МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ «РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»

Институт информационных наук и технологического образования
Кафедра информационных технологий и электронного обучения

Разработка бэкенда веб-системы для театральной индустрии

На 10 листах

Действует с «23» декабря 2022 г.

СОГЛАСОВАНО
Власова Е.З., проф. кафедры ИТиЭО
_____
Дата
_____

Санкт-Петербург

2022

Оглавление

НАЗНАЧЕНИЕ РАЗРАБОТКИ

Предметом разработки является бэкенд веб-системы для театральной индустрии.

Назначением системы является предоставление возможности:

- просмотра информации о театральных произведениях, постановках и фестивалях

- покупки билетов на внешних сервисах

- оставить отзывы на театральные произведения, постановки и фестивали

ТРЕБОВАНИЯ К ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

ФУНКЦИОНАЛЬНЫЕ ТРЕБОВАНИЯ

Общие требования к системе

Бэкенд должен предоставлять информацию посредством RESTful API. Формат информации – JSON.

Система должна регулировать сессии пользователей, выдавать информацию в зависимости от аутентификации и роли пользователя.

Пользователи должны иметь возможность получать информацию об их личных кабинетах.

Ошибки выводятся в формате JSON по схеме, показанной на рисунке 1.

Рисунок 1. Схема объекта ошибки

Требования к аутентификации и авторизации

Пользователю должна быть предоставлена возможность авторизации и аутентификации. Метод проверки подлинности – JWT.

В системе должно быть три роли – администратор, модератор и обычный пользователь. В будущем может появиться четвёртая роль – автор произведений. Если пользователь только что зарегистрировался, ему должна выдаваться роль обычного пользователя.

Бэкенд должен создавать, подписывать и отправлять JWT клиенту.

Модераторы и администраторы могут добавлять произведения.

Требования к подсистеме пьес

Используя маршруты подсистемы пьес, пользователь API должен иметь возможность получать информацию о пьесах, узнать авторов пьес, смотреть отзывы на пьесы, а также фильтровать информацию по критериям:

- год создания

- наличие текста

- наличие постановок

- тип произведения

Фильтры принимаются в качестве query-параметров.

Модераторы и администраторы должны иметь возможность пользоваться маршрутами для добавления, редактирования и удаления информации.

Примерный список маршрутов показан на рисунке 2.

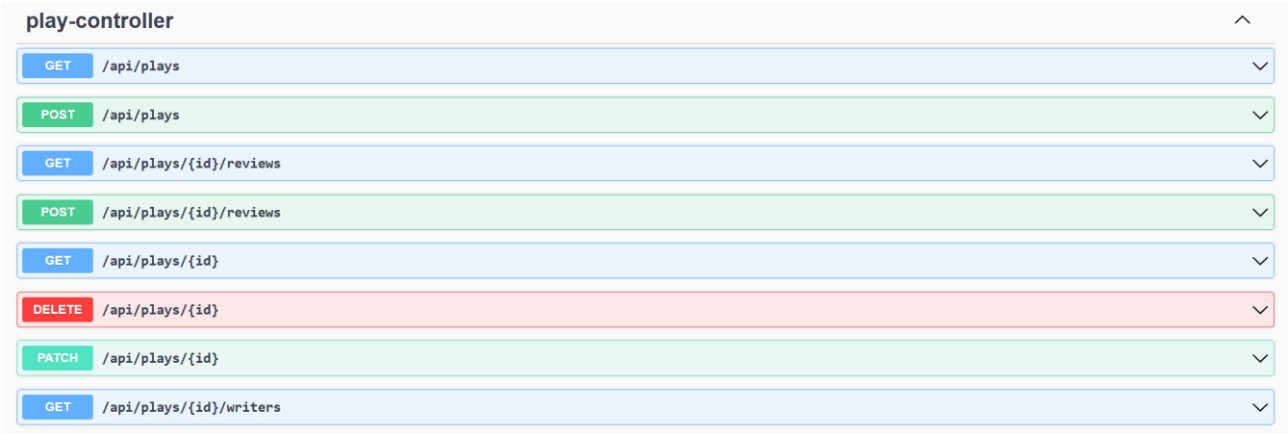

Рисунок 2. Список маршрутов подсистемы пьес

Примерная выводимая информация в формате JSON показана на рисунке 3.

