САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа 2: Знакомство с микрофреймворком Express. Знакомство с ORM Sequelize

Выполнил:

Конев Антон

Группа К33402

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. Собственная модель пользователя выглядит следующим образом:
   - ❖ firstName (Имя)
   - ❖ lastName (Фамилия)
   - ❖ email (Почта)
   - ❖ password (Пароль)
   - ❖ username (Логин)

```
'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  2 usages
  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.
     */
    2 usages
    static associate(models) {
      // define association here
    }
  }
  User.init( attributes: {
    firstName: DataTypes.STRING,
    lastName: DataTypes.STRING,
    email: DataTypes.STRING,
    password: DataTypes.STRING,
    username: DataTypes.STRING
  }, options: {
    sequelize,
    modelName: 'User',
  });
  return User;
};
```

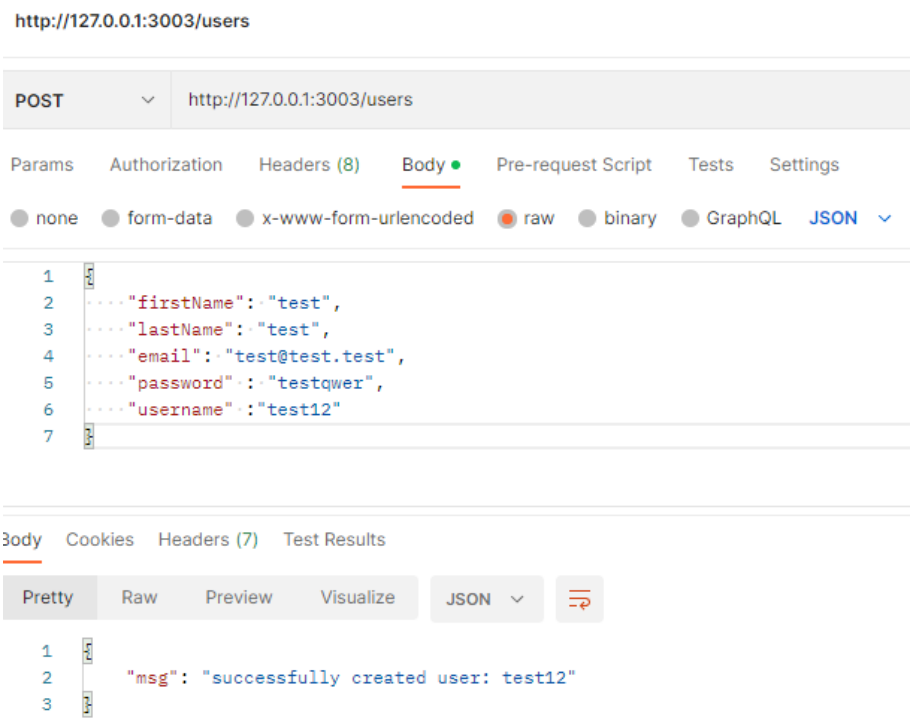
2. Реализация CRUD-методов средствами Express + Sequelize

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

const app = express()
const port = 3003


app.use(express.json());

app.get('/', async (req : … , res : Response<ResBody, LocalsObj> ) => {
    res.send( body: 'useless route')
})

app.get('/users/:id', async (req : … , res : Response<ResBody, LocalsObj> ) => {
    const user = await db.User.findByPk(req.params.id)

    console.log('user is', user)

    if (user) {
        return res.send(user.toJSON())
    }

    return res.send( body: {"msg": "user is not found"})
})

app.get('/users', async (req : … , res : Response<ResBody, LocalsObj> ) => {
    const users = await db.User.findAll()

    if (users) {
        return res.send(users)
    }

    return res.send( body: {"msg": "no users in db"})
})

