

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Практическая работа №2

Выполнил:
Золотов Павел

Группа:
К33401

Проверил:
Добряков Д. И.

Санкт-Петербург

2022 г.

Задача

- 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

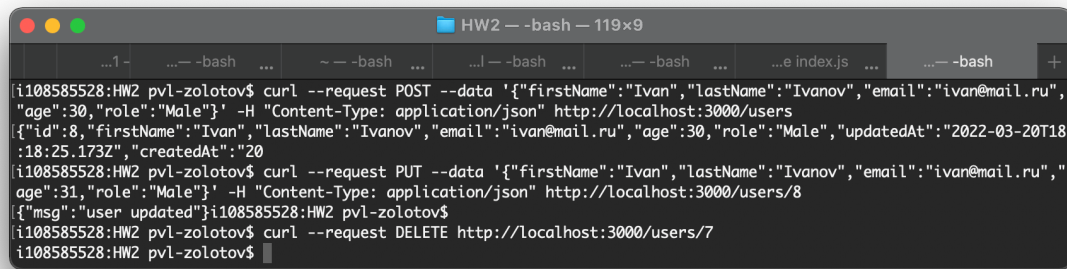
Созданная модель пользователя:

```
1  'use strict';
2  const {
3    Model
4  } = require('sequelize');
5  module.exports = (sequelize, DataTypes) => {
6    class User extends Model {
7      /**
8       * Helper method for defining associations.
9       * This method is not a part of Sequelize lifecycle.
10      * The `models/index` file will call this method automatically.
11      */
12      static associate(models) {
13        // define association here
14      }
15    }
16    User.init({
17      firstName: DataTypes.STRING,
18      lastName: DataTypes.STRING,
19      email: DataTypes.STRING,
20      age: DataTypes.INTEGER,
21      role: DataTypes.ENUM('Male', 'Female')
22    }, {
23      sequelize,
24      modelName: 'User',
25    });
26    return User;
27  };
28
```

CRUD-методы для работы с пользователем:

