

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Практическая работа 2

Выполнил:

Иконенко Данил

Группа К33402

Проверил:

Добряков Д. И.

Санкт-Петербург

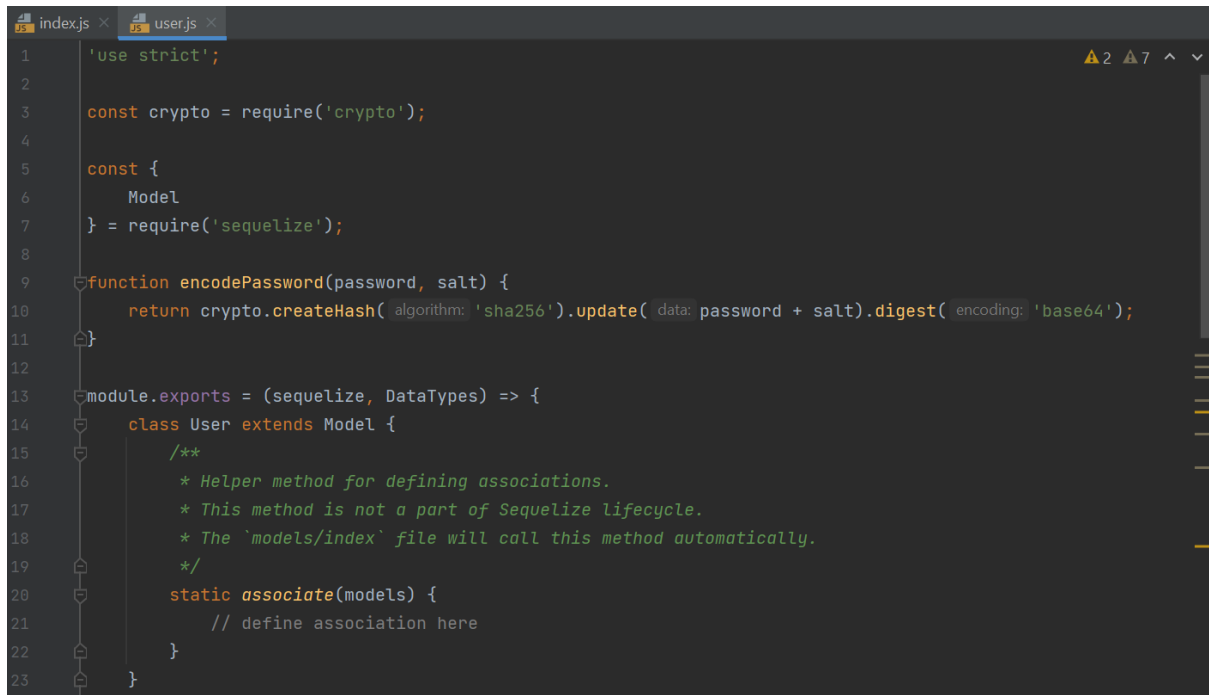
2022 г.

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

Модель пользователя:



```
1  'use strict';
2
3  const crypto = require('crypto');
4
5  const {
6    Model
7  } = require('sequelize');
8
9  function encodePassword(password, salt) {
10     return crypto.createHash('sha256').update(password + salt).digest('base64');
11  }
12
13  module.exports = (sequelize, DataTypes) => {
14     class User extends Model {
15       /**
16        * Helper method for defining associations.
17        * This method is not a part of Sequelize lifecycle.
18        * The 'models/index' file will call this method automatically.
19        */
20       static associate(models) {
21         // define association here
22       }
23     }
24  }
```

```

25     User.init( attributes: {
26         firstName: DataTypes.STRING,
27         lastName: DataTypes.STRING,
28         email: DataTypes.STRING,
29         password: {
30             type: DataTypes.STRING,
31             set(value) {
32                 // Store hash generated from raw password and salt
33                 this.setDataValue('password', encodePassword(value, this.salt));
34             }
35         },
36         salt: DataTypes.STRING
37     }, options: {
38         defaultScope: {
39             // Do not return salt and password
40             attributes: {exclude: ['password', 'salt']},
41         },
42         hooks: {
43             beforeCreate: (instance, options) => {
44                 instance.salt = crypto.randomBytes( size: 16).toString( encoding: 'base64');
45             }
46         },
47         sequelize,
48         modelName: 'User',
49     });
50     return User;
51 };
52

