САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 3

Выполнила:

Афанасьева Ирина Максимовна

Группа:

К33402

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Задание

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

Ход работы

Решено выделить механизм аутентификации и модель пользователя в отдельный микросервис. Для проверки того, что пользователь авторизован, добавлен новый эндпоинт:

```
validateToken = async (request: any, response: any) => {
    const {body} = request
    const {accessToken} = body
    try {
        const payload = jwt.verify(accessToken, jwtConfig.secret)
        // @ts-ignore
        const user = await this.userService.getById(payload.id)
        response.send({'valid': true, 'user': user})
    } catch (e: any) {
        response.status(401).send({'valid': false})
    }
}
```

Для пользователя сохранена структура API из ЛР №2. Перенесенные эндпоинты проксируются с помощью библиотеки Axios:

```
class UserController {
    private async requestProxy(url: string, method: any, expressResponse: any, body: any = null) {
        axios({
            method: method,
            url: `http://${authConfig.host}:${authConfig.port}${url}`,
            data: body
        }).then((resp) =>{
            expressResponse.status(resp.status).send(resp.data)
        }).catch((error) => {
            expressResponse.status(error.response.status).send(error.response.data)
        })
    }
```

Middleware для авторизации переписан для создания запроса к микросервису авторизации:

```
const customJwtStrategy = new Strategy(async function (token: any, done: any) {
    axios.post(
        `http://${authConfig.host}:${authConfig.port}/api/auth/validate`,
        {'accessToken': token}
    ).then((resp) => {
        if (resp.status == 200 && resp.data.valid) {
            const user = resp.data.user
            done(null, user)
        } else {
            done(null, false)
        }
    }).catch((error) => {
        done(error)
    })
})
```

Вывод

Я разделила монолитное веб-приложение на node.js на два микросервиса, взаимодействующих при помощи API.