Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ
О ЛАБОРАТОРНОЙ РАБОТЕ №3

по теме: Typescript основы
по дисциплине: Бэк-энд разработка

Специальность:
09.03.03 Мобильные и сетевые технологии

Проверил:
Добряков Д.И. _____
Дата: «10» июня 2022г.
Оценка _____

Выполнил:
студент группы К33401
Фоменко Иван

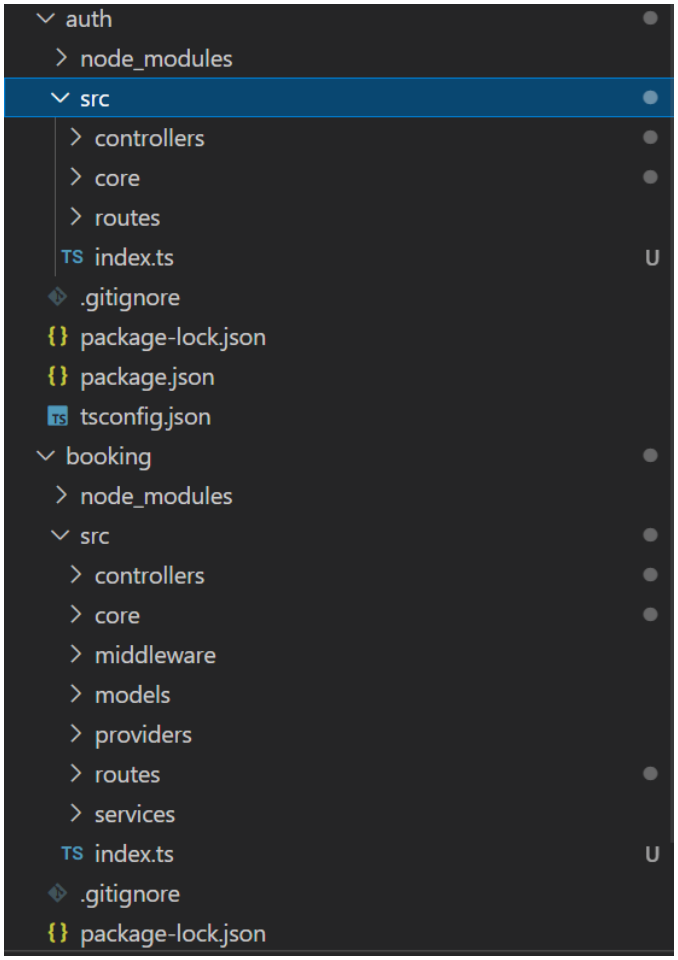
Санкт-Петербург 2022 г.

Цель работы:

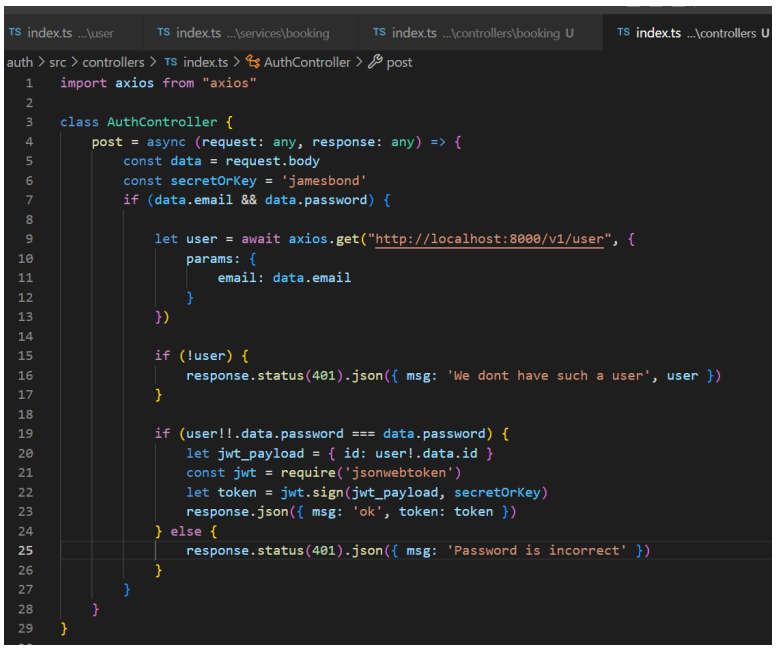
- Платформа для поиска и бронирования номера в отеле/квартире/хостеле (https://airbnb.com)
- Вход
- Регистрация
- Страница бронирований пользователя
- Страница для поиска номера с возможностью выбора города, времени заселения, количеству гостей
- Необходимо реализовать отдельный микросервис, выполняющий какую-либо содержательную функцию из всего арсенала функций вашего приложения

