

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Бэк-энд разработка

**Отчет**

по практической работе №2 «Знакомство с ORM Sequelize»

Выполнил:

Поляков Андрей

Группа  
К33402

Проверил:  
Добряков Д. И.

Санкт-Петербург

2022 г.

## Задача

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

## Ход работы

1) Создана модель пользователя, имеющая следующие поля:

- Email
- Имя
- Фамилия
- Возраст

```
1  'use strict';
2  const {
3    Model
4  } = require('sequelize');
5  module.exports = (sequelize, DataTypes) => {
6    class User extends Model {
7      /**
8       * Helper method for defining associations.
9       * This method is not a part of Sequelize lifecycle.
10      * The `models/index` file will call this method automatically.
11      */
12      static associate(models) {
13        // define association here
14      }
15    }
16    User.init({
17      email: DataTypes.STRING,
18      firstName: DataTypes.STRING,
19      lastName: DataTypes.STRING,
20      age: DataTypes.INTEGER
21    }, {
22      sequelize,
23      modelName: 'User',
24    });
25    return User;
26  };
```

2) Реализованы запросы CRUD методов с помощью Express и Sequelize:

- Get /users – список всех пользователей
- Get /users/:id – информация о пользователе
- Put /users/:id – изменить информацию о пользователе
- Delete /users/:id – удалить пользователя

GET http://127.0.0.1:3000/u: ●

GET http://127.0.0.1:3000/u: ●

+

...

http://127.0.0.1:3000/users

GET

▼

http://127.0.0.1:3000/users

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

Query Params

	KEY	VALUE
	Key	Value

BodyCookiesHeaders (7)Test Results

Pretty

Raw

Preview

Visualize

JSON ▼

☰

1

[

2

{

3

"id": 1,

4

"email": "test1@example.com",

5

"firstName": "Test1",

6

"lastName": "Test1",

7

"age": 1,

8

"createdAt": "2022-05-17T13:13:53.344Z",

9

"updatedAt": "2022-05-17T13:13:53.344Z"

10

},

11

{

12

"id": 5,

13

"email": "test2@example.com",

14

"firstName": "Test2",

15

"lastName": "Test2",

16

"age": 6,

17

"createdAt": "2022-05-17T13:34:12.749Z",

18

"updatedAt": "2022-05-17T13:37:53.993Z"

19

}

20

]

GET http://127.0.0.1:3000/u: ●

GET http://127.0.0.1:3000/u: ●

+ ...

http://127.0.0.1:3000/users/5

GET



http://127.0.0.1:3000/users/5

Params

Authorization

Headers (6)

Body

Pre-request Script

Tests

Settings

☒ none

☐ form-data

☐ x-www-form-urlencoded

☐ raw

☐ binary

☐ GraphQL

This request does not have a body

Body

Cookies

Headers (7)

Test Results



Pretty

Raw

Preview

Visualize

JSON



```
1 {  
2   "id": 5,  
3   "email": "test2@example.com",  
4   "firstName": "Test2",  
5   "lastName": "Test2",  
6   "age": 6,  
7   "createdAt": "2022-05-17T13:34:12.749Z",  
8   "updatedAt": "2022-05-17T13:37:53.993Z"  
9 }
```

POST http://127.0.0.1:3000/users

GET http://127.0.0.1:3000/users



http://127.0.0.1:3000/users

POST



http://127.0.0.1:3000/users

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON



```
1 {  
2   "email": "test2@example.com",  
3   "firstName": "Test2",  
4   "lastName": "Test2",  
5   "age": "2"  
6 }
```

Body

Cookies

Headers (7)

Test Results



200 OK

Pretty

Raw

Preview

Visualize

JSON



```
1 {  
2   "id": 5,  
3   "email": "test2@example.com",  
4   "firstName": "Test2",  
5   "lastName": "Test2",  
6   "age": "2",  
7   "updatedAt": "2022-05-17T13:34:12.749Z",  
8   "createdAt": "2022-05-17T13:34:12.749Z"
```

PUT http://127.0.0.1:3000/u: GET http://127.0.0.1:3000/u: + ...

http://127.0.0.1:3000/users/5

PUT http://127.0.0.1:3000/users/5

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

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   ... "age": "6"
3 }
```

Body Cookies Headers (7) Test Results 200 C

Pretty Raw Preview Visualize JSON

```
1 {
2   "msg": "user information has been updated"
3 }
```

3) Реализован метод получения пользователя по email.

GET http://127.0.0.1:3000/u: ● GET http://127.0.0.1:3000/u: ● + ...

http://127.0.0.1:3000/users/email/test1@example.com

GET http://127.0.0.1:3000/users/email/test1@example.com

Params Authorization Headers (6) Body Pre-request Script Tests

● none ● form-data ● x-www-form-urlencoded ● raw ● binary ● GraphQL

This request does not have a body

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1 {
2   "id": 1,
3   "email": "test1@example.com",
4   "firstName": "Test1",
5   "lastName": "Test1",
6   "age": 1,
7   "createdAt": "2022-05-17T13:13:53.344Z",
8   "updatedAt": "2022-05-17T13:13:53.344Z"
9 }
```

## Вывод

В ходе работы были освоены фреймворк Express и ORM Sequelize, основы работы с моделями, создание CRUD методов и фильтров.