

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение высшего образования  
**«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**  
**(МОСКОВСКИЙ ПОЛИТЕХ)**

КУРСОВОЙ ПРОЕКТ  
По курсу Проектирование веб-сервисов  
ТЕМА «Разработка веб-приложения для системы управления  
электронной библиотекой»

Выполнил Голодяев Максим Андреевич  
Группа 221-329  
Проверил Кружалов Алексей Сергеевич

Москва, 2024

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»  
(**МОСКОВСКИЙ ПОЛИТЕХ**)

УТВЕРЖДАЮ  
заведующая кафедрой «Инфокогнитивные технологии»  
/ Е. А. Пухова /  
«\_\_\_» 2024 г.

**ЗАДАНИЕ**  
**на выполнение курсовой работы (проекта)**

Голодяеву Максиму Андреевичу,

обучающемуся группы 221-329, направления подготовки 09.03.01  
«Информатика и вычислительная техника» по дисциплине «Разработка веб-  
приложений» на тему «Разработка веб-приложения для системы управления  
электронной библиотекой»

- Исходные данные к работе (проекту): информационные ресурсы в сети интернет, научные публикации в открытой печати.
- Содержание задания по курсовой работе (проекту) – перечень вопросов, подлежащих разработке:

Разрабатываемый вопрос	Объем от всего задания, %	Срок выполнения	Примечание
Раздел 1. Анализ предметной области			
Задача 1.1. Обзор существующих программных продуктов по теме работы			
Задача 1.2. Анализ программных инструментов разработки веб-приложений			
Задача 1.3. Формулировка цели и задач работы			
Раздел 2. Проектирование веб-приложения			
Задача 2.1. Анализ целевой аудитории			
Задача 2.2. Описание функциональности приложения (диаграмма вариантов использования, user story и т. д.)			
Задача 2.3. Проектирование модели данных (ER-диаграмма, логическая и физическая схемы БД)			
Задача 2.4. Разработка макетов страниц (Wireframe)			
Раздел 3. Разработка веб-приложения			
Задача 3.1. Разработка базовой структуры приложения и вёрстка шаблонов страниц			
Задача 3.2. Реализация аутентификации пользователей			
Задача 3.3. Реализация CRUD-интерфейса для книг, авторов и читателей			
Задача 3.4. Реализация отчетности по наличию и активности читателей для администратора			
Задача 3.5 Реализация возможности просмотра и фильтрации книг по различным критериям			
Задача 3.6 Реализация загрузки и хранения книг			
Раздел 4. Оформление итогов работы			
Задача 4.1. Создание Git-репозитория с кодом проекта			
Задача 4.2. Деплой приложения на хостинг			
Задача 4.3. Оформление отчёта о проделанной работе			

Руководитель курсовой работы (проекта): преподаватель кафедры «Инфокогнитивные технологии»

«\_\_\_» 2024 г. \_\_\_\_\_ А. С. Кружалов  
Дата выдачи задания «\_\_\_» 2024 г.  
«\_\_\_» 2024 г.  
Дата сдачи выполненной работы (проекта)  
«\_\_\_» 2024 г. \_\_\_\_\_ (подпись) (И. О. Фамилия)  
Задание принял к исполнению  
«\_\_\_» 2024 г. \_\_\_\_\_

## ЛИСТ ЗАМЕЧАНИЙ

## СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ.....</b>	<b>5</b>
<b>1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ .....</b>	<b>6</b>
1.1 Обзор существующих программных продуктов .....	6
1.2 Анализ программных инструментов разработки .....	8
1.3 Формулировка цели и задач работы.....	11
<b>2 ПРОЕКТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ .....</b>	<b>12</b>
2.1 Анализ целевой аудитории.....	12
2.2 Описание функциональности приложения.....	13
2.3 Концептуальное проектирование базы данных .....	14
2.4 Разработка макета страниц .....	17
<b>3 РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ .....</b>	<b>24</b>
<b>4 ОФОРМЛЕНИЕ ИТОГОВ РАБОТЫ .....</b>	<b>31</b>
<b>ЗАКЛЮЧЕНИЕ .....</b>	<b>32</b>
<b>СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ.....</b>	<b>33</b>

## **ВВЕДЕНИЕ**

В современном информационном обществе эффективное управление ресурсами информации становится важнейшим элементом успешной работы в различных сферах деятельности. С ростом объёма электронных материалов и скорости доступа к ним, необходимость в эффективных инструментах управления этой информацией становится более острой. Однако, существующие решения в области управления электронными библиотеками не всегда отвечают потребностям пользователей и не обеспечивают достаточного уровня удобства и функциональности.

Целью данного проекта является разработка веб-приложения для управления электронной библиотекой. Основные задачи включают в себя создание удобного и интуитивно понятного интерфейса для добавления, организации и поиска электронных материалов, а также обеспечение возможности работы с различными типами файлов.

Актуальность проблемы подчёркивается повседневной необходимостью эффективного управления информацией как в рабочей, так и в личной сфере. Множество существующих продуктов на рынке предоставляют базовый функционал управления электронными материалами, однако требуется более гибкое и персонализированное решение, способное адаптироваться под индивидуальные потребности пользователей.

В ходе работы будут проанализированы существующие подходы и программные продукты в области управления электронными библиотеками, а также определены требования пользователей. В результате будет разработано веб-приложение, которое представляет собой современный инструмент эффективного управления электронной библиотекой, способный удовлетворить потребности пользователей из различных сфер деятельности.

# **1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **1.1 Обзор существующих программных продуктов**

Для проведения анализа необходимо учитывать следующие критерии:

- Функциональность: оценка доступных функций управления электронной библиотекой, таких как добавление, организация, поиск материалов, работа с метаданными, метки, и т.д.
- Удобство использования: оценка интерфейса пользователя, его интуитивной понятности, удобства навигации и доступности основных функций. Важно обратить внимание на удобство чтения и поиска, возможности сортировки и фильтрации, а также наличие возможности персонализации интерфейса.
- Поддержка платформ: исследование возможности использования приложений на различных платформах и их совместимость с различными операционными системами.

Рассмотрим несколько существующих на рынке веб-приложений для управления электронной библиотекой, используя данные критерии:

### **1.1.1 ЛитРес (litres.ru)**

- Функциональность: ЛитРес предлагает широкий спектр функций, включая возможность просмотра и покупки электронных книг, а также аудиокниг. Система позволяет организовать коллекции книг и провести поиск по различным параметрам. Есть возможность оценки и отзывов, а также синхронизации библиотеки между устройствами.
- Удобство использования: Интерфейс ЛитРеса интуитивно понятен, навигация по сайту и мобильным приложениям удобна. Есть возможность настройки предпочтений, таких как размер и шрифт текста.
- Поддержка платформ: ЛитРес предоставляет доступ к библиотеке через веб-версию, мобильные приложения для iOS и Android, а также десктопные приложения для Windows и macOS.

