МИНОБРНАУКИ РОССИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

**ВЫСШЕГО ОБРАЗОВАНИЯ**

**“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”**

**(ФГБОУ ВО «ВГУ»)**

Факультет компьютерных наук

Кафедра программирования и информационных технологий

ComicNova

Разработка онлайн-платформ для чтения и публикации комиксов и книг

Курсовая работа

09.03.04 Программная инженерия

Информационные системы и сетевые технологии

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_\_\_\_\_ *С.Д.Махортов, д.ф.- м.н.,* \_\_.\_\_.20\_\_

Обучающийся \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*М.A.*И. *Абдиладиф, 3 курс, д/о*

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *А.А.* *Вахтин. доцент*

Воронеж 2024

# **Содержание**

[Введение 4](#_Toc168805616)

[1 Постановка задач 5](#_Toc168805617)

[1.1 Требования к разрабатываемой системе 5](#_Toc168805618)

[1.1.1 Функциональные требования 5](#_Toc168805619)

[1.1.1.1 Для клиентов 5](#_Toc168805620)

[1.1.1.2 Для администраторов 6](#_Toc168805621)

[1.1.2 Технические требования 6](#_Toc168805622)

[1.2 Требования к интерфейсу 7](#_Toc168805623)

[1.3 Задачи, решаемые в процессе разработки 7](#_Toc168805624)

[2 Проектирование приложения 8](#_Toc168805625)

[2.1 Требования к приложению. 8](#_Toc168805626)

[2.2 Проектирование интерфейса. 10](#_Toc168805627)

[3 Анализ предметной области 14](#_Toc168805628)

[3.1 Терминология (глоссарий) предметной области 14](#_Toc168805629)

[3.2 Обзор аналогов 15](#_Toc168805630)

[3.2.1 Amazon Kindle 15](#_Toc168805631)

[3.2.2 Bookmate 17](#_Toc168805632)

[3.3 Моделирование системы 19](#_Toc168805633)

[3.3.1 ER-диаграмма 19](#_Toc168805634)

[3.3.2 Диаграмма классов 20](#_Toc168805635)

[3.3.3 Диаграмма состояний 21](#_Toc168805636)

[3.3.4 Диаграмма прецедентов 23](#_Toc168805637)

[3.3.5 Диаграмма последовательности 26](#_Toc168805638)

[3.4 Аналитика веб-приложения 28](#_Toc168805639)

[4 Реализация 31](#_Toc168805640)

[4.1 Средства реализации 31](#_Toc168805641)

[4.2 Реализация серверной части веб-приложения 32](#_Toc168805642)

[4.3 Реализация клиентской части веб-приложения 33](#_Toc168805643)

[Заключение 34](#_Toc168805644)

[Список используемых источников 35](#_Toc168805645)

[Приложение 37](#_Toc168805646)

# **Введение**

В современном мире цифровых технологий и интернет-коммуникаций книги остаются неотъемлемой частью культурного и интеллектуального развития общества. Однако формат их потребления значительно изменился с появлением электронных книг и онлайн-платформ для чтения. Это дало начало новому направлению в индустрии публикаций — разработке и созданию платформ для публикации и чтения книг в цифровом формате. Такие платформы не только упрощают процесс публикации произведений, делая его доступным для широкого круга авторов, но и предоставляют читателям удобный инструмент для доступа к огромному количеству литературных работ.

Цель данной курсовой работы — разработка и анализ функциональности онлайн-платформы для публикации и чтения книг, которая поддерживает как авторов, так и читателей, предоставляя им необходимые инструменты для издания и взаимодействия с контентом. Платформа предлагает возможности для управления профилями пользователей, публикации книг, чтения и аннотирования текстов, а также управления жанровой классификацией и рейтингами популярности произведений.

В ходе исследования будет проведен анализ требований к функционалу подобных платформ, обзор существующих решений на рынке, а также разработка архитектуры и дизайна собственного программного решения. Особое внимание будет уделено вопросам удобства пользовательского интерфейса, безопасности данных и масштабируемости системы. Таким образом, разработка онлайн-платформы для публикации и чтения книг является актуальной задачей, отвечающей требованиям современного литературного процесса и потребностям пользователей, исследующих новые технологические возможности в области чтения и публикации литературы.

# **Постановка задач**

Целью нашего проекта является создание веб-платформы для чтения и публикации книг, которая предоставит пользователям удобный и многофункциональный интерфейс для взаимодействия с контентом и другими пользователями.

