САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 1: Миграция написанного API на микросервисную архитектуру

Выполнил:

Оспельников Алексей

Группа К3440

Проверил:
Добряков Д. И.

Санкт-Петербург

2026 г.

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

Ход работы

Монолит был разделен на следующие микросервисы:
1) Сервис аутентификации
2) Сервис сообщений
3) Главный сервис аренды

Для главного сервиса и сообщений были созданы PostgreSQL базы данных

Затем все эти сервисы были созданы в одном docker-compose файле

```
services:
  main:
    build:
      context: ./main
    container_name: main
    ports:
      - "3000:3000"
    depends_on:
      - postgres
    env_file:
      - ./main/.env
  messages:
    build:
      context: ./messages
    container_name: messages
    ports:
      - "5000:5000"
    depends_on:
      - message_db
    env_file:
      - ./main/.env
  rabbitmq:
    image: rabbitmq:3-management
    container_name: rabbitmq
    ports:
      - "5672:5672"
      - "15672:15672"
    environment:
      RABBITMQ_DEFAULT_USER: guest
      RABBITMQ_DEFAULT_PASS: guest

  message_db:
    image: "postgres:17.2"
    container_name: message_db
    restart: always
```

```
    ports:
      - "6543:6543"
    env_file:
      - ./messages/db.env
    volumes:
      - db_data:/var/lib/postgresql/data
  postgres:
    image: "postgres:17.2"
    container_name: postgres
    restart: always
    env_file:
      - ./main/db.env
    volumes:
      - ./main/init.sql:/docker-entrypoint-initdb.d/init.sql
      - db_data:/var/lib/postgresql/data
volumes:
  db_data:
```

Также для каждого микросервиса были созданы Dockerfile

```
FROM node:18-slim
WORKDIR /auth
COPY package*.json ./RUN npm install
COPY . .
EXPOSE 9000
CMD ["npx", "ts-node", "src/index.ts"]
```

Вывод

За время работы монолитная архитектура была разделена на 3 микросервиса и между ними было организовано сетевое взаимодействие, также была организована контейниризация с помощью Docker