### **1.1.2 Bookmate ([bookmate.com](http://bookmate.com))**

- Функциональность: Bookmate предлагает доступ к широкому ассортименту электронных книг и аудиокниг. Пользователи могут добавлять книги в свою коллекцию, отмечать прочитанные и ставить оценки. Есть возможность обсуждения книг в сообществе.
- Удобство использования: Интерфейс Bookmate прост и удобен в использовании. Есть функции персонализации и рекомендации книг на основе предпочтений пользователя.
- Поддержка платформ: Bookmate доступен через веб-версию, мобильные приложения для iOS и Android, а также имеется возможность чтения книг через десктопные приложения для Windows и macOS.

### **1.1.3 MyBook ([mybook.ru](http://mybook.ru))**

- Функциональность: MyBook предоставляет доступ к электронным книгам, аудиокнигам, а также журналам и газетам. Пользователи могут добавлять книги в избранное, оценивать и комментировать. Есть возможность синхронизации библиотеки между устройствами.
- Удобство использования: Интерфейс MyBook простой и понятный, навигация по сайту и мобильным приложениям удобна. Система рекомендаций помогает пользователям находить интересные книги.
- Поддержка платформ: MyBook доступен через веб-версию, мобильные приложения для iOS и Android, а также имеются десктопные приложения для Windows и macOS.

Исходя из проведенного анализа существующих онлайн библиотек в России, можно сделать вывод о том, что на рынке уже существует широкий выбор платформ с различными функциональными возможностями и уровнями удобства использования. Эти библиотеки предоставляют доступ к разнообразным электронным книгам и аудиокнигам, позволяют организовывать коллекции, отмечать прочитанное и оценивать материалы.

Однако, в процессе анализа также становится очевидным, что каждая из существующих платформ имеет свои особенности и ограничения, которые могут не полностью соответствовать потребностям конкретного пользователя или группы пользователей. Некоторые платформы могут предлагать более широкий выбор книг и аудиокниг, в то время как другие могут обладать более удобным интерфейсом или интеграцией с социальными сетями.

## **1.2 Анализ программных инструментов разработки**

Рассмотрим несколько инструментов, через которые можно реализовать веб-приложение:

### **1.2.1 HTML и CSS**

HTML (HyperText Markup Language) и CSS (Cascading Style Sheets) являются основными языками разметки и стилей для веб-страниц. HTML определяет структуру содержимого страницы, а CSS задает ее визуальное оформление.

#### **Преимущества**

- Простота изучения и использования
- Поддержка множества браузеров
- Гибкий дизайн

#### **Недостатки**

- Отсутствие программной логики
- Ограниченнная анимация и интерактивность

### **1.2.2 Bootstrap**

Bootstrap — это бесплатный и открытый фреймворк для разработки веб-приложений с использованием HTML, CSS и JavaScript. Он предоставляет набор готовых компонентов и стилей, которые значительно упрощают создание современных и адаптивных веб-интерфейсов.

#### **Преимущества**

- Быстрый старт
- Адаптивный дизайн

- Кросбраузерная совместимость

#### **Недостатки**

- Стандартизация дизайна
- Дополнительный объем

### **1.2.3 Python с использованием фреймворка Django**

Python - высокоуровневый язык программирования с простым и понятным синтаксисом. Django — это высокоуровневый веб-фреймворк для Python, который упрощает разработку веб-приложений.

#### **Преимущества**

- Простота и читаемость кода
- Богатый набор инструментов
- Высокая производительность

#### **Недостатки**

- Ограничения гибкости

### **1.2.4 Python с использованием фреймворка Flask**

Flask — это легкий и гибкий веб-фреймворк для Python, который предоставляет инструменты для создания веб-приложений. Он основан на принципе WSGI и предлагает минималистичный подход к разработке, оставляя разработчику большую свободу в выборе инструментов и архитектуры приложения.

#### **Преимущества**

- Простота использования
- Гибкость
- Масштабируемость
- Расширяемость

#### **Недостатки**

- Меньше "из коробки" функциональности
- Больше свободы может привести к большему количеству архитектурных решений

### **1.2.5 MySQL**

MySQL — это популярная реляционная база данных, отлично подходящая для хранения и управления данными в веб-приложениях. Она обеспечивает надежное хранение данных и эффективную обработку запросов.

#### **Преимущества**

- Надежность и производительность
- SQL язык запросов
- Широкая поддержка
- Масштабируемость

#### **Недостатки:**

- Требуется обращение к базе данных
- Сложность масштабирования

### **1.2.6 PostgreSQL**

PostgreSQL — это мощная и открытая реляционная база данных с широким набором функций, которая хорошо подходит для хранения и управления данными в веб-приложениях. Она предоставляет надежное и эффективное хранилище данных.

#### **Преимущества**

- Надежность и производительность
- Расширенные возможности
- SQL язык запросов
- Открытая лицензия и активное сообщество

#### **Недостатки**

- Требуется настройка
- Большое потребление ресурсов

Каждый из этих инструментов имеет свои преимущества и недостатки, и выбор зависит от конкретных требований проекта, потребностей разработчика и контекста использования. Например, для быстрого прототипирования или маленьких проектов Flask может быть

предпочтительным выбором, в то время как для крупных и сложных веб-приложений Python в сочетании с более крупными фреймворками, такими как Django или Flask, может быть более подходящим.

### **1.3 Формулировка цели и задач работы**

Основная цель курсового проекта «Разработка веб-приложения для системы управления электронной библиотекой», следующая: создание удобного и эффективного инструмента для поиска, чтения и управления электронными книгами в рамках электронной библиотеки. Приложение должно удовлетворять потребности как библиотекарей, так и обычных пользователей, делая процесс поиска и доступа к информации максимально удобным и эффективным.

Задачи проекта:

1. Анализ предметной области.
2. Проектирование веб-приложения.
3. Разработка веб-приложения.
4. Тестирование и отладка.
5. Оформление итогов работы.

## **2 ПРОЕКТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ**

### **2.1 Анализ целевой аудитории**

В рамках системы будет три основных актора:

#### **2.1.1 Администратор**

Администратор — это пользователь, управляющий всей системой, ответственный за её настройку, безопасность, обслуживание и общее функционирование.

##### **Потребности и ожидания**

Администратор ожидает от системы удобного интерфейса для управления пользователями и их правами доступа, возможности добавления и удаления книг из библиотеки.

##### **Требования**

Приложение должно предоставлять административный интерфейс с возможностью управления пользователями, добавления и удаления книг, настройки прав доступа и просмотра отчетов о работе системы.

#### **2.1.2 Библиотекарь**

Библиотекарь — это пользователь, управляющий содержимым библиотеки, добавляющий и редактирующий книги, следящий за их доступностью и организацией каталога.

