САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №2: Знакомство с ORM Sequelize

Выполнил:

Омар Сизей

Группа К33412

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

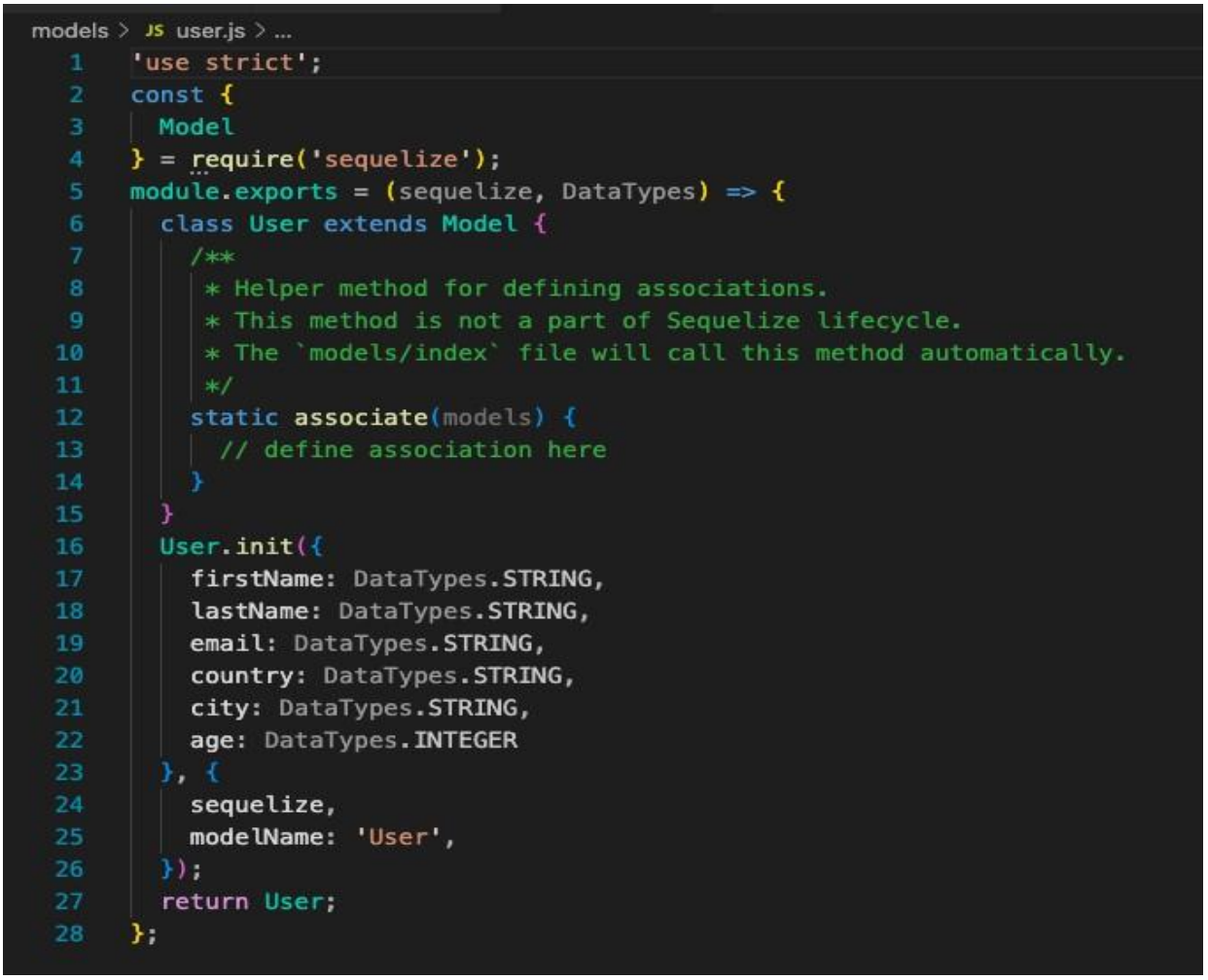
Задача

- 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



