Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»
Факультет инфокоммуникационных технологий

ОТЧЕТ

по Домашней работе № 2

по теме: «ORM Sequelize»
по дисциплине: Бек-энд разработка

Специальность:
09.03.03 Мобильные и сетевые технологии

Проверил:
Добряков Д. И. _____
Дата: «__» _____ 202__г.
Оценка _____

Выполнил:
студент группы K33401
Чернов Е. К.

Санкт-Петербург
2023

ЦЕЛЬ РАБОТЫ

Получить практические навыки по работе с *ORM Sequelize*.

ВЫПОЛНЕНИЕ

1 Продумать свою собственную модель пользователя

Модель пользователя *«user»* имеет следующий вид:

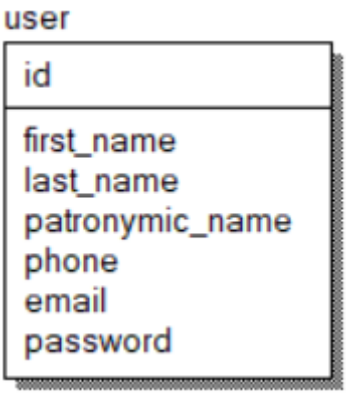

Рисунок 1 - Модель пользователя

Данная модель имеет следующие поля:

- *id - primary key, integer, not null,*
- *first_name - string(20), not null,*
- *last_name - string(20), not null,*
- *patronymic_name - string(20), null,*
- *phone - string(20), null,*
- *email - string(30), not null,*
- *password - string(30), not null.*

2 Реализовать *CRUD*-методы для *Express* и *Sequelize*

Подготавливаем проект, выполняем следующие шаги:

1) создадим проект и инициализируем его с помощью команды *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*.

Создаем 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 first_name:string, last_name:string, patronymic_name:string, phone:string, email:string, password:string*,

3) делаем миграцию командой *npx sequelize db:migrate*.

Реализуем *CRUD*-методы:

1) в файле *«index.js»* инициализируем *express* и связываем путь *http* с *route* (рисунок 2),

2) в папке *«routes»* прописываем пути и методы (рисунок 3),

3) в папке *«services»* прописываем обращения к бд (рисунок 4).

```
let express = require('express');

let indexRouter = require('./routes/index');
let usersRouter = require('./routes/user');

let app = express();

app.use('/', indexRouter);
app.use('/user', usersRouter);

module.exports = app;
```

Рисунок 2 - файл «./index.js»

```
let express = require('express');
let router = express.Router();

const { getAll, getUser, createUser } = require('../services/user')

// GET user list
router.get( path: '/', handlers: async (req : … , res : Response<ResBody, LocalsObj> , next : NextFunction ) => {
    res.send(await getAll());
});

// GET user by id
router.get( path: '/:id', handlers: async (req : … , res : Response<ResBody, LocalsObj> ) => {
    res.send(await getUser(req.params.id))
})

// POST create user
router.post( path: '/req', handlers: async (req : … , res : Response<ResBody, LocalsObj> ) => {
    res.send(await createUser(req.query))
})


module.exports = router;
```

Рисунок 3 - файл *«./routes/user.js»*

user.js

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

async function getAll() {
    const users = await db.User.findAll()
    let userList = []

    for (const user of users) {
        userList.push(user.toJSON())
    }

    if (userList) {
        return userList
    }

    return "users are not found"
}

async function getUser(id) {
    const user = await db.User.findByPk(id)

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

    return "user is not found"
}

async function createUser(usr) {
    const user = await db.User.create({
        first_name: usr.first_name,
```

user.js

```
    if (user) {
        return user.toJSON()
    }

    return "user is not found"
}

async function createUser(usr) {
    const user = await db.User.create({
        first_name: usr.first_name,
        last_name: usr.last_name,
        patronymic_name: usr.patronymic_name,
        phone: usr.phone,
        email: usr.email,
        password: usr.password
    })

