# САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №2: Знакомство с ORM Sequelize

Выполнила: Барышева З. А. Группа: K33412

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":

```
'use strict';
    const {
    Model
     } = require('sequelize');
    module.exports = (sequelize, DataTypes) => {
      class User extends Model {
        static associate(models) {
      User.init({
       username: DataTypes.STRING,
11
        password: DataTypes.STRING,
12
        email: DataTypes.STRING,
13
        birthday: DataTypes.DATEONLY
14
15
         sequelize,
        modelName: 'User',
17
       });
       return User;
     };
```

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*:

```
app.post('/create', async (req, res) => {
         const body = req.body
17
         console.log(body);
         const r = await create user(body)
         res.send(r)
21
     })
22
     app.get('/users', async(req, res) =>{
23
         const users = await get all users()
24
25
         res.json(users)
     })
     app.get('/users/:userId', async (req, res) =>{
28
         const id = req.params.userId
29
         res.json(await get user by id(id))
     })
32
     app.delete('/users/:userId', async (req, res) => {
         const id = req.params.userId
         const r = await delete user by id(id)
         res.send(r)
     })
     app.put('/users/:userId', async (req, res) => {
         const id = req.params.userId
         const user data = req.body
41
         const r = await update user(id, user data)
42
         res.send(r)
```

## Файл func.js:

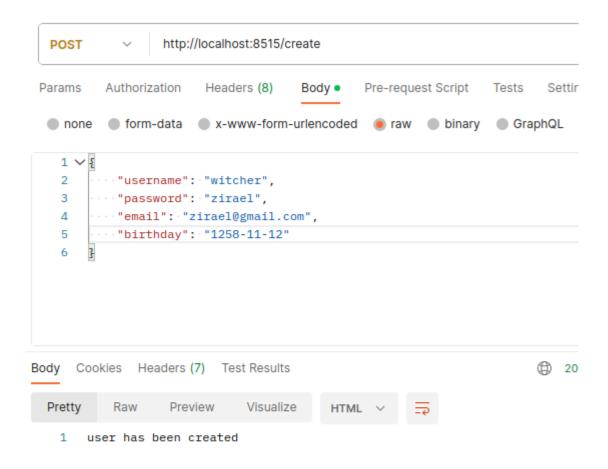
```
module.exports = {
         get all users: async function () {
             const users = await db.User.findAll()
             return users
         },
         get user by id: async function (userId) {
                 const user = await db.User.findByPk(userId)
                 return user
12
13
             catch(e) {
                 return null
15
         },
         create user: async function (user) {
17
             try {
                 console.log(user);
                 await db.User.create(user)
                 return "user has been created"
21
22
23
             catch(e){
                 return "fail"
25
         delete user by id: async function (userId) {
             console.log(userId);
29
             try {
                 await db.User.destroy({where: {id: userId}})
                 return "user has been deleted"
32
             catch(e) {
                 return "fail"
         update user: async function (userId, user) {
             console.log(userId);
             try {
                 await db.User.update(user, {where: {id: userId}})
                 return "user has been updated"
42
             catch(e) {
                 return "fail"
       }:
```

