



УНИВЕРСИТЕТ ИТМО

**Санкт-петербургский национальный исследовательский университет  
информационных технологий, механики и оптики**

**Факультет инфокоммуникационных технологий**

**Лабораторная работа 3**  
**По дисциплине Бекенд-разработка**

Выполнил: Каратецкая М.Ю.

Группа № К33402

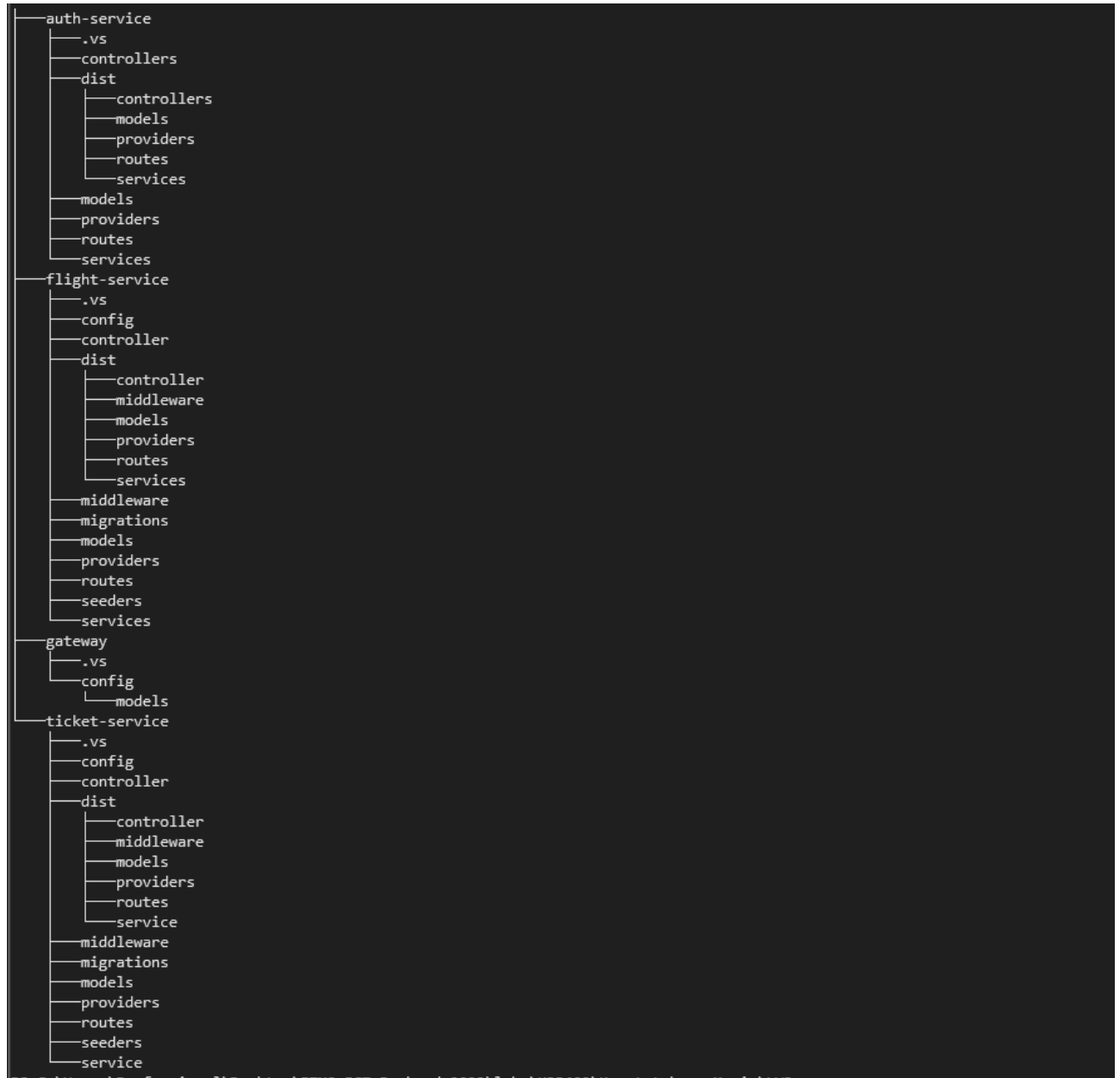
Проверил: Добряков Д.И.

