САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

по лабораторной работе №2

Выполнил:

Кузгиев Адам

Группа
K33402

Проверил:
Добряков Д. И.

Санкт-Петербург

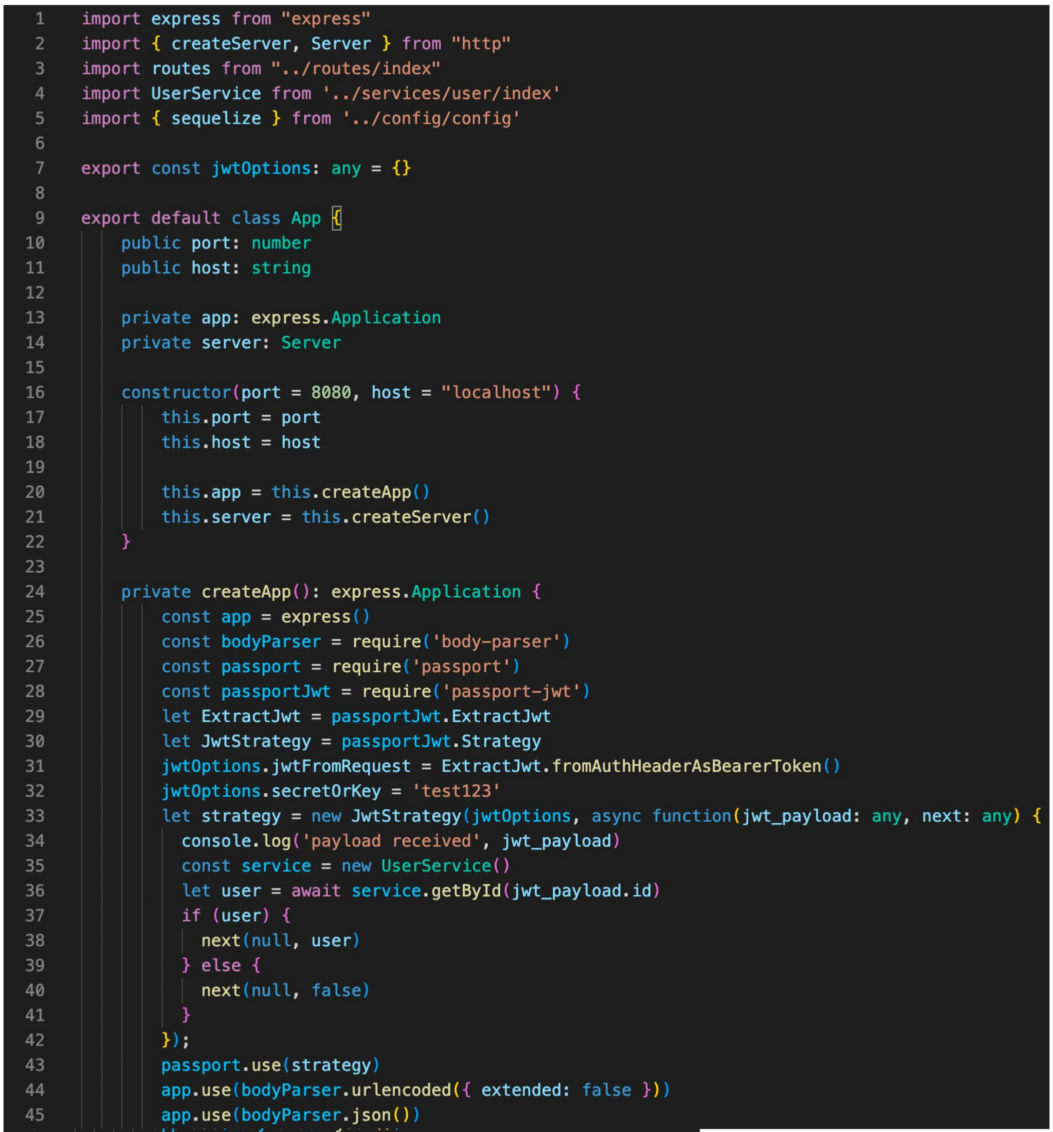
2022 г.

Задача

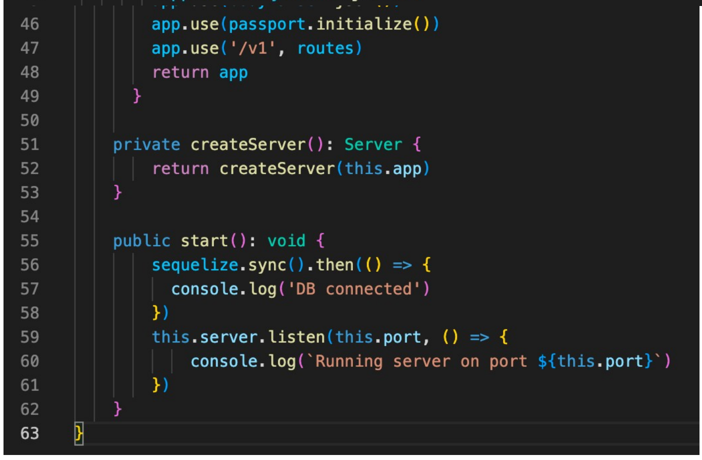
Для выполнения работы я выбрал вариант «Платформа для поиска и бронирования номера в отеле/квартире/хостеле». В ходе работы необходимо реализовать RESTful API средствами express + typescript, используя ранее написанный boilerplate.