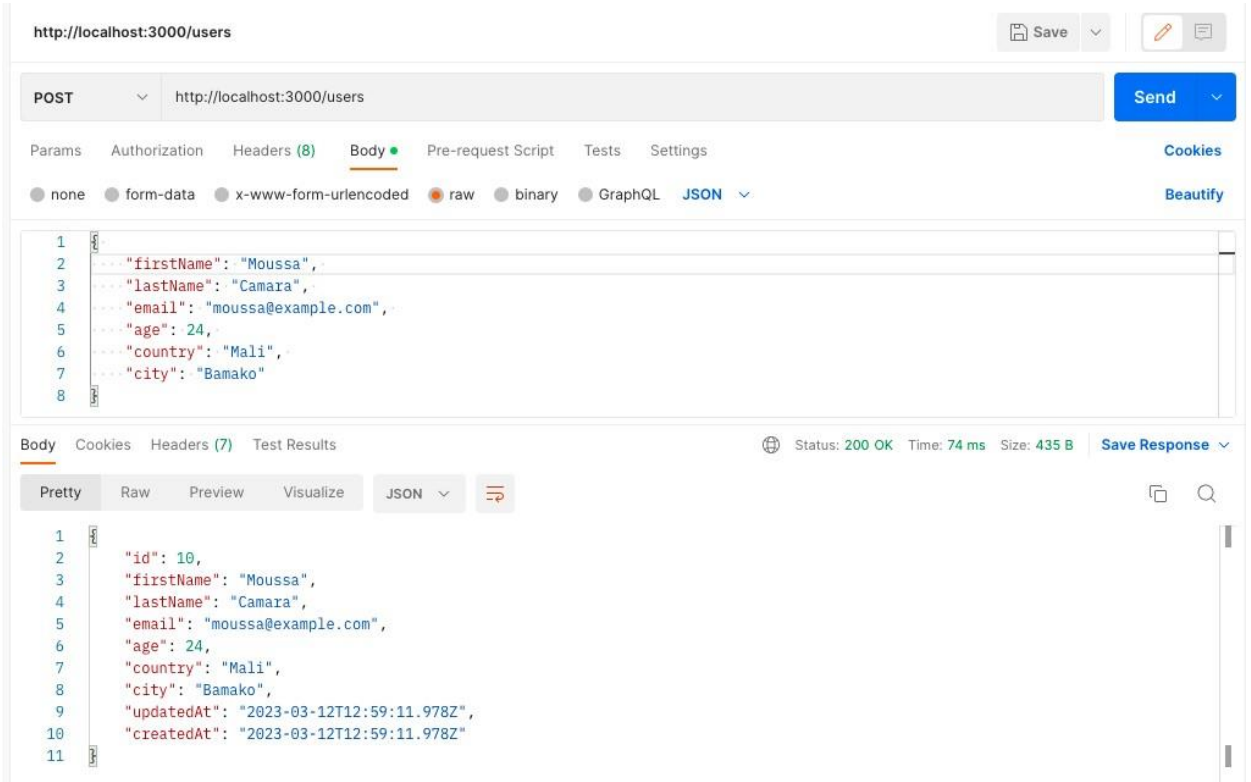
```
models > JS user.js > ...
1   'use strict';
2   const {
3     Model
4   } = require('sequelize');
5   module.exports = (sequelize, DataTypes) => {
6     class User extends Model {
7       /**
