

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Backend-Разработка

Отчет

Лабораторная работа 1

Выполнил:
Бункута Натан

Группа:
К33412

Проверил:
Добрияков Д.И

Санкт-Петербург

2023 г.

Задание:

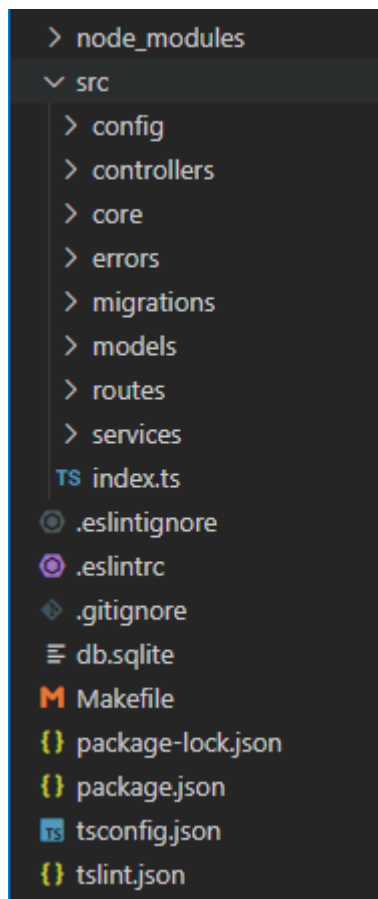
Нужно написать свой boilerplate на express + sequelize / TypeORM + typescript.

Должно быть явное разделение на:

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

Ход работы:

Boilerplate components



Models

models/index.ts

```
src > models > TS index.ts > ...
1  import { Sequelize } from 'sequelize';
2
3  const env = process.env.NODE_ENV || 'development';
4  const config = require(__dirname + '/../config/config.js')[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 };
```

models/user.ts

```
src > models > TS user.ts > [User] > hometown
9  }
10 interface UserCreationAttributes
11 | extends Optional<UserAttributes, 'email'> { }
12
13 interface UserInstance extends Model<UserAttributes, UserCreationAttributes>, UserAttributes {
14 |   createdAt?: Date;
15 |   updatedAt?: Date;
16 | }
17
18 const User = sequelize.define<UserInstance>(
19 |   'User',
20 |   {
21 |     username: {
22 |       type: DataTypes.STRING,
23 |     },
24 |     password: {
25 |       type: DataTypes.STRING,
26 |     },
27 |     email: {
28 |       type: DataTypes.STRING,
29 |     },
30 |     hometown: {
31 |       type: DataTypes.STRING,
32 |     }
33 |   },
34 | );
35
36 export default User
```

Controllers

index.ts

```
src > controllers > users > TS index.ts > UserController > getByUsername > "error 404"
1  import UserService from "../../services/users/index";
2
3  class UserController {
4      private userService = new UserService
5
6      constructor() {
7          this.userService = new UserService()
8      }
9
10     post = async (request: any, response: any) => {
11         const { body } = request
12
13         try {
14             const user = await this.userService.create(body)
15
16             response.status(201).send(user)
17         } catch (error: any) {
18             response.status(400).send({ "error 400": error.message })
19         }
20     }
21
22     getAll = async (request: any, response: any) => {
23         try {
24             const users = await this.userService.getAll()
25
26             response.send(users)
27         } catch (error: any) {
28             response.status(404).send({ "error 404": error.message })
29         }
30     }
31 }
32
```

```

33     getById = async (request: any, response: any) => {
34         try {
35             const user = await this.userService.getById(
36                 Number(request.params.id)
37             )
38
39             response.send(user)
40         } catch (error: any) {
41             response.status(404).send({ "error 404": error.message })
42         }
43     }
44
45     getByUsername = async (request: any, response: any) => {
46         try {
47             const user = await this.userService.getByUsername(
48                 request.params.username
49             )
50
51             response.send(user)
52         } catch (error: any) {
53             response.status(404).send({ "error 404": error.message })
54         }
55     }
56 }
57
58 export default UserController

```

Routes

routes/index.ts

```

src > routes > v1 > users > TS index.ts > ...
1  ∨ import express from "express";
2  import UserController from "../../controllers/users/index";
3
4  const router: express.Router = express.Router()
5
6  const controller: UserController = new UserController()
7
8  router.route('/add/')
9      .post(controller.post)
10
11 router.route('/profiles')
12     .get(controller.getAll)
13
14 router.route('/profile/id/:id')
15     .get(controller.getById)
16
17 router.route('/profile/username/:username')
18     .get(controller.getByUsername)
19
20
21 export default router

