Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Факультет Инфокоммуникационных технологий (ИКТ)

Образовательная программа Мобильные и сетевые технологии

О Т Ч Е Т

Back-end домашнее задание №2

Тема задания: «Знакомство с ORM Sequelize»

Выполнил: Мамедов Тогрул, группа К33402

Проверил: Добряков Давид Ильич

Санкт-Петербург
2023

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

Ход работы:

Инициализируем проект и скачиваем необходимые инструменты, сохранив их в файл конфигурации.

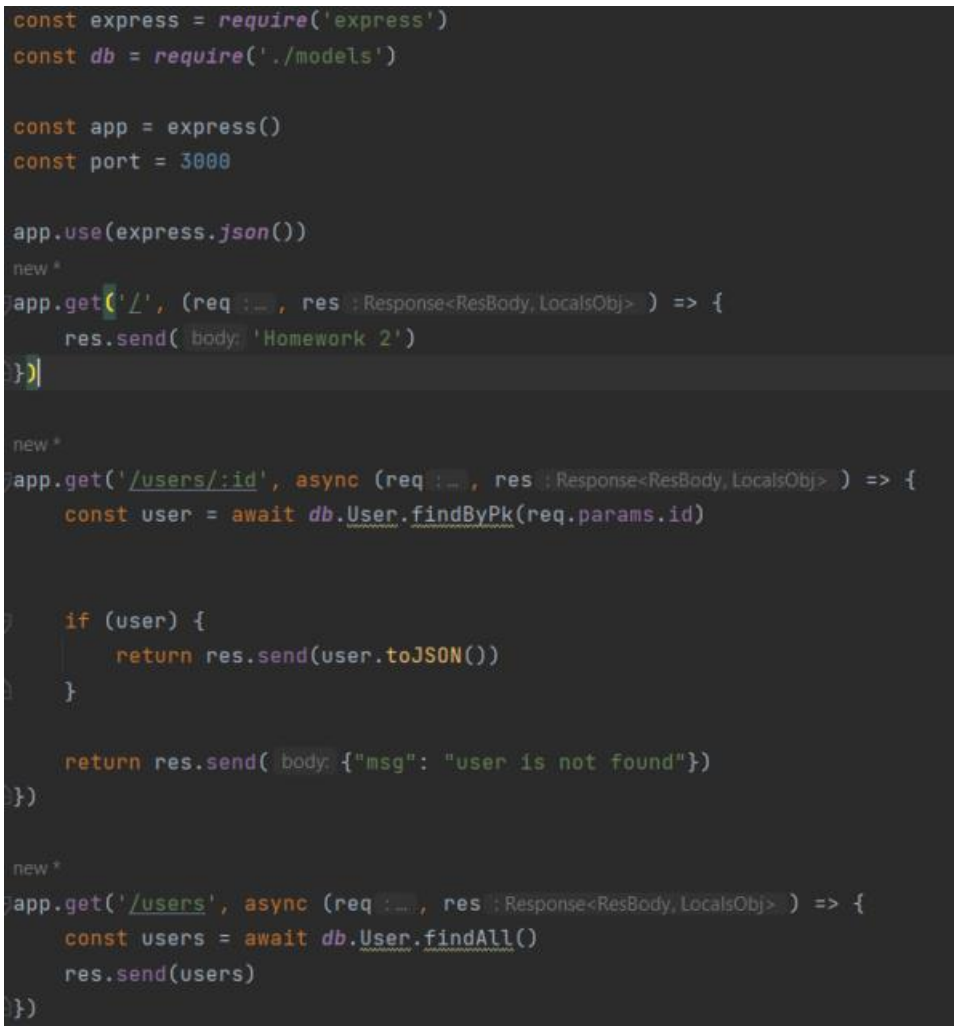
С помощью sequelize-cli создаем модель пользователя и миграции:

npx sequelize-cli model:generate --name User --attributes firstName:string,lastName:string,email:string,password:string,img_url:string,u sername:string

npx sequelize-cli db:migrate

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

Файл index.js с нужными запросами:



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

const app = express()
const port = 3000

app.use(express.json())
new *
app.get('/', (req : _ , res : Response<ResBody, LocalsObj> ) => {
    res.send( body: 'Homework 2')
})

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


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

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

new *
app.get('/users', async (req : _ , res : Response<ResBody, LocalsObj> ) => {
    const users = await db.User.findAll()
    res.send(users)
})
```

