САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка
Отчет
Лабораторная работа №3

Выполнил:
Коровин Александр

Группа:
K33402

Проверил:
Добряков Д. И.

Санкт-Петербург
2022 г.

Задача: реализовать отдельный микросервис, выполняющий какую-либо содержательную функцию из всего арсенала функции вашего приложения.

Ход работы

Входе работы было решено разделить приложение на два сервиса: основной сервис и сервис авторизации. В сервис авторизации включил регистрацию, логин, CRUD методы по работе с пользователем.

1)

```ts
import axios from "axios"
import { config } from "../config/config";

class UserController {
    private async requestProxy(url: string, method: any, expressResponse: any, body: any = null) {
        axios({
            method: method,
            url: `http://${config.auth_service.host}:${config.auth_service.port}${url}`,
            data: body
        }).then((res) =>{
            expressResponse.status(res.status).send(res.data)
        }).catch((error) => {
            expressResponse.status(error.response ? error.response.status : 500).send(error.response ? error.response.data : {"error": error.message})
        })
    }

    register = async (request: any, response: any) => {
        const {body} = request

        await this.requestProxy('/users/register', 'post', response, body)
    }

    get = async (request: any, response: any) => {
        const {user} = request
        if (user) {
            await this.requestProxy(`/users/get/${user.id}`, 'get', response)
        } else {
            response.status(401).send({'detail': 'Not authenticated'})
        }
    }
    getList = async (request: any, response: any) => {
        const {users} = request
        if (users) {
            await this.requestProxy(`/users/get/`, 'get', response)
        } else {
            response.status(401).send({'detail': 'Not authenticated'})
        }
    }

    update = async (request: any, response: any) => {
        const {body, user} = request
        if (user) {
            await this.requestProxy(`/users/update/${user.id}`, 'patch', response, body)
        } else {
            response.status(401).send({'detail': 'Not authenticated'})
        }
    }

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

    delete = async (request: any, response: any) => {
        const {body, user} = request
        await this.requestProxy(`/users/delete/${user.id}`, 'delete', response, body)
    }
}

export default UserController
```

Реализовал контроллер по взаимодействию с сервисом авторизации

2)

```
const loginUser = (req: Request, res: Response, next: NextFunction) => {
  const { email, password } = req.body;
  return User.findOne({ email }).then((user) => {
    if (user) {
      const validPassword = checkPassword(user, password);
      if (validPassword) {
        const token = generateAccessToken(email);
        return res.status(201).json({ token });
      } else {
        return res.status(400).json({ message: "Wrong password!" });
      }
    } else {
      return res.status(400).json({ message: "User is not found!" });
    }
  });
};
```

Пример функции логина из UserController сервиса авторизации. В него входит: проверка пароля, генерация токена.

3)

```
import bcrypt from "bcrypt";

export default (user: any, password: string) => {
  return bcrypt.compareSync(password, user.password);
};

```

Проверка пароля

```
import { config } from "../config/config";
import { sign } from "jsonwebtoken";

export default (email: string) => {
  return sign(email, config.jwt.key);
};

```

Генерация токена

```
import bcrypt from "bcrypt";

export default (password: string): string =>
  bcrypt.hashSync(password, bcrypt.genSaltSync(8));

```

Хэширование пароля

Вывод

Я разделил монолитное веб-приложение на два сервиса: основной сервис по работе с отелями и сервис авторизации.