Санкт-Петербург  
2022

**Цель:** Необходимо реализовать отдельный микросервис, выполняющий какую-либо содержательную функцию из всего арсенала функций вашего приложения.

## Ход работы:

### 1. структура



### 2. Ход работы

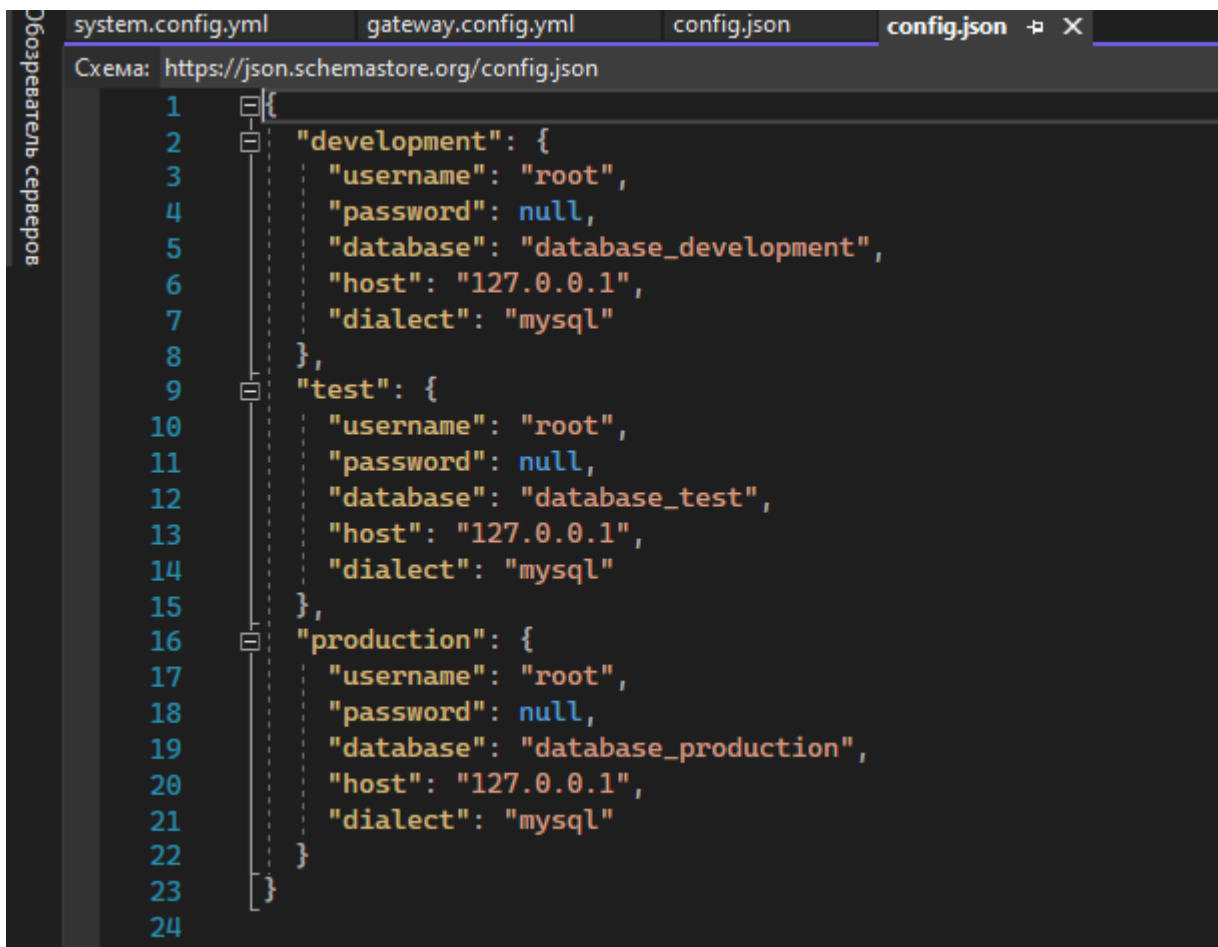
Я решила разделить все приложение на 3 микросервиса - Авторизация, сервис полетов, сервис билетов. Как видно структура у микросервисов схожая, за исключением того, что у авторизации нет config.json.

