

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

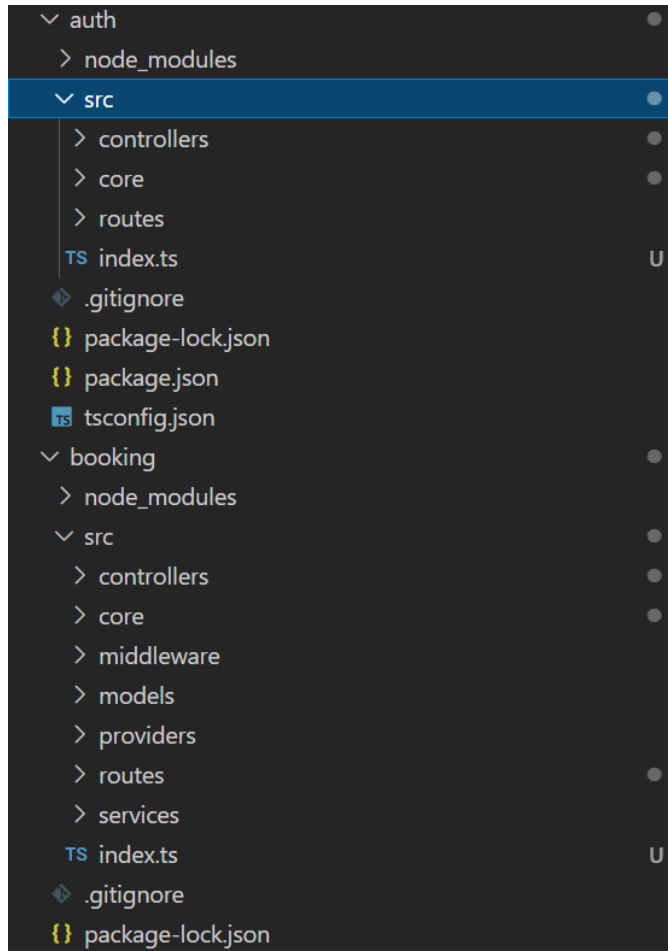
Санкт-Петербург 2022 г.

Цель работы:

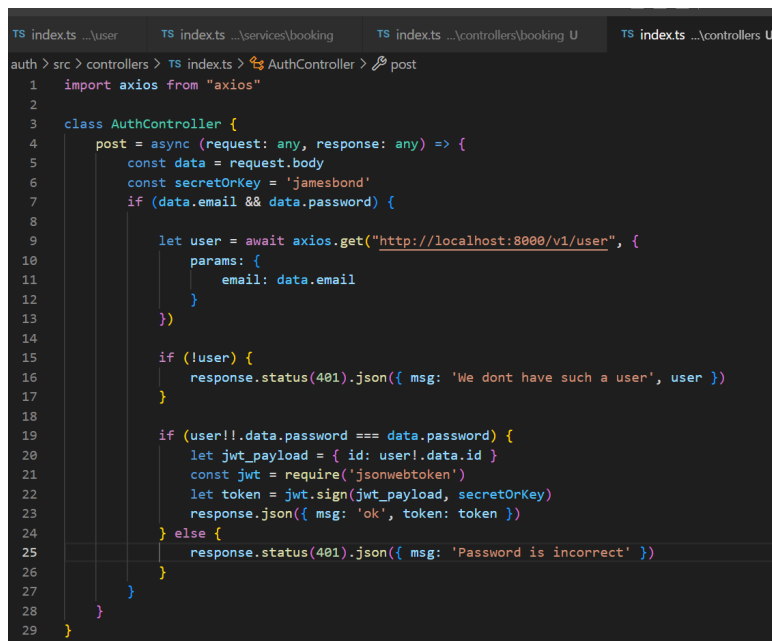
- Платформа для поиска и бронирования номера в отеле/квартире/хостеле (<https://airbnb.com>)
- Вход
- Регистрация
- Страница бронирований пользователя
- Страница для поиска номера с возможностью выбора города, времени заселения, количеству гостей
- Необходимо реализовать отдельный микросервис, выполняющий 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
dexts ...services\booking TS index.ts ...controllers\booking U TS index.ts ...controllers
auth > src > core > TS index.ts > App > createApp
1 import express from "express"
2 import { createServer, Server } from "http"
3 import routes from "../routes/index"
4 import bodyParser from "body-parser"
5
6 class App {
7   public port: number
8   public host: string
9
10  private app: express.Application
11  private server: Server
12
13  constructor(port = 8100, host = "localhost") {
14    this.port = port
15    this.host = host
16
17    this.app = this.createApp()
18    this.server = this.createServer()
19  }
20
21  private createApp(): express.Application {
22    const app = express()
23    app.use(bodyParser.urlencoded({extended: false}))
24    app.use(bodyParser.json())
25    app.use('/v1', routes)
26
27    return app
28  }
29
30  private createServer(): Server {
31    const server = createServer(this.app)
32
33    return server
34  }
35
36  public start(): void {
37    this.server.listen(this.port, () => {
38      console.log(`Running server on port ${this.port}`)
39    })
40  }
41 }
42
43 export default App
```

4. Router auth

