

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа № 1: Boilerplate на express +
sequelize + typescript.

Выполнил:

Безруков Андрей
Группа К33402

Проверил:

Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

Нужно написать свой boilerplate на express + sequelize / TypeORM + typescript.

Должно быть явное разделение на:

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

Ход работы

Файл package.json отвечает за установление зависимостей, создание скриптов и другую информацию о проекте

```
{ } package.json > ...
1  {
2    "name": "lw1",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "prestart": "npm run build",
8      "start": "nodemon dist/index.js",
9      "build": "npx tsc",
10     "lint": "npx eslint . --ext .ts"
11   },
12   "author": "",
13   "license": "ISC",
14   "devDependencies": {
15     "@types/express": "^4.17.17",
16     "@types/node": "^18.14.6",
17     "@types/node-fetch": "^2.6.2",
18     "@typescript-eslint/eslint-plugin": "^5.54.0",
19     "@typescript-eslint/parser": "^5.54.0",
20     "dotenv": "^16.0.3",
21     "eslint": "^8.35.0",
22     "nodemon": "^2.0.21",
23     "sequelize-cli": "^6.6.0",
24     "ts-node": "^10.9.1",
25     "tslint": "^6.1.3",
26     "typescript": "^4.9.5"
27   },
28   "dependencies": {
29     "express": "^4.18.2",
30     "node-fetch": "^3.3.0",
31     "sequelize": "^6.29.0",
32     "sqlite3": "^5.1.4"
33   }
34 }
```

Модель пользователя – models/user.ts включает в себя следующие поля: id, имя, фамилия, отчество, электронная почта, ник и пароль

```
src > db > models > TS user.ts > User > username
1  import { DataTypes, Model, Optional } from 'sequelize';
2  import { sequelize } from '.';
3
4  interface User {
5      id: number;
6      name: string;
7      surname: string;
8      middlename: string;
9      email: string;
10     username: string;
11     password: string;
12 };
13
14 interface UserCreation
15     extends Optional<User, 'id'> { }
16
17 interface UserInstance
18     extends Model<User, UserCreation>,
19     User {
20     createdAt?: Date;
21     updatedAt?: Date;
22 }
23
24 const User = sequelize.define<UserInstance>(
25     'User',
26     {
27         id: {
28             allowNull: false,
29             autoIncrement: true,
30             primaryKey: true,
```

```
31         type: DataTypes.INTEGER,
32         unique: true,
33     },
34     name: {
35         allowNull: false,
36         type: DataTypes.TEXT,
37     },
38     surname: {
39         allowNull: false,
40         type: DataTypes.TEXT,
41     },
42     middlename: {
43         allowNull: true,
44         type: DataTypes.TEXT,
45     },
46     email: {
47         allowNull: false,
48         type: DataTypes.TEXT,
49     },
50     username: {
51         allowNull: false,
52         type: DataTypes.TEXT,
53     },
54     password: {
55         allowNull: false,
56         type: DataTypes.TEXT,
57     },
58 }
59 );
```

Sequelize был использован при работе с sqlite. Файл models/index.ts

```
src > db > models > TS index.ts > ...
1  import { Sequelize } from 'sequelize';
2
3  const env = process.env.NODE_ENV || 'development';
4  const config = require(__dirname + '/../../config.json')[env];
5
6  const sequelize = config.url
7    ? new Sequelize(config.url, config)
8    : new Sequelize(config.database, config.username, config.password, config);
9
10 export { Sequelize, sequelize };
```

CRUD-методы были реализованы через контроллер controllers/index.ts, который позволяет обрабатывать разные запросы пользователя, например, получить пользователя по электронной почте или удалить его.

```
src > controllers > TS index.ts > UserController > get
1  import UserService from "../services"
2
3  class UserController {
4    private userService: UserService
5    constructor() {
6      this.userService = new UserService()
7    }
8    get = async (request: any, response: any) => {
9      try {
10         const users = await this.userService.getAll()
11         return response.json(users);
12       } catch (e: any) {
13         response.status(404).send({ "error": e.message })
14       }
15     }
16     getByID = async (request: any, response: any) => {
17       try {
18         const user = await this.userService.getById(request.params.id)
19         return response.json(user);
20       } catch (error: any) {
21         response.status(404).send({ "error": error.message })
22       }
23     }
24     getByEmail = async (request: any, response: any) => {
25       try {
26         const user = await this.userService.getByEmail(request.params.email)
27         return response.json(user);
28       } catch (error: any) {
29         response.status(404).send({ "error": error.message })
30       }
31     }
32   }
```

```

31     }
32     post = async (request: any, response: any) => {
33         try {
34             const { body } = request
35             const user = await this.userService.create(body);
36             return response.json({user, msg: "created" })
37         } catch (error: any) {
38             response.status(400).send({ "error": error.message })
39         }
40     }
41     put = async (request: any, response: any) => {
42         try {
43             const { body } = request;
44             const user = await this.userService.update(request.params.id, body)
45             return response.json({user, msg: 'user was updated' })
46         } catch (error: any) {
47             response.status(404).send({ "error": error.message })
48         }
49     }
50     delete = async (request: any, response: any) => {
51         try {
52             const user = await this.userService.delete(request.params.id)
53             return response.json({msg: 'user was deleted' })
54         } catch (error: any) {
55             response.status(404).send({ "error": error.message })
56         }
57     }
58 }
59
60 export default UserController

