Тестовое задание: Разработка RESTful API для управления коллекцией книг с расширенными функциями

Описание

Вам нужно разработать RESTful API для управления коллекцией книг. API должно позволять пользователям добавлять книги, просматривать список книг, обновлять информацию о книгах и удалять книги. Кроме того, API должно иметь функции для управления пользователями и их ролями с использованием битовых масок.

Требования

1. Добавление книги

- HTTP метод: POST

- Эндпоинт: /books

- Тело запроса: JSON с полями title, author, publicationDate, genres

- Ответ: JSON с данными добавленной книги

- Требует аутентификации (только для пользователей с ролью "администратор")

2. Получение списка книг

- HTTP метод: GET

- Эндпоинт: /books

- Ответ: JSON массив с данными всех книг

3. Получение книги по ID

- HTTP метод: GET

- Эндпоинт: /books/:id

- Ответ: JSON с данными книги

4. Обновление информации о книге

- HTTP метод: PUT

- Эндпоинт: /books/:id

- Тело запроса: JSON с полями title, author, publicationDate, genres

- Ответ: JSON с данными обновленной книги

- Требует аутентификации (только для пользователей с ролью "администратор")

5. Удаление книги

- HTTP метод: DELETE

- Эндпоинт: /books/:id

- Требует аутентификации (только для пользователей с ролью "администратор")

6. Регистрация пользователя

- HTTP метод: POST

- Эндпоинт: /users/register

- Тело запроса: JSON с полями username, password, email

- Подтверждение email через письмо

- Ответ: JSON с данными зарегистрированного пользователя

7. Аутентификация пользователя

- HTTP метод: POST

- Эндпоинт: /users/login

- Тело запроса: JSON с полями username, password

- Ответ: JSON с токеном JWT

8. Получение информации о текущем пользователе

- HTTP метод: GET

- Эндпоинт: /users/me

- Ответ: JSON с данными текущего пользователя

- Требует аутентификации

9. Изменение роли пользователя

- HTTP метод: PUT

- Эндпоинт: /users/:id/role

- Тело запроса: JSON с полем role (используйте битовые маски для ролей)

- Ответ: JSON с данными обновленного пользователя

- Требует аутентификации (только для пользователей с ролью "администратор")

Дополнительные требования

- Использовать Node.js и один из фреймворков (Express.js, Koa.js и т.д.)

- Использовать базу данных (например, MongoDB, PostgreSQL или другую на ваше усмотрение)

- Код должен быть хорошо структурирован и читабелен

- Реализация аутентификации и авторизации с использованием JWT

Оценка

Ваше задание будет оцениваться по следующим критериям:

- Правильность работы API

- Чистота и структура кода

- Использование лучших практик разработки