

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №3

Выполнил:

Егоров Мичил

Группа

К33401

Проверил:

Добряков Д. И.

Санкт-Петербург

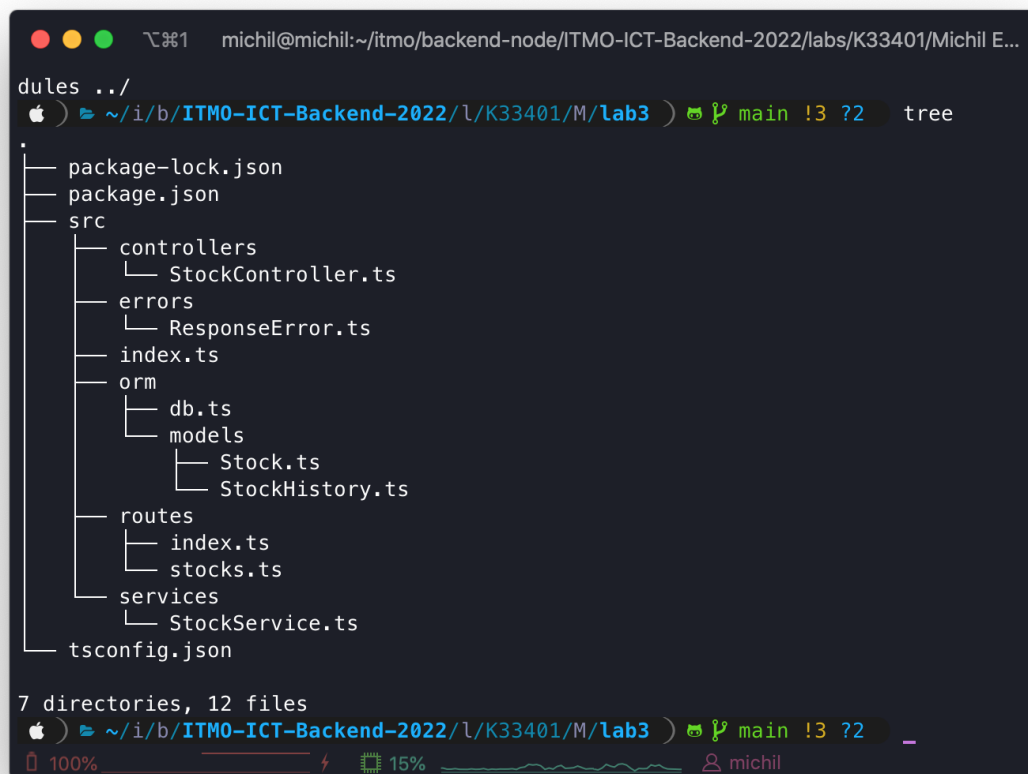
2022 г.

Задача

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

Ход работы

1. Структура приложения



```
michil@michil:~/itmo/backend-node/ITMO-ICT-Backend-2022/labs/K33401/Michil E...
dules ../
~/i/b/ITMO-ICT-Backend-2022/l/K33401/M/lab3 ) main !3 ?2 tree
├─ package-lock.json
├─ package.json
├─ src
│   ├── controllers
│   │   └─ StockController.ts
│   ├── errors
│   │   └─ ResponseError.ts
│   ├── index.ts
│   ├── orm
│   │   ├── db.ts
│   │   └─ models
│   │       ├── Stock.ts
│   │       └─ StockHistory.ts
│   ├── routes
│   │   ├── index.ts
│   │   └─ stocks.ts
│   └─ services
│       └─ StockService.ts
└─ tsconfig.json

7 directories, 12 files
~/i/b/ITMO-ICT-Backend-2022/l/K33401/M/lab3 ) main !3 ?2 _
100% 15% michil
```