## Проверка работы в postman:

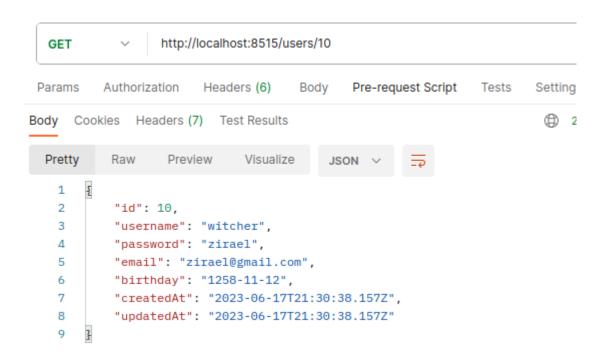
#### Все пользователи

```
http://localhost:8515/users
  GET
          Authorization
                         Headers (6)
                                       Body
 Params
                                               Pre-request Script
                                                                  Tests
                                                                           Setting
Body Cookies Headers (7) Test Results
                                                                            2
  Pretty
            Raw
                    Preview
                               Visualize
   13
                "username": "hennesy",
   14
                "password": "qwert",
                "email": null,
   15
                "birthday": null,
   16
                "createdAt": "2023-06-17T20:38:49.914Z",
   17
   18
                "updatedAt": "2023-06-17T21:27:36.023Z"
   19
            },
   20
                "id": 8,
   21
                "username": "timsort",
   22
   23
                "password": "quick",
   24
                "email": null,
                "birthday": "2010-08-01",
   25
                "createdAt": "2023-06-17T20:41:47.415Z",
   26
                "updatedAt": "2023-06-17T21:28:41.450Z"
   27
   28
            },
   29
                "id": 9,
   30
   31
                "username": "zemljanichka",
                "password": "zemljanichka",
   32
                "email": "z@gmail.com",
   33
   34
                "birthday": "2001-11-12",
                "createdAt": "2023-06-17T20:42:00.308Z",
   35
                "updatedAt": "2023-06-17T20:42:00.308Z"
   36
```

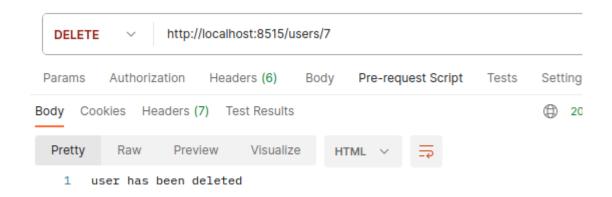
#### Создание пользователя:



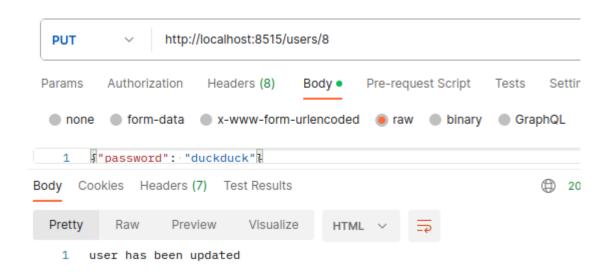
#### Получение одного пользователя:



#### Удаление пользователя:



### Изменение пользователя:



После всех манипуляций список пользователей:

```
GET
                  http://localhost:8515/users
         Authorization Headers (6)
                                      Body Pre-request Script
Body Cookies Headers (7) Test Results
           Raw
                              Visualize
 Pretty
                   Preview
   2
   3
               "id": 4,
               "username": "timofeevnik",
   4
               "password": null,
               "email": null,
   7
               "birthday": null,
   8
               "createdAt": "2023-06-17T20:34:35.692Z",
   9
               "updatedAt": "2023-06-17T21:06:34.097Z"
  10
  11
               "id": 8,
  12
               "username": "timsort",
  13
               "password": "duckduck",
  14
  15
               "email": null,
  16
               "birthday": "2010-08-01",
               "createdAt": "2023-06-17T20:41:47.415Z",
  17
                "updatedAt": "2023-06-17T21:34:25.957Z"
  18
  19
  28
  21
               "id": 9,
               "username": "zemljanichka",
               "password": "zemljanichka",
  24
               "email": "z@gmail.com",
  25
               "birthday": "2001-11-12",
               "createdAt": "2023-06-17T20:42:00.308Z",
  26
  27
                "updatedAt": "2023-06-17T20:42:00.308Z"
  28
           ζ,
  29
               "id": 10,
               "username": "witcher",
  31
               "password": "zirael",
  32
               "email": "zirael@gmail.com",
  33
               "birthday": "1258-11-12",
  34
               "createdAt": "2023-06-17T21:30:38.157Z",
  35
               "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.