САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэкенд-энд разработка

Отчет

Лабораторная работа №2

Выполнил:
Золотов Павел

Группа:
К33401

Проверил:
Добряков Д. И.

Санкт-Петербург

2022 г.

Задача

Для выполнения работы был выбран вариант «платформа для поиска и бронирования номера в отеле/квартире/хостеле». По выбранному варианту необходимо будет реализовать RESTful API средствами express + typescript (используя ранее написанный boilerplate).

Ход работы

Файл /core/index.ts. В этом файле происходит инициализация Express, подключение к базе данных:

```
import express from "express"
import { createServer, Server } from "http"
import routes from "../routes/index"
import { sequelize } from '../config/config'
import UserService from '../services/user/index'

export const jwtOptions: any = {}

export default class App {
  public port: number
  public host: string

  private app: express.Application
  private server: Server

  constructor(port = 8000, host = "localhost") {
      this.port = port
      this.host = host

      this.app = this.createApp()
      this.server = this.createServer()
  }

  private createApp(): express.Application {
      const app = express()
      const bodyParser = require('body-parser')
      const passport = require('passport')
      const passportJwt = require('passport-jwt')
      let ExtractJwt = passportJwt.ExtractJwt
      let JwtStrategy = passportJwt.Strategy
      jwtOptions.jwtFromRequest = ExtractJwt.fromAuthHeaderAsBearerToken()
      jwtOptions.secretOrKey = 'test123'
      let strategy = new JwtStrategy(jwtOptions, async function(jwt_payload: any,
next: any) {
        console.log('payload received', jwt_payload)
        const service = new UserService()
        let user = await service.getById(jwt_payload.id)
        if (user) {
```

```
          next(null, user)
        } else {
          next(null, false)
        }
      });
      passport.use(strategy)
      app.use(bodyParser.urlencoded({ extended: false }))
      app.use(bodyParser.json())
      app.use(passport.initialize())
      app.use('/v1', routes)
      return app
    }

  private createServer(): Server {
      return createServer(this.app)
  }

  public start(): void {
      sequelize.sync().then(() => {
        console.log('DB connected')
      })
      this.server.listen(this.port, () => {
          console.log(`Running server on port ${this.port}`)
      })
  }
}
```

Модель Отеля:

```
import { Table, Column, Model, Min, Max, HasMany } from 'sequelize-typescript'
import Booking from '../booking/Booking'

@Table
export default class Hotel extends Model {
  @Column
  name: string

  @Column
  description: string

  @Min(1)
  @Max(5)
  @Column
  rating: number

  @Min(0)
  @Column
  capacity: number

  @Column
  city: string
```

```
  @HasMany(() => Booking)
  bookings: Booking[]
}
```

Сервис для получения данных по отелям:

```
import Hotel from '../../models/hotel/Hotel'
import { sequelize } from '../../config/config'

export default class BookingService {

    private repo = sequelize.getRepository(Hotel)

    add(hotel: any) {
        return this.repo.create(hotel)
    }

    getFiltered(city_param: string, min_rating: number) {
        const { Op } = require("sequelize")
        return this.repo.findAll( { where: { city: city_param, rating: { [Op.gte]:
min_rating } } } )
    }
}
```

Контроллер для запроса отелей:

```
import HotelService from '../../services/hotel/index'

export default class HotelController {

    private service = new HotelService()

    post = async (request: any, response: any) => {
        try {
            await this.service.add(request.body)
            response.send('Successfully added')
        } catch (error: any) {
            response.status(400).send(error.message)
        }
    }

    get = async (request: any, response: any) => {
        try {
            const data = await this.service.getFiltered(request.query.city,
request.query.rating)
            response.send(data)
        } catch (error: any) {
            response.status(400).send(error.message)
        }
    }
}
```

