Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Кубанский государственный технологический университет»

(ФГБОУ ВО «КубГТУ»)

Кафедра информационных систем и программирования

ОТЧЕТ

о прохождении преддипломной практики

на \_ИП Дубовсков Денис Александрович, г. Краснодар\_\_

(предприятие, населенный пункт)

Студента \_Савицкого Романа Владиславовича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

группы \_16-КБ-ПР1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

факультета \_ИКСИБ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_

направление подготовки 09.03.03 Прикладная информатика\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель практики от предприятия \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М.А. Александров

(подпись, печать)

Руководитель практики от кафедры\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Г. Волик

(подпись)

Председатель комиссии - Янаева М.В., зав. кафедрой ИСП, доц;

Члены комиссии: Мурлина В.А., доц.;

Мурлин А.Г., доц;

Волик А.Г., ст. препод.

Отчет защищен на оценку\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Краснодар

2020

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Кубанский государственный технологический университет»

(ФГБОУ ВО «КубГТУ»)

Институт компьютерных систем и информационной безопасности

Кафедра информационных систем и программирования

УТВЕРЖДАЮ

Зав. кафедрой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

27.03.2020 М.В. Янаева

дата

**Задание**

**на** \_\_\_\_\_\_\_\_ **преддипломную**\_\_\_\_\_\_\_\_\_\_\_\_  **практику**

Студенту \_Савицкому Роману Владиславовичу\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

группы \_16-КБ-ПР1\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

факультета \_ИКСИБ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_

направление подготовки 09.03.03 Прикладная информатика\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема задания на практику Разработка web-приложения «Литература». Backend-разработка. Комплексный проект

Место прохождения практики\_\_ ИП Дубовсков Денис Александрович

Календарный план прохождения практики:

|  |  |  |
| --- | --- | --- |
| № п/п | Этап практики | Срок |
| 1 | Ознакомление с деятельностью предприятия | 27.04.2020 – 04.05.2020 |
| 2 | Изучение и анализ предметной области | 04.05.2020 – 11.05.2020 |
| 3 | Проектирование веб-приложения | 11.05.2020 – 18.05.2020 |
| 4 | Подведение итогов и написание отчета | 18.05.2020 – 24.05.2020 |

Руководитель практики от кафедры\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Г. Волик

(подпись)

Задание принял студент \_\_\_\_\_\_\_\_\_ \_\_\_ Р.В. Савицкий

(подпись)

**ОТЗЫВ ПРОФИЛЬНОЙ ОРГАНИЗАЦИИ**

**о прохождении студентом (кой) практики**

Студент(ка) Савицкий Роман Владиславович

(фамилия, имя, отчество, полностью)

4 курса 09.03.04 «Программная инженерия» направления 16-КБ-ПР1 группы

Прошел (ла) преддипломную практику

(вид практики)

на ИП Дубовсков Денис Александрович, г. Краснодар

(наименование предприятия, населенный пункт)

1. Сроки практики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(соблюдены, не соблюдены)

1. Нарушение трудовой и исполнительной дисциплины\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(допускал (а), не допускал (а))

1. Общий уровень теоретической подготовки\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(достаточный,недостаточный)

4. Способность работать с технической документацией\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(проявил (а), не проявил (а))

5. Уровень коммуникабельности\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

( низкий, средний, высокий)

6. Получение рабочей профессии\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(да(разряд), нет)

1. Оценка прохождения практики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

8. Имеется ли перспектива трудоустройства на профильной организации после окончания университета \_\_\_\_\_\_\_\_\_

(да, нет)

Руководитель практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_И.О. Фамилия

от профильной организации (должность, подпись)

М.П.

**Реферат**

Курсовой проект: 30 с., 9 рис., 1 табл., 5 источников, 4 листинга

БАЗЫ ДАННЫХ, СУЩНОСТЬ, ER-ДИАГРАММА, MS SQL SERVER, ЯЗЫК C#, BACKEND, ASP NET CORE, ENTITY FRAMEWORK, КЛАСС, МЕТОД, МОДЕЛЬ, КОНТРОЛЛЕР, ПРЕДСТАВЛЕНИЕ, СЕРВЕР

Объектом исследования является веб-приложение «Литература». Данное предложение содержит книги, размещенные в различных читательских клубах, а также пользователей системы, которые могут приобретать данные книги.

Цель курсового проекта состоит в разработке веб-приложения, написанного на языке C# в среде Visual Studio. Для хранения информации о системе используется реляционная база данных MS SQL Server.

Курсовой проект содержит описание предметной области, проектирование структуры базы данных и ее реализация средствами СУБД, проектирование и реализация десктоп и веб-приложения.

Результатом проекта является многофункциональное приложение, организующая взаимодействие пользователей с объектами системы, где посетители ресурса смогут находить и приобретать нужные им книги, а администраторы будут поддерживать контроль работы всей системы.

**Содержание**

[Введение 6](#_Toc41660196)

[1 Постановка задачи 7](#_Toc41660197)

[1.1 Описание предметной области 7](#_Toc41660198)

[1.2 Требования к структуре ПО и базе данных 8](#_Toc41660199)

[2 Backend-разработка 10](#_Toc41660200)

[2.1 ASP.NET Core 10](#_Toc41660201)

[2.2 Паттерн MVC 11](#_Toc41660202)

[3 Проектирование и создание базы данных 14](#_Toc41660203)

[3.1 Описание сущностей 14](#_Toc41660204)

[3.2 Построение концептуальной и логической диаграмм 15](#_Toc41660205)

[3.3 Построение ER-диаграммы 17](#_Toc41660206)

[3.4 Создание реляционной базы данных 18](#_Toc41660207)

[4 Взаимодействие пользователя с веб-сайтом 25](#_Toc41660208)

[4.1 Авторизация на сайте 25](#_Toc41660209)

[4.2 Добавление новой книги на сайт 27](#_Toc41660210)

[Заключение 30](#_Toc41660211)

[Список использованных источников 31](#_Toc41660212)

# **Введение**

Предметной областью данного курсового проекта является информационная система «Библиотека». Ее роль заключается в хранении различных произведений и другой печатной информации для общественного пользования.