app.post( path: '/users', handlers: async (req : … , res : Response<ResBody, LocalsObj> ) => {
    try {
        const user = await db.User.create(req.body)
        return res.send( body: {"msg": `successfully created user: ${user.toJSON().username}`})
    } catch (e) {
        return res.send( body: {"msg": "failed to create user"})
    }
})
```

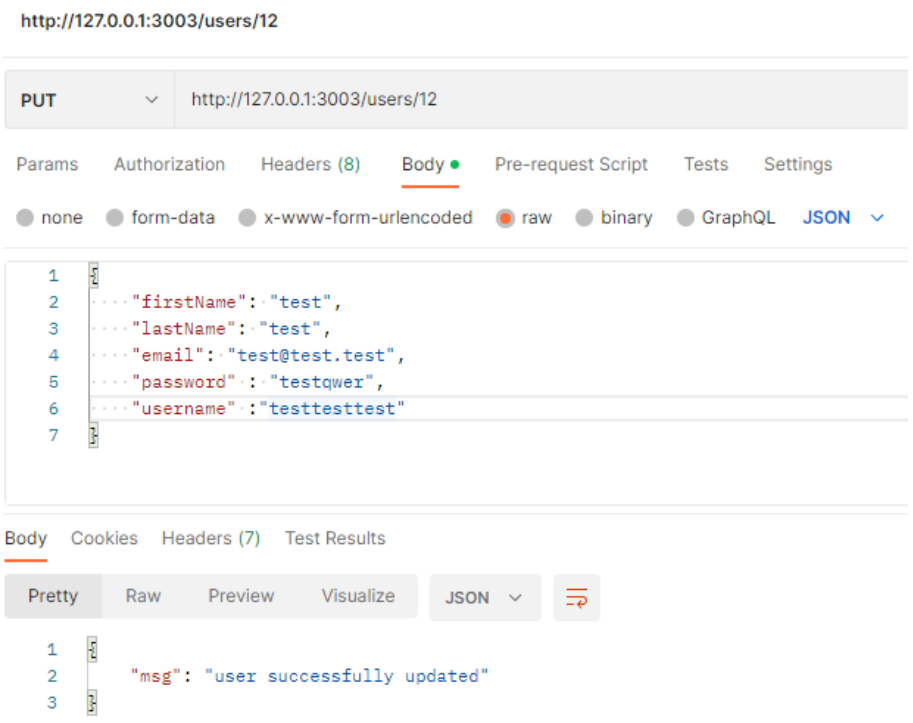
```
app.put( path: '/users/:id', handlers: async (req : ... , res : Response<ResBody, LocalsObj> ) => {
    const user = await db.User.findByPk(req.params.id)

    if (user) {
        try {
            user.update(req.body, {where: {id: req.params.id}})
            return res.send( body: {"msg": "user successfully updated"})
        } catch (e) {
            return res.send( body: {"msg": "failed to update user"})
        }
    }

    return res.send( body: {"msg": "user not found"})
})

app.delete( path: '/users/:id', handlers: async (req : ... , res : Response<ResBody, LocalsObj> ) => {
    const user = await db.User.findByPk(req.params.id)

    if (user) {
        try {
            user.destroy({where: {id: req.params.id}})
            return res.send( body: {"msg": "user successfully deleted"})
        } catch (e) {
            return res.send( body: {"msg": "failed to delete user"})
        }
    }

    return res.send( body: {"msg": "user not found"})
})

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

3. Проверка работы CRUD-методов с помощью Postman

CREATE (POST)

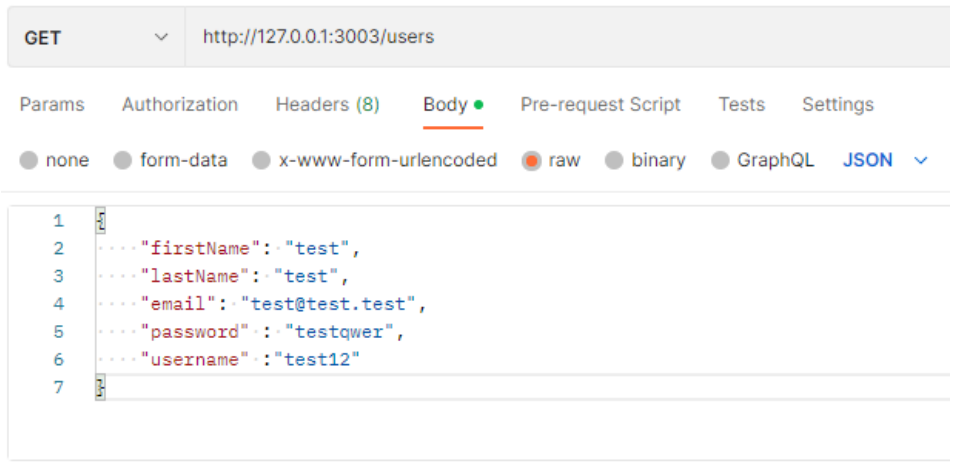


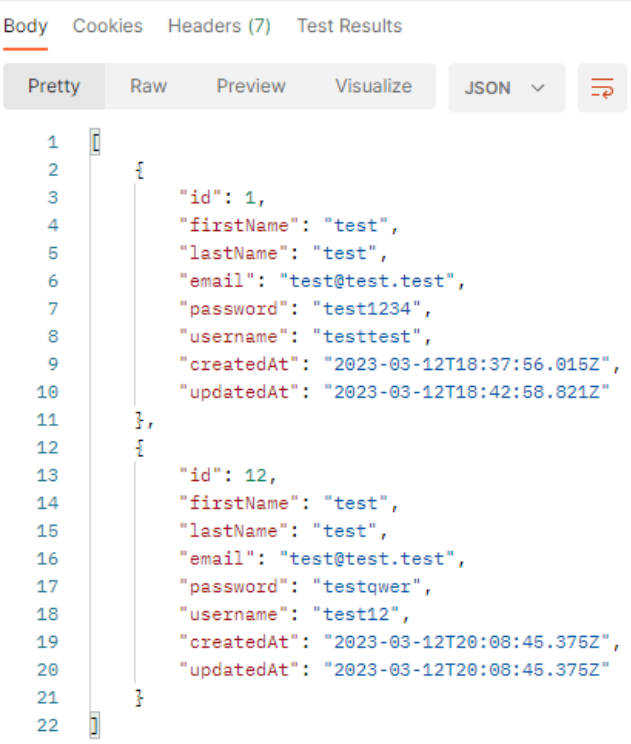
UPDATE (PUT)



