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

по Домашней работе 1

Тема задания: «npm, node.js»

Дисциплина: Бек-энд разработка

Специальность: 09.03.03 Прикладная информатика

Проверил:	Выполнил:
Добряков Д. И. _____	Балцат К. И.,
Дата: «__»_____2023 г.	студент группы К33401
Оценка: _____	

Санкт-Петербург
2023

ЦЕЛЬ РАБОТЫ

Цель данной работы: получить практические навыки работы с ORM Sequalize.

ВЫПОЛНЕНИЕ

1 Продумать свою собственную модель пользователя

Модель пользователя "user" имеет следующие поля (Таблица 1).

User

Id	Primary_key, integer, not null
Name	string(20), not_null
Surname	string(20)
Phone	string(20)
email	string(30), not_null
Password	string(30), not_null

Таблица 1

2 Реализовать набор из CRUD-методов для работы с пользователями средствами Express + Sequelize

Подготавливаем проект по шагам:

1. Создадим проект и инициализируем его командой `npm init`,
2. загрузим пакеты «express», «sequelize» и «sqlite3», с помощью команды `npm i <package_name> -S`,
3. установим «Sequelize CLI» командой `npm install --save-dev sequelize-cli`.

Создадим модель пользователя.

1. Генерируем структуру через «Sequelize CLI» командой `npx sequelize init`,
2. создаем модель "user" командой `npx sequelize-cli model:generate --name User --attributes name:string,surname:string,phone:string,email:string,password:string`
3. делаем миграцию командой `npx sequelize db:migrate`.

Реализуем CRUD-методы в файле *index.js* (рисунок 1).

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

app.get('/users', async (req, res) => {
  let users = []
  if (req.query && req.query.email) {
    users = await db.User.findAll({ where: { email: req.query.email } })
  } else {
    users = await db.User.findAll()
  }
  console.log(users)
  return res.send(users)
})

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

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 deleted"})
  }
  return res.send({"msg": "User is not found"})
})

app.put('/users/:id', async (req, res) => {
  const user = await db.User.findByPk(req.params.id);
  if (user) {
    try {
      user.update(req.body, {where: {id: req.params.id}});
      return res.send({"msg": "Successfully updated!"})
    } catch (e) {
      return res.send({"msg": e})
```

Рисунок 1 – CRUD-методы для модели USER.

3 Написать запрос для получения пользователя по *id/email*

Проведем демонстрацию работоспособности.

1. Создание пользователя (рисунок 2).

2. Отображение всех пользователей (рисунок 3).

3. Получение пользователя по id (рисунок 4).

4. Получение пользователя по email (рисунок 5).

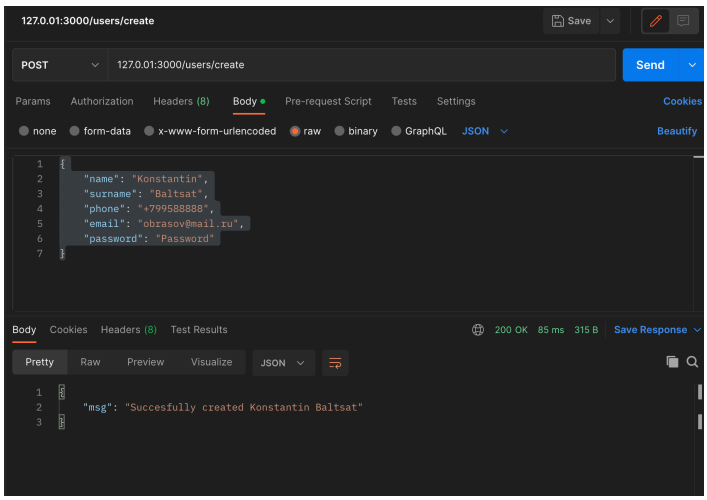

Рисунок 2 – Создание пользователя.

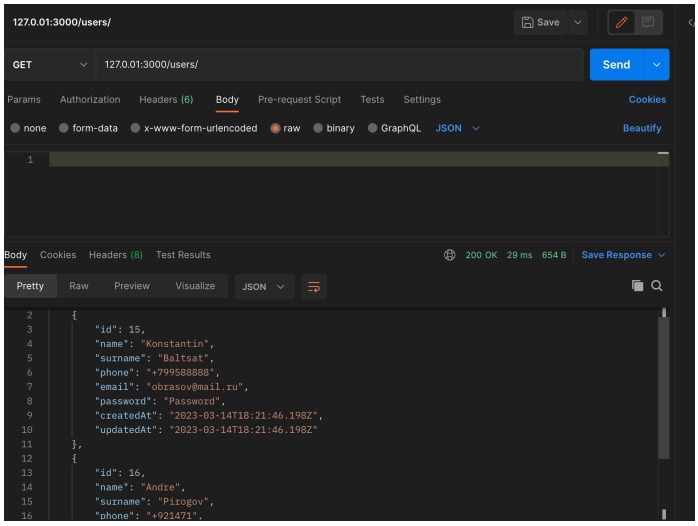

Рисунок 3 – Отображение пользователей.

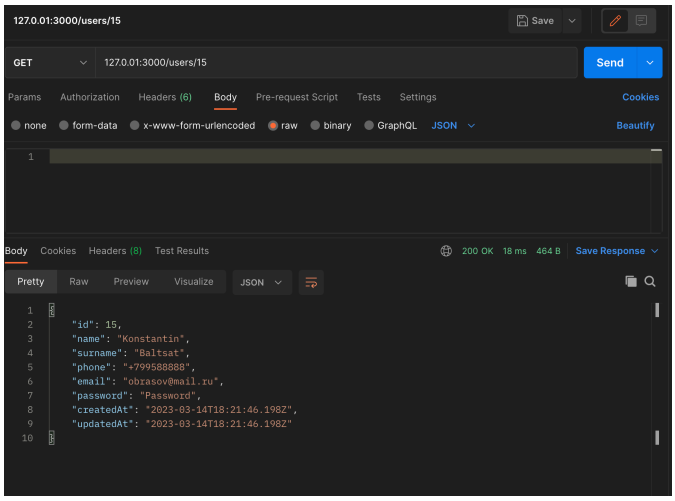

Рисунок 4 – получение пользователя по id.

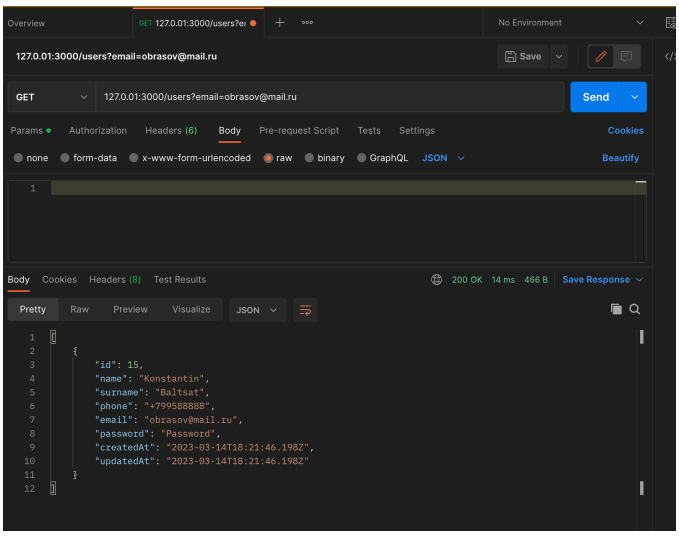

Рисунок 5 – получение пользователя по email.

ВЫВОД

В результате работы я получил практические навыки по работе с express и sequelize, а также попрактиковался в ПО Postman.