

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ  
УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №2

Выполнил:  
Кулёмин С. А.  
Группа К33401

Проверил:  
Добряков Д. И.

Санкт-Петербург  
2022 г.

## Задание

- 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

## Модель пользователя

```
'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class User extends Model {
    /**
     * Helper method for defining associations.
     * This method is not a part of Sequelize lifecycle.
     * The `models/index` file will call this method automatically.
     */
    static associate(models) {
      // define association here
    }
  }
  User.init({
    attributes: {
      firstName: DataTypes.STRING,
      lastName: DataTypes.STRING,
      username: DataTypes.STRING,
      password: DataTypes.STRING
    },
    options: {
      sequelize,
      modelName: 'User',
    }
  });
  return User;
};
```

## Работа с sequelize

npx sequelize-cli init

npx sequelize-cli model:generate --name User --attributes

firstName:string,lastName:string,username:string,password:string

npx sequelize-cli db:migrate

## Приложение

```
const express = require('express')
const db = require('./models')
const bodyParser = require("body-parser");

const app = express()
const port = 3000

app.use(bodyParser.json())

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})

app.get('/', (req, res) => {
  res.send('Hello World!')
})
```

## CRUD-методы

Create

```
//CREATE
app.post('/add', async (req, res) => {
  const user = await db.User.create(req.body)
  res.send(user.toJSON())
})
```

## Read

```
//READ
app.get('/get', async (req, res) => {
  const users = await db.User.findAll()
  res.send(users)
})

app.get('/users/id/:id', async (req, res) => {
  const user = await db.User.findById(req.params.id)
  if (user) {
    res.send(user.toJSON())
  } else {
    res.status(404).send({'msg': 'user not found'})
  }
})

app.get('/users/username/:username', async (req, res) => {
  const user = await db.User.findOne({
    where: {
      username: req.params.username
    }
  })
  if (user) {
    res.send(user.toJSON())
  } else {
    res.status(404).send({'msg': 'user not found'})
  }
})
})
```

## Update

```
//UPDATE
app.put('update/:id', async (req, res) => {
  const user = await db.User.findById(req.params.id)
  if (user) {
    await db.User.update(req.body, {
      where: {
        id: req.params.id
      }
    })
  }
})
})
```

## Delete

```
//DELETE
app.delete('/delete/:id', async (req, res) => {
  const user = await db.User.findByIdPk(req.params.id)
  if (user) {
    user.destroy();
    res.sendStatus( statusCode: 200 ).send({ 'msg': 'user deleted' })
  }
  res.status(404).send({ 'msg': 'user not found' })
})
```

## Проверка работы

### Добавление пользователя

POST

localhost:3000/add

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

```
1 {
2   ... "firstName": "Ivan",
3   ... "lastName": "Ivanov",
4   ... "username": "ivanivanov",
5   ... "password": "ivan123456"
6 }
```

Body

Cookies

Headers (6)

Test Results

Status: 200 OK

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "id": 6,
3   "firstName": "Ivan",
4   "lastName": "Ivanov",
5   "username": "ivanivanov",
6   "password": "ivan123456",
7   "updatedAt": "2022-04-16T16:06:50.056Z",
8   "createdAt": "2022-04-16T16:06:50.056Z"
9 }
```

## Чтение списка пользователей

GET localhost:3000/get

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 (6) Test Results

Pretty Raw Preview Visualize JSON

```
2  {
3    "id": 4,
4    "firstName": "S",
5    "lastName": "K",
6    "username": "S",
7    "password": "P",
8    "createdAt": "2022-04-16T15:26:09.671Z",
9    "updatedAt": "2022-04-16T15:26:09.671Z"
10 },
11  {
12    "id": 5,
13    "firstName": "Test",
14    "lastName": "Test",
15    "username": "test",
16    "password": "test",
17    "createdAt": "2022-04-16T15:26:57.732Z",
18    "updatedAt": "2022-04-16T15:26:57.732Z"
19  },
20  {
21    "id": 6,
22    "firstName": "Ivan",
23    "lastName": "Ivanov",
24    "username": "ivanivanov",
25    "password": "ivan123456",
26    "createdAt": "2022-04-16T16:06:50.056Z",
27    "updatedAt": "2022-04-16T16:06:50.056Z"
```

GET

localhost:3000/users/id/4

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 Cookies Headers (6) Test Results

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "id": 4,
3   "firstName": "S",
4   "lastName": "K",
5   "username": "S",
6   "password": "P",
7   "createdAt": "2022-04-16T15:26:09.671Z",
8   "updatedAt": "2022-04-16T15:26:09.671Z"
9 }
```

GET

localhost:3000/users/username/ivanivanov

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 Cookies Headers (6) Test Results

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "id": 6,
3   "firstName": "Ivan",
4   "lastName": "Ivanov",
5   "username": "ivanivanov",
6   "password": "ivan123456",
7   "createdAt": "2022-04-16T16:06:50.056Z",
8   "updatedAt": "2022-04-16T16:06:50.056Z"
9 }
```



## Обновление пользователя

PUT

localhost:3000/update/4

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

```
1 {  
2   ... "username": "Slay",  
3   ... "password": "PpppWwWw"  
4 }
```

Body

Cookies

Headers (6)

Test Results

Pretty

Raw

Preview

Visualize

JSON

```
1 {  
2   "id": 4,  
3   "firstName": "Someove",  
4   "lastName": "Kkkk",  
5   "username": "Slay",  
6   "password": "PpppWwWw",  
7   "createdAt": "2022-04-16T15:26:09.671Z",  
8   "updatedAt": "2022-04-16T16:12:56.539Z"  
9 }
```

## Удаление пользователя

The screenshot displays a REST client interface with two tabs. The top tab is for a **DELETE** request to `localhost:3000/delete/4`. The bottom tab is for a **GET** request to `localhost:3000/get`. The **Body** tab is selected in the bottom tab, showing a JSON body with the following content:

```
1 {
2   ... "username": "Slay",
3   ... "password": "PpppWwWw"
4 }
```

Below the body tab, there are tabs for **Body**, **Cookies**, **Headers (6)**, and **Test Results**. The **Body** tab is selected, showing a JSON response in the **Pretty** format:

```
1 [
2   {
3     "id": 5,
4     "firstName": "Test",
5     "lastName": "Test",
6     "username": "test",
7     "password": "test",
8     "createdAt": "2022-04-16T15:26:57.732Z",
9     "updatedAt": "2022-04-16T15:26:57.732Z"
10  },
11  {
12    "id": 6,
13    "firstName": "Test",
14    "lastName": "Test",
15    "username": "test",
16    "password": "test",
17    "createdAt": "2022-04-16T15:26:57.732Z",
18    "updatedAt": "2022-04-16T15:26:57.732Z"
19  }
20 ]
```

## Вывод

В ходе выполнения работы я ознакомился с основами express и sequelize. Кроме того, был создан скрипт для управления базой данных (создания, удаления, обновления записей).