САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

по лабораторной работе №2

Выполнил:

Поляков Андрей

Группа К33402

Проверил: Добряков Д. И.

Санкт-Петербург

Задача

Для выполнения работы я выбрал вариант «Платформа для поиска и бронирования номера в отеле/квартире/хостеле». В ходе работы необходимо реализовать RESTful API средствами express + typescript, используя ранее написанный boilerplate.

Ход работы

Файл соге:

```
import express from "express'
import { createServer, Server } from "http"
import routes from "../routes/index'
import UserService from '../services/user/index'
import { sequelize } from '../config/config'
export const jwtOptions: any = {}
export default class App {
   public port: number
   public host: string
   private app: express.Application
   private server: Server
    constructor(port = 8080, 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)
         let user = await service.getById(jwt_payload.id)
if (user) {
          next(null, user)
           next(null, false)
       passport.use(strategy)
       app.use(bodyParser.urlencoded({ extended: false }))
      app.use(bodyParser.json())
       app.use(passport.initialize())
   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 express from "express
       import routes from "../routes/index"
import UserService from '../services/user/index'
import { sequelize } from '../config/config'
       export const jwtOptions: any = {}
       export default class App [
            private app: express.Application
private server: Server
            constructor(port = 8080, host = "localhost") {
                  this.port = port
this.host = host
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
                  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)
                    next(null, false)
}
                  passport.use(strategy)
                  app.use(bodyParser.urlencoded({ extended: false }))
                  app.use(bodyParser.json())
```

Модель бронирования:

```
import { Table, Column, Model, IsDate, Min, ForeignKey, BelongsTo } from 'sequelize-typescript
import User from '../user/User'
import Hotel from '../hotel/Hotel'
export default class Booking extends Model ₹
 @IsDate
 @Column
 startDate: Date
 @IsDate
 @Column
 endDate: Date
 @Min(1)
  @Column
  capacity: number
  @ForeignKey(() => User)
  @Column
  userId: number
  @BelongsTo(() => User)
  user: User
  @ForeignKey(() => Hotel)
  @Column
  hotelId: number
  @BelongsTo(() => Hotel)
  hotel: Hotel
}
```

Модель пользователя:

```
import { Table, Column, Model, HasMany } from 'sequelize-typescript'
     import Booking from '../booking/Booking'
     @Table
     export default class User extends Model {
 5
6
       @Column
       nickname: string
       @Column
10
       name: string
11
12
       @Column
13
       surname: string
14
15
       @Column
       email: string
17
18
       @Column
19
       password: string
20
21
       @HasMany(() => Booking)
22
       bookings: Booking[]
23
```

