WEB. Разработка с Nest JS.

Урок 21. Практика. Интеграция сервиса отзывов.

Техническое задание на разработку микросервиса отзывов о товарах.

Цель.

Разработать микросервис для управления отзывами о товарах, который будет хранить и обрабатывать отзывы пользователей о товарах. Сервис должен работать независимо от других сервисов и использовать свою собственную базу данных MongoDB для хранения данных.

Требования к сервису.

Формат отзыва.

Отзыв должен иметь следующую структуру:

```
{
    "user_id": "string",
    "good_id": "string",
    "report": {
        "title": "string",
        "report_info": "string",
        "rating": 5
    }
}
```

- user_id: Идентификатор пользователя, оставившего отзыв (строка).
- good_id: Идентификатор товара, к которому относится отзыв (строка).
- report.title: Заголовок отзыва (строка).
- report.report_info: Текст отзыва (строка).
- report.rating: Оценка товара (целое число от 1 до 5).

База данных.

Для хранения данных используется MongoDB.

У каждого отзыва должна быть уникальная запись в базе данных.

MongoDB должна быть локализована для использования только этим сервисом (доступ к ней не предоставляется другим сервисам).

Функциональные требования.

Создание отзыва.

Сервис должен позволять пользователям создавать новые отзывы через API. Отзыв содержит user_id, good_id, и сам отчет с заголовком, текстом и рейтингом.

Получение отзывов.

Сервис должен предоставлять возможность получать список отзывов по товарам через API. Опционально можно запрашивать все отзывы или отзывы конкретного товара по его good_id.

Удаление отзыва.

Сервис должен предоставлять возможность удалять отзывы через API по идентификатору отзыва.

Редактирование отзыва.

Сервис должен позволять пользователям редактировать существующие отзывы.

Ограничения.

- Сервис не должен обращаться к другим сервисам напрямую. Например, данные о пользователях и товарах, такие как их названия или описания, не запрашиваются у других микросервисов (пользователи и товары).
- Все необходимые данные (user_id, good_id) должны быть переданы при создании или редактировании отзыва.
- Вся логика работы с данными должна быть сосредоточена в этом сервисе.

API Сервиса: Определение маршрутов для взаимодействия с отзывами через REST API.

POST /reviews: Создание нового отзыва.

Тело запроса:

```
{
    "user_id": "string",
    "good_id": "string",
    "report": {
        "title": "string",
        "report_info": "string",
        "rating": 5
    }
}
```

Ответ: 201 Created с данными нового отзыва или сообщение об ошибке.

GET /reviews: Получение списка всех отзывов.

Ответ: Список всех отзывов в формате:

```
[
    {
        "user_id": "string",
        "good_id": "string",
        "report": {
            "title": "string",
            "report_info": "string",
            "rating": 5
        }
    },
    ...
]
```

GET /reviews/{good_id}: Получение отзывов о конкретном товаре по его идентификатору good_id.

Ответ: Список отзывов по этому товару в формате:

```
[
    {
        "user_id": "string",
        "good_id": "string",
        "report": {
            "title": "string",
            "report_info": "string",
            "rating": 5
        }
    },
    ...
]
```

PUT /reviews/{review_id}: Редактирование существующего отзыва по его идентификатору review_id.

Тело запроса: Может содержать измененные поля отзыва, например:

```
{
    "report": {
        "title": "Updated title",
```

```
        "report_info": "Updated review text",
        "rating": 4
    }
}
```

Ответ: 200 OK или сообщение об ошибке.

DELETE /reviews/{review_id}: Удаление отзыва по его идентификатору review_id.

Ответ: 200 OK или сообщение об ошибке.

Архитектурные требования.

Сервис должен быть разработан как микросервис, который может быть масштабируемым и легко интегрируемым с существующей микросервисной архитектурой.

NestJS должен быть использован как основной фреймворк для разработки.

MongoDB должна быть настроена как отдельный контейнер в составе инфраструктуры проекта с использованием docker-compose. Этот контейнер должен быть доступен только для сервиса отзывов.

Интеграция с инфраструктурой.

Сервис отзывов должен быть развернут в контейнере с использованием Docker.

docker-compose.yml должен быть обновлен для добавления нового контейнера для MongoDB и самого сервиса отзывов.

Пример конфигурации docker-compose.yml:

```
version: '3'
services:
  reviews-service:
    build: ./reviews-service
    ports:
      - "3003:3000"
    environment:
      - MONGODB_URI=mongodb://mongodb:27017/reviewsdb
    depends_on:
      - mongodb
  mongodb:
    image: mongo:latest
    ports:
      - "27017:27017"
    volumes:
      - ./data/db:/data/db
```

Безопасность.

Для взаимодействия с базой данных MongoDB должен быть настроен пароль для доступа и использован безопасный механизм хранения данных.

Доступ к MongoDB должен быть ограничен только для сервиса отзывов.

Тестирование.

После завершения разработки необходимо провести тестирование API с помощью инструментов, таких как Postman.

Необходимо убедиться, что создаются, редактируются и удаляются отзывы, а данные корректно сохраняются в MongoDB.

Ожидаемый результат.

Разработанный микросервис для управления отзывами, который:

- Хранит и обрабатывает отзывы о товарах.
- Работает с собственной базой данных MongoDB.
- Не взаимодействует с другими микросервисами напрямую.
- Имеет готовый набор API для работы с отзывами (создание, получение, редактирование, удаление).