САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа #4

Выполнил:

Игнатьев Алексей

К3340

Проверил:
Добряков Д. И.

Санкт-Петербург

2025 г.

Задание:
реализовать Dockerfile для каждого сервиса;
написать общий docker-compose.yml;
настроить сетевое взаимодействие между сервисами.

Ход работы

1. **Реализация Dockerfile для каждого сервиса**

В ходе задачи реализованы dockerfile для каждого сервиса

```
FROM ubuntu:24.04

ARG version=22

RUN apt-get update -y && apt-get install curl unzip -y \
    && curl -fsSL https://fnm.vercel.app/install | bash -s -- --install-dir
'./fnm' \
    && cp ./fnm/fnm /usr/bin && fnm install $version && apt-get install npm
netcat-traditional -y \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

# Set working directory
WORKDIR /app

ENV NODE_OPTIONS="--max-old-space-size=4096"
# Copy package.json and package-lock.json
COPY package*.json ./

# Install dependencies
RUN npm cache clean --force && npm install --legacy-peer-deps

# Copy the rest of the application code
COPY . .

# Generate Prisma Client code
RUN npx prisma generate

# Build application
RUN npm run build

EXPOSE 8003
EXPOSE 587


CMD ["npm", "run", "start:prod"]
```

MainService

```
FROM ubuntu:24.04

ARG version=20

RUN apt-get update -y && apt-get install curl unzip -y \
    && curl -fsSL https://fnm.vercel.app/install | bash -s -- --install-dir
```

```
'./fnm' \
    && cp ./fnm/fnm /usr/bin && fnm install $version && apt-get install npm
netcat-traditional -y \
    && apt-get clean \
    && rm -rf /var/lib/apt/lists/*

# Set working directory
WORKDIR /app

ENV NODE_OPTIONS="--max-old-space-size=4096"
# Copy package.json and package-lock.json
COPY package*.json ./

# Install dependencies
RUN npm cache clean --force && npm install --legacy-peer-deps

# Copy the rest of the application code
COPY . .

# Generate Prisma Client code
RUN npx prisma generate

# Build application
RUN npm run build

EXPOSE 8003
EXPOSE 587

COPY entrypoint.sh /app/entrypoint.sh
RUN chmod +x /app/entrypoint.sh

CMD ["/bin/bash", "/app/entrypoint.sh"]
```

2. Реализация общего docker compose файла

```
3. services:
     admin-service:
       container_name: admin-service
       restart: always
       build:
         context: ./admin-service
         dockerfile: Dockerfile
       ports:
         - '${ADMIN_SERVICE_PORT}:${ADMIN_SERVICE_PORT}'
       env_file:
         - '.env.build'
       depends_on:
         - postgres
         - minio
         - redis
         - main_service
       networks:
         - auction-network

     main_service:
       container_name: main_service
       restart: always
       build:
         context: ./main-service
```

```
      dockerfile: Dockerfile
    ports:
      - '8000:8000'
    env_file:
      - '.env.build'
    depends_on:
      - postgres
      - minio
      - redis
    networks:
      - auction-network

  promocode_service:
    container_name: promocode_service
    restart: always
    build:
      context: ./promocode-service
      dockerfile: Dockerfile
    ports:
      - '${PROMOCODE_SERVICE_PORT}:${PROMOCODE_SERVICE_PORT}'
    env_file:
      - '.env.build'
    depends_on:
      - main_service
    networks:
      - auction-network

  bonus_service:
    container_name: bonus_service
    restart: always
    build:
      context: ./bonus-service
      dockerfile: Dockerfile
    ports:
      - '${BONUS_SERVICE_PORT}:${BONUS_SERVICE_PORT}'
    env_file:
      - '.env.build'
    depends_on:
      - main_service
    networks:
      - auction-network

  postgres:
    container_name: auction_postgres
    image: postgres:latest
    env_file:
      - '.env.build'
    environment:
      POSTGRES_DB: ${POSTGRES_DB}
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
    ports:
      - 5432:5432
    volumes:
      - auction-pgdata:/var/lib/postgresql/data
    healthcheck:
      test: ['CMD-SHELL', 'pg_isready -U ${POSTGRES_USER} -d ${POST-
GRES_DB}']
      interval: 10s
```

```
        timeout: 5s
        retries: 5
        start_period: 10s
      restart: unless-stopped
      networks:
        - auction-network

    minio:
      container_name: auction_minio
      image: minio/minio
      restart: always
      ports:
        - 9000:9000
        - 9001:9001
      volumes:
        - auction-minio-storage:/data
      env_file:
        - '.env'
      environment:
        MINIO_ROOT_USER: ${POSTGRES_USER}
        MINIO_ROOT_PASSWORD: ${POSTGRES_PASSWORD}
        MINIO_ACCESS_KEY: ${MINIO_ACCESS_KEY}
        MINIO_SECRET_KEY: ${MINIO_SECRET_KEY}
      command: server --console-address ":9001" /data
      networks:
        - auction-network

    redis:
      container_name: auction_redis
      image: redis:latest
      restart: always
      healthcheck:
        test: ['CMD', 'redis-cli', 'ping']
        interval: 10s
        timeout: 60s
        retries: 5
        start_period: 10s
      networks:
        - auction-network

  volumes:
    auction-pgdata:
    auction-pgadmin-data:
    auction-minio-storage:

  networks:
    auction-network:
      driver: bridge
```

Вывод

- В ходе выполнения лабораторной работы выполнена реализация микросервисов в dockerfile

- Написан общий docker compose файл