Ход работы

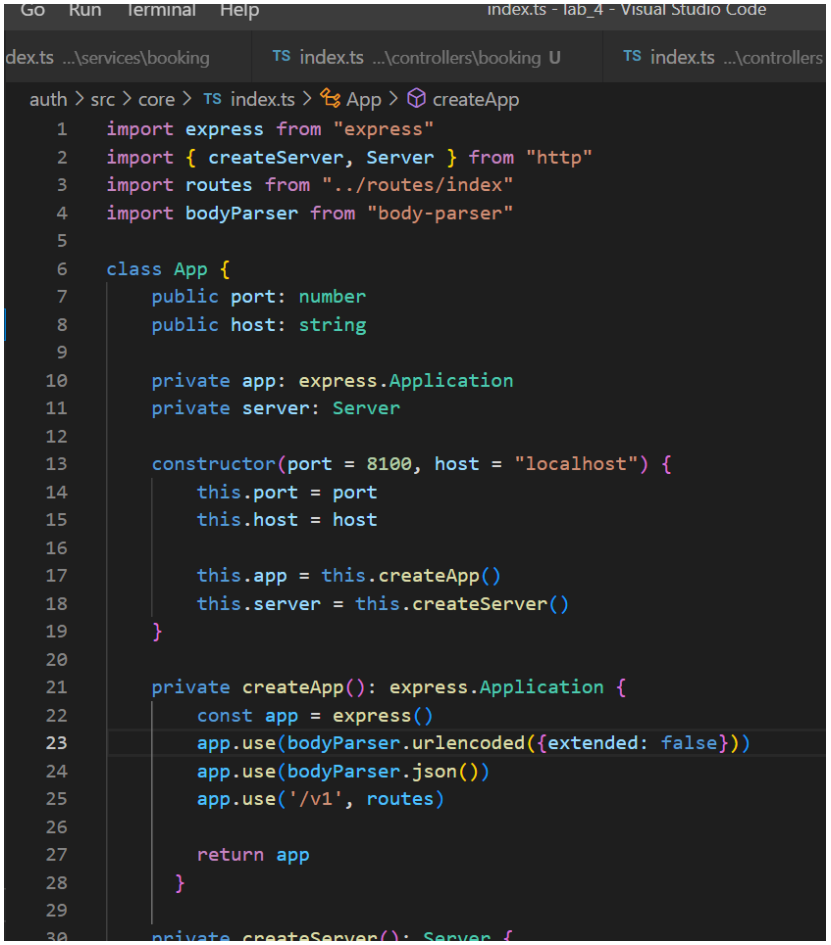
1. Два основных микросервиса



2. Controller auth



3. Core auth


Go Run Terminal Help
index.ts - lab_4 - Visual Studio Code
dex.ts ...\services\booking
TS index.ts ...\controllers\booking U
TS index.ts ...\controllers
auth > src > core > TS index.ts > App > createApp
import express from "express"
import { createServer, Server } from "http"
import routes from "../routes/index"
import bodyParser from "body-parser"
class App {
public port: number
public host: string
private app: express.Application
private server: Server
constructor(port = 8100, host = "localhost") {
this.port = port
this.host = host
this.app = this.createApp()
this.server = this.createServer()
}
private createApp(): express.Application {
const app = express()
app.use(bodyParser.urlencoded({extended: false}))
app.use(bodyParser.json())
app.use('/v1', routes)
return app
}


private createServer(): Server {
const server = createServer(this.app)
return server
}
public start(): void {
this.server.listen(this.port, () => {
console.log(`Running server on port ${this.port}`)
})
}
}
export default App

4. Router auth

auth > src > routes > TS index.ts > ...

```ts
import express from "express"
import AuthController from "../controllers/index"

const router: express.Router = express.Router()

const authController = new AuthController()

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

export default router
```

