## САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №4

Выполнил: Кривцов Павел Группа K33402

Проверил: Добряков Д. И.

## Задача

Необходимо упаковать приложение в docker-контейнеры и обеспечить сетевое взаимодействие между различными его частями.

## Ход работы

Логику работы микросервисов оставим без изменений и напишем к каждому по докерфайлу. Выглядеть он будет следующим образом и отличаться только портами для каждого микросервиса:

```
FROM node:19.8.1-alpine

WORKDIR /gateway

COPY package*.json ./

RUN npm install

COPY . .

EXPOSE 8000

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

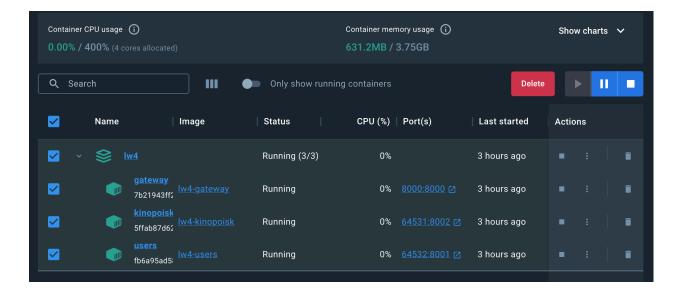
Файл docker-compose будет выглядеть так (видно, что есть сущность gateway, зависимая от users и kinopoisk):

```
version: '3.9'
4 🗲 🖯 gateway:
         container_name: gateway
     build:
         context: ./gateway
     depends_on:
          - users
       - kinopoisk
       ports:
        - "127.0.0.1:8000:8000"
       restart: always
14 d 🖯 users:
        container_name: users
       build:
         context: ./users
       ports:
         - "8001"
19
        restart: always
21 <mark>द</mark> 🖯 kinopoisk:
         container_name: kinopoisk
         build:
          context: ./kinopoisk
        ports:
         - "8002"
        restart: always
```

Немного модернизируем переадресацию в index.ts файле гейтвэя, чтобы вместо localhost указывалось название докер-контейнера, к которому направлен запрос.

```
18
     for (const m in microservices) {
           app.all( path: ^/${m}/*^, handlers: async (req: any, res: any) => {
               const url = `http://${m}:${microservices[m]}${req.originalUrl}`;
               try {
                   const response = await axios({
                       method: req.method,
                       url: url,
                        data: req.body,
                           auth: req.headers.auth
                   });
                   res.status(response.status).send(response.data);
               } catch (e) {
                   if (e.response) {
                        res.status(e.response.status).send(e.response.data);
                   } else {
                        res.status(500).send('Internal Server Error');
               }
           });
```

В результате запускается 3 докер-контейнера, которые общаются с внешним миром через gateway на 8000-м порту.



## Вывод

В ходе работы удалось создать docker-compose и наладить сетевое взаимодействие между разными docker-контейнерами (микросервисами) приложения. Документация к приложению:

https://documenter.getpostman.com/view/29513351/2s9YBz3b3P