##### **Потребности и ожидания**

Библиотекарь ожидает от системы удобного интерфейса для добавления и редактирования книг, управления каталогом, мониторинга доступности книг и обслуживания запросов пользователей.

##### **Требования**

Приложение должно предоставлять библиотекарю инструменты для управления книгами, добавления и редактирования их описаний, управления категориями и авторами.

#### **2.1.3 Читатель**

Читатель — это конечный пользователь, который использует приложение для поиска и чтения.

## **Потребности и ожидания**

Читатель хочет удобного интерфейса для поиска и фильтрации книг, возможности читать книги онлайн.

## **Требования**

Приложение должно обеспечивать удобный поиск и возможность чтения онлайн с сохранением прогресса.

### **2.2 Описание функциональности приложения**

- Регистрация и аутентификация:
  - Система должна позволять пользователям регистрироваться и аутентифицироваться с помощью логина и пароля.
- Управление книгами:
  - Администратор должен иметь возможность добавлять, редактировать и удалять книги из каталога.
  - Для каждой книги должны храниться следующие данные: название, автор(ы), жанр, издательство, год издания, обложка.
- Управление библиотекарями:
  - Администратор должен иметь возможность добавлять, редактировать и удалять информацию об библиотекарях.
  - Для каждого библиотекаря должны храниться следующие данные: имя, фамилия, уникальный идентификатор.
  - Система должна позволять библиотекарям добавлять новые книги в электронную библиотеку.
- Аренда книг:
  - Пользователи должны иметь возможность арендовать электронные книги.
  - Система должна регистрировать дату аренды и срок пользования книгой.
  - После окончания срока аренды доступ к книге должен блокироваться.

- Чтение книг:
  - Пользователи должны иметь возможность читать книги онлайн внутри системы.
- Управление пользователями:
  - Администратор должен иметь возможность просматривать и управлять данными пользователей, включая редактирование и блокировку учетных записей.
  - Пользователи должны иметь возможность просматривать и редактировать свои профили.
- Функция поиска:
  - Пользователи должны иметь возможность искать книги по названию.
- Генерация отчетов:
  - Администратор должен иметь доступ к отчетам о деятельности системы, таким как статистика аренды, популярность книг и т.д.
- Безопасность:
  - Система должна обеспечивать защиту пользовательских данных и конфиденциальную информацию о книгах.
  - Доступ к административным функциям должен быть защищен аутентификацией и авторизацией.
- Интерфейс:
  - Пользовательский интерфейс должен быть интуитивно понятным и легким в использовании.
  - Система должна быть адаптивной для использования на различных устройствах, таких как компьютеры, планшеты и смартфоны.

## **2.3 Концептуальное проектирование базы данных**

### **2.3.1 Сущности**

Сущностями в базе данных будут выступать книги, категории, стеллажи, серии, издательства, продавцы, роли и аренды.

### **2.3.2 Атрибуты**

Создадим таблицу «Книги» добавив в неё соответствующие атрибуты.

Таблица будет выглядеть как показано в таблице 1.

Таблица 1 – Книги

Наименование параметра	Тип данных	Размер	Диапазон значений
Ключ книги	Числовой		
Автор	Текстовой		
Название	Текстовой		
Ключ серия	Числовой		
Ключ издатель	Числовой		

Создадим таблицу «Категория» добавив в неё соответствующие атрибуты. Таблица будет выглядеть как показано в таблице 2.

Таблица 2 – Категория

Наименование параметра	Тип данных	Размер	Диапазон значений
Ключ категории	Числовой		
Название	Текстовой		

Создадим таблицу «Серия» добавив в неё соответствующие атрибуты. Таблица будет выглядеть как показано в таблице 3.

Таблица 3 – Серия

Наименование параметра	Тип данных	Размер	Диапазон значений
Ключ серия	Числовой		
Название	Текстовой		

Создадим таблицу «Издательства» добавив в неё соответствующие атрибуты. Таблица будет выглядеть как показано в таблице 4.

Таблица 4 – Издательства

Наименование параметра	Тип данных	Размер	Диапазон значений
Ключ издательства	Числовой		
Название	Текстовой		

Создадим таблицу «Пользователи» добавив в неё соответствующие атрибуты. Таблица будет выглядеть как показано в таблице 5.

Таблица 5 – Пользователи

Наименование параметра	Тип данных	Размер	Диапазон значений
Ключ продавца	Числовой		
ФИО	Текстовой	256	
Логин	Текстовой		
Хэш пароль	Текстовой	256	
Ключ роли	Числовой		

Создадим таблицу «Роли» добавив в неё соответствующие атрибуты. Таблица будет выглядеть как показано в таблице 6.

Таблица 6 – Роли

Наименование параметра	Тип данных	Размер	Диапазон значений
Ключ роли	Числовой		
Название	Текстовой		

Создадим таблицу «Аренды» добавив в неё соответствующие атрибуты. Таблица будет выглядеть как показано в таблице 7.

Таблица 7 – Аренды

Наименование параметра	Тип данных	Размер	Диапазон значений
Ключ аренды	Числовой		
Дата начала	Дата	8	ГГГГ.ММ.ДД
Дата конца	Дата	8	ГГГГ.ММ.ДД
Ключ книги	Числовой		
Ключ пользователя	Числовой		

### 2.3.3 Физическая модель базы данных

Er-диаграмма и физическое определение полей показано на рисунке 1.