Цель работы – реализация базы данных средствами СУБД, а также разработка приложения, позволяющее манипулировать с ее данными. Оно должно обладать удобным и понятным пользовательским интерфейсом, а также содержать реализацию для всего функционала системы.

В качестве системы управления базами данных используется MS SQL Server 2017. Для проектирования структуры базы данных и приложения применяется онлайн-сервис Creately, позволяющий разрабатывать диаграммы на языке моделирования UML. Разработка приложения осуществлена на объектно-ориентированном языке высокого уровня C# в среде разработки Visual Studio 2019.

Результатом работы будет приложение, осуществляющие функционирование информационной системы «Библиотека». Используя данное приложение, пользователь сможет найти и приобрести интересующие его произведения. Он также будет иметь возможность стать частью одного из читательских клубов, где он сможет общаться с другими участниками, оставлять свои рецензии на различные произведения и даже публиковать свои.

Выполнение курсового проекта поспособствует повышению квалификации в области программирования, а также приобретению новых знания для проектирования структуры базы данных и ее реализации средствами СУБД.

# **1 Постановка задачи**

* 1. **Описание предметной области**

Библиотека – информационное, культурное, образовательное учреждение, располагающее организованным фондом документов и представляющее их во временное пользование абонентам, а также осуществляющее другие библиотечные услуги.

Деятельность библиотек по обслуживанию читателей осуществляется следующим образом. Библиотечный абонемент предоставляет читателю право получить издание из библиотеки в своё полное распоряжение на определённый срок. В другом случае читатель имеет возможность знакомиться с книгой только в помещении библиотеки (как правило, в специально отведённом «читательском зале»). В некоторых библиотеках работает только абонемент или только читальный зал, в других эти формы обслуживания сочетаются, хотя не для всех единиц хранения возможны обе.

Информационная система «Библиотека» должна позволять:

– хранить основные сведения о книгах в библиотеке (инвентарный номер, автор, название, год издания, раздел знаний), журналах и газетах, посетителях библиотеки (код читателя, фамилия, имя, отчество, группа, категория), сотрудниках библиотеки, наличии книги (журнала, газеты) в библиотеке;

– производить ввод новых данных в систему и редактировать имеющиеся данные;

– находить сведения о конкретных книгах (журналах, газетах) и посетителях;

– содержать различные справки, ведомости, отчеты;

– находить информацию по заданной тематике.

Целью данного курсового проекта является создание ПО, предназначенного для автоматизации информационной системы «Библиотека». Она осуществляет три основные функции: хранение, поиск, и выдача на определенное время имеющихся в наличие книг. Пользователи разделены на три типа по уровню доступа к системе: гость, обычный пользователь и администратор.

В дополнении к этому, данная система содержит информационные порталы «Читательские клубы», являющиеся аналогами веб-форумов, где пользователи системы могут вести общение, оставлять рецензии на произведения и публиковать свои.

* 1. **Требования к структуре ПО и базе данных**

Разобрав предметную область и определившись с целью создания ПО для информационной системы «Библиотека», выделим следующий перечень требований к системе.

Требования к архитектуре:

– для хранения данных системы используется реляционная база данных MS SQL Server 2017;

– для взаимодействия клиентов ПО с системой используется локальный веб-сервер IIS;

– для написания серверной части веб-приложения используется язык программирования C#;

– для написания клиентской части веб-приложения используются языки разметки HTML/CSS.

Требования к ПО:

– организация основных методов для взаимодействия с базой данных, таких, как создание, редактирование и удаление данных;

– отображение данных пользователю в понятном для него виде;

– удобный и отзывчивый пользовательский интерфейс;

– обеспечение безопасного и надежного хранения личных данных;

– стабильная производительность и устойчивость к сбоям;

– тестирование и дальнейшее сопровождение ПО.

Требования к структуре базы данных:

– целостность БД;

– многократное использование данных;

– быстрый поиск и получение данных по запросам;

– уменьшение излишней избыточности сущностей и их связей;

– нормализация отношений (использования нормальных форм);

– защита данных от искажения, уничтожения и несанкционированного доступа.

Требования к информационной системе «Библиотека»:

1. Функции для книг:

– xранение книг и отображение их сведений;

– поиск книги по ключевым словам;

– возможность добавления новых книг, а также удаления или изменения старых.

2. Функции для пользователей:

– регистрация нового пользователя;

– отображения данных о пользователях системы;

– взятие под залог книги на определенное время;

– занятие место в очереди за получением определенной книги;

– потеря книги.

3. Функции для администратора:

– добавление, удаление и редактирование данных;

– выборки из базы данных для составления отчетов на печать.

# **2 Backend-разработка**

Под back-end разработкой понимается программирование веб-сайта на стороне сервера. Описываются данные, которые используются в приложении, а также логика работы с этими данными. Для разработки серверной части приложения программист использует набор необходимых ему аппаратно-программных средств.

Основным компонентом является сервер приложения. С аппаратной точки зрения сервер выполнят роль хранилища всех данных, используемых для работы приложения. Обычно в качестве сервера используются персональные компьютеры или совокупность компьютеров и других аппаратных средств в виде рабочих станций. С точки зрения программного обеспечения используется понятие веб-сервера. Веб-сервер отвечает за доступ пользователей сайта к данным, расположенных на физическом сервере. Данный процесс осуществляется за счет различных компонентов, таких как HTTP-протокол и URL-адреса.

В наши дни для создания сайтов или других веб-приложений обычно используются различные фреймворки или платформы для веб-разработки. В данном работе используется технология ASP.NET Core.

**2.1 ASP.NET Core**

ASP.NET Core – платформа для создания различных веб-приложений от компании Microsoft. Данный фреймворк был выбран в качестве основного средства разработки за счет следующих преимуществ:

1. Быстрота и удобство разработки. ASP.NET Core содержит различные стартовые шаблоны проектирования. При создании нового проекта, фреймворк автоматически определяет конфигурацию приложения, его взаимодействие с сервером и так далее. Благодаря этому разработчик уже при первой компиляции проекта сможет запустить приложение на сервере и начать внедрение собственных идей и наработок, экономя свои силы и время.

2. Кроссплатформенность. Фреймворк может быть использован на всех известных операционных системах: Windows, Linux или MacOS. Все веб-приложения, созданные в этой среде, могут быть запущены на любой из вышеперечисленных ОС.

