САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Практическая/Лабораторная работа

Выполнил:
Кривцов Павел
Группа K33402

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

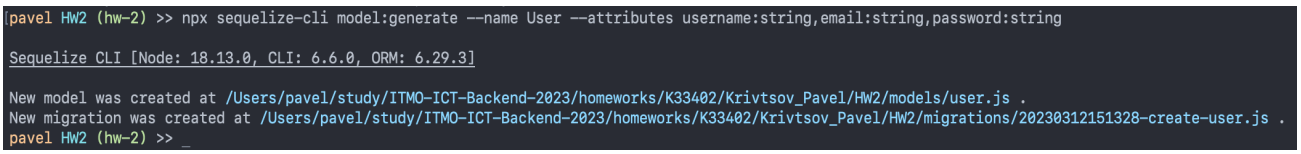
Задачи

- 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.

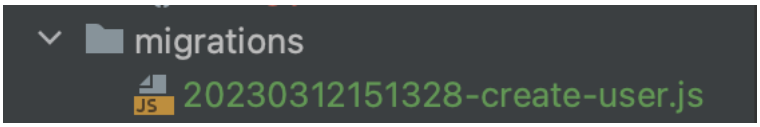
Ход работы

1. Создадим собственную модель пользователя. Она будет содержать в себе следующие поля: username, email, password. Для этого воспользуемся *sequelize-cli* командой:

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



После чего в файловой системе проекта появится новые миграция и модель:



```
'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
    }
  }
  User.init( attributes: {
    username: DataTypes.STRING,
    email: DataTypes.STRING,
    password: DataTypes.STRING
  }, options: {
    defaultScope: {
      attributes:{exclude:['password']},
    },
    sequelize,
    modelName: 'User',
  });
  return User;
};
```

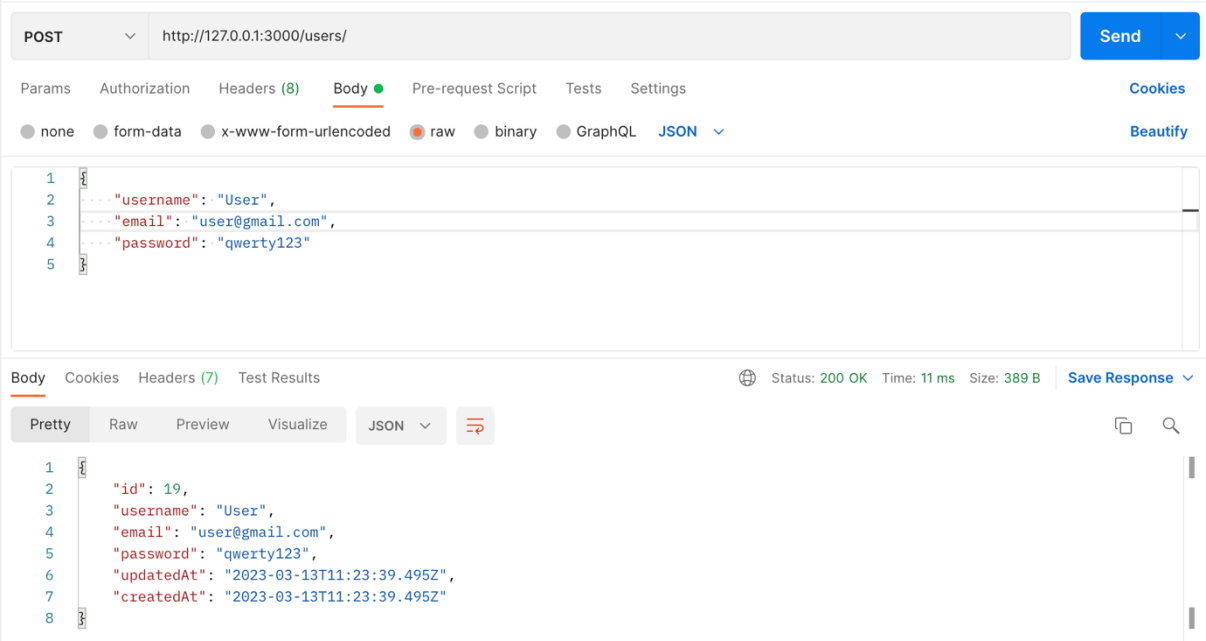
Применим появившуюся миграцию с помощью команды `npx sequelize-cli db:migrate`, после чего в файловой системе появится файл базы данных.