```
app.post( path: '/users', handlers: async (req : _ , res : Response<ResBody, LocalsObj> ) => {
    try {
        console.log(req.body)
        const user = await db.User.create(req.body);
        res.send(user.toJSON());
    } catch (e) {
        res.status( code: 400).send( body: {"detail": "Failed to create user"});
    }
})

new *
app.put( path: '/users/:id', handlers: async (req : _ , res : Response<ResBody, LocalsObj> ) => {
    try {
        console.log(req.body)
        const user = await db.User.update(req.body, {where: {id: req.params.id}});
        res.send( body: {'status_code': 'User updated'});
    } catch (err) {
        res.send(err);
    }
})

new *
app.delete( path: '/users/:id', handlers: async (req : _ , res : Response<ResBody, LocalsObj> ) => {
    const user = await db.User.destroy({where: {id: req.params.id}})
    if (user) {
        res.send( body: {'status_code': 'User deleted'})
    } else {
        res.status( code: 404).send( body: {'error_code': 'User not found'})
    }
})

new *
app.listen(port, hostname: () => {
```

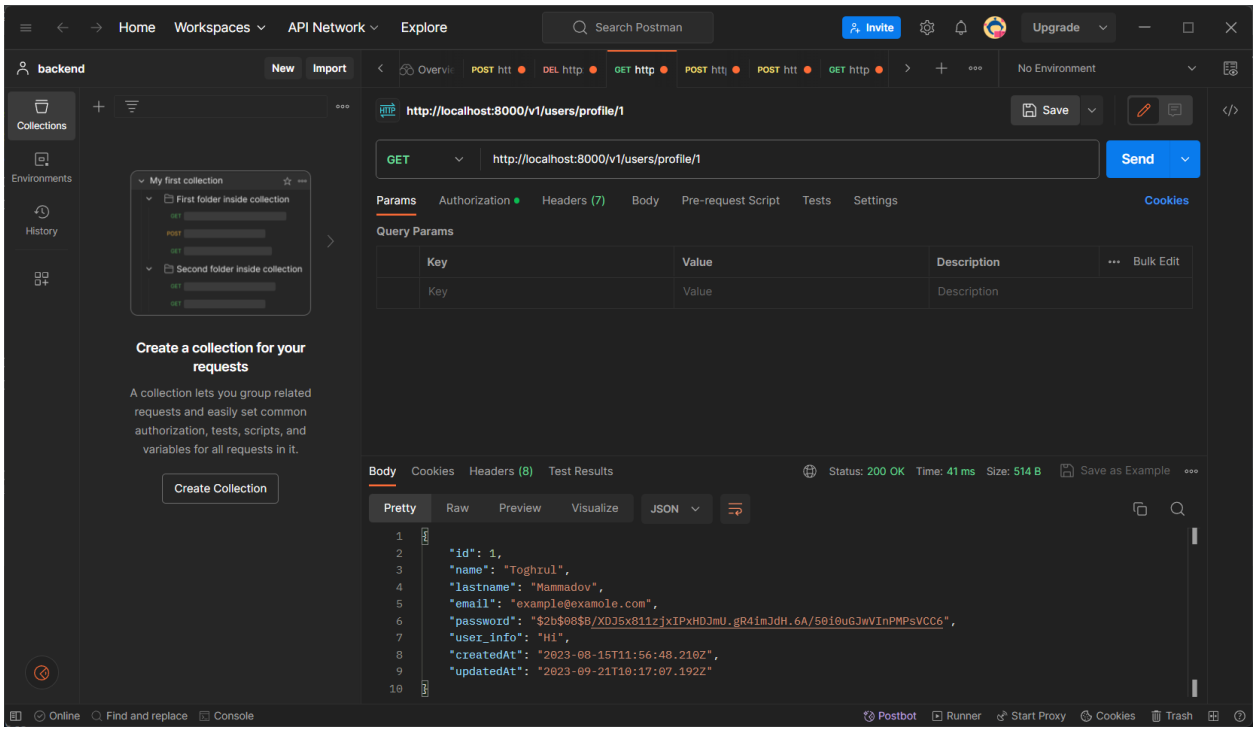
Файл для создания пользователя:

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

1 usage  new *
async function main() {
  await db.User.create({
    firstName: 'Anastasia',
    lastName: 'Konik',
    email: 'konik.ftl@mail.ru',
    password: '123konik123',
    username: 'anastasiakonik'
  })
}

main()
```

