

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Бэк-энд разработка

**Отчет**

**Практическая работа №2: Знакомство с ORM Sequelize**

Выполнил:

Попов Ньургун

К33412

Проверил:

Добряков Д. И.

Санкт-Петербург

2023 г.

## Задача

Нужно выполнить:

- 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

Я выбрал следующие поля:

firstName, lastName, email, password, age

С помощью sequelize-cli создаем модель пользователя:

```
npx sequelize-cli model:generate --name User --attributes
firstName:string,lastName:string,email:string,password:string,
age:string
```

Делаем миграцию:

```
npx sequelize-cli db:migrate
```

```
'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({
    firstName: DataTypes.STRING,
    lastName: DataTypes.STRING,
    email: DataTypes.STRING,
    password: DataTypes.STRING,
    address: DataTypes.STRING,
    age: DataTypes.STRING
  }, {
    sequelize,
    modelName: 'User',
  });
  return User;
};
```

В файле index.js пишем запросы для CRUD-методов:

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

const app = express()
const port = 3000

app.use(express.json())

app.get('/', (req, res) => {
  res.send('Homework 2')
})

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

app.get('/users/:id', async (req, res) => {
  const user = await db.User.findById(req.params.id)

  console.log('user is', user)

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

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

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

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

app.delete('/users/:id', async (req, res) => {
  const user = await db.User.destroy({where: {id: req.params.id}})
  if (user) {
    res.send({'status_code': 'User deleted'})
  } else {
    res.status(404).send({'error_code': 'User not found'})
  }
})

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

- GET запрос на получение всех пользователей

GET http://localhost:3000/users/

Params Authorization Headers (8) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE	DESCRIPTION
--	-----	-------	-------------

Body Cookies Headers (7) Test Results Status: 200 OK Time: 22 ms Size: 848 B

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "id": 1,
4     "firstName": "test1",
5     "lastName": "test1",
6     "email": "test1@example.com",
7     "password": "1234",
8     "address": "nevsky1",
9     "age": "31",
10    "createdAt": "2023-04-05T18:23:19.862Z",
11    "updatedAt": "2023-04-05T18:23:19.862Z"
12  },
13  {
14    "id": 2,
15    "firstName": "test2",
16    "lastName": "test2",
17    "email": "test2@example.com",
18    "password": "1234",
19    "address": "nevsky2",
20    "age": "32",
21    "createdAt": "2023-04-05T18:25:07.155Z",
22    "updatedAt": "2023-04-05T18:25:07.155Z"
23  }
```

- POST запрос на добавление нового пользователя

POST http://localhost:3000/users/

Params Authorization Headers (8) Body Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

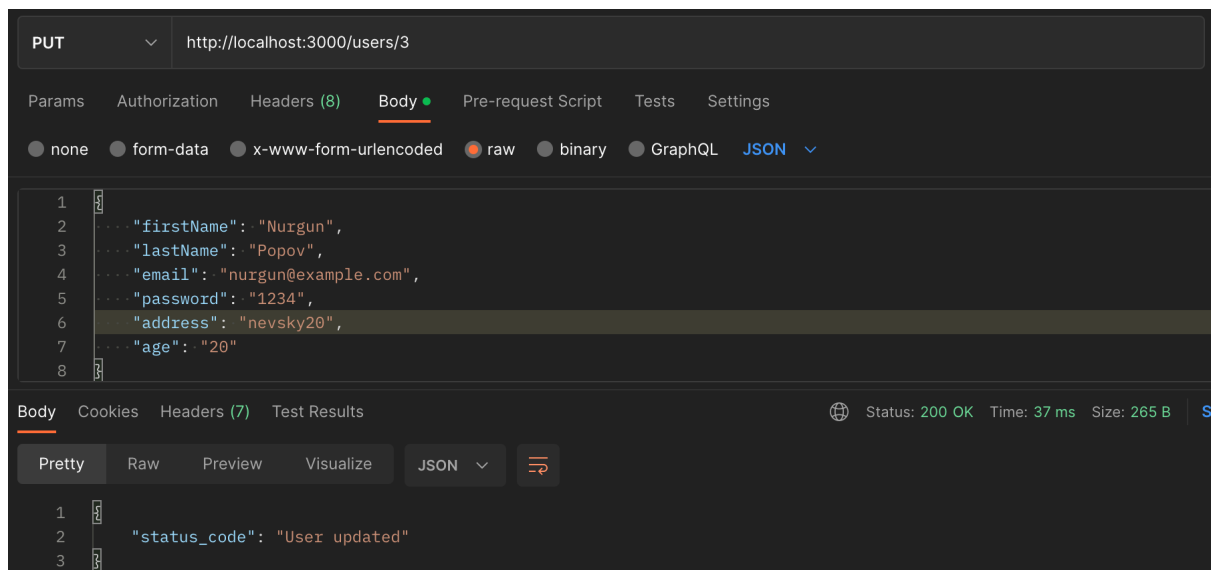
```
1 {
2   "firstName": "Nurgun",
3   "lastName": "Popov",
4   "email": "nurgun@example.com",
5   "password": "1234",
6   "address": "nevsky",
7   "age": "20"
8 }
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 17 ms Size: 439 B

