

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бек-энд разработка

Отчет

Лабораторная работа 3: DI, IoC, Развёртывание,
микросервисы, CI/CD

Выполнил:

Скороходова Елена

Группа
K33392

Проверил:

Добряков Д. И.

Санкт-Петербург

2024 г.

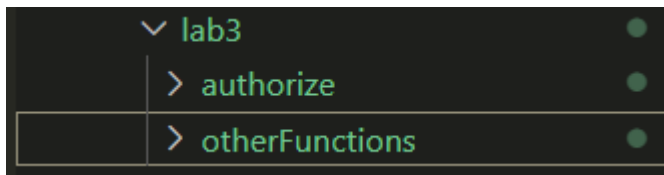
Задачи

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

Ход работы

Было решено выделить авторизацию, регистрацию и заполнение профиля в микросервис.

Для начала функционал был разделен на две папки: `authorize` с логикой авторизации, регистрации и заполнения профиля; `otherFunctions` с остальными функциями приложения.



В `AuthController` добавляется функция проверки переданного токена авторизации.

```
TS AuthController.ts U X TS UserBookService.ts M TS AuthController.ts M X
labs > K33392 > Skorokhodova_Elena > lab3 > authorize > src > controllers > TS AuthController.ts > 4
1 import { Request, Response } from "express";
2 import { AuthService } from "../services/AuthService";
3 import { UserService } from "../services/UserService";
4 import jwt, { JwtPayload } from "jsonwebtoken";
5
6 class AuthController {
7   static async register(req: Request, res: Response): Promise<void> {
8     try {
9       const { name, email, password } = req.body;
10      const token = await AuthService.registerUser(name, email, password);
11      res.status(201).json({ token });
12    } catch (error) {
13      if (error instanceof Error) {
14        res.status(400).json({ error: error.message });
15      } else {
16        res.status(400).json({ error: "An unknown error occurred" });
17      }
18    }
19  }
20
21   static async login(req: Request, res: Response): Promise<void> {
22     try {
23       const { email, password } = req.body;
24       const token = await AuthService.loginUser(email, password);
25       res.status(200).json({ token });
26     } catch (error) {
27       if (error instanceof Error) {
28         res.status(401).json({ error: error.message });
29       } else {
30         res.status(401).json({ error: "An unknown error occurred" });
31       }
32     }
33   }
34
35   static async verify(req: Request, res: Response): Promise<void> {
36     try {
37       const token = req.body.token;
38       if (!token) {
39         res.status(401).json({ error: "Token is missing" });
40       }
41       const decoded = jwt.verify(token, process.env.JWT_SECRET);
42       res.status(200).json({ decoded });
43     } catch (error) {
44       if (error instanceof Error) {
45         res.status(401).json({ error: error.message });
46       } else {
47         res.status(401).json({ error: "An unknown error occurred" });
48       }
49     }
50   }
51 }
52
53 export { AuthController };
end-2024 > labs > K33392 > Skorokhodova_Elena > lab2 > src > controllers > TS AuthController.ts > 4
1 import { Request, Response } from "express";
2 import { AuthService } from "../services/AuthService";
3
4 class AuthController {
5   static async register(req: Request, res: Response): Promise<void> {
6     try {
7       const { name, email, password } = req.body;
8       const token = await AuthService.registerUser(name, email, password);
9       res.status(201).json({ token });
10    } catch (error) {
11      if (error instanceof Error) {
12        res.status(400).json({ error: error.message });
13      } else {
14        res.status(400).json({ error: "An unknown error occurred" });
15      }
16    }
17  }
18
19   static async login(req: Request, res: Response): Promise<void> {
20     try {
21       const { email, password } = req.body;
22       const token = await AuthService.loginUser(email, password);
23       res.status(200).json({ token });
24     } catch (error) {
25       if (error instanceof Error) {
26         res.status(401).json({ error: error.message });
27       } else {
28         res.status(401).json({ error: "An unknown error occurred" });
29       }
30     }
31   }
32 }
33
34 export { AuthController };
35
```

Так как проверка токена теперь выполняется отдельно, то используем функцию `fetch` для отправки запроса на сервис аутентификации.

