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

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

по направлению: 09.03.04 Программная инженерия

(код и наименование направления)

на тему: «Разработка web-приложения «Литература». Backend-разработка. Комплексный проект»

(наименование темы)

ИКСИБ.КИСП.09.03.04.012.ПП

(обозначение документа)

Автор  / Р.В. Савицкий / (подпись, дата, расшифровка подписи)

Руководитель / д-р техн. наук, проф. В.Н. Марков / (подпись, дата, расшифровка подписи)

Консультанты:

Раздел информационной

безопасности проекта / д-р техн. наук., проф. Т.Г. Короткова / (подпись, дата, расшифровка подписи)

Нормоконтролер  / ст. преп. А.А. Ковтун/

(подпись, дата, расшифровка подписи)

Выпускная квалификационная работа  
допущена к защите

(дата)

Заведующая кафедрой / канд. техн. наук, доц. М.В. Янаева / (подпись)

Краснодар

2020Министерство 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

Заведующая кафедрой ИСП

канд. техн. наук, доц.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_М.В. Янаева

« 14 » мая 2019 г.

**ЗАДАНИЕ**

**НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ**

по направлению 09.03.04 Программная инженерия

(шифр и наименование)

студенту Савицкому Роману Владиславовичу

(фамилия, имя, отчество)

Тема выпускной квалификационной работы: «Разработка web-приложения «Литература». Backend-разработка. Комплексный проект»

Утверждена приказом ректора университета от 13.05.2020 г. № 650-Ст

Руководитель д-р техн. наук, проф. В.Н. Марков

(должность, фамилия, инициалы)

Консультанты:

Раздел информационной

безопасности проекта д-р техн. наук, проф. Т.Г. Короткова

(должность, фамилия, инициалы)

Нормоконтролер ст. преп. А.А. Ковтун

(должность, фамилия, инициалы)

Срок сдачи выпускной квалификационной работы на кафедру 23.06.2020 г.

**Содержание выпускной квалификационной работы**

Введение

1 Нормативные ссылки

2 Термины, определения и сокращения

3 Анализ предметной области

4 Постановка задачи и техническое задание

5 Обзор используемых технологий, сред разработки и паттернов программирования

6 Проектирование и создание базы данных

7 Разработка серверной части приложения

8 Взаимодействие пользователя с веб-сайтом

9 Безопасность и экологичность проекта

Заключение

Список использованных источников

Общее количество листов ПЗ 57

**Объем иллюстративной части**

1. Проектирование реляционной базы данных.

2. ER-диаграмма.

3. Серверная часть приложения.

4. Использование веб-приложения.

Общее количество слайдов иллюстративной части 10

**Список основной и рекомендуемой литературы**

1. Чамберс Дж., Пэкетт Д. Тиммс С. ASP.NET Core. Разработка приложений – Пер. с англ. – СПб : Питер, 2018 – 464 с.

**Календарный план выполнения выпускной квалификационной работы**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Месяц | Числа месяца | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | |
| Март |  | | | | | | | | | | | | | | | *Описание требований к ПП и постановка задачи ВКР* | | | | | | | *Анализ предметной области ВКР.*  *Техническое задание на ПП* | | | | | | | | | |
| Апрель | *Анализ предметной области ВКР.*  *Техническое задание на ПП* | | | | | *Обзор технологий*  *и сред программирования* | | | | | | | *Построение модели ПП. Проектирование БД* | | | | | | | | | | | | | | | | | | | |
| Май | *Реализация серверной части приложения* | | | | | | | | | | | | | | | | | | | | | | | *Безопасность и экологичность проекта* | | | | | | | | |
| Июнь | *Оформление пояснительной записки.* | | | | | | | | | | | | | | *Нормоконтроль, антиплагиат.*  *Сдача ВКР на кафедру* | | | | | | | | | *Защита ВКР* | | | | | | | |  |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_Р.В. Савицкий Руководитель\_\_\_\_\_\_ д-р техн. наук, проф. В.Н. Марков

(подпись, дата) (подпись, дата)

**Реферат**

Выпускная квалификационная работа содержит 57 страницы, 9 рисунков, 2 таблицы, 9 листингов, 7 источников.

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

Целью выпускной квалификационной работы является разработка серверной части веб-приложения «Литература».

В результате выполнения выпускной квалификационной работы разработан веб-сайт, на котором пользователи могут читать произведения различных авторов, опубликовывать собственные произведения и делиться ими с другими пользователями. Также на сайте разработан веб-форум, где пользователи могут общаться друг с другом и делиться своими мнениями по прочитанным книгам.

Программное обеспечение создано на платформе ASP.NET Core. Приложение написано на языке программирования C# в среде разработки Visual Studio. Для создания базы данных используется СУБД MS SQL Server.

Содержание

[Введение 8](#_Toc42721091)

[1 Нормативные ссылки 9](#_Toc42721092)

[2 Термины, их сокращения и определения 10](#_Toc42721093)

[3 Анализ предметной области 12](#_Toc42721094)

[4 Постановка задачи и техническое задание 14](#_Toc42721095)

[4.1 Требования к системе 14](#_Toc42721096)

[4.2 Требования к архитектуре 14](#_Toc42721097)

[4.3 Требования для пользователей 15](#_Toc42721098)

[5 Обзор используемых технологий, сред разработки и паттернов программирования 18](#_Toc42721099)

[5.1 Язык программирования C# 18](#_Toc42721100)

[5.2 Visual Studio 2019 19](#_Toc42721101)

[5.3 ASP.NET Core 3.1 20](#_Toc42721102)

[5.4 Паттерн MVC 21](#_Toc42721103)

[5.5 MS SQL Server 2019 22](#_Toc42721104)

[6 Проектирование и создание базы данных 24](#_Toc42721105)

[6.1 Описание сущностей 24](#_Toc42721106)

[6.2 Проектирование концептуальной и логической диаграмм 25](#_Toc42721107)

[6.3 Проектирование ER-диаграммы 27](#_Toc42721108)

[6.4 Создание реляционной базы данных 29](#_Toc42721109)

[7 Разработка серверной части приложения 35](#_Toc42721110)

[7.1 Сервисы 35](#_Toc42721111)

[7.2 Контроллеры 39](#_Toc42721112)

[7.3 Валидация данных 41](#_Toc42721113)

[8 Взаимодействие пользователя с веб-сайтом 44](#_Toc42721114)

[8.1 Авторизация на сайте 44](#_Toc42721115)

[8.2 Добавление новой книги на сайт 46](#_Toc42721116)

[9 Оценка положительного эффекта 49](#_Toc42721117)

[10 Безопасность и экологичность проекта 50](#_Toc42721118)

[10.1 Значение и задачи безопасности жизнедеятельности 50](#_Toc42721119)

[10.2 Анализ условий труда и мероприятия по защите от воздействия вредных производственных факторов 51](#_Toc42721120)

[10.3 Обеспечение электробезопасности 54](#_Toc42721121)

[10.4 Обеспечение пожарной безопасности 54](#_Toc42721122)

[Заключение 56](#_Toc42721123)

[Список использованных источников 57](#_Toc42721124)

# Введение

В наше время Интернет является первым местом, куда человек обращается за получением необходимой ему информации. Процесс общения и взаимодействия между людьми также зачастую происходит через глобальную сеть. Тенденция развития веб-технологий продолжает расти с каждым дней. Это приводит к возникновению новых возможностей как и в IT-индустрии, так и в повседневной жизни.

Несмотря на это, книги все также являются неотъемлемой частью человеческой культуры. Они служат основный источником вдохновения, приобретения новых идей и знаний. Однако главная особенности книги заключается в том, что через нее протекает связь человечества с историей.

Целью выпускной квалификационной работы являлась разработка веб-приложения «Литература». Используя данное приложение, пользователь сможет найти всю информацию об интересующей его книги, оставлять свои рецензии и мысли по поводу прочитанного произведения, а также публиковать собственные. Также в приложении есть веб-форум, на котором пользователи могут добавлять свои темы для обсуждения любых новостей или простого общения друг с другом.

В рамках данной работы необходимо реализовать серверную часть веб-приложения, которая позволяет осуществить взаимодействие между пользователями через сеть Интернет, а также организовать логику для работы с данными, используемые информационной системой.

Выполнение ВКР поспособствует повышению квалификации в области программирования и разработки серверной части веб-приложений, а также приобретению новых знания в области проектирования баз данных.