Ход работы

Файл core:

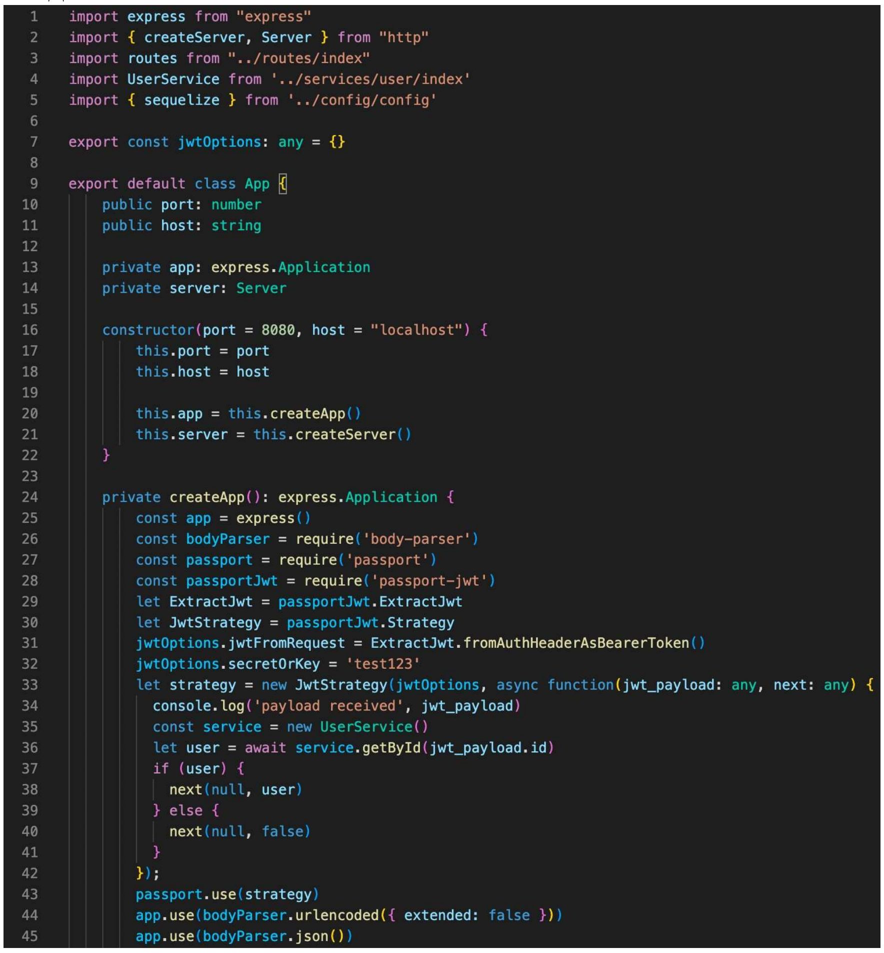


```
import express from "express"
import { createServer, Server } from "http"
import routes from "../routes/index"
import UserService from '../services/user/index'
import { sequelize } from '../config/config'

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)
          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())
```