3. Расширяемость. Так как ASP.NET Core состоит из различных независимых компонентов и библиотек, мы с легкостью можем расширить их реализацию с помощью внедрения зависимостей и механизма. Также можно использовать собственные компоненты с последующем расширением функциональности.

4. Открытый исходный код. На данный момент фреймворк является opensource-проектом, а все его коды и файлы размещены на GitHub.

Основным языком программирования на ASP.NET Core является C#, а средой разработки – Visual Studio. Данный факт является основным в пользу моего личного выбора данной технологии для разработки серверной части веб-приложения.

**2.2 Паттерн MVC**

Данный паттерн предназначен для упрощения процесса создания приложения за счет разделения графического интерфейса от бизнес логики и хранимых данных. Приложение условно разделяется на три компонента: модель, представление и контроллер (рисунок 1):

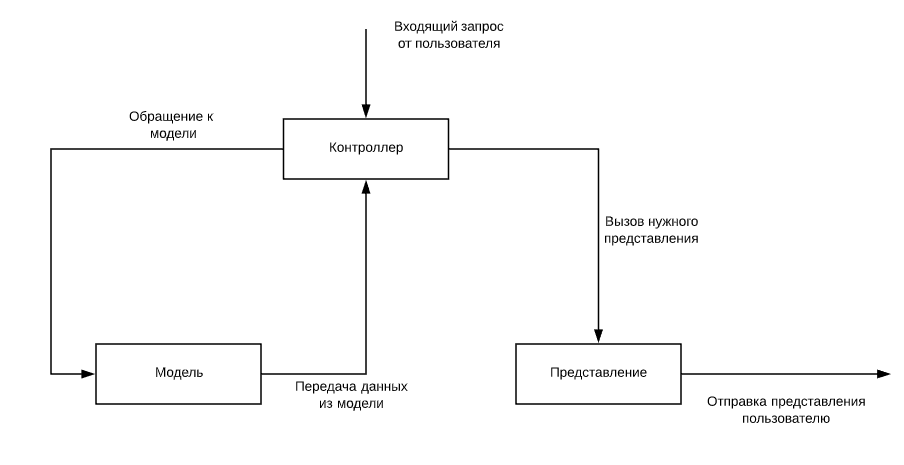


Рисунок 1 – Визуальная модель паттерна MVC

Модель (model) содержит описание данных приложения на сервере, а также логику управления с ними. В данном проекте также используется модель представления данных (view-model), которая используется представлениями для отображения данных, переданных с сервера.

Представление (view) описывает пользовательский интерфейс приложения. В случае разработки веб-сайта обычно используются HTML-страницы в связке с языком стилей CSS. Также представления содержат логику отображения данных и их передачи на сервер. Обычно для этого используется язык JavaScript, однако в ASP.NET Core доступна технология Razor Pages, которая позволяет внедрять C# код прямо на пользовательской стороне.

Контроллер (controller) представляет компонент, связывающий представление с моделью. Содержит логику обработки пользовательских запросов, передачи их на модель. Затем контроллер отправляет пользователю представление с определенным выводом в зависимости от результатов обработки данных моделью.

В ASP.NET Core реализация моделей и контроллеров осуществляется при помощи разработки классов на языке C#. Их реализация и является основой backend-разработки на ASP.NET Core MVC.

# **3 Проектирование и создание базы данных**

Первым этапом в разработке веб-приложения является проектирование базы данных. Так как в данном проекте будет использоваться реляционная БД, изначально необходимо абстрактно распределить данные по схожим чертам и характеристикам. Для этого определим сущности объектов и виды отношений между ними, после чего наступает этап проектирования концептуальной модели БД. Затем начинается этап проектирования логической модели БД, на котором необходимо упростить связи между сущностями таким образом, чтобы их организация была оптимальной и возможной для проектирования физической модели БД. Последним этапом является создание БД на физическом уровне. Для этого будет использована система РСУБД MS SQL Server, общая структура которой представлена в виде ER-диаграммы.

**3.1 Описание сущностей**

После этапов анализа предметной области и разбора технического задания можно приступить к проектированию и реализации базы данных. Определив требования к информационной системе, выявим ее сущности, на основе которых будет разработана база данных.

При рассмотрении структуры БД были выявлены следующие сущности:

«Книги» – ключевая сущность нашего веб-сайта. У книг может быть название, автор, дата издания, содержание и другие атрибуты, которые будут уточнены при проектировании реляционной БД.

«Теги» - у каждой книги будет свой набор тегов, по которым их можно будет разделять на группы или выполнять поиск книг по определенным тегам.

«Пользователи» – это все люди, которые каким-либо образом взаимодействуют с данной информационной системой. Это могут быть случайные пользователи, посетившие наш сайт впервые, зарегистрированные пользователи, модераторы, администраторы и другие. У них должно быть имя, возраст и роль.

«Роли пользователей» – из-за возможности пользователей сайта обладать различными ролями с определенными правами, данная сущность становится необходимой в реализации.

«Форум» - на нашем сайте будет реализован форум, на котором различные пользователи смогут оставлять темы для обсуждений, вести дискуссии с другими пользователями по той или иной теме и так далее.

«Комментарии» - под каждой темой на форуме пользователи смогут оставлять комментарии, высказывая свое мнение по тому или иному вопросу.

«Оценки» - сущность, являющаяся хранилищем оценок для книг и тем на форуме. Пользователь сможет оценить любое произведение на свое усмотрение.

**3.2 Построение концептуальной и логической диаграмм**

Описав все сущности БД, можно приступить к определению типов связей между ними (таблица 1).

Таблица 1 – Типы связи сущностей

|  |  |  |
| --- | --- | --- |
| Название первой сущности | Тип  связи | Название второй  сущности |
| Пользователи | 1:N | Книги |
| Пользователи | 1:N | Форум |
| Пользователи | 1:N | Комментарии |
| Пользователи | 1:N | Оценки |
| Книги | 1:N | Комментарии |
| Книги | 1:N | Оценки |
| Форум | 1:N | Комментарии |
| Форум | 1:N | Оценки |
| Комментарии | 1:N | Оценки |
| Роли пользователей | N:N | Пользователи |
| Теги | N:N | Книги |

После определения всех типов связей между сущностями системы, реализуем концептуальную диаграмму, показанную на рисунке 2.