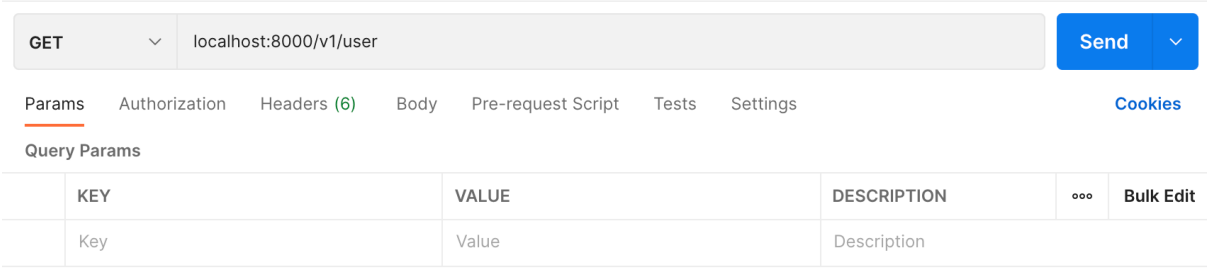
Конфиг:

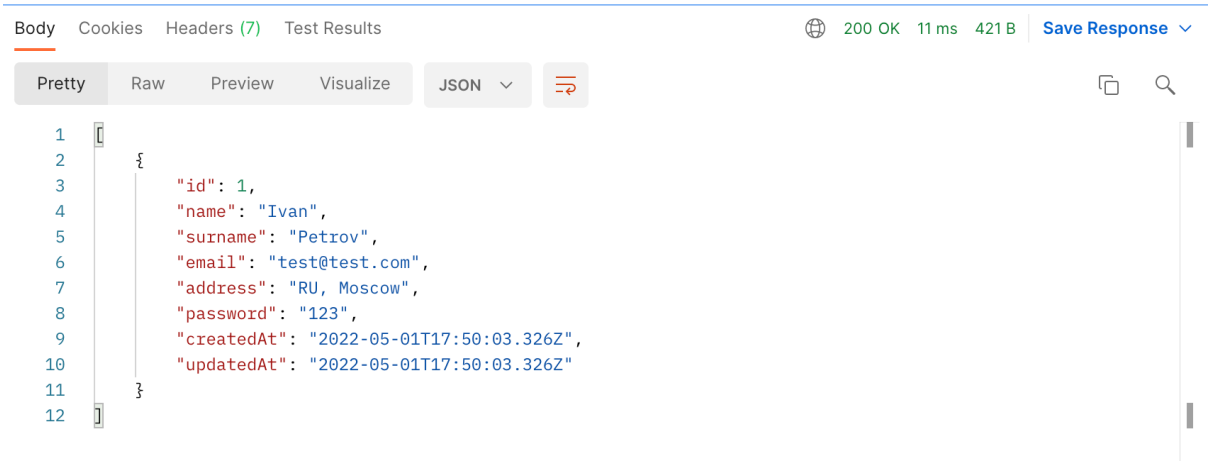
```
import { Sequelize } from 'sequelize-typescript'
import User from '../models/user/User'
import Hotel from '../models/hotel/Hotel'
import Booking from '../models/booking/Booking'

export const sequelize = new Sequelize({
  database: 'example_db',
  dialect: 'sqlite',
  username: 'root',
  password: '',
  storage: ':memory:',
  models: [User, Hotel, Booking],
  repositoryMode: true
})
```