## **1.1 Требования к разрабатываемой системе**

### **1.1.1 Функциональные требования**

Функциональные требования к разрабатываемой системе определяют основные возможности и сервисы, которые должны быть реализованы для обеспечения работы веб-платформы для чтения и публикации книг.

#### **1.1.1.1 Для клиентов**

* **Регистрация и авторизация:** Пользователи должны иметь возможность создать аккаунт, указав электронную почту, имя пользователя и пароль. Авторизация должна поддерживать восстановление забытого пароля и изменение текущего пароля.
* **Чтение книг:** Интерфейс чтения должен поддерживать различные форматы книг, предоставлять возможность выбора шрифта, размера текста и цветовой схемы для максимального удобства чтения.
* **Публикация контента:** Пользователи могут загружать собственные книги, указывая название, автора, описание и загружая обложку и файл книги.
* **Поиск и фильтрация:** Необходимы инструменты для поиска книг по названию, автору или ключевым словам, а также фильтрация по жанрам и другим параметрам.

#### **1.1.1.2 Для администраторов**

* **Управление пользователями:** Администраторы могут просматривать список всех зарегистрированных пользователей, добавлять, изменять и удалять пользователей.
* **Управление книгами:** Администраторы могут добавлять новые книги, редактировать информацию о книгах и удалять книги из системы.
* **Управление жанрами:** Администраторы могут добавлять, изменять и удалять жанры книг.
* **Управление ролями**: Администраторы могут управлять ролями пользователей в системе, назначать и изменять роли.

### **1.1.2 Технические требования**

* **База данных:** Использование MySQL для хранения данных пользователей, книг, жанров и ролей.
* **Бэкенд:** Разработка серверной части на Spring Boot, обеспечивающая REST API для взаимодействия с фронтендом.
* **Фронтенд**: Разработка клиентской части с использованием React, включая адаптивный дизайн для различных устройств.
* **Безопасность**: Использование Spring Security для аутентификации и авторизации, защита данных с помощью JWT.

## **1.2 Требования к интерфейсу**

* **Простота и интуитивность:** Интерфейс должен быть понятным и удобным для пользователей всех возрастных групп.
* **Адаптивность:** Интерфейс должен корректно отображаться на различных устройствах, включая мобильные телефоны, планшеты и десктопы.
* **Локализация:** Поддержка русского языка как основного

## **1.3 Задачи, решаемые в процессе разработки**

* **Проектирование архитектуры системы:** Создание модульной и масштабируемой архитектуры для обеспечения высокой производительности и удобства поддержки.
* **Разработка базы данных:** Определение структуры базы данных, которая будет оптимизирована для быстрых запросов и безопасного хранения данных.
* **Реализация бизнес-логики:** Кодирование логики на стороне сервера и клиента для обработки данных и взаимодействия с пользователем.
* **Интеграция безопасности**: Внедрение механизмов аутентификации и авторизации для защиты пользовательских данных и доступа к функциям управления.
* **Тестирование:** Проведение тестирования функциональности и безопасности системы для обеспечения надежности и стабильности работы.

# **2 Проектирование приложения**

## **2.1 Требования к приложению.**

Функциональные требования описывают поведение системы и предоставляемый пользователю функционал. В данном разделе указаны основные функции и особенности, которые должны быть реализованы в приложении Comic Nova. Функциональные требования: Регистрация и авторизация пользователей: Пользователи должны иметь возможность создавать учетные записи и входить в систему с помощью электронной почты и пароля. Должна быть возможность сброса пароля через электронную почту. Профили пользователей: Пользователи должны иметь возможность просматривать и редактировать свои профили, включая информацию о прочитанных и желаемых к прочтению книгах. В профилях должна отображаться история прочитанных книг и оставленных рецензий. Просмотр и поиск книг: Приложение должно предоставлять возможность просмотра списка доступных книг с возможностью фильтрации и сортировки по различным параметрам (название, автор, жанр). Пользователи должны иметь возможность искать книги по ключевым словам. Просмотр подробной информации о книге: Пользователи должны иметь возможность просматривать детальную информацию о книге, включая название, автора, жанр, описание, обложку и рейтинг. Должна быть возможность оставлять и читать рецензии на книги. Чтение книг онлайн: Приложение должно предоставлять возможность чтения книг в режиме онлайн с использованием встроенного читалки, поддерживающей различные форматы файлов. Публикация книг: Авторы должны иметь возможность загружать и публиковать свои книги, включая загрузку обложки и содержимого в формате PDF или ePub. Приложение должно поддерживать модерацию загружаемого контента. Управление пользователями и контентом (административная панель): Администраторы должны иметь возможность управлять пользователями, книгами, жанрами и ролями через административную панель. Должны быть реализованы функции добавления, редактирования и удаления данных. Нефункциональные требования: Кроссплатформенность: Приложение должно корректно работать на всех популярных операционных системах, таких как Windows, Linux и Mac OS. Производительность: Приложение должно быстро загружать страницы и обрабатывать запросы пользователей. Должна быть обеспечена масштабируемость для поддержки большого количества пользователей и данных. Безопасность: Должна быть реализована защита данных пользователей с использованием современных методов шифрования. Для авторизации пользователей должен использоваться JWT (JSON Web Token). Интерфейс пользователя: Приложение должно иметь интуитивно понятный и простой в использовании интерфейс. Должен быть реализован адаптивный дизайн для корректного отображения на различных устройствах. Доступность: Приложение должно быть доступно на нескольких языках, включая русский и английский. Должны быть предусмотрены функции для улучшения доступности для пользователей с ограниченными возможностями.

