

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 №1
по теме: Typescript основы
по дисциплине: Бэк-энд разработка

Специальность:
09.03.03 Мобильные и сетевые технологии

Проверил:
Добряков Д.И. _____
Дата: «25» марта 2022г.
Оценка _____

Выполнил:
студент _____ группы
К33401
Фоменко Иван

Санкт-Петербург 2022 г.

Цель работы:

Написать свой boilerplate на express + sequelize / TypeORM + typescript

Должно быть явное разделение на:

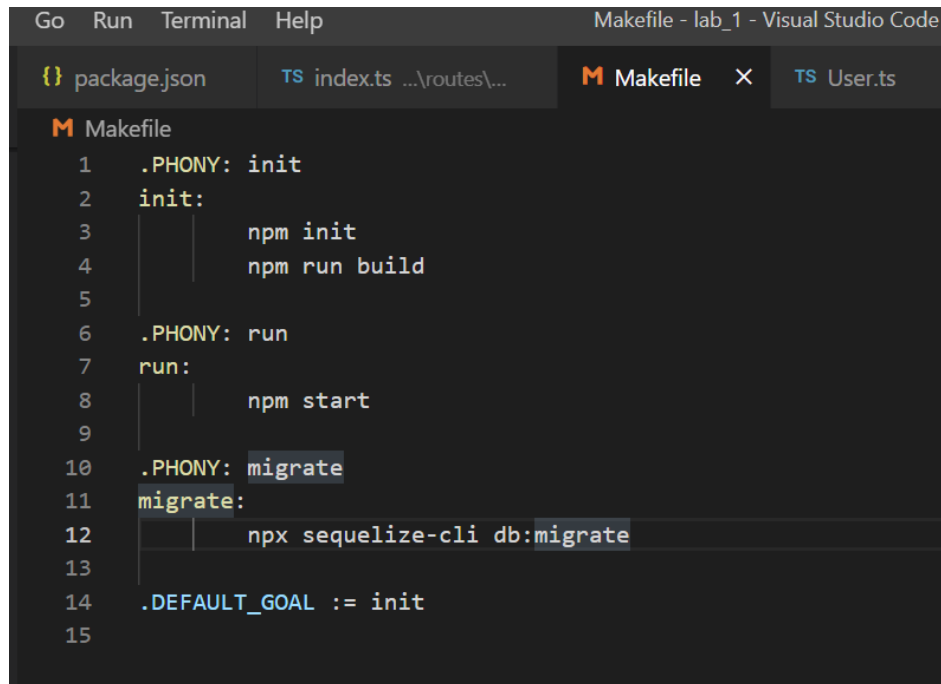
- модели
- контроллеры
- роуты
- сервисы для работы с моделями (реализуем паттерн “репозиторий”)

Составить Makefile, который будет автоматизировать рутинные действия, такие как:

- проведение миграций через sequelize;
- запуск приложения;
- установка зависимостей и сборка приложения.