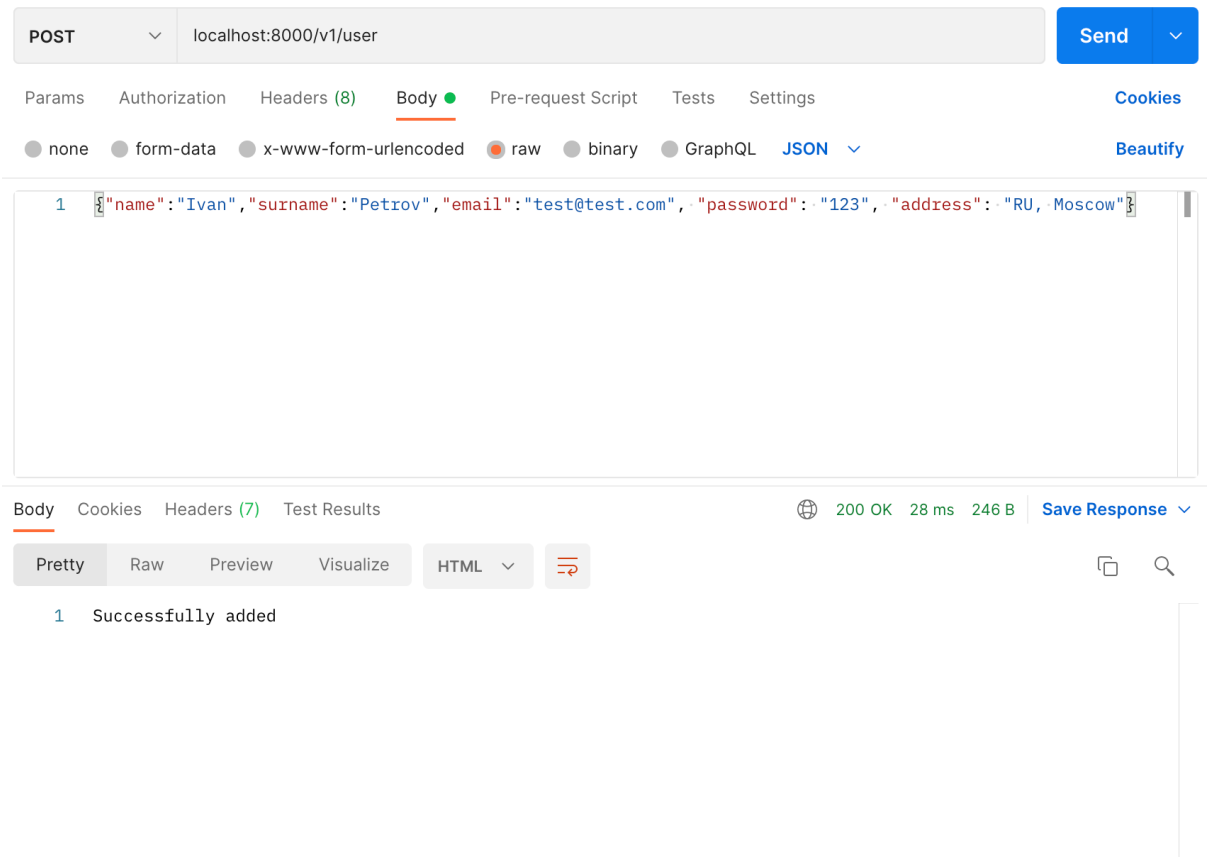
Настройка роутов:

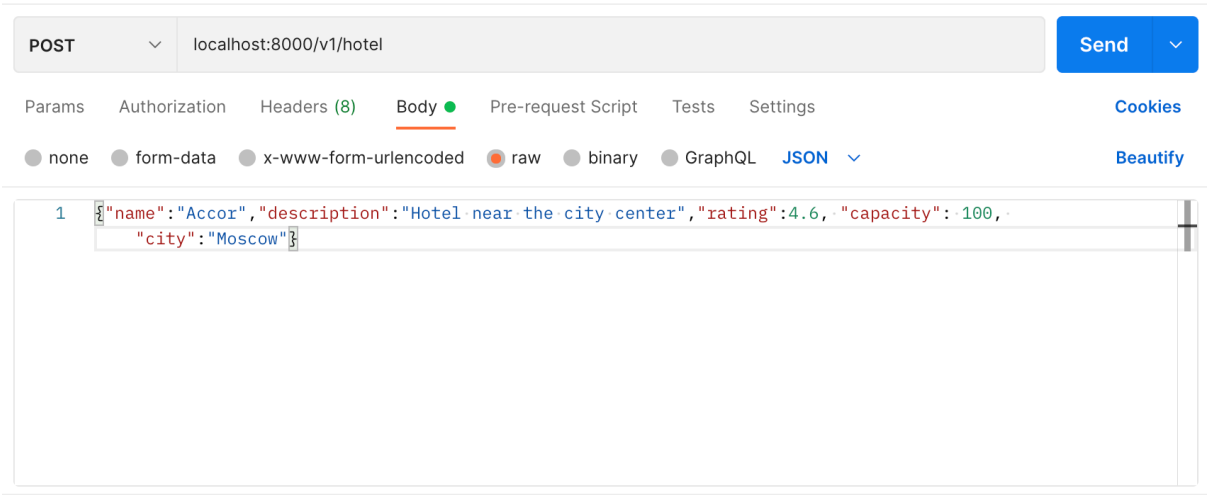
```
import express from "express"
import UserController from '../controllers/user/index'
import HotelController from '../controllers/hotel/index'
import BookingController from '../controllers/booking/index'
import AuthController from "../controllers/auth"

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

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

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)

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

export default router
```