## **2.2 Проектирование интерфейса.**

На изображении показана главная страница онлайн-платформы "ComicNova" для чтения и публикации комиксов и романов. В верхнем меню расположены разделы: жанры, комиксы, романы и другие

A hand holding a stack of books

Description automatically generated

Рисунок 1- Главная страница сайта

A screenshot of a computer

Description automatically generated

Рисунок 2- записи пользователя

На рисунке 2 Форма для входа в систему.

A screenshot of a computer

Description automatically generated

Рисунок 3 – Профиль пользователя.

На рисунке 3 Отображение информации о пользователе и его активности на платформе.

A screenshot of a book

Description automatically generated

Рисунок 4 – Список всех книг.

На рисунке 8 Список всех книг (AllBooks): Отображение всех доступных книг на платформе.

A screenshot of a computer

Description automatically generated

Рисунок 5 – Публикация книги

На рисунках 5 Форма для загрузки и публикации новой книги

# **3 Анализ предметной области**

## **3.1 Терминология (глоссарий) предметной области**

Администратор (Administrator): Пользователь системы, имеющий расширенные права доступа для управления пользователями, книгами, жанрами и ролями в системе.

Клиент (Client): Конечный пользователь системы, который может просматривать, читать книги и управлять своим профилем.

JWT (JSON Web Token): Метод безопасной передачи информации между клиентом и сервером в виде JSON-объекта, который можно подтвердить и доверять благодаря своей подписи.

REST API: Архитектурный подход к взаимодействию с веб-сервисами, в котором используются стандартные HTTP-методы для получения и отправки данных.

Spring Boot: Фреймворк для разработки веб-приложений на языке Java, упрощающий процесс настройки и публикации приложений.

React: Библиотека JavaScript для создания пользовательских интерфейсов, основанная на компонентах, что позволяет эффективно обновлять и управлять пользовательским интерфейсом при изменении данных.

PDF Viewer: Компонент или инструмент, позволяющий отображать содержимое PDF-файлов в браузере.

Система управления базами данных (MYSQL): Программное обеспечение для создания, управления и манипуляции данными в базе данных.

Функциональные требования (Functional Requirements): Требования, определяющие конкретные действия или функции, которые должна выполнять разрабатываемая система.

Технические требования: Требования, касающиеся программного и аппаратного обеспечения системы, включая необходимые спецификации для поддержки и выполнения функциональных требований

### **3.2 Обзор аналогов**

### **3.2.1 Amazon Kindle**

Электронная библиотека и сервис для чтения книг от Amazon. Предлагает покупку и загрузку электронных книг, а также удобное приложение для чтения книг на различных устройствах

A screenshot of a book

Description automatically generated

Рисунок 6 – Интерфейс сайт Amazon Kindle

.

**Недостатками сайта Amazon Kindle являются:**

* **Стоимость контента:** Хотя Amazon Kindle предлагает широкий выбор книг, многие из них требуют покупки, что может быть дорогостоящим, особенно для частых читателей или студентов.Ограниченное устройство: Пользователи могут столкнуться с ограничениями при использовании файлов купленных книг на других устройствах из-за DRM (Digital Rights Management), что ограничивает использование контента только на продуктах Amazon.
* **Зависимость от интернета:** Для загрузки книг требуется интернет-соединение, что может быть проблемой в местах с ограниченным или отсутствующим доступом к интернету.
* **Монополизация:** Amazon контролирует значительную часть рынка электронных книг, что может стимулировать ценообразование и ограничивать разнообразие доступного контента.
* **Воздействие на владение данными:** Amazon собирает данные о чтении пользователей, что вызывает опасения по поводу приватности и использования этих данных.