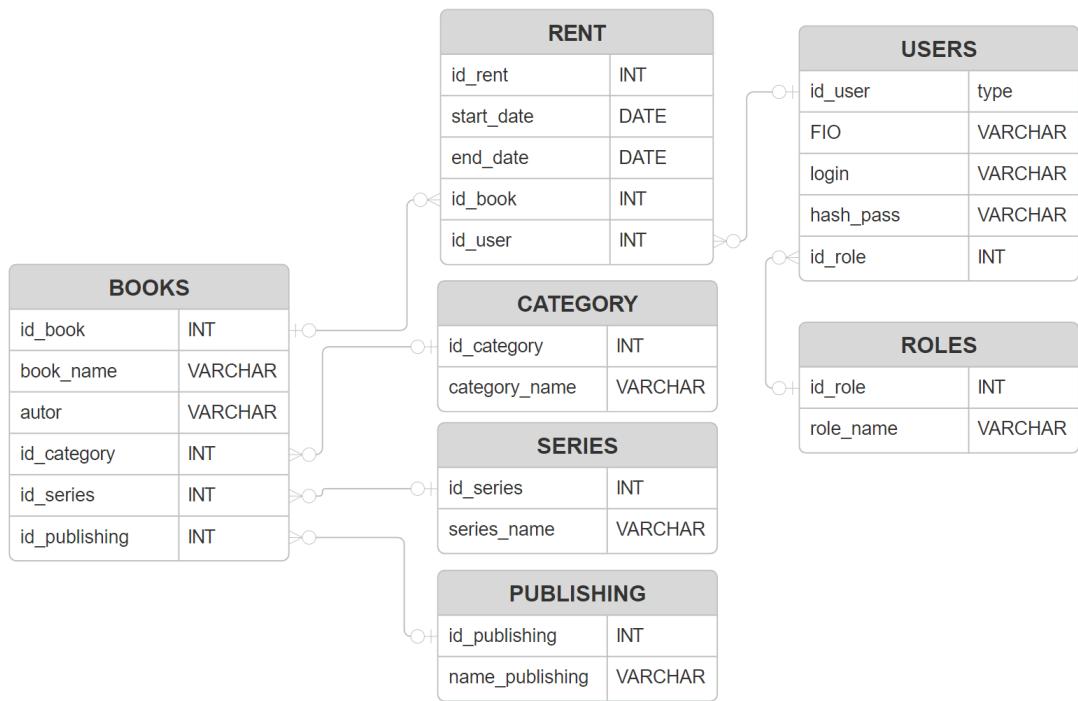


Рисунок 1 – ER-модель

## 2.4 Разработка макета страниц

Макет главной страницы представлен на рисунке 2.

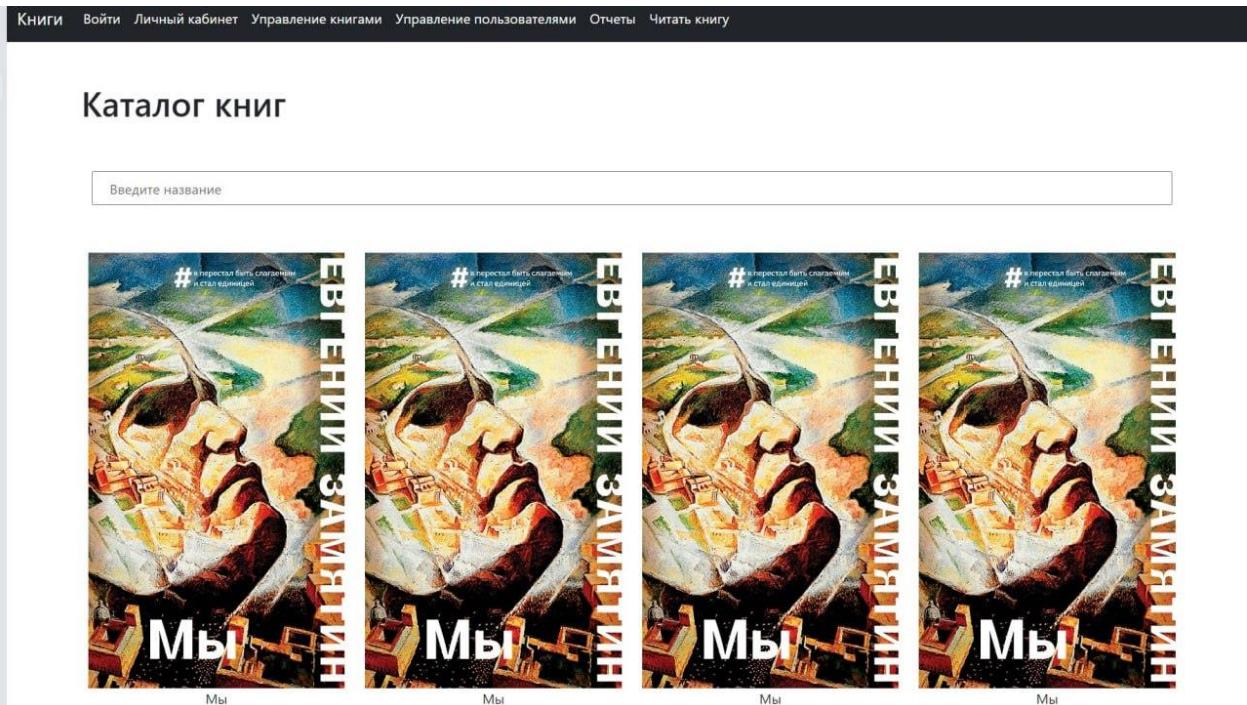


Рисунок 2 – Главная страница

Макет страницы с арендой представлен на рисунке 3.

Книги Войти Личный кабинет Управление книгами Управление пользователями Отчеты Читать книгу

## Аренда книги

Название

Автор

Жанр

Срок аренды

 1 месяц ▾

**Арендовать**

Контакты

Рисунок 3 – Макет аренды

Макет страницы «Читать книгу» представлен на рисунке 4.

Книги Войти Личный кабинет Управление книгами Управление пользователями Отчеты Читать книгу

## Название книги

Lorem ipsum dolor sit, amet consectetur adipisicing elit. Molestias odio dicta modi provident deleniti debitis obcaecati? Reprehenderit, exercitationem ut animi veritatis doloremque dolores itaque dolorum totam maiores, perspiciatis quidem nulla!

Контакты

Рисунок 4 – «Читать книгу»

Макет страницы с авторизацией представлен на рисунке 5.

Книги Войти Личный кабинет Управление книгами Управление пользователями Отчеты Читать книгу

## Авторизация

Логин

Пароль

Password

Запомнить меня

**Войти**

**Зарегистрироваться**

Контакты

Рисунок 5 – Авторизация

Макет страницы регистрации представлен на рисунке 6.

Книги Войти Личный кабинет Управление книгами Управление пользователями Отчеты Читать книгу

## Регистрация

Логин

Введите логин

Пароль

Введите пароль

Подтверждение пароля

Повторите пароль

Имя

Введите имя

Фамилия

Введите фамилию

Запомнить меня

**Зарегистрироваться**

Контакты

Рисунок 6 – Регистрация

Макет страницы личного кабинета представлен на рисунке 7.

## Личный кабинет

Логин

Логин

Пароль

Пароль

Имя

Имя

Фамилия

Фамилия

Edit

## Книги, читаемые в данный момент

№

Название книги

Автор

1

Мы

Замятин

Открыть

Убрать

Контакты

Рисунок 7 – Личный кабинет

Макет страницы по управлению книгами представлен на странице 8.

## КНИГИ

№

Автор

Название

Категория

Серия

Издатель

1

Замятин

Мы

Роман-антиутопия

1

Литрес

Edit

Delete

Добавить книгу

Контакты

Рисунок 8 – Управление книгами

Макет страницы с добавлением книги представлен на рисунке 9.

## Добавить книгу

Автор

Введите автор

Название

Введите название книги

Категория

Введите категорию

Серия

Введите серию

Издатель

Введите издателя

**Сохранить**

Контакты

## Рисунок 9 – Добавление книги

Макет страницы управление пользователями представлен на рисунке 10.