```

Представления:

```

1  const express = require('express');
2  const bodyParser = require('body-parser')
3  const db = require('./models');
4
5  const app = express();
6  const port = 5000;
7
8  // Parse application/x-www-form-urlencoded
9  app.use(bodyParser.urlencoded({ extended: false }))
10 // Parse application/json
11 app.use(bodyParser.json())
12
13
14 app.get('/api/users/:id', async (req, res) => {
15     const user = await db.User.findByPk(req.params.id);
16
17     if (user) {
18         res.send(user.toJSON());
19     }
20
21     res.status( code: 404).send( body: {"detail": "User not found"});
22 }

```

```

24 app.get('/api/users', async (req, res) => {
25     const users = await db.User.findAll();
26     res.send(users);
27 })
28
29 app.post( path: '/api/users', handlers: async (req, res) => {
30     try {
31         const user = await db.User.create(req.body);
32         await user.reload();
33         res.send(user.toJSON());
34     } catch (e) {
35         console.log(e);
36         res.status( code: 400).send( body: {"detail": "Failed to create user"});
37     }
38
40 app.put( path: '/api/users/:id', handlers: async (req, res) => {
41     const user = await db.User.findByPk(req.params.id);
42     if (user) {
43         try {
44             user.update(req.body, {where: {id: req.params.id}});
45             user.reload();
46             res.send(user.toJSON());
47         } catch (e) {
48             console.log(e);
49             res.status( code: 400).send( body: {"detail": "Failed to update user"});
50         }
51     } else {
52         res.status( code: 404).send( body: {"detail": "User not found"});
53     }
54 }

```

```

57 app.delete( path: '/api/users/:id', handlers: async (req, res) => {
58     const user = await db.User.destroy({where: {id: req.params.id}})
59     if (user) {
60         res.send( body: {"detail": "User deleted"});
61     }
62     res.status( code: 404).send( body: {"detail": "User not found"});
63 })
64
65 app.listen(port, hostname: () => {
66     console.log(`Started app on port ${port}`);
67 })

```

Создание:

http://localhost:5000/api/users

Save

POST http://localhost:5000/api/users Send

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

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

```
1 {"firstName": "foo", "lastName": "bar", "email": "foobar@example.com", "password": "12345"}
2
```

Body Cookies Headers (7) Test Results 200 OK 233 ms 386 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 4,
3   "firstName": "foo",
4   "lastName": "bar",
5   "email": "foobar@example.com",
6   "createdAt": "2022-06-07T03:54:25.832Z",
7   "updatedAt": "2022-06-07T03:54:25.832Z"
8 }
```

Чтение:

http://localhost:5000/api/users

Save

GET http://localhost:5000/api/users Send

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

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (7) Test Results 200 OK 301 ms 388 B Save Response

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "firstName": "foo",
5     "lastName": "bar",
6     "email": "foobar@example.com",
7     "createdAt": "2022-06-07T03:23:35.776Z",
8     "updatedAt": "2022-06-07T03:23:35.776Z"
9   }
10 ]
```

http://localhost:5000/api/users/1

Save

GET

http://localhost:5000/api/users/1

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	KEY	VALUE	DESCRIPTION		Bulk Edit
	Key	Value	Description		

Body

Cookies

Headers (7)

Test Results

200 OK 35 ms 386 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

"id": 1,

"firstName": "foo",

"lastName": "bar",

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

"createdAt": "2022-06-07T03:23:35.776Z",

"updatedAt": "2022-06-07T03:23:35.776Z"