Рисунок 3. Схема данных

Требования к подсистеме авторов

Используя маршруты подсистемы пьес, пользователь API должен иметь возможность получать информацию об авторах и смотреть их пьесы.

Допускается фильтрация результатов по различным критериям.

Модераторы и администраторы должны иметь возможность пользоваться маршрутами для добавления, редактирования и удаления информации.

Примерный список маршрутов показан на рисунке 4.

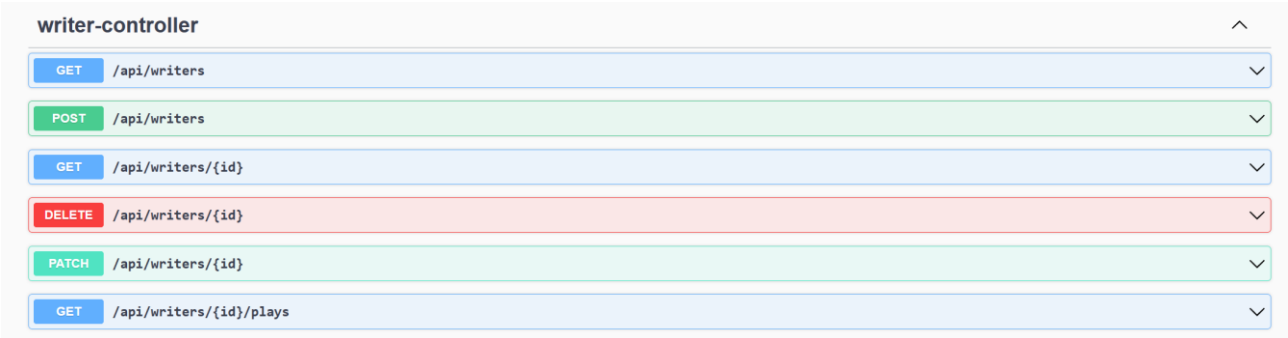

Рисунок 4. Список маршрутов подсистемы авторов

Примерная выводимая информация в формате JSON показана на рисунке 5.

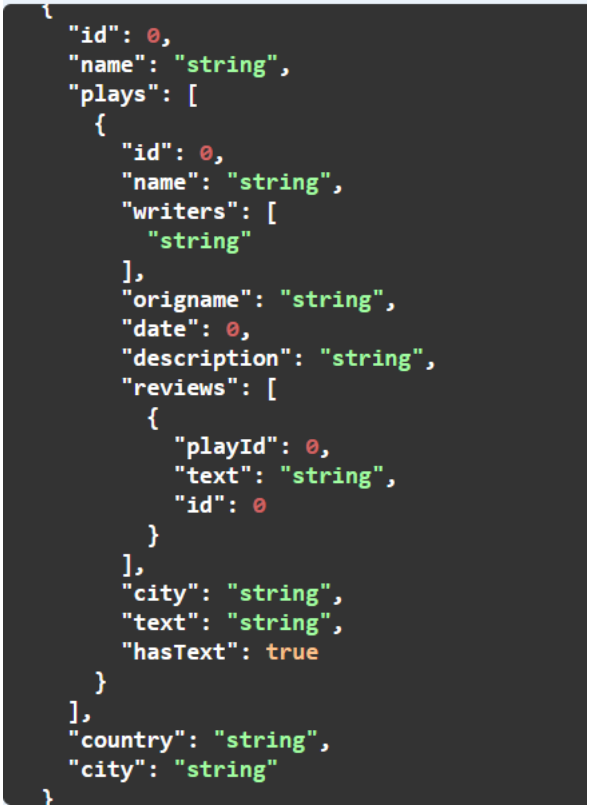

Рисунок 5. Схема данных

Требования к подсистеме отзывов

Авторизованные пользователи должны иметь возможность отправлять отзывы к пьесам, используя оценки по различным критериям.

Отзывы также позволяется редактировать и удалять.

Модераторы и администраторы должны иметь возможность удалять отзывы.

Примерный список маршрутов подсистемы отзывов показан на рисунке 6.