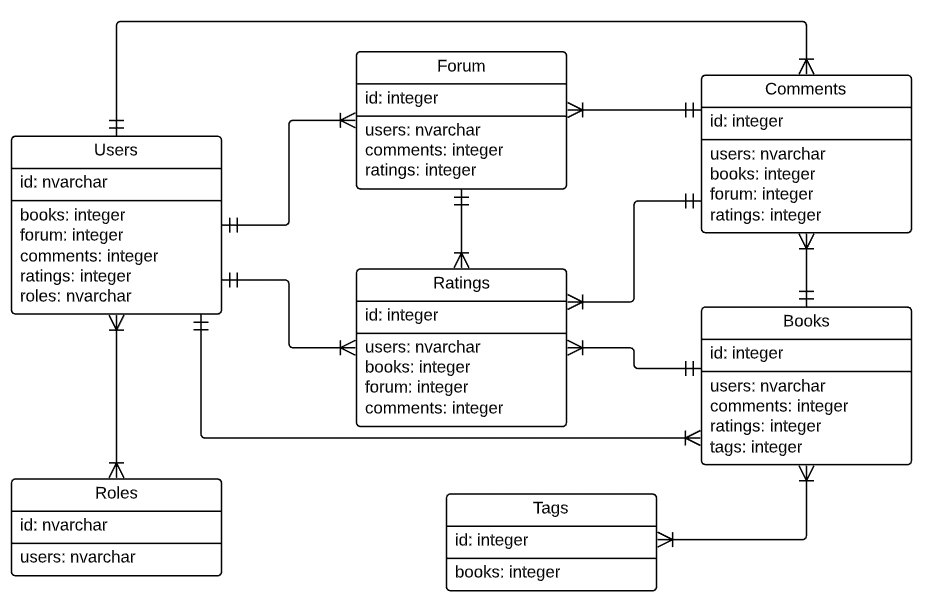


Рисунок 2 – Концептуальная диаграмма БД

Далее упростим концептуальную модель путём создания смежных таблиц между таблицами со связью «многие ко многим», так как в реляционной модели базы данных связь «многие ко многим» не может быть реализована напрямую. Такими являются связи между таблицами «Книги»--«Теги» и «Пользователи»--«Роли». Создадим две связующие таблицы «КнигиТеги» и «РолиПользователи». Затем добавляем связь «один ко многим» между каждой из первоначальных таблиц к связующим, которые должны находиться на стороне «многие» с обеих сторон.

Упростив отношения между таблицами, построим логическую модель базы данных (рисунок 3):

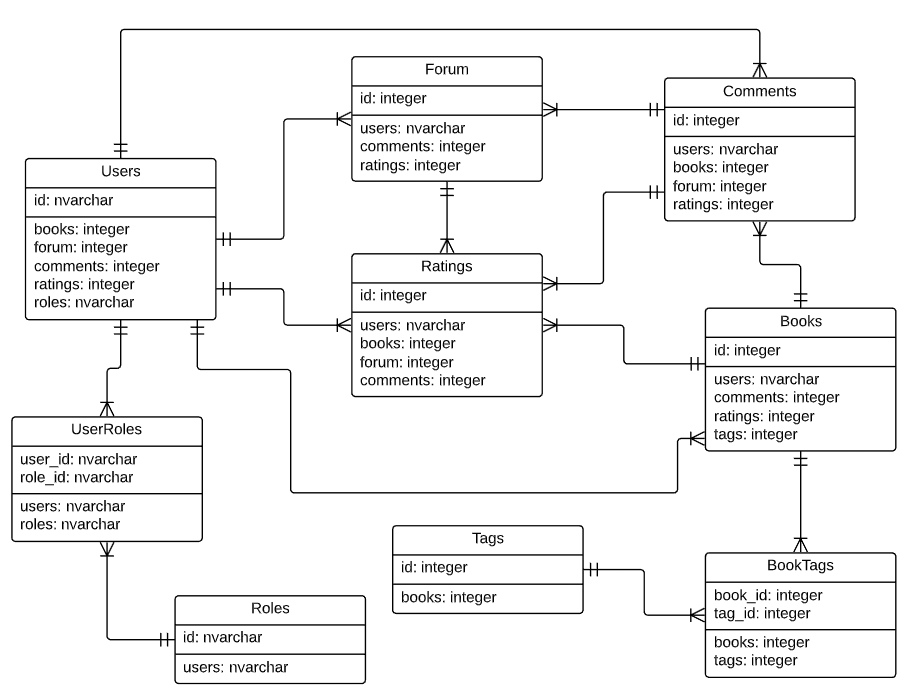


Рисунок 3 – Логическая диаграмма БД

**3.3 Построение ER-диаграммы**

На основании спроектированной логической модели начинается этап разработки таблиц для будущей реляционной БД. Для проектирования используем утилиту SQL Server Management Studio (SSMS). Проанализировав все сущности и типы отношений между ними, необходимо определить наборы атрибутов с их типами данных. Используя утилиту SSMS, спроектируем ER-диаграмму, представляющую РБД со всеми ее таблицами, атрибутами и типами данных для этих таблиц, а также типами отношений между всеми таблицами. Полученная ER-диаграмма представлена на рисунке 4.