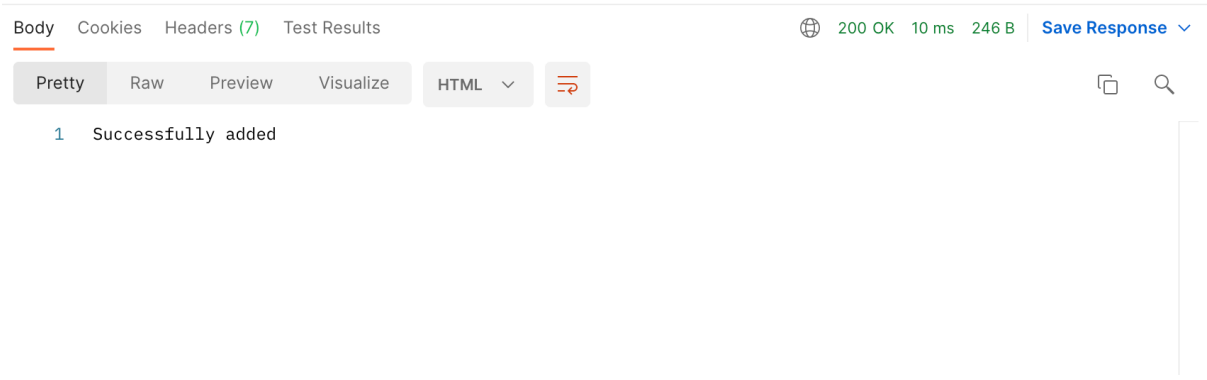
Демонстрация работы

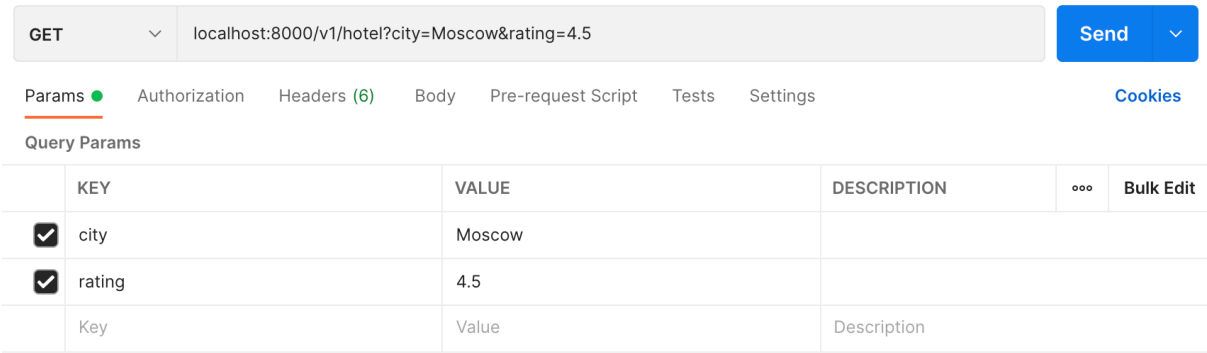


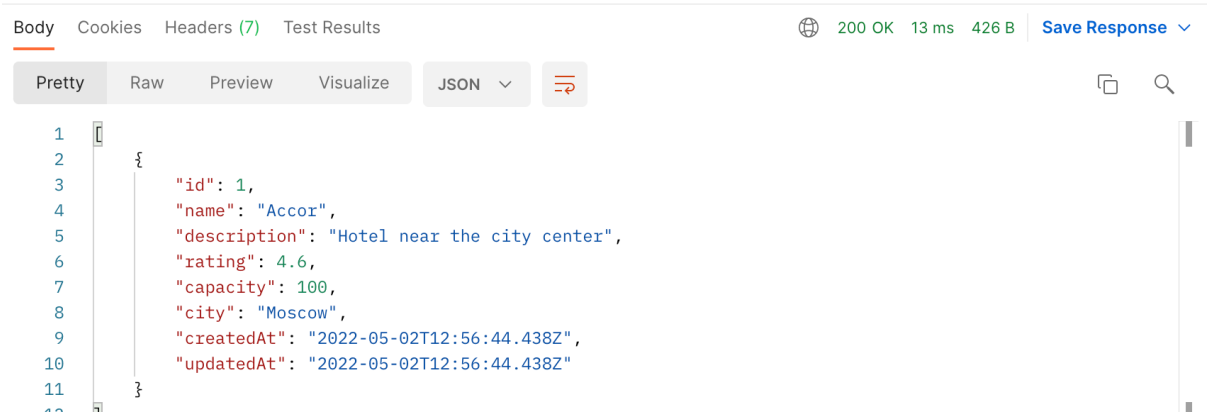


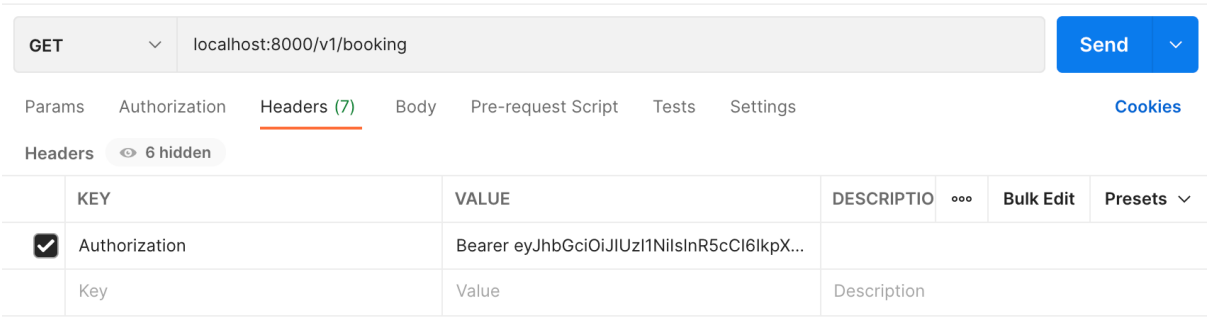


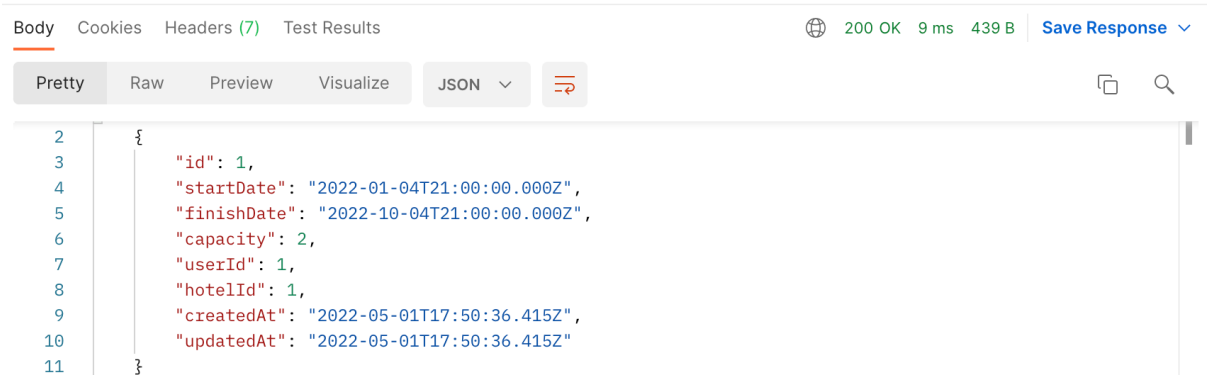

POST
localhost:8000/v1/hotel
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
1 {"name":"Accor","description":"Hotel near the city center","rating":4.6, "capacity": 100,
"city":"Moscow"}