## Пользователи

№	Логин	Фамилия	Имя	Роль	
1	ывыв	Иванов	Иван	Пользователь	<b>Edit</b> <b>Delete</b>

Контакты

## Рисунок 10 – Управление пользователями

Макет страницы изменения пользователей представлена на рисунке 11.

Книги Войти Личный кабинет Управление книгами Управление пользователями Отчеты Читать книгу

## Изменение пользователей

Логин  
Логин

Пароль  
Пароль

Имя  
Введите имя

Фамилия  
Введите фамилию

Роль  
Администратор

**Сохранить**

Контакты

This screenshot shows a form titled 'Изменение пользователей' (Change user). It contains several input fields: 'Логин' (Login), 'Пароль' (Password), 'Имя' (Name) with placeholder 'Введите имя' (Enter name), 'Фамилия' (Surname) with placeholder 'Введите фамилию' (Enter surname), 'Роль' (Role) set to 'Администратор' (Administrator), and a dropdown arrow. At the bottom is a blue 'Сохранить' (Save) button.

Рисунок 11 – Изменение пользователей

Макет страницы по статистике аренд представлена 12.

Книги Войти Личный кабинет Управление книгами Управление пользователями Отчеты Читать книгу

Статистика аренды Статистика популярности книг

## Статистика аренды

№	Название книги	Автор	Количество аренд
1	Мы	Замятин	10
1	Мы	Замятин	20
1	Мы	Замятин	1

Контакты

This screenshot shows a table titled 'Статистика аренды' (Rental statistics). It displays three rows of data: 'Мы' by 'Замятин' with 10, 20, and 1 rentals respectively. Above the table are two navigation links: 'Статистика аренды' and 'Статистика популярности книг'. A dark footer bar at the bottom contains the text 'Контакты'.

Рисунок 12 – Статистика аренды

Макет страницы со статистикой популярности книг представлен на рисунке 13.

Статистика аренды Статистика популярности книг

## Статистика популярности книг

Название книги	Автор	Место по популярности
Мы	Замятин	1

Контакты

Рисунок 13 – Статистика популярности книг

### 3 РАЗРАБОТКА ВЕБ-ПРИЛОЖЕНИЯ

Программа была написана на языке python3 с использованием фреймфорка Flask, также с использованием HTML и CSS.

Далее будут представлены различные функции из программы.

#### Листинг 1 – Автоматическое удаление по истечению срока аренды

```
36 # Функция для удаления записи из таблицы rent при истечении срока аренды,
37 # будет вызываться каждый раз, когда пользователь будет заходить на страницу "Читать книгу" (read_book.html)
38 usage
39 def check_rent_expiration(id_user, id_book):
40     try:
41         current_date = datetime.now().date()
42         rent = db.session.query(Rent).filter_by(id_user=id_user, id_book=id_book).first()
43         if rent and rent.end_date <= current_date:
44             db.session.delete(rent)
45             db.session.commit()
46             return True
47         return False
48     except SQLAlchemyError as e:
49         db.session.rollback()
50         flash( message: 'Произошла ошибка при удалении просроченной аренды.', category: 'danger')
51         return False
```

#### Листинг 2 – Главная страница

```
54 @app.route('/')
55 <>> def index():
56     page = request.args.get('page', 1, type=int)
57     book_name = request.args.get('book_name', '')
58     query = db.select(Books)
59     if book_name:
60         query = query.filter(Books.book_name.ilike(f'%{book_name}%'))
61     pagination = db.paginate(query, per_page=9, page=page)
62     books = pagination.items
63     return render_template( template_name_or_list: "index.html", books=books, pagination=pagination, book_name=book_name)
```

#### Листинг 3 – Личный кабинет

```
67 @app.route('/personal_account')
68 @login_required
69 <>> def personal_account():
70     user_record = db.session.query(Users).filter_by(id_user=current_user.id_user).one()
71     rented_books = db.session.query(Books.id_book, Books.book_name, Books.author) \
72         .join(Rent, Rent.id_book == Books.id_book) \
73         .filter(Rent.id_user == current_user.id_user) \
74         .all()
75     return render_template( template_name_or_list: "personal_account.html", user_record=user_record, rented_books=rented_books)
```

#### Листинг 4 – Удаление арендованной книги из личного кабинета

```
79 @app.route( rule: '/<int:id_book>/delete', methods=['POST'])
80 @login_required
81 <>> def delete(id_book):
82     try:
83         rent_to_delete = db.session.query(Rent).filter_by(id_book=id_book).first()
84         db.session.delete(rent_to_delete)
85         db.session.commit()
86     except SQLAlchemyError as e:
87         db.session.rollback()
88         flash( message: f'Произошла ошибка при удалении записи: {str(e)}', category: 'danger')
89         return render_template('personal_account.html')
90
91     flash( message: 'Арендованная книга успешно удалена', category: 'success')
92     return redirect(url_for('personal_account'))
```

## Листинг 5 – Аренда книги

```
96     @app.route( rule: '/rent', methods=['GET', 'POST'])
97     @login_required
98 <> def rent():
99         id_book = request.args.get( key: 'id_book', type=int)
100        if request.method == 'POST':
101            rent_term = request.form.get('rent_term')
102            if rent_term == '3_days': end_date = date.today() + timedelta(days=3)
103            elif rent_term == '5_days': end_date = date.today() + timedelta(days=5)
104            elif rent_term == 'week': end_date = date.today() + timedelta(weeks=1)
105            elif rent_term == '2_week': end_date = date.today() + timedelta(weeks=2)
106            elif rent_term == 'month': end_date = date.today() + timedelta(days=30)
107            try:
108                # Проверка на то, что книга уже арендована,
109                existing_rent = db.session.query(Rent).filter_by(
110                    id_book=id_book, id_user=current_user.id_user
111                ).first()
112                if existing_rent:
113                    flash( message: 'Аренда на эту книгу уже оформлена.', category: 'danger')
114                    return redirect(url_for( endpoint: 'rent', id_book=id_book))
115                new_rent = Rent(
116                    start_date=date.today(), end_date=end_date, id_book=id_book, id_user=current_user.id_user
117                )
118                db.session.add(new_rent)
119                db.session.commit()
120                flash( message: 'Аренда успешно оформлена', category: 'success')
121                return redirect(url_for( endpoint: 'read_book', id_book=id_book))
122            except SQLAlchemyError as e:
123                db.session.rollback()
124                flash( message: 'Произошла ошибка при оформлении аренды. Попробуйте снова.', category: 'danger')
125                return redirect(url_for( endpoint: 'rent', id_book=id_book))
126
127        id_book = request.args.get( key: 'id_book', type=int)
128        book = db.session.query(
129            Books.id_book,
130            Books.book_name,
131            Books.author,
132            Books.id_series,
133            Books.id_publishing,
134            Category.category_name,
135            Series.series_name,
136            Publishing.name_publishing
137        ).outerjoin(Category, Books.id_category == Category.id_category
138            ).outerjoin(Series, Books.id_series == Series.id_series
139                ).outerjoin(Publishing, Books.id_publishing == Publishing.id_publishing
140                    ).filter(Books.id_book == id_book).one()
141
142        return render_template( template_name_or_list: "rent.html", book=book)
```