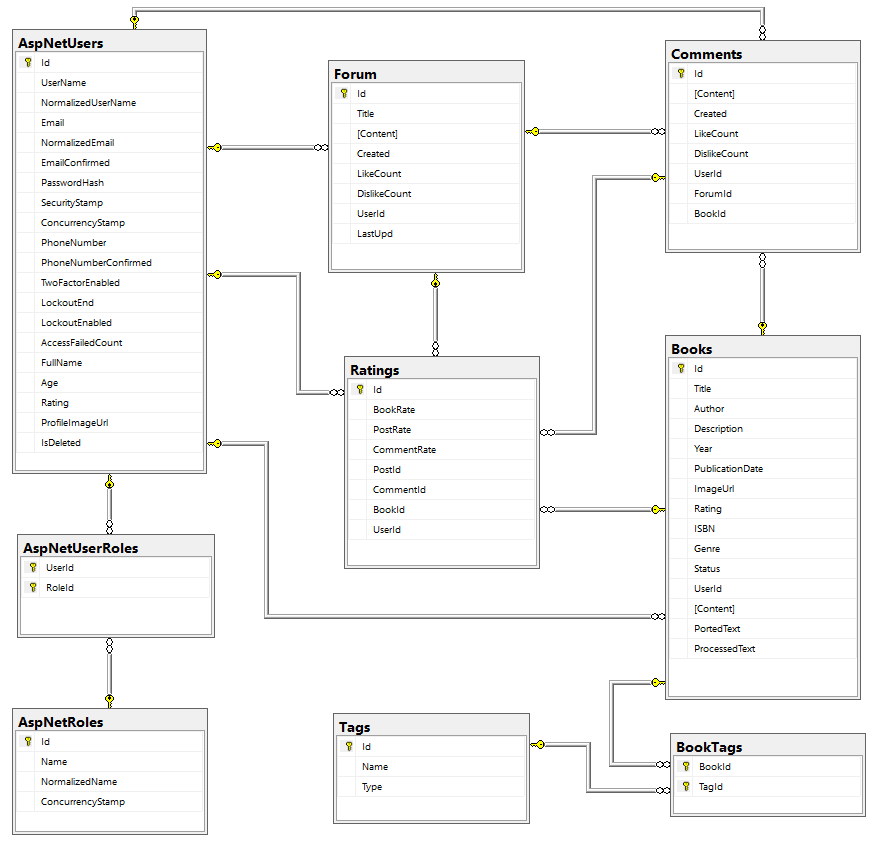


Рисунок 4 – ER-диаграмма

**3.4 Создание реляционной базы данных**

Для создания РБД на основе спроектированной ER-диаграммы, мы не будем применять классический подход создания БД с помощью конструктора или SQL-запросов. Учитывая, что мы создаем веб-приложение с применением технологии ASP.NET Core и паттерна MVC, гораздо удобнее воспользоваться встроенным фреймворком Entity Framework Core (EFC).

EFC – это ORM (Object-Relational Mapping) технология, позволяющая работать с реляционными базами данных на абстрактном уровне, т.е. абстрагироваться от физического представления БД и ее таблиц. Она также позволяет автоматизировать процесс создания и обновления БД за счет встроенных функций генерации SQL-кода и его исполнения. Мы же воспользуемся подходом Code First, который заключается в изначальном определении модели контекста данных, и уже исходя из этих моделей EFC реализует БД со всеми ее таблицами и связями.

Изначально нам нужно создать модели наших сущностей. Для этого определим классы на языке C#. Название классов будет соответствовать названию таблиц БД, а их определение – атрибутам этих таблиц и связями между ними. Рассмотрим на примере класса «Book», определяющий модель книги (листинг 1):

Листинг 1 – Класс «Book»

public class Book

{

public int Id { get; set; }

[Required]

public string Title { get; set; }

[Required]

public string Author { get; set; }

public string Description { get; set; }

[Required]

public int Year { get; set; }

[Required]

public DateTime PublicationDate { get; set; }

public string ImageUrl { get; set; }

[Required]

public double Rating { get; set; }

[Required]

public string Genre { get; set; }

[Required]

public string Status { get; set; }

[Required]

public string Content { get; set; }

//FK's

public User User { get; set; }

public IEnumerable<BookTag> BookTags { get; set; }

public IEnumerable<Comment> Comments { get; set; }

public IEnumerable<Rating> Ratings { get; set; }

}

Для определения атрибутов таблицы «Book» в классе используются публичные свойства. Тип этих свойств определяет тип данных атрибутов базы данных. Атрибут «Required» определяет свойство модели как «NOT NULL» в БД. Для определения атрибута как внешнего ключа используется тип класса модели, на которую ссылается текущая таблица. Если она связана отношением «один ко многим», то тип класса оборачиваем в коллекцию. Для данной модели определим следующие свойства:

* Id – первичный ключ, является основный для данной модели;
* Title – название книги;
* Author – имя автора книги;
* Description – краткое описание книги;
* Year – год издания книги;
* PublicationDate – дата публикации книги на сайте;
* Genre – жанр произведений книги;
* Rating – рейтинг книги среди пользователей;
* Status – статус книги, определяющий, доступна ли данная книга к прочтению;
* Content – содержание книги;
* ImageUrl – физический путь к файлу обложки данной книги
* User – внешний ключ, определяющий связь «один ко многим» с таблицей «User»;
* Comments - внешний ключ, определяющий связь «один ко многим» с таблицей «Comments»;
* Ratings - внешний ключ, определяющий связь «один ко многим» с таблицей «Ratings»;
* BookTags - внешний ключ, определяющий связь «многие ко многим» с таблицей «Tags» через связующую таблицу «BookTags».

Таким же образом создаем остальные классы моделей. После этого необходимо создать класс контекста данных для взаимодействия всех классов моделей с БД. Для этого создаем новый класс «AppDbContext», приведенный в листинге 2:

Листинг 2 – Класс «AppDbContext»

public class AppDbContext : IdentityDbContext<User>

{

public AppDbContext(DbContextOptions<AppDbContext> options)

: base(options) { }

protected override void OnModelCreating(ModelBuilder builder)

{

base.OnModelCreating(builder);

}

public DbSet<User> User { get; set; }

public DbSet<Book> Books { get; set; }

public DbSet<Tag> Tags { get; set; }

public DbSet<BookTag> BookTags { get; set; }

public DbSet<Forum> Forum { get; set; }

public DbSet<Comment> Comments { get; set; }

public DbSet<Rating> Ratings { get; set; }

}

Данный класс наследует базовую реализацию от класса «IdentityDbContext<User>», определенной в конструкторе. Этот класс предоставляет реализацию шаблона проектирования ASP.NET Identity, который предоставляет разработчику встроенную систему авторизации и аутентификации. Затем мы создаем публичные свойства для каждой из наших моделей, причем тип моделей оборачиваем в коллекцию типа «DbSet». Класс «DbSet» предоставляет коллекцию всех сущностей в контексте для наших моделей. Таким образом, Entity Framework будет получать доступ к каждой из таблиц РБД через эти свойства.