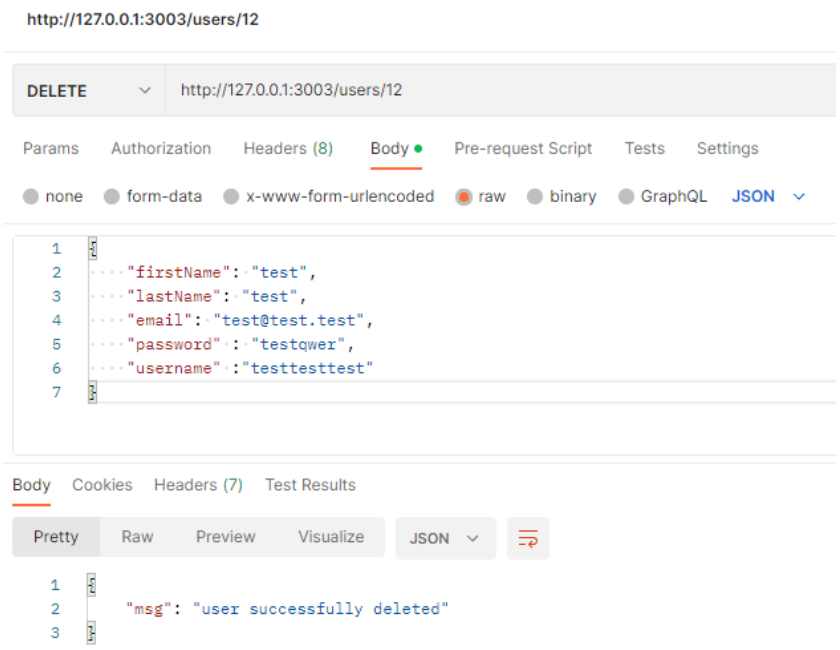
READ (GET)

http://127.0.0.1:3003/users

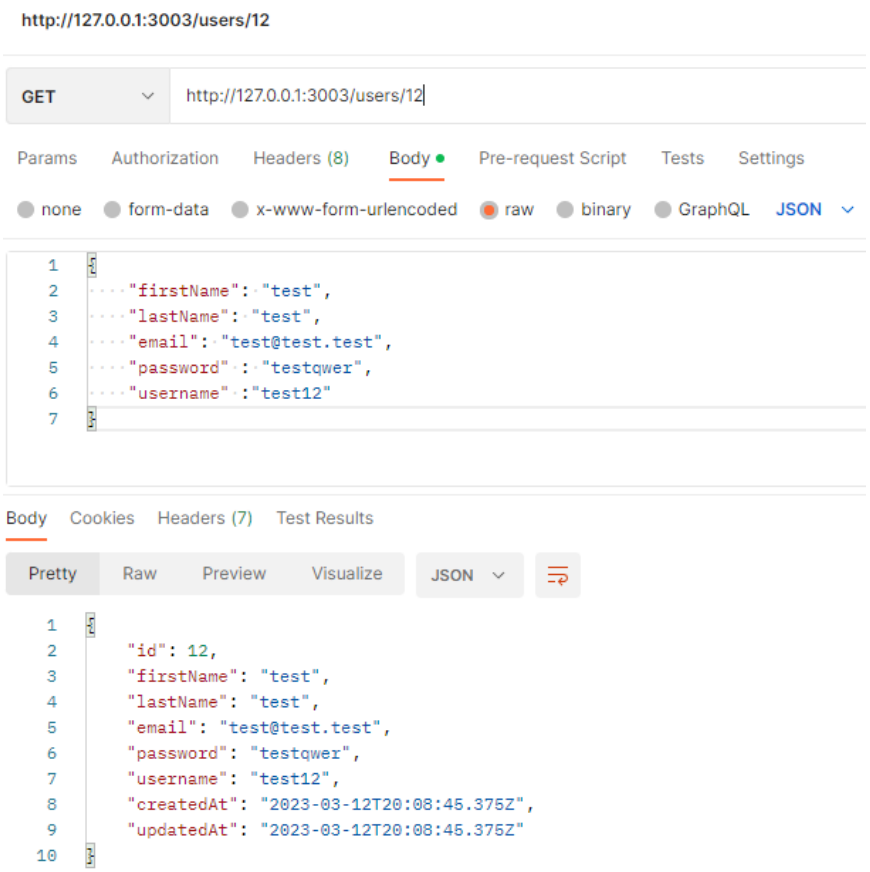




DELETE



4. Запрос для получения пользователя по id



Вывод

В ходе домашней работы была придумана собственная модель пользователя с помощью ORM Sequelize. Также реализованы CRUD-методы для работы с пользователем средствами Express и Sequelize. Была добавлена возможность поиска пользователя по id.