САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

по лабораторной работе №3

Выполнил:

Кузгиев Адам

Группа
К33402

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

В ходе работы был создан микросервис для аутентификации юзеров.

Контроллер микросервиса аутентификации:

```
import axios from "axios"

export default class AuthController {

    post = async (request: any, response: any) => {
        const { email, password } = request.body
        const secretOrKey = 'test101'
        if (email && password) {
            let user = await axios.get("http://localhost:8080/v1/user", {
                params: {
                    email: email
                }
            })
            if (!user) {
                response.status(401).json({ msg: 'No such user found', user })
            }
            if (user!!.data.password === password) {
                let payload = { id: user!.data.id }
                const jwt = require('jsonwebtoken')
                let token = jwt.sign(payload, secretOrKey)
                response.json({ msg: 'ok', token: token })
            } else {
                response.status(401).json({ msg: 'Password is incorrect' })
            }
        }
    }
}
```

Роутер аутентификации

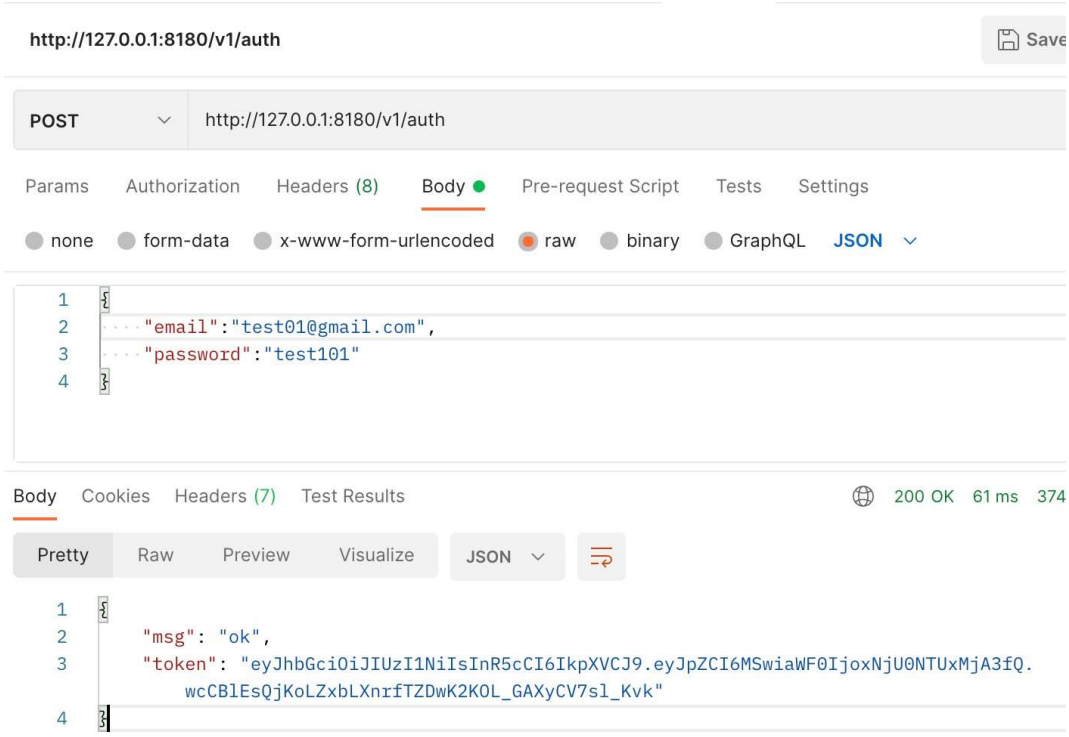
```
import express from "express"
import AuthController from "../controllers/auth"

const router: express.Router = express.Router()

const authController = new AuthController()

router
  .route('/auth')
  .post(authController.post)

export default router
```

Основной код приложения для работы с бронированием отелей был вынесен в микросервис “Base”.

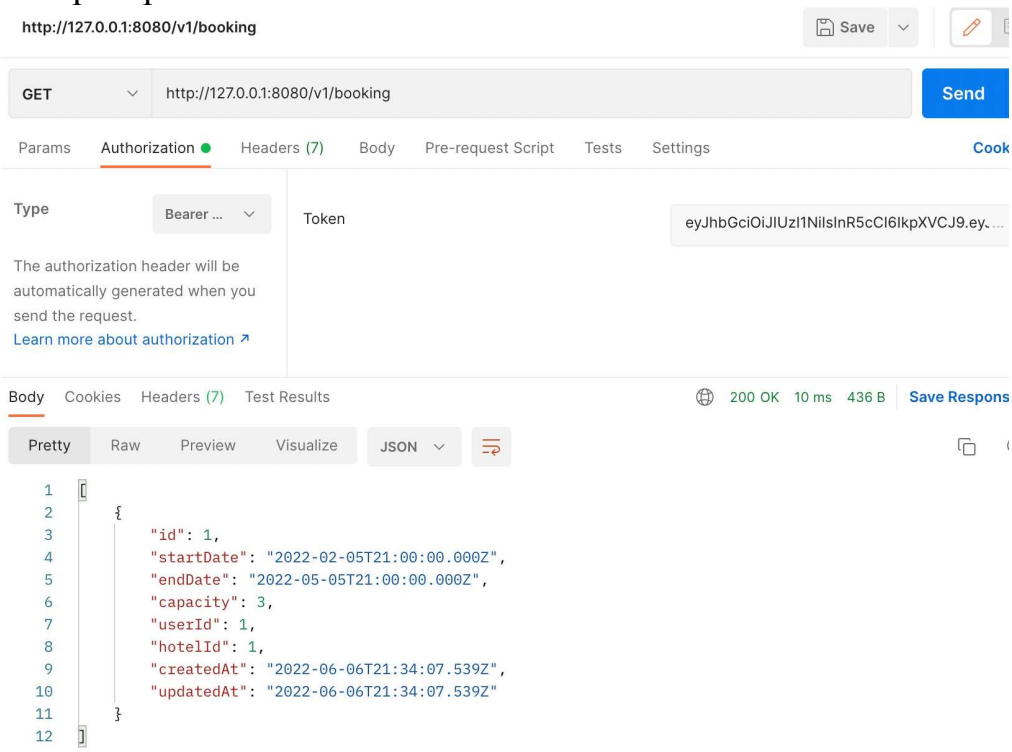
```ts
import express from "express"
import UserController from '../controllers/user/index'
import HotelController from '../controllers/hotel/index'
import BookingController from '../controllers/booking/index'

const router: express.Router = express.Router()
const passport = require('passport')

const userController = new UserController()
const hotelController = new HotelController()
const bookingController = new BookingController()

router
  .route('/user')
  .get(userController.get)
  .post(userController.post)

router
  .route('/hotel')
  .get(hotelController.get)
  .post(hotelController.post)

router
  .route('/booking')
  .get(passport.authenticate('jwt', { session: false }), bookingController.get)
  .post(passport.authenticate('jwt', { session: false }), bookingController.post)

export default router
```

Примеры работы:

Микросервис “auth”:



Микросервис “base”:



Вывод

В ходе работы было создано два микросервиса, работающие друг с другом посредством http запросов. Первый микросервис осуществляет аутентификацию пользователей, а второй – основную работу с приложением – бронирование отелей и прочие взаимодействия с ними.