# 1 Нормативные ссылки

В настоящей выпускной квалификационной работе использованы ссылки на следующие нормативные документы:

1. ГОСТ Р 1.12-2004. Стандартизация в Российской Федерации;

2. ГОСТ 20886-85 Организация данных в системах обработки данных. Термины и определения.

3. ГОСТ 34.321-96 Информационные технологии. Система стандартов по базам данных. Эталонная модель управления данными.

4. ГОСТ 15971-90 Системы обработки информации. Термины и определения.

5. СанПиН 2.2.2/2.4.1340-03 Гигиенические требования к персональным электронно-вычислительным машинам и организации работы.

6. Трудовой кодекс Российской Федерации от 30.12.2001 № 197-ФЗ (ред. от 10.01.2016).

7. ТОИ Р-45-084-01 Типовая инструкция по охране труда при работе на персональном компьютере.

8. ГОСТ 12.0.003-2015 Система стандартов безопасности труда. Опасные и вредные производственные факторы. Классификация.

9. ГОСТ Р 2.2.2006-05 Руководство по гигиенической оценке факторов рабочей среды и трудового процесса. Критерии и классификация условий труда.

10. ГОСТ 12.1.019-2017 Электробезопасность.

11. Технический регламент о требованиях пожарной безопасности от 22.07.2008 №123-ФЗ (ред. от 29.07.2017).

# 2 Термины, их сокращения и определения

В данной части ВКР используются термины в соответствии с их определениями и сокращениями, которые регламентированы следующими нормативными документами:

– ГОСТ Р 1.12-2004. Стандартизация в Российской Федерации;

– ГОСТ 34.321-96 Информационные технологии. Система стандартов по базам данных. Эталонная модель управления данными;

– ГОСТ 33707-2016 (ISO/IEC 2382:2015) Информационные технологии (ИТ). Словарь;

– ГОСТ 19.101-77 Единая система программной документации (ЕСПД). Виды программ и программных документов

– ГОСТ 15971-90 Системы обработки информации. Термины и определения;

– ГОСТ 20886-85 Организация данных в системах обработки данных.

Термины, их сокращения и определения:

**1 Информация –** осознанныесведения об окружающем мире

**2 Данные** – представление информации в формальном виде, подходящем для хранения, обработки и передачи.

**3 База данных (БД)** – именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области.

**4 Реляционная база данных (РБД)** – набор данных с переопределенными связями между ними.

**5 Система управления базами данных (СУБД)** – совокупность языковых и программных средств, предназначенных для создания, управления, сопровождения и совместного использования БД многими пользователями.

**6 Программное обеспечение (ПО**) – программа или множество программ, используемых для управления компьютеров.

**7 Язык программирования (ЯП)** – формальный язык, предназначенный для записи компьютерных программ.

**8 Объектно-ориентированное программирование (ООП) –** методология программирования, основанная на представлении программы в виде совокупности объектов.

**9 Операционная система (ОС) –** совокупность программ, организующих управление аппаратными средствами компьютера.

**10 Structured Query Language (SQL)** – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

**11 Object-Relational Mapping (ORM)** – технология, связывающая процесс разработки базы данных с объектно-ориентированным подходом программирования.

**12 Uniform Resource Locator (URL)** – система унифицированных адресов электронных ресурсов, или единообразный определитель местонахождения ресурса.

**13 HyperText Transfer Protocol (HTTP)** – протокол прикладного уровня, используется передачи произвольных данных.

**14 Integrated Development Environment (IDE)** – система программных средств, используемая программистами для разработки программного обеспечения.

**15 Фреймворк** – платформа, определяющая структуру программной системы, объединяет воедино различные компоненты программного проекта и упрощает процесс разработки.

# 3 Анализ предметной области

Основным объектом исследования при разработке серверной части веб-приложения является понятие данных. Все данные, используемые в приложении, должны быть организованны. Процесс организации достигается за счет разбиения данных по группам, связанных между собой по смыслу, со схожими параметрами и характеристиками. Между получившимися абстрактными группами данных выстраиваются связи, вследствие чего проектируется модель БД.

При работе с данными в приложении важнейшим пунктом является соблюдение условия конфиденциальности и сохранности. Пользователь должен быть защищен от возможности утечки своих данных и непреднамеренного доступа к чужим. Также система не должна содержать данные в неправильной форме, которые каким-либо образом приводят пользователя в состояния заблуждения и непонимания, а также к появлению ошибок в системе. Поэтому важным этапом является разработка безотказной системы валидации данных.

Также серверная часть приложения позволяет организовать все возможные манипуляции с данными, такие как:

– добавление;

– удаление;

– изменение;

– передача;

– чтение;

– поиск;

– выборка и другие.

Под backend-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-протоколы и URL-адреса. [5]

# 4 Постановка задачи и техническое задание

Для понимания сути проектирования и разработки ПО необходимо правильно сформулировать постановку задачи. Она содержит формулировку технического задания на основе входной и выходной информации системы.

Техническое задание содержит необходимые требования к программному продукту, а также выбор технологий и средств программирования для разработки серверной части веб-приложения.

4.1 Требования к системе

Разработка серверной части веб-приложения «Литература» должна удовлетворять следующим требованиям:

– разработка методов и алгоритмов для работы с данными, которые хранятся на сервере;

– доступ ко всем алгоритмам, описывающие логику работы веб-сайта должен быть ограничен для всех пользователей, не являющихся разработчиками;

– любые данные, поступающие от пользователей на сервер и обратно, должны проходить систему валидации на серверной части;

– обработка пользовательских запросов;

– отправление ответов на клиентскую часть приложения;

– высокая скорость работы системы;

– стабильная производительность и устойчивость к сбоям.

4.2 Требования к архитектуре

Архитектура веб-приложения имеет следующие требования:

– основной платформой для разработки серверной части приложения является фреймворк ASP.NET Core 3.1;

– в качестве локального веб-сервера приложения используется ПО ISS;

– для удаленного развертывания приложения применяется веб-сервер на базе облачной технологии Azure;

– для хранения данных системы используется реляционная СУБД MS SQL Server 2019;

– для написания серверной части веб-приложения используется язык программирования C# 8.0;

– в основе разработки структуры приложения используется архитектурный паттерн программирования MVC.

Дополнительные технологии и паттерны, используемы в процессе разработки:

– утилита SQL Server Management Studio;

– ORM технология Entity Framework Core;

– система аутентификации ASP.NET Identity;

– паттерн Dependency Injection.

4.3 Требования для пользователей

Пользователи веб-приложения «Литература» разграничены по ролям. Каждая роль обладает своим набором функций и требований. Основными являются следующие роли: «Пользователь», «Гость» и «Администратор».

«Гость» – это условная роль, обозначающая пользователя, который не зарегистрирован в системе. Для нее определены следующие функции:

– ознакомление с функционалом веб-сайта;

– навигация по веб-сайту;

– просмотр каталога книг;

– поиск книги;

– чтение книги;

– просмотр тем на форуме;

– просмотр комментариев других пользователей;

– регистрация на веб-сайте.

«Пользователей» –  это роль для тех пользователей, которые прошли регистрацию на сайте. Их персональные данные сохраняются в БД. Они содержат все функции, что и пользователи с ролью «Гость», а также ряд следующих функций:

– добавление новой книги на сайт с возможностью ее удаления или изменения содержимого в данной книги;

– добавление новой темы на форуме с возможностью ее удаления или изменения содержимого в данной теме;

– добавление комментария к теме на форме с возможностью его удаления или изменения содержимого в данном комментарии;

– оценка прочитанной книги;

– оценка темы на форуме;

– отправка жалобы на другого пользователя;

– редактирование или удаление собственного профиля.

Пользователь с ролью «Администратор» обладает возможностью использования функционала для работы с данными на сайте. Он также должен следить за деятельностью других пользователей. Для него определены следующие функции:

– изменение или удаление любой книги на сайте;

– изменение или удаление любой темы на форуме;

– изменение или удаление любой пользовательского комментария;

– добавление и изменение набора тегов для книг;

– рассмотрение всех жалоб и предложений от других пользователей;

– блокировка или восстановление аккаунта пользователя;

– управление системой кластеризации книг.

В данном приложении также присутствуют дополнительные роли: «СуперАдминистратор», «Флудер» и «Заблокированный».