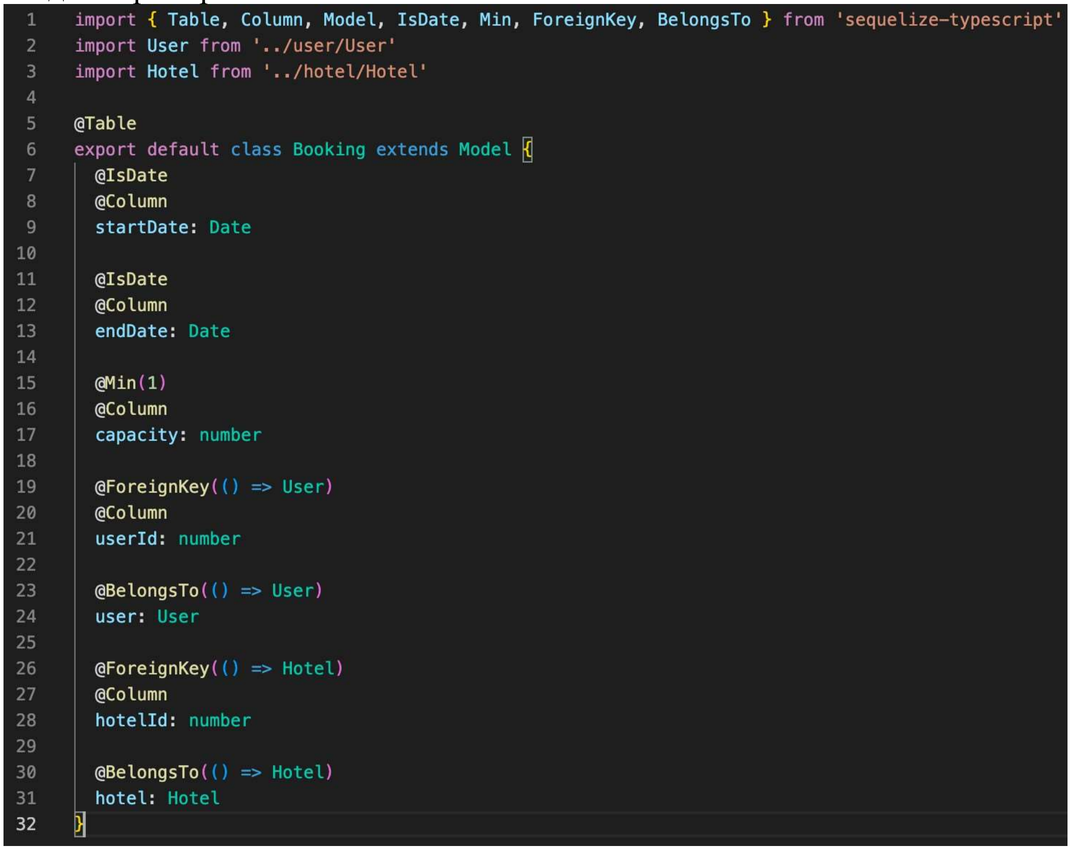
```
        app.use(passport.initialize())
        app.use('/v1', routes)
        return app
    }

    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:



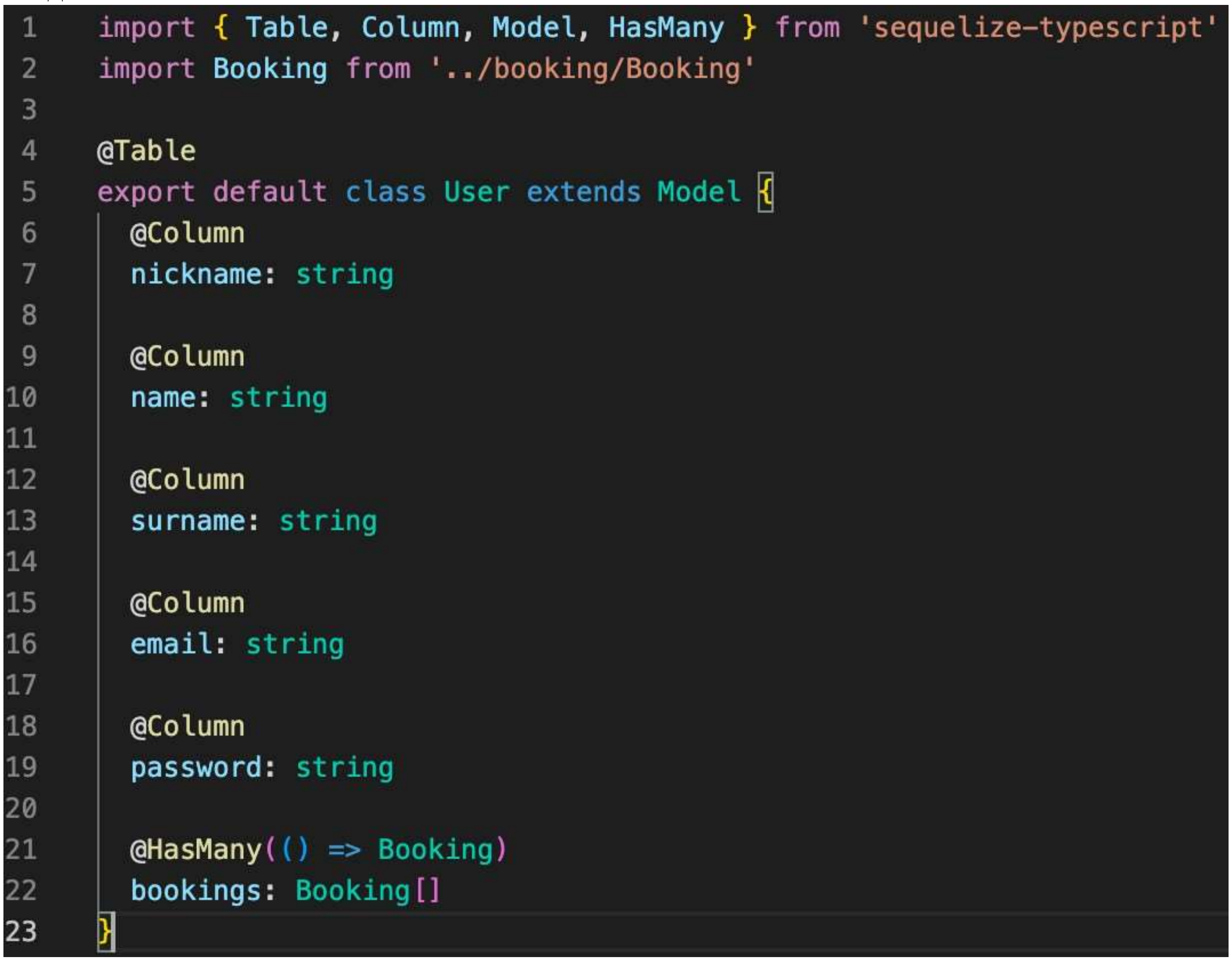
```
import express from "express"
import { createServer, Server } from "http"
import routes from "../routes/index"
import UserService from '../services/user/index'
import { sequelize } from '../config/config'