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

    return "user is not create"
}

module.exports = { getAll, getUser, createUser }
```

Рисунок 4 - файл *«./services/user.js»*

3 Написать запрос для получения пользователя по *id*

Запрос для получения всех пользователей отображен на рисунке 5.

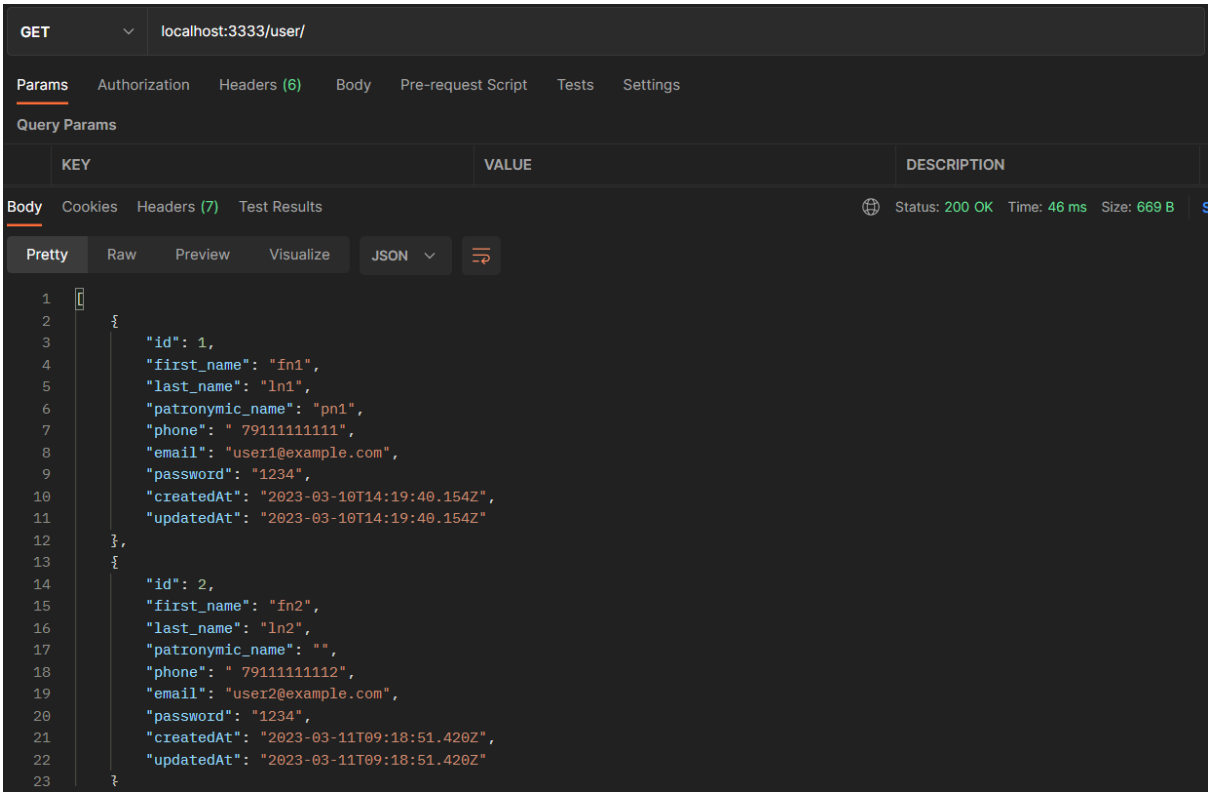

Рисунок 5 - *GET* запрос на получение всех пользователей

