САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Фронд-энд разработка

Отчет

Лабораторная работа 2

Выполнил:
Комиссаров Александр
К33402

Проверил:
Добряков Д. И.

Санкт-Петербург

2022 г.

Задача

По выбранному варианту необходимо будет реализовать RESTful API средствами express + typescript (используя ранее написанный boilerplate). Выбранный вариант – сервис бронирования отелей.

Ход работы

Добавлены модели отеля и резервирования
Пример модели:

```
import { Table, Column, Model, IsDate, Min, ForeignKey, BelongsTo } from 'sequelize-typescript'
import User from '../user/User'
import Hotel from '../hotel/Hotel'

@Table
export default class Booking extends Model {
  @IsDate
  @Column
  startDate: Date

  @IsDate
  @Column
  finishDate: Date

  @Min(1)
  @Column
  capacity: number

  @ForeignKey(() => User)
  @Column
  userId: number

  @BelongsTo(() => User)
  user: User

  @ForeignKey(() => Hotel)
  @Column
  hotelId: number

  @BelongsTo(() => Hotel)
  hotel: Hotel
}
```

Соответственно, для новых моделей добавлены контроллеры с методами post и get.

Пример контроллера:

```
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
            booking.userId = request.user.id
            await this.service.add(booking)
            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.getForUser(request.user.id)
            response.send(data)
        } catch (error: any) {
            response.status(400).send(error.message)
        }
    }
}
```

Также добавлен контроллер авторизации:

```
import UserService from "../../services/user";
import { jwtOptions } from "../../core/index";

export default class AuthController {

    post = async (request: any, response: any) => {
        const { email, password } = request.body
        if (email && password) {
            const service = new UserService()
            let user = await service.getByEmail(email)
            if (!user) {
                response.status(401).json({ msg: 'No such user found', user })
            }
            if (user!.password === password) {
                let payload = { id: user!.id }
                const jwt = require('jsonwebtoken')
                let token = jwt.sign(payload, jwtOptions.secretOrKey)
                response.json({ msg: 'ok', token: token })
            } else {
                response.status(401).json({ msg: 'Password is incorrect' })
            }
        }
    }
}
```

Реализованы сервисы для добавления и фильтрации отелей, создания резервирования. Сервис пользователя расширен, добавлена возможность поиска по ID и электронной почте

```
import User from '../../models/user/User'
import { sequelize } from '../../config/config'

export default class UserService {

    private repo = sequelize.getRepository(User)

    add(name: string, surname: string, email: string, password: string, address: string) {
        this.repo.create({ name: name, surname: surname, email: email, password: password, address: address })
    }

    getAll() {
        return this.repo.findAll()
    }

    getById(id_param: number) {
        return this.repo.findOne({ where: { id: id_param }})
    }

    getByEmail(email_param: string) {
        return this.repo.findOne({ where: { email: email_param }})
    }
}
```

Добавлены роуты для новых страниц

```
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
```

Вывод

В результате работы был реализован RESTful API для сервиса бронирования отелей средствами express + typescript, в качестве основы использовался ранее написанный boilerplate.