САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа № 3
«Микросервисы»

Выполнила:
Коник А. А.
Группа К33402

Проверил:
Добряков Д. И.

Санкт-Петербург

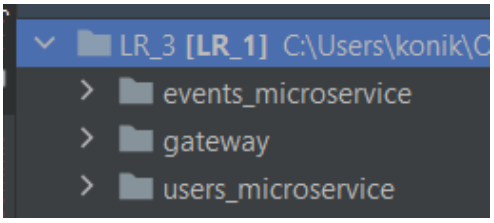
2023 г.

Задача

Необходимо реализовать отдельный микросервис, выполняющий какую-либо содержательную функцию из всего арсенала функций вашего приложения.

Ход работы

В своем проекте я выделила два микросервиса: один – для регистрации и авторизации пользователя, второй – для мероприятий и записи на них.



Клиент отправляет запрос на порт 8000, на котором запущен gateway, который в свою очередь распределяет запросы либо к микросервису пользователей (порт 9001), либо к микросервису мероприятий (порт 9000), в зависимости от искомого юрла.

```
const express = require('express');
const axios = require('axios');

const app = express();
const port = 8000;

app.use(express.json());


new *
app
    .all( path: '/events/*', handlers: async (req: any, res: any) => {
        const url = `http://localhost:9000${req.originalUrl}`;

        try {
            const response = await axios({
                method: req.method,
                url: url,
                data: req.body,
            });
            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');
            }
        }
    });
```

```
app
    .all( path: '/users/*', handlers: async (req: any, res: any) => {
        const url = `http://localhost:9001${req.url}`;

        try {
            const response = await axios({
                method: req.method,
                url: url,
                data: req.body,
            });
            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');
            }
        }
    });

new *
app.listen(port, callback: () => {
    console.log(`Running gateway on port ${port}`);
});
```

Микросервисы пользователей и мероприятий изолированы и запускаются отдельно, не зависимо друг от друга, со своими базами данных.

В контроллер записей на мероприятия (создание, удаление, получение записи) была добавлена логика проверки авторизованности пользователя, так как passport.authenticate находится в другом микросервисе.

```
post = async (request: Request, response: Response) => {
    const {body} = request // only target event; id comes from token via auth service

    try {
        const auth_resp = await axios({
            url: "localhost:8000/users/profile",
            headers: request.headers
        })
        if (auth_resp.status === 200) {
            body.userId = auth_resp.data.id
            const enroll = await this.enrollService.create(body)
            response.status( code: 201).send(enroll)
        } else {
            response.status(auth_resp.status).send()
        }
    } catch (error: any) {
        response.status( code: 400).send( body: {error: error.message})
    }
}
```

Вывод

В ходе работы я разобралась в отличиях монолитной и микросервисной архитектур, узнала о преимуществах микросервисной архитектуры, но и поняла, почему ее редко используют для маленьких приложений. Также я разделила свой проект на два микросервиса с использованием gateway.