Запрос для получения пользователя по id отображен на рисунке 6.

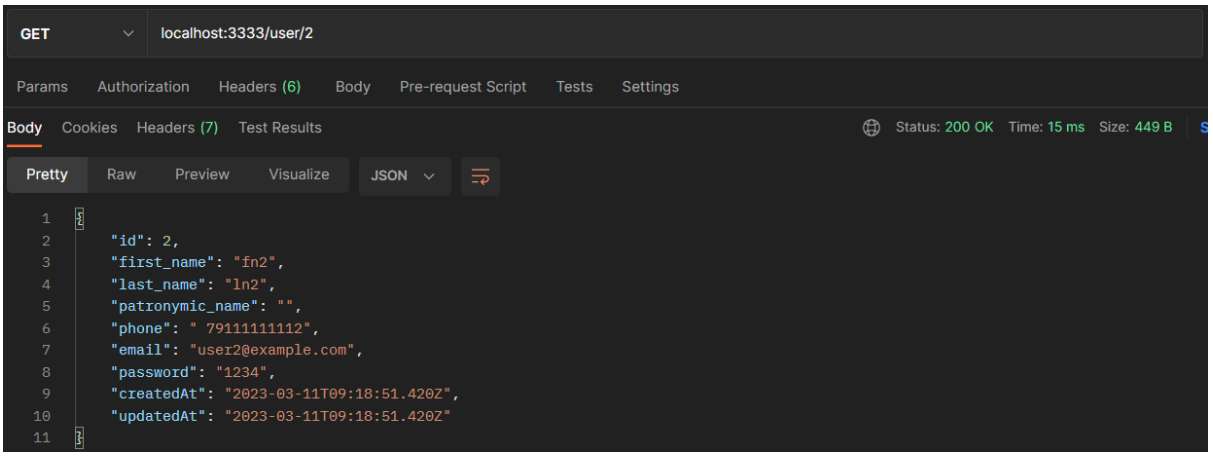

Рисунок 6 - *GET* запрос на получение пользователя по *id*

Запрос на создание пользователя отображен на рисунке 7.

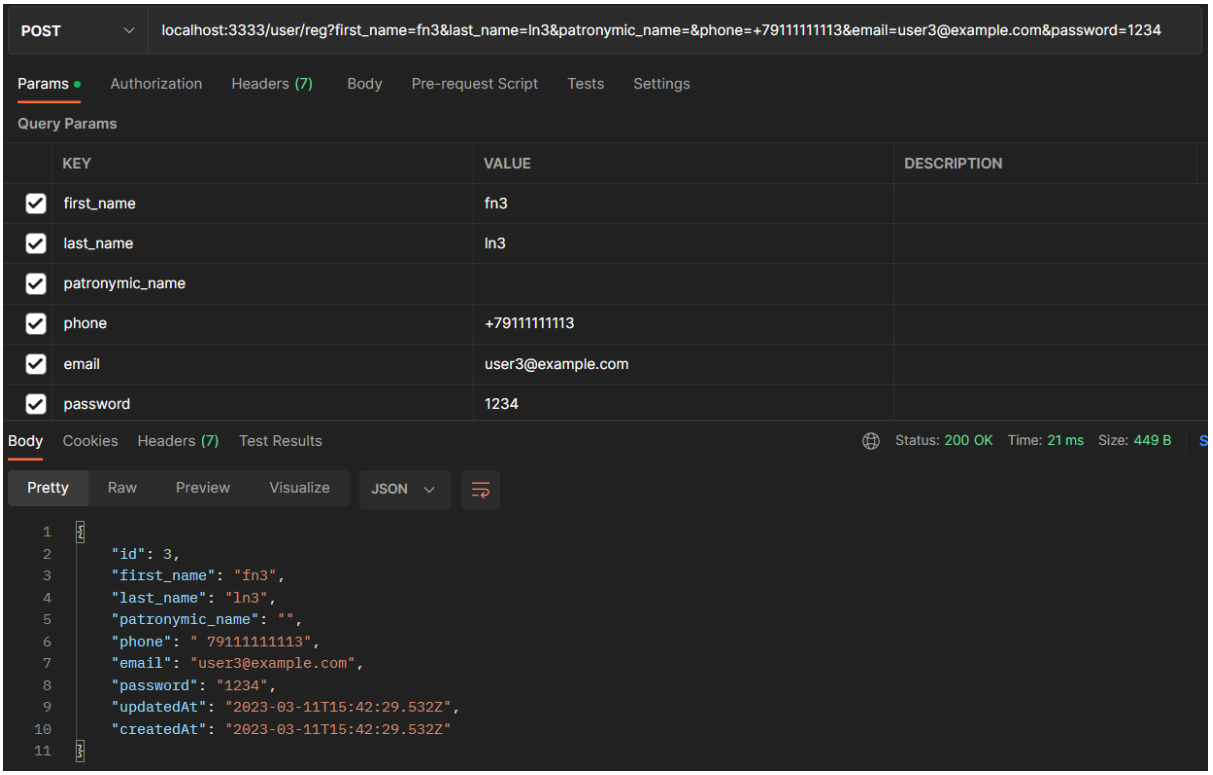

Рисунок 7 - *POST* запрос на создание пользователя

ВЫВОД

В результате работы получил практические навыки по работе с *ORM Sequelize*.