```
auth > src > routes > TS index.ts > ...
1 import express from "express"
2 import AuthController from "../controllers/index"
3
4 const router: express.Router = express.Router()
5
6 const authController = new AuthController()
7
8 router
9   .route('/auth')
10  .post(authController.post)
11
12 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
1 import BookingService from "../../services/booking"
2 import UserService from "../../services/user"
3 import HotelService from "../../services/hotel"
4
5 class BookingController {
6   private bookingService = new BookingService()
7   private userService = new UserService()
8   private hotelService = new HotelService()
9
10  get = async (request: any, response: any) => {
11    try{
12      const data = await this.bookingService.getBookings(request.user.id)
13      response.send(data)
14    } catch(error: any){
15      response.status(400).send(error.message)
16    }
17  }
18
19  post = async (request: any, response: any) => {
20    try{
21      const booking = request.body
22      booking.userId = request.user.id
23      const usero = await this.userService.getById(booking.userId)
24      const hotelo = await this.hotelService.getById(booking.hotelId)
25      if(usero && usero.age > 17 && hotelo && hotelo.capacity > booking.visitors){
26        await this.bookingService.add([booking.arrival, booking.departure,
27          booking.visitors, booking.userId, booking.hotelId])
28        response.send('Successfully added booking')
29      } else {
30        response.status(400).send('Out of space or you are too young')
31      }
32    } catch(error: any){
33      response.status(400).send(error.message)
34    }
35  }
36 }
37
38 export default BookingController
```

6. Controller hotel

```
TS index.ts ...controllers\hotel X TS index.ts ...services\hotel TS index.ts ...user TS index.ts ...services\booking TS index.ts ...contr
booking > src > controllers > hotel > TS index.ts > ...
1 import HotelService from "../../services/hotel"
2
3 class HotelController {
4     private service = new HotelService()
5
6     get = async (request: any, response: any) => {
7         try{
8             if(request.query.town) {
9                 const data = await this.service.getWithParameters(request.query.town, request.query.type)
10                response.send(data)
11            } else {
12                const data = await this.service.getAll()
13                response.send(data)
14            }
15        } catch(error: any){
16            response.status(400).send(error.message)
17        }
18    }
19
20    post = async (request: any, response: any) => {
21        try{
22            const hotel = request.body
23            await this.service.add(hotel.name, hotel.town, hotel.capacity, hotel.type)
24            response.send('Successfully added hotel')
25        } catch(error: any){
26            response.status(400).send(error.message)
27        }
28    }
29 }
```

7. Controller user

```
booking > src > controllers > user > TS index.ts > ...
1 import UserService from '../../services/user/index'
2
3 class UserController {
4     private service = new UserService()
5
6     get = async (request: any, response: any) => {
7         try{
8             if(request.query.email) {
9                 console.log(`Searching user ${request.query.email}`)
10                const data = await this.service.getByEmail(request.query.email)
11                response.send(data)
12            } else {
13                const data = await this.service.getAll()
14                response.send(data)
15            }
16        } catch(error: any){
17            response.status(400).send(error.message)
18        }
19    }
20
21    post = async (request: any, response: any) => {
22        try{
23            const user = request.body
24            await this.service.add(user.name, user.surname, user.email, user.password, user.age)
25            response.send('Successfully added user')
26        } catch(error: any){
27            response.status(400).send(error.message)
28        }
29    }
30 }
```

8. Core booking