8        * Helper method for defining associations.
9        * This method is not a part of Sequelize lifecycle.
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**

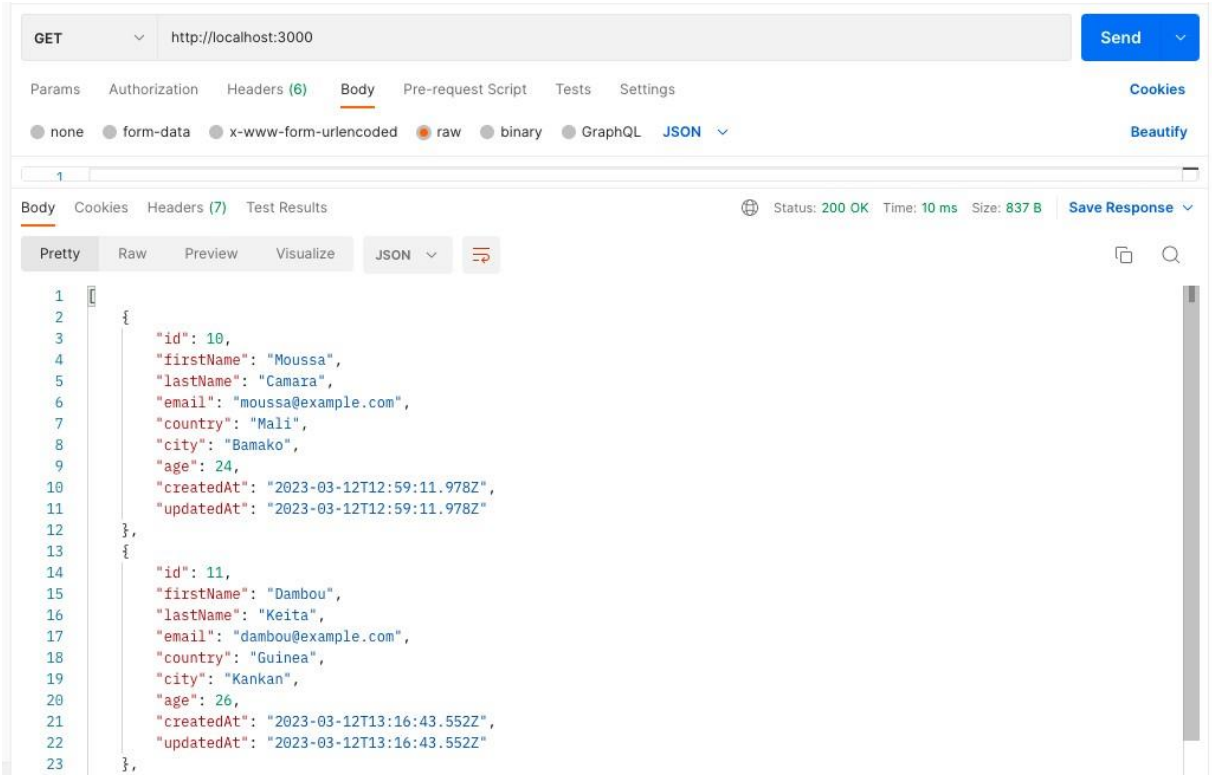
```
const express = require('express')
const db = require('./models')