```

Роутинг (эндпоинты) был реализован в routes/index.ts и опирается на контроллер

```

src > routes > TS index.ts > [⌕] router
1   import express from "express"
2   import UserController from "../controllers"
3
4   const router = express.Router()
5
6   const userController = new UserController()
7
8   router.route('/users').get(userController.get)
9
10  router.route('/users/id/:id').get(userController.getbyID)
11
12  router.route('/users/email/:email').get(userController.getbyEmail)
13
14  router.route('/users').post(userController.post)
15
16  router.route('/users/:id').put(userController.put)
17
18  router.route('/users/:id').delete(userController.delete)
19
20  export default router

```

Сервисы для работы с моделями в services/index.ts

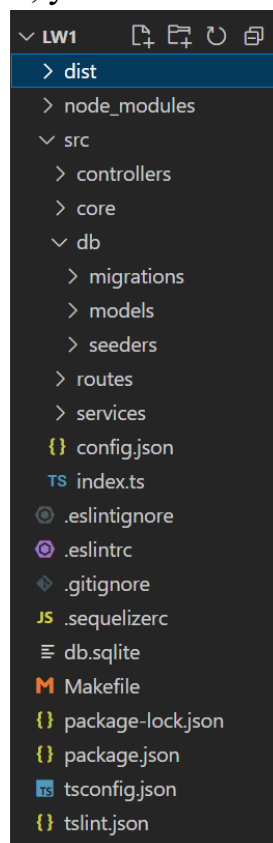
```
src > services > TS index.ts > UserService
1  import User from '../db/models/user';
2
3  class UserService {
4
5      async create(userInfo: any) {
6          try {
7              const newUser = await User.create(userInfo)
8              return newUser
9          } catch (e: any) {
10             throw new Error(e)
11          }
12      }
13
14      async getAll(){
15          const users = await User.findAll()
16
17          if (users) return users
18          else return { "msg": "users were not found" }
19      }
20
21      async getById(id: number) {
22          const user = await User.findPk(id)
23
24          if (user) return user
25
26          throw new Error(`user with id ${id} was not found`)
27      }
28
29      async getByEmail(email: string) {
```

```
src > services > TS index.ts > UserService
31      const user = await User.findOne({ where: { email: email } })
32
33      if (user) return user
34
35      throw new Error(`user with email ${email} was not found`)
36  }
37
38  async update(id:number, data: any) {
39      try {
40          const user = await User.findPk(id)
41          if (user) {
42              user.update(data)
43          }
44          return user
45      } catch (e: any) {
46          throw new Error(e)
47      }
48  }
49
50  async delete(id:number) {
51      try {
52          await User.destroy({where: {id:id}})
53      } catch (e: any) {
54          throw new Error(e)
55      }
56  }
57
58  }
59
60  export default UserService
```

Также благодаря sequelize-cli были проведены миграции (db:migrate) и добавлены в папку migrations, добавлены сидеры (seed:generate) в папку seeders. Именно они позволили добавить пользователей.

```
src > db > seeders > JS 20230304165946-user.js > ...
1  'use strict';
2
3  module.exports = {
4    up: (queryInterface, Sequelize) => {
5      return queryInterface.bulkInsert('Users', [{
6        name: 'Polina',
7        surname: 'Savinova',
8        middlename: 'Andreevna',
9        email: '@polisavgmail.com',
10       username: 'polisav',
11       password: 'crazygirl',
12       createdAt: new Date(),
13       updatedAt: new Date()
14     },
15     {
16       name: 'Galina',
17       surname: 'Vasnetsova',
18       middlename: 'Dmitrievna',
19       email: 'papinadochka@gmail.com',
20       username: 'galyavas',
21       password: 'nerdgirl',
22       createdAt: new Date(),
23       updatedAt: new Date()
24     }
25   ], {});
26    },
27    down: () => {}
28  };
29
```

Структура проекта в итоге выглядит следующим образом – папки dist (собранный проект), node_modules, src (рабочая папка) и дополнительные файлы (гитигнор, база данных, установленные пакеты и др.)



Пример get-запроса по электронной почте

GET

http://127.0.0.1:9000/users/email/papinadochka@gmail.com

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	KEY	VALUE	DESCRIPTION		Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (7)

Test Results

200 OK 177 ms 464 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1  {
2    "id": 2,
3    "name": "Galina",
4    "surname": "Vasnetsova",
5    "middlename": "Dmitrievna",
6    "email": "papinadochka@gmail.com",
7    "username": "galyavas",
8    "password": "nerdgirl",
9    "createdAt": "2023-03-04T17:10:53.897Z",
10   "updatedAt": "2023-03-04T17:10:53.897Z"
11 }
```

Пример get-запроса по id

GET

http://127.0.0.1:9000/users/id/4

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	KEY	VALUE	DESCRIPTION		Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (7)

Test Results

200 OK 83 ms 455 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1  {
2    "id": 4,
3    "name": "Boris",
4    "surname": "Konovalov",
5    "middlename": "Alexandrovich",
6    "email": "@gmail.com",
7    "username": "boryanakone",
8    "password": "nerdboy",
9    "createdAt": "2023-03-04T17:10:53.897Z",
10   "updatedAt": "2023-03-04T17:10:53.897Z"
11 }
```

Пример delete-запроса

http://127.0.0.1:9000/users/4

Save

Send

DELETE

http://127.0.0.1:9000/users/4

Params Authorization Headers (8) Body • Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (7) Test Results

200 OK 24 ms 261 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "msg": "user was deleted"
3 }
```

Вывод

Мною были получены навыки разработки boilerplate на express + sequelize + typescript. Этот шаблон проекта позволяет просматривать, добавлять, удалять, изменять данные о пользователях.