САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бек-энд разработка

Отчет

Домашняя работа №2

Выполнил:

Кобелев Л.К.

K33401

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

1. Продумать свою собственную модель пользователя
2. Реализовать набор из CRUD-методов для работы с пользователями средствами Express + Sequelize
3. Написать запрос для получения пользователя по id/email

Ход работы

Инициализируем проект:

`npm init`

Устанавливаем `express`, `sequelize`, `sequelize cli` и `sqlite`:

`npm install express -S`

`npm install sequelize -S`

`npm install --save-dev sequelize-cli`

`npm install sqlite3 -S`

Инициализируем:

`npx sequelize init`

Создадим модель пользователя, которая будет состоять из имени, почты и пароля:

`npx sequelize-cli model:generate --name User --attributes username:string,email:string,password:string`

Делаем миграцию, предварительно поменяв в конфиге `mysql` на `sqlite` и добавив хранилище:

`npx sequelize-cli db:migrate`

Создаем `index.js`, который будет выступать сервером:

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

const app = express()
const port = 8080

app.use(bodyParser.json())
app.use(express.urlencoded({ extended: true }));

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

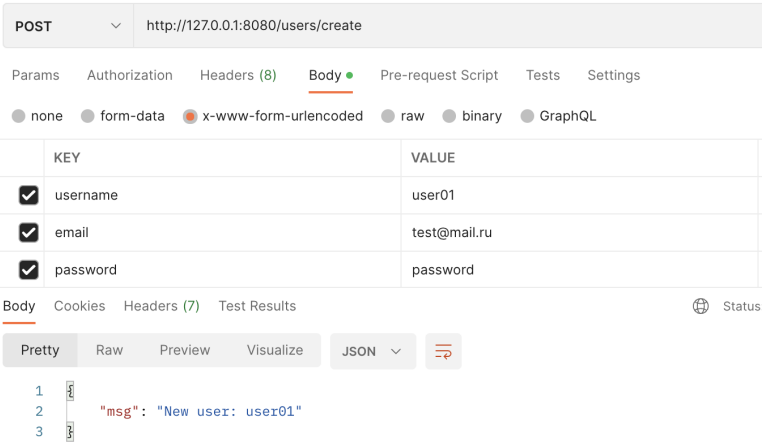
Реализуем CRUD операции для модели User:

Код

Create User:

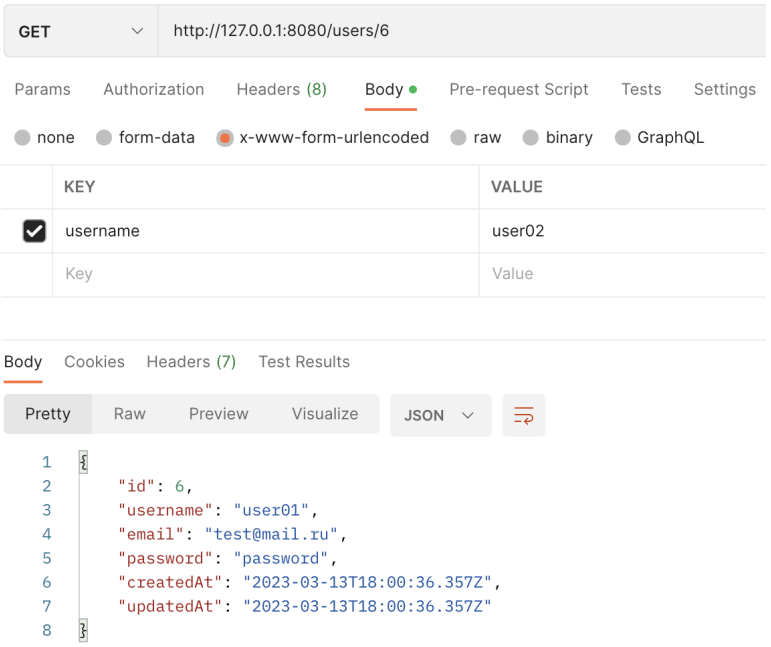
```
app.post('/users/create', async (req, res) => {
    try {
        const user = await db.User.create(req.body);
        return res.send({"msg": "New user: " + user.username})
    } catch (error) {
        return res.send({"msg": error})
    }
});
```

Проверка



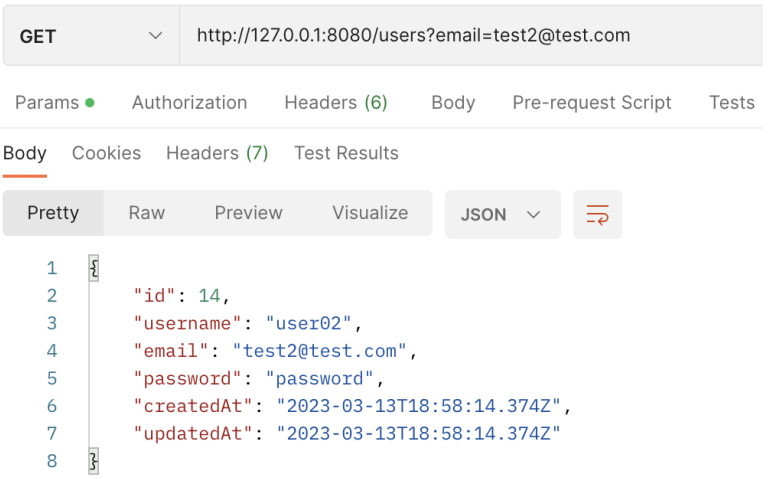
Get User by PK:

```
app.get('/users/:id', async (req, res) => {
    const user = await db.User.findByPk(req.params.id)
    if (user) {
        return res.send(user.toJSON())
    }
    return res.send({"msg": "User does not exist"})
})
```