const app = express()
const port = 3000
const bodyParser = require('body-parser');
app.use(bodyParser.json());


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

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

app.get('/users/email', async (req, res) => {
  const user = await db.User.findOne({ where: { email: req.query.email } })
  if (user) {
    res.send(user)
  } else {
    res.send({"status": "error"})
  }
})

app.get('/users/id', async (req, res) => {
  const user = await db.User.findByPk(req.query.id)
  if (user) {
    res.send(user)
  } else {
    res.send({"status": "error"})
  }
```

```
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}`)
})
```

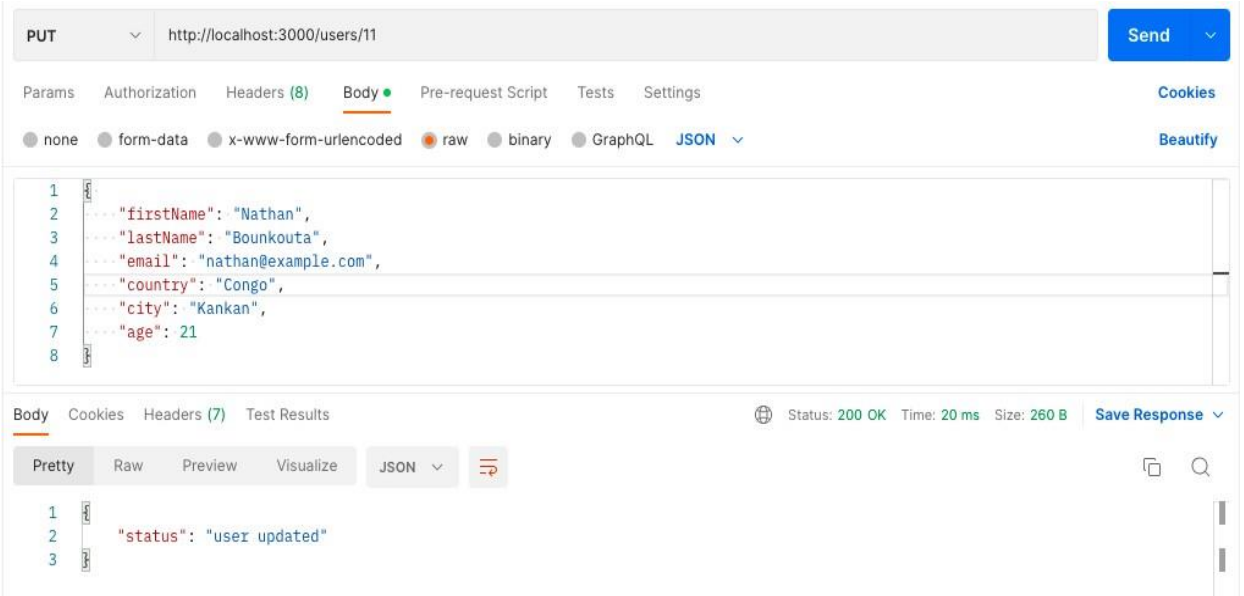
CREATE (POST):



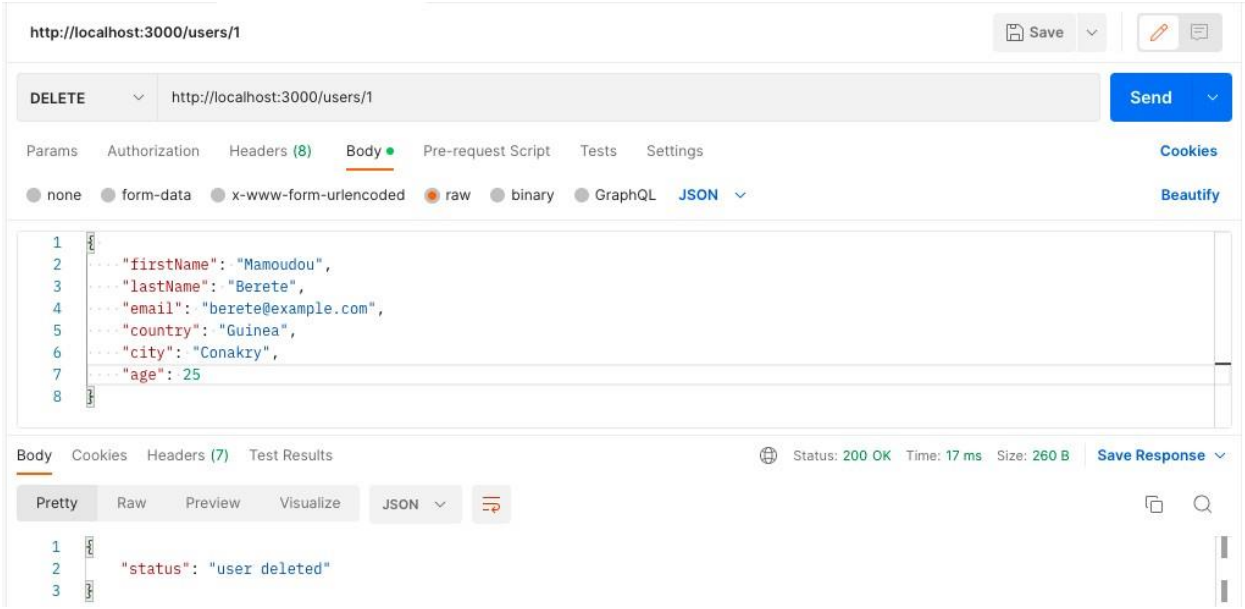
READ (GET):


GET
http://localhost:3000
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 10 ms
Size: 837 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": 10,
"firstName": "Moussa",
"lastName": "Camara",
"email": "moussa@example.com",
"country": "Mali",
"city": "Bamako",
"age": 24,
"createdAt": "2023-03-12T12:59:11.978Z",
"updatedAt": "2023-03-12T12:59:11.978Z"
},
{
"id": 11,
"firstName": "Dambou",
"lastName": "Keita",