### **3.2.2 Bookmate**

Приложение для чтения книг по подписке, предлагающее доступ к широкой библиотеке книг на разных языках, включая аудиокниги и поддержку различных форматов.

A screenshot of a book

Description automatically generated

Рисунок 7 – Интерфейс сайт Bookmate

**Недостатками сайта Amazon Kindle являются:**

* **Ограниченный выбор книг:** В то время как Bookmate предлагает доступ к тысячам книг, его библиотека может не включать некоторые нишевые или специализированные темы, которые доступны в других ресурсах.
* **Качество перевода и публикации:** Некоторые пользователи отмечают проблемы с качеством переводов и опечатками, особенно в книгах на иностранных языках.
* **Требование подписки:** Доступ к большинству контента на платформе требует активной подписки, что может оттолкнуть пользователей, желающих однократно приобрести книгу.
* **Функциональность приложения:** Некоторые пользователи могут столкнуться с техническими проблемами или ограничениями функционала приложения, включая синхронизацию между устройствами.
* **Региональные ограничения:** Доступность сервиса и некоторого контента может быть ограничена в зависимости от региона пользователя, что ограничивает глобальное использование.

## **3.3 Моделирование системы**

### **3.3.1 ER-диаграмма**

A diagram of a computer

Description automatically generated

Рисунок 8 – ER-диаграмма

### **3.3.2 Диаграмма классов**

A screenshot of a computer

Description automatically generated

Рисунок 9 – Диаграмма классов

### **3.3.3 Диаграмма состояний**

A screenshot of a computer screen

Description automatically generated

Рисунок 10 – Диаграмма состояний со стороны Клиента

A screenshot of a computer

Description automatically generated

Рисунок 11 – Диаграмма состояний со стороны Администратора

### **3.3.4 Диаграмма прецедентов**

A diagram of a person's diagram

Description automatically generated

Рисунок 12 – Общая Диаграмма Прецедентов

A diagram of a diagram

Description automatically generated

Рисунок 13 – Диаграмма прецедентов со стороны Клиента

A diagram of a person's diagram

Description automatically generated

Рисунок 14 – Диаграмма прецедентов со стороны Администратора

### **3.3.5 Диаграмма последовательности**

A diagram with text and arrows

Description automatically generated with medium confidence

Рисунок 15 – Диаграмма Последовательности для Клиента

A screenshot of a computer screen

Description automatically generated

Рисунок 16 – Диаграмма Последовательности для Администратора

### **3.4 Аналитика веб-приложения**

Сервис «Яндекс.Метрика» выступает в качестве системы сбора данных об использовании приложений пользователями, поскольку позволяет быстрее и проще всего настраивать метрики для веб-приложений, имеет понятный интерфейс и доступное руководство по использованию.

A screenshot of a computer

Description automatically generated

Рисунок 17 – Общие показатели

**Проcмотры** – Количество просмотров страниц на сайте. Значение "30" указывает на то, что страницы сайта были просмотрены 30 раз.

**Визиты** – Общее количество визитов на сайт. В данном случае "1" означает, что за рассматриваемый период был зафиксирован всего один визит.

**Посетители** – Общее число уникальных посетителей, которые посещали сайт. Здесь также указано "1", что говорит о том, что сайт посетил один уникальный пользователь.

Ниже расположен график Посещаемости сайта, который показывает динамику визитов, посетителей и просмотров страниц на сайте за определённый период времени. На графике выделены следующие категории:

**Визиты** (фиолетовый) – показывает число визитов по дням.

**Посетители** (синий) – отображает количество уникальных посетителей по дням.

**Просмотры** (зелёный) – демонстрирует количество просмотров страниц по дням.

A screenshot of a computer

Description automatically generated

Рисунок 18 – Поведение пользователей на сайте

На предоставленном изображении из Yandex.Metrica представлены три различные метрики, отображающие поведение пользователей на сайте:

**Новые и возвращающиеся пользователи:**

График показывает, что на сайте был зафиксирован 1 новый посетитель в последний измеряемый день (8 июня). На графике отсутствуют данные о возвращающихся пользователях, что может означать, что все посетители в этот день были новыми.

**Время на сайте:**

На втором графике показано время, проведённое пользователем на сайте в последний день. Значение составляет 33 минуты и 59 секунд, что свидетельствует о сравнительно продолжительном взаимодействии пользователя с сайтом. Это может указывать на заинтересованность пользователя контентом сайта или на наличие сложного для навигации интерфейса.