Далее нам необходимо организовать подключения к базе данных. Для этого обратимся к файлу «appsettings.json», подавленном в каталог проект а системой при его создании. Добавим в него параметр, определяющий строку подключения к БД (листинг 3):

Листинг 3 – Определения строки подключения БД:

"ConnectionStrings": {

"DefaultConnection": "Server=(localdb)\\mssqllocaldb;Database=MyLib;Trusted\_Connection=True;MultipleActiveResultSets=True"},

На этапе разработки приложения наш сервер будет расположен локально на компьютере и будет обращаться к базе данных на облегченном движке SQL Server Express. Далее определяем конфигурацию нашего веб-сайта. Для этого переопределим в классе «Startup» метод «ConfigureServices» следующим образом (листинг 4):

Листинг 4 – Определение метода «ConfigureServices»:

public void ConfigureServices(IServiceCollection services)

{

services.AddDbContext<AppDbContext>(options =>

options.UseSqlServer(

Configuration.GetConnectionString("DefaultConnection")));

services.AddIdentity<User, IdentityRole>()

.AddEntityFrameworkStores<AppDbContext>()

.AddDefaultTokenProviders();

services.AddControllersWithViews(); }

Интерфейс «IServiceCollection» позволяет организовать механизм внедрения зависимостей (dependency injection), который позволяет организовать взаимодействие объектов между собой через абстракции, упрощая расширяемость и адаптируемость всей системы. Интерфейс имеет ряд готовых сервисов, которые можно использовать для настройки конфигурации проекта. Для организации подключения базы данных к проекту используем метод сервиса «AddDbContext». Для внедрения функционала ASP.NET Identity используем метод «AddIdentity». В конце добавим метод «AddControllersWithViews» для организации паттерна MVC в проект.

Теперь, когда все модели БД определены и конфигурация проекта полностью настроена, можно приступать к развертыванию БД на сервере. Сделать это можно при помощи миграций. Для создания миграций откроем консоль диспетчера пакетов NuGet. Затем выбираем проект, где хранятся созданные классы моделей, и создаем первую миграцию с помощью консольной команды «Add-Migration ""», где на месте кавычек задаем любое название миграции. После выполнения данной команды в проект автоматически добавляется папка с миграциями, а также класс «AppDbContextModelSnapshot», в котором сгенерирован код для создания БД на основе наших моделей. Чтобы запустить его на исполнение, воспользуемся командой «Update-Database». Начнется сборка проекта, и, если не возникнет никаких конфликтов или ошибок, то проект скомпилируется и все текущие миграции автоматически применятся. База данных будет создана. На рисунке 5 показан вывод консоли диспетчера пакетов:

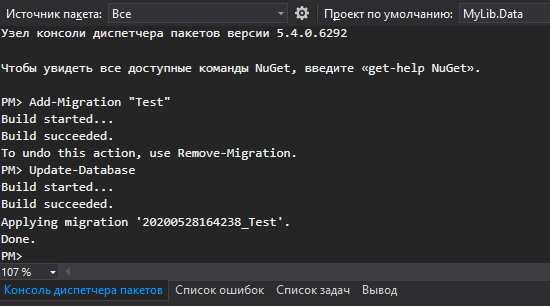


Рисунок 5 – Результат создания БД с помощью консольных команд

Система миграций удобна тем, что в любой момент времени программист сможет обновить структуру БД, просто изменив классы моделей и добавив новые миграции, либо же откатить проект к старым миграциям с помощью специальных консольных команд.

# **4 Взаимодействие пользователя с веб-сайтом**

**4.1 Авторизация на сайте**

Если пользователь хочет получить доступ к основным функциям сайта, таким как добавление собственных произведений в каталог или публикации тем на форуме, ему необходимо авторизоваться. Для этого необходимо нажать на кнопку «Войти», расположенной в правой части навигационной панели сайта. После этого он попадает на форму авторизации, представленную на рисунке 6:

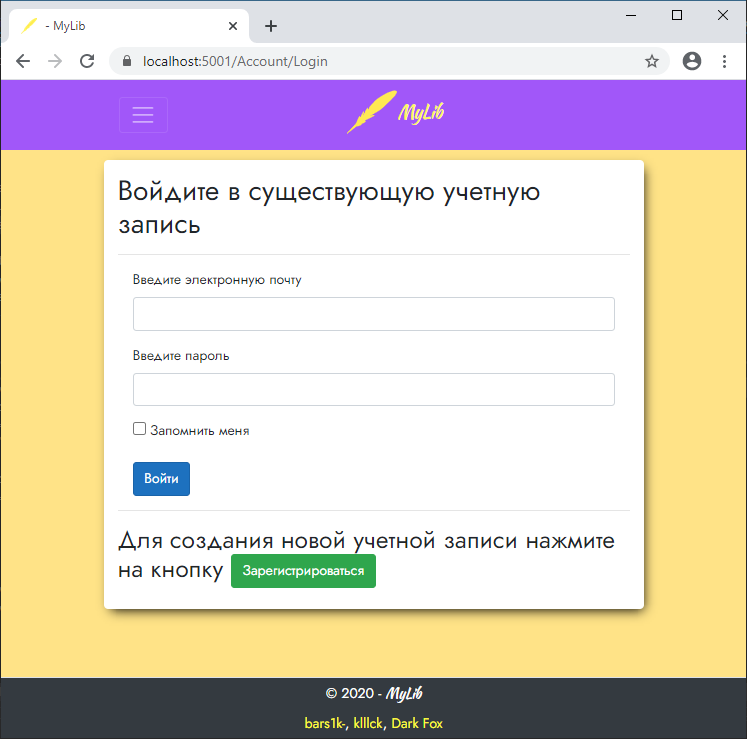


Рисунок 6 – Форма авторизации

В данной форме пользователю необходимо ввести свой адрес электронной почты и пароль. Если вы хотите оставаться в системе под данным аккаунтом и при повторном посещении сайта не вводить эти же данные вновь, то можете воспользоваться функцией «Запомнить меня». Если у вас еще нет учетной записи, то вы можете ее создать, нажав на кнопку «Зарегистрироваться». Вас переадресует на форму регистрации, представленной на рисунке 7:

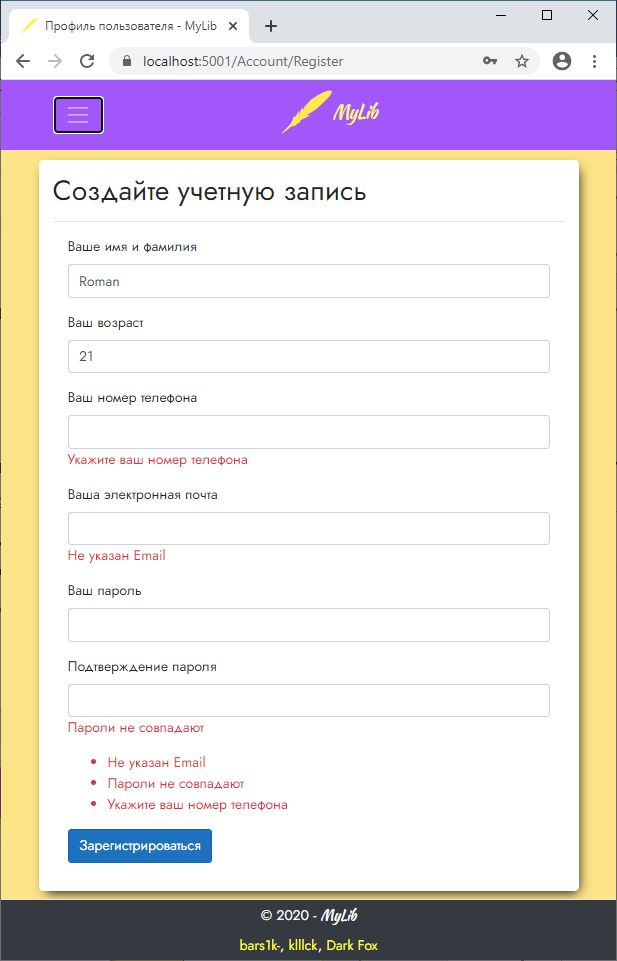


Рисунок 7 – Форма регистрации

Здесь необходимо пройти стандартную процедуру регистрации: ввести ваши имя и фамилию, возраст, номер телефона, адрес электронной почты, придумать и ввести пароль, удовлетворяющий требованиям сайта, и подтвердить его. Все эти данные являются обязательными при регистрации. Поэтому, если вы вдруг указали не валидные данные или вовсе забыли их указать, ничего страшного не произойдет. Система валидации укажет пользователю на все места с ошибками, как показано на рисунке 7. При успешной регистрации новая учетная запись пользователя добавляется в базу данных, после чего пользователь сможет войти на сайт. Если ему захочется изменить свои личные данные или, например, загрузить личную фотографию профиля, ему необходимо нажать на кнопку «Настройки профиля», расположенной в навигационной панели.

**4.2 Добавление новой книги на сайт**

Пользователь может опубликовать любую книгу на нашем сайте. Для этого необходимо перейти в раздел «Каталог», где расположенные все произведения, хранящиеся в базе данных сайта. При нажатии на кнопку «Добавить книгу», пользователь переходит на страницу с формой публикации новой книги, представленной на рисунке 8:

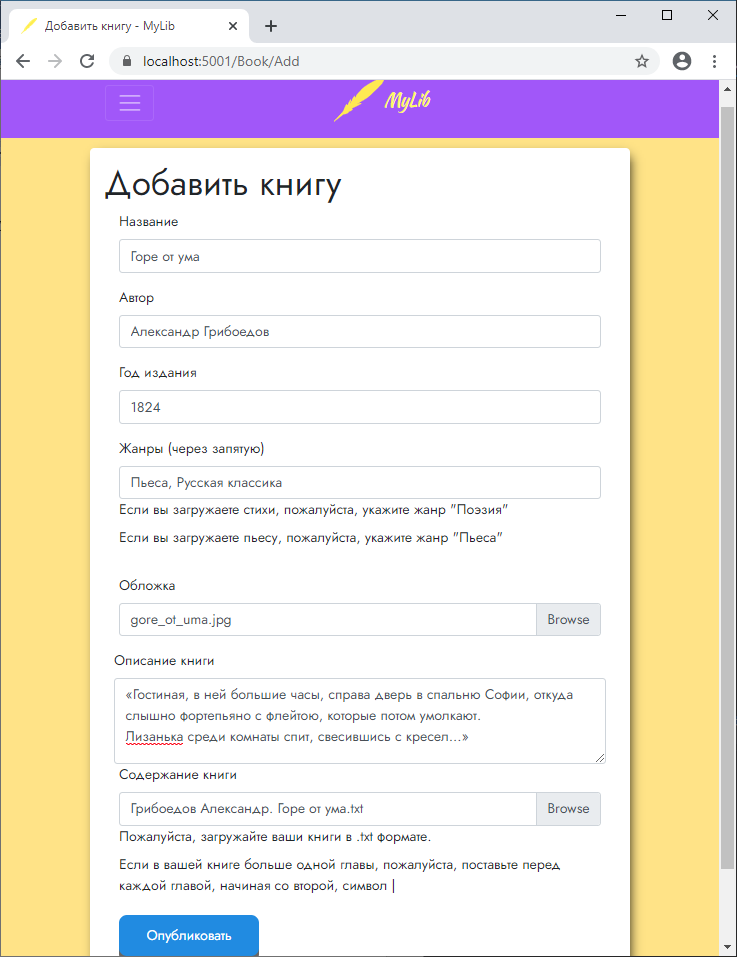


Рисунок 8 – Форма публикации новой книги

Как и на форме регистрации, пользователь должен ввести в текстовые поля все данные книги. Краткое описание книги является необязательным пунктом. При добавлении обложки необходимо нажать на кнопку «Browse», после чего откроется проводник. В нем необходимо выбрать файл изображения любого формата. В текстовом поле появится название файла, ссылка на расположение которого будет добавлена в БД. Таким же образом добавляем содержание книги, только вместо файла изображения выбираем файл формата .txt с текстом произведения. Нажимаем на кнопку «Опубликовать», после чего нас переадресовывают на страницу каталога, где в списке «Последние добавленные» будет располагаться именно эта книга (рисунок 9):