Эти 3 микросервиса находятся на 3 разных портах - <http://localhost:3000> – авторизация, <http://localhost:3001> – сервис полетов, и <http://localhost:3000> для сервиса билетов.

### 3. Сервис полетов.

На примере сервиса полетов разберем структуру.

#### Config.json



```
1  {
2    "development": {
3      "username": "root",
4      "password": null,
5      "database": "database_development",
6      "host": "127.0.0.1",
7      "dialect": "mysql"
8    },
9    "test": {
10     "username": "root",
11     "password": null,
12     "database": "database_test",
13     "host": "127.0.0.1",
14     "dialect": "mysql"
15   },
16   "production": {
17     "username": "root",
18     "password": null,
19     "database": "database_production",
20     "host": "127.0.0.1",
21     "dialect": "mysql"
22   }
23 }
24
```

#### Контроллер

```

1  import FlightService from "../services/Flight"
2  import Flight from "../models/Flight";
3
4  class FlightController {
5      private flightService: FlightService = new FlightService();
6
7      get = async (request: any, response: any) => {
8          const id = request.params.id
9          try {
10             const flight: Flight = await this.flightService.get(id)
11             response.send(flight)
12         }
13         catch (error: any) {
14             response.status(404)
15         }
16     }
17
18     filter = async (request: any, response: any) => {
19         const body = request.body
20         try {
21             const flights: Flight[] = await this.flightService.filter(body)
22             response.send(flights)
23         }
24         catch (error: any) {
25             response.status(400)
26         }
27     }
28
29     create = async (request: any, response: any) => {
30         try {
31             const flight: Flight = await this.flightService.create(request.body)
32             response.send(flight)
33         }
34         catch (error: any) {
35             response.status(400).send()
36         }
37     }
38 }
39
40 export default FlightController

```

Middleware - Auth.ts

```
s\Professional\Desktop\ITMO-ICT-Backend-2022\src\data  🔑 token"
import axios from "axios"

async function auth(req: any, res: any, next: any) {
  const token: string = req.headers["authorization"].replace("Bearer ", "")
  axios.get("http://localhost:3000/auth/check", {
    data: {
      "token": token
    }
  }).then(response => {
    if (response.status == 200) {
      next()
    }
    else {
      res.status(401).send("")
    }
  }).catch(error => {
    res.status(401).send("")
  })
}

export default auth
```

## Models

```
Professional\Desktop\ITMO-ICT-Backend-2022\src\models\flight  🔑 Flight  🔑 id
import { AllowNull, AutoIncrement, Column, DataType, Model, PrimaryKey, Table, Unique } from "sequelize-typescript"

@Table
class Flight extends Model {
  @AutoIncrement
  @PrimaryKey
  @Column({ type: DataType.INTEGER })
  id: number

  @Unique
  @AllowNull(false)
  @Column({ type: DataType.STRING })
  code: string

  @Column({ type: DataType.INTEGER })
  ticketsTotal: number

  @AllowNull(false)
  @Column({ type: DataType.STRING })
  departureFrom: string

  @AllowNull(false)
  @Column({ type: DataType.STRING })
  arrivalTo: string

  @AllowNull(false)
  @Column({ type: DataType.DATE })
  departure: Date

  @AllowNull(false)
  @Column({ type: DataType.DATE })
  arrival: Date

  @AllowNull(false)
  @Column({ type: DataType.STRING })
  planeCode: string
}
```

