если полка пуста, то будет выведено сообщение об ошибке.

Метод FilterBooksByStatus – предназначен для фильтрации книг по статусу.

## 4.3 Таблицы базы данных

User – таблица для пользователя, содержит в себе следующие поля: user\_id, name, email, password, date, profileimagePath

Review – таблица для отзывов, содержит в себе следующие поля: review\_id, user\_id, book\_id, rating, text, date

Genre – таблица жанра, содержит следующие поля: genre\_id, name

Friendship – таблица друзья, содержит в себе следующие поля: friendship\_id, user\_id, friend\_id, isAccepted

Bookshelf – таблица полки, содержит следующие поля: bookshelf, book\_id, data\_added, user\_id, state.

Books – таблица книги, содержит следующие поля: book\_id, title, author\_id, genre\_id, publication\_year, pages, imagepath, description.

Author – таблица авторы, содержит следующие поля: author\_id, name, birth.

## 4.4 Основные кнопки приложения

AddBookButton\_Click – переход на страницу добавления книг.

MenuItem\_Review\_Click – переход на страницу отзывы.

MenuItem\_Friends\_Click – переход на страницу друзья.

MenuItem\_Books\_Click – переход на страницу книги.

MenuItem\_MyShelf\_Click- переход на страницу полка.

MenuItem\_AboutMe\_Click – переход на страницу о себе.

btnRegistration\_Click – переход на страницу регистрации.

btnRegistration\_Click – кнопка для выхода из аккаунта.

SaveReviewButton\_Click – кнопка для сохранения отзыва.

DeleteBookButton\_Click – кнопка для удаления книги с полки.

btnBack\_Click – кнопка для перехода на главную.

btnUploadImage\_Click – кнопка для загрузки фотографии.

btnSave\_Click – кнопка для сохранения данных о пользователе.

btnEdit\_Click – кнопка для редактирования данных о пользователе.

SaveButton\_Click – кнопка для сохранения книги.

SelectImageButton\_Click – кнопка для добавления фотографии книги.

SaveStatusButton\_Click – кнопка для сохранения статуса книги.

# 5. Разработка API

API (Application Programming Interface) — это интерфейс прикладного программирования, который позволяет различным программным системам взаимодействовать друг с другом.

Для работы с API необходимо создать следующий шаблон:

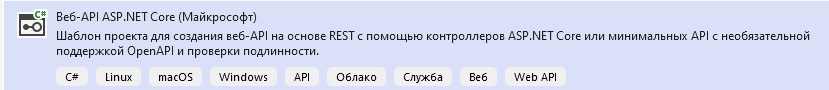


Рис. 33 «Шаблон для работы с API»

Добавим модели и контроллеры

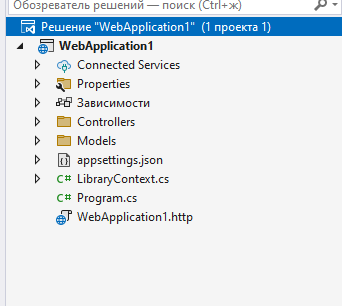


Рис. 34 «Структура API»

Код:

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

namespace WebApplication1.Controllers

{

[Route("api/[controller]")]

[ApiController]

public class LibraryController : ControllerBase

{

private readonly LibraryContext \_context;

public LibraryController (LibraryContext context)

{

\_context = context;

}

// GET: api/User

[HttpGet]

public async Task<ActionResult<IEnumerable<User>>> GetAll()

{

return await \_context.Users.ToListAsync();

}

// GET: api/user/{id}

[HttpGet("{id}")]

public async Task<ActionResult<User>> GetById(int id)

{

var user = await \_context.Users.FindAsync(id);

if (user is null)

return NotFound();

return user;

}

// POST: api/user/{id}

[HttpPost("{UserId}")]

public async Task<IActionResult> Create(User user)

