

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Backend-Разработка

Отчет

Лабораторная работа 3

Выполнил:
Бункута Натан

Группа:
К33412

Проверил:
Добрияков Д.И

Санкт-Петербург

2023 г.

Задание:

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

Выполнение работы

Микросервис, который я решил реализовать, касается процесс аутентификации пользователя, чтобы отделить его от модели.

Была добавлена новая конечная точка (Endpoint) , чтобы проверить, авторизован ли пользователь

```
92     validateToken = async (request: any, response: any) => {
93         const {body} = request
94         const {accessToken} = body
95         try {
96             const payload = jwt.verify(accessToken, jwtConfig.secret)
97
98             const user = await this.userService.getById(payload.id)
99             response.send({'valid': true, 'user': user})
100         } catch (e: any) {
101             response.status(401).send({'valid': false})
102         }
103     }
104 }
105
106 export default UserController
```

Для пользователя, структура API из лаб-2 не менялась.

```
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)
        })
    }

    post = async (request: any, response: any) => {
        const {body} = request
        await this.requestProxy('/api/users', 'post', response, body)
    }
}
```

Так выглядеть Middleware для создание запрос 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)
  })
})
```

Применение Endpoints

The screenshot shows a REST client interface with a POST request to `http://localhost:5000/api/auth/login`. The request body is a JSON object with `email` and `password` fields. The response is a 200 OK status with a JSON body containing `accessToken` and `refreshToken`.

Request:

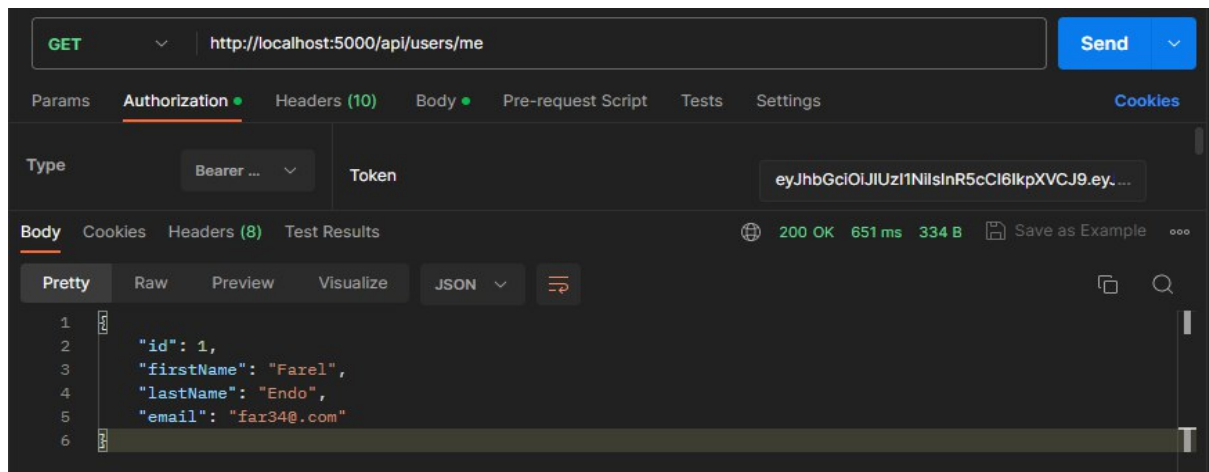
```
POST http://localhost:5000/api/auth/login
```

Body:

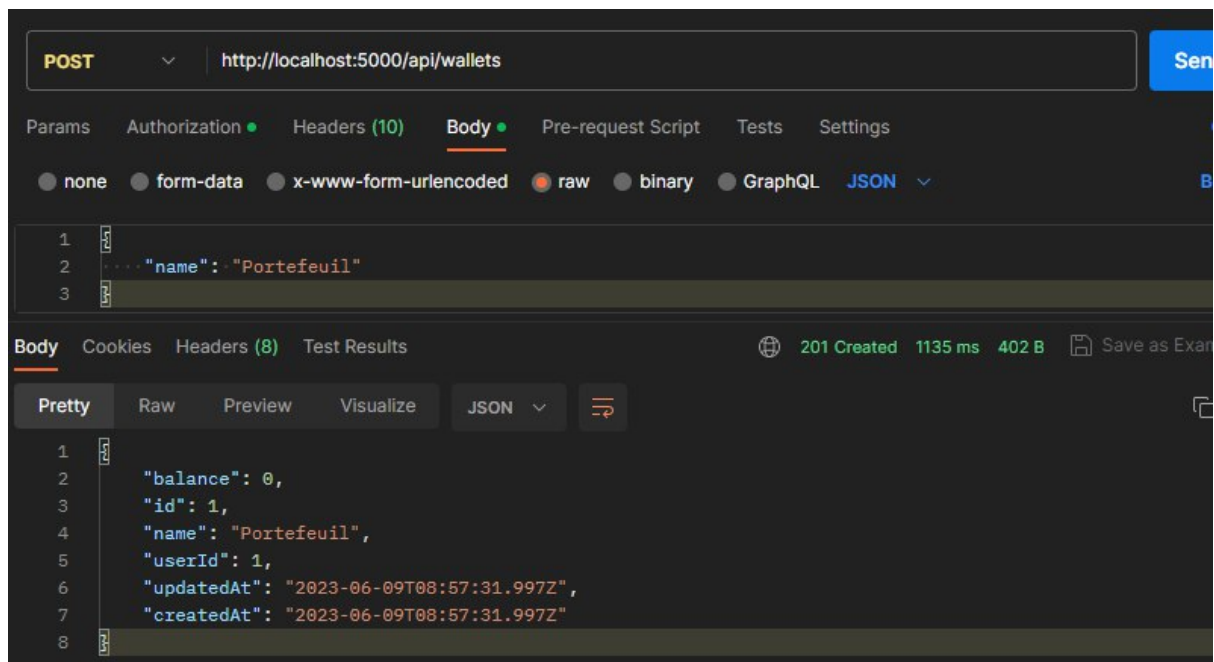
```
{
  "email": "far34@.com",
  "password": "Mosai23"
}
```

Response:

```
{
  "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im5wZWl0IjoxNjg2Mjk5Njk3fQ.eyJpZCI6Im5wZWl0IjoxNjg2Mjk5Njk3fQ.",
  "refreshToken": "0f22ef94-f201-4c10-a8f2-ba05c2005c3b"
}
```



Основной сервис



Вывод:

Получилось разделить основной сервис приложения на два микросервиса, которые взаимодействуют с помощью API.