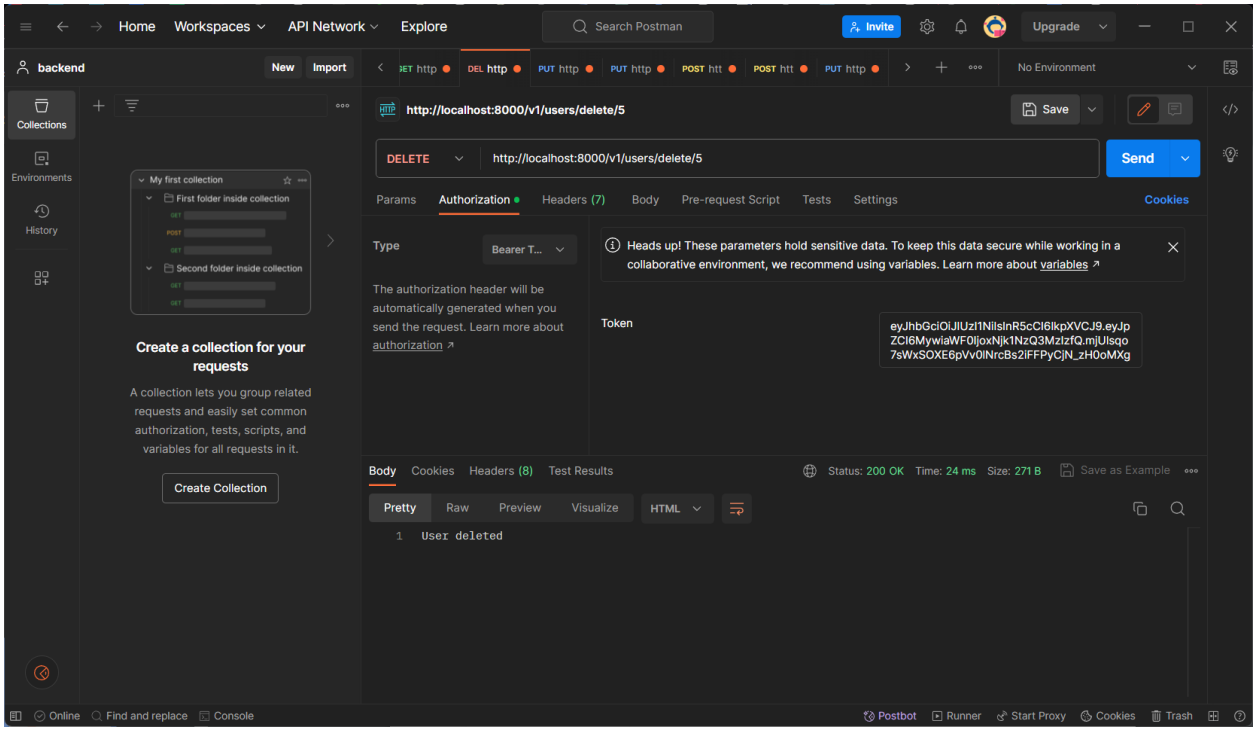
Проверка работы сервера:

Запросы в postman:

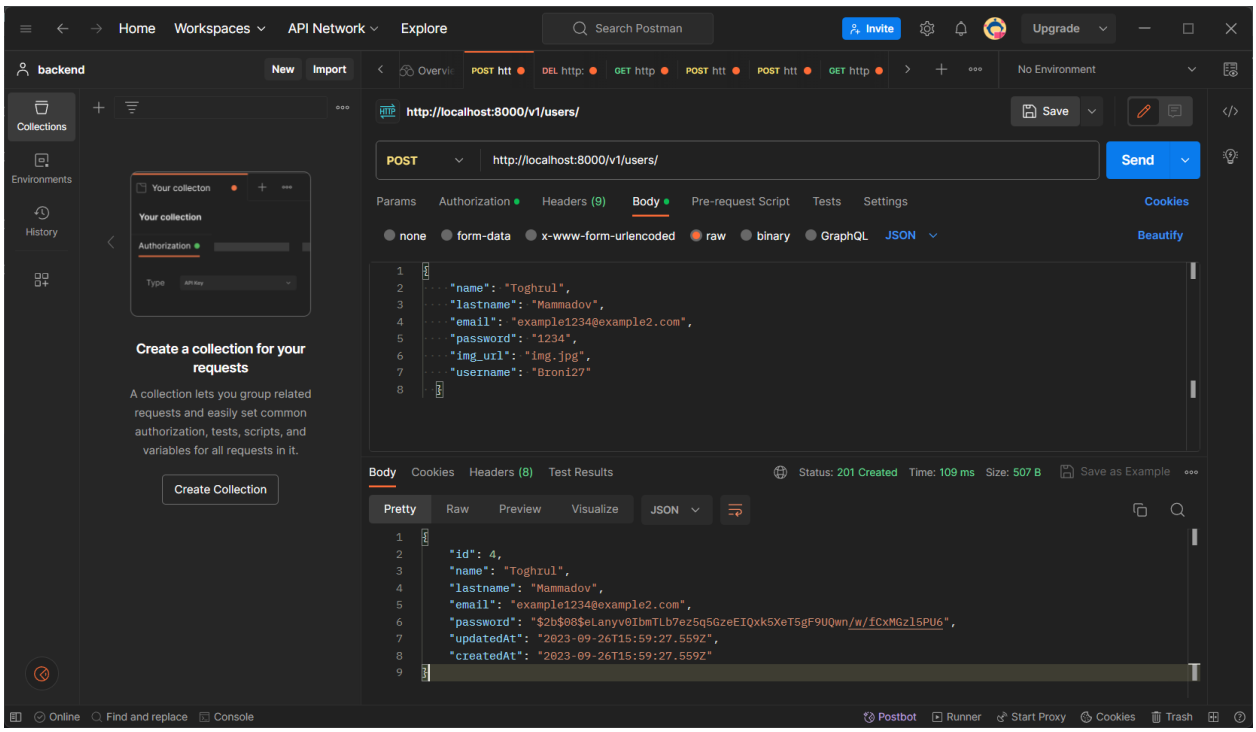
– получение пользователей:



– удаление пользователя по id



– добавление пользователя:



Вывод:

В ходе работы я познакомился с ORM Sequelize и создал с ее помощью модели, а также написал запросы к бд с помощью Express.