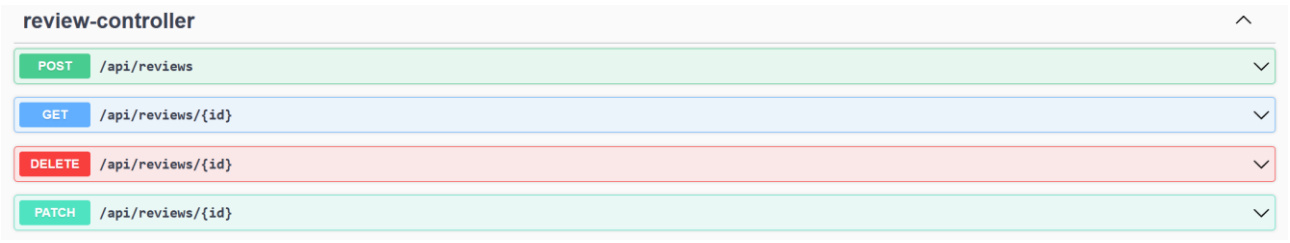

Рисунок 6. Список маршрутов подсистемы отзывов

ТРЕБОВАНИЯ К ВИДАМ ОБЕСПЕЧЕНИЯ

Требования к архитектуре сервиса

Сервис должен работать, используя клиент-серверную архитектуру, посредством RESTful API. Фронтенд должен отправлять запросы пользователей на сервер, а сервер отвечать, используя JSON-формат.

В качестве системной архитектуры приложения позволяется выбрать Clean Architecture. Должны быть реализованы классы контроллеров, которые принимают запросы пользователей и отправляют ответы. Контроллеры в свою очередь вызывают методы сервисов. В классах сервисов должна быть реализована бизнес-логика приложения. В качестве доступа к базе данных следует использовать паттерн «Репозиторий». Методы репозиториев используют генерацию SQL-запросов на основе классов моделей.

Классы данных должны работать по схеме Data Transfer Object (DTO) -> Model -> ViewModel.

Сервис должен принимать данные по схеме DTO и конвертировать их в модель. Отправлять ответ сервер должен в виде ViewModel, которые конвертируются из моделей.

Требования к базе данных

В качестве СУБД следует выбрать PostgreSQL. Все данные моделей должны храниться в БД.

Требования к стеку

Сервис должен быть написан на языке программирования Kotlin.

В качестве фреймворка следует выбрать Spring Boot.

Фреймворк данных – Spring Data JPA, провайдер ORM – Hibernate.

Для написания миграций для базы данных нужно выбрать Liquibase.

Для обеспечения безопасности, авторизации и аутентификации следует выбрать Spring Security.

В качестве системы сборки проекта нужно выбрать Gradle.

Для логгирования фреймворк Spring предоставляет библиотеку Logback, для JSON-маппинга – Jackson.

Контейнер сервлетов – Apache Tomcat.

Для Unit-тестирования можно выбрать фреймворк JUnit, для Mock-тестирования – Mockito.

Требования к ПО на стороне сервера

- ОС – на выбор: Ubuntu Server 20.04 или 22.04, Debian 11

- Прокси-сервер – Nginx 1.22 или выше

- СУБД – PostgreSQL 14 или выше

Требования к техническому обеспечению

- Процессор – любой современный Intel Xeon или аналогичный процессор производства AMD

- Оперативная память – 2 Гб RAM DDR4 или лучше

- Хранение данных – 25 Гб SSD или выше

ЭТАПЫ РАЗРАБОТКИ

Этап	**Содержание работ**
1. Подготовительный этап	1.1. постановка задачи; 1.2. разработка основных положений веб-сайта; 1.3. разработка функциональной структуры и перечня задач; 1.4. постановка основных требований к разработке; 1.5. согласование и утверждение технического задания; 1.6. подготовка тестового сервера.
2. Техническое проектирование	2.1. уточнение состава техники и ПО 2.2. проектирование архитектуры сайта, в том числе маршрутов 2.3. проектирование базы данных
3. Разработка программной части	3.1. разработка программного кода на языке Kotlin (фреймворк Spring Boot) для каждой поставленной задачи; 3.2. реализация паттерна Clean Architecture 3.3. реализация механизмов защиты
4. Подготовка к эксплуатации	4.1. дебаггинг; 4.2. написание тестов (Unit и Mock); 4.3. подготовка production-среды 4.4. разворачивание сервиса на сервере
5. Ввод в эксплуатацию	5.1. публикация сайта 5.2. продвижение сайта 5.3. добавление рекламных интеграций