

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Практическая работа №2
Знакомство с ORM Sequelize

Выполнила:
Зайцева А. А.
Группа К33402

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

- Продумать свою собственную модель пользователя
- Реализовать набор из CRUD-методов для работы с пользователями средствами Express + Sequelize
- Написать запрос для получения пользователя по id/email

Ход работы

1. Продумать свою собственную модель пользователя

Разработанная модель пользователя включает следующие поля:

- **id** (integer, autoincrement, pk) – создается автоматически;
- **createdAt** (date) – создается автоматически;
- **updatedAt** (date) – создается автоматически;
- **email** (string, unique, not null);
- **password** (string, not null);
- **firstName** (string);
- **lastName** (string);
- **dob** (date).

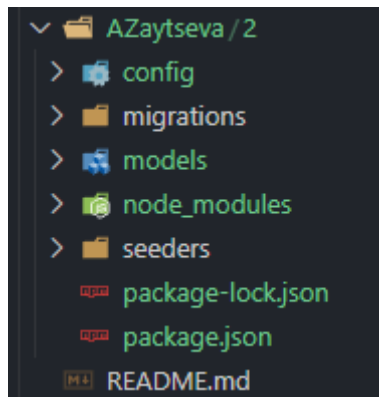
2. Реализовать набор из CRUD-методов для работы с пользователями средствами Express + Sequelize

Сначала инициализируем проект и установим все необходимые зависимости:

```
$ npm init
$ npm install --save sequelize sqlite3 express
$ npm install --save-dev sequelize-cli nodemon
```

Теперь инициализируем проект с помощью sequelize-cli:

```
$ npx sequelize init
```



Теперь создадим модель и миграцию пользователя с помощью sequelize-cli:

```
$ npx sequelize-cli model:generate --name User --  
attributes  
email:string,password:string,firstName:string,lastName:st  
ring,dob:date
```

Теперь отредактируем получившиеся файлы модели и миграции в соответствии с разработанной нами моделью пользователя:

```
User.init({  
  email: {  
    type: DataTypes.STRING,  
    allowNull: false,  
    unique: true,  
  },  
  password: {  
    type: DataTypes.STRING,  
    allowNull: false,  
  },  
  firstName: DataTypes.STRING,  
  lastName: DataTypes.STRING,  
  dob: DataTypes.DATE  
, {
```

```

async up(queryInterface, Sequelize) {
  await queryInterface.createTable('Users', {
    id: {
      allowNull: false,
      autoIncrement: true,
      primaryKey: true,
      type: Sequelize.INTEGER
    },
    email: {
      allowNull: false,
      unique: true,
      type: Sequelize.STRING,
    },
    password: [
      allowNull: false,
      type: Sequelize.STRING
    ],
    firstName: {
      type: Sequelize.STRING
    },
    lastName: {
      type: Sequelize.STRING
    },
    dob: {
      type: Sequelize.DATE
    },
    createdAt: {
      allowNull: false,
      type: Sequelize.DATE
    },
    updatedAt: {
      allowNull: false,
      type: Sequelize.DATE
    }
  });
},

```

Запустим миграцию с помощью команды:

```
$ npx sequelize-cli migrate
```

```
~/p/s/6/ITMO-ICT-Backend-2023/h/K33402/AZaytseva/2 | hw2 ?1 npx sequelize db:migrate

Sequelize CLI [Node: 18.12.1, CLI: 6.6.0, ORM: 6.29.3]

Loaded configuration file "config/config.json".
Using environment "development".
== 20230312170940-create-user: migrating =====
== 20230312170940-create-user: migrated (0.084s)
```

Для тестирования создадим seed с тестовыми пользователями базы данных:

```
$ npx sequelize-cli seed:generate --name demo-user
```

```
homeworks > K33402 > AZaytseva > 2 > seeders > JS 20230312173315-demo-user.js > [?] <unknown>

1  'use strict';
2
3  /** @type {import('sequelize-cli').Migration} */
4  module.exports = {
5    async up (queryInterface, Sequelize) {
6      return queryInterface.bulkInsert('Users', [
7        {
8          firstName: 'First',
9          lastName: 'User',
10         email: 'first@example.com',
11         password: 'Testtest123',
12         dob: new Date('2002-02-02'),
13         createdAt: new Date(),
14         updatedAt: new Date()
15       },
16       {
17         firstName: 'Second',
18         lastName: 'User',
19         email: 'second@example.com',
20         password: 'Testtest123',
21         dob: new Date('2002-02-02'),
22         createdAt: new Date(),
23         updatedAt: new Date()
24       },
25     ]);
26   },
27
28   async down (queryInterface, Sequelize) {
29     return queryInterface.bulkDelete('Users', null, {});
30   }
31 };
32
```

Запустим seed командой:

```
$ npx sequelize-cli db:seed:all
```

```
~/p/s/6/ITMO-ICT-Backend-2023/homeworks/K33402/AZaytseva/2 | hw2 ?1 npx sequelize-cli db:seed:all
Sequelize CLI [Node: 18.12.1, CLI: 6.6.0, ORM: 6.29.3]

Loaded configuration file "config/config.json".
Using environment "development".
== 20230312173315-demo-user: migrating =====
== 20230312173315-demo-user: migrated (0.033s)
```

Теперь напишем набор CRUD-методов для работы с пользователем.

Были реализованы следующие методы:

- GET /users[?email]
- GET /users/:id
- POST /users
- PUT /users/:id
- DELETE /users
- DELETE /users/:id

Полный код реализованных методов доступен в директории практической работы в файле index.js.

3. Написать запрос для получения пользователя по id/email

Для получения по id:

```
curl -X GET http://localhost:3000/users/2
```

Для получения по email:

```
curl -X GET \
'http://localhost:3000/users?email=third%40example.com'
```

Вывод

В ходе второй практической работы я научилась работать с orm npm пакетом sequelize. Была разработана модель пользователя, подготовлена и запущена миграция средствами sequelize-cli. С помощью фреймворка express был разработан набор CRUD методов для работы с сущностью пользователя. Набор реализованных методов был протестирован в программе Postman. Там же были сгенерированы curl запросы.