Техническое задание на разработку приложения FleaMarket

Общая информация

Название проекта: FleaMarket

Цель проекта: Создание RESTful веб-приложения для управления продавцами и их товарами на виртуальной торговой площадке.

Исполнитель: Можейко Дмитрий

1. Введение

Данное техническое задание описывает требования к разработке веб-приложения FleaMarket, предназначенного для управления продавцами и их товарами на виртуальной торговой площадке. Приложение должно обеспечить возможность управления продавцами, добавления, редактирования и удаления товаров, а также просмотра информации о продавцах и их товарах.

2. Бизнес-модель

2.1 Продавцы

1. **Регистрация продавца:**
   - Продавец должен иметь уникальное имя (username).
   - Продавец может зарегистрироваться, указав свое имя. Имя не может быть пустым или уже занятым.
2. **Управление продавцами:**
   - Возможность изменения имени продавца.
   - Удаление продавца.
3. **Просмотр информации о продавцах:**
   - Возможность получения списка всех зарегистрированных продавцов.
   - Возможность получения информации о конкретном продавце и его товарах.

2.2 Товары

1. **Управление товарами:**
   - Добавление товара к продавцу. Товар должен иметь уникальное имя в рамках одного продавца.
   - Изменение имени товара.
   - Удаление товара.
2. **Просмотр информации о товарах:**
   - Возможность получения списка всех товаров конкретного продавца.
   - Возможность получения информации о продавце, которому принадлежит товар (по ID товара).

3. Функциональные требования

Описание эндпоинтов

1. Получение всех продавцов

- **Метод:** `GET`
- **URL:** `/sellers/getAllSellers`
- **Описание:** Возвращает список всех зарегистрированных продавцов.
- **Принимает:** Нет входных параметров.
- **Выбрасывает:** Нет исключений.

- **Возвращает:** JSON-список всех продавцов.

2. Создание продавца

- **Метод:** `POST`
- **URL:** `/sellers/crteateSeller`
- **Описание:** Создает нового продавца с указанным именем пользователя.
- **Принимает:** Параметр запроса `seller` - имя нового продавца.
- **Выбрасывает:** `IllegalArgumentException` если имя пустое, `SellerTakenException` если имя уже занято.
- **Возвращает:** Сообщение об успешном создании продавца.

3. Изменение имени продавца

- **Метод:** `PATCH`
- **URL:** `/sellers/updateSellerName`
- **Описание:** Изменяет имя существующего продавца.
- **Принимает:** Параметры запроса `oldUsername` - текущее имя продавца, `newUsername` - новое имя продавца.
- **Выбрасывает:** `SellerNotFoundException` если продавец с текущим именем не найден, `IllegalArgumentException` если новое имя пустое, `SellerTakenException` если новое имя уже занято.
- **Возвращает:** Сообщение об успешном изменении имени продавца.

4. Удаление продавца

- **Метод:** `DELETE`
- **URL:** `/sellers/deleteSeller`
- **Описание:** Удаляет продавца по его имени пользователя.
- **Принимает:** Параметр запроса `seller` - имя пользователя продавца.
- **Выбрасывает:** `SellerNotFoundException` если продавец с данным именем не найден.
- **Возвращает:** Сообщение об успешном удалении продавца.

5. Получение всех товаров продавца

- **Метод:** `GET`
- **URL:** `/sellers/getAllSellerPtroducts`
- **Описание:** Возвращает список всех товаров, ассоциированных с продавцом.
- **Принимает:** Параметр запроса `seller` - имя пользователя продавца.
- **Выбрасывает:** `SellerNotFoundException` если продавец с данным именем не найден.
- **Возвращает:** JSON-список товаров продавца.

6. Добавление товара к продавцу

- **Метод:** `POST`
- **URL:** `/sellers/addProduct`
- **Описание:** Добавляет новый товар к указанному продавцу.
- **Принимает:** Параметры запроса `product` - имя товара, `seller` - имя пользователя продавца.
- **Выбрасывает:** `IllegalArgumentException` если имя товара пустое, `ProductTakenException` если товар уже добавлен, `SellerNotFoundException` если продавец с данным именем не найден.
- **Возвращает:** Сообщение об успешном добавлении товара.

7. Удаление товара

- **Метод:** `DELETE`
- **URL:** `/sellers/deleteProduct`
- **Описание:** Удаляет товар из ассортимента продавца.
- **Принимает:** Параметры запроса `product` - имя товара, `seller` - имя пользователя продавца.
- **Выбрасывает:** `ProductNotFoundException` если товар с данным именем не найден, `SellerNotFoundException` если продавец с данным именем не найден.
- **Возвращает:** Сообщение об успешном удалении товара.

8. Изменение имени товара

- **Метод:** `PATCH`
- **URL:** `/sellers/updateProduct`
- **Описание:** Изменяет имя товара, ассоциированного с продавцом.
- **Принимает:** Параметры запроса `oldProductName` - текущее имя товара, `newProductName` - новое имя товара, `seller` - имя пользователя продавца.
- **Выбрасывает:** `ProductNotFoundException` если товар с текущим именем не найден, `ProductTakenException` если новое имя товара уже занято, `SellerNotFoundException` если продавец с данным именем не найден, `IllegalArgumentException` если имя товара пустое.
- **Возвращает:** Сообщение об успешном изменении имени товара.

9. Получение продавца по ID товара

- **Метод:** `GET`
- **URL:** `/products/getProcuctSeller`
- **Описание:** Возвращает продавца, ассоциированного с указанным товаром.
- **Принимает:** Параметр запроса `productId` - ID товара.
- **Выбрасывает:** `ProductNotFoundException` если товар с данным ID не найден.
- **Возвращает:** Продавца, ассоциированного с указанным товаром.

4. Нефункциональные требования

1. **Документация:** Встроенная документация API через Swagger. Просмотр документации доступен после запуска приложения по url: http://localhost:8080/swagger-ui/index.html#/

5. Тестирование

1. **Unit-тесты:** Должны быть покрыты 100% бизнес-логики.

2. Для написания тестов использовать **Junit** и **Mockito**.
3. Тесты запускаются при сборке проекта.

6. Отчеты Sonar

Отчеты Sonar Cloud к проекту можно посмотреть по url:https://sonarcloud.io/summary/overall?id=k9targex_FleaMarket

7. Обработка ошибок

Приложение содержит глобальный обработчик ошибок, который обрабатывает различные исключения и возвращает соответствующие HTTP-ответы:

- **400 Bad Request**: обрабатывается при неправильных запросах (например, отсутствует обязательный параметр).
- **404 Not Found**: обрабатывается, когда ресурс не найден (например, продавец или товар не существует).
- **409 Conflict**: обрабатывается при конфликтных ситуациях (например, имя продавца или товара уже занято).
- **405 Method Not Allowed**: обрабатывается при использовании неподдерживаемого HTTP-метода.
- **500 Internal Server Error**: обрабатывается при любых других необработанных исключениях.

8. Технологии

Проект должен быть реализован с использованием следующих технологий и инструментов:

- Java
- Spring Boot
- Hibernate/JPA для работы с базой данных
- PostgreSQL в качестве системы управления базами данных
- Swagger для документации API
- Postman для тестирования API