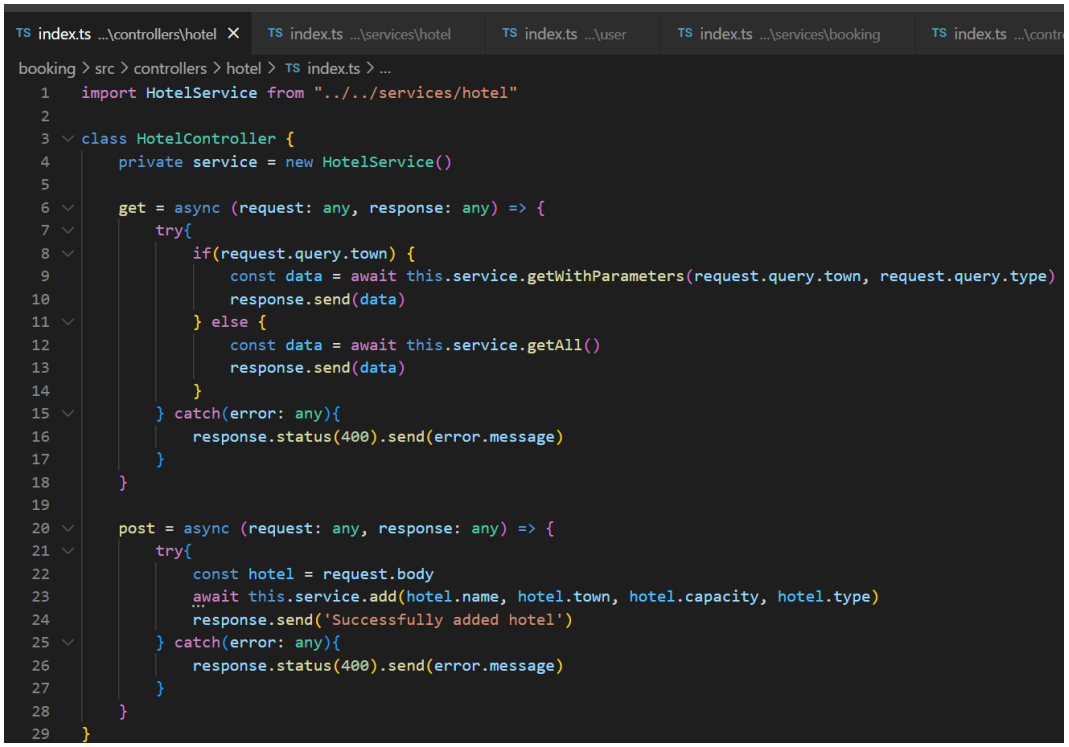
5. Controller booking

TS index.ts ...\controllers\hotel | TS index.ts ...\services\hotel | TS index.ts ...\user | TS index.ts ...\services\booking | TS index.ts

booking > src > controllers > booking > TS index.ts > BookingController > post

```ts
import BookingService from "../../services/booking"
import UserService from "../../services/user"
import HotelService from "../../services/hotel"

class BookingController {
    private bookingService = new BookingService()
    private userService = new UserService()
    private hotelService = new HotelService()

    get = async (request: any, response: any) => {
        try{
            const data = await this.bookingService.getBookings(request.user.id)
            response.send(data)
        } catch(error: any){
            response.status(400).send(error.message)
        }
    }

    post = async (request: any, response: any) => {
        try{
            const booking = request.body
            booking.userId = request.user.id
            const usero = await this.userService.getById(booking.userId)
            const hotelo = await this.hotelService.getById(booking.hotelId)
            if(usero && usero.age > 17 && hotelo && hotelo.capacity > booking.visitors){
                await this.bookingService.add(booking.arrival, booking.departure,
                    booking.visitors, booking.userId, booking.hotelId)
                response.send('Successfully added booking')
            } else {
                response.status(400).send('Out of space or you are too young')
            }
        } catch(error: any){
            response.status(400).send(error.message)
        }
    }
}

export default BookingController
```