Изменение:

http://localhost:5000/api/users/1

PUT

▼

http://localhost:5000/api/users/1

Params

Authorization

Headers (8)

Body

●

Pre-request Script

Tests

Settings

●

 none

●

 form-data

●

 x-www-form-urlencoded

●

 raw

●

 binary

●

 GraphQL

JSC

1

{ "firstName": "new_name", "password": "new_password" }

2

|

Body

Cookies

Headers (7)

Test Results

🔊

Pretty

Raw

Preview

Visualize

JSON ▼

🔧

1

{

2

"id": 1,

3

"firstName": "new_name",

4

"lastName": "bar",

5

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

6

"createdAt": "2022-06-07T03:23:35.776Z",

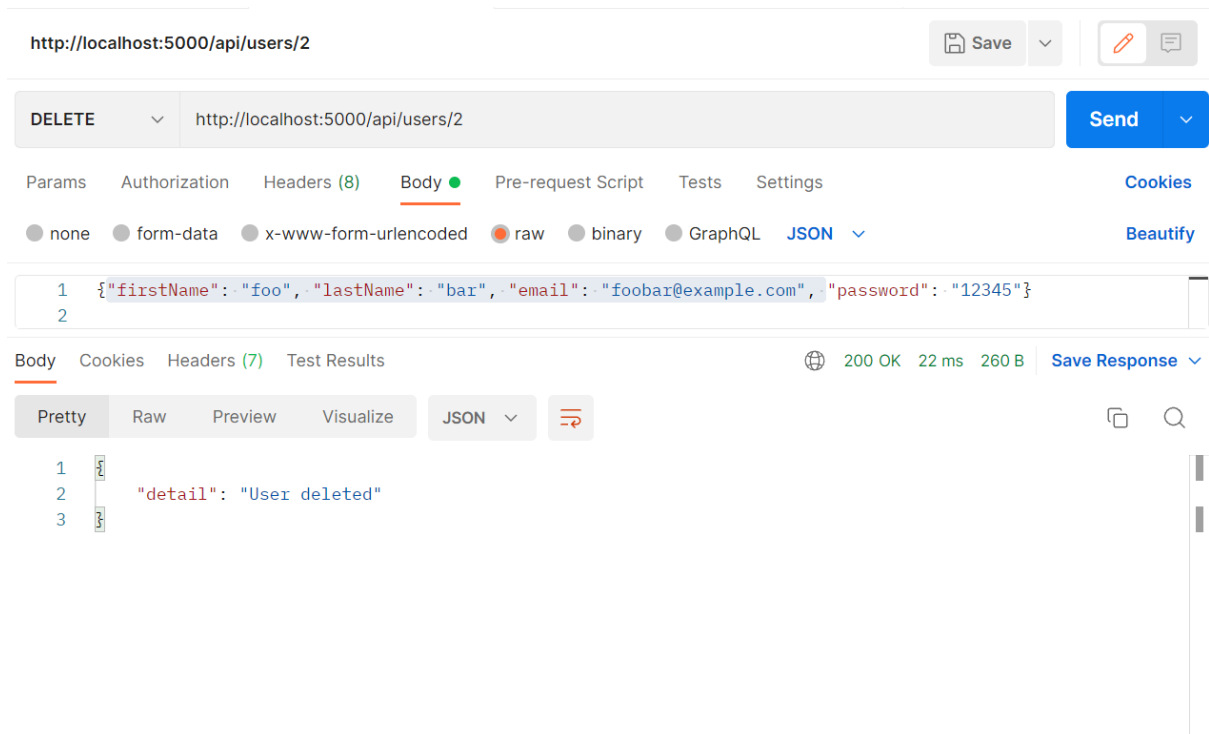
7

"updatedAt": "2022-06-07T03:42:17.048Z"

8

}

Удаление:



Вывод

Я научился работе с моделями средствами ORM Sequelize и созданию API средствами Express