САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Практическая/Лабораторная работа

Выполнил:

Пластун Елизавета Олеговна

Группа K33402

Проверил:
Добряков Д. И.

Санкт-Петербург

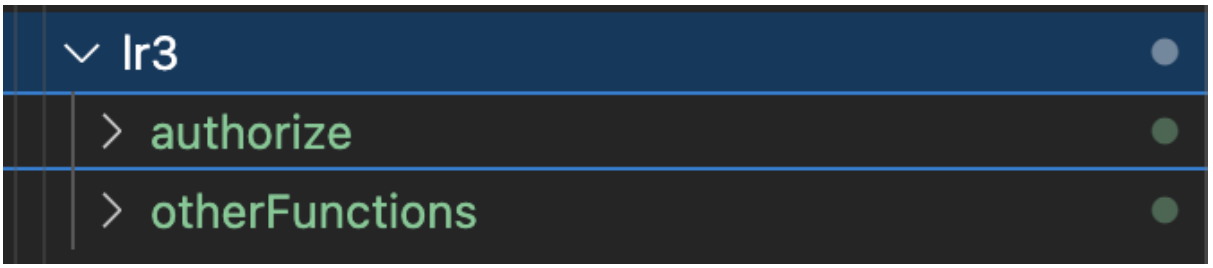
2022 г.

Задача

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.

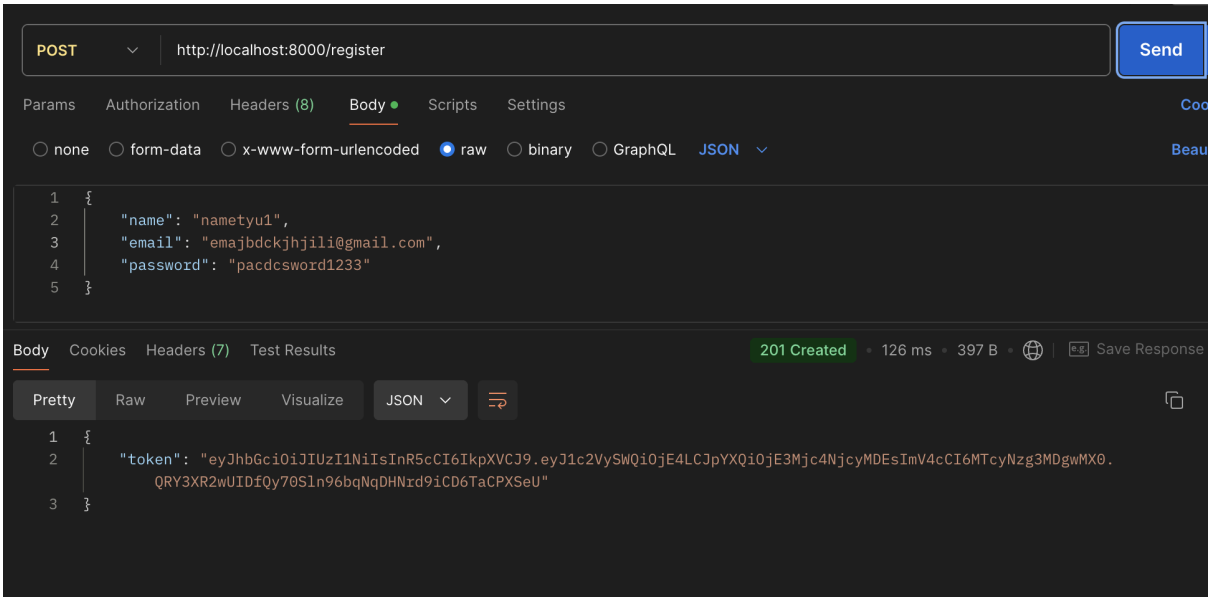
```
class AuthController {
  static async register(req: Request, res: Response): Promise<void> {
    try {
      const { name, email, password } = req.body;
      const token = await AuthService.registerUser(name, email, password);
      res.status(201).json({ token });
    } catch (error) {
      if (error instanceof Error) {
        res.status(400).json({ error: error.message });
      } else {
        res.status(400).json({ error: "An unknown error occurred" });
      }
    }
  }

  static async login(req: Request, res: Response): Promise<void> {
    try {
      const { email, password } = req.body;
      const token = await AuthService.loginUser(email, password);
      res.status(200).json({ token });
    } catch (error) {
      if (error instanceof Error) {
        res.status(401).json({ error: error.message });
      } else {
        res.status(401).json({ error: "An unknown error occurred" });
      }
    }
  }

  static async verify(req: Request, res: Response): Promise<void> {
    try {
      const token = req.body.token;
      if (!token) {
        res.status(400).json({ error: "token was not provided" });
        return;
      }
      const payload = jwt.verify(token, process.env.JWT_SECRET_KEY as string);
      const userId = (payload as JwtPayload).userId;
```