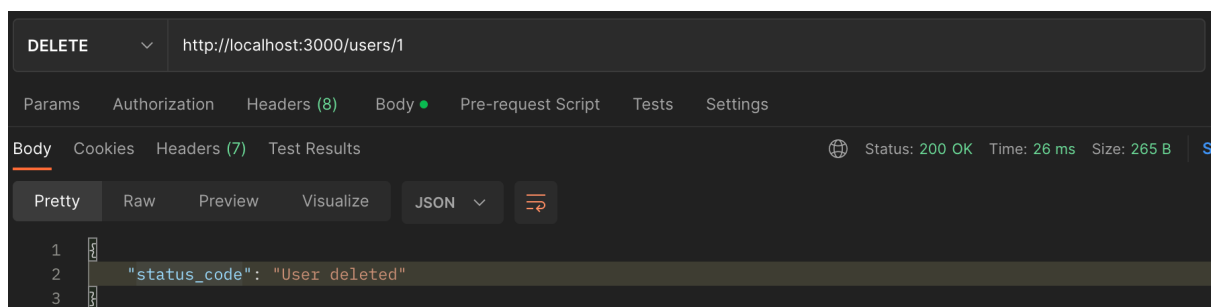
Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 4,
3   "firstName": "Nurgun",
4   "lastName": "Popov",
5   "email": "nurgun@example.com",
6   "password": "1234",
7   "address": "nevsky",
8   "age": "20",
9   "updatedAt": "2023-04-05T18:32:41.675Z",
10  "createdAt": "2023-04-05T18:32:41.675Z"
11 }
```

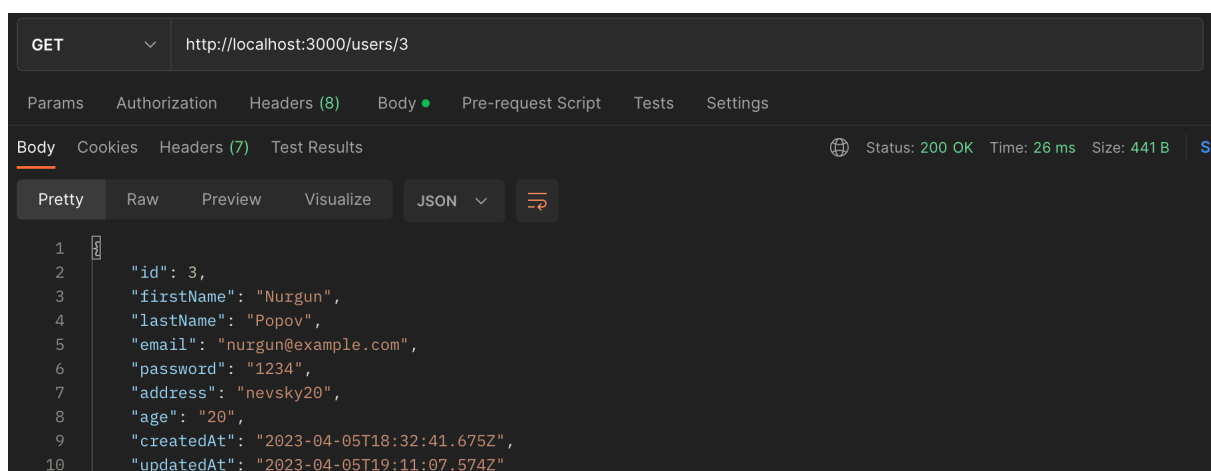
- PUT запрос на изменение информации о конкретном пользователе



- DELETE запрос на удаление конкретного пользователя



- GET запрос на получение пользователя по id



## **Вывод**

В результате выполненной работы: были освоены базовые знания об ORM Sequelize, была создана модель Users, были реализованы наборы из CRUD-методы с помощью фреймворка Express.