

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №2: Знакомство с ORM Sequelize

Выполнила:
Барышева З. А.
Группа:
К33412

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 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

В модели пользователя будут поля:

1. Имя пользователя
2. Пароль
3. Email
4. Дата рождения

Команда для создания модели:

```
npx sequelize-cli model:generate --name User --attributes  
username:string,password:string,email:string, birthday:dateonly
```

Модель пользователя в файле “models/user.js”:

```
1  'use strict';  
2  const {  
3    Model  
4  } = require('sequelize');  
5  module.exports = (sequelize, DataTypes) => {  
6    class User extends Model {  
7      static associate(models) {  
8      }  
9    }  
10   User.init({  
11     username: DataTypes.STRING,  
12     password: DataTypes.STRING,  
13     email: DataTypes.STRING,  
14     birthday: DataTypes.DATEONLY  
15   }, {  
16     sequelize,  
17     modelName: 'User',  
18   });  
19   return User;  
20 };
```

Реализованы методы:

1. Получение всех пользователей (**GET** /users)
2. Получение пользователя по id (**GET** /users/{id})
3. Удаление пользователя по id (**DELETE** /users/{id})
4. Изменение информации о пользователе по id (**PUT** /users/{id})
5. Создание нового пользователя (**POST** /create)

Вышеуказанные методы, реализованы в файлах *index.js* и *func.js*

Файл *index.js*:

```
16 app.post('/create', async (req, res) => {
17   const body = req.body
18   console.log(body);
19   const r = await create_user(body)
20   res.send(r)
21 })
22
23 app.get('/users', async(req, res) =>{
24   const users = await get_all_users()
25   res.json(users)
26 })
27
28 app.get('/users/:userId', async (req, res) =>{
29   const id = req.params.userId
30   res.json(await get_user_by_id(id))
31 })
32
33 app.delete('/users/:userId', async (req, res) => {
34   const id = req.params.userId
35   const r = await delete_user_by_id(id)
36   res.send(r)
37 })
38
39 app.put('/users/:userId', async (req, res) => {
40   const id = req.params.userId
41   const user_data = req.body
42   const r = await update_user([id, user_data])
43   res.send(r)
44 })
```

Файл *func.js*:

```
3  module.exports = {
4    get_all_users: async function () {
5      const users = await db.User.findAll()
6      return users
7    },
8    get_user_by_id: async function (userId) {
9      try {
10       const user = await db.User.findPk(userId)
11       return user
12     }
13     catch(e) {
14       return null
15     }
16   },
17   create_user: async function (user) {
18     try {
19       console.log(user);
20       await db.User.create(user)
21       return "user has been created"
22     }
23     catch(e){
24       return "fail"
25     }
26   },
27   delete_user_by_id: async function (userId) {
28     console.log(userId);
29     try {
30       await db.User.destroy({where: {id: userId}})
31       return "user has been deleted"
32     }
33     catch(e) {
34       return "fail"
35     }
36   },
37   update_user: async function (userId, user) {
38     console.log(userId);
39     try {
40       await db.User.update(user, {where: {id: userId}})
41       return "user has been updated"
42     }
43     catch(e) {
44       return "fail"
45     }
46   },
47 };
48
```

Проверка работы в postman:

Все пользователи

GET http://localhost:8515/users

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

Body Cookies Headers (7) Test Results 2

Pretty Raw Preview Visualize JSON

```
13  "username": "hennesy",
14  "password": "qwert",
15  "email": null,
16  "birthday": null,
17  "createdAt": "2023-06-17T20:38:49.914Z",
18  "updatedAt": "2023-06-17T21:27:36.023Z"
19  },
20  {
21    "id": 8,
22    "username": "timsort",
23    "password": "quick",
24    "email": null,
25    "birthday": "2010-08-01",
26    "createdAt": "2023-06-17T20:41:47.415Z",
27    "updatedAt": "2023-06-17T21:28:41.450Z"
28  },
29  {
30    "id": 9,
31    "username": "zempljanichka",
32    "password": "zempljanichka",
33    "email": "z@gmail.com",
34    "birthday": "2001-11-12",
35    "createdAt": "2023-06-17T20:42:00.308Z",
36    "updatedAt": "2023-06-17T20:42:00.308Z"
37  }
```

Создание пользователя:

The screenshot shows a REST client interface with a POST request to `http://localhost:8515/create`. The 'Body' tab is selected, showing a JSON payload for creating a user. The response is also shown in the 'Body' tab, indicating the user was created successfully.

POST `http://localhost:8515/create`

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

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL

```
1 {
2   "username": "witcher",
3   "password": "zirael",
4   "email": "zirael@gmail.com",
5   "birthday": "1258-11-12"
6 }
```

Body Cookies Headers (7) Test Results 20

Pretty Raw Preview Visualize HTML

```
1 user has been created
```

Получение одного пользователя:

The screenshot shows a REST client interface with a GET request to `http://localhost:8515/users/10`. The 'Body' tab is selected, showing a JSON response for a user with ID 10. The response includes fields for id, username, password, email, birthday, createdAt, and updatedAt.

GET `http://localhost:8515/users/10`


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

Body Cookies Headers (7) Test Results 2


Pretty Raw Preview Visualize JSON



```
1 {
2   "id": 10,
3   "username": "witcher",
4   "password": "zirael",
5   "email": "zirael@gmail.com",
6   "birthday": "1258-11-12",
7   "createdAt": "2023-06-17T21:30:38.157Z",
8   "updatedAt": "2023-06-17T21:30:38.157Z"
9 }
```

Удаление пользователя:

DELETE  http://localhost:8515/users/7


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

Body Cookies Headers (7) Test Results  200

Pretty Raw Preview Visualize HTML  

1 user has been deleted


Изменение пользователя:



PUT  http://localhost:8515/users/8

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

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL

1 { "password": "duckduck" }

Body Cookies Headers (7) Test Results  200

Pretty Raw Preview Visualize HTML  

1 user has been updated

После всех манипуляций список пользователей:

```
GET http://localhost:8515/users

Body
Cookies
Headers (7)
Test Results

Pretty Raw Preview Visualize JSON

[
  {
    "id": 4,
    "username": "timofeevnik",
    "password": null,
    "email": null,
    "birthday": null,
    "createdAt": "2023-06-17T20:34:35.692Z",
    "updatedAt": "2023-06-17T21:06:34.097Z"
  },
  {
    "id": 8,
    "username": "timsort",
    "password": "duckduck",
    "email": null,
    "birthday": "2010-08-01",
    "createdAt": "2023-06-17T20:41:47.415Z",
    "updatedAt": "2023-06-17T21:34:25.957Z"
  },
  {
    "id": 9,
    "username": "zemljanichka",
    "password": "zemljanichka",
    "email": "z@gmail.com",
    "birthday": "2001-11-12",
    "createdAt": "2023-06-17T20:42:00.308Z",
    "updatedAt": "2023-06-17T20:42:00.308Z"
  },
  {
    "id": 10,
    "username": "witcher",
    "password": "zirael",
    "email": "zirael@gmail.com",
    "birthday": "1258-11-12",
    "createdAt": "2023-06-17T21:30:38.157Z",
    "updatedAt": "2023-06-17T21:30:38.157Z"
  }
]
```

Вывод

В ходе работы была создана модель пользователя, реализованы методы из набора CRUD для работы с пользователями. Работоспособность всех реализованных методов была проверена с помощью postman.