**Глубина просмотра:**

Под графиками расположена метрика "Глубина просмотра" с итоговым значением 30,00. Это означает, что в среднем пользователи просматривали 30 страниц за время своего визита. Высокое значение глубины просмотра может свидетельствовать о высоком интересе к материалам сайта или, например, о проведении исследований или сбора информации на сайте.

# **4 Реализация**

## **4.1 Средства реализации**

Для создания и функционирования веб-приложения были выбраны следующие технологии и инструменты:

Java / Spring Boot: Использование Spring Boot для разработки серверной части приложения, что обеспечивает удобство конфигурации и встроенные решения для безопасности, управления данными и обработки запросов.

MySQL: Выбор MySQL в качестве системы управления базами данных обусловлен его высокой производительностью, надежностью и поддержкой больших объемов данных.

React: Для клиентской части применяется React, позволяющий создавать динамичные и адаптивные пользовательские интерфейсы с использованием компонентного подхода.

JWT (JSON Web Tokens): Применение JWT для аутентификации и авторизации пользователей, что обеспечивает безопасность передачи данных между клиентом и сервером.

Hibernate: Использование Hibernate как ORM-инструмента для упрощения работы с базой данных и сокращения количества написанного кода.

Spring Security: Обеспечение безопасности приложения с помощью Spring Security, включая защиту от XSS и CSRF атак, а также управление сессиями.

Maven: Использование Maven как инструмента для автоматизации сборки проекта, управления зависимостями и сборки конечного продукта.

Git: Применение системы контроля версий Git для управления изменениями в коде, что позволяет упростить совместную работу над проектом.

## **4.2 Реализация серверной части веб-приложения**

Серверная часть веб-приложения была реализована с использованием фреймворка Spring Boot, что позволило обеспечить легкость развертывания и масштабируемость. Основные аспекты реализации:

REST API: Разработка RESTful API для обеспечения взаимодействия между клиентской и серверной частями. API поддерживает стандартные HTTP-методы для CRUD операций (создание, чтение, обновление, удаление данных).

Безопасность: Интеграция Spring Security для аутентификации пользователей через JWT, что позволяет контролировать доступ к определенным ресурсам на основе ролей пользователя.

Обработка данных: Использование Hibernate для маппинга объектов Java на таблицы базы данных и управления транзакциями.

Обработка исключений: Разработка глобальной обработки исключений для управления ошибками и исключениями во всем приложении.

## **4.3 Реализация клиентской части веб-приложения**

Клиентская часть реализована с использованием библиотеки React, что обеспечило модульность и удобство разработки интерфейсов:

Компонентный подход: Разработка функциональных компонентов для различных частей интерфейса (например, формы регистрации, список книг, профиль пользователя).

Маршрутизация: Использование React Router для управления навигацией в одностраничном приложении (SPA).

Взаимодействие с сервером: Использование Axios для отправки HTTP-запросов к серверу, получения данных и их отображения в компонентах.

Управление состоянием: Применение React Hooks и Context API для управления состояниями приложения и данных пользователя на клиенте.

# **Заключение**

Проект разработки веб-платформы для управления библиотечным контентом предоставил комплексное решение для администраторов и конечных пользователей, позволяя им эффективно управлять, просматривать и читать книги онлайн. Использование передовых технологий, включая React для фронтенда и Node.js для бэкенда, обеспечило создание масштабируемой и производительной системы, которая удобна в использовании и доступна с различных устройств.

В ходе проекта были разработаны детальные структурные диаграммы, которые помогли визуализировать архитектуру системы и обеспечить логичную организацию потоков данных. Эти диаграммы способствовали глубокому пониманию взаимодействий внутри платформы и были ключевыми в достижении целей по оптимизации работы системы.

В заключительной фазе проекта было подтверждено, что чётко определённый пользовательский интерфейс и внимание к деталям в пользовательском опыте значительно повышают удовлетворённость пользователей и способствуют их вовлечённости. Планы по дальнейшему развитию включают регулярное тестирование, адаптацию к новым требованиям пользователей и технологическим трендам, что позволит платформе оставаться актуальной в меняющемся цифровом ландшафте.

Таким образом, данный проект не только значительно улучшил процесс управления и доступа к библиотечным ресурсам, но и создал основу для постоянного развития и адаптации платформы, обеспечивая её долгосрочную жизнеспособность и релевантность в сфере цифровых библиотечных услуг.