Пользователь с ролью «СуперАдминистратор» обладает всеми функциями, что и «Администратор», однако у него есть доступ к системе управления ролями пользователей. Ее функции следующие:

– добавление новых ролей;

– изменение или удаление существующих ролей;

– добавление новой роли пользователю;

– удаление роли у пользователя.

Роль «Флудер» назначаются тем пользователям, которые плохо ведут себя на форуме. На их функции накладываются следующие ограничения:

– ограничение возможности создавать новые темы на форуме.

– ограничение возможности комментировать на форуме;

Роль «Заблокированный» достается тем пользователям, которые нарушали правила сайта, а также получившие множество жалоб от других пользователей или замечаний от администрации сайта. Для данной роли блокируется весь пользовательский функционал.

Роли «Флудер» и «Заблокированный» могут выдавать администраторы сайта.

# 5 Обзор используемых технологий, сред разработки и паттернов программирования

Первым делом необходимо определиться со средой разработки и набором технологий, которые будут использоваться для написания ПО. На основе анализа предметной области и технического задания, для основной платформы разработки был выбран фреймворк ASP.NET Core 3.1 в связке со средой разработки Visual Studio 2019. Учитывая это, в качестве основного инструмента для написания кода приложения выбран язык программирования C#. Для проектирование структуры приложения будет использоваться паттерн MVC. В качестве СУБД был выбран MS SQL Server 2019.

5.1 Язык программирования C#

C# является объектно-ориентированным ЯП высокого уровня, разработанный компанией Microsoft в 1998 году. Несмотря на его достаточно небольшой возраст, C# уже добился больших успехов в ИТ-индустрии. На данном языке можно создавать любые типы приложений: от небольших консольных программ до масштабных веб-приложений или компьютерных видеоигр.

Язык C# содержит в себе реализацию всех принципов ООП:

– инкапсуляция;

– наследование;

– полиморфизм;

– абстрагирование;

– иерархичность;

– параллелизм;

– сохраняемость.

Он также поддерживает статическую типизацию данных и перегрузку операторов. Все это позволяет языку C# создавать крупные и сложные, и в тоже время гибкие и легко расширяемые приложения.

Главным же преимуществом данного ЯП является то, что он был специально разработан для работы с платформой .NET. Благодаря этому, программисту, использующего язык C#, открываются следующие возможности:

– объемная встроенная библиотека классов;

– большой стек различных технологий;

– кроссплатформенность;

– возможность использовать другие ЯП, параллельно с языком C#.

Язык C# с каждым годом активно развивается, приобретая ряд дополнительных интересных функций, увеличивающие его потенциал, а также изменений, приводящих к исправлению предыдущих ошибок и недостатков.

5.2 Visual Studio 2019

Visual Studio – это IDE, которая является основной средой разработки приложений любого типа с использованием ЯП C#. Для разработки веб-приложения «Литература» была выбрана бесплатная общеобразовательная версия Visual Studio Community. Она обладает всеми необходимыми инструментами для быстрого и эффективного написания программного кода:

– удобный и продвинутый интерфейс приложения;

– технология IntelliSense значительно повышает производительность труда программиста за счет функций автозаполнения кода и рекомендаций по его улучшению и оптимизации;

– темп работы повышается за счет функций быстрого перемещения по проекту;

– среда разработки сама указывает на все синтаксические и семантические ошибки в коде проекта, описывает причины возникновения этих ошибок, а также предлагает помощь по их устранению;

– поддержка большинства современных языков программирования и языков разметки;

– имеются решения для совместной разработки проекта и его сопровождения;

– универсальное средство для работы с библиотеками и пакетами NuGet;

– набор средств для проведения отладки и автоматического тестирования проекта;

– средства для компиляции решения и его развертывание в облачной технологии Microsoft Azure.

5.3 ASP.NET Core 3.1

ASP.NE Core – платформа для создания различных веб-приложений от компании Microsoft. Данный фреймворк был выбран в качестве основного средства разработки ПО 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.

5.4 Паттерн MVC

Данный архитектурный 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 5.1):

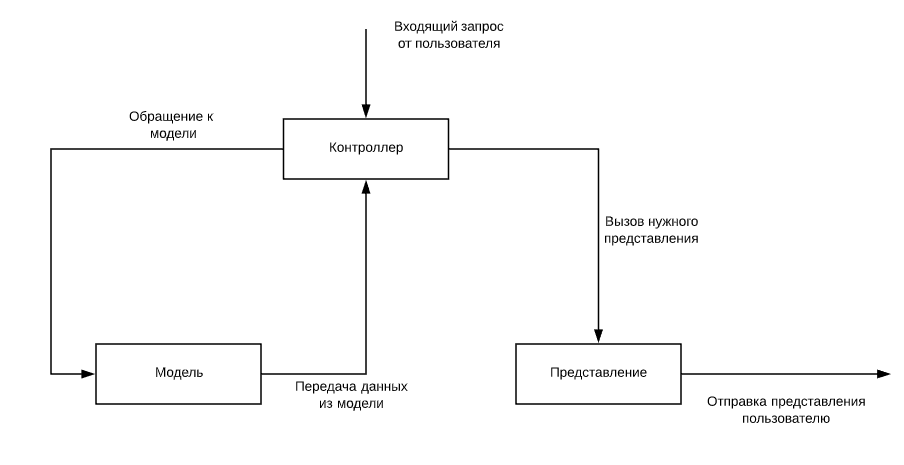


Рисунок 5.1 – Визуальная модель паттерна MVC

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 на клиентской стороне.

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

5.5 MS SQL Server 2019

MS SQL Server – это реляционной СУБД, является одной из самых распространенных систем управления базами данных в мире, наряду с Oracle SQL. Характеризуется следующими качествами:

– высокая производительность;

– надежность;

– безопасность;

– простота в использовании.

Зачастую используется в совокупности с платформой .NET. На протяжении долгого времени была доступна только для ОС Windows, однако с недавного времени ее можно развернуть и на ОС Linux.

MS SQL Server использует классическую структуру реляционной модели данных. Для взаимодействия с БД применяется язык SQL. Для взаимодействия и управления базами данных применяется среда проектирования SQL Server Management Studio. Для разработкой и сопровождения БД также можно использовать среду Visual Studio. Таким образом мы имеем следующие возможности для работы с данной СУБД:

– через запросы на языке SQL;

– с помощью утилиты SSMS;

– с использованием библиотек платформы .NET.

# 6 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 СУБД MS SQL Server. Общая структура РБД будет представлена в виде ER-диаграммы.

6.1 Описание сущностей

После этапов анализа предметной области и разбора технического задания можно приступить к проектированию и реализации БД. Определив требования к информационной системе, 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.

«Жалобы» – пользователи сайта могут отправлять жалобы друг другу. Жалоба содержит код причины с ее пояснением в виде текстового сообщения. Все жалобы должны обрабатываться администрацией сайта.

6.2 Проектирование концептуальной и логической диаграмм

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

Таблица 6.1 – Типы связей между сущностями

|  |  |  |
| --- | --- | --- |
| **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 |

*Окончание таблицы 6.1*

|  |  |  |
| --- | --- | --- |
| Книги | 1:N | Оценки |
| Форум | 1:N | Комментарии |
| Форум | 1:N | Оценки |
| Комментарии | 1:N | Оценки |
| Жалобы | 1:N | Пользователи |
| Жалобы | 1:N | Комментарии |
| Роли пользователей | N:N | Пользователи |
| Теги | N:N | Книги |

После определения всех типов связей между сущностями системы, реализуем концептуальную диаграмму, показанную на рисунке 6.1.

