

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №4

Выполнила:

Исмагилова Карина
К3344

Проверил:

Добряков Д. И.

Санкт-Петербург

2025 г.

Задача

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

Ход работы

Сперва реализовали Dockerfile для каждого микросервиса. Пример Dockerfile для recipe-service:

```
FROM node:20-alpine

WORKDIR /app

COPY package*.json ./
RUN npm install

COPY . .

RUN chmod +x node_modules/.bin/tsx

EXPOSE 3002

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

Для остальных микросервисов написали аналогичные файлы, с необходимым портом 3000 - 3006

Затем создали общий docker-compose.yml:

```
services:
  users-service:
    build: ./users-service
    ports:
      - "3001:3001"
    env_file:
      - .env
    depends_on:
      - postgres
    networks:
      - app-network
```

```
articles-service:
  build: ./articles-service
  ports:
    - "3004:3004"
  env_file:
    - .env
  networks:
    - app-network

feedback-service:
  build: ./feedback-service
  ports:
    - "3005:3005"
  env_file:
    - .env
  networks:
    - app-network

files-service:
  build: ./files-service
  ports:
    - "3003:3003"
  env_file:
    - .env
  networks:
    - app-network

preference-service:
  build: ./preference-service
  ports:
    - "3006:3006"
  env_file:
    - .env
  networks:
    - app-network

recipes-service:
  build: ./recipes-service
  ports:
    - "3002:3002"
  env_file:
    - .env
  networks:
    - app-network

gateway:
  build: ./gateway
```




























```
ports:
  - "3000:3000"
env_file:
  - .env
depends_on:
  - users-service
  - articles-service
  - feedback-service
  - files-service
  - preference-service
  - recipes-service
networks:
  - app-network

postgres:
  image: postgres:15
  container_name: postgres
  restart: always
  environment:
    POSTGRES_DB: ${DB_NAME}
    POSTGRES_USER: ${DB_USERNAME}
    POSTGRES_HOST_AUTH_METHOD: trust
  ports:
    - "5432:5432"
  volumes:
    - pgdata:/var/lib/postgresql/data
  networks:
    - app-network

networks:
  app-network:
    driver: bridge

volumes:
  pgdata:
```

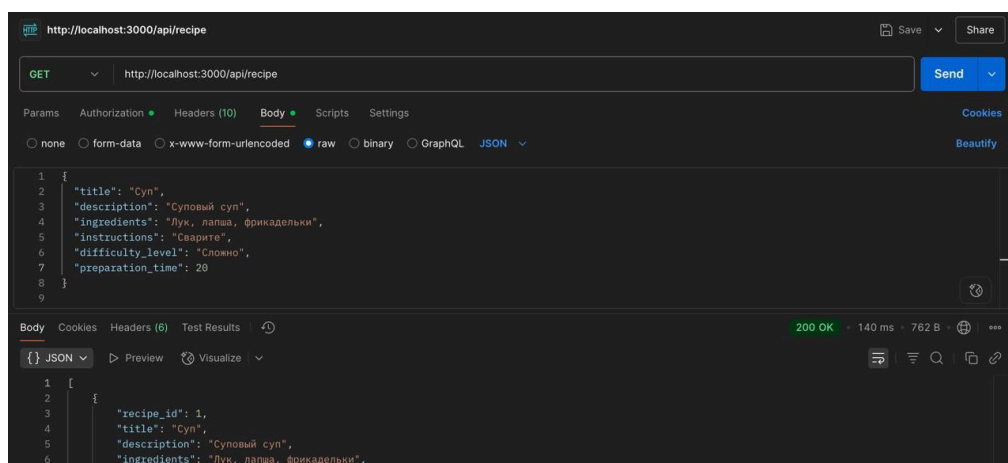
У нас получилось 8 контейнеров, отдельные для базы данных и gateway и 6 для микросервисов:

Name	Container ID	Image	Port(s)	Actions
backend	-	-	-	  
recipes-service-1	8198c8e0d785	backend-re	3002:3002 ↗	  
postgres	2d8bed00b1c1	postgres:1	5432:5432 ↗	  
preference-service	ef6aad69d6c	backend-pr	3006:3006 ↗	  
feedback-service-1	17e1f4d42d20	backend-fe	3005:3005 ↗	  
articles-service-1	3f7bf944fc61	backend-ar	3004:3004 ↗	  
users-service-1	2f6df1fc6b76	backend-us	3001:3001 ↗	  
files-service-1	9f12b68dbb58	backend-fil	3003:3003 ↗	  
gateway-1	1aa442dafa67	backend-ga	3000:3000 ↗	  

Также поменяли config.ts в gateway, мы указали адреса, чтобы gateway знал, куда перенаправлять запросы к каждому микросервису. Это обеспечивает изолированное взаимодействие между сервисами внутри Docker-сети и позволяет обращаться к ним по именам контейнеров, а не по localhost:

```
export const services = {
  user: 'http://users-service:3001',
  recipe: 'http://recipes-service:3002',
  file: 'http://files-service:3003',
  article: 'http://articles-service:3004',
  feedback: 'http://feedback-service:3005',
  preference: 'http://preference-service:3006',
};
```

Затем проверили работоспособность, можно увидеть что запись создается и получается:



Внутри контейнера выполнили `curl http://recipes-service:3002/`, чтобы проверить, что сервис работает и отвечает на запросы, а также убедиться, что маршрут доступен и данные успешно получаются из базы.

```
/app # curl http://recipes-service:3002/
[{"recipe_id":1,"title":"Суп","description":"Суповый суп","ingredients":"Лук, лапша, фрикадельки","instructions":"Сварите","difficulty_level":"Сложно","preparation_time":20,"created_at":"2025-06-01T10:41:23.270Z","updated_at":"2025-06-01T10:41:23.270Z","user":{"user_id":1,"username":"Annet","email":"Anna@example.com","password":"$2a$08$66C.nrpPoBHbleh4Ms4kfe8Z0RFPicoiaBTSV46wvqdfMcLxk2mXi","bio":"I LOVE Cooooking"},"files":[],"categories":[],"comments":[]}]
/app #
```

Вывод: Все микросервисы были упакованы в отдельные Docker-контейнеры и объединены с помощью `docker-compose`. Получилось также реализовать корректное взаимодействие сервисов по внутренним именам в Docker-сети. Gateway перенаправляет запросы на нужные сервисы в зависимости от маршрута.