САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Практическая работа №2

Выполнил:

Таначев Егор

Группа K33412

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

Для начала работы установим все необходимые пакеты, в частности: *express*, *sequilize*, *sequilize-cli* и *sqlite3*. Модель пользователя будет состоять из 5 атрибутов: *firstName* – имя, *lastName* – фамилия, *email* – электронная почта, *password* – пароль, *qualification* – квалификация (Front-end, Back-end, DevOps, Full-Stack и т.д.).

Для создания модели пользователя используем команду: *npx sequelize-cli model:generate --name User --attributes firstName:string, lastName:string, email:string, password:string, qualification:string*.

Созданная модель представлена в файле user.js, как показано на Рисунке 1.

```
'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
    }
  }
```

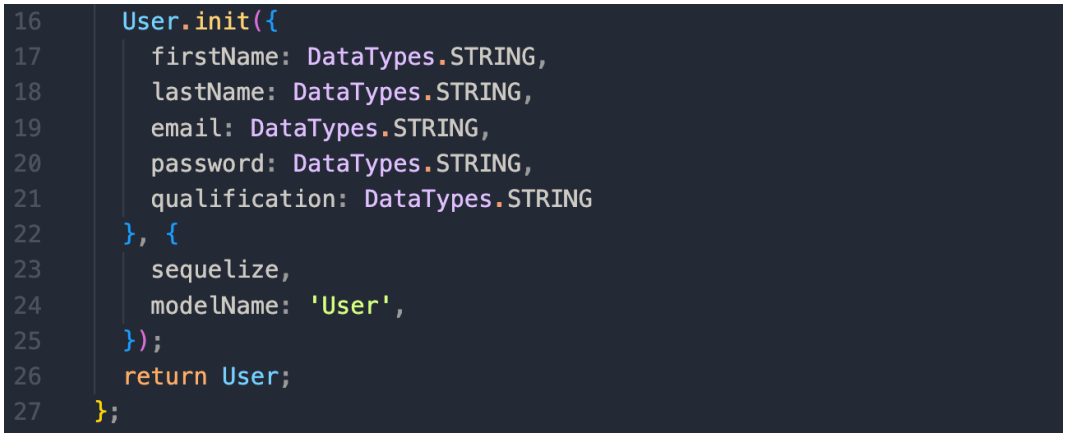


Рисунок 1 – Модель пользователя в user.js

Модель пользователя выглядит следующим образом, как показано на Таблице 1.

firstName	String	Имя
lastName	String	Фамилия
Email	String	Электронная почта
password	String	Пароль
qualification	String	Квалификация

Таблица 1 – Модель пользователя

Напишем CRUD-методы для получения, изменения информации о пользователях и удаления, как показано на Рисунке 2.

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

app.put('/user/:id', async (req, res) => {
    const user = await db.User.update(req.body, {where: {id: req.params.id}})
    const iduser = req.params.id
    return res.send(`Информация о пользователе ${iduser} была обновлена`)
})

app.delete('/user/:id', async (req, res) => {
    const user = await db.User.destroy({where: {id: req.params.id}})
    const iduser = req.params.id
    return res.send(`Пользователь под номером ${iduser} удален из базы данных`)
})