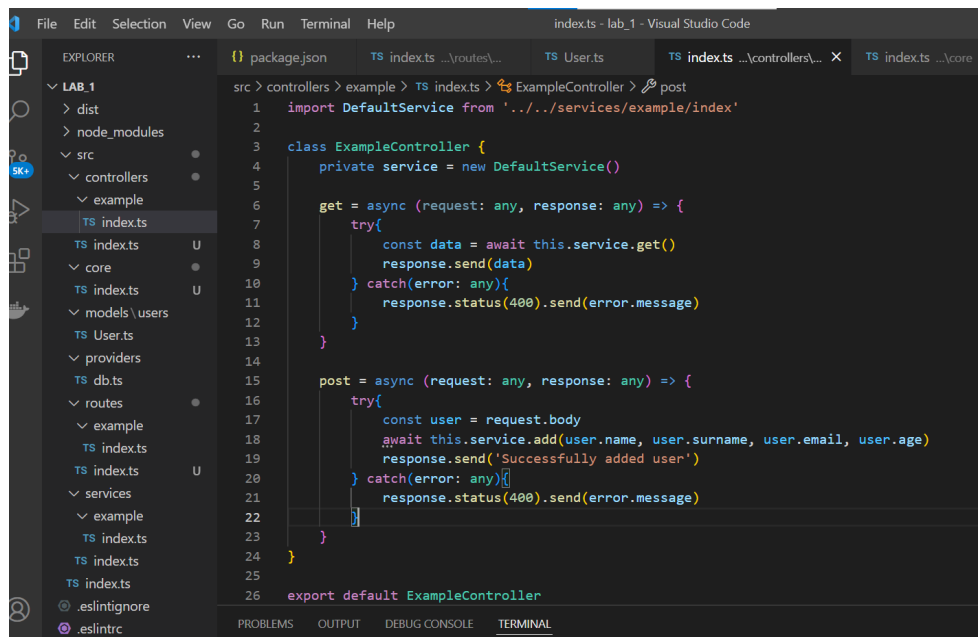
Ход работы

1. Makefile



```
1  .PHONY: init
2  init:
3      npm init
4      npm run build
5
6  .PHONY: run
7  run:
8      npm start
9
10 .PHONY: migrate
11 migrate:
12     npx sequelize-cli db:migrate
13
14 .DEFAULT_GOAL := init
15
```

2. Controller



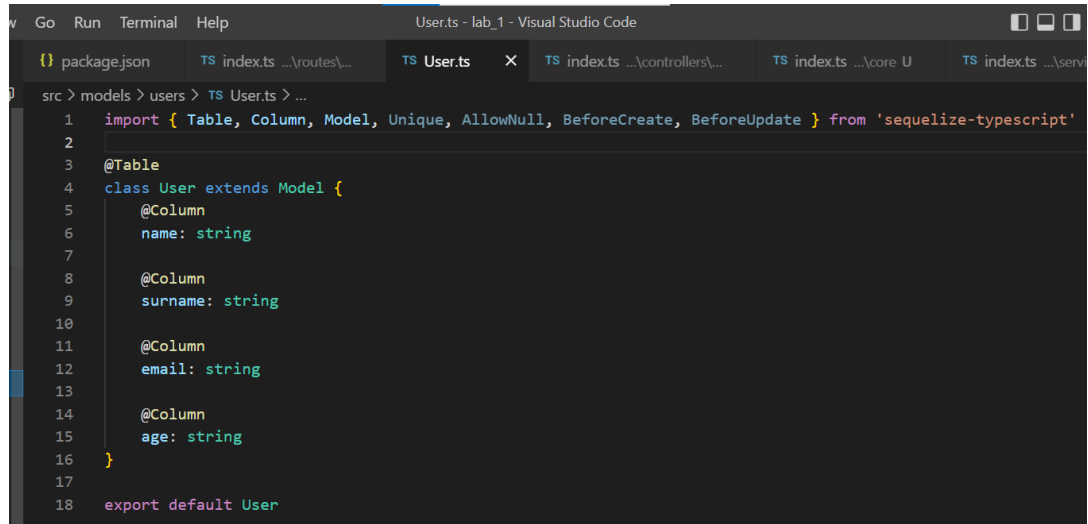
```
1  import DefaultService from '../services/example/index'
2
3  class ExampleController {
4      private service = new DefaultService()
5
6      get = async (request: any, response: any) => {
7          try{
8              const data = await this.service.get()
9              response.send(data)
10             } catch(error: any){
11                 response.status(400).send(error.message)
12             }
13         }
14
15         post = async (request: any, response: any) => {
16             try{
17                 const user = request.body
18                 await this.service.add(user.name, user.surname, user.email, user.age)
19                 response.send('Successfully added user')
20             } catch(error: any){
21                 response.status(400).send(error.message)
22             }
23         }
24     }
25
26     export default ExampleController
```