2. Модели

```
src > orm > models > Users > User
1 import {Column, Entity, OneToMany, OneToOne, PrimaryGeneratedColumn} from "typeorm";
2 import {Portfolio} from "../Portfolio";
3 import {AuthToken} from "../Token";
4
5 @Entity('User')
6 export class User {
7     @PrimaryGeneratedColumn()
8     id: number;
9
10    @Column({ unique: true })
11    email: string;
12
13    @Column()
14    password: string;
15
16    @OneToMany(() => Portfolio, (portfolio) => portfolio.user, {onDelete: "CASCADE"})
17    portfolios: Portfolio[];
18
19    @OneToOne(() => AuthToken, (token) => token.user, {onDelete: "CASCADE"})
20    authToken: AuthToken
21 }
```

```
src > orm > models > Tokens > AuthToken
1 import {Column, Entity, OneToOne, JoinColumn, PrimaryGeneratedColumn} from "typeorm";
2 import {User} from "../User";
3
4 @Entity('authToken')
5 export class AuthToken {
6     @PrimaryGeneratedColumn()
7     id: number;
8
9     @Column({ type: 'int'})
10    userId: number
11
12    @OneToOne(() => User, (user) => user.authToken, {onDelete: "CASCADE"})
13    @JoinColumn({ name: "userId" })
14    user: User
15
16    @Column()
17    token: string
18 }
```

```
src > orm > models > Stock.ts > Stock
3 import {PortfolioStock} from "../PortfolioStock";
4 import {Portfolio} from "../Portfolio";
5
6 @Entity('stock')
7 export class Stock {
8   @PrimaryGeneratedColumn()
9   id: number;
10
11   @Column()
12   name: string;
13
14   @Column()
15   description: string;
16
17   @Column({type: 'timestamp'})
18   created_at: Date;
19
20   @Column()
21   lastPrice: number;
22
23   @OneToMany(() => StockHistory, (stockHistory) => stockHistory.stock, {onDelete: "CASCADE"})
24   history: StockHistory[];
25
26   @OneToMany(() => PortfolioStock, (ps) => ps.stock, {onDelete: "CASCADE"})
27   portfolios: PortfolioStock[];
28 }
```

```
src > orm > models > StockHistory.ts > StockHistory > price
1 import {Column, Entity, JoinColumn, ManyToOne, PrimaryGeneratedColumn} from "typeorm";
2 import {Stock} from "../Stock";
3
4 @Entity('stockHistory')
5 export class StockHistory {
6   @PrimaryGeneratedColumn()
7   id: number;
8
9   @Column({ type: 'int' })
10   stock_id: number;
11
12   @ManyToOne(() => Stock, (stock) => stock.history, {onDelete: "CASCADE"})
13   @JoinColumn({ name: 'stock_id' })
14   stock: Stock;
15
16   @Column()
17   price: number;
18
19   @Column({type: 'timestamp'})
20   timestamp: Date;
21 }
```

```
src > orm > models > Portfolio.ts > Portfolio > stocks
1 import {Column, Entity, JoinColumn, ManyToOne, OneToMany, PrimaryGeneratedColumn} from "typeorm";
2 import {User} from "../User";
3 import {PortfolioStock} from "../PortfolioStock";
4 import {Stock} from "../Stock";
5
6 @Entity('portfolio')
7 export class Portfolio {
8     @PrimaryGeneratedColumn()
9     id: number;
10
11     @Column()
12     name: string
13
14     @Column({type: 'int'})
15     user_id: number
16
17     @Column({default: 0})
18     balance: number
19
20     @ManyToOne(() => User, (user) => user.portfolios, {onDelete: "CASCADE"})
21     @JoinColumn({name: 'user_id'})
22     user: User;
23
24     @OneToMany(() => PortfolioStock, (ps) => ps.portfolio, {onDelete: "CASCADE"})
25     stocks: PortfolioStock[];
26 }
```

```
src > orm > models > PortfolioStock.ts > PortfolioStock
1 import {Column, Entity, JoinColumn, ManyToOne, PrimaryGeneratedColumn} from "typeorm";
2 import {Portfolio} from "../Portfolio";
3 import {Stock} from "../Stock";
4
5 @Entity('PortfolioStock')
6 export class PortfolioStock {
7     @PrimaryGeneratedColumn()
8     id: number;
9
10     @Column({ type: 'int' })
11     stock_id: number
12
13     @ManyToOne(() => Stock, (stock) => stock.portfolios, {onDelete: "CASCADE"})
14     @JoinColumn({name: 'stock_id'})
15     stock: Stock;
16
17     @Column({type: 'timestamp'})
18     timestamp: Date;
19
20     @Column()
21     price: number;
22
23     @Column({default: 0})
24     amount: number;
25 }
26
27
28
29
30
31
32
```

3. Контроллеры

```
src > controllers > StockController.ts > updatePrice > stock
1  import StockService from '../services/StockService'
2
3
4  class StockController {
5      private stockService: StockService
6
7      constructor() {
8          this.stockService = new StockService()
9      }
10
11      get = async (request: any, response: any) => {
12          try {
13              const stock = await this.stockService.getById(
14                  +request.params.id
15              )
16              response.send(stock)
17          } catch (error: any) {
18              response.status(404).send({
19                  "error": error.message
20              })
21          }
22      }
23
24      list = async (request: any, response: any) => {
25          try {
26              const stocks = await this.stockService.list(
27                  new Date(request.params.at_least | 0)
28              )
29              response.send({stocks: stocks})
30          } catch (error: any) {
31              response.status(404).send({
32                  "error": error.message
33              })
34          }
35      }
36
37      create = async (request: any, response: any) => {
38          try {
39              const stock = await this.stockService.create(
```

```
src > controllers > StockController.ts > updatePrice > stock
37      create = async (request: any, response: any) => {
38          try {
39              const stock = await this.stockService.create(
40                  request.body
41              )
42              response.send(stock)
43          } catch (error: any) {
44              response.status(404).send({
45                  "error": error.message
46              })
47          }
48      }
49
50      getStockHistory = async (request: any, response: any) => {
51          try {
52              const stocks = await this.stockService.getStockHistory(
53                  +request.params.id
54              )
55              response.send({stocks: stocks})
56          } catch (error: any) {
57              response.status(404).send({
58                  "error": error.message
59              })
60          }
61      }
62
63      updatePrice = async (request: any, response: any) => {
64          try {
65              const stock = await this.stockService.updatePrice(
66                  request.body
67              )
68              response.send(stock)
69          } catch (error: any) {
70              response.status(404).send({
71                  "error": error.message
72              })
73          }
74      }
75  }
```