{

\_context.Users.Add(user);

await \_context.SaveChangesAsync();

return CreatedAtAction(nameof(GetById), new { id = user.UserId }, user);

}

// PUT: api/user/5

// The method below will handle the PUT requests

[HttpPut("{Id}")]

public async Task<IActionResult> PutUser(int id, User user)

{

if (id != user.UserId)

{

return BadRequest();

}

\_context.Entry(user).State = EntityState.Modified;

try

{

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!UserExists(id))

{

return NotFound();

}

else

{

throw;

}

}

return NoContent();

}

private bool UserExists(int id)

{

return \_context.Users.Any(e => e.UserId == id);

}

// DELETE: api/user/5

[HttpDelete("{id}")]

public async Task<IActionResult> DeleteUser(int id)

{

var user = await \_context.Users.FindAsync(id);

if (user == null)

{

return NotFound();

}

\_context.Users.Remove(user);

await \_context.SaveChangesAsync();

return NoContent();

}

}

}

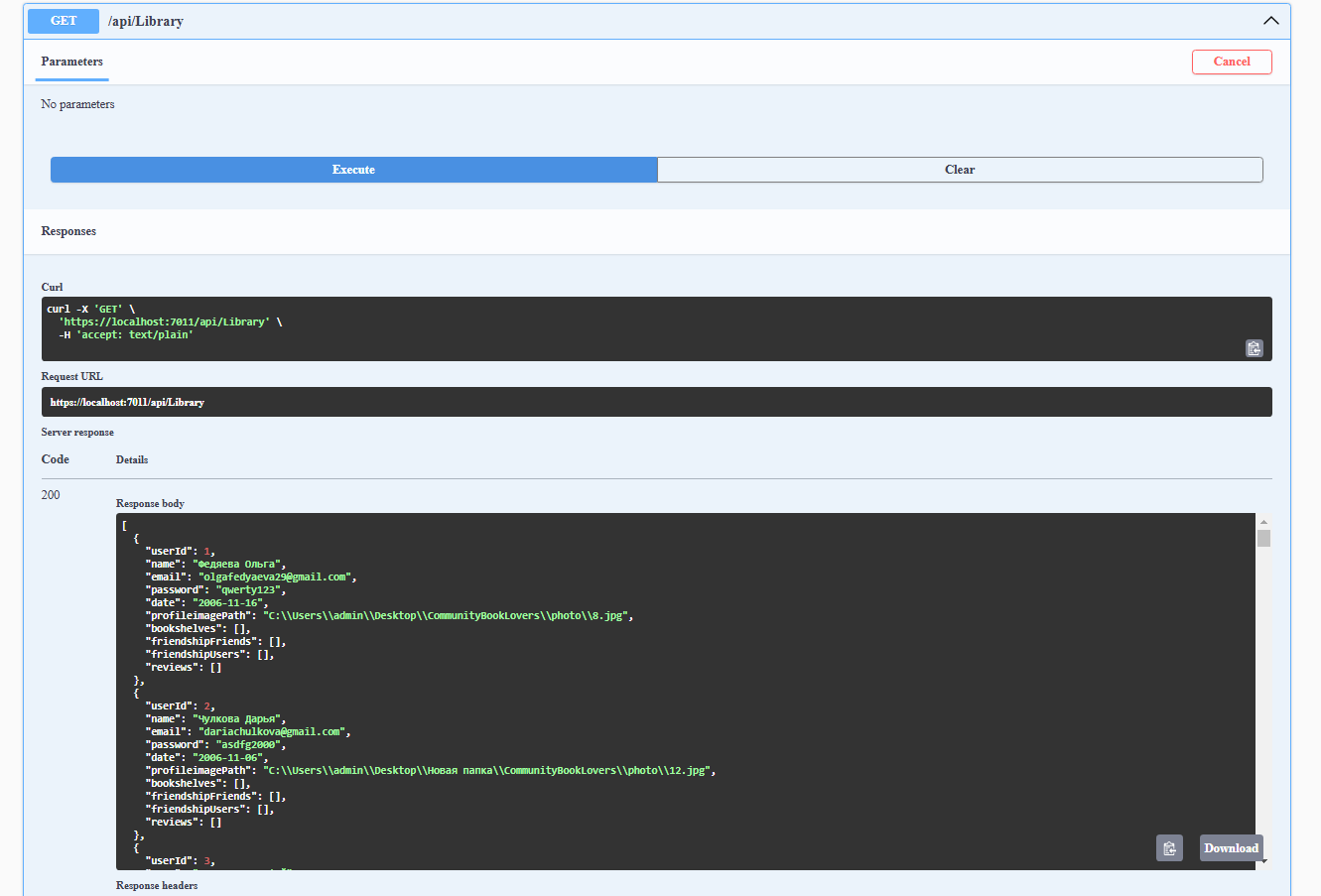


Рис. 35 «Просмотр всех данных»

