САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа № 3

Выполнил:

Кононов Степан

Группа

К32392

Проверил:
Добряков Д. И.

Санкт-Петербург

2024г.

Лабораторная работа 3

Задание

1. Необходимо реализовать отдельный микросервис, выполняющий какую-либо содержательную функцию из всего арсенала функций вашего приложения.
2. Необходимо упаковать ваше приложение в docker-контейнеры и обеспечить сетевое взаимодействие между различными частями вашего приложения, а также настроить общение микросервисов между собой посредством RabbitMQ. Делать это можно как с помощью docker-compose так и с помощью docker swarm. При разумном использовании swirl вы получите дополнительные баллы.

Ход работы

Auth Service (auth-service):

- Реализован на базе Express.js.
- Имеет два основных эндпойнта для регистрации (/auth/register) и авторизации (/auth/login).
- При успешной авторизации генерируется JWT-токен, который затем используется для доступа к функционалу основного сервиса.
- RabbitMQ используется для отправки данных пользователя основному сервису через очередь сообщений auth_queue.

Main Service (main-service):

- Реализован на базе Express.js.
- Имеет несколько эндпойнтов для управления продуктами и покупками:
  - Создание продукта, правка информации, удаление и пр.
  - Получение списка товаров с учетом остатков и скидок.

    - Обработка операций покупки товара и управление складскими единицами.
- Для проверки прав пользователя на выполнение операций используется middleware, который отправляет запрос в очередь сообщения через RabbitMQ в auth-service для валидации токена.

RabbitMQ:

- Настроен для обработки обменов между auth-service и main-service.
- Используются очереди для передачи сообщений о данных пользователей и других внутренних данных.

Процесс контейнеризации

Compose

```yaml
version: '3'

services:
  rabbitmq:
    image: rabbitmq
    expose:
      - 5672

  auth-service:
    build:
      context: ./
      dockerfile: Dockerfile_auth
    tty: true
    volumes:
      - ./auth-service/db:/app/db
    ports:
      - "3001:3001"
    command: sh -c "npm start"
    depends_on:
      - rabbitmq

  main-service:
    build:
      context: ./
      dockerfile: Dockerfile_main
    tty: true
    volumes:
      - ./main-service/db:/app/db
    ports:
      - "3000:3000"
    command: sh -c "npm start"
    depends_on:
      - rabbitmq
      - auth-service
```

Auth сервис

```
FROM node:20

WORKDIR /usr/src/app

COPY ./auth-service/package*.json ./

RUN npm ci
RUN npm install sqlite3 --save

COPY ./auth-service/ .

EXPOSE 3001
CMD [ "npm", "start" ]
```

Main сервис

```
FROM node:20

WORKDIR /usr/src/app

COPY ./main-service/package*.json ./

RUN npm ci
RUN npm install sqlite3 --save

COPY ./main-service/ .

EXPOSE 3000
CMD [ "npm", "start" ]
```

Запускаем через `docker-compose up –build`

Вывод

В ходе лабораторной работы были развёрнуты микросервисы для авторизации и управления товарами с согласованной бизнес-логикой. Использование Docker позволило упростить процесс развертывания и поддержки сервиса, а RabbitMQ позволил обеспечить согласованное взаимодействие между микросервисами. Подобная архитектура решений может быть успешно применена в реальных проектах для повышения отказоустойчивости и масштабируемости системы.