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)
          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())
```

Модель бронирования:



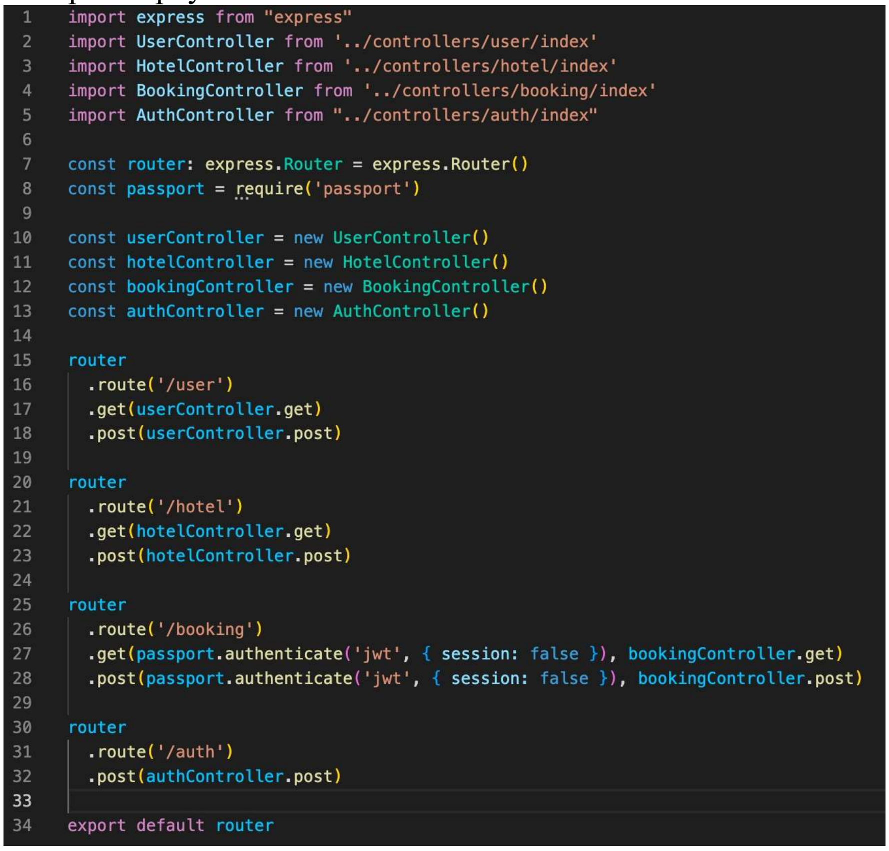
```
import { Table, Column, Model, IsDate, Min, ForeignKey, BelongsTo } from 'sequelize-typescript'
import User from '../user/User'
import Hotel from '../hotel/Hotel'

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

  @IsDate
  @Column
  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:



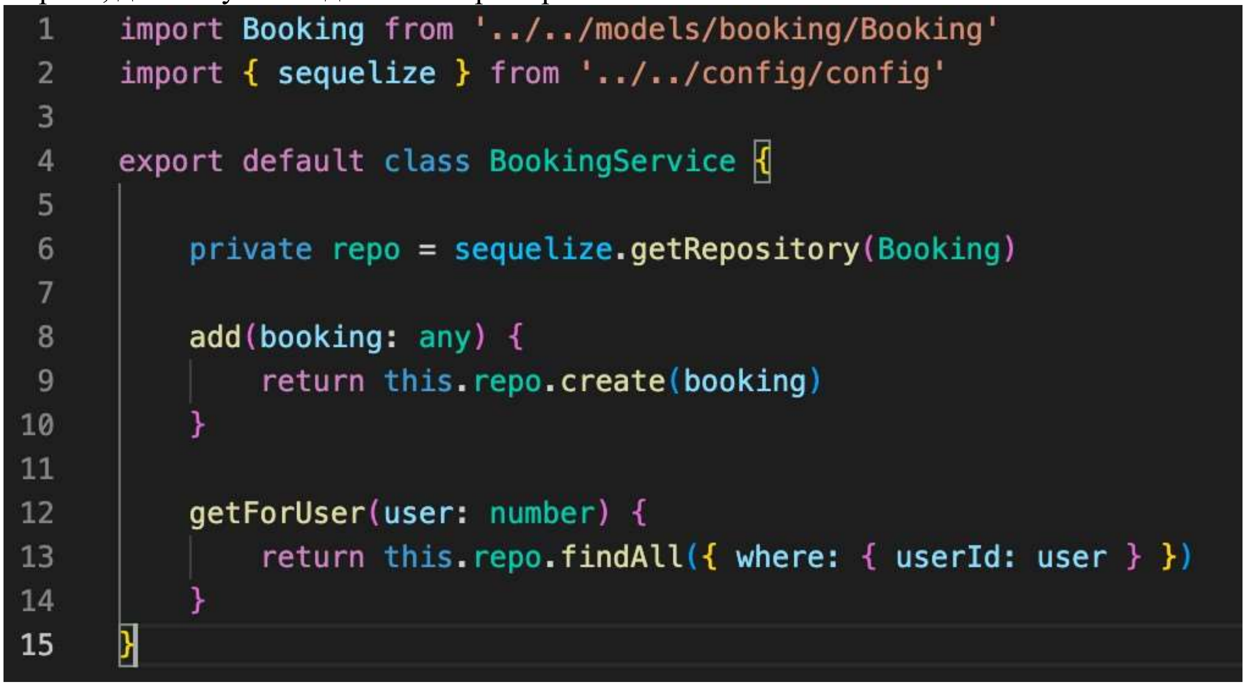
```
import { Table, Column, Model, HasMany } from 'sequelize-typescript'
import Booking from '../booking/Booking'

@Table
export default class User extends Model {
  @Column
  nickname: string

  @Column
  name: string

  @Column
  surname: string

  @Column
  email: string

  @Column
  password: string

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

Настройка роутов:



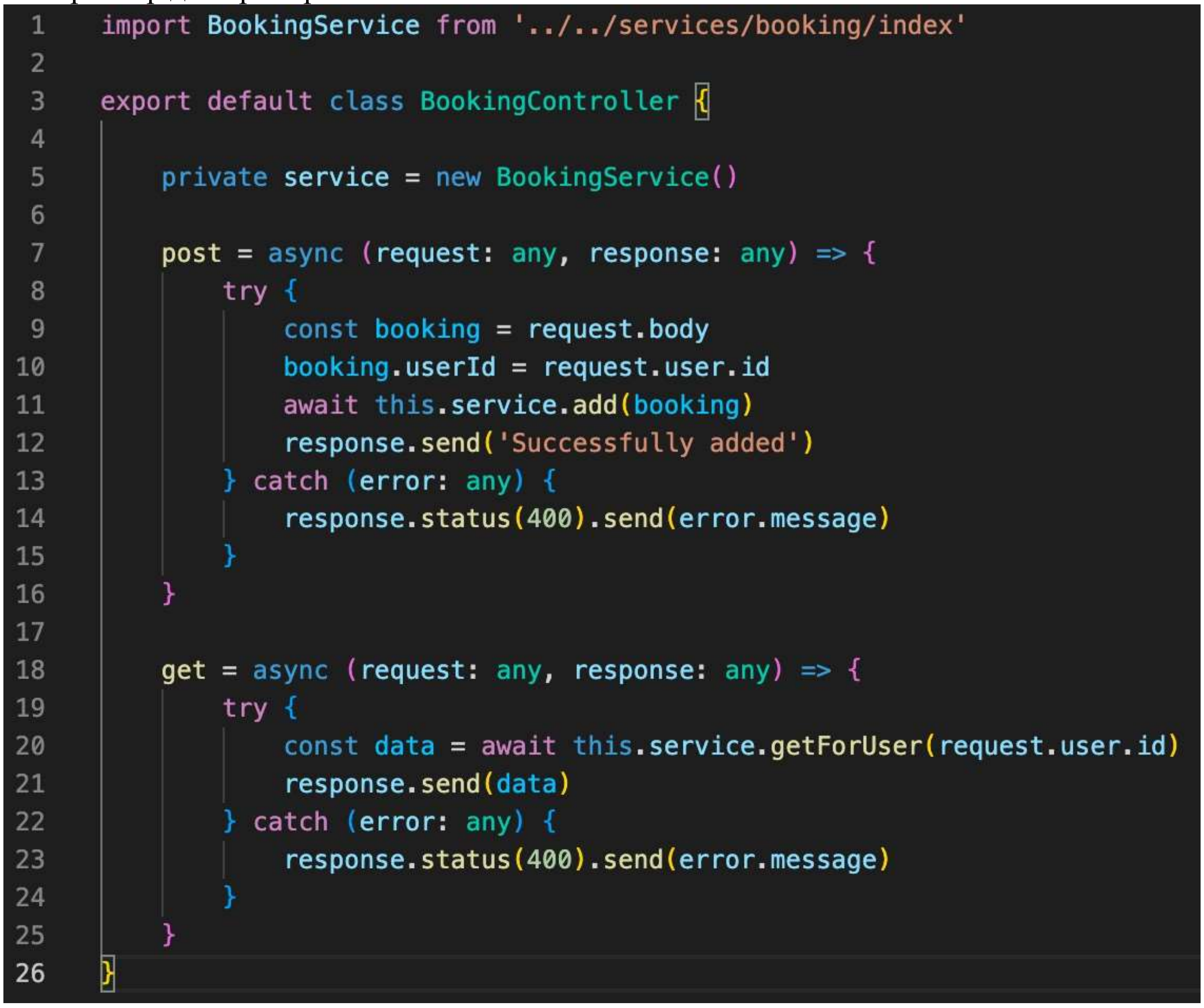
```
import express from "express"
import UserController from '../controllers/user/index'
import HotelController from '../controllers/hotel/index'
import BookingController from '../controllers/booking/index'
import AuthController from "../controllers/auth/index"