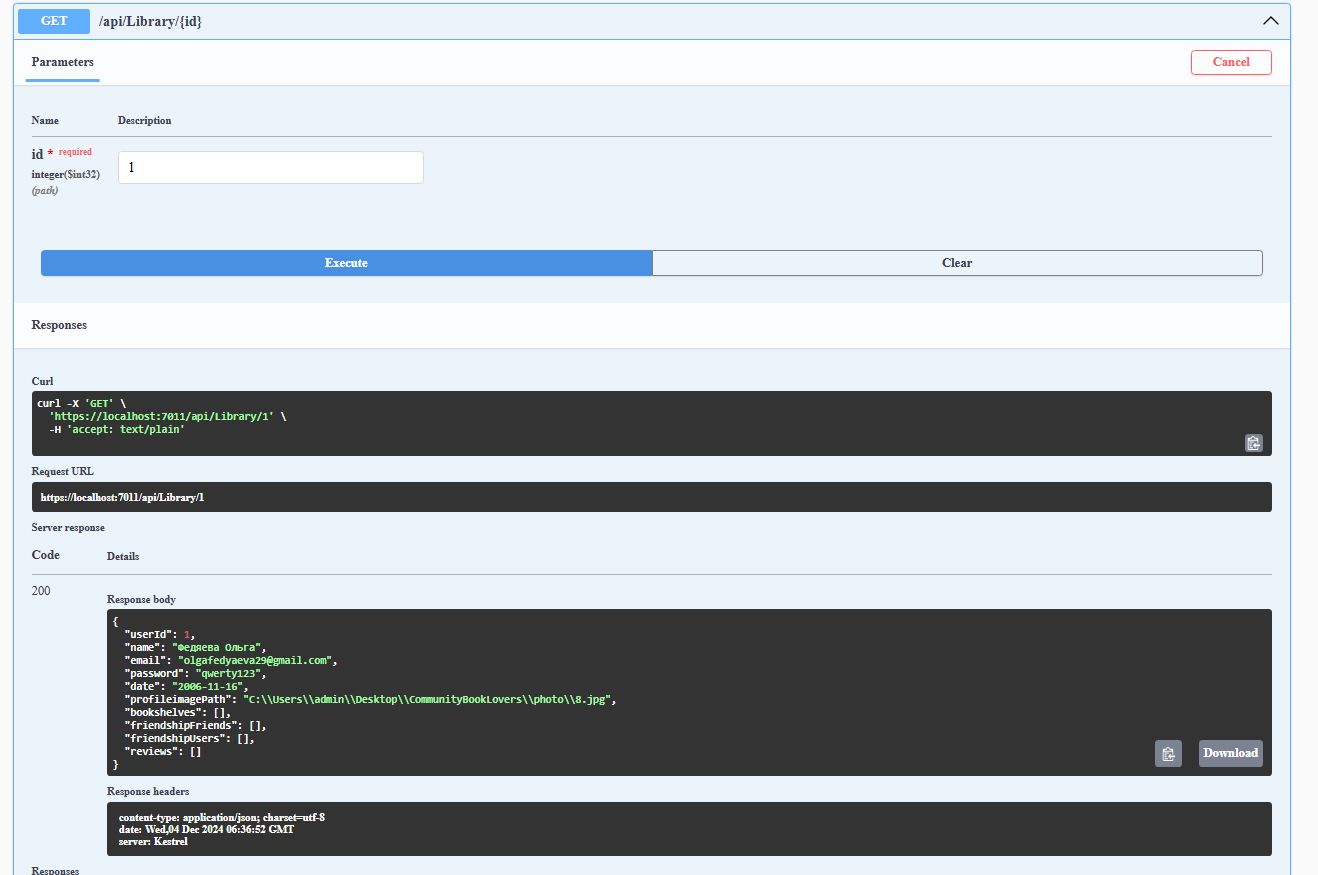


Рис. 36 «Просмотр пользователя по id»

