

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №2: Знакомство с ORM Sequelize

Выполнил:

Траоре Мамуду

Группа К33412

Проверил:

Добряков Д. И.

Санкт-Петербург

2023 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

1. Модель пользователя состоит из 6 полей:

- First name
- Last name
- email
- Country
- City
- Age

```
models > JS user.js > ...
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      firstName: DataTypes.STRING,
18      lastName: DataTypes.STRING,
19      email: DataTypes.STRING,
20      country: DataTypes.STRING,
21      city: DataTypes.STRING,
22      age: DataTypes.INTEGER
23    }, {
24      sequelize,
25      modelName: 'User',
26    });
27    return User;
28  };
```

2. Набор из CRUD-методов для работы с пользователями средствами Express + Sequelize

```
1  const express = require('express')
2  const db = require('./models')
3
4  const app = express()
5  const port = 3000
6  const bodyParser = require('body-parser');
7  app.use(bodyParser.json());
8
9
10 app.post('/users', async (req, res) => {
11   const user = await db.User.create(req.body)
12   res.send(user)
13 })
14
15 app.get('/', async (req, res) => {
16   const users = await db.User.findAll()
17   res.send(users)
18 })
19
20 app.get('/users/email', async (req, res) => {
21   const user = await db.User.findOne({ where: { email: req.query.email } })
22   if (user) {
23     res.send(user)
24   } else {
25     res.send({ "status": "error" })
26   }
27 })
28
29 app.get('/users/id', async (req, res) => {
30   const user = await db.User.findByPk(req.query.id)
31   if (user) {
32     res.send(user)
33   } else {
34     res.send({ "status": "error" })
35   }
36 })
```

```

app.delete('/users/:id', async (req, res) => {
  const user = await db.User.destroy({ where: { id: req.params.id } })
  if (user) {
    res.send({"status": "user deleted"})
  } else {
    res.send({"status": "error"})
  }
})

app.put('/users/:id', async (req, res) => {
  const num = await db.User.update(req.body, { where: { id: req.params.id } })
  if (num == 1) {
    res.send({"status": "user updated"})
  } else {
    res.send({"status": "error"})
  }
})

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

```

CREATE (POST):

The screenshot shows a REST client interface with the following details:

- URL:** http://localhost:3000/users
- Method:** POST
- Body (JSON):**

```

{
  "firstName": "Moussa",
  "lastName": "Camara",
  "email": "moussa@example.com",
  "age": 24,
  "country": "Mali",
  "city": "Bamako"
}

```
- Status:** 200 OK
- Time:** 74 ms
- Size:** 435 B
- Response (JSON):**

```

{
  "id": 10,
  "firstName": "Moussa",
  "lastName": "Camara",
  "email": "moussa@example.com",
  "age": 24,
  "country": "Mali",
  "city": "Bamako",
  "updatedAt": "2023-03-12T12:59:11.978Z",
  "createdAt": "2023-03-12T12:59:11.978Z"
}

```

READ (GET):

GET http://localhost:3000

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

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

Status: 200 OK Time: 10 ms Size: 837 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 10,
4     "firstName": "Moussa",
5     "lastName": "Camara",
6     "email": "moussa@example.com",
7     "country": "Mali",
8     "city": "Bamako",
9     "age": 24,
10    "createdAt": "2023-03-12T12:59:11.978Z",
11    "updatedAt": "2023-03-12T12:59:11.978Z"
12  },
13  {
14    "id": 11,
15    "firstName": "Dambou",
16    "lastName": "Keita",
17    "email": "dambou@example.com",
18    "country": "Guinea",
19    "city": "Kankan",
20    "age": 26,
21    "createdAt": "2023-03-12T13:16:43.552Z",
22    "updatedAt": "2023-03-12T13:16:43.552Z"
23  },
24 }
```

UPDATE (PUT):

PUT http://localhost:3000/users/11

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

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

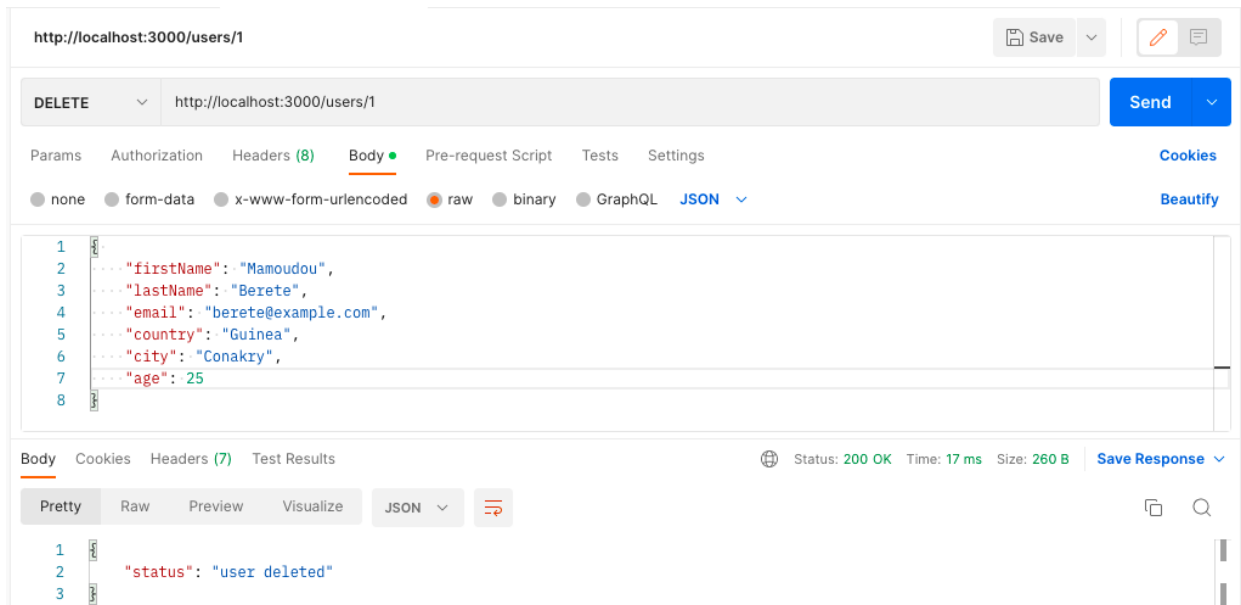
Status: 200 OK Time: 20 ms Size: 260 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "firstName": "Nathan",
3   "lastName": "Bounkouta",
4   "email": "nathan@example.com",
5   "country": "Congo",
6   "city": "Kankan",
7   "age": 21
8 }
```

```
1 {
2   "status": "user updated"
3 }
```