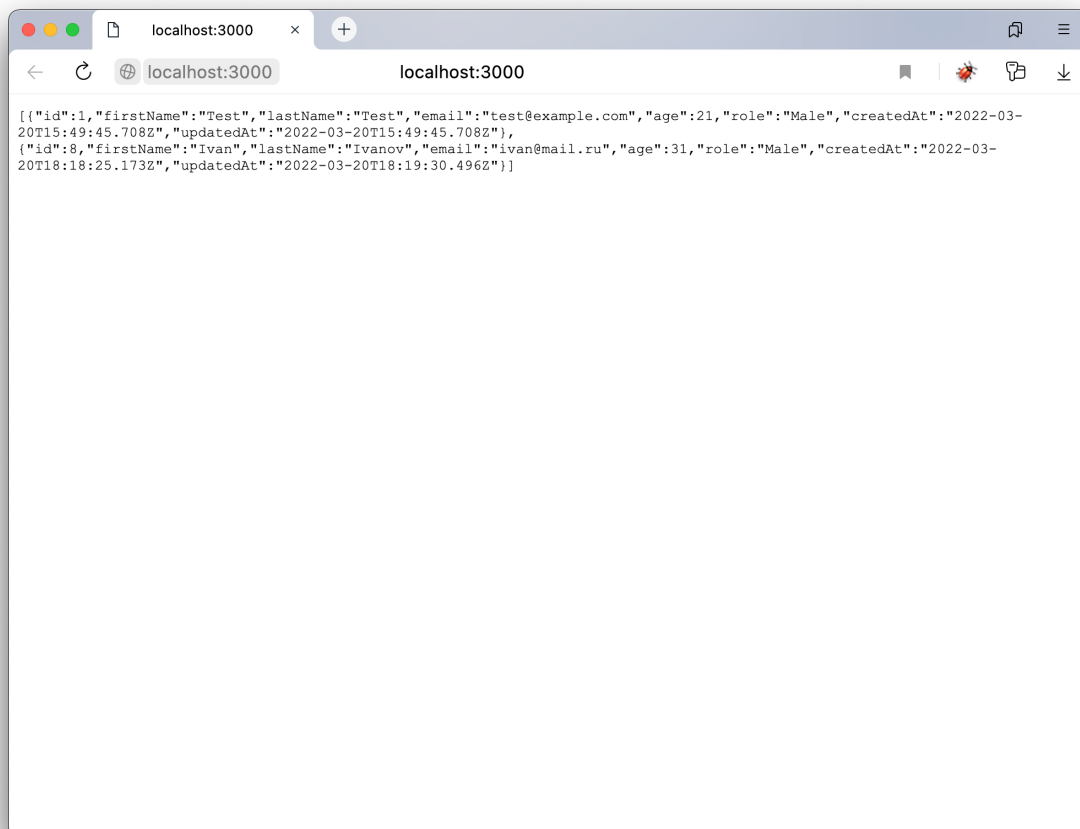
```
1  const express = require('express')
2  const bodyParser = require('body-parser');
3  const db = require('./models')
4
5  const app = express()
6  const port = 3000
7
8  app.use(bodyParser.json());
9
10 app.get('/', async (req, res) => {
11   const users = await db.User.findAll()
12   res.send(users)
13 })
14
15 app.post('/users', async (req, res) => {
16   const user = await db.User.create(req.body)
17   res.send(user.toJSON())
18 })
19
20 app.get('/users/id/:id', async (req, res) => {
21   const user = await db.User.findByPk(req.params.id)
22   if (user) {
23     res.send(user.toJSON())
24   } else {
25     res.status(404).send({"msg": "user is not found"})
26   }
27 })
28
29 app.get('/users/email/:email', async (req, res) => {
30   const user = await db.User.findOne({ where: { email: req.params.email } })
31   if (user) {
32     res.send(user.toJSON())
33   } else {
34     res.status(404).send({"msg": "user is not found"})
35   }
36 })
37
38 app.put('/users/:id', async (req, res) => {
39   const num = await db.User.update(req.body, { where: { id: req.params.id } })
40   if (num == 1) {
41     res.send({"msg": "user updated"})
42   } else {
43     res.status(404).send({"msg": "user is not found"})
44   }
45 })
46
47 app.delete('/users/:id', async (req, res) => {
48   const user = await db.User.destroy({ where: { id: req.params.id } })
49   if (user) {
50     res.send({"msg": "user deleted"})
51   } else {
52     res.status(404).send({"msg": "user is not found"})
53   }
54 })
55
56 app.listen(port, () => {
57   console.log(`App listening on port ${port}`)
58 })
```

Пример работы с API через curl:



```
HW2 -- -bash -- 119x9
[1108585528:HW2 pvl-zolotov$ curl --request POST --data '{"firstName":"Ivan","lastName":"Ivanov","email":"ivan@mail.ru","age":30,"role":"Male"}' -H "Content-Type: application/json" http://localhost:3000/users
{"id":8,"firstName":"Ivan","lastName":"Ivanov","email":"ivan@mail.ru","age":30,"role":"Male","updatedAt":"2022-03-20T18:18:25.173Z","createdAt":"2022-03-20T18:18:25.173Z"}
[1108585528:HW2 pvl-zolotov$ curl --request PUT --data '{"firstName":"Ivan","lastName":"Ivanov","email":"ivan@mail.ru","age":31,"role":"Male"}' -H "Content-Type: application/json" http://localhost:3000/users/8
{"msg":"user updated"}
[1108585528:HW2 pvl-zolotov$ curl --request DELETE http://localhost:3000/users/7
[1108585528:HW2 pvl-zolotov$
```

Пример работы с API через браузер:



Вывод

В результате выполнения работы были получены навыки создания простого API с помощью микрофреймворка Express и ORM Sequelize.