

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа 1. “Написание Boilerplate для
разработки сервиса с использованием TS, Sequelize и
Express”

Выполнил:

Тимофеев Николай

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

1. Создадим npm package

```
labs > K33402 > Timofeev_Nikolai > LR2 > package.json > {} scripts > migrate
2   "name": "lr1",
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    "migrate": "npx sequelize db:migrate"
12  },
13  "author": "Timofeev Nikolas",
14  "license": "",
15  "dependencies": {
16    "bcrypt": "^5.1.0",
17    "body-parser": "^1.20.2",
18    "cors": "^2.8.5",
19    "express": "^4.18.2",
20    "passport": "^0.6.0",
21    "passport-jwt": "^4.0.1",
22    "reflect-metadata": "^0.1.13",
23    "sequelize": "^6.30.0",
24    "sequelize-typescript": "^2.1.5",
25    "sequelize-typescript-migration-v2": "^0.0.2-beta.6",
26    "sqlite3": "^5.1.6",
27    "tsc": "^10.0.2",
28    "typeorm": "^0.3.13",
29    "uuid": "^9.0.0"
30  },
```

2. Структура проекта

```
30 directories, 27 files
timofeev41 on MacBook-Pro-Nikolas.local in ~/Projects/ITMO-ICT-Backend
● $ tree ../../LR1
../../LR1
├── db.sqlite
├── nodemon.json
├── package-lock.json
├── package.json
├── src
│   ├── configs
│   │   └── db.mjs
│   ├── controllers
│   │   └── users
│   │       └── User.ts
│   ├── core
│   │   └── index.ts
│   ├── errors
│   │   └── users
│   │       └── User.ts
│   ├── index.ts
│   ├── middlewares
│   │   └── passport.ts
│   ├── models
│   │   ├── auth
│   │   │   └── RefreshToken.ts
│   │   └── users
│   │       └── User.ts
│   ├── providers
│   │   └── db.ts
│   ├── routes
│   │   └── v1
│   │       ├── index.ts
│   │       └── users
│   │           └── User.ts
│   ├── services
│   │   ├── auth
│   │   │   └── RefreshToken.ts
│   │   └── users
│   │       └── User.ts
│   └── utils
│       ├── checkPassword.ts
│       └── hashPassword.ts
├── tsconfig.json
└── tslint.json
```

Корень исходных файлов **src** включает в себя следующие директории:

- **core** – точка входа в приложение, объединяющая в себе все составляющие;
 - **configs** – файлы конфигурации (файл для подключения к БД);
 - **controllers** – контроллеры, отвечающие за логику обработки http-запросов;
 - **models** – модели sequelize;
 - **providers** – точки доступа к данным;
 - **routes** – описание маршрутов express;
 - **services** – службы, которые содержат запросы к базе данных и возвращают объекты или выдают ошибки;
 - **utils** – вспомогательные файлы, которые используются во всем приложении
- **middlewares** - содержит аутентификацию с использованием passport.js.

3. Модели

Token:

```
You, 3 дня назад | 1 author (You)
import { Table, Column, Model, Unique, AllowNull, ForeignKey } from 'sequelize-typescript'
import User from '../users/User'

You, 3 дня назад | 1 author (You)
@Table
class RefreshToken extends Model {
  @Unique
  @AllowNull(false)
  @Column
  token: string

  @ForeignKey(() => User)
  @Column
  userId: number
}

export default RefreshToken
```

User:

You, 3 дня назад | 1 author (You)

@Table

```
class User extends Model {
```

```
    @Unique
```

```
    @Column
```

```
    username: string
```

```
    @Column
```

```
    firstName: string
```

```
    @Column
```

```
    lastName: string
```

```
    @Unique
```

```
    @Column
```

```
    email: string
```

```
    @AllowNull(false)
```

```
    @Column
```



```
    password: string
```

You, 3 дня назад • feat: lri added ...

4. Контроллеры

UserController:

```
class UserController {  
    private userService: UserService;  
  
    constructor() {  
        this.userService = new UserService();  
    }  
  
    get = async (request: any, response: any) => {  
    };  
  
    post = async (request: any, response: any) => {  
    };  
  
    me = async (request: any, response: any) => {  
    };  
  
    auth = async (request: any, response: any) => {  
    };  
  
    getAll = async (request: any, response: any) => {  
    };  
  
    getByUsername = async (request: any, response: any) => {  
    };  
  
    refreshToken = async (request: any, response: any) => {  
    };  
}
```

Методы класса UserController:

- get: находит пользователя по id;
- post: создание нового пользователя;
- me: возвращает данные о пользователе, полученные из объекта запроса;
- auth: генерирует новый токен доступа и токен обновления, если пользователь залогинился;
- refreshToken: генерирует новый JWT токен;
- getAll: получает информацию по всем пользователям
- getByUsername: находит пользователя

5. Сервисы

UserService - сервис для управления пользователями

```
class UserService {  
  >   async getById(id: number) : Promise<User> { ...  
    }  
  
  >   async create(userData: Partial<User>): Promise<User> { ...  
    }  
  
  >   async getAll() { ...  
    }  
  
  >   async getByUsername(username: string) { ...  
    }  
  
  >   async checkPassword(email: string, password: string) : Promise<any> { ...  
    }  
}
```

You, 3 дня назад • feat: lr1 added

6. Роуты

User.ts - авторизация и работа с юзерами

```
router.route("/reg").post(controller.post);
router.route("/account").get(passport.authenticate("jwt", { session: false }), controller.me);
router.route("/account/:id").get(controller.get);
router.route("/login").post(controller.auth);
router.route("/refresh").post(controller.refreshToken);
router.route("/accounts").get(controller.getAll);
router.route("/accounts/:username").get(controller.getByUsername);
```

Вывод

В ходе первой лабораторной работы был разработан boilerplate с продуманной структурой и необходимыми конфигурационными файлами с помощью следующих инструментов: express, sequelize/TypeORM, typescript.