3. Core

```
package.json x TS index.ts ...\routes\... TS User.ts TS index.ts ...\controllers\... TS index.ts ...\core U x
src > core > TS index.ts > App > createApp
1 import express from "express"
2 import { createServer, Server } from "http"
3 import routes from "../routes/example/index"
4 import sequelize from "../providers/db"
5 import bodyParser from "body-parser"
6
7 class App {
8   public port: number
9   public host: string
10
11   private app: express.Application
12   private server: Server
13
14   constructor(port = 8000, host = "localhost") {
15     this.port = port
16     this.host = host
17
18     this.app = this.createApp()
19     this.server = this.createServer()
20   }
21
22   private createApp(): express.Application {
23     const app = express()
24     app.use(bodyParser.urlencoded({extended: false}))
25     app.use(bodyParser.json())
26     app.use('/v1', routes)
```

```
src > core > TS index.ts > App > createApp
22   private createApp(): express.Application {
23     const app = express()
24     app.use(bodyParser.urlencoded({extended: false}))
25     app.use(bodyParser.json())
26     app.use('/v1', routes)
27
28     return app
29   }
30
31   private createServer(): Server {
32     const server = createServer(this.app)
33
34     return server
35   }
36
37   public start(): void {
38     sequelize.sync().then(()=>{
39       console.log('Connected to Database')
40     })
41     this.server.listen(this.port, () => {
42       console.log(`Running server on port ${this.port}`)
43     })
44   }
45 }
46
47 export default App
```

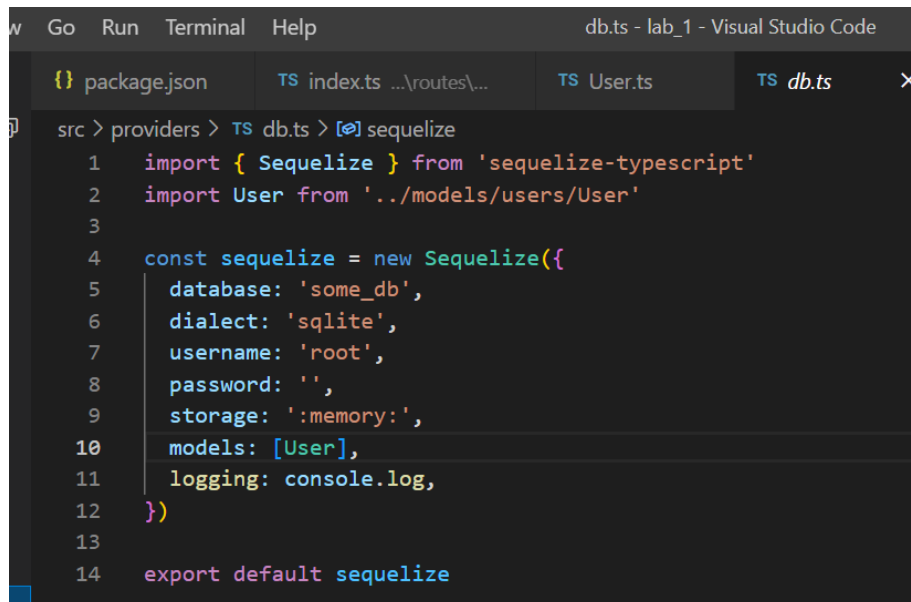
4. User model



A screenshot of the Visual Studio Code editor showing the file `Users.ts` in the `src > models > users` directory. The code defines a `User` model using `sequelize-typescript`. It imports `Table`, `Column`, `Model`, `Unique`, `AllowNull`, `BeforeCreate`, and `BeforeUpdate` from `'sequelize-typescript'`. The `User` class extends `Model` and is decorated with `@Table`. It has four columns: `name`, `surname`, `email`, and `age`, each decorated with `@Column`. The `User` class is exported as the default export.

```
1 import { Table, Column, Model, Unique, AllowNull, BeforeCreate, BeforeUpdate } from 'sequelize-typescript'
2
3 @Table
4 class User extends Model {
5   @Column
6   name: string
7
8   @Column
9   surname: string
10
11   @Column
12   email: string
13
14   @Column
15   age: string
16 }
17
18 export default User
```