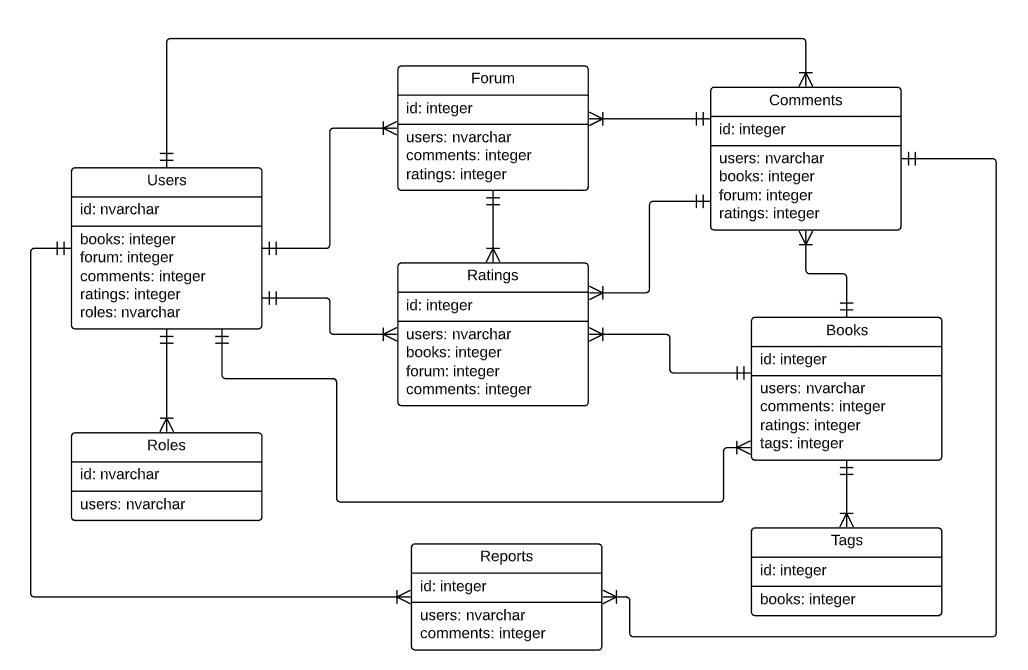


Рисунок 6.1 – Концептуальная диаграмма БД

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» и «Теги», а также «Пользователи» и «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 6.2):

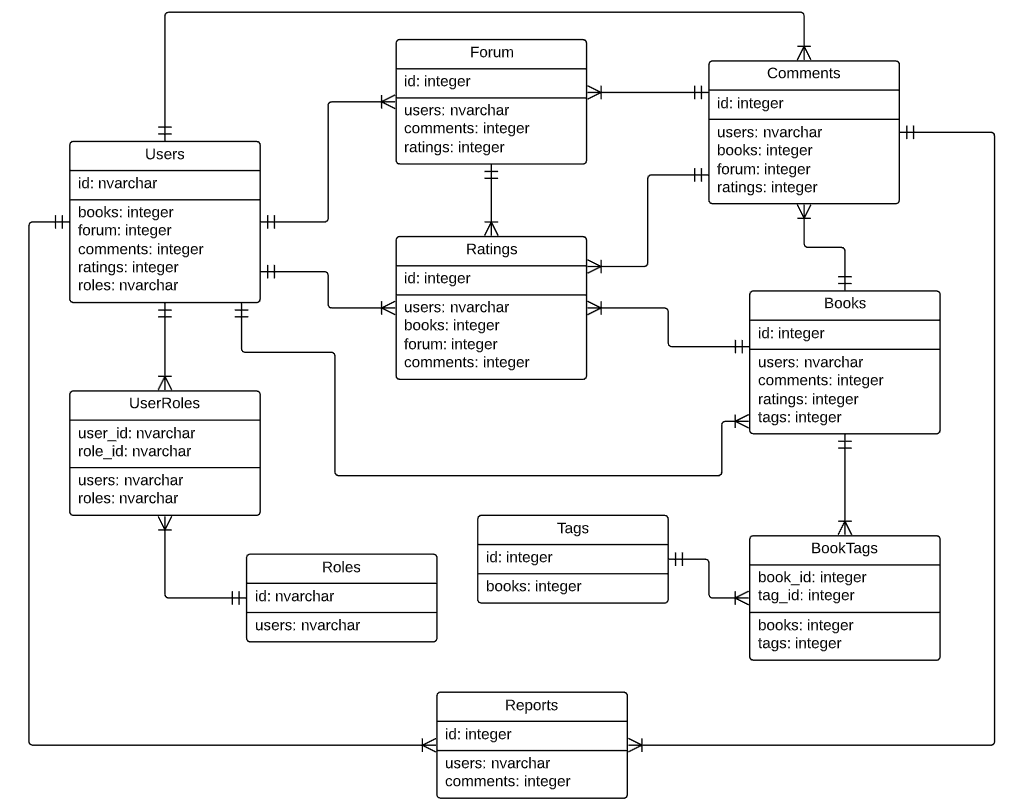


Рисунок 6.2 – Логическая диаграмма БД

6.3 Проектирование ER-диаграммы

На основании спроектированной логической модели начинается этап разработки таблиц для будущей реляционной БД. 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 с их типами данных, а также типами отношений между всеми таблицами. Полученная ER-диаграмма представлена на рисунке 6.3:

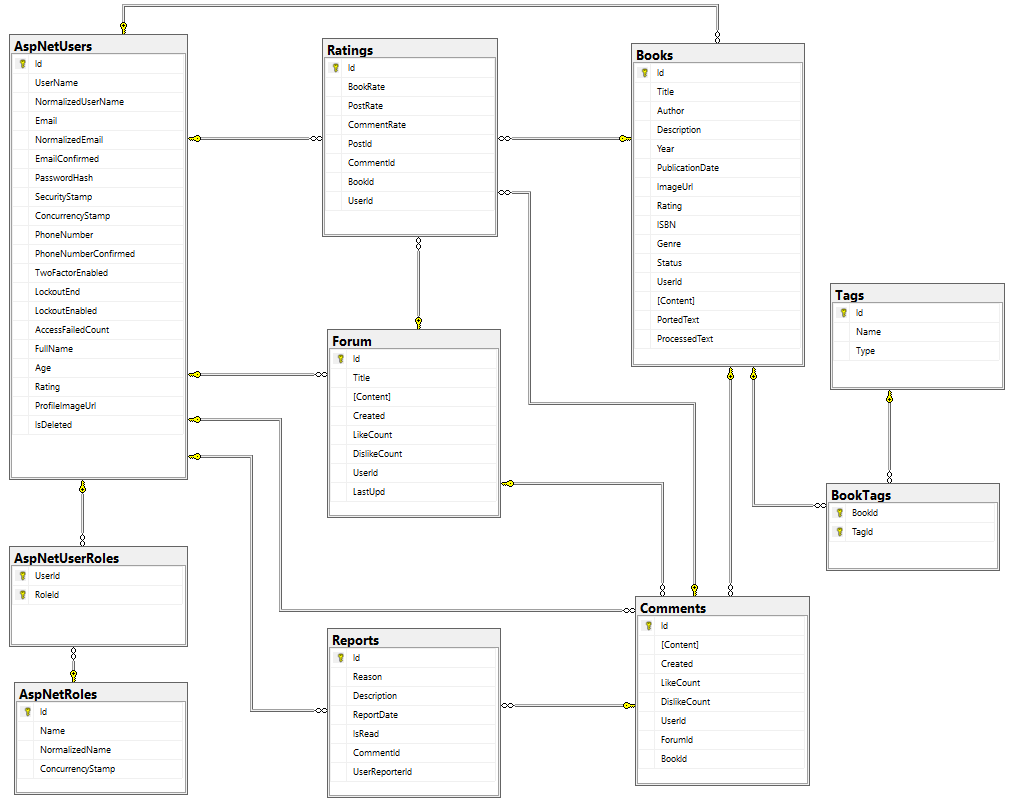


Рисунок 6.3 – ER-диаграмма

6.4 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 разрабатываем приложение, используя ООП подход, гораздо лучше и эффективнее использовать фреймворк Entity Framework Core.

Entity Framework Core – это ORM технология, позволяющая работать с РБД на абстрактном уровне, т.е. абстрагироваться от физического представления БД и ее таблиц. [7] 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, Entity Framework Core самостоятельно реализует БД на физическом уровне.

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 6.1):

Листинг 6.1 – 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; }

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 6.2:

Листинг 6.2 – 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 6.3):

Листинг 6.3 – 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 6.4):

Листинг 6.4 – 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. [6] 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 6.4 показан вывод консоли диспетчера пакетов:

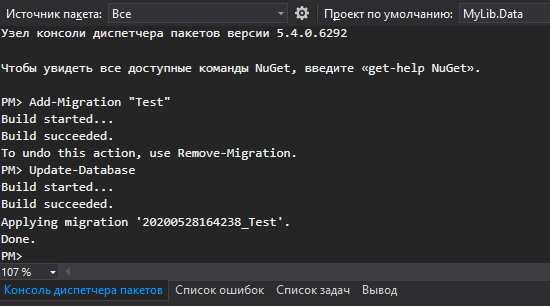


Рисунок 6.4 – Результат создания БД с помощью консольных команд

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.