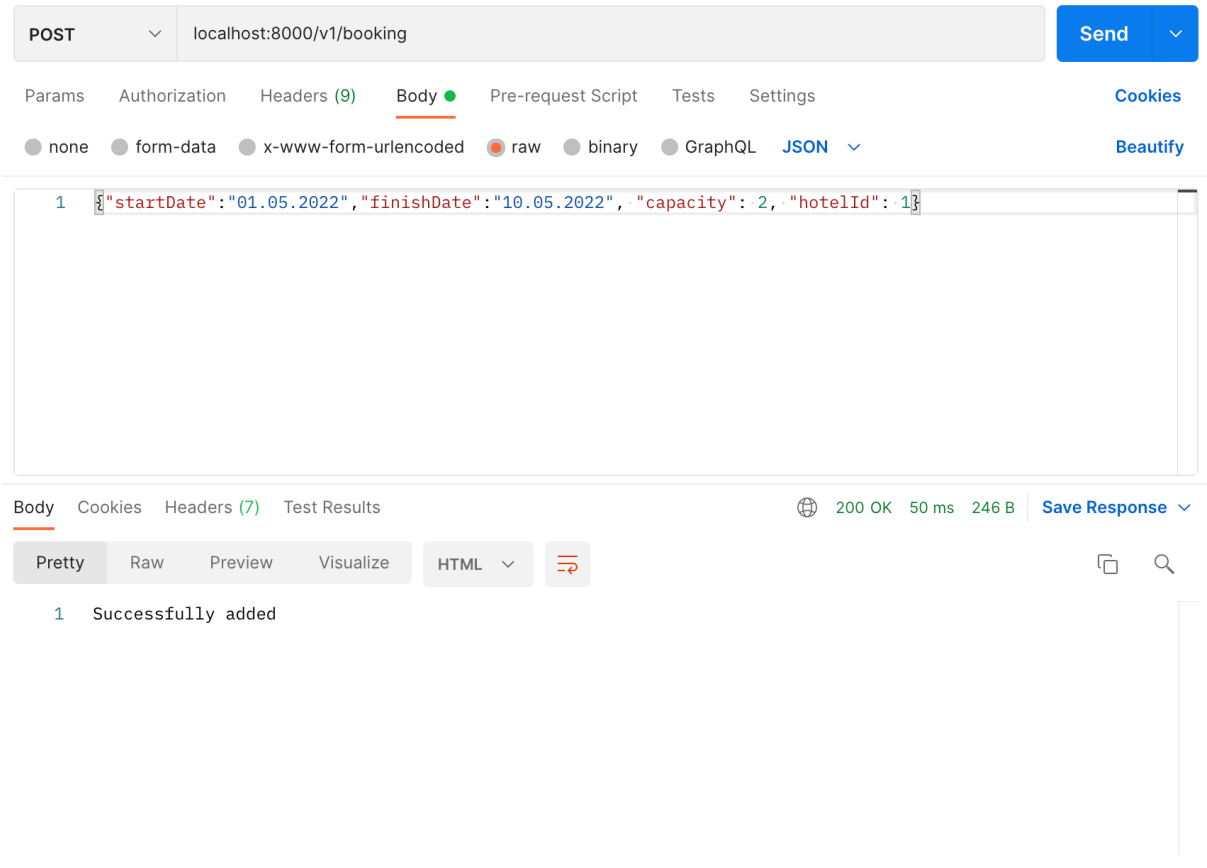

Body
Cookies
Headers (7)
Test Results
200 OK
10 ms
246 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
1 Successfully added