DELETE:



The screenshot shows a Postman interface for a DELETE request to `http://localhost:3000/users/1`. The request body is a JSON object representing a user. The response is a JSON object indicating the user was deleted.

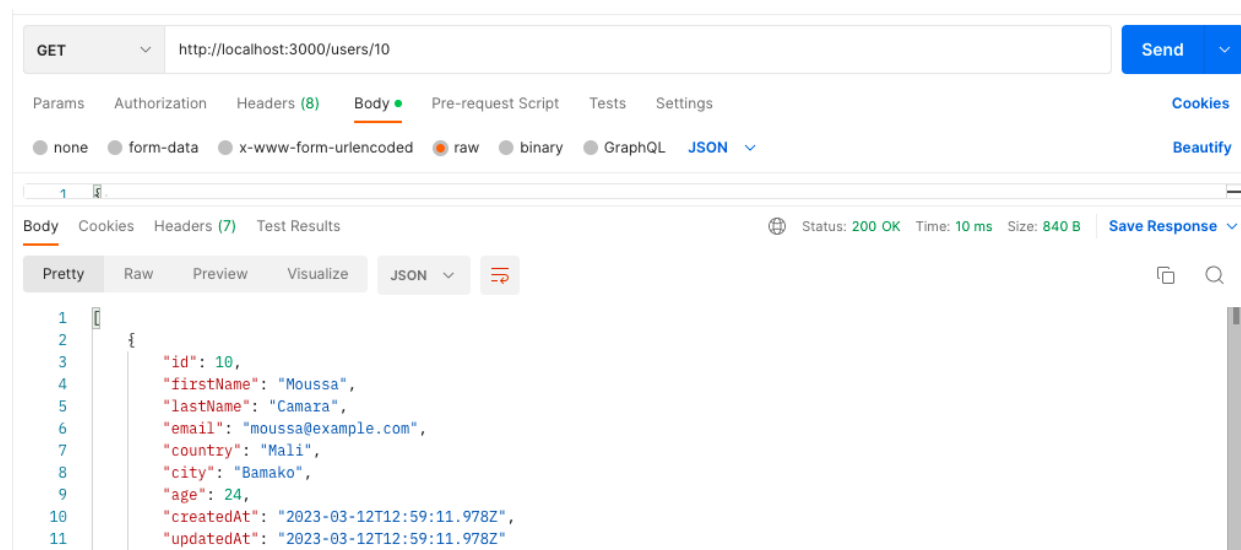
Request:

```
1 {
2   "firstName": "Mamoudou",
3   "lastName": "Berete",
4   "email": "berete@example.com",
5   "country": "Guinea",
6   "city": "Conakry",
7   "age": 25
8 }
```

Response:

```
1 {
2   "status": "user deleted"
3 }
```

3. Запрос для получения пользователя по id



The screenshot shows a Postman interface for a GET request to `http://localhost:3000/users/10`. The response is a JSON object representing a user with id 10.

Request:

```
1 GET http://localhost:3000/users/10
```

Response:

```
1 {
2   "id": 10,
3   "firstName": "Moussa",
4   "lastName": "Camara",
5   "email": "moussa@example.com",
6   "country": "Mali",
7   "city": "Bamako",
8   "age": 24,
9   "createdAt": "2023-03-12T12:59:11.978Z",
10  "updatedAt": "2023-03-12T12:59:11.978Z"
11 }
```

Вывод

В данном домашнем задании удалось освоить CRUD-методы для работы с пользователями средствами Express + Sequelize. Были проведены запросы по добавлению, удалению и обновлению информации о пользователе с помощью Postman.