После отправки запроса на сервис аутентификации происходит проверка `resp.ok`, чтобы определить, успешно ли прошла проверка токена. Если ответ не успешен, возвращается статус 401.

Условием для проверки токена является `jwt.verify()`, который проверяет целостность и валидность токена.

```

1 import { Request, Response } from "express";
2 import jwt from "jsonwebtoken";
3 import process from "process";
4
5 export const authUsersMiddleware = async (
6   req: Request,
7   res: Response,
8   next: any
9 ) => {
10   try {
11     if (
12       (req.path === "/users" && req.method === "POST") ||
13       req.path === "/login" ||
14       req.path === "/register"
15     )
16       return next();
17     if (!req.headers.authorization)
18       return res.status(403).json({ message: "Unauthorized" });
19     const token = req.headers.authorization.split(" ")[1];
20     if (!token) return res.status(403).json({ message: "Unauthorized" });
21     const resp = await fetch(`${process.env.AUTH_SERVICE}/token/verify`, {
22       method: "POST",
23       body: JSON.stringify({ token: token }),
24       headers: { "Content-Type": "application/json" },
25     });
26     if (!resp.ok) {
27       res.sendStatus(401);
28       return;
29     }
30     next();
31   } catch (e) {
32     res.sendStatus(401);
33   }
34 };
35

```

Пример запроса регистрации:

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** http://localhost:8001/register
- Body (JSON):**

```

1 {
2   "name": "Helen",
3   "email": "he1@gmail.com",
4   "password": "12345"
5 }
6

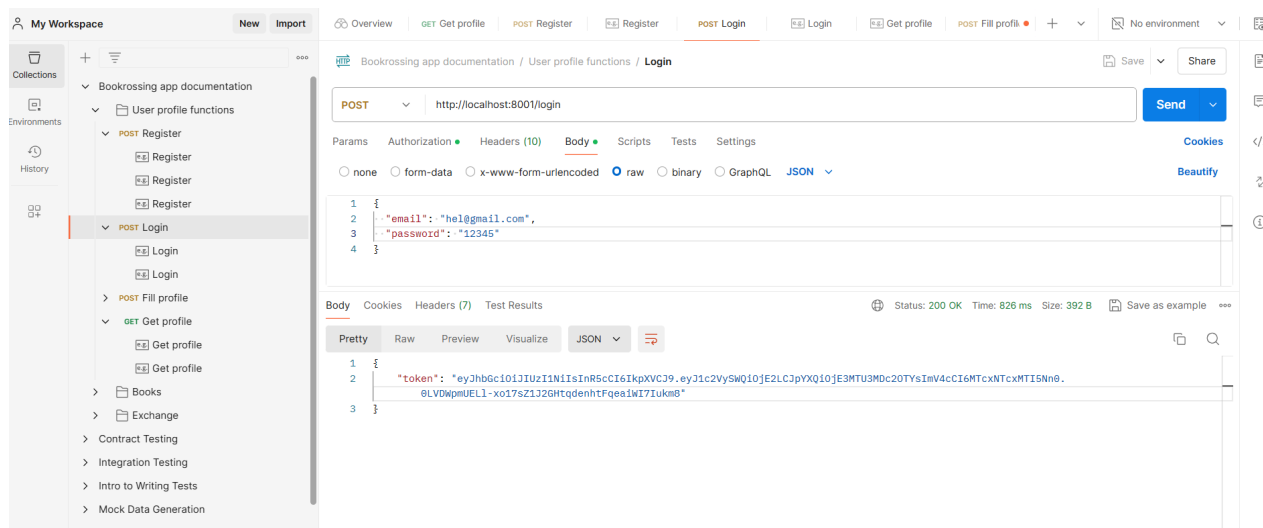
```
- Status:** 201 Created
- Time:** 1146 ms
- Size:** 397 B
- Body (JSON):**

```

1 {
2   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQ10jE2L0pYXQ10jE3MTU3M0c2NjMsInV4cCI6MTcxNTcxMTI2M3B9.dpVv07WnLYjYzT9MH8UQ2F410LEcFX_cbbxwAWBy0"
3 }

```

Пример запроса авторизации:



Вывод

В третьей лабораторной работе удалось выделить логику авторизации, регистрации, профиля в отдельный микросервис.