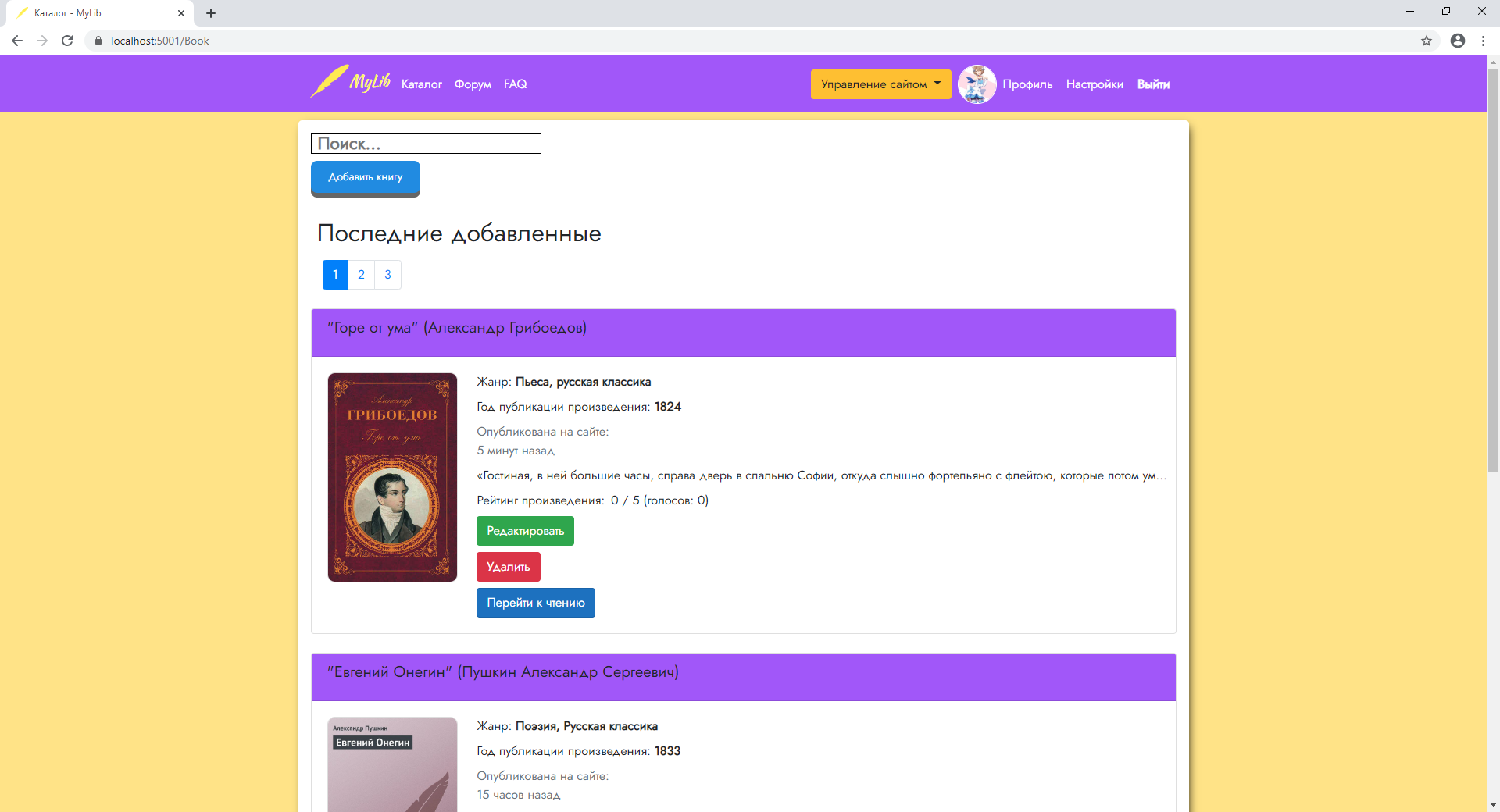


Рисунок 9 – Раздел «Последние добавленные»

# **Заключение**

В ходе выполнения данного курсового проекта была разработана база данных для информационной системе «Библиотека». Также под нее было разработано десктоп-приложение, позволяющие пользователю манипулировать с данными и выполнять различные действия с БД, такие как добавление, удаление или редактирование данных. По мимо этого пользователь мог выполнять различные запросы с выборками данных из разных таблиц и выводить результаты в форму для печати.

Для данной информационной системы также был разработан веб-сайт, эмулирующий структуры онлайн библиотеки. В нем есть возможность регистрации пользователей, которые могут найти нужную им книгу в каталоге библиотеки и взять ее под залог за определенную плату. После возвращения книги обратно в библиотеку, на счет их библиотечных карточек возвращалась исходная сумма. Пользователи также могут оставлять свои рецензии к книгам и общаться с другими пользователями в читательских клубах. Для внесения изменений в структуру данных сайта и поддержания контроля в базу данных внесены специальные администраторские учетные записи.

При разработке данной системы были изучены методы проектирования реляционных баз данных с помощью системы управления MS SQL Server. Также были получены новые знания в области разработки клиент-серверного онлайн-приложения с использованием технологии ASP.NET Core и объектно-ориентированного языка программирования C#.

При дальнейшем сопровождении и улучшении информационной системы библиотеки, разработанное ПО может превратиться из небольшого веб-сайта до полноценного веб-портала, в котором пользователи по всему миру смогут найти именно то произведения, которое им необходимо на данный момент времени.

**Список использованных источников**

1. Джеймс, Р. Грофф SQL. Полное руководство / Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. - М.: Вильямс, 2014. - 960 c.

2. Тейлор, Аллен SQL для чайников / Аллен Тейлор. - М.: Вильямс, 2014. - 416 c.

3. Дж. Рихтер CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C#. 4-е изд., 2015. – 869с.

4. Эндрю Троелсен. Язык программирования C# 7.0 и платформа .NET 4.0. 6-е изд., 2016. – 1312 с.

5. Адам, Фримен ASP.NET Core MVC с примерами на C# для профессионалов. Руководство / Фримен Адам. - М.: Диалектика / Вильямс, 2017. - 483 c.