5. DB provider



A screenshot of the Visual Studio Code editor showing the file `db.ts` in the `src > providers` directory. The code initializes a `Sequelize` instance using `sequelize-typescript`. It imports `Sequelize` from `'sequelize-typescript'` and `User` from `'../models/users/User'`. The `Sequelize` instance is configured with `database: 'some_db'`, `dialect: 'sqlite'`, `username: 'root'`, `password: ''`, `storage: ':memory:'`, `models: [User]`, and `logging: console.log`. The `Sequelize` instance is exported as the default export.

```
1 import { Sequelize } from 'sequelize-typescript'
2 import User from '../models/users/User'
3
4 const sequelize = new Sequelize({
5   database: 'some_db',
6   dialect: 'sqlite',
7   username: 'root',
8   password: '',
9   storage: ':memory:',
10  models: [User],
11  logging: console.log,
12 })
13
14 export default sequelize
```

6. Router

```
Go Run Terminal Help index.ts - lab_1 - Visual Studio Code
package.json TS index.ts ...\routes\... X TS User.ts TS db.ts
src > routes > example > TS index.ts > ...
1 import express from "express"
2 import ExampleController from "../../controllers/example/index"
3
4 const router: express.Router = express.Router()
5
6 const exampleController = new ExampleController()
7
8 router
9   .route('/')
10    .get(exampleController.get)
11    .post(exampleController.post)
12
13 export default router
```

7. Services

```
View Go Run Terminal Help index.ts - lab_1 - Visual Studio Code
... ejson TS index.ts ...\routes\... TS User.ts TS db.ts TS index.ts ...\controllers\... TS index.ts
src > services > example > TS index.ts > [0] default
1 import User from '../../models/users/User'
2 import sequelize from '../../providers/db'
3
4 class DefaultService {
5   private repo = sequelize.getRepository(User)
6
7   add(name: string, surname:string, email:string, age: string) {
8     this.repo.create({name: name, surname: surname, email: email, age: age})
9   }
10
11   get(){
12     return this.repo.findAll()
13   }
14 }
15
16 export default DefaultService
```