# **Список используемых источников**

1. React.js Official Documentation [Электронный ресурс]. – Режим доступа: https://reactjs.org/docs/getting-started.html - Заглавие с экрана. (Дата обращения: 08.06.2024).
2. Node.js официальная документация [Электронный ресурс]. – Режим доступа: https://nodejs.org/en/docs/ - Заглавие с экрана. (Дата обращения: 08.06.2024).
3. Express.js Framework Documentation [Электронный ресурс]. – Режим доступа: https://expressjs.com/ - Заглавие с экрана. (Дата обращения: 08.06.2024).
4. MySQL Official Documentation [Электронный ресурс]. – Режим доступа: https://dev.mysql.com/doc/ - Заглавие с экрана. (Дата обращения: 08.06.2024).
5. HTML5 Rocks - A resource for open web HTML5 developers [Электронный ресурс]. Режим доступа: https://www.html5rocks.com/ - Заглавие с экрана. (Дата обращения: 08.06.2024).
6. CSS Tricks - Tips, Tricks, and Techniques on using Cascading Style Sheets [Электронный ресурс]. – Режим доступа: https://css-tricks.com/ - Заглавие с экрана. (Дата обращения: 08.06.2024).
7. Sass Documentation [Электронный ресурс]. – Режим доступа: https://sass-lang.com/documentation/ - Заглавие с экрана. (Дата обращения: 08.06.2024).
8. 12. ГОСТ 7.32-2001 [Электронный ресурс]. Режим доступа:https://kpfu.ru/portal/docs/F1867381138/gost7\_32\_2001.pdf
9. Amazon Kindle [Электронный ресурс]. – Режим доступа: https://www.amazon.com/Kindle-eBooks/b?ie=UTF8&node=154606011 - Заглавие с экрана. (Дата обращения: 08.06.2024).
10. Bookmate [Электронный ресурс]. – Режим доступа: https://bookmate.com/ - Заглавие с экрана. (Дата обращения: 08.06.2024).

# **Приложение**

package com.example.comnovbackend.controller;  
  
import ch.qos.logback.classic.Logger;  
import com.example.comnovbackend.config.JWTGenerator;  
import com.example.comnovbackend.dto.UserDTO;  
import com.example.comnovbackend.models.User;  
import com.example.comnovbackend.models.UserProfile;  
import com.example.comnovbackend.service.UserService;  
import org.slf4j.LoggerFactory;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
import org.springframework.security.core.Authentication;  
import org.springframework.security.core.AuthenticationException;  
import org.springframework.security.core.context.SecurityContextHolder;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UsernameNotFoundException;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.HashMap;  
import java.util.Map;  
  
@RestController  
@RequestMapping("/api/users")  
public class UserController {  
  
 private final UserService userService;  
  
 @Autowired  
 private AuthenticationManager authenticationManager;  
  
 @Autowired  
 private JWTGenerator jwtGenerator;  
 private static final Logger *logger* = (Logger) LoggerFactory.*getLogger*(UserController.class);  
  
  
  
 public UserController(UserService userService) {  
 this.userService = userService;  
 }  
  
 @PostMapping("/register")  
 public ResponseEntity<User> registerUser(@RequestBody UserDTO userDTO) {  
 User user = userService.registerUser(userDTO);  
 return ResponseEntity.*ok*(user);  
 }  
  
 @PostMapping("/login")  
 public ResponseEntity<?> loginUser(@RequestBody UserDTO userDTO) {  
 try {  
 // Authenticate the user  
 Authentication authentication = authenticationManager.authenticate(  
 new UsernamePasswordAuthenticationToken(userDTO.getEmail(), userDTO.getPassword())  
 );  
  
 SecurityContextHolder.*getContext*().setAuthentication(authentication);  
  
 // Generate JWT token  
 String token = jwtGenerator.generateToken(authentication);  
  
 // Create the response body with the token and user ID  
 Map<String, Object> response = new HashMap<>();  
 response.put("accessToken", token);  
 response.put("userId", userDTO.getEmail()); // assuming email as user ID for simplicity  
  
 return ResponseEntity.*ok*(response);  
 } catch (Exception e) {  
 *logger*.error("Login failed for user {}: {}", userDTO.getEmail(), e.getMessage(), e);  
 return ResponseEntity.*status*(HttpStatus.*UNAUTHORIZED*).build();  
 }  
 }  
  
  
  
 @GetMapping("/profile")  
 public ResponseEntity<?> getUserProfile(Authentication authentication) {  
 if (authentication == null || "anonymousUser".equals(authentication.getName())) {  
 return ResponseEntity.*status*(HttpStatus.*UNAUTHORIZED*).body("Authentication is required");  
 }  
  