# 7 Разработка серверной части приложения

Закончив с проектированием РБД, необходимо начать разработку серверной части приложения. Для этого создадим классы сервисов и контроллеров. Сервисы – это классы программы, определяющие логику работы с данными в приложении. Они отвечают за добавление, удаление, изменение, выборки и другие операции с данными в БД. Контроллеры играют роль «связующего» между моделями и представлениями. Считывая запросы от пользователей, они формируют обращаются к методам сервисов. Получив результаты обработки данных, контроллеры формируют ответ, который посылают пользователю, вызывая нужное представление. Если результатом является ошибка, контроллер отправить пользователю сообщения об этой ошибке. Вся серверная часть написана на языке C#. При разработке используются стандартные системные библиотеки, а также библиотеки фреймворков .NET Core и Entity Framework Core.

7.1 Сервисы

В качестве примера рассмотрим структуру сервиса для работы с книгами. Сначала создаем интерфейс «IBook», определяющий следующий набор методов (листинг 7.1):

Листинг 7.1 Интерфейс «IBook»

public interface IBook

{

Task Add(Book newBook);

Task Delete(int id);

Task DeleteAll();

Task Edit(Book book);

Task AddReview(Comment comment);

Book GetById(int id);

IEnumerable<Book> GetAll();

IEnumerable<Book> GetFilteredBooks(string searchQuery);

IEnumerable<Book> GetLatestBooks(int n);

IEnumerable<Book> GetBookWithTags(IEnumerable<Book> books, string searchQuery);

}

Для реализации интерфейса «IBook» создадим класс «BookService». Добавим приватное поле «context» и инициализируем его в конструкторе. Через это поле будет проходить обращения к контексту базы данных, определенного в классе «AppDbContext». Метод «GetAll» возвращает все объекты книг из БД через поле контекста, которое обращение к свойству «Books». Также необходимо получить доступ ко данным из всех таблиц, на которые ссылается текущая. Для этого используем метод «Include», передавая в него навигационное свойство связующей таблицы.

Теперь, когда у нас есть все объекты книг, мы можем достать нужную нам книгу через ее уникальный номер. Эта логика определена в методе «GetById». Используя метод «FirstOrDefault» библиотеки LINQ, найдем книгу с Id равным текущему. В результате мы либо возвращаем объект книги, либо null, если книги с данным Id не была найдена.

Пользователь может найти нужную ему книгу, написав ее название или имя автора в строку поиска. Метод «GetFilteredBooks» принимает эту строку. Сначала идет проверка на то, содержит ли данная строка какие-либо символы или она пустая. Если строка пустая, то возвращаем весь список книг. С помощью метода «ToLower» получаем копию данной строки, в которой все символы переведены в нижний регистр. Это упрощает дальнейшую работу алгоритма. Вызываем свойства названия и имени автора для каждой книги из всего списка книг с использованием метода «Contains». В данный метод определят, встречается ли указанная подстрока внутри данной строки. В качестве строки выступает название книги или имя ее автора, а в качестве подстроки – поисковая строка из запроса. Если встречается, то возвращаем объект этой книги в коллекцию, а если нет, то пропускаем его. В итоге метод «GetFilteredBooks» вернет коллекцию с теми книгами, которые прошли проверку.

Для операций добавления, удаления и изменения книг в БД определены следующие три метода: «Add», «Edit» и «Delete». Их реализация очень схожа между друг другом. Сначала нужно обратиться к контексту БД, вызвать оттуда коллекцию книг и применить соответствующий метод из библиотеки EF Core. Затем необходимо вызвать метод «SaveChangesAsync», который сохраняет все изменения, вызванные в БД. «Асинхронное сохранение позволяет избежать блокировки потока при записи изменений в базу данных. Эта функция помогает предотвратить замораживание пользовательского интерфейса многофункционального клиентского приложения». [6] Реализация описанного выше сервиса представлена в листинге 7.2:

Листинг 7.2 Класс «BookService»

public class BookService : IBook

{

private readonly AppDbContext \_context;

public BookService(AppDbContext context)

{

\_context = context;

}

public IEnumerable<Book> GetAll()

{

return \_context.Books

.Include(b => b.User)

.Include(b => b.Comments)

.Include(b => b.Ratings)

.Include(b => b.BookTags).ThenInclude(bt => bt.Tag);

}

public Book GetById(int id)

{

return GetAll().FirstOrDefault(b => b.Id == id);

}

public IEnumerable<Book> GetLatestBooks(int n)

{

return GetAll()

.OrderByDescending(b => b.PublicationDate).Take(n);

}

public IEnumerable<Book> GetFilteredBooks(string searchQuery)

{

if (!string.IsNullOrEmpty(searchQuery))

{

var query = searchQuery.ToLower();

return GetAll()

.Where(book =>

book.Title.ToLower().Contains(query)

|| book.Author.ToLower().Contains(query);

}

else return GetAll();

}

public async Task Add(Book newBook)

{

\_context.Books.Add(newBook);

await \_context.SaveChangesAsync();

}

public async Task Delete(int id)

{

var book = GetById(id);

var rates = \_context.Ratings

.Where(r => r.Book.Id == book.Id);

foreach (var rate in rates)

\_context.Ratings.Remove(rate);

\_context.Books.Remove(book);

await \_context.SaveChangesAsync();

}

public async Task Edit(Book book)

{

\_context.Books.Update(book);

await \_context.SaveChangesAsync();

}

7.2 Контроллеры

Класс «BookController» содержит логику по обработки запросов пользователей, связанных с получением любой информации, относящийся ко всем доступным книгам на сайте. К примеру, если пользователь хочет посмотреть вес список книг, он обращается к серверу через GET-запрос. Этот запрос обрабатывается в методе «Index», реализация которого представлена в листинге 7.3:

Листинг 7.3 Метод «Index» класса «BookController»

public IActionResult Index(string searchQuery, int page = 1)

{

int pageSize = 3;

var books = \_bookService.GetFilteredBooks(searchQuery).OrderByDescending(x => x.PublicationDate).ToList();

var noResults = !string.IsNullOrEmpty(searchQuery) && !books.Any();

var booksOnPage = books

.Select(book => new BookViewModel

{

BookId = book.Id,

Title = book.Title,

ImageUrl = book.ImageUrl,

Author = book.Author,

Genre = book.Genre,

Rating = book.Rating,

PublicationDate = book.PublicationDate,

UserId = book.User.Id,

Year = book.Year,

Description = book.Description

}).Skip((page - 1)\*pageSize).Take(pageSize).ToList();

PageViewModel pageViewModel = new PageViewModel(books.Count(), page, pageSize);

var model = new BookIndexModel()

{

PageViewModel = pageViewModel,

Books = booksOnPage

};

return View(model);

}

Через поле «bookService», представляющее объект класса «BookService», мы обращаемся к методам сервиса «BookService», реализация которого описана ранее в работе. Получим весь набор книг с сервера, после чего преобразуем его в список, упорядоченный по дате публикации книги на сайте. Далее необходимо создать специальную модель для передачи всех полученных данных с сервера на представление. Для этого в цикле создаем новые объекты класса «BookViewModel». Определяем свойства этих объектов, передавая в них значения из списка книг, полученных с сервера. В итоге передаем полученный список обратно на представление с помощью метода «View», передав управление клиентской части.

Так как на сервере хранится большое количество книг, их представление должно быть разбито на некоторое количество пронумерованных страниц, каждая из которых содержит фиксированный набор этих книг. Логика разбиения всего массива книг на части также определена на сервере. Переменная «pageSize» определяет, какое количество книг будет размещено на странице сайта. Номер текущей странице передается в метод «Index» в виде переменной «page». Если пользователь не указывает номер страницы явным образов, то по умолчанию задается первая страница. Затем из всего списка книг пропускаем нужное количество через метод «Skip», пока не достигнем нужной. Затем, с помощью метода «Take», берем то количество книг, которое заданно в переменной «pageSize». Возвращаем полученный список в представление, также возвращаем объект класса «PageViewModel», который определяет набор параметров для пагинации, механизм которой реализуется на клиентской части приложения.

7.3 Валидация данных

Валидация данных играет одну из ключевых ролей при разработке серверной части веб-приложения. Она позволяет проверять на корректность данные, отправленные на сервер с клиентской стороны, а также обрабатывать эти данные. Рассмотрим механизм валидации на примере добавления новой книги на сервер.

