

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Бэкенд-энд разработка

Отчет

Лабораторная работа №3

Выполнил:  
Золотов Павел

Группа:  
К33401

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 выполнения работы был реализован микросервис для авторизации пользователей

Код контроллера авторизации после добавления микросервиса:

```
import axios from "axios"

export default class AuthController {

  post = async (request: any, response: any) => {
    const { email, password } = request.body
    const secretOrKey = 'test123'
    if (email && password) {
      let user = await axios.get("http://localhost:8000/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' })
    }
  }
}
```

Роуты микросервиса авторизации:

```
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
```

Остальной код для работы с отелями и бронированием номеров был вынесен в микросервис main

Роуты микросервиса main:

```
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
```

Демонстрация работы микросервиса авторизации:

POST localhost:8100/v1/auth

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 { "email": "test@test.com", "password": "123" }
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 34 ms

Pretty Raw Preview Visualize JSON

```
1 {
2   "msg": "ok",
3   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNjUzOTkyMDc1fQ.qRiIQ66pIa2ZG7Aq1ThEeDMLqkU1obG9R11JE8NKNSI"
4 }
```

## Демонстрация работы микросервиса main:

The screenshot displays a REST client interface for a GET request to `localhost:8000/v1/booking`. The 'Headers' tab is active, showing an 'Authorization' header with a Bearer token. The 'Body' tab shows a JSON response with details about a booking.

**Headers**

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJp...	
Key	Value	Description

**Body** Status: 200 OK Time: 10 ms

JSON

```
1 {
2   {
3     "id": 1,
4     "startDate": "2022-01-04T21:00:00.000Z",
5     "finishDate": "2022-10-04T21:00:00.000Z",
6     "capacity": 2,
7     "userId": 1,
8     "hotelId": 1,
9     "createdAt": "2022-05-31T10:05:48.380Z",
10    "updatedAt": "2022-05-31T10:05:48.380Z"
11  }
12 }
```

## Вывод

По итогам работы было реализовано 2 микросервиса, взаимодействующих между собой с помощью http запросов. Первый микросервис отвечает за авторизацию пользователей, второй за работу с отелями и бронированиями.