

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа 2

Выполнил:

Трофимов Андрей

Группа К33402

Проверил

: Добряков Д.

И.

Санкт-Петербург

2024 г.

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/email

Ход работы

1. Инициализируем модуль: *npm init*

```
> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (test)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /Users/rybalkooleg/ITMO/ITMO-ICT-Backend-2024/homeworks/K33392/Рыбалко_Олег/test/package.json:
{
  "name": "test",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes) yes
```

2. Установим зависимости:

npm i express sequelize sqlite3 sequelize-cli

3. Инициализируем sequelize: *npx sequelize init*

4. Сгенерируем модель при помощи sequelize-cli

npx sequelize-cli model:generate --name User --attributes "username:string, email:string, password:string, firstName:string, lastName:string, isAdmin:boolean"

5. Проведем миграцию

6. Создадим endpoint для получения всех пользователей

```
// Get all users
app.get('/users', async (req, res) => {
  try {
    res.json(await db.User.findAll())
  } catch (error) {
    res.status(500).json({ error: error.message })
  }
})
```

7. Создадим endpoint для создания нового пользователя

```
5 // Create a new user
6 app.post('/users', async (req, res) => {
7   try {
8     const user = await db.User.create(req.body)
9     res.status(200).json(user)
10  } catch (error) {
11    res.status(400).json({ error: error.message })
12  }
13 })
```

8. Создадим endpoint для получения пользователя по id

```
24 // Get user by ID
25 app.get('/users/:id', async (req, res) => {
26   try {
27     const user = await db.User.findByPk(req.params.id)
28     if (!user) {
29       res.status(404).json({ error: 'User not found' })
30     } else {
31       res.json(user)
32     }
33   } catch (error) {
34     res.status(500).json({ error: error.message })
35   }
36 })
```

9. Создадим endpoint для изменения данных пользователя

```

38 // Update user by ID
39 app.patch('/users/:id', async (req, res) => {
40   try {
41     const user = await db.User.findByPk(req.params.id)
42     if (!user) {
43       res.status(404).json({ error: 'User not found' })
44     } else {
45       await user.update(req.body)
46       res.json(user)
47     }
48   } catch (error) {
49     res.status(500).json({ error: error.message })
50   }
51 })

```

10. Создадим endpoint для удаление пользователя

```

53 // Delete user by ID
54 app.delete('/users/:id', async (req, res) => {
55   try {
56     const user = await db.User.findByPk(req.params.id)
57     if (!user) {
58       res.status(404).json({ error: 'User not found' })
59     } else {
60       await user.destroy()
61       res.status(200).send()
62     }
63   } catch (error) {
64     res.status(500).json({ error: error.message })
65   }
66 })

```

Вывод

В данной домашней работе удалось написать HTTP сервер, который обрабатывает запросы для CRUD-операций над пользователем при помощи библиотеки `express` и `sequelize` для работы с базой данных.