"email": "dambou@example.com",
"country": "Guinea",
"city": "Kankan",
"age": 26,
"createdAt": "2023-03-12T13:16:43.552Z",
"updatedAt": "2023-03-12T13:16:43.552Z"
},

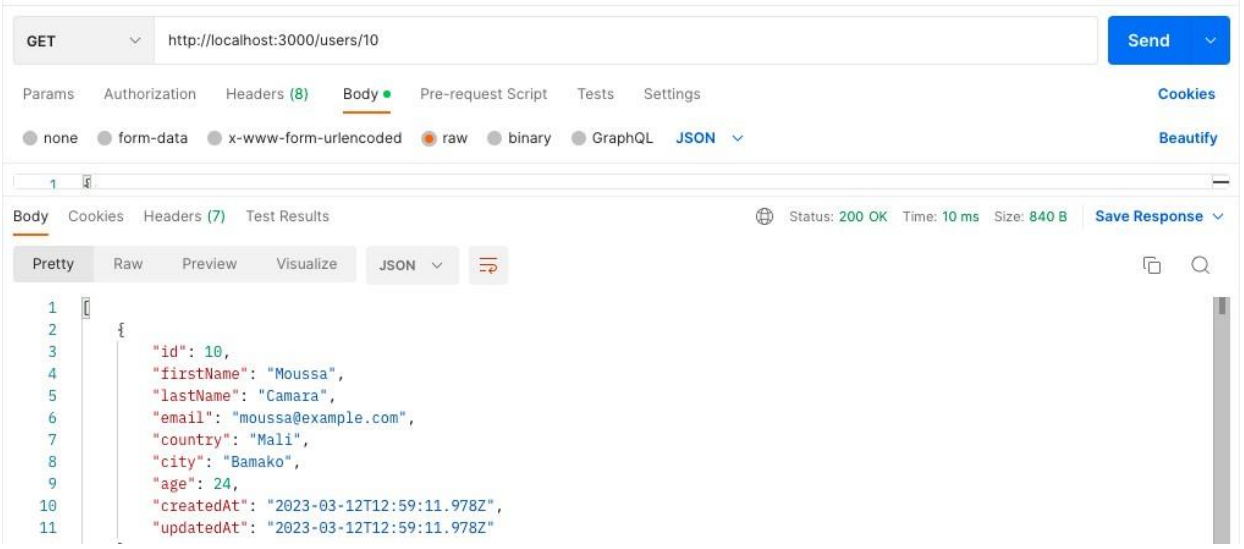
UPDATE (PUT):


PUT
http://localhost:3000/users/11
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"firstName": "Nathan",
"lastName": "Bounkouta",
"email": "nathan@example.com",
"country": "Congo",
"city": "Kankan",
"age": 21
}
Body
Cookies
Headers (7)
Test Results
Status: 200 OK
Time: 20 ms
Size: 260 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"status": "user updated"
}

DELETE:



3. Запрос для получения пользователя по id



Вывод

В данном домашнем задании я понял проектирование и реализацию пользовательских моделей с использованием Express и Sequelize. Я научился создавать методы CRUD для управления пользовательскими данными, включая создание, чтение, обновление и удаление пользовательских записей. Дополнительно я научился писать запросы на получение пользовательских данных по id о пользователе с помощью Postman.. Эти знания пригодятся в будущих проектах, где мне нужно будет работать с пользовательскими данными и реализовывать аутентификацию и авторизацию пользователей.