 UserDetails userDetails = (UserDetails) authentication.getPrincipal();  
 String email = userDetails.getUsername();  
 try {  
 UserDTO user = userService.getUserByEmail(email);  
 System.*out*.println("Sending user data with books: " + user.getBooks()); // Log to check  
 return ResponseEntity.*ok*(user); // Assuming conversion here  
 } catch (Exception e) {  
 *logger*.error("Profile fetch failed for user {}: {}", email, e.getMessage());  
 return ResponseEntity.*status*(HttpStatus.*INTERNAL\_SERVER\_ERROR*).body("Unable to fetch profile");  
 }  
 }  
  
}

package com.example.comnovbackend.controller;  
  
import com.example.comnovbackend.dto.BookDTO;  
import com.example.comnovbackend.models.Book;  
import com.example.comnovbackend.repository.BookRepository;  
import com.example.comnovbackend.service.BookService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.web.multipart.MultipartFile;  
  
import java.security.Principal;  
import java.util.List;  
import java.util.stream.Collectors;  
  
@RestController  
@RequestMapping("/api/books")  
public class BookController {  
  
 @Autowired  
 private BookService bookService;  
  
 @Autowired  
 private BookRepository bookRepository;  
  
 @PostMapping("/publish")  
 public ResponseEntity<?> publishBook(@RequestParam("title") String title,  
 @RequestParam("genre") String genre,  
 @RequestParam("description") String description,  
 @RequestParam("cover") MultipartFile cover,  
 @RequestParam("content") MultipartFile content,  
 Principal principal) {  
 if (principal == null) {  
 return ResponseEntity.*status*(HttpStatus.*UNAUTHORIZED*).body("User must be logged in.");  
 }  
 try {  
 String email = principal.getName(); // Assuming the principal's name is the user's email  
 System.*out*.println("Publishing book for user: " + email);  
 Book book = bookService.publishBook(title, genre, description, cover, content, email);  
 return ResponseEntity.*ok*(book);  
 } catch (Exception e) {  
 System.*out*.println("Error while publishing book: " + e.getMessage());  
 return ResponseEntity.*badRequest*().body("Error while publishing the book: " + e.getMessage());  
 }  
 }  
  
 @GetMapping("")  
 public ResponseEntity<List<BookDTO>> getAllBooks() {  
 try {  
 List<Book> books = bookRepository.findAll();  
 List<BookDTO> bookDTOs = books.stream().map(book -> new BookDTO(book)).collect(Collectors.*toList*());  
 return ResponseEntity.*ok*(bookDTOs);  
 } catch (Exception e) {  
 return ResponseEntity.*status*(HttpStatus.*INTERNAL\_SERVER\_ERROR*).body(null);  
 }  
 }  
  
 @GetMapping("/user")  
 public ResponseEntity<?> getBooksByUserEmail(Principal principal) {  
 if (principal == null) {  
 return ResponseEntity.*status*(HttpStatus.*UNAUTHORIZED*).body("User must be logged in.");  
 }  
 try {  
 String email = principal.getName(); // Assuming the principal's name is the user's email  
 List<Book> books = bookService.getBooksByUserEmail(email);  
 return ResponseEntity.*ok*(books);  
 } catch (Exception e) {  
 System.*out*.println("Error while fetching books for user: " + e.getMessage());  
 return ResponseEntity.*badRequest*().body("Error while fetching books for user: " + e.getMessage());  
 }  
 }  
  
  
  
 @GetMapping("/user/{id}")  
 public ResponseEntity<?> getBooksByUserId(@PathVariable Long id) {  
 try {  
 List<Book> books = bookService.getBooksByUserId(id);  
 return ResponseEntity.*ok*(books);  
 } catch (Exception e) {  
 System.*out*.println("Error while fetching books for user: " + e.getMessage());  
 return ResponseEntity.*badRequest*().body("Error while fetching books for user: " + e.getMessage());  
 }  
 }  
  
 @GetMapping("/{id}")  
 public ResponseEntity<BookDTO> getBookById(@PathVariable Long id) {  
 Book book = bookRepository.findById(id)  
 .orElseThrow(() -> new RuntimeException("Book not found"));  
 return ResponseEntity.*ok*(new BookDTO(book));  
 }  
  