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/index"
const router: express.Router = express.Router()
const passport = require('passport')
const authController = new AuthController()
 .route('/user')
 .get(userController.get)
 .post(userController.post)
 .route('/hotel')
 .get(hotelController.get)
  .post(hotelController.post)
  .route('/booking')
 .get(passport.authenticate('jwt', { session: false }), bookingController.get)
  .post(passport.authenticate('jwt', { session: false }), bookingController.post)
  .route('/auth')
  .post(authController.post)
```

Сервис, для получения данных о бронированиях:

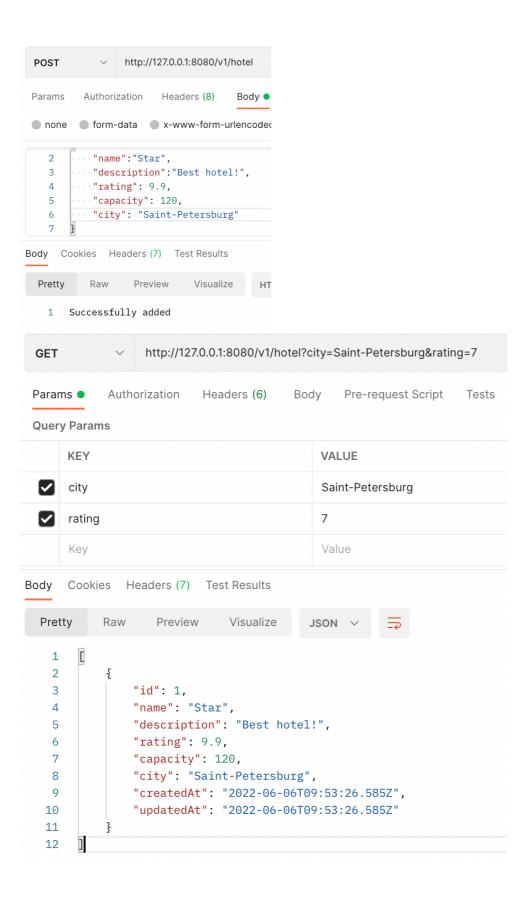
```
import Booking from '../../models/booking/Booking'
      import { sequelize } from '../../config/config'
 2
     export default class BookingService {
 4
         private repo = sequelize.getRepository(Booking)
         add(booking: any) {
 9
              return this.repo.create(booking)
10
11
12
         getForUser(user: number) {
13
              return this.repo.findAll({ where: { userId: user } })
14
15
```

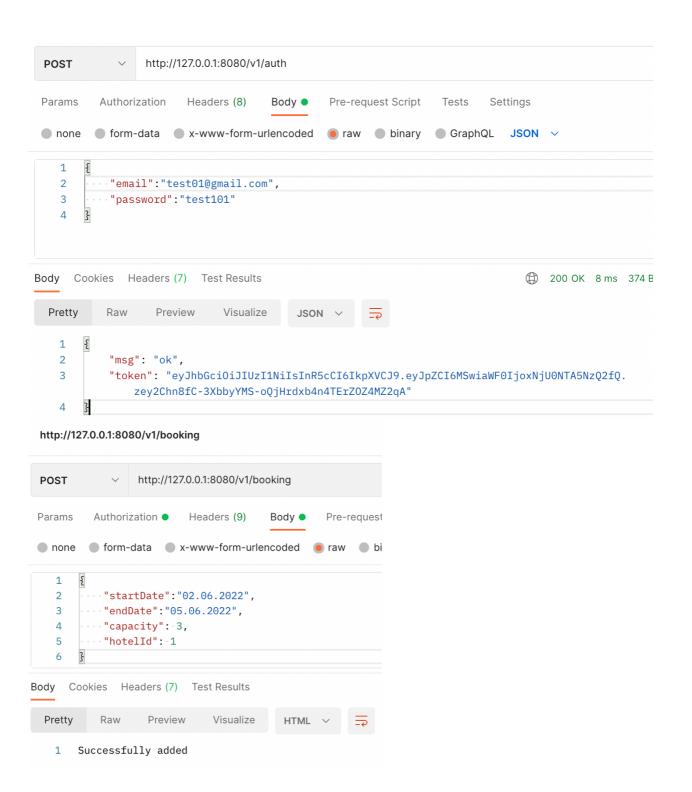
Контроллер для бронирования:

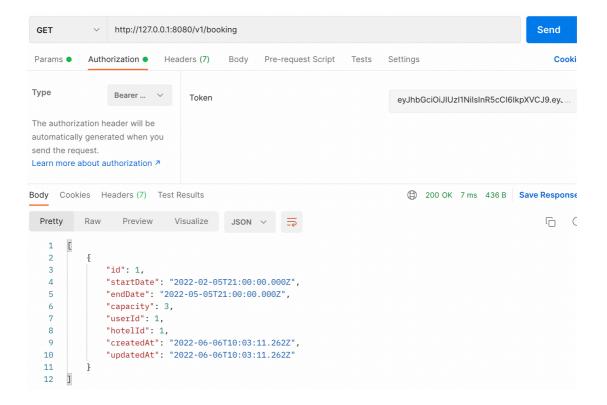
```
import BookingService from '../../services/booking/index'
     export default class BookingController {
         private service = new BookingService()
         post = async (request: any, response: any) => {
             try {
                 const booking = request.body
10
                  booking.userId = request.user.id
11
                  await this.service.add(booking)
12
                  response.send('Successfully added')
13
             } catch (error: any) {
14
                  response.status(400).send(error.message)
15
16
17
         get = async (request: any, response: any) => {
18
19
             try {
20
                 const data = await this.service.getForUser(request.user.id)
21
                  response.send(data)
22
              } catch (error: any) {
23
                  response.status(400).send(error.message)
24
25
26
```

Примеры работы:

```
http://127.0.0.1:8080/v1/user
 POST
 Params
            Authorization
                            Headers (8)
                                             Body •
 none form-data x-www-form-urlencoded
    1
          "nickname":"test01",
    2
          "name": "testName",
    3
          "surname": "testSurname",
    4
          "password":"test101",
    5
           "email":"test01@gmail.com"
    6
    7
       Cookies Headers (7) Test Results
  Pretty
             Raw
                       Preview
                                    Visualize
                                                  HTI
         Added'test01'user
 http://127.0.0.1:8080/v1/user
                http://127.0.0.1:8080/v1/user
 GET
        Authorization
Params
                      Headers (6)
                                        Pre-request Script
                                  Body
                                                         Tests
Query Params
     KEY
                                       VALUE
                                       Value
     Key
Body Cookies Headers (7) Test Results
  Pretty
          Raw
                           Visualize
                 Preview
                                      JSON V
       1
   2
          {
   3
              "id": 1,
              "nickname": "test01",
              "name": "testName",
   5
              "surname": "testSurname",
   6
   7
              "email": "test01@gmail.com",
   8
              "password": "test101",
              "createdAt": "2022-06-06T09:50:39.986Z",
   9
              "updatedAt": "2022-06-06T09:50:39.986Z"
  10
  11
  12
```







Вывод

В ходе работы был реализован RESTful API средствами Express, Typescript и Sequelize. Для авторизация использовался Passport и Passport JWT.