2. Реализуем CRUD методы для работы с пользователем в роутере `users.js`, который будем подключать к приложению по пути “/users” в запросе:

```
const usersRouter = require('./routes/users');
app.use('/users', usersRouter);
```

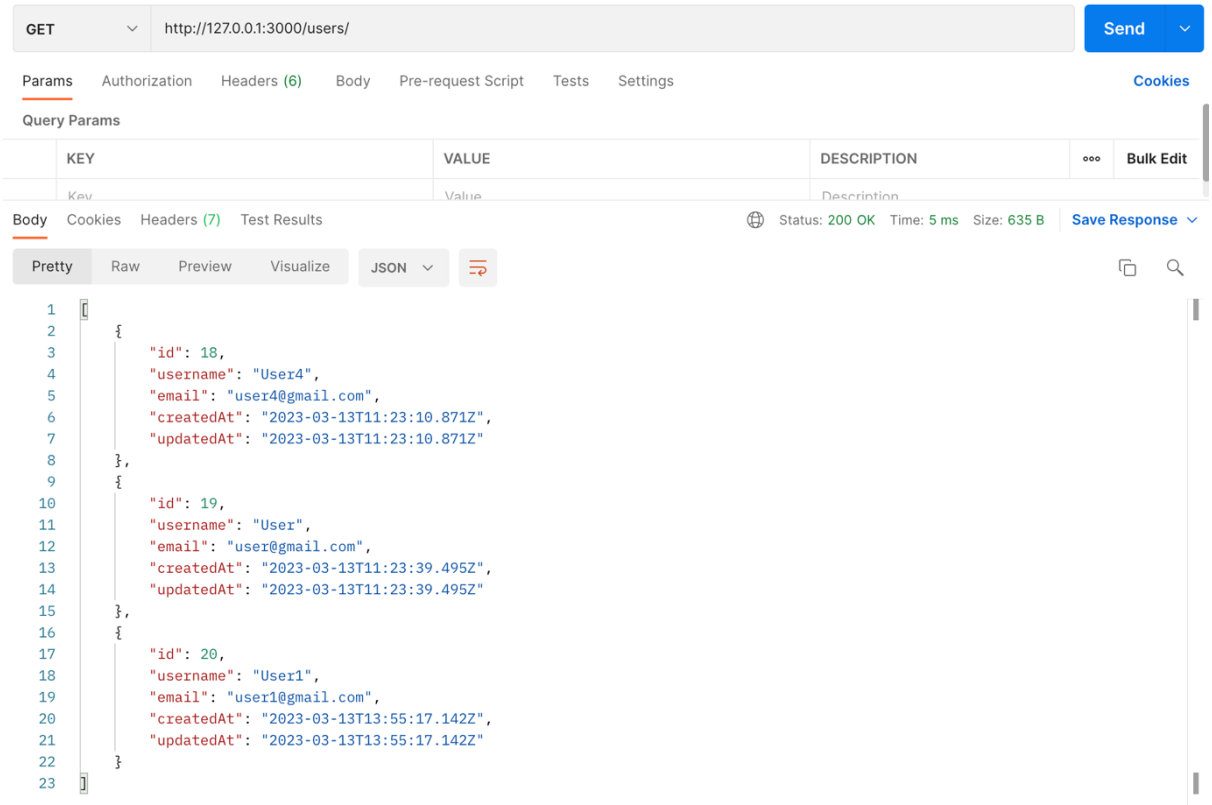
- Create

```
// CREATE user
router.post( path: '/', handlers: async (req, res) => {
  console.log(req.body)
  try {
    const user = await db.User.create(req.body);
    res.status( code: 200).send(user.toJSON());
  } catch (err) {
    res.status( code: 400).send( body: {"msg": err});
  }
});
```