## Листинг 6 – Чтение книги

```
145     @app.route('/read_book')
146     @login_required
147 <> def read_book():
148         id_book = request.args.get( key: 'id_book', type=int)
149         user_rents = db.session.query(Rent).filter_by(id_user=current_user.id_user).order_by(desc(Rent.id_rent)).all()
150         if not user_rents:
151             flash( message: 'У вас нет арендованных книг.', category: 'danger')
152             return redirect(url_for('index'))
153         if not id_book:
154             last_rent = db.session.query(Rent).filter_by(id_user=current_user.id_user).order_by(desc(Rent.id_rent)).first()
155             id_book = last_rent.id_book
156         # Проверка, что срок аренды не истек
157         check_rent = check_rent_expiration(current_user.id_user, id_book)
158         if check_rent:
159             return redirect(url_for('index'))
160         book_name = db.session.execute(
161             db.select(Books.book_name).filter_by(id_book=id_book)
162         ).scalar_one_or_none()
163         return render_template( template_name_or_list: "read_book.html", book_name=book_name)
```

## Листинг 7 – Разграничение доступа

```
14     def admin_or_librarian(role):
15         def decorator(func):
16             @wraps(func)
17             def decorated(*args, **kwargs):
18                 if role == 'admin' and not current_user.is_admin():
19                     flash( message: 'Доступ к этой странице разрешен только администраторам.', category: 'danger')
20                 elif role == 'admin_librarian' and not (current_user.is_admin() or current_user.is_librarian()):
21                     flash( message: 'Доступ к этой странице разрешен только администраторам и библиотекарям.', category: 'danger')
22                 else:
23                     return func(*args, **kwargs)
24                 return redirect(url_for('index'))
25             return decorated
26         return decorator
```

## Листинг 8 – Управление книгами

```
28     # Страница управления книгами с пагинацией
29     @bp.route('/manage_books')
30     @login_required
31     @admin_or_librarian('admin_librarian')
32 <>> def manage_books():
33         page = request.args.get('page', 1, type=int)
34         query = db.session.query(
35             Books.id_book,
36             Books.book_name,
37             Books.author,
38             Category.category_name,
39             Series.series_name,
40             Publishing.name_publishing
41         ).outerjoin(Category, Books.id_category == Category.id_category
42         ).outerjoin(Series, Books.id_series == Series.id_series
43         ).outerjoin(Publishing, Books.id_publishing == Publishing.id_publishing)
44
45         pagination = query.paginate(per_page=10, page=page)
46         books = pagination.items
47         return render_template( template_name_or_list: "books/manage_books.html", books=books, pagination=pagination)
```

## Листинг 9 – Загрузка изображений

```
112     @bp.route( rule: '/upload_image', methods=['POST'])
113     @login_required
114     @admin_or_librarian('admin_librarian')
115     def upload_image():
116         id_book = request.args.get( key: 'id_book', type=int)
117         if 'image' not in request.files:
118             flash( message: 'Нет файла для загрузки', category: 'danger')
119             return redirect(url_for( endpoint: 'books.edit_books', id_book=id_book))
120
121         file = request.files['image']
122         if file.filename == '':
123             flash( message: 'Не выбрано изображение для загрузки', category: 'danger')
124             return redirect(url_for( endpoint: 'books.edit_books', id_book=id_book))
125
126         # Когда добавляем новое изображение при редактировании, то старое изображение переименовываем
127         if file:
128             filename = secure_filename(f'{id_book}.jpg')
129             old_filepath = os.path.join(current_app.static_folder, 'images', filename)
130             new_filepath = os.path.join(current_app.static_folder, 'images', f'{id_book}_old.jpg')
131
132             if os.path.exists(old_filepath):
133                 os.rename(old_filepath, new_filepath)
134
135             file.save(os.path.join(current_app.static_folder, 'images', filename))
136             flash( message: 'Изображение успешно загружено', category: 'success')
137
138         return redirect(url_for( endpoint: 'books.edit_books', id_book=id_book))
```

## Листинг 10 – Добавление новой книги

```
141 @bp.route('/new_book', methods=['GET', 'POST'])
142 @login_required
143 @admin_or_librarian('admin_librarian')
144 <> def new_book():
145     if request.method == 'POST':
146         author = request.form['author']
147         book_name = request.form['nameBook']
148         category_name = request.form['category']
149         series_name = request.form.get('seria', '')
150         publishing_name = request.form['publisher']
151         image = request.files['image']
152     try:
153         category = db.session.query(Category).filter_by(category_name=category_name).first()
154         if not category:
155             category = Category(category_name=category_name)
156             db.session.add(category)
157         series = db.session.query(Series).filter_by(series_name=series_name).first()
158         if not series and series_name:
159             series = Series(series_name=series_name)
160             db.session.add(series)
161         publishing = db.session.query(Publishing).filter_by(name_publishing=publishing_name).first()
162         if not publishing:
163             publishing = Publishing(name_publishing=publishing_name)
164             db.session.add(publishing)
165         db.session.commit()
166         new_book = Books(
167             author=author,
168             book_name=book_name,
169             id_category=category.id_category,
170             id_series=series.id_series if series else None,
171             id_publishing=publishing.id_publishing
172         )
173         db.session.add(new_book)
174         db.session.commit()
175         if not image: flash(message='Добавьте изображение', category='warning')
176         else:
177             filename = secure_filename(f'{new_book.id_book}.jpg')
178             image.save(os.path.join('static/images', filename))
179     except SQLAlchemyError as e:
180         db.session.rollback()
181         flash(message='Произошла ошибка при добавлении. Попробуйте снова.', category='danger')
182         return redirect(url_for('books.new_books'))
183     flash(message='Книга успешно добавлена', category='success')
184     return redirect(url_for('books.manage_books'))
185
186 return render_template("books/new_book.html")
```

## Листинг 11 – Удаление книги

```
188 @bp.route('/<int:id_book>/delete_book', methods = ['POST'])
189 @login_required
190 <> def delete_bokk(id_book):
191     try:
192         book_to_delete = db.session.query(Books).filter_by(id_book=id_book).first()
193         db.session.delete(book_to_delete)
194         db.session.commit()
195         flash(message='Выбранная книга успешно удалена', category='success')
196     except SQLAlchemyError as e:
197         db.session.rollback()
198         flash(message=f'Произошла ошибка при удалении книги: {str(e)}', category='danger')
199
200
201 return redirect(url_for('books.manage_books'))
```