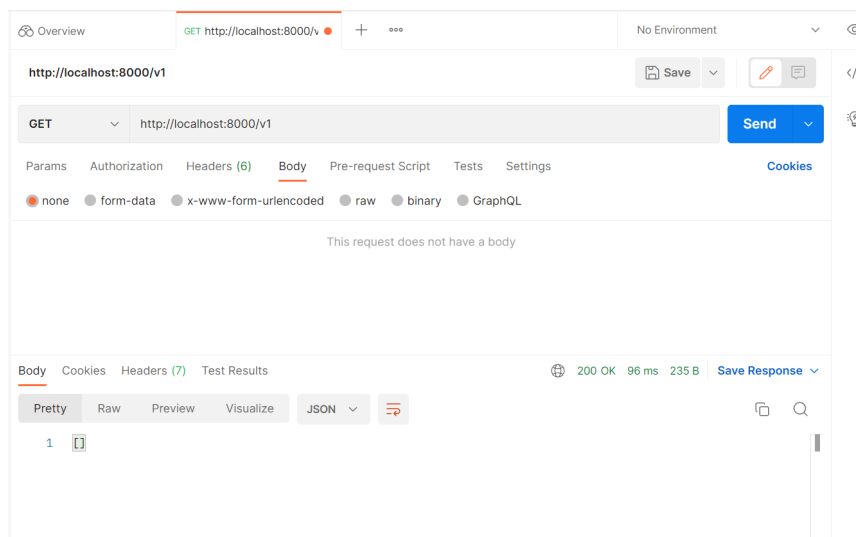
8. Src index

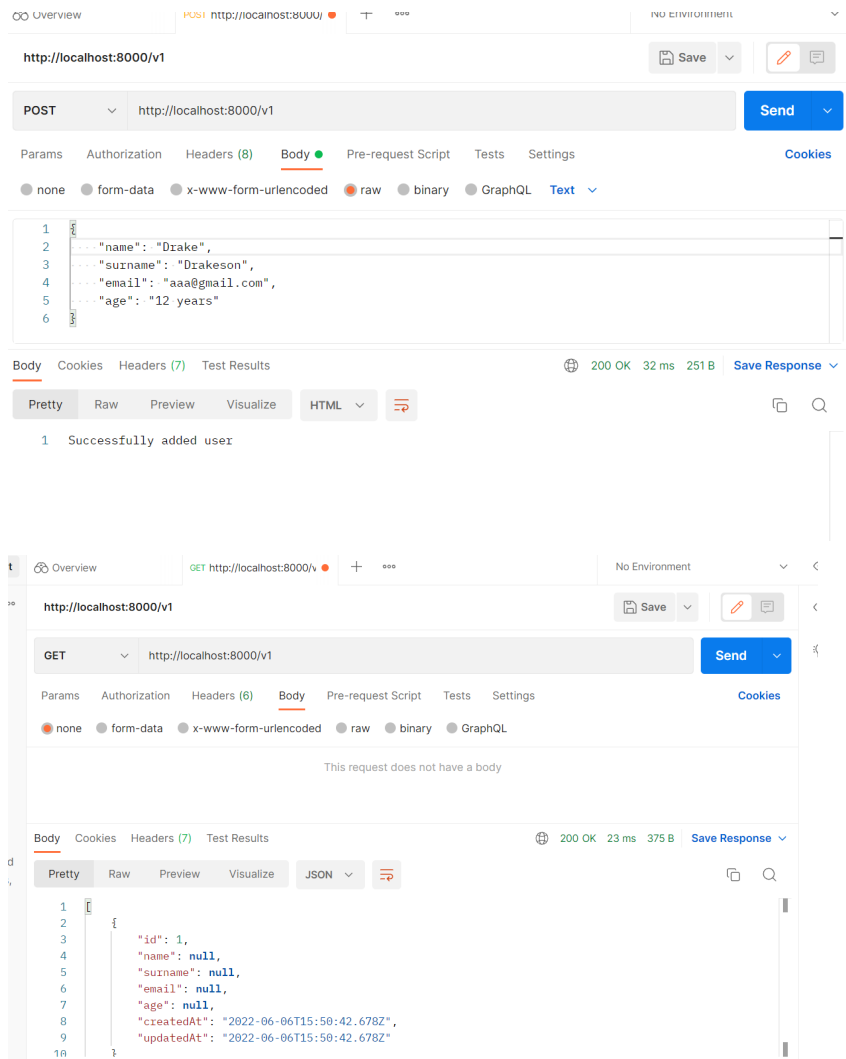
```
View Go Run Terminal Help index.ts - lab_1 - Visual Studio Code
package.json TS index.ts ...\routes\... TS User.ts
src > TS index.ts > ...
1 import App from "./core/index"
2
3 const app = new App()
4
5 app.start()
```

9. packages

```
12   "migrate": "npx sequelize db:migrate"
13 },
14   "author": "",
15   "license": "ISC",
16   "devDependencies": {
17     "@types/express": "^4.17.13",
18     "@types/node": "^17.0.27",
19     "@types/validator": "^13.7.3",
20     "@typescript-eslint/eslint-plugin": "^5.21.0",
21     "@typescript-eslint/parser": "^5.21.0",
22     "eslint": "^8.14.0",
23     "nodemon": "^2.0.15",
24     "sequelize-cli": "^6.4.1",
25     "ts-node": "^10.7.0",
26     "tslint": "^6.1.3",
27     "typescript": "^4.6.3"
28   },
29   "dependencies": {
30     "body-parser": "^1.20.0",
31     "express": "^4.18.0",
32     "reflect-metadata": "^0.1.13",
33     "sequelize": "^6.20.1",
34     "sequelize-typescript": "^2.1.3",
35     "sqlite3": "^5.0.8"
36   }
37 }
```

10. Output





Вывод

В ходе работы я создал boilerplate для дальнейшей работы по варианту