Providers – db.ts

```
Flight.ts Flight.ts Auth.ts Flight.ts system.config.yml gateway.config.yml config.json config.json
C:\Users\Professional\Desktop\ITMO-ICT-Backend-2021\src\services\flight.ts {} "db" Sequelize
1 import { Sequelize } from "sequelize-typescript";
2
3 import Flight from "../models/Flight";
4
5 const sequelize = new Sequelize({
6   database: 'flight',
7   dialect: 'sqlite',
8   storage: 'flight.sqlite'
9 })
10 sequelize.addModels([Flight,])
11
12 sequelize.sync().catch(error => console.log(error))
13
14
15 async function testDb() {
16   try {
17     await sequelize.authenticate()
18     console.log('Successfully connected')
19   }
20   catch (error) {
21     console.log('Error during connection to database')
22     console.log(error)
23   }
24 }
25
26 testDb()
27
28 export default sequelize
```

Routers – flight.ts

```
flight.ts db.ts Flight.ts Flight.ts Auth.ts Flight.ts system.config.yml gateway.config.yml config.json
C:\Users\Professional\Desktop\ITMO-ICT-Backend-2021\src\services\flight.ts {} "Flight" default
2 import FlightController from '../controller/Flight';
3
4 import auth from '../middleware/Auth';
5
6 const router: express.Router = express.Router()
7 const controller: FlightController = new FlightController()
8
9 router.route("/flights/:id").get(auth, controller.get)
10 router.route("/flights").get(auth, controller.filter)
11 router.route("/flights").post(auth, controller.create)
12
13 export default router
```

Services flight.ts

```
db.ts  Flight.ts  Flight.ts  Auth.ts  Flight.ts  system.config.yml  gateway.config.yml  config.json
Professional\Desktop\TMO-ICT-Backend-2021  filter  arrivalTime

import Flight from "../models/Flight"
import { Sequelize, Op } from "sequelize";

class FlightService {

  async get(id: number) {
    const flight: Flight | null = await Flight.findByPk(id);
    if (flight == null) {
      throw new Error
    }
    return flight;
  }

  async filter(fo: any) {
    let where: any = {}

    if (fo.departureTimeFrom || fo.departureTimeTo) {
      let departureTime: any;

      if (fo.departureTimeFrom && fo.departureTimeTo) {
        departureTime = {
          [Op.gte]: fo.departureTimeFrom,
          [Op.lte]: fo.departureTimeTo
        }
      }
      else if (fo.departureTimeFrom) {
        departureTime = { [Op.gte]: fo.departureTimeFrom }
      }
      else if (fo.departureTimeTo) {
        departureTime = { [Op.lte]: fo.departureTimeTo }
      }
      where.departure = departureTime
    }
  }
}
```

Index.ts

```
package.json  nodemon.json  index.ts  Flight.ts  db.ts  Flight.ts  Flight.ts  Auth.ts  Flight.ts
C:\Users\Professional\Desktop\TMO-ICT-Backend-2021  {} "index"  server.listen() callback

4  import { createServer } from 'http';
5
6  import sequelize from '../providers/db';
7  import router from './routes/Flight';
8  import auth from './middleware/Auth';
9
10 var app = express();
11 var _sequelize = sequelize;
12
13 app.use(express.json())
14 app.use(cookieParser())
15 // app.use(auth)
16 app.use(router)
17
18
19 const server = createServer(app)
20 const port = 3001
21 server.listen(port, () => console.log('Running on port ${port}'))

105 %  2  0  Стр: 21  Симв: 66  Пробелы  CTRL
PowerShell д...азработчиков  PowerShell для разработчиков
```

## Вывод:

В ходе лабораторной работы весь проект для авиабилетов был разделен на микросервиса – сервис полетов, сервис билетов и авторизация.