## Листинг 12 – CRUD интерфейс по управлению пользователями

```
10  # Страница Управление пользователями
11  @bp.route('/manage_users')
12  @login_required
13  @admin_or_librarian('admin')
14 <>> def manage_users():
15      users = db.session.query(
16          Users.id_user, Users.login, Users.fio, Roles.role_name
17      ).outerjoin(Roles, Users.id_role == Roles.id_role).all()
18      return render_template(template_name_or_list: "users/manage_users.html", users=users)
19  # Редактирование пользователя
20  @bp.route(rule: '/edit_users', methods=['GET', 'POST'])
21  @login_required
22 <>> def edit_users():
23      id_user = request.args.get(key: 'id_user', type=int)
24      user = db.session.query(Users).filter(Users.id_user == id_user).one()
25      if request.method == 'POST':
26          login = request.form['login']
27          first_name = request.form['name']
28          last_name = request.form['lastName']
29          middle_name = request.form['middleName']
30          role_id = request.form.get('role_id')
31          user.login = login
32          user.fio = f'{last_name} {first_name} {middle_name}'
33          if current_user.is_admin(): user.id_role = role_id
34          try:
35              db.session.commit()
36              flash(message: 'Данные пользователя успешно обновлены', category: 'success')
37              return redirect(url_for('users.manage_users'))
38          except SQLAlchemyError as e:
39              db.session.rollback()
40              flash(message: f'Произошла ошибка при обновлении данных: {str(e)}', category: 'danger')
41          return render_template(template_name_or_list: "users/edit_users.html", user=user)
42  # Удаление пользователя
43  @bp.route(rule: '/<int:id_user>/delete_user', methods=['POST'])
44  @login_required
45 <>> def delete_user(id_user):
46      try:
47          user_to_delete = db.session.query(Users).filter_by(id_user=id_user).first()
48          db.session.delete(user_to_delete)
49          db.session.commit()
50          flash(message: 'Выбранный пользователь успешно удалена', category: 'success')
51      except SQLAlchemyError as e:
52          db.session.rollback()
53          flash(message: f'Произошла ошибка при удалении учетной записи: {str(e)}', category: 'danger')
54          return render_template('users/manage_users.html')
55      return redirect(url_for('users.manage_users'))
```

## Листинг 13 – Авторизация

```
23  @bp.route(rule: '/login', methods=['GET', 'POST'])
24 <>> def login():
25     if request.method == 'POST':
26         login = request.form.get('login')
27         password = request.form.get('password')
28         if login and password:
29             user = db.session.execute(db.select(Users).filter_by(login=login)).scalar_one_or_none()
30             if user and user.check_password(password):
31                 login_user(user)
32                 flash(message: 'Вы успешно аутентифицированы.', category: 'success')
33                 next = request.args.get('next')
34                 return redirect(next or url_for('index'))
35             else:
36                 flash(message: 'Введены неверные логин и/или пароль', category: 'danger')
37             else:
38                 flash(message: 'Заполните все поля', category: 'danger')
39             return render_template('auth/login.html')
```

## Листинг 14 – Регистрация

```
42     @bp.route(rule: '/reg', methods=['GET', 'POST'])
43 <> def reg():
44     if request.method == 'POST':
45         login = request.form.get('login')
46         password = request.form.get('password')
47         confirm_password = request.form.get('confirmPassword')
48         name = request.form.get('name')
49         last_name = request.form.get('lastName')
50         middle_name = request.form.get('middleName')
51         if password != confirm_password:
52             flash(message: 'Пароли не совпадают.', category: 'danger')
53             return render_template('auth/reg.html')
54     try:
55         existing_user = db.session.execute(db.select(Users).filter_by(login=login)).scalar_one_or_none()
56         if existing_user:
57             flash(message: 'Пользователь с таким логином уже существует.', category: 'danger')
58             return render_template('auth/reg.html')
59         new_user = Users(
60             fio=f'{last_name} {name} {middle_name or ""}',
61             login=login, hash_pass=user.set_password(password), id_role=3
62         )
63         db.session.add(new_user)
64         db.session.commit()
65         flash(message: 'Вы успешно зарегистрировались.', category: 'success')
66         login_user(new_user)
67         return redirect(url_for('index'))
68     except SQLAlchemyError as e:
69         db.session.rollback()
70         flash(message: f'Произошла ошибка при сохранении данных: {str(e)}', category: 'danger')
71         return render_template('auth/reg.html')
72     return render_template('auth/reg.html')
```

## Листинг 15 – Статистика популярности книг

```
29     @bp.route('/popular_stats')
30     @login_required
31     @admin_or_librarian('admin')
32 <> def popular_stats():
33     rent_statistics = db.session.query(
34         Books.book_name,
35         Books.author,
36         func.count(Rent.id_book).label('rent_count')
37     ).join(Rent, Books.id_book == Rent.id_book
38         ).group_by(Books.book_name, Books.author
39             ).order_by(desc('rent_count')).all()
40     ranked_books = []
41     current_rank = 0
42     current_count = None
43     for book in rent_statistics:
44         if book.rent_count != current_count:
45             current_rank += 1
46             current_count = book.rent_count
47         ranked_books.append({
48             'rank': current_rank,
49             'book_name': book.book_name,
50             'author': book.author,
51             'rent_count': book.rent_count
52         })
53     return render_template(template_name_or_list: "stats/popular_stats.html", ranked_books=ranked_books)
```

## Листинг 16 – Статистика аренды книг

```
12     # Статистика аренды книг
13     @bp.route('/rent_stats')
14     @login_required
15     @admin_or_librarian('admin')
16 <> def rent_stats():
17     rent_statistics = db.session.query(
18         Books.author,
19         Books.book_name,
20         func.count(Rent.id_book).label('rent_count'),
21         Books.id_book
22     ).join(Rent, Books.id_book == Rent.id_book
23         ).group_by(Books.author, Books.book_name, Books.id_book
24             ).all()
25     return render_template(template_name_or_list: "stats/rent_stats.html", rent_statistics=rent_statistics)
```

## Листинг 17 – Скачивание статистики книг

```
57 @bp.route('/rent_stats_export.csv')
58 @login_required
59 @admin_or_librarian('admin')
60 def rent_stats_export():
61     rent_stats = db.session.query(
62         Books.author, Books.book_name,
63         func.count(Rent.id_book).label('rent_count'), Books.id_book
64     ).join(Rent, Books.id_book == Rent.id_book
65     ).group_by(Books.author, Books.book_name, Books.id_book
66     ).all()
67     result = ''
68     fields = ['book_name', 'author', 'rent_count']
69     result += ','.join(fields) + '\n'
70     for record in rent_stats:
71         result += ','.join([str(getattr(record, field)) for field in fields]) + '\n'
72     return send_file(BytesIO(result.encode()), as_attachment=True, mimetype='text/csv',
73                      download_name='rent_stats_export.csv')
```