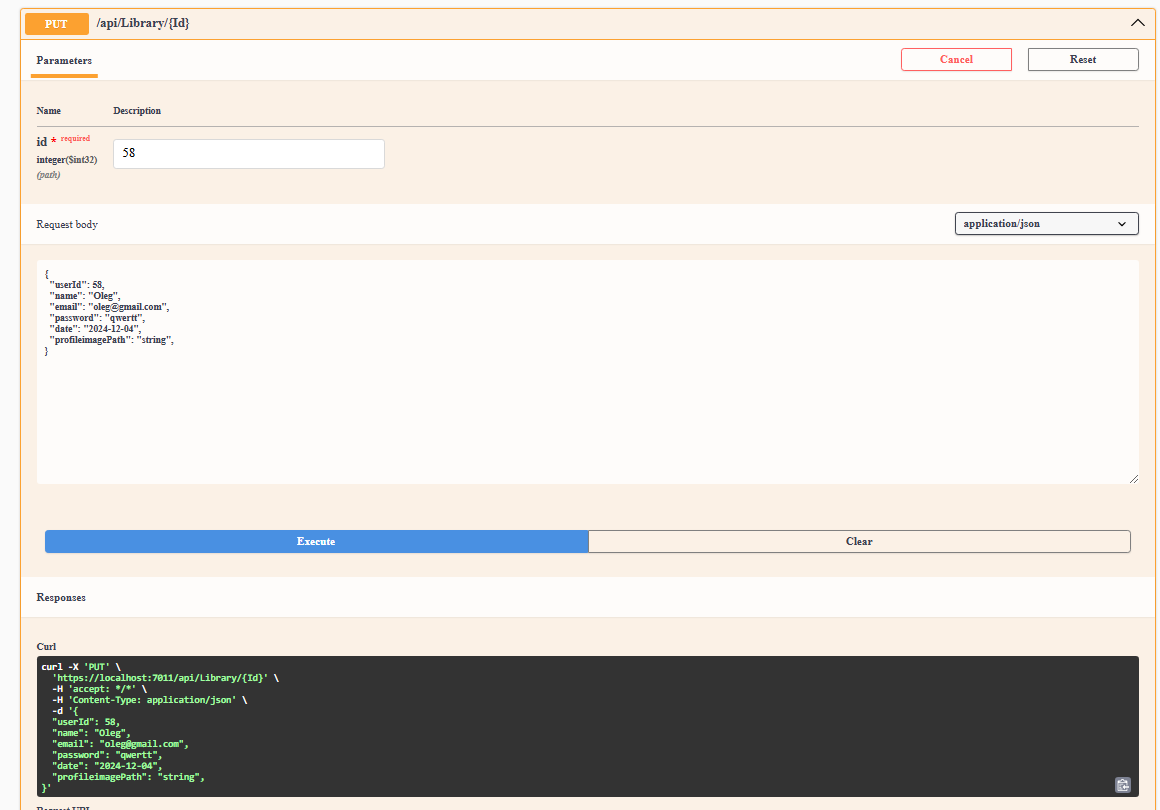


Рис. 37 «Обновление участников»

