**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"]

1. **Реализация общего docker compose файла**
2. 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 ${POSTGRES\_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