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

const 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
```

Сервис, для получения данных о бронированиях:



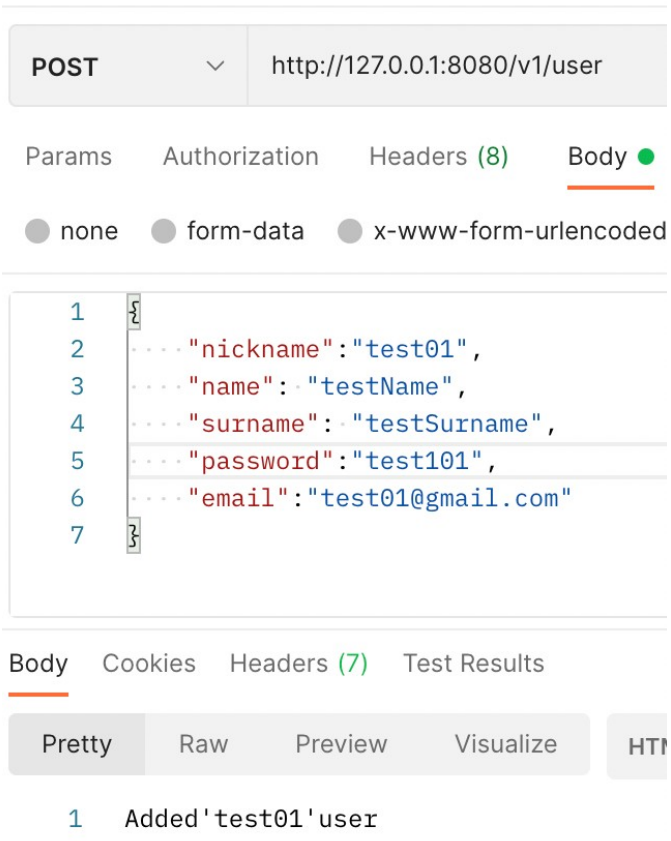
```
import Booking from '../../models/booking/Booking'
import { sequelize } from '../../config/config'

export default class BookingService {

    private repo = sequelize.getRepository(Booking)

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

    getForUser(user: number) {
        return this.repo.findAll({ where: { userId: user } })
    }
}
```

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
            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)
        }
    }
}
```