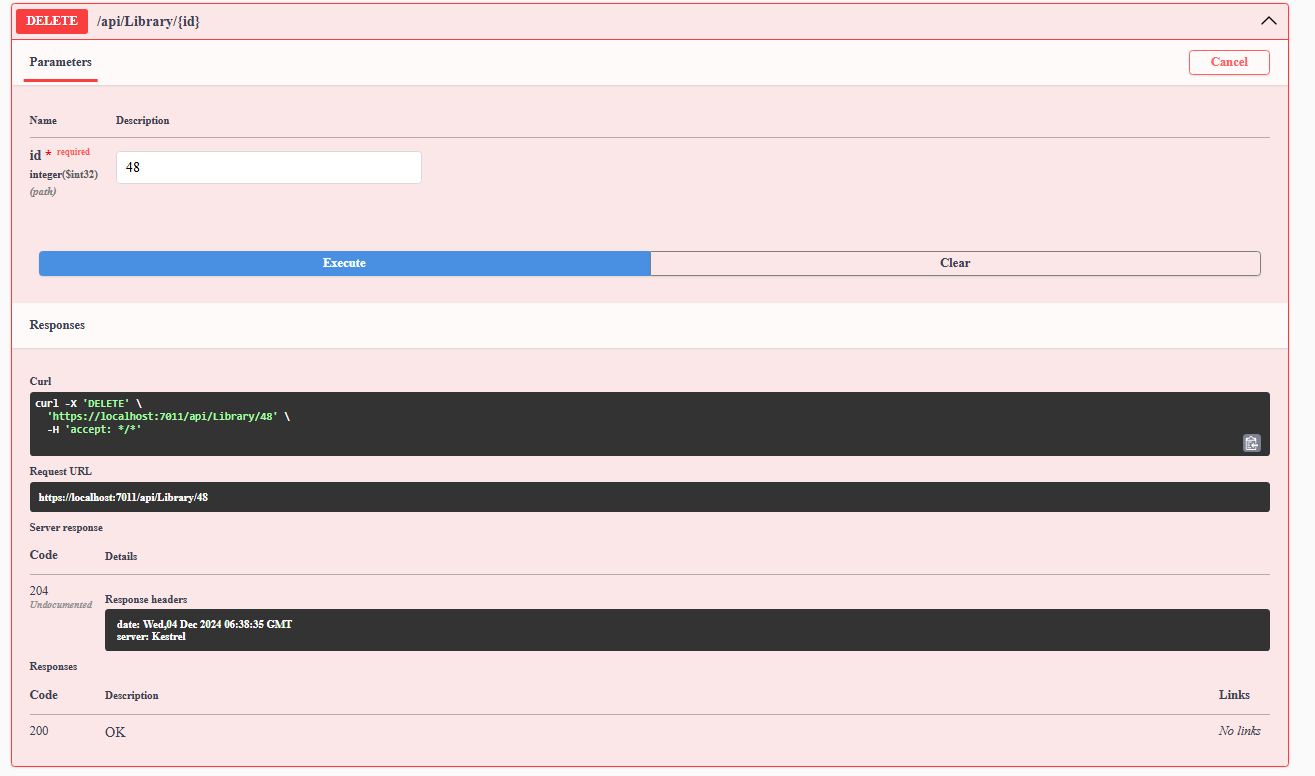


Рис. 38 «Удаление пользователя по id»