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

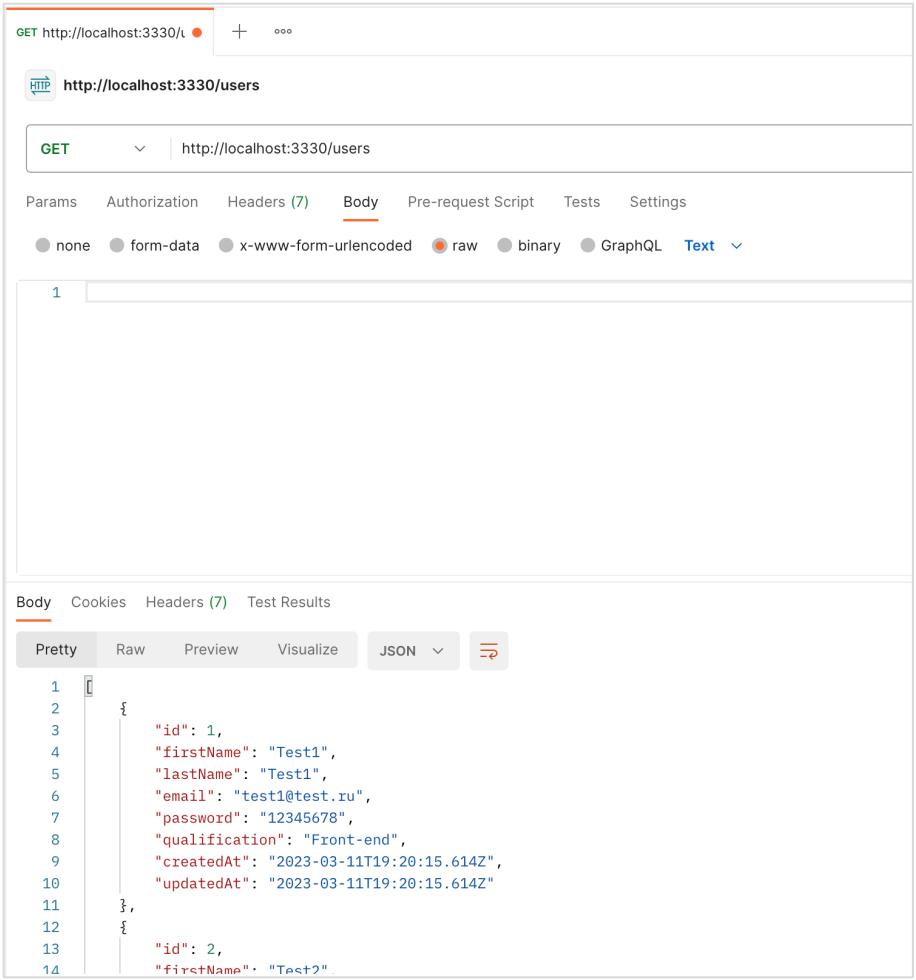
Рисунок 2 – CRUD-методы в index.js

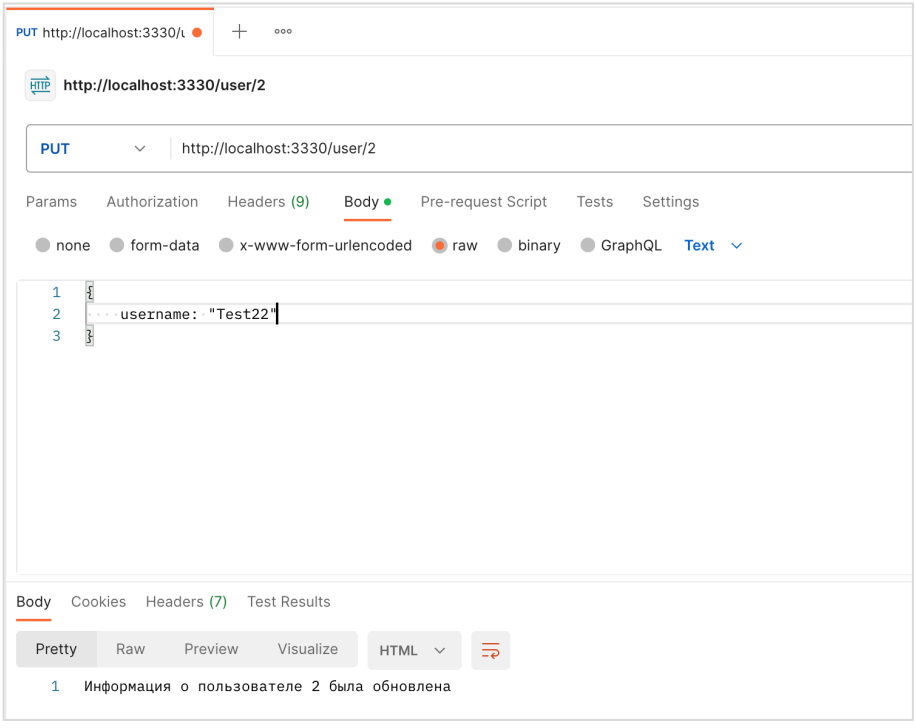
Для создания пользователей будем использовать файл creatUser.js, как показано на Рисунке 4.

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

async function main() {
  await db.User.create({
    firstName: 'Test1',
    lastName: 'Test1',
    email: 'test1@test.ru',
    password: '12345678',
    qualification: 'Front-end'
  })
}