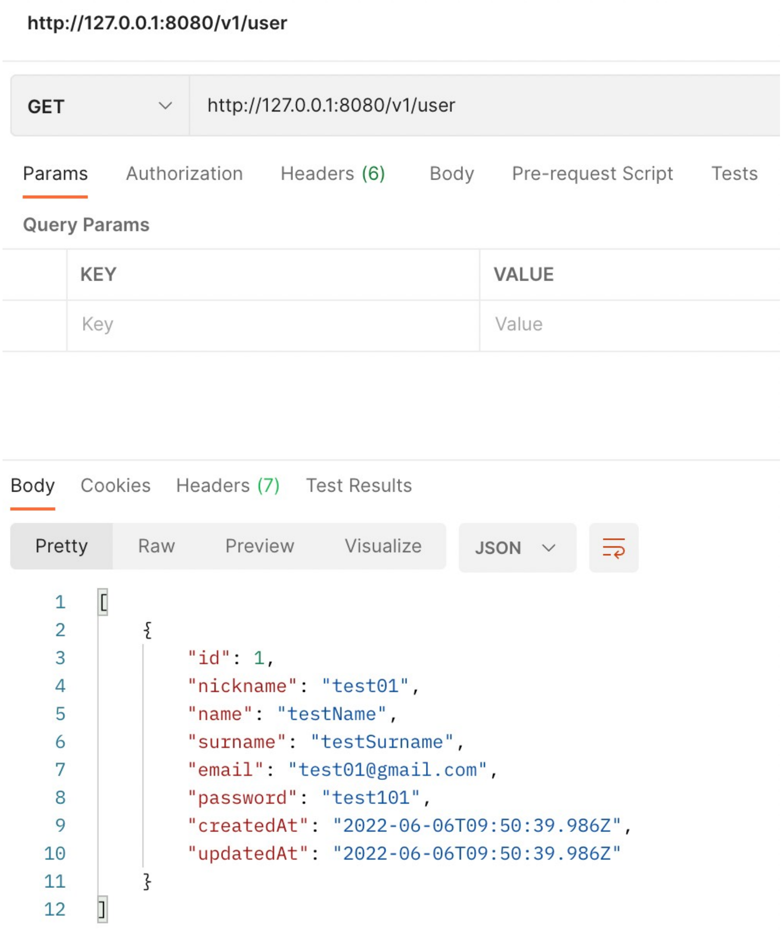
Примеры работы:



POST http://127.0.0.1:8080/v1/user

```
{
    "nickname":"test01",
    "name": "testName",
    "surname": "testSurname",
    "password":"test101",
    "email":"test01@gmail.com"
}
```

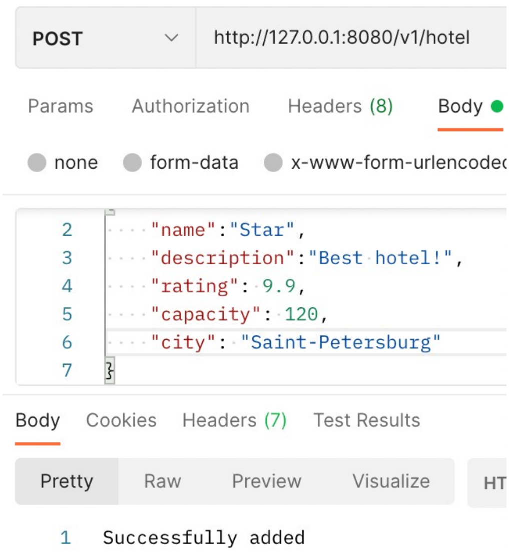
```
Added'test01'user
```



http://127.0.0.1:8080/v1/user

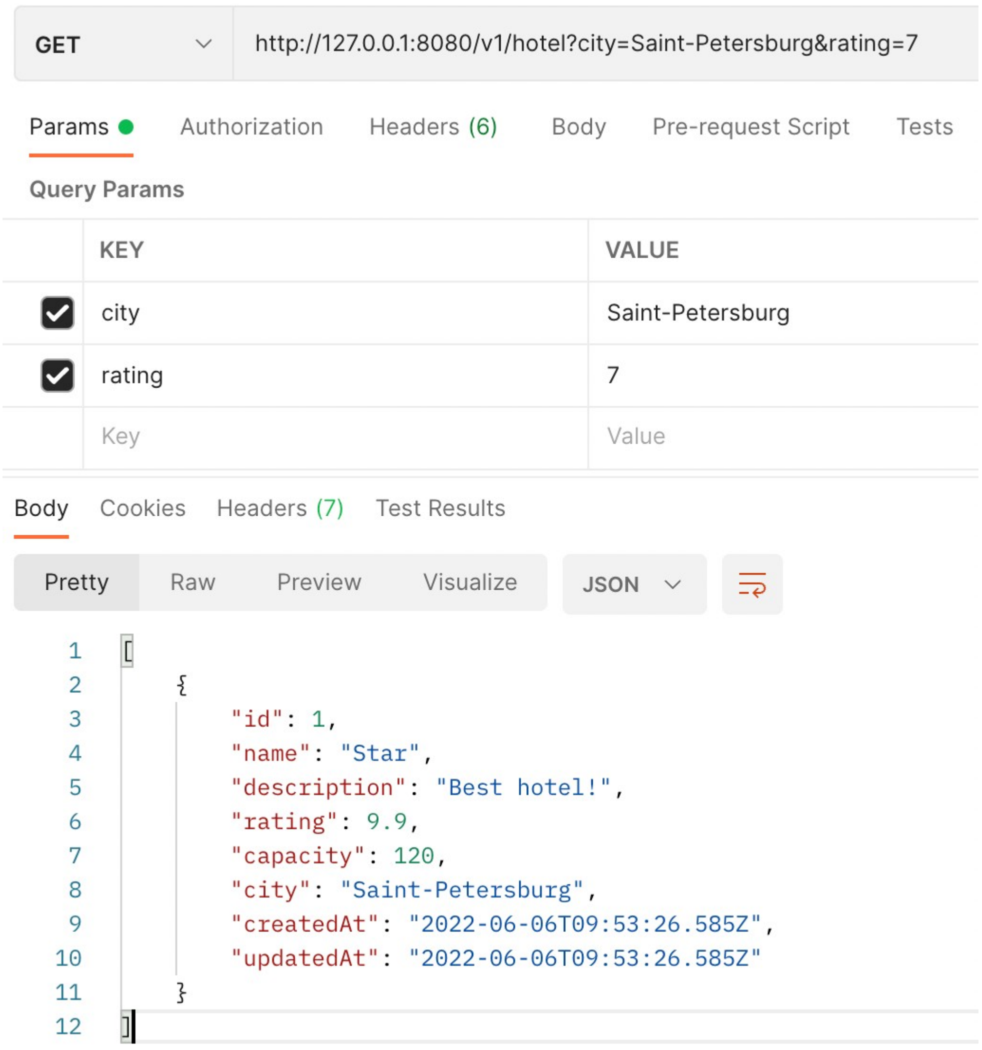
GET http://127.0.0.1:8080/v1/user

```
[
    {
        "id": 1,
        "nickname": "test01",
        "name": "testName",
        "surname": "testSurname",
        "email": "test01@gmail.com",
        "password": "test101",
        "createdAt": "2022-06-06T09:50:39.986Z",
        "updatedAt": "2022-06-06T09:50:39.986Z"
    }
]
```