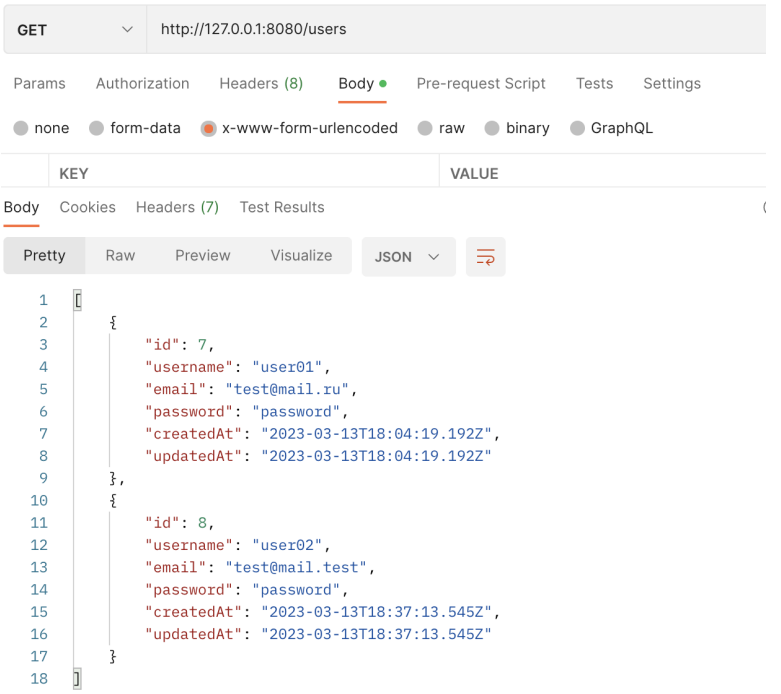
Get User by e-mail:

```
app.get('/users', async (req, res) => {
    let users = await db.User.findAll()
    if (req.query.email) {
        users = users.find(user => {
            return user.email === req.query.email
        })
    }
    return res.send(users)
});
```



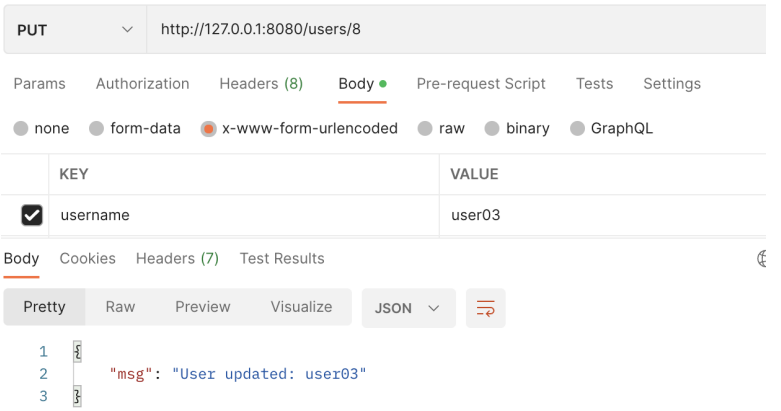
Get All Users:

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



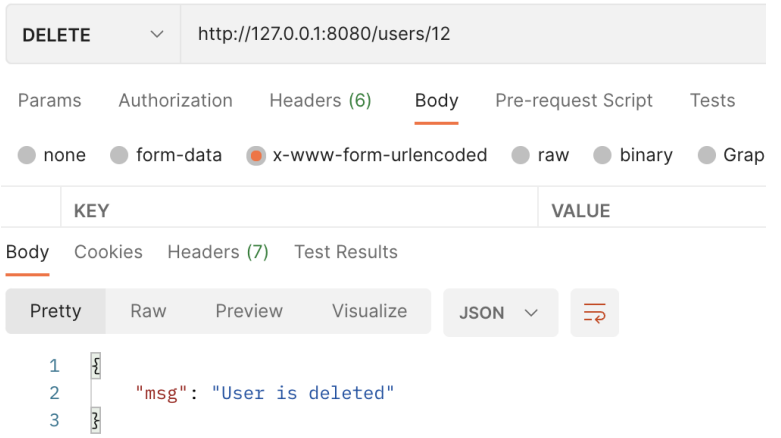
Update User:

```
app.put('/users/:id', async (req, res) => {
    const user = await db.User.findByPk(req.params.id);
    if (user) {
        try {
            user.update(req.body);
            return res.send({"msg": "User updated: " + user.username})
        } catch (error) {
            return res.send({"msg": error})
        }
    }
    return res.send({"msg": "User does not exist"})
})
```



Delete User:

```
app.delete('/users/:id', async (req, res) => {
    const user = await db.User.destroy({where: {id: req.params.id}})
    if (user) {
        return res.send({"msg": "User is deleted"})
    }
    return res.send({"msg": "User does not exist"})
})
```



Вывод

Были опробованы микрофреймворк Express и ORM Sequelize.