```

Services

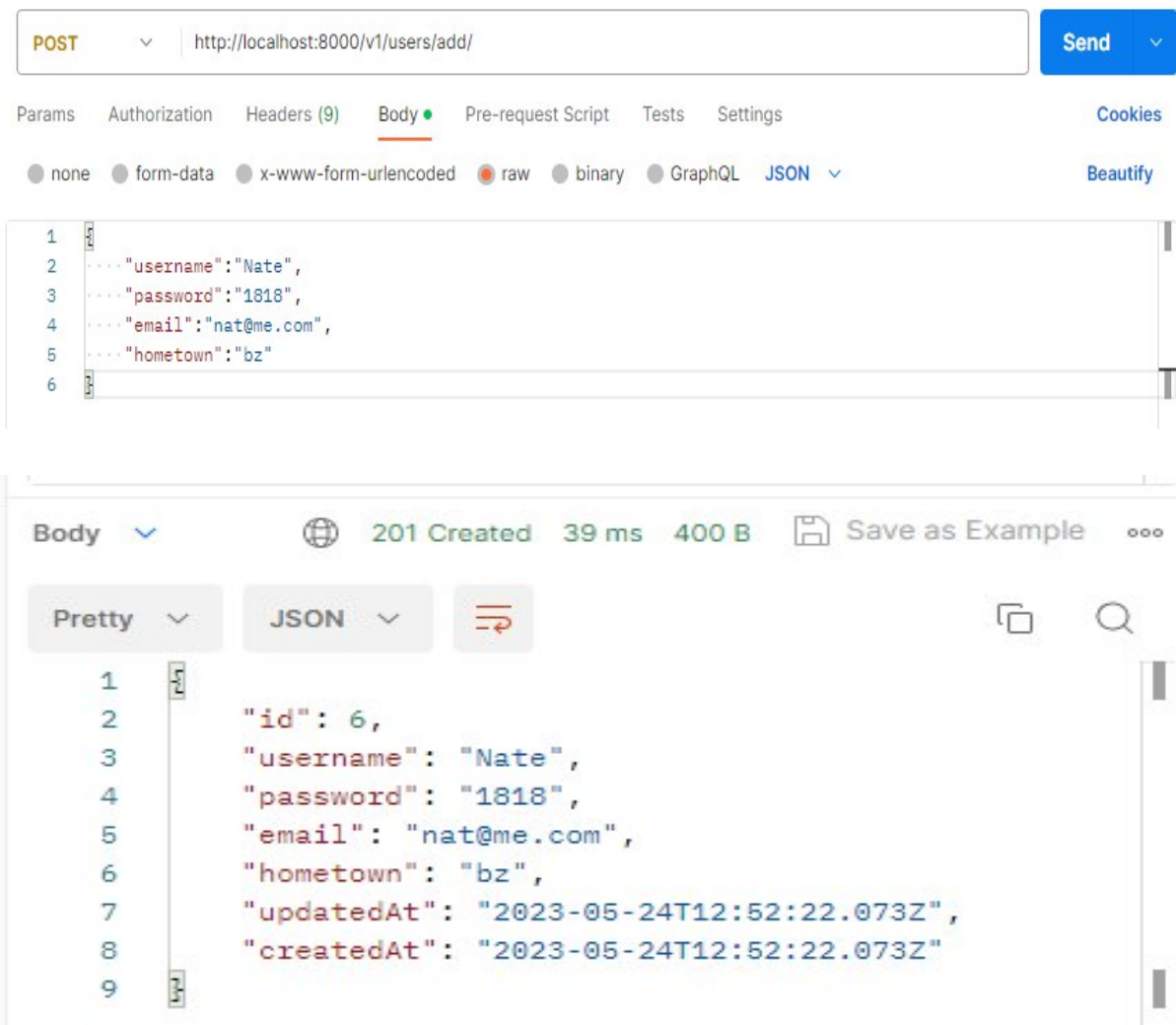
users/index.ts

```
src > services > users > TS index.ts > ...
1  ✓ import UserError from "../../errors/users/index";
2  import User from "../../models/user"
3
4  class UserService {
5
6      async create(userData: any) {
7          try {
8              console.log(userData)
9              const user = await User.create(userData)
10
11              return user.toJSON()
12          } catch (e: any) {
13              console.log(e)
14              const errors = e.errors.map((error: any) => error.message)
15
16              throw new UserError(errors)
17          }
18      }
19
20      async getAll() {
21          const users = await User.findAll()
22
23          if (users) return users
24
25          throw new UserError('Sorry, No user found!')
26      }
27
28      async getById(id: number) {
29          const user = await User.findByPk(id)
30
31          if (user) return user.toJSON()
32
33          throw new UserError('Sorry, No such id user found !')
34      }
35
```

```
35
36      async getByUsername(username: string) {
37          const user = await User.findOne({
38              where: {
39                  username: username
40              }
41          })
42
43          if (user) return user.toJSON()
44
45          throw new UserError('Sorry, such username not found !')
46      }
47  }
48
49  export default UserService
```