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.

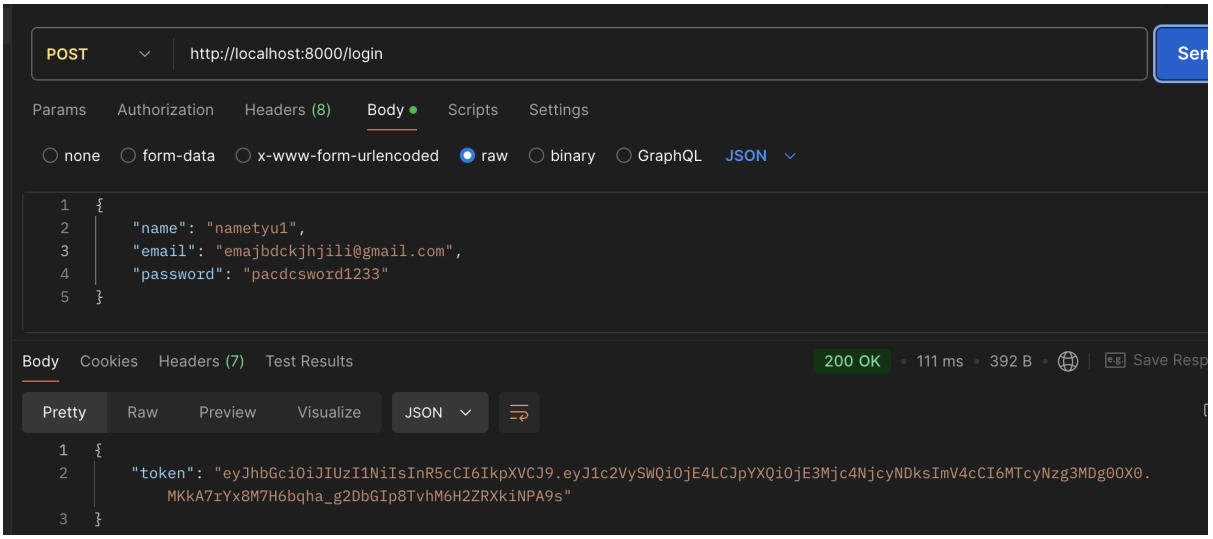
labs > K33402 > Plastun Elizaveta > lr3 > otherFunctions > src > middlewares > TS auth.ts > ...

```ts
import { Request, Response } from "express";
import jwt from "jsonwebtoken";
import process from "process";

export const authUsersMiddlware = async (
  req: Request,
  res: Response,
  next: any
) => {
  try {
    if (
      (req.path === "/users" && req.method === "POST") ||
      req.path === "/login" ||
      req.path === "/register"
    )
      return next();
    if (!req.headers.authorization)
      return res.status(403).json({ message: "Unauthorized" });
    const token = req.headers.authorization.split(" ")[1];
    if (!token) return res.status(403).json({ message: "Unauthorized" });
    const resp = await fetch(`${process.env.AUTH_SERVICE}/token/verify`, {
      method: "POST",
      body: JSON.stringify({ token: token }),
      headers: { "Content-Type": "application/json" },
    });
    if (!resp.ok) {
      res.sendStatus(401);
      return;
    }
    next();
  } catch (e) {
    res.sendStatus(401);
  }
};

```

Пример запроса регистрации:



Пример запроса авторизации:



Вывод

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