6. Controller hotel



```ts
import HotelService from "../../services/hotel"

class HotelController {
    private service = new HotelService()

    get = async (request: any, response: any) => {
        try{
            if(request.query.town) {
                const data = await this.service.getWithParameters(request.query.town, request.query.type)
                response.send(data)
            } else {
                const data = await this.service.getAll()
                response.send(data)
            }
        } catch(error: any){
            response.status(400).send(error.message)
        }
    }

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

7. Controller user

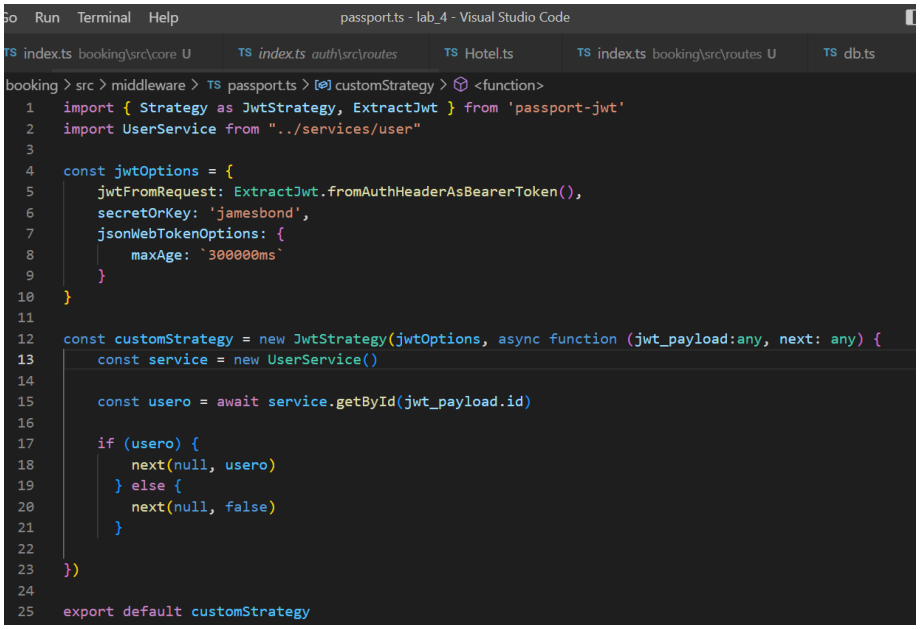
```ts
import UserService from '../../services/user/index'

class UserController {
    private service = new UserService()

    get = async (request: any, response: any) => {
        try{
            if(request.query.email) {
                console.log(`Searching user ${request.query.email}`)
                const data = await this.service.getByEmail(request.query.email)
                response.send(data)
            } else {
                const data = await this.service.getAll()
                response.send(data)
            }
        } catch(error: any){
            response.status(400).send(error.message)
        }
    }

    post = async (request: any, response: any) => {
        try{
            const user = request.body
            await this.service.add(user.name, user.surname, user.email, user.password, user.age)
            response.send('Successfully added user')
        } catch(error: any){
            response.status(400).send(error.message)
        }
    }
}
```

8. Core booking

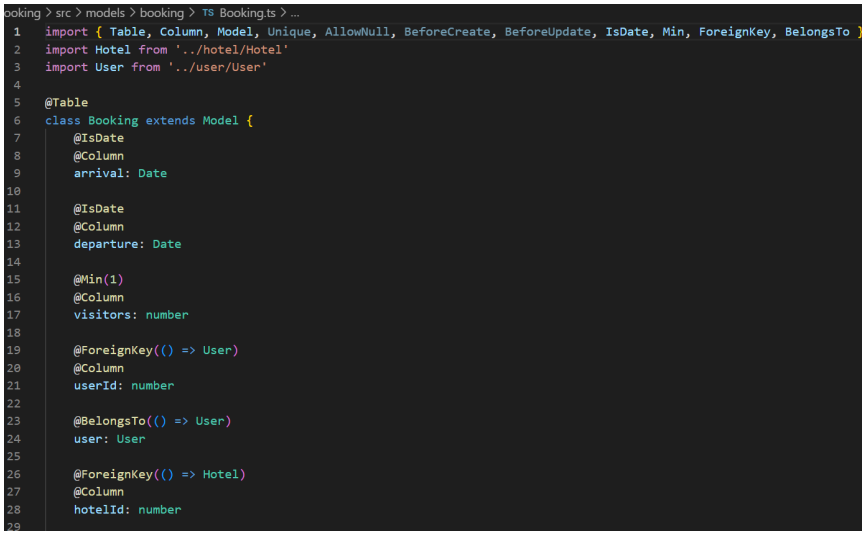
booking > src > core > TS index.ts > App > createApp

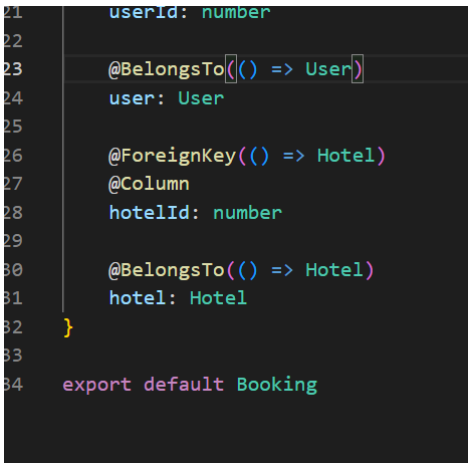
```ts
import express from "express"
import { createServer, Server } from "http"
import routes from "../routes/index"
import {sequelize} from "../providers/db"
import bodyParser from "body-parser"
import customStrategy from "../middleware/passport"
import passport from "passport"