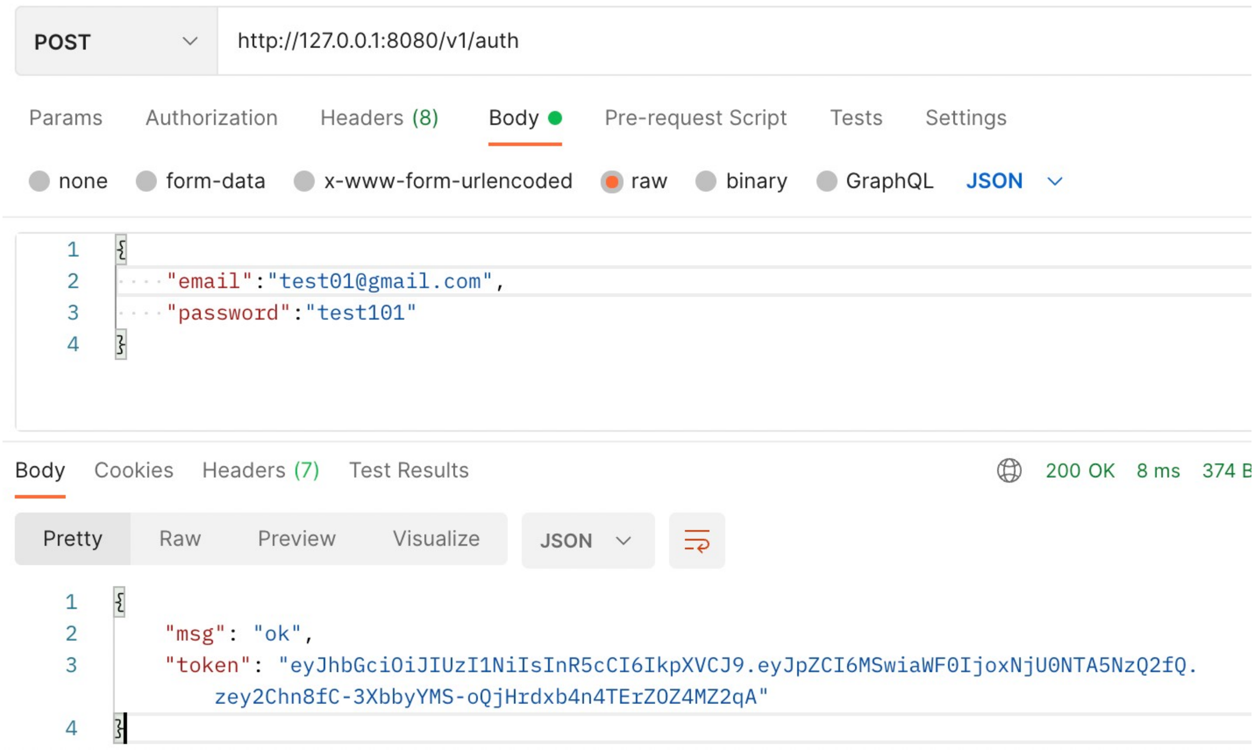
```
    "name":"Star",
    "description":"Best hotel!",
    "rating": 9.9,
    "capacity": 120,
    "city": "Saint-Petersburg"
}
```

```
Successfully added
```



	KEY	VALUE
✓	city	Saint-Petersburg
✓	rating	7
	Key	Value

```
[
    {
        "id": 1,
        "name": "Star",
        "description": "Best hotel!",
        "rating": 9.9,
        "capacity": 120,
        "city": "Saint-Petersburg",
        "createdAt": "2022-06-06T09:53:26.585Z",
        "updatedAt": "2022-06-06T09:53:26.585Z"
    }
]
```