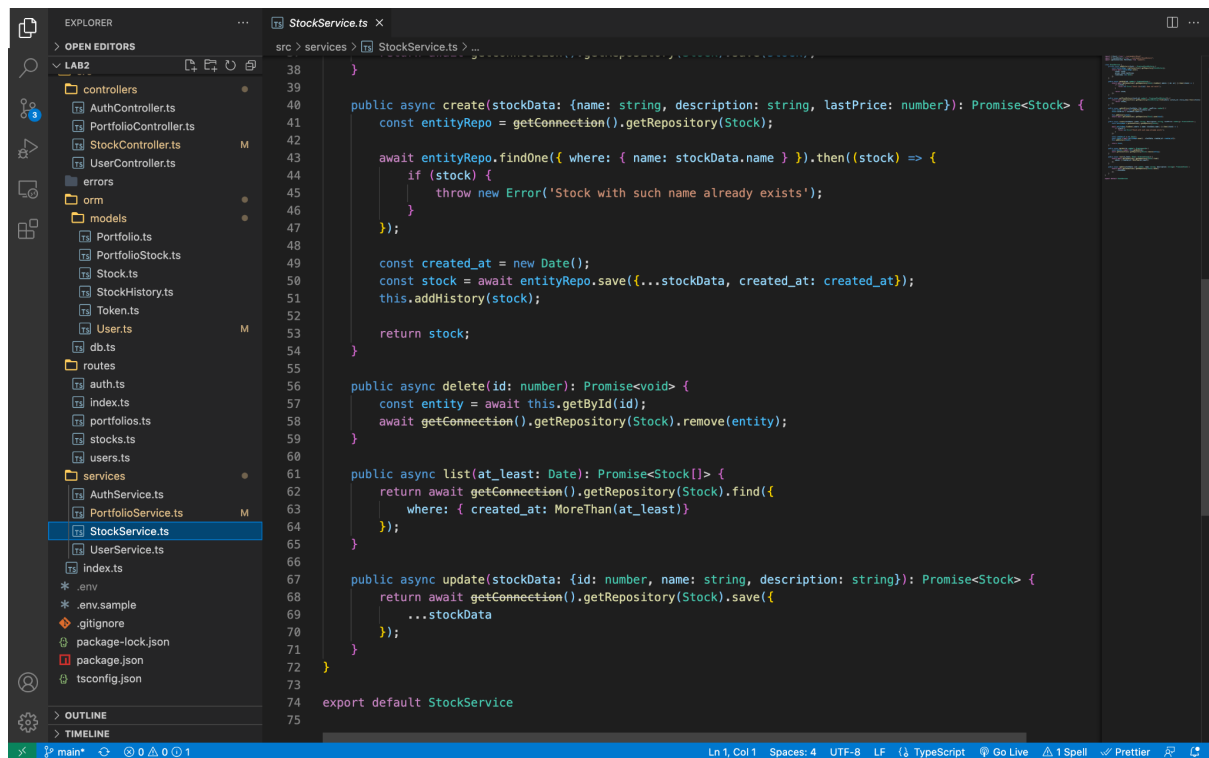
```
src > controllers > StockController.ts > updatePrice > stock
67
68     response.send(stock)
69   } catch (error: any) {
70     response.status(404).send({
71       "error": error.message
72     })
73   }
74 }
75
76 delete = async (request: any, response: any) => {
77   try {
78     const stock = await this.stockService.delete(+request.params.id)
79     response.send({status: 'ok'});
80   } catch (error: any) {
81     response.status(404).send({
82       "error": error.message
83     })
84   }
85 }
86
87 update = async (request: any, response: any) => {
88   try {
89     const stock = await this.stockService.update({
90       id: +request.params.id,
91       ...request.body,
92     })
93     response.send(stock);
94   } catch (error: any) {
95     response.status(404).send({
96       "error": error.message
97     })
98   }
99 }
100 }
101
102 export default StockController
103
```

4. Routes

```
src > routes > index.ts > ...
1 import express from "express"
2 import stockRoutes from "../stocks"
3
4
5 const router = express.Router()
6
7 router.use(["/stocks", stockRoutes])
8
9 export default router

src > routes > stocks.ts > ...
1 import express from "express"
2 import StockController from "../../controllers/StockC
3
4 const router = express.Router()
5
6 const controller = new StockController()
7
8 router.route("/:id")
9   .get(controller.get)
10
11 router.route("/")
12   .get(controller.list)
13
14 router.route("/history/:id")
15   .get(controller.getStockHistory)
16
17 router.route("/update_price")
18   .post(controller.updatePrice)
19
20 router.route("/")
21   .post(controller.create)
22
23 router.route("/:id")
24   .delete(controller.delete)
25
26
27 export default router
28
```

5. Сервисы



Вывод

В ходе данной лабораторной работы мы реализовали микросервис, который отвечает за функционал ценных валют.