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()
        passport.use(customStrategy)
        app.use(bodyParser.urlencoded({extended: false}))
        app.use(bodyParser.json())
        app.use(passport.initialize())
        app.use('/v1', routes)

        return app
    }

    private createServer(): Server {
        const server = createServer(this.app)

        return server
    }

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

export default App
```

9. Middleware passport



10. Model Booking





11. Model hotel

booking > src > models > hotel > TS Hotel.ts > Hotel > capacity

```ts
import { Table, Column, Model, Unique, AllowNull, BeforeCreate, BeforeUpdate, HasMany, Min, Max } from 'sequelize-typ
import Booking from '../booking/Booking'

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

    @Column
    town: string

    @Min(0)
    @Max(1000)
    @Column
    capacity: number

    @Column
    type: string

    @HasMany(() => Booking)
    booking: Booking[]
}

export default Hotel
```

12. Model User

booking > src > models > user > TS User.ts > ...

```ts
import { Table, Column, Model, Unique, AllowNull, BeforeCreate, BeforeUpdate, Min, HasMany } from 'sequelize-typesc
import Booking from '../booking/Booking'

@Table
class User extends Model {
    @Column
    name: string

    @Column
    surname: string

    @Column
    email: string

    @Column
    password: string

    @Min(0)
    @Column
    age: number

    @HasMany(() => Booking)
    booking: Booking[]
}

export default User
```

13. Provider DB

booking > src > providers > TS db.ts > ...

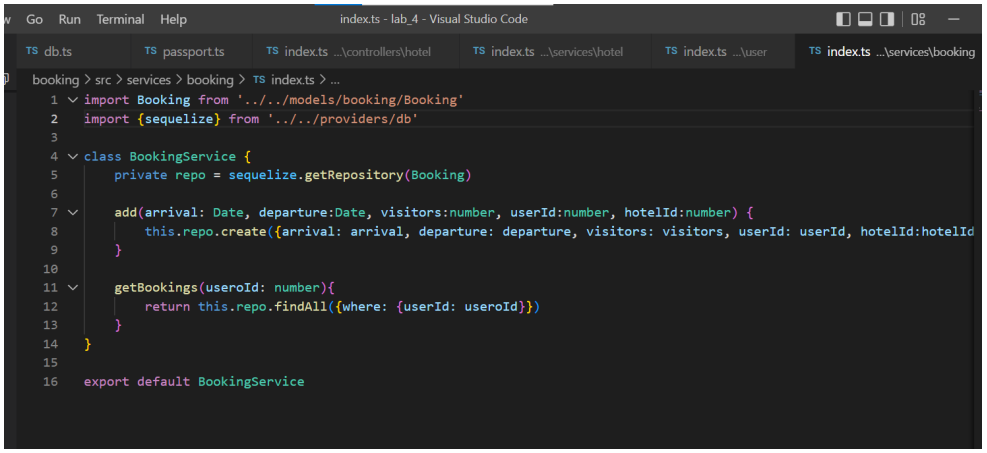
```ts
import { Sequelize } from 'sequelize-typescript'
import User from '../models/user/User'
import Booking from '../models/booking/Booking'
import Hotel from '../models/hotel/Hotel'

