

**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:

```
1  import axios from "axios"
2
3  export default class AuthController {
4
5      post = async (request: any, response: any) => {
6          const { email, password } = request.body
7          const secretOrKey = 'test101'
8          if (email && password) {
9              let user = await axios.get("http://localhost:8080/v1/user", {
10                  params: {
11                      email: email
12                  }
13              })
14              if (!user) {
15                  response.status(401).json({ msg: 'No such user found', user })
16              }
17              if (user!.data.password === password) {
18                  let payload = { id: user!.data.id }
19                  const jwt = require('jsonwebtoken')
20                  let token = jwt.sign(payload, secretOrKey)
21                  response.json({ msg: 'ok', token: token })
22              } else {
23                  response.status(401).json({ msg: 'Password is incorrect' })
24              }
25          }
26      }
27  }
```

Роутер аутентификации

```
1  import express from "express"
2  import AuthController from "../controllers/auth"
3
4  const router: express.Router = express.Router()
5
6  const authController = new AuthController()
7
8  router
9      .route('/auth')
10     .post(authController.post)
11
12  export default router
```

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

```
1 import express from "express"
2 import UserController from '../controllers/user/index'
3 import HotelController from '../controllers/hotel/index'
4 import BookingController from '../controllers/booking/index'
5
6 const router: express.Router = express.Router()
7 const passport = require('passport')
8
9 const userController = new UserController()
10 const hotelController = new HotelController()
11 const bookingController = new BookingController()
12
13 router
14   .route('/user')
15   .get(userController.get)
16   .post(userController.post)
17
18 router
19   .route('/hotel')
20   .get(hotelController.get)
21   .post(hotelController.post)
22
23 router
24   .route('/booking')
25   .get(passport.authenticate('jwt', { session: false }), bookingController.get)
26   .post(passport.authenticate('jwt', { session: false }), bookingController.post)
27
28 export default router
```

Примеры работы:

Микросервис “auth”:

The screenshot displays a REST client interface for a POST request to `http://127.0.0.1:8180/v1/auth`. The request body is a JSON object with email and password fields. The response shows a 200 OK status with a message and a JWT token.

Request URL: `http://127.0.0.1:8180/v1/auth`

Method: **POST**

Body (JSON):

```
{
  "email": "test01@gmail.com",
  "password": "test101"
}
```

Response (JSON):

```
{
  "msg": "ok",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNjU0NTUxMjA3fQ.wcB1EsQjKoLZxbLXnrfTZDwK2KOL_GAXyCV7sl_Kvk"
}
```

Status: 200 OK, 61 ms, 374 bytes

## Микросервис “base”:

http://127.0.0.1:8080/v1/booking

GET http://127.0.0.1:8080/v1/booking

Authorization

Type: Bearer ... Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...

The authorization header will be automatically generated when you send the request.  
[Learn more about authorization](#)

Body

200 OK 10 ms 436 B

```
1 {
2   {
3     "id": 1,
4     "startDate": "2022-02-05T21:00:00.000Z",
5     "endDate": "2022-05-05T21:00:00.000Z",
6     "capacity": 3,
7     "userId": 1,
8     "hotelId": 1,
9     "createdAt": "2022-06-06T21:34:07.539Z",
10    "updatedAt": "2022-06-06T21:34:07.539Z"
11  }
12 }
```

## 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.