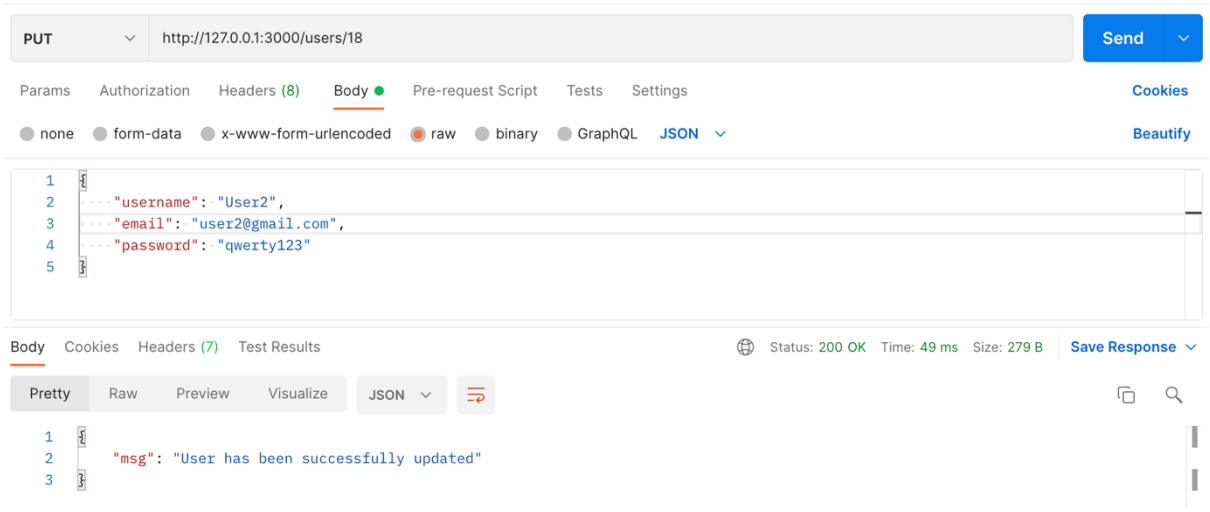
- Read

```
// GET users listing
router.get( path: '/', handlers: async (req, res) => {
  const users = await db.User.findAll()
  res.status( code: 200).send(users)
});
```



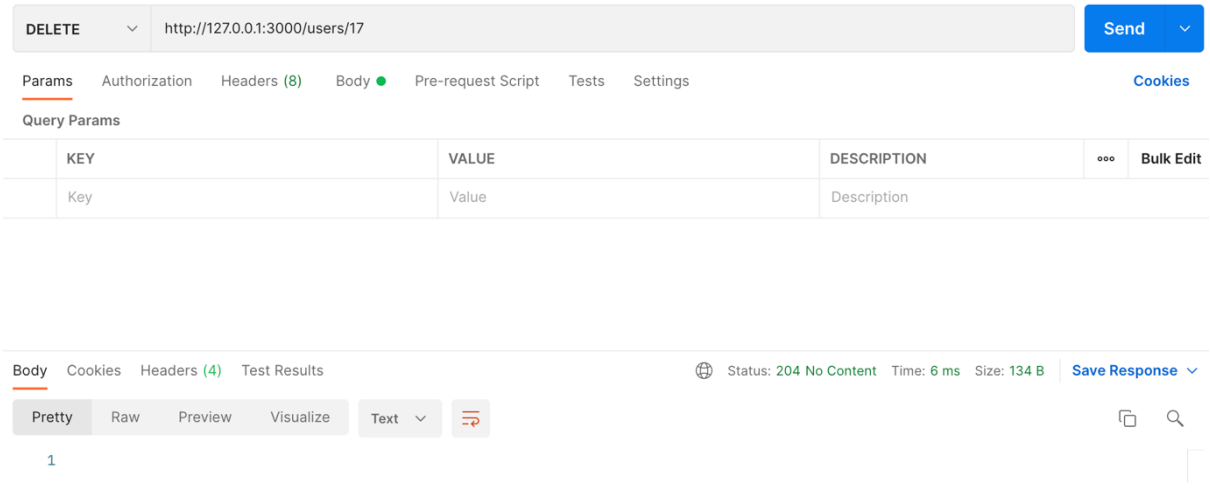
- Update

```
// UPDATE user
router.put( path: '/:id', handlers: async (req, res) => {
  try {
    const user = await db.User.update(req.body, { where: { id: req.params.id } });
    res.status( code: 200).send( body: { "msg": "User has been successfully updated" });
  } catch (err) {
    res.status( code: 400).send( body: { "msg": err });
  }
})
```