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

14. Router booking

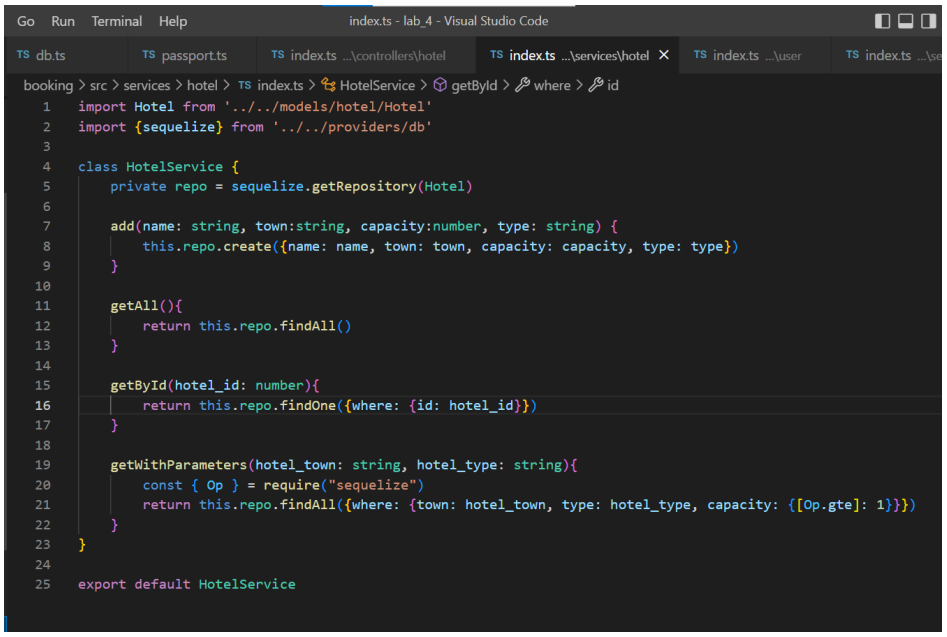
booking > src > routes > TS index.ts > router

```ts
import express from "express"
import UserController from "../controllers/user"
import HotelController from "../controllers/hotel"
import BookingController from "../controllers/booking"
import passport from "passport"

const router: express.Router = express.Router()

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('/bookings')
    .get(passport.authenticate('jwt', { session: false }), bookingController.get)
    .post(passport.authenticate('jwt', { session: false }), bookingController.post)

export default router
```

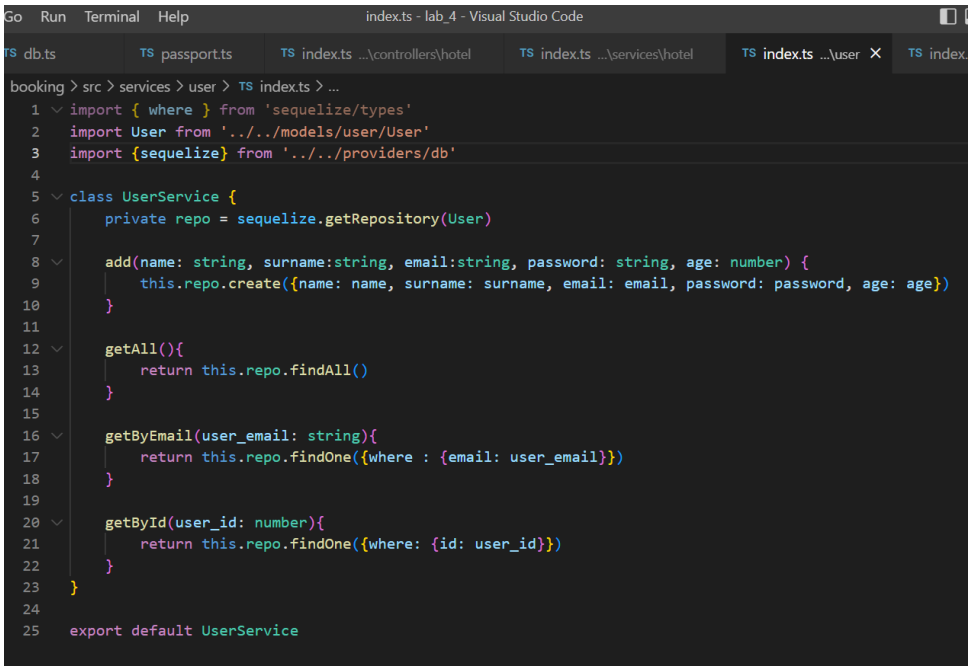
15. Services Booking



16. Services Hotel



17. Services User



Вывод

В ходе работы я реализовал сервис для поиска отелей с помощью typescript с отдельным микросервисом под авторизацию