Проверка в Postman

Add a new user



Get list of all users

GET ▼ | http://localhost:8000/v1/users/profiles Send ▼

Params Authorization Headers (9) **Body** ● Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL **JSON** ▼ Beautify

```
1  [
2    {
3      "username": "Omar",
4      "password": "20203",
5      "email": "omodel@afro.com",
6      "hometown": "Gambia"
7    },
8    {
9      "username": "Durhel",
10     "password": "20223",
11     "email": "dubuzz@congo.com",
12     "hometown": "Nkayi",
13     "createdAt": "2023-05-24T13:20:53.017Z",
14     "updatedAt": "2023-05-24T13:20:53.017Z"
15   },
16   {
17     "id": 10,
18     "username": "Omar",
19     "password": "20203",
20     "email": "omodel@afro.com",
21     "hometown": "Gambia",
22     "createdAt": "2023-05-24T13:21:54.150Z",
23     "updatedAt": "2023-05-24T13:21:54.150Z"
24   }
25 ]
```

Body Cookies Headers (7) Test Results 200 OK 8 ms 1.9 KB Save as Example ⋮

Pretty Raw Preview Visualize **JSON** ▼ ⌵

```
76  {
77    "username": "Durhel",
78    "password": "20223",
79    "email": "dubuzz@congo.com",
80    "hometown": "Nkayi",
81    "createdAt": "2023-05-24T13:20:53.017Z",
82    "updatedAt": "2023-05-24T13:20:53.017Z"
83  },
84  {
85    "id": 10,
86    "username": "Omar",
87    "password": "20203",
88    "email": "omodel@afro.com",
89    "hometown": "Gambia",
90    "createdAt": "2023-05-24T13:21:54.150Z",
91    "updatedAt": "2023-05-24T13:21:54.150Z"
92  }
93 }
```

Find user by id

GET ▼ | http://localhost:8000/v1/users/profile/id/9 Send ▼

Params Authorization Headers (9) **Body** ● Pre-request Script Tests Settings Cookies

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL **JSON** ▼ Beautify

```
1  {
2    "id": 9,
3    "username": "Durhel",
4    "password": "20223",
5    "email": "dubuzz@congo.com",
6    "hometown": "Nkayi",
7    "createdAt": "2023-05-24T13:20:53.017Z",
8    "updatedAt": "2023-05-24T13:20:53.017Z"
9  }
```

Body Cookies Headers (7) Test Results 200 OK 8 ms 407 B Save as Example ⋮

Pretty Raw Preview Visualize **JSON** ▼ ⌵

Find user by his username

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8000/v1`
- Method:** `GET`
- Path:** `http://localhost:8000/v1/users/profile/username/Mamoudou`
- Body:**

```
1 {  
2   ... "username": "Omar",  
}
```
- Response:** `200 OK`, `5 ms`, `413 B`. Status: `Save as Example`.
- Body Content (JSON):**

```
1 {  
2   "id": 8,  
3   "username": "Mamoudou",  
4   "password": "129819",  
5   "email": "fakoudou@guin.com",  
6   "hometown": "Conakry",  
7   "createdAt": "2023-05-24T13:19:28.060Z",  
8   "updatedAt": "2023-05-24T13:19:28.060Z"  
9 }
```

Potential error exemple

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8000/v1`
- Method:** `GET`
- Path:** `http://localhost:8000/v1/users/profile/username/mamoudou`
- Body:** (Empty)
- Response:** `404 Not Found`, `6 ms`, `287 B`. Status: `Save as Example`.
- Body Content (JSON):**

```
1 {  
2   "error": "User with this username not found"  
3 }
```

Вывод:

При выполнении данной лабораторной работы, мы научили писать простой boilerplate с использованием средств TypeScript, Express + Sequelize. Были разделены компоненты как модели, контроллеры, роуты и сервисы.