- Delete

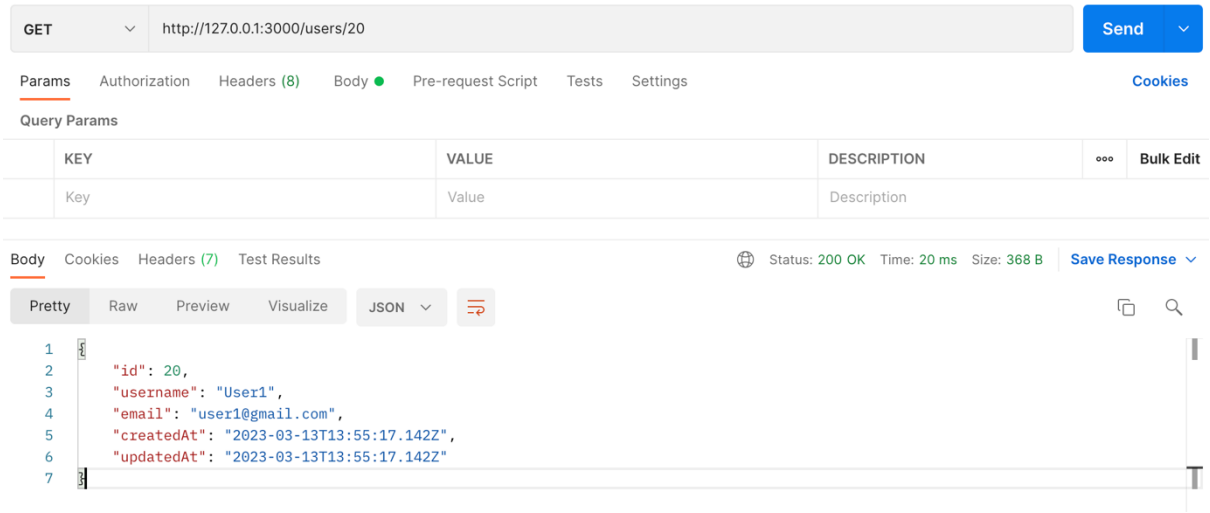
```
// DELETE user
router.delete( path: '/:id', handlers: async (req, res) => {
  const user = await db.User.destroy({ where: { id: req.params.id } })
  if (user) {
    res.status( code: 204).send()
  } else {
    res.status( code: 404).send( body: {'msg': 'User not found'})
  }
})
```



3. Напишем запрос для получения пользователя по id

```
// GET user by id
router.get( path: '/:id', handlers: async (req, res) => {
  const user = await db.User.findByPk(req.params.id)
  if (user) {
    return res.status( code: 200).send(user.toJSON())
  }

  return res.status( code: 404).send( body: {"msg": "user not found"})
})
```



Выводы

В ходе выполнения работы была создана модель пользователя и реализованы CRUD-методы для работы с ней с использованием ORM Sequelize и Express.