## Листинг 19 – Создание таблиц

```
22 def upgrade():
23     # ### commands auto generated by Alembic - please adjust! ###
24     op.create_table('category',
25         *columns: sa.Column(_name_pos: 'id_category', sa.Integer(), nullable=False),
26         sa.Column(_name_pos: 'category_name', sa.String(length=128), nullable=False),
27         sa.PrimaryKeyConstraint(*columns: 'id_category', name=op.f('pk_category'))
28     )
29     op.create_table('publishing',
30         *columns: sa.Column(_name_pos: 'id_publishing', sa.Integer(), nullable=False),
31         sa.Column(_name_pos: 'name_publishing', sa.String(length=128), nullable=False),
32         sa.PrimaryKeyConstraint(*columns: 'id_publishing', name=op.f('pk_publishing'))
33     )
34     op.create_table('roles',
35         *columns: sa.Column(_name_pos: 'id_role', sa.Integer(), nullable=False),
36         sa.Column(_name_pos: 'role_name', sa.String(length=128), nullable=False),
37         sa.PrimaryKeyConstraint(*columns: 'id_role', name=op.f('pk_roles'))
38     )
39     op.create_table('series',
40         *columns: sa.Column(_name_pos: 'id_series', sa.Integer(), nullable=False),
41         sa.Column(_name_pos: 'series_name', sa.String(length=128), nullable=False),
42         sa.PrimaryKeyConstraint(*columns: 'id_series', name=op.f('pk_series'))
43     )
44     op.create_table('books',
45         *columns: sa.Column(_name_pos: 'id_book', sa.Integer(), nullable=False),
46         sa.Column(_name_pos: 'book_name', sa.String(length=128), nullable=False),
47         sa.Column(_name_pos: 'author', sa.String(length=128), nullable=False),
48         sa.Column(_name_pos: 'id_category', sa.Integer(), nullable=False),
49         sa.Column(_name_pos: 'id_series', sa.Integer(), nullable=True),
50         sa.Column(_name_pos: 'id_publishing', sa.Integer(), nullable=False),
51         sa.ForeignKeyConstraint(columns: ['id_category'], refcolumns: ['category.id_category'], name=op.f('fk_books_id_category_category')),
52         sa.ForeignKeyConstraint(columns: ['id_publishing'], refcolumns: ['publishing.id_publishing'], name=op.f('fk_books_id_publishing_publishing')),
53         sa.ForeignKeyConstraint(columns: ['id_series'], refcolumns: ['series.id_series'], name=op.f('fk_books_id_series_series')),
54         sa.PrimaryKeyConstraint(*columns: 'id_book', name=op.f('pk_books'))
55     )
56     op.create_table('users',
57         *columns: sa.Column(_name_pos: 'id_user', sa.Integer(), nullable=False),
58         sa.Column(_name_pos: 'fio', sa.String(length=128), nullable=False),
59         sa.Column(_name_pos: 'Login', sa.String(length=32), nullable=False),
60         sa.Column(_name_pos: 'hash_pass', sa.String(length=256), nullable=False),
61         sa.Column(_name_pos: 'id_role', sa.Integer(), nullable=False),
62         sa.ForeignKeyConstraint(columns: ['id_role'], refcolumns: ['roles.id_role'], name=op.f('fk_users_id_role_roles')),
63         sa.PrimaryKeyConstraint(*columns: 'id_user', name=op.f('pk_users')),
64         sa.UniqueConstraint(*columns: 'login', name=op.f('uq_users_login'))
65     )
66     op.create_table('rent',
67         *columns: sa.Column(_name_pos: 'id_rent', sa.Integer(), nullable=False),
68         sa.Column(_name_pos: 'start_date', sa.Date(), nullable=False),
69         sa.Column(_name_pos: 'end_date', sa.Date(), nullable=False),
70         sa.Column(_name_pos: 'id_book', sa.Integer(), nullable=False),
71         sa.Column(_name_pos: 'id_user', sa.Integer(), nullable=False),
72         sa.ForeignKeyConstraint(columns: ['id_book'], refcolumns: ['books.id_book'], name=op.f('fk_rent_id_book_books')),
73         sa.ForeignKeyConstraint(columns: ['id_user'], refcolumns: ['users.id_user'], name=op.f('fk_rent_id_user_users')),
74         sa.PrimaryKeyConstraint(*columns: 'id_rent', name=op.f('pk_rent'))
75     )
76     # ### end Alembic commands ###
77 data_upgrades()
```

#### **4 ОФОРМЛЕНИЕ ИТОГОВ РАБОТЫ**

Итоговый проект находится по ссылке:

[https://github.com/gemcheck/golodyaev\\_web\\_cur\\_2024.](https://github.com/gemcheck/golodyaev_web_cur_2024)

Сайт был развернут на хостинге Московского Политеха со следующей ссылкой: golodyaev-cur-2024.std-255.ist.mospolytech.ru

## **ЗАКЛЮЧЕНИЕ**

Таким образом, разработка веб-приложения для управления электронной библиотекой является актуальной задачей, обусловленной быстрым ростом объёма электронных материалов и необходимостью их эффективного управления. В рамках данного проекта были проанализированы существующие решения и выявлены их недостатки, что позволило определить ключевые требования пользователей.

Созданное веб-приложение предлагает удобный и интуитивно понятный интерфейс для добавления, организации и поиска электронных материалов, а также поддержку различных типов файлов, что обеспечивает его гибкость и персонализированность. Реализация данного проекта способствует повышению эффективности работы с информацией, удовлетворяя потребности как профессиональной, так и личной сферы.

## **СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ**

1. Документация Flask: <https://flask.palletsprojects.com/en/2.0.x/>
2. Документация MySQL: <https://dev.mysql.com/doc/refman/8.0/en/>
3. SQLAlchemy для Flask: <https://docs.sqlalchemy.org/en/14/>
4. Управление пользователями и аутентификация в Flask: <https://flask-login.readthedocs.io/en/latest/>
5. Flask-User Documentation. (n.d.). Retrieved from <https://flask-user.readthedocs.io/en/latest/>
6. Дополнительные библиотеки и инструменты: <https://getbootstrap.com/docs/5.0/getting-started/introduction/>
7. Stack Overflow. (n.d.). <https://stackoverflow.com/>
8. Курс «Основы веб-приложений» (СИПИ): <https://c1447.c.3072.ru/course/view.php?id=13080>
9. Курс «Разработка веб-приложений» (СИПИ): <https://c1447.c.3072.ru/course/view.php?id=13940>