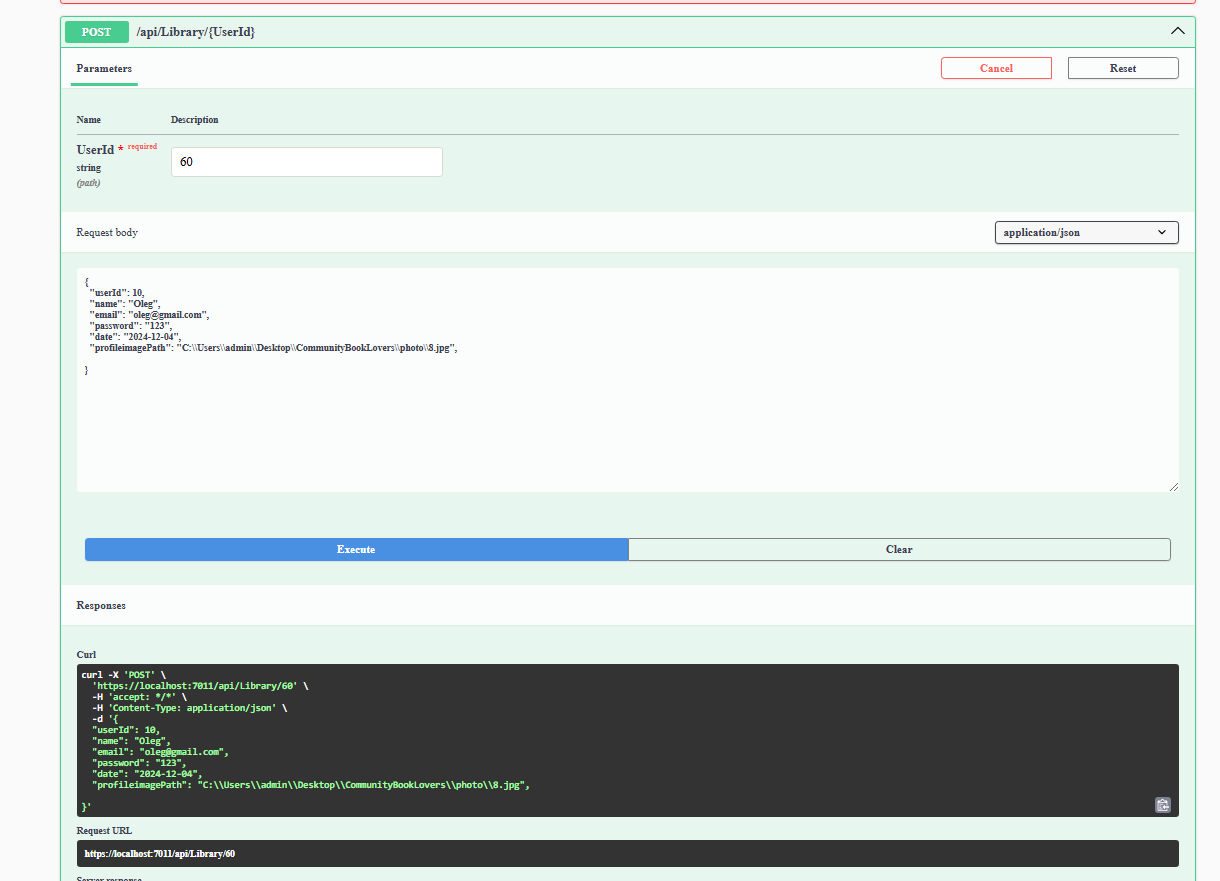


Рис. 39 «Добавление участника»

# 6. Библиотека классов

Библиотека классов — это проект в программировании, который представляет собой набор классов, методов, интерфейсов, и других компонентов, которые можно использовать в других проектах.

Итак, необходимо создать проект для Библиотеки классов.

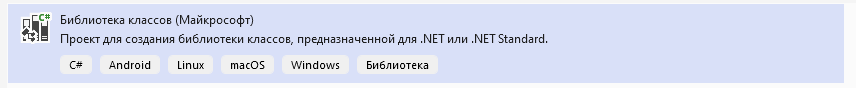


Рис. 40 «Шаблон для библиотеки классов»

Добавим модели.

