

Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

О Т Ч Е Т
по лабораторной работе №4
«Контейнеризация написанного приложения средствами Docker»
курса «Бэкенд разработка»

Выполнили:

Бахарева М.А., К3342

Привалов К.А., К3342

Проверил:

Добряков Д.И.

Санкт-Петербург,

2025

Содержание

Ход работы.....	3
Задание.....	3
Основная часть.....	4
Создание Dockerfile.....	4
Написание Docker Compose файла.....	4
Вывод.....	8

Ход работы

Задание

Лабораторная работа заключается в контейнеризации приложения средствами Docker. Поставлены следующие задачи:

- Реализовать Dockerfile для сервисов;
- Написать Docker Compose файл;
- Настроить сетевое взаимодействие между сервисами.

Основная часть

Создание Dockerfile

Для того, чтобы наше приложение работало в любой среде, необходимо написать Dockerfile, инструкцию, по которой выполняется установка зависимостей, других процедур и запуск приложения.

Для всех сервисов получился общий Dockerfile:

```
FROM node:22-alpine AS build

WORKDIR /app

COPY package.json package-lock.json* ./
RUN npm install --production=false

COPY . .

RUN npm run build

FROM node:22-alpine

WORKDIR /app

RUN apk add --no-cache bash

COPY --from=build /app/dist ./dist
COPY --from=build /app/node_modules ./node_modules
COPY --from=build /app/package.json ./

EXPOSE 3000

COPY wait-for-it.sh ./

CMD ["sh", "-c", "./wait-for-it.sh db:5432 -- node dist/index.js"]
```

Написание Docker Compose файла

Теперь важно, чтобы наши сервисы могли взаимодействовать между собой. Создадим compose проект, который включает в себя базу данных,

шлюз и микросервисы. Настроим вольюм для персистентного хранения данных БД и сети для взаимодействия сервисов:

```
services:
  db:
    image: postgres:17-alpine
    container_name: postgres
    environment:
      POSTGRES_USER: ${POSTGRES_USER}
      POSTGRES_PASSWORD: ${POSTGRES_PASSWORD}
      POSTGRES_DB: ${POSTGRES_DB}
    ports:
      - "5432:5432"
    volumes:
      - db_data_lr4:/var/lib/postgresql/data
      - ./init.sql:/docker-entrypoint-initdb.d/init.sql
    networks:
      - backend_lr4_network
  gateway:
    build:
      context: ../gateway
      dockerfile: ../deploy/Dockerfile
    container_name: gateway
    ports:
      - "3000:3000"
    environment:
      - USERS_SERVICE_URL=http://users-service:3000
      - PROPERTIES_SERVICE_URL=http://property-service:3000
      - CHATS_SERVICE_URL=http://chats-service:3000
    depends_on:
      - db
      - users-service
      - property-service
      - chats-service
    networks:
      - backend_lr4_network
  users-service:
    build:
      context: ../user-service
      dockerfile: ../deploy/Dockerfile
    container_name: users_service
```

```
ports:
  - "3001:3000"
depends_on:
  - db
environment:
  - POSTGRES_HOST=db
  - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
  - POSTGRES_USER=${POSTGRES_USER}
  - POSTGRES_PORT=5432
  - POSTGRES_DB=users
  - JWT_SECRET=${JWT_SECRET}
  - JWT_EXPIRATION=${JWT_EXPIRATION}
networks:
  - backend_lr4_network
property-service:
  build:
    context: ../property-service
    dockerfile: ../deploy/Dockerfile
  container_name: property_service
  ports:
    - "3002:3000"
  depends_on:
    - db
  environment:
    - POSTGRES_HOST=db
    - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
    - POSTGRES_USER=${POSTGRES_USER}
    - POSTGRES_PORT=5432
    - JWT_SECRET=${JWT_SECRET}
    - POSTGRES_DB=properties
  networks:
    - backend_lr4_network
chats-service:
  build:
    context: ../chats-service
    dockerfile: ../deploy/Dockerfile
  container_name: chats_service
  ports:
    - "3003:3000"
  environment:
    - POSTGRES_HOST=db
```

```

- POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
- POSTGRES_USER=${POSTGRES_USER}
- POSTGRES_PORT=5432
- POSTGRES_DB=chats
- SERVICE_KEY=${SERVICE_KEY}
- JWT_SECRET=${JWT_SECRET}
- PROPERTY_SERVICE_URL=http://property-service:3000

depends_on:
  - db

networks:
  - backend_lr4_network

volumes:
  db_data_lr4:
    driver: local

networks:
  backend_lr4_network:
    driver: bridge

```

Для повышенной изоляции микросервисов добавим тривиальный SQL скрипт, который создает базы данных для каждого микросервиса:

```

CREATE DATABASE users;

CREATE DATABASE properties;

CREATE DATABASE chats;

```

Теперь запустим наш compose проект и увидим следующие логи, которые говорят об успешной работе сервисов:

```

postgres      2025-06-01 13:47:40.663 UTC [41] LOG:  database system is shut down
postgres      | done
postgres      server stopped
postgres      PostgreSQL init process complete; ready for start up.
postgres      2025-06-01 13:47:40.723 UTC [1] LOG:  starting PostgreSQL 17.2 on aarch64-unknown-linux-musl, compiled by gcc (Alpine 14.2.0) 14.2.0, 64-bit
postgres      2025-06-01 13:47:40.723 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
postgres      2025-06-01 13:47:40.723 UTC [1] LOG:  listening on IPv6 address ":::", port 5432
postgres      2025-06-01 13:47:40.726 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
postgres      2025-06-01 13:47:40.729 UTC [59] LOG:  database system was shut down at 2025-06-01 13:47:40 UTC
postgres      2025-06-01 13:47:40.732 UTC [1] LOG:  database system is ready to accept connections
chats_service  wait-for-it.sh: db:5432 is available after 3 seconds
users_service  wait-for-it.sh: db:5432 is available after 4 seconds
property_service wait-for-it.sh: db:5432 is available after 4 seconds
gateway        wait-for-it.sh: db:5432 is available after 3 seconds
gateway        Gateway running on port 3000
users_service  [Data Source has been initialized!
users_service  [Server is running on port 3000
chats_service  [Data Source has been initialized!
chats_service  [Server is running on port 3000
property_service [Data Source has been initialized!
property_service [Server is running on port 3000

```

Вывод

В ходе работы нам удалось вновь поработать с Docker. Мы успешно выполнили поставленные задачи, и теперь наши сервисы успешно работают в любом окружении.