POST http://127.0.0.1:8080/v1/auth

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

none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

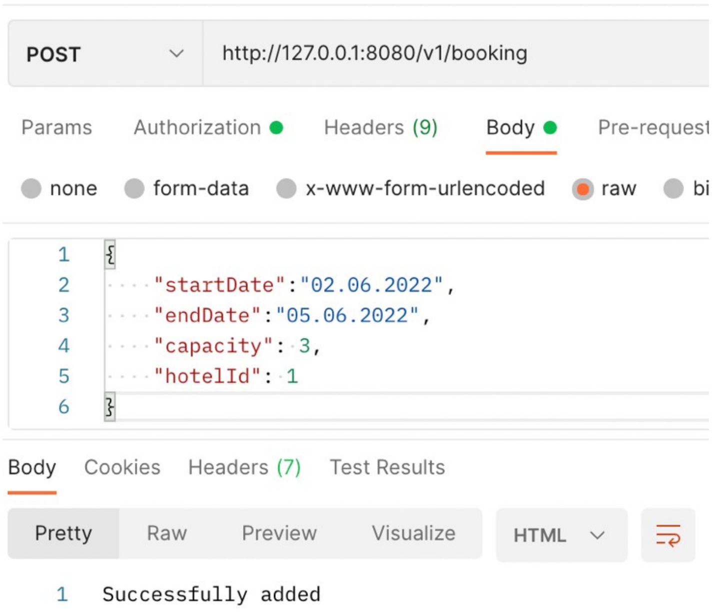
```
{
    "email":"test01@gmail.com",
    "password":"test101"
}
```

Body | Cookies | Headers (7) | Test Results   200 OK 8 ms 374 B

Pretty | Raw | Preview | Visualize | JSON

```
{
    "msg": "ok",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwiaWF0IjoxNjU0NTA5NzQ2fQ.
        zey2Chn8fC-3XbbyYMS-oQjHrdxb4n4TErZOZ4MZ2qA"
}
```

http://127.0.0.1:8080/v1/booking


POST http://127.0.0.1:8080/v1/booking

Params | Authorization | Headers (9) | Body | Pre-request

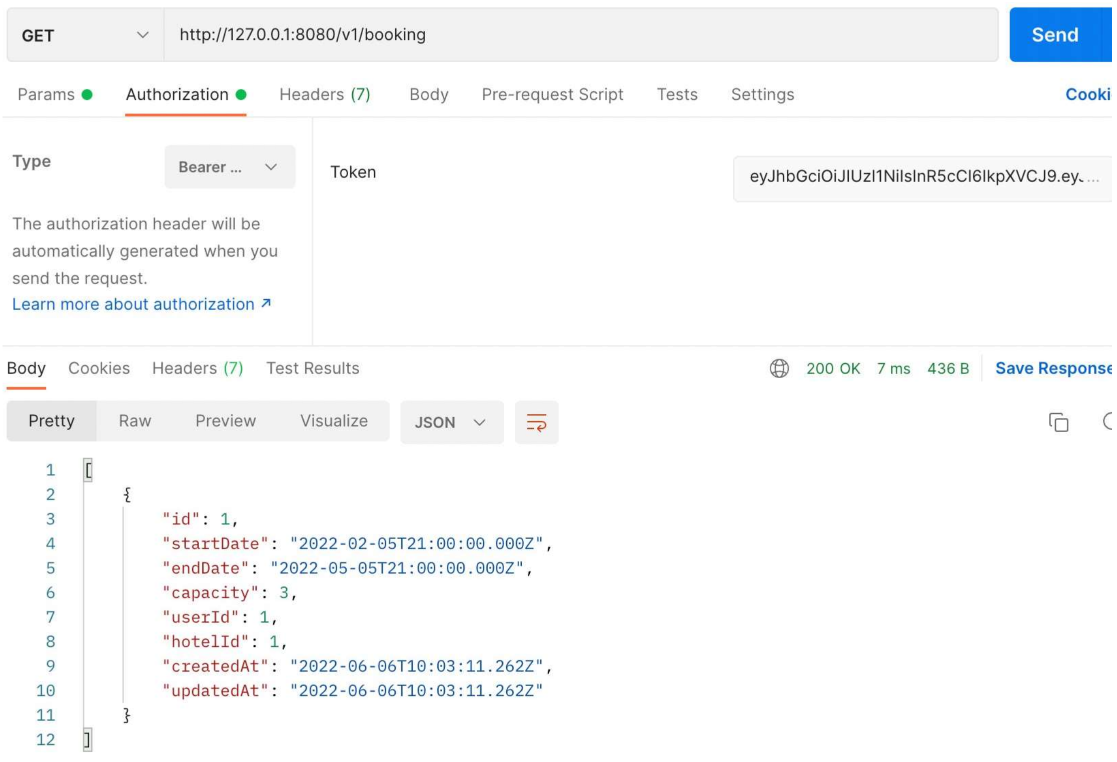
none | form-data | x-www-form-urlencoded | [✓ raw] | bi

```
{
    "startDate":"02.06.2022",
    "endDate":"05.06.2022",
    "capacity": 3,
    "hotelId": 1
}
```

Body | Cookies | Headers (7) | Test Results

Pretty | Raw | Preview | Visualize | HTML

```
Successfully added
```



http://127.0.0.1:8080/v1/booking

Type: Bearer ...

The authorization header will be automatically generated when you send the request.
Learn more about authorization ↗

Token: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ey...

200 OK 7 ms 436 B

```
[
    {
        "id": 1,
        "startDate": "2022-02-05T21:00:00.000Z",
        "endDate": "2022-05-05T21:00:00.000Z",
        "capacity": 3,
        "userId": 1,
        "hotelId": 1,
        "createdAt": "2022-06-06T10:03:11.262Z",
        "updatedAt": "2022-06-06T10:03:11.262Z"
    }
]
```

Вывод

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