main()
```

Рисунок 3 – Создание пользователя в creatUser.js

Теперь протестируем CRUD-методы с помощью POSTMAN, как показано на Рисунка 4–6.


GET http://localhost:3330/
http://localhost:3330/users
GET
http://localhost:3330/users
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
Text
Body
Cookies
Headers (7)
Test Results
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": 1,
"firstName": "Test1",
"lastName": "Test1",
"email": "test1@test.ru",
"password": "12345678",
"qualification": "Front-end",
"createdAt": "2023-03-11T19:20:15.614Z",
"updatedAt": "2023-03-11T19:20:15.614Z"
},
{
"id": 2,


PUT http://localhost:3330/
http://localhost:3330/user/2
PUT
http://localhost:3330/user/2
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
Text
{
username: "Test22"
}
Body
Cookies
Headers (7)
Test Results
Pretty
Raw
Preview
Visualize
HTML
Информация о пользователе 2 была обновлена

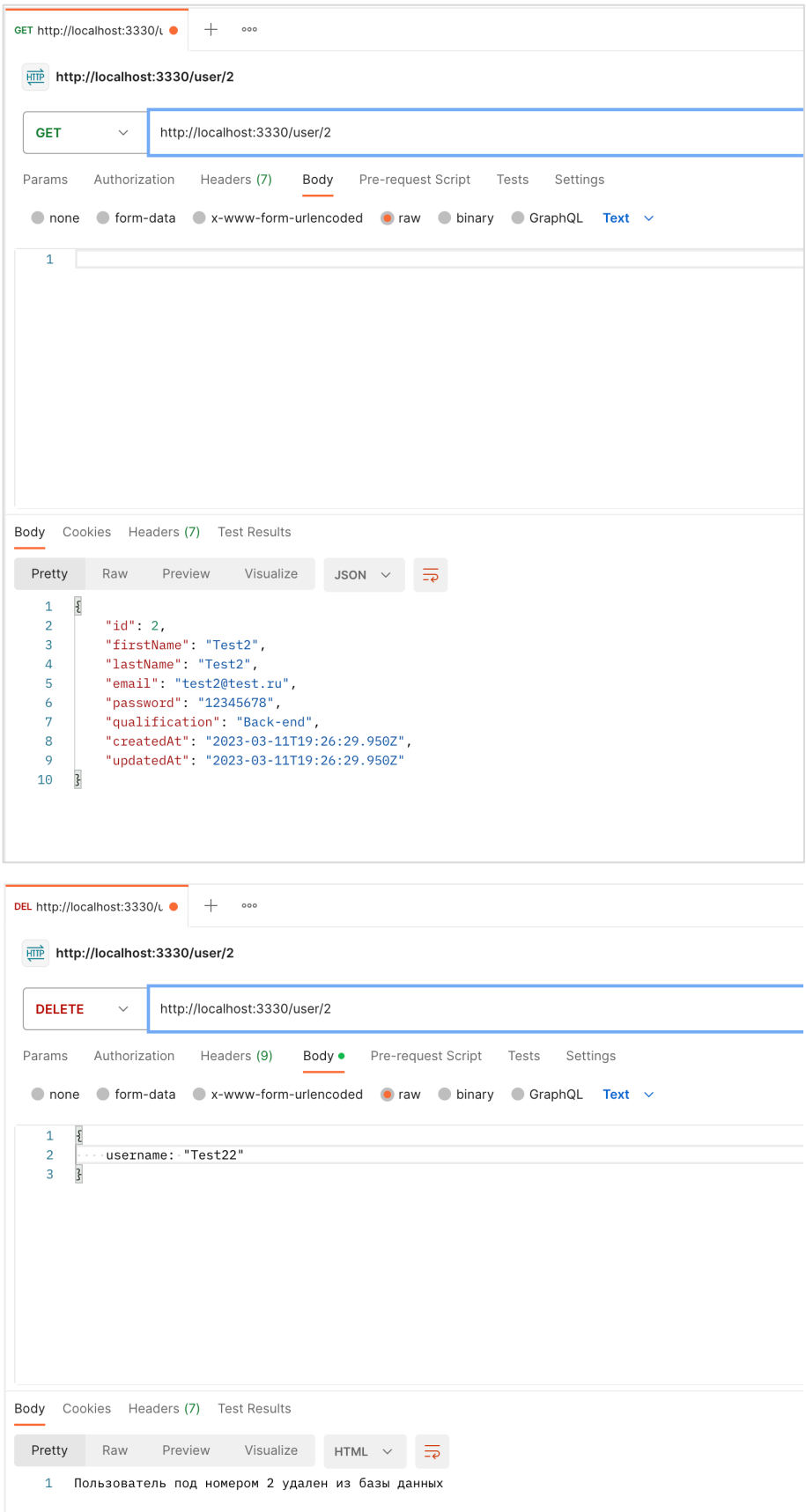


Рисунок 4 – Тестирование запросов в Postman

Вывод

В результате работы была успешно продумана и реализована модель пользователя. Для работы с данными пользователей был реализован набор CRUD-методов, позволяющих создавать, читать, обновлять и удалять записи о пользователях в базе данных.

Для реализации CRUD-методов были использованы средства Express и Sequelize, что позволило значительно упростить процесс работы с базой данных и ускорить разработку.

Также был успешно написан запрос для получения пользователя по id/email, который позволяет быстро находить нужного пользователя в базе данных и выполнять с ним необходимые действия.

В результате выполнения данной практической работы были получены ценные навыки работы с базами данных, а также углублены знания в области разработки веб-приложений с использованием фреймворка Express и ORM Sequelize.