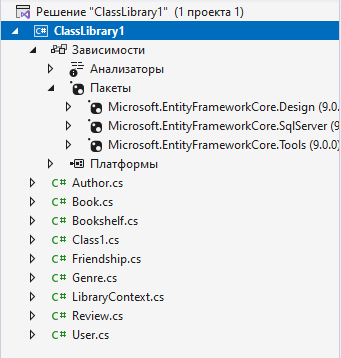


Рис. 41 «Структура библиотеки классов»

После нажимаем запуск, где появится следующее окно:

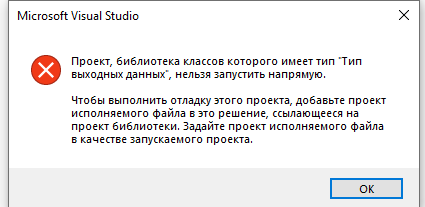


Рис. 42 «Запуск библиотеки классов»

После создадим проект, где будет создавать тесты.

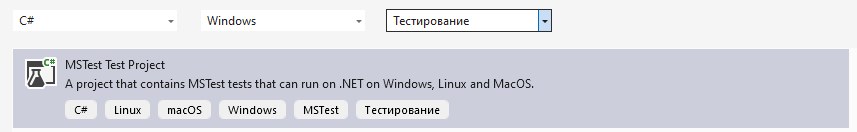


Рис. 43 «Шаблон для тестов»

После создания проекта, заходим в зависимости и добавляем ссылку на сборку. После добавления:

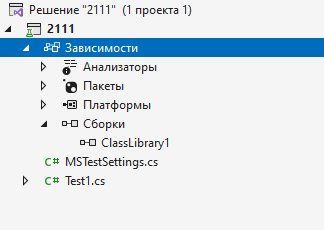


Рис. 44 «Структура проекта тестов»

Теперь напишем сами тесты. Код:

using ClassLibrary1;

namespace \_2111

{

[TestClass]

public sealed class Test1

{

[TestMethod]

public void TestHighRatedBooks()

{

var book = new Book { BookId = 1, Title = "Война и Мир" };

var review1 = new Review { ReviewId = 1, BookId = book.BookId, Rating = 5, Text = "Хорошая!" };

var review2 = new Review { ReviewId = 2, BookId = book.BookId, Rating = 4, Text = "Очень хорошая книга." };

var review3 = new Review { ReviewId = 3, BookId = book.BookId, Rating = 2, Text = "Не понравилось." };

book.Reviews = new List<Review> { review1, review2, review3 };

var threshold = 4;

var highRatedReviews = book.Reviews.Where(r => r.Rating >= threshold).ToList();

Console.WriteLine($"Book '{book.Title}' has {highRatedReviews.Count} reviews with rating >= {threshold}:");

foreach (var review in highRatedReviews)

{

Console.WriteLine($"- Review ID: {review.ReviewId}, Rating: {review.Rating}, Text: '{review.Text}'");

}

}

[TestMethod]

public void TestBooksByGenre()

{

var genre1 = new Genre { GenreId = 1, Name = "Роман" };

var genre2 = new Genre { GenreId = 2, Name = "Фантастика" };

var book1 = new Book { BookId = 1, Title = "Война и Мир", Genre = genre1 };

var book2 = new Book { BookId = 2, Title = "1984", Genre = genre2 };

var book3 = new Book { BookId = 3, Title = "Гарри Поттер", Genre = genre2 };

var allBooks = new List<Book> { book1, book2, book3 };

var romanBooks = allBooks.Where(b => b.Genre?.Name == "Роман").ToList();

Console.WriteLine("'Роман' genre:");

foreach (var book in romanBooks)

{

Console.WriteLine($"- {book.Title}");

}

}

[TestMethod]

public void TestAddAndRemoveReview()