Для каждого свойства модели представления книги, определенной в классе «BookViewModel», добавляем атрибут «Required». Он означает, что свойство с этим атрибутам не может принимать значение null, т.е. должно быть задано каким-либо образом. Также в этом атрибуте можно определить свойство «ErrorMessage». Этому свойству присваивается строка, которая передается на представление и указывает пользователю, какую именно ошибку он допустил при заполнении формы добавления книги. Также можно использовать и другие атрибуты валидации. Например, атрибут «Range» позволяет задать интервал для целочисленного значения, а атрибут «StringLength» задает длину строки. Пример использования атрибутов валидации представлен в листинге 7.4:

Листинг 7.4 – Атрибуты валидации

[Required(ErrorMessage = "Введите название книги")]

[StringLength(50, ErrorMessage = "Длина названия книги не должна превышать 50 символов!")]

public string Title { get; set; }

[Required(ErrorMessage = "Введите название автора книги")]

[StringLength(30, ErrorMessage = "Длина имени автора книги не должна превышать 30 символов!")]

public string Author { get; set; }

[Required(ErrorMessage = "Введите год издания книги")]

[Range(1, 2020, ErrorMessage = "Недопустимое значение. Введите точный год издания книги")]

public int Year { get; set; }

Валидация на стороне сервера определяется в контроллере на основе класса «ModelStateDictionary». При проверке модели данных, переданной из представления контроллеру, свойство «IsValid» вернет значение true, если все атрибуты модели удовлетворяют требования атрибутов валидации, иначе возвращается false. Однако возникают ситуации, когда в контроллер передаются значения, не связанные с моделью данных. Тогда эти значения необходимо проверять вручную. В метод «AddBook» класса «BookController», помимо модели представления книги, используется сторонний сервис «UserManager», с помощью которого контроллер получает значение Id пользователя. Однако есть вероятность того, метод вернет пустое значение, если пользователь не будет найден на сервере. Поэтому необходимо задать дополнительную проверку на null. В случае ошибки контроллер вызовет представление «NotFound». Валидация в методе «AddBook» представлена в листинге 7.5:

Листинг 7.5 – Валидация в методе «AddBook»

[HttpPost]

public async Task<IActionResult> AddBook(BookViewModel model)

{

if (ModelState.IsValid)

{

var userId = \_userManager.GetUserId(User);

if (userId == null)

{

ViewBag.ErrorMessage = $"Пользователь с Id : {userId} не может быть найден";

return NotFound();

}

var user = \_userManager.FindByIdAsync(userId).Result;

await \_bookService.Add(newBook);

return RedirectToAction("Index");

}

return View(model);

}

# 8 Взаимодействие пользователя с веб-сайтом

8.1 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 8.1:

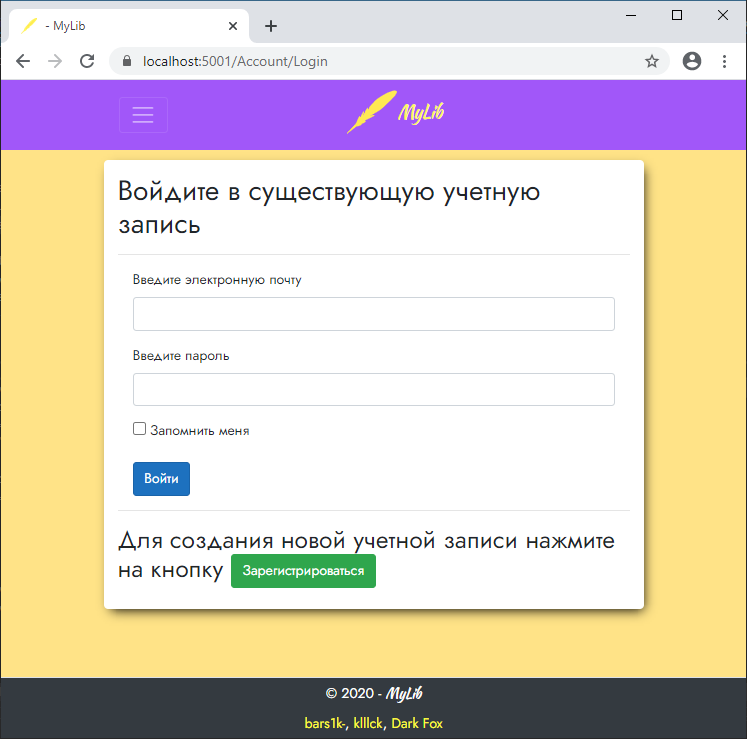


Рисунок 8.1 – Форма авторизации

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 8.2:

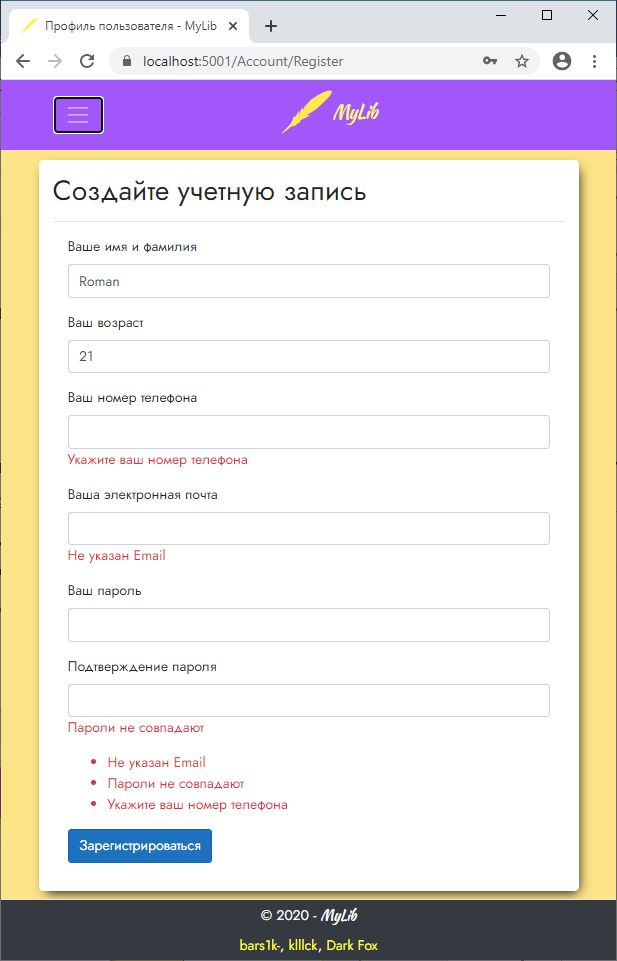


Рисунок 8.2 – Форма регистрации

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.

8.2 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 8.3:

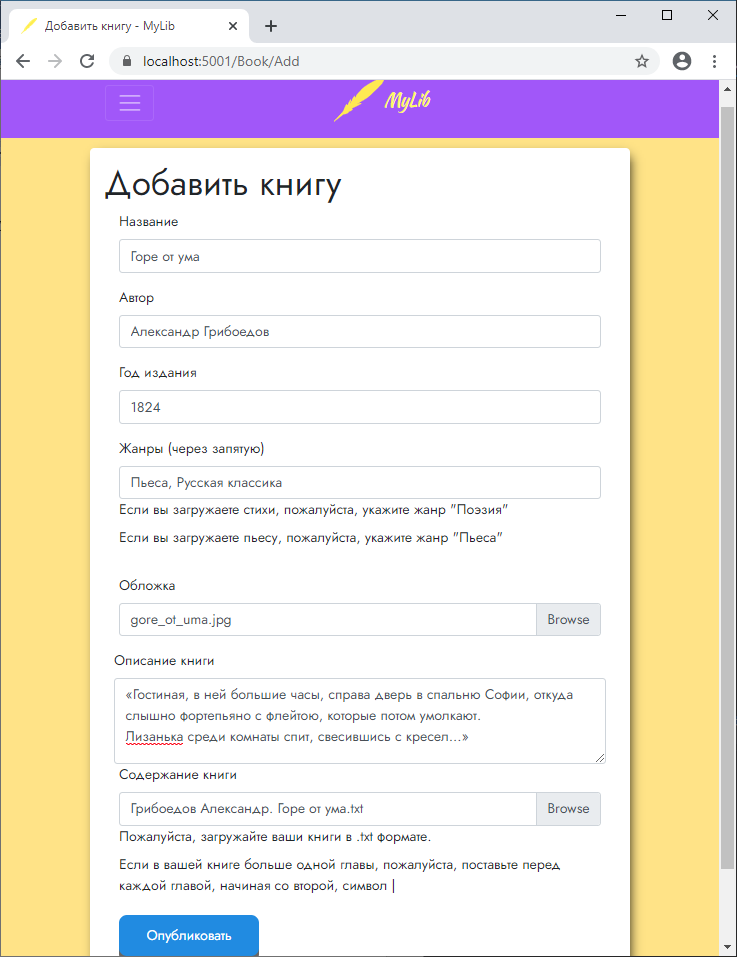


Рисунок 8.3 – Форма публикации новой книги

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 8.4):