```
booking > src > core > TS index.ts > App > createApp
1  import express from "express"
2  import { createServer, Server } from "http"
3  import routes from "../routes/index"
4  import { sequelize } from "../providers/db"
5  import bodyParser from "body-parser"
6  import customStrategy from "../middleware/passport"
7  import passport from "passport"
8
9  class App {
10     public port: number
11     public host: string
12
13     private app: express.Application
14     private server: Server
15
16     constructor(port = 8000, host = "localhost") {
17         this.port = port
18         this.host = host
19
20         this.app = this.createApp()
21         this.server = this.createServer()
22     }
23
24     private createApp(): express.Application {
25         const app = express()
26         passport.use(customStrategy)
27         app.use(bodyParser.urlencoded({ extended: false }))
28         app.use(bodyParser.json())
29         app.use(passport.initialize())
30         app.use('/v1', routes)
31
32         return app
33     }
34
35     private createServer(): Server {
36         const server = createServer(this.app)
37
38         return server
39     }
40
41     public start(): void {
42         sequelize.sync().then(() => {
43             console.log('Connected to Database')
44         })
45         this.server.listen(this.port, () => {
46             console.log(`Running server on port ${this.port}`)
47         })
48     }
49 }
50
51 export default App
```

9. Middleware passport

```
book > src > middleware > TS passport.ts > customStrategy > <function>
1 import { Strategy as JwtStrategy, ExtractJwt } from 'passport-jwt'
2 import UserService from "../services/user"
3
4 const jwtOptions = {
5   jwtFromRequest: ExtractJwt.fromAuthHeaderAsBearerToken(),
6   secretOrKey: 'jamesbond',
7   jsonWebTokenOptions: {
8     maxAge: '300000ms'
9   }
10 }
11
12 const customStrategy = new JwtStrategy(jwtOptions, async function (jwt_payload:any, next: any) {
13   const service = new UserService()
14
15   const usero = await service.getById(jwt_payload.id)
16
17   if (usero) {
18     next(null, usero)
19   } else {
20     next(null, false)
21   }
22 })
23
24
25 export default customStrategy
```

10. Model Booking

```
book > src > models > booking > TS Booking.ts > ...
1 import { Table, Column, Model, Unique, AllowNull, BeforeCreate, BeforeUpdate, IsDate, Min, ForeignKey, BelongsTo } from 'sequelize-typescript'
2 import Hotel from '../hotel/Hotel'
3 import User from '../user/User'
4
5 @Table
6 class Booking extends Model {
7   @IsDate
8   @Column
9   arrival: Date
10
11   @IsDate
12   @Column
13   departure: Date
14
15   @Min(1)
16   @Column
17   visitors: number
18
19   @ForeignKey(() => User)
20   @Column
21   userId: number
22
23   @BelongsTo(() => User)
24   user: User
25
26   @ForeignKey(() => Hotel)
27   @Column
28   hotelId: number
29
30   @BelongsTo(() => Hotel)
31   hotel: Hotel
32 }
33
34 export default Booking
```


11. Model hotel

```
booking > src > models > hotel > TS Hotel.ts > Hotel > capacity
1  import { Table, Column, Model, Unique, AllowNull, BeforeCreate, BeforeUpdate, HasMany, Min, Max } from 'sequelize-typescript'
2  import Booking from '../booking/Booking'
3
4  @Table
5  class Hotel extends Model {
6      @Column
7      name: string
8
9      @Column
10     town: string
11
12     @Min(0)
13     @Max(1000)
14     @Column
15     capacity: number
16
17     @Column
18     type: string
19
20     @HasMany(() => Booking)
21     booking: Booking[]
22 }
23
24 export default Hotel
```

12. Model User

```
booking > src > models > user > TS User.ts > ...
1  import { Table, Column, Model, Unique, AllowNull, BeforeCreate, BeforeUpdate, Min, HasMany } from 'sequelize-typescript'
2  import Booking from '../booking/Booking'
3
4  @Table
5  class User extends Model {
6      @Column
7      name: string
8
9      @Column
10     surname: string
11
12     @Column
13     email: string
14
15     @Column
16     password: string
17
18     @Min(0)
19     @Column
20     age: number
21
22     @HasMany(() => Booking)
23     booking: Booking[]
24 }
25
26 export default User
```

13. Provider DB