{

var book = new Book { BookId = 1, Title = "Война и Мир" };

var user = new User { UserId = 1, Name = "Лев Толстой" };

var review1 = new Review { ReviewId = 1, BookId = book.BookId, UserId = user.UserId, Rating = 5, Text = "Шедевр!" };

var review2 = new Review { ReviewId = 2, BookId = book.BookId, UserId = user.UserId, Rating = 4, Text = "Отлично, но не без недостатков." };

book.Reviews.Add(review1);

book.Reviews.Add(review2);

Console.WriteLine($"Book '{book.Title}' has {book.Reviews.Count} reviews.");

book.Reviews.Remove(review1);

Console.WriteLine($"After removal, the book '{book.Title}' has {book.Reviews.Count} reviews.");

}

[TestMethod]

public void TestAddBookToBookshelf()

{

var book = new Book { BookId = 1, Title = "Война и Мир" };

var user = new User { UserId = 1, Name = "Лев Толстой" };

var shelf1 = new Bookshelf

{

Bookshelf1 = 1,

Book = book,

User = user,

State = "Читаю",

DataAdded = DateOnly.FromDateTime(DateTime.Now)

};

var shelf2 = new Bookshelf

{

Bookshelf1 = 2,

Book = book,

User = user,

State = "Прочитано",

DataAdded = DateOnly.FromDateTime(DateTime.Now)

};

book.Bookshelves.Add(shelf1);

book.Bookshelves.Add(shelf2);

Console.WriteLine($"Book '{book.Title}' is on {book.Bookshelves.Count} shelves.");

foreach (var shelf in book.Bookshelves)

{

Console.WriteLine($"Shelf {shelf.Bookshelf1}: {shelf.State}");

}

}

}

}

TestHighRatedBooks – тест предназначен для фильтрации отзыва по рейтингу.

TestBooksByGenre – тест проверяет, как происходит фильтрация книги по жанру.

TestAddAndRemoveReview – тест на проверку добавления и удаления отзывов для книги.

TestAddBookToBookshelf – тест проверяет, как книга добавляется на полку и какой у неё статус, то есть «Читаю», «Прочитал», «Хочу прочитать».

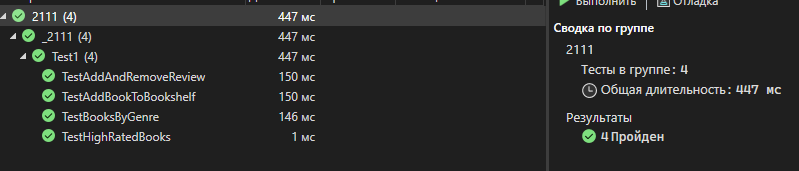


Рис. 45 «Проверка тестов»

# Заключение

Итак, приложение выполнено в соответствии с техническим заданием, есть дополнительное приложение API и библиотеку классов.

В будущем хочется расширить приложение, а именно реализовать чтение книг в самом приложении, сделать ограничение прав доступа, улучшить вкладку «Друзья», а также личную полку читателя.

Реализовать создание книжных клубов, для тех, кто хочет обсудить книги, будет возможность создавать книжные клубы, где пользователи смогут приглашать других участников и обсуждать выбранные книги.

Улучшить поиск и фильтрацию, чтобы пользователи смогли искать книги по несколькими критерия одновременно.

Реализовать персонализированные рекомендации для пользователей, которые будет анализировать предпочтения пользователей и на основе прочитанных книг будет предлагать новые произведения, подходящие по жанру, автору и стилю.

Реализовать функцию достижения, пользователи смогут получать баллы за выполнение различных действий, за прочитанные книги, написанные отзывы.

Возможно разработать коллекцию книг или тематические подборки, к примеру, «Классическая литература», «Книги для саморазвития», чтобы пользователи могли делиться этими коллекциями с другими.

Также возможное обновление пользовательского интерфейса.