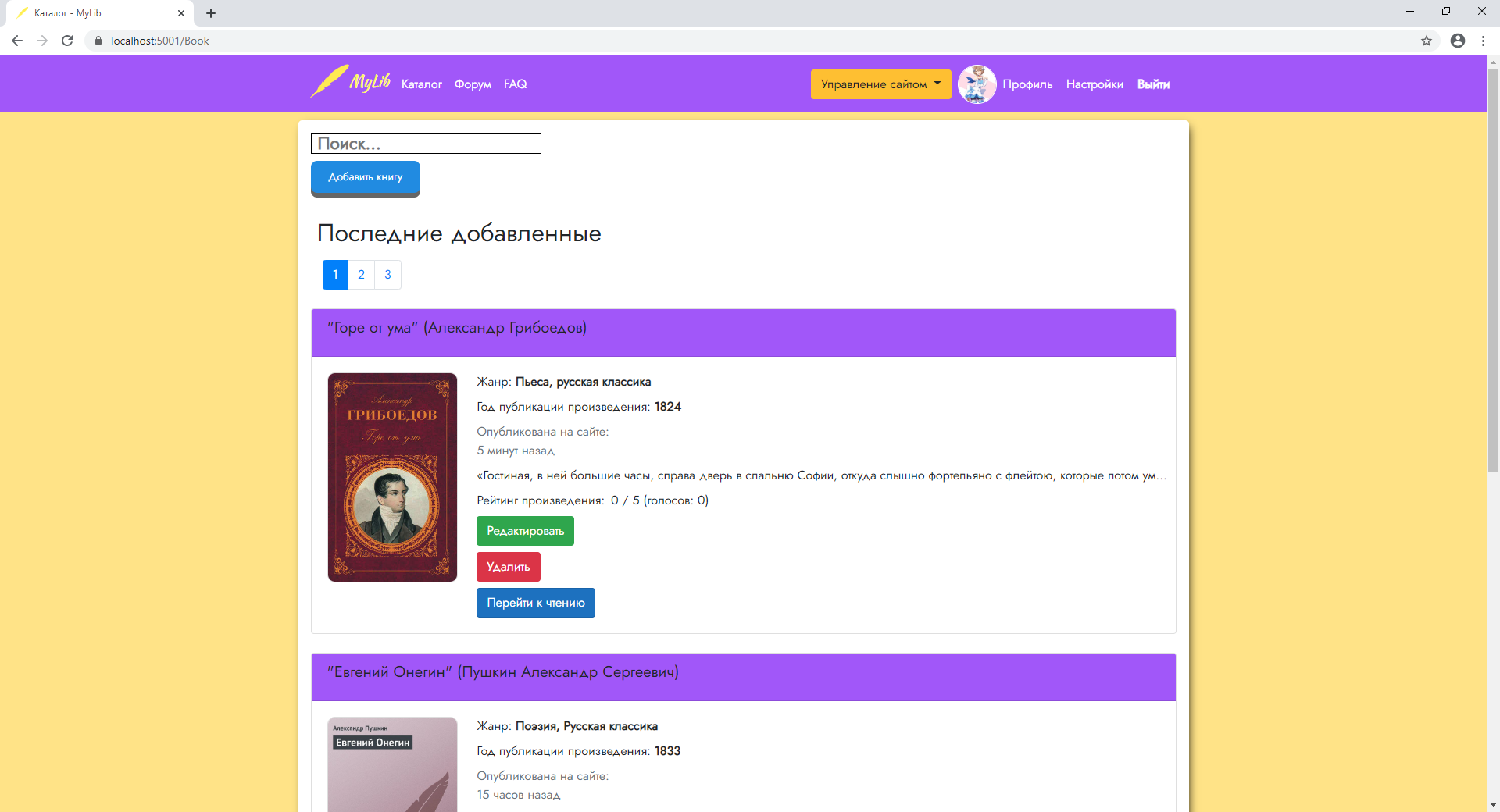


Рисунок 8.4 – Раздел «Последние добавленные»

# 9 Оценка положительного эффекта

На данный момент в сети Интернет существует большое количество сайтов по типу электронных библиотек. Перед началом проектирования нашего веб-приложения были изучены самые разные варианты подобных сайтов. Так как темой ВКР является разработка непосредственно серверной части приложения, будем сравнивать функционал нашего сайта с другими.

Сайт proza.ru является известным российским литературным веб-порталом, на котором пользователи могут свободно публиковать свои произведения. Здесь каталог произведений разбит на группы по разным критериям подбора, которые вручную задаются самими разработчиками сайта. В нашей системе присутствует уникальная система кластеризации, которая сама распределяет все произведения на группы. Алгоритм кластеризации, разработанный другим участником нашего комплексного проекта, внедрен на сервер приложения.

Другой особенностью нашего сайта является наличия тегов. Для любого произведения добавляется любое количество тегов. По тегам пользователь с легкостью может найти нужную ему книгу. На сайте proza.ru поиск осуществляется только по названию книги или имени автора.

Еще одной особенностью веб-приложения «Литература» является наличие пользовательского форума. Наш форум не имеет особых различий с форумами на других сайтах. На форуме добавляются темы, в которых ведется дискуссия между пользователями. На сайте proza.ru нет форума, однако там есть раздел, на котором расположены все пользовательские рецензии на то или иное произведение.

На последнем этапе разработки было проведено тестирование функционала приложения несколькими пользователями с целью выявления каких-либо багов и ошибок. Результаты были крайне положительными. Все недочеты были исправлены.

**10 Безопасность и экологичность проекта**

**10.1 Значение и задачи безопасности жизнедеятельности**

Несмотря на то, что в настоящее время уровень развития технологий шагнул далеко вперед, негативное воздействие персональных компьютеров и других электронно-вычислительных устройств на организм человека все еще присутствует, хоть и в меньшем объеме. Более серьезной проблемой является усиленная зависимость от различных электронных гаджетов у нынешнего поколения. С каждым годом среднее время препровождения человека за компьютером все увеличивается. Все это приводит к ухудшению состояния человеческого здоровья. Появляются такие симптомы как плохое самочувствие, переутомление, нарушения сна, головные боли, нервное и эмоциональное напряжения. Самым главным негативным фактором является нарушение зрительных функций, так как современному человеку приходится долгое время смотреть на экраны различных устройств.

Так как исключить необходимость в использовании компьютера или интернета в жизни современного человека невозможно, желательно минимизировать влияние неблагоприятных факторов на организм человека, либо вовсе их ликвидировать. Ряд мероприятий и требований по безопасности и охраны труда при работе с персональными компьютерами и другими электро-вычислительными средствами регламентирован в следующих нормативных документах:

– СанПиН 2.2.2/2.4.1340‑03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы»;

– ТОИ Р‑45‑084‑01 «Типовая инструкция по охране труда при работе на персональном компьютере»;

– ГОСТ 12.0.003‑2015 «Система стандартов безопасности труда (ССБТ). Опасные и вредные производственные факторы. Классификация»;

– «Трудовой кодекс Российской Федерации» от 30.12.2001 № 197-ФЗ (ред. от 10.01.2016).

**10.2 Анализ условий труда и мероприятия по защите от воздействия вредных производственных факторов**

Приложением «Литература» является веб-сайт, на котором пользователи могут находить и читать различные литературные произведения в режиме онлайн. При длительном чтении любой информации с экрана монитора возникают различные негативные последствия: усталость, переутомление, напряженность и другие нарушения зрительных функций глаз. Постоянное воздействие статического, электромагнитного, рентгеновского и ультрафиолетового излучения, исходящего от ПК, также способствуют к появлению таких отрицательных реакций на организм человека, как слабость, утомляемость, нарушение сна и общего самочувствия. Также человек, работающий за компьютером, длительное время пребывает в положении «сидя», что приводит к перенапряжению мышц спины, рук и ног.