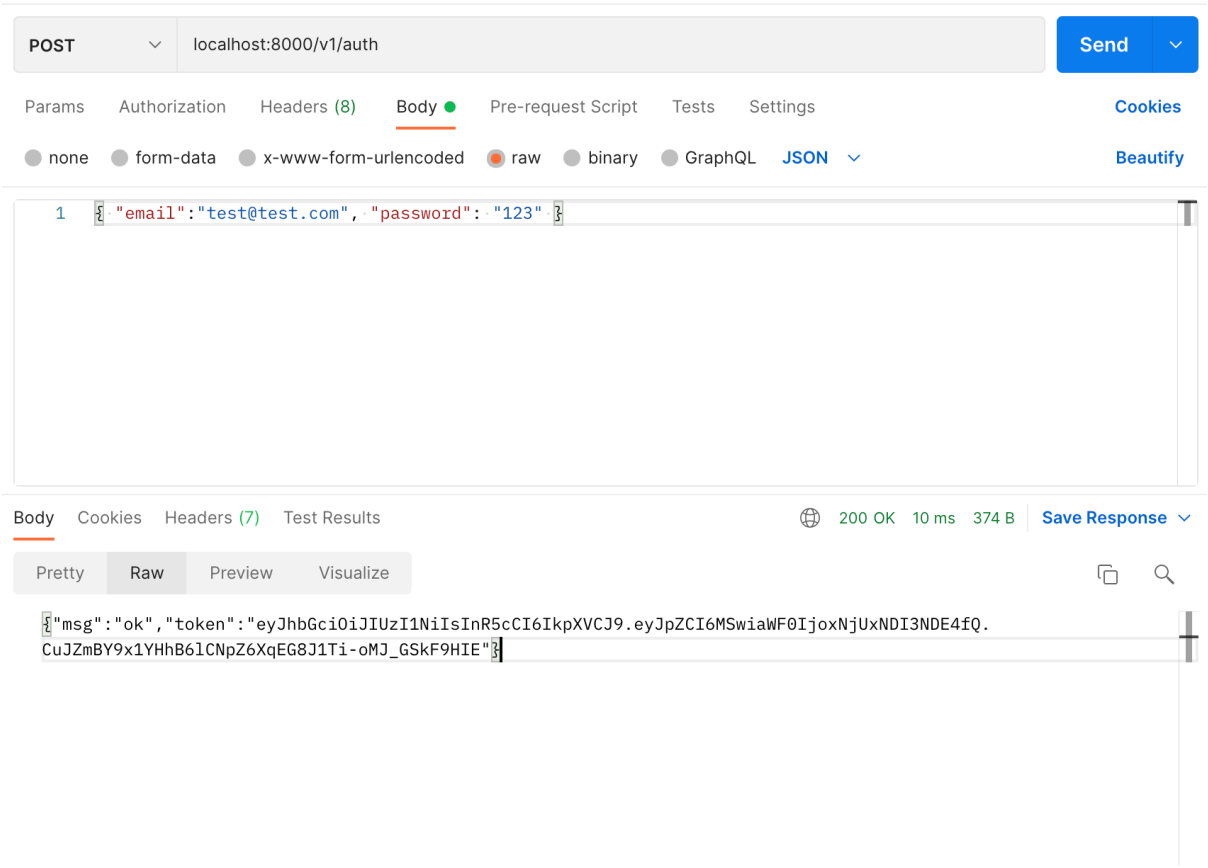

GET
localhost:8000/v1/hotel?city=Moscow&rating=4.5
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
city
Moscow
rating
4.5
Key
Value
Description


Body
Cookies
Headers (7)
Test Results
200 OK
13 ms
426 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
1 [
2 {
3 "id": 1,
4 "name": "Accor",
5 "description": "Hotel near the city center",
6 "rating": 4.6,
7 "capacity": 100,
8 "city": "Moscow",
9 "createdAt": "2022-05-02T12:56:44.438Z",
10 "updatedAt": "2022-05-02T12:56:44.438Z"
11 }


GET
localhost:8000/v1/booking
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Headers
6 hidden
KEY
VALUE
DESCRIPTIO
Bulk Edit
Presets
Authorization
Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpX...
Key
Value
Description


Body
Cookies
Headers (7)
Test Results
200 OK
9 ms
439 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": 1,
"startDate": "2022-01-04T21:00:00.000Z",
"finishDate": "2022-10-04T21:00:00.000Z",
"capacity": 2,
"userId": 1,
"hotelId": 1,
"createdAt": "2022-05-01T17:50:36.415Z",
"updatedAt": "2022-05-01T17:50:36.415Z"
}


POST
localhost:8000/v1/booking
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{"startDate":"01.05.2022","finishDate":"10.05.2022", "capacity": 2, "hotelId": 1}
Body
Cookies
Headers (7)
Test Results
200 OK
50 ms
246 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
Successfully added



Вывод

По итогам работы был реализован RESTful API средствами Express, Typescript, Sequelize. Для авторизации использовались библиотеки Passport и Passport JWT.