 @GetMapping("/genre/{genre}")  
 public ResponseEntity<?> getBooksByGenre(@PathVariable String genre) {  
 try {  
 List<Book> books = bookService.getBooksByGenre(genre);  
 return ResponseEntity.*ok*(books);  
 } catch (Exception e) {  
 System.*out*.println("Error while fetching books for genre: " + e.getMessage());  
 return ResponseEntity.*badRequest*().body("Error while fetching books for genre: " + e.getMessage());  
 }  
 }  
  
 @GetMapping("/search")  
 public ResponseEntity<List<BookDTO>> searchBooks(@RequestParam String query) {  
 try {  
 List<Book> books = bookService.searchBooks(query);  
 List<BookDTO> bookDTOs = books.stream()  
 .map(book -> new BookDTO(book))  
 .collect(Collectors.*toList*());  
 return ResponseEntity.*ok*(bookDTOs);  
 } catch (Exception e) {  
 return ResponseEntity.*status*(HttpStatus.*INTERNAL\_SERVER\_ERROR*).body(null);  
 }  
 }  
  
  
}

package com.example.comnovbackend.controller;  
  
import ch.qos.logback.classic.Logger;  
import com.example.comnovbackend.config.JWTGenerator;  
import com.example.comnovbackend.dto.UserDTO;  
import com.example.comnovbackend.models.User;  
import com.example.comnovbackend.models.UserProfile;  
import com.example.comnovbackend.service.UserService;  
import org.slf4j.LoggerFactory;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.HttpStatus;  
import org.springframework.http.ResponseEntity;  
import org.springframework.security.authentication.AuthenticationManager;  
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;  
import org.springframework.security.core.Authentication;  
import org.springframework.security.core.AuthenticationException;  
import org.springframework.security.core.context.SecurityContextHolder;  
import org.springframework.security.core.userdetails.UserDetails;  
import org.springframework.security.core.userdetails.UsernameNotFoundException;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.HashMap;  
import java.util.Map;  
  
@RestController  
@RequestMapping("/api/users")  
public class UserController {  
  
 private final UserService userService;  
  
 @Autowired  
 private AuthenticationManager authenticationManager;  
  
 @Autowired  
 private JWTGenerator jwtGenerator;  
 private static final Logger *logger* = (Logger) LoggerFactory.*getLogger*(UserController.class);  
  
  
  
 public UserController(UserService userService) {  
 this.userService = userService;  
 }  
  
 @PostMapping("/register")  
 public ResponseEntity<User> registerUser(@RequestBody UserDTO userDTO) {  
 User user = userService.registerUser(userDTO);  
 return ResponseEntity.*ok*(user);  
 }  
  
 @PostMapping("/login")  
 public ResponseEntity<?> loginUser(@RequestBody UserDTO userDTO) {  
 try {  
 // Authenticate the user  
 Authentication authentication = authenticationManager.authenticate(  
 new UsernamePasswordAuthenticationToken(userDTO.getEmail(), userDTO.getPassword())  
 );  
  
 SecurityContextHolder.*getContext*().setAuthentication(authentication);  
  
 // Generate JWT token  
 String token = jwtGenerator.generateToken(authentication);  
  
 // Create the response body with the token and user ID  
 Map<String, Object> response = new HashMap<>();  
 response.put("accessToken", token);  
 response.put("userId", userDTO.getEmail()); // assuming email as user ID for simplicity  
  
 return ResponseEntity.*ok*(response);  
 } catch (Exception e) {  
 *logger*.error("Login failed for user {}: {}", userDTO.getEmail(), e.getMessage(), e);  
 return ResponseEntity.*status*(HttpStatus.*UNAUTHORIZED*).build();  
 }  
 }  
  
  
  
 @GetMapping("/profile")  
 public ResponseEntity<?> getUserProfile(Authentication authentication) {  
 if (authentication == null || "anonymousUser".equals(authentication.getName())) {  
 return ResponseEntity.*status*(HttpStatus.*UNAUTHORIZED*).body("Authentication is required");  
 }  
  
 UserDetails userDetails = (UserDetails) authentication.getPrincipal();  
 String email = userDetails.getUsername();  
 try {  
 UserDTO user = userService.getUserByEmail(email);  
 System.*out*.println("Sending user data with books: " + user.getBooks()); // Log to check  
 return ResponseEntity.*ok*(user); // Assuming conversion here  
 } catch (Exception e) {  
 *logger*.error("Profile fetch failed for user {}: {}", email, e.getMessage());  
 return ResponseEntity.*status*(HttpStatus.*INTERNAL\_SERVER\_ERROR*).body("Unable to fetch profile");  
 }  
 }  
  
}