САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №1

Boilerplate на express + sequelize / TypeORM + typescript

Выполнила:
Зайцева А. А.
Группа K33402

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

В качестве основы проекта было использовано приложение из практической работы №2, а также пример, размещенный по адресу https://github.com/kantegory/express-sequelize-boilerplate.

Прежде всего был создан файл **package.json**:

```
{
  "name": "1",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "prestart": "npm run build",
    "start": "nodemon dist/index.js",
    "build": "npx tsc",
    "lint": "npx eslint . --ext .ts",
    "migrate": "npx sequelize db:migrate",
    "seed": "npx sequelize-cli db:seed:all"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.1.0",
    "cors": "^2.8.5",
    "express": "^4.18.2",
    "reflect-metadata": "^0.1.13",
    "sequelize": "^6.30.0",
    "sequelize-typescript": "^2.1.5",
    "sqlite3": "^5.1.6"
  },
  "devDependencies": {
    "@types/bcrypt": "^5.0.0",
```

```
    "@types/cors": "^2.8.13",
    "@types/express": "^4.17.17",
    "@types/ini": "^1.3.31",
    "@types/node": "^18.15.11",
    "@types/validator": "^13.7.14",
    "@typescript-eslint/eslint-plugin": "^5.57.0",
    "@typescript-eslint/parser": "^5.57.0",
    "eslint": "^8.37.0",
    "ini": "^4.0.0",
    "nodemon": "^2.0.22",
    "sequelize-cli": "^6.6.0",
    "ts-node": "^10.9.1",
    "typescript": "^5.0.3"
  }
}
```

В него вошли необходимые скрипты и зависимости для работы приложения.

Также помимо файла **package.json**, были созданы необходимые файлы конфигурации окружения, а именно:

- **.editorconfig**
- **.eslintrc.js**
- **.sequelizerc**
- **nodemon.json**
- **tsconfig.json**

После этого была создана структура директорий проекта. Корень исходный файлов **src** включает в себя следующие директории:

- **app** – точка входа в приложение, объединяющая в себе все составляющие;
- **config –** файлы конфигурации;
- **controllers –** контроллеры, обеспечивающие доступ к бизнес-логике приложения извне;
- **migrations –** файлы миграций sequelize;
- **models –** модели sequelize;
- **providers –** точки доступа к данным;
- **routes –** описание маршрутов express;
- **seeders –** описание автоматически-генерируемых тестовых данных для sequelize;

- **services** – описание объектов доступа к моделям sequelize;
- **utils** – вспомогательные файлы.

Основными отличиями от примера являются:

1. Типизация файла configParser.ts:

```
import ini from 'ini'
import fs from 'fs'
import { Dialect } from 'sequelize';

export const enum ConfigModule {
    DATABASE = 'DATABASE',
    SERVER = 'SERVER',
}

export interface DatabaseConfig {
    name: string;
    dialect: Dialect;
    username: string;
    password: string;
    storage: string;
    host: string;
    port: number;
}

export interface ServerConfig {
    port: number;
    host: string;
}

type Config<T extends ConfigModule> = T extends ConfigModule.DATABASE ?
DatabaseConfig : ServerConfig;

export const parseConfig = <T extends ConfigModule>(path: string,
moduleName: T) => {
    const configData: string = fs.readFileSync(path, 'utf-8');
    const parsedConfig: Config<T> = ini.parse(configData)[moduleName];

    return parsedConfig;
}
```

2. Объединение функций по работе с паролями в файл passwords.ts:

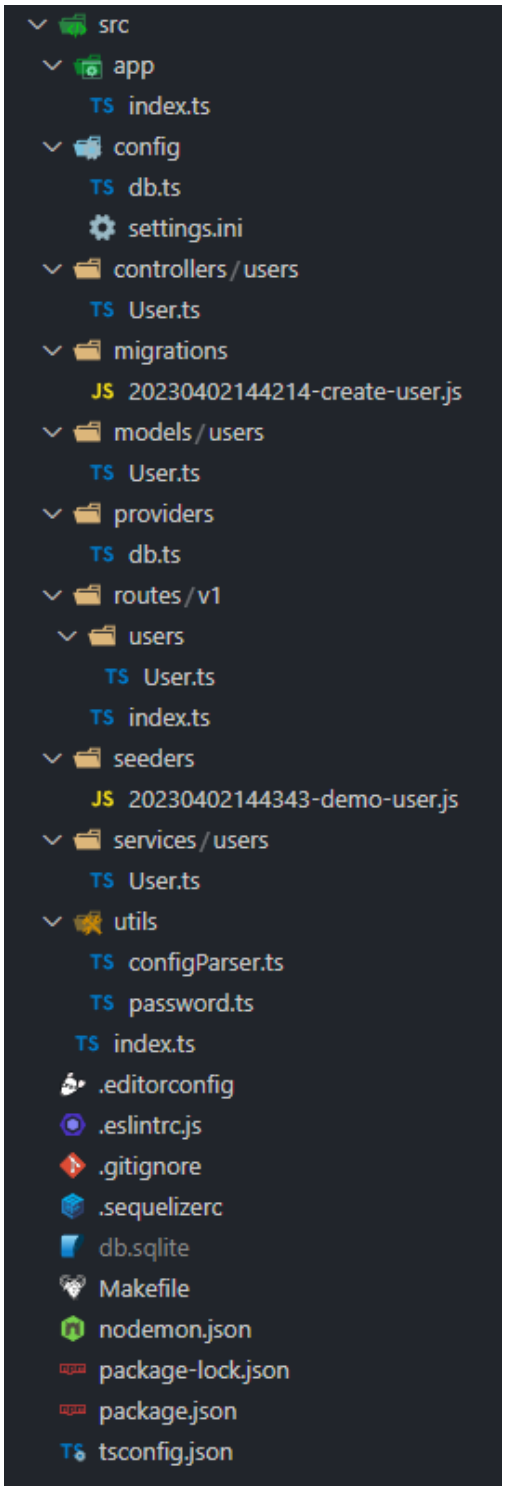
```
import bcrypt from 'bcrypt'
import User from '../models/users/User'

export const hashPassword = (password: string) =>
bcrypt.hashSync(password, bcrypt.genSaltSync(8))
```

```
export const checkPassword = (user: User, password: string) =>
bcrypt.compareSync(password, user.password)
```

3. Убрана панель администрирования и поддержка авторизации пользователя по JWT в силу высокой специфичности данного функционала.

Итоговая иерархия проекта представлена на рисунке:



Вывод

В ходе первой лабораторной работы мной был разработан boilerplate с использованием веб-фреймворка express, orm-инструмента sequelize (typescript версия) и языка программирования typescript. Получившийся boilerplate впоследствии может быть использован для быстрого и удобного развертывания приложений разной сложности и направленности.