```
booking > src > providers > TS db.ts > ...
1  import { Sequelize } from 'sequelize-typescript'
2  import User from '../models/user/User'
3  import Booking from '../models/booking/Booking'
4  import Hotel from '../models/hotel/Hotel'
5
6  export const sequelize = new Sequelize({
7    database: 'some_db',
8    dialect: 'sqlite',
9    username: 'root',
10   password: '',
11   storage: ':memory:',
12   models: [User, Hotel, Booking],
13   repositoryMode: true,
14   logging: console.log,
15 })
16
```

14. Router booking

```
booking > src > routes > TS index.ts > [⌘] router
1  import express from "express"
2  import UserController from "../controllers/user"
3  import HotelController from "../controllers/hotel"
4  import BookingController from "../controllers/booking"
5  import passport from "passport"
6
7  const router: express.Router = express.Router()
8
9  const userController = new UserController()
10 const hotelController = new HotelController()
11 const bookingController = new BookingController()
12
13 router
14   .route('/user')
15   .get(userController.get)
16   .post(userController.post)
17
18
19 router
20   .route('/hotel')
21   .get(hotelController.get)
22   .post(hotelController.post)
23
24
25 router
26   .route('/bookings')
27   .get(passport.authenticate('jwt', { session: false }), bookingController.get)
28   .post(passport.authenticate('jwt', { session: false }), bookingController.post)
29
30 export default router
```

15. Services Booking

```
Go Run Terminal Help index.ts - lab_4 - Visual Studio Code
TS db.ts TS passport.ts TS index.ts ...controllers\hotel TS index.ts ...services\hotel TS index.ts ...user TS index.ts ...services\booking

booking > src > services > booking > TS index.ts > ...
1 import Booking from '../../models/booking/Booking'
2 import {sequelize} from '../../providers/db'
3
4 class BookingService {
5     private repo = sequelize.getRepository(Booking)
6
7     add(arrival: Date, departure: Date, visitors: number, userId: number, hotelId: number) {
8         this.repo.create({arrival: arrival, departure: departure, visitors: visitors, userId: userId, hotelId: hotelId})
9     }
10
11     getBookings(userId: number){
12         return this.repo.findAll({where: {userId: userId}})
13     }
14 }
15
16 export default BookingService
```

16. Services Hotel

```
Go Run Terminal Help index.ts - lab_4 - Visual Studio Code
TS db.ts TS passport.ts TS index.ts ...controllers\hotel TS index.ts ...services\hotel X TS index.ts ...user TS index.ts ...services\booking

booking > src > services > hotel > TS index.ts > HotelService > getById > where > id
1 import Hotel from '../../models/hotel/Hotel'
2 import {sequelize} from '../../providers/db'
3
4 class HotelService {
5     private repo = sequelize.getRepository(Hotel)
6
7     add(name: string, town: string, capacity: number, type: string) {
8         this.repo.create({name: name, town: town, capacity: capacity, type: type})
9     }
10
11     getAll(){
12         return this.repo.findAll()
13     }
14
15     getById(hotel_id: number){
16         return this.repo.findOne({where: {id: hotel_id}})
17     }
18
19     getWithParameters(hotel_town: string, hotel_type: string){
20         const { Op } = require("sequelize")
21         return this.repo.findAll({where: {town: hotel_town, type: hotel_type, capacity: {[Op.gte]: 1}}})
22     }
23 }
24
25 export default HotelService
```

17. Services User

```
Go Run Terminal Help index.ts - lab_4 - Visual Studio Code
TS db.ts TS passport.ts TS index.ts ...controllers\hotel TS index.ts ...services\hotel TS index.ts ...user X TS index.ts ...services\booking

booking > src > services > user > TS index.ts > ...
1 import { where } from 'sequelize/types'
2 import User from '../../models/user/User'
3 import {sequelize} from '../../providers/db'
4
5 class UserService {
6     private repo = sequelize.getRepository(User)
7
8     add(name: string, surname: string, email: string, password: string, age: number) {
9         this.repo.create({name: name, surname: surname, email: email, password: password, age: age})
10     }
11
12     getAll(){
13         return this.repo.findAll()
14     }
15
16     getByEmail(user_email: string){
17         return this.repo.findOne({where: {email: user_email}})
18     }
19
20     getById(user_id: number){
21         return this.repo.findOne({where: {id: user_id}})
22     }
23 }
24
25 export default UserService
```

Вывод

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