Для минимизации описанных выше негативных факторов влияния ПК на здоровье человека создаются все необходимые условия для комфортной работы. Также следует соблюдать нормы трудового процесса по устранению тяжести и напряженности.

Чтобы минимизировать и ликвидировать последствия, описанные выше, созданы комфортные условия для работы, а также соблюдены все нормы по устранению тяжести и напряжённости трудового процесса. Проведем анализ оценки напряженности трудового процесса пользователя ПК, используя нормативный документ Р 2.2.2006‑05 «Руководство по гигиенической оценке факторов рабочей среды и трудового процесса. Критерии и классификация условий труда». Результаты анализа продемонстрированы в таблице 10.1:

Таблица 10.1 – Оценка напряженности трудового процесса

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Показатели** | **Класс условий труда** | | | | | |
| **1** | | | **2** | **3.1** | **3.2** |
| 1 Интеллектуальные нагрузки | | | | | | |
| 1.1 Содержание работы | |  | |  | + |  |
| 1.2 Восприятие информации и их оценка | |  | |  | + |  |
| 1.3 Распределение функций по степени сложности задания | |  | | + |  |  |
| 1.4 Характер выполняемой работы | |  | | + |  |  |
| 2 Сенсорные нагрузки | | | | | | |
| 2.1 Длительность сосредоточенного наблюдения (% времени смены) | |  | |  | + |  |
| 2.2 Плотность сигналов (световых, звуковых) и сообщений в среднем за 1 час работы | |  | | + |  |  |
| 2.3 Число производственных объектов одновременного наблюдения | | + | |  |  |  |
| 2.4 Размер объекта различения (при расстоянии от глаз работающего до объекта различения не более 0,5 м) в мм при длительности сосредоточенного наблюдения (% времени смены) | |  | | + |  |  |
| 2.5 Работа с оптическими приборами при длительности сосредоточения наблюдения | | + | |  |  |  |
| 2.6 Наблюдение за экранами видеотерминалов (часов в смену) | |  | |  |  | + |
| 2.7 Нагрузка на слуховой анализатор | |  | | + |  |  |
| 2.8 Нагрузка на голосовой аппарат | | + | |  |  |  |
| 3 Эмоциональные нагрузки | | | | | | |
| 3.1 Степень ответственности за результат собственной деятельности. Значимость ошибки | | | + |  |  |  |
| 3.2 Степень риска для собственной жизни | | |  |  |  | + |
| 3.3 Ответственность за безопасность других лиц | | | + |  |  |  |
| 3.4 Количество конфликтных ситуаций в течение смены – от 1 до 3. | | |  | + |  |  |

*Окончание таблицы 10.1*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 Монотонность нагрузок | | | | |
| 4.1 Число элементов, необходимых для реализации простого задания или многократно повторяющихся операций | + |  |  |  |
| 4.2 Продолжительность выполнения простых заданий или повторяющихся операций |  |  | + |  |
| 4.3 Время активных действий |  |  | + |  |
| 4.4 Монотонность производственной обстановки (время пассивного наблюдения за ходом техпроцесса в % от времени смены) | + |  |  |  |
| 5 Режим работы | | | | |
| 5.1 Фактическая продолжительность рабочего дня |  | + |  |  |
| 5.2 Сменность работы |  |  | + |  |
| 5.3 Наличие регламентированных перерывов и их продолжительность | + |  |  |  |
| Количество показателей в каждом классе | 8 | 7 | 6 | 2 |
| Общая оценка напряженности труда | 3.2 | | | |

После проведённого анализа были получены следующие результаты: В 1 класс входят 8 показателей, во 2 класс – 7 показателей, в класс 3.1 – 6 показателя, а в класс 3.2 – 2 показателя. Так как не менее 6 показателей отнесены к классу 3.1 и от 1 до 5 показателей относятся к классу 3.2, то уровень напряженности трудового процесса является *вредным (труд напряженный 2-й степени).*

Так как пользователь, работающий за персональным компьютером, подвергает риску свое здоровье, можно добиться снижения уровня напряженности труда. Для этого можно снизить усталость глаз и напряженность мышц рук, ног и спины путем увеличения времени перерывов между работой. Для повышения эффективности рекомендуется проделывать различные физические упражнения, включая зрительные. Соблюдение данных рекомендаций приведет к снижению уровня показателей 2.6 и 3.2. Тогда общий уровень напряженности труда можно будет считать *допустимым*.

**10.3 Обеспечение электробезопасности**

Основным документом, регламентирующим нормы электробезопасности, является ГОСТ Р 12.1.009-2009 «Система стандартов безопасности труда (ССБТ). Электробезопасность.»

Так как веб-сайт оперирует с большими объемами данных, для их хранения необходимы сервера. Они находятся в состоянии круглосуточной беспрерывной работы, постоянно потребляя большое количество электроэнергии.

Для обеспечения электробезопасности используются следующие технические средства: электропроводка с качественной изоляцией, электрогенераторы и бесперебойные устройства. Проводятся организованные мероприятия с назначением квалифицированных лиц по проведению ремонта технических устройств и проверок электрооборудования. Периодически проводится осмотр с целью выявления повреждений изоляции. В серверных помещениях соблюден умеренный уровень опасности: температура от 20° до 25° по Цельсию, относительная влажность воздуха не превышает 50%, отсутствует токопроводящая пыль и влага. Также предусмотрено заземление серверных помещений для защиты от опасного воздействия электрического тока.

**10.4 Обеспечение пожарной безопасности**

Пожарная безопасность для серверных помещений регламентируется Федеральным законом № 123-ФЗ от 22 июля 2008 г. «Технический регламент о требованиях пожарной безопасности» и Правилом противопожарного режима Российской Федерации, утвержденные постановлением Правительства РФ от 25 июля 2012 г. № 390 «О противопожарном режиме».

По данным нормам и документам помещения обеспечиваются системой пожарной безопасности. Для обеспечения высокого уровня пожарной безопасности выполняются следующие меры:

– размещены эвакуационные планы для помещений на всех этажах;

– размещены таблички с номерами пожарных служб;

– размещены знаки, запрещающие курение;

– установлены системы пожаротушения и пожарной сигнализации;

– установлены всевозможные средства для ликвидации очагов пожара;

– определены сроки прохождения противопожарного инструктажа и занятий по пожарно-техническому минимуму c назначением ответственных лиц, отвечающих за их проведение.

Соблюдение всех вышеописанных мер значительно уменьшают вероятность возникновения чрезвычайных ситуаций. За соблюдением норм и правил пожарной безопасности отвечает руководитель организации.

# Заключение

В результате выполнения выпускной квалификационной работы была разработана серверная часть веб-приложения «Литература», предоставляющее пользователю функционал онлайн-библиотеки. В нем есть возможность авторизации пользователей, система пользовательских ролей, позволяющая разграничить доступ к функционалу веб-сайта. Пользователи могут находить и читать различные литературные произведения в онлайн режиме, оставлять свои рецензии к книгам, а также общаться с другими пользователями на форуме. Для внесения изменений в структуру данных сайта и поддержания контроля в базу данных внесены специальные администраторские учетные записи.

В ходе разработке проекта были созданы модели для взаимодействия приложения с данными из БД, сервисы для определения логики работы приложения и контролеры, позволяющие определить взаимодействие пользователей с веб-сервером.

При разработке данной информационной системы были изучены методы проектирования реляционных баз данных с использованием технологии Entity Framework Core и Code First подхода. Также были получены новые знания в области backend-разработки, используя фреймворк ASP.NET Core и объектно-ориентированный язык программирования C#.

Несмотря на то, что идея о создании онлайн-библиотеки не является новой, сайт «Литература» собрал в себе лучшие черты других веб-приложений, а также обзавелся уникальными новшествами. При дальнейшем сопровождении и улучшении данной информационной системы, разработанное ПО может превратиться из небольшого веб-сайта до полноценного веб-портала мирового масштаба.

# 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.

6. Документация по ASP.NET Core. Компания Microsoft / MSDN [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/aspnet/core/?view=aspnetcore-3.1 (дата обращения 25.05.2020)

7. Документация по Entity Framework Core. Компания Microsoft / MSDN. [Электронный ресурс]. URL: <https://docs.microsoft.com/ru-ru/ef/core/> (дата обращения 02.06.2020)