

Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего
образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет
по домашней работе №2
по дисциплине «Бэкенд-разработка»

Выполнил: Холодов-Воронцов А. А.

Факультет: Инфокоммуникационных технологий

Группа: K33412

Проверил: Добряков Д. И.



Санкт-Петербург 2023

Задание

- 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

1. index.js

```
Kholodov-Vorontsov HW2 createUser.js
1 const express = require('express')
2 const db = require('./models')
3
4 const bp = require('body-parser')
5 const app = express()
6 const port = 3000
7
8 app.use(bp.json())
9 app.use(bp.urlencoded({ extended: true }))
10
11
12 app.get('/', (req, res) => {
13   res.send('Hello')
14 })
15
16 app.get('/users', async (req, res) => {
17   const user = await db.User.findAll()
18   console.log('user is', user)
19   res.send(user)
20 })
21
22
23 app.get('/users/email/:email', async (req, res) => {
24   const user = await db.User.findOne({ where: { email: req.params.email } })
```

```
ov-Vorontsov HW2 createUser.js
23 app.get('/users/email/:email', async (req, res) => {
24   const user = await db.User.findOne({ where: { email: req.params.email } })
25   console.log('user is', user)
26   res.send(user.toJSON())
27 })
28
29 app.get('/users/:id', async (req, res) => {
30   const user = await db.User.findById(req.params.id)
31   console.log('user is', user)
32   res.send(user.toJSON())
33 })
34
35 app.get('/users/delete/:id', async (req, res) => {
36   const user = await db.User.destroy({
37     where: {
38       id: req.params.id
39     }
40   })
41   console.log('user was deleted!', user)
42   res.send('user was deleted!')
43 })
44
45 app.put('/users/update/:id', async (req, res) => {
46   db.User.findOne({ where: { id: req.params.id } }) ...
```

```
ov-Vorontcov HW2 createUser.js start Git
models\index.js HW2\index.js createUser.js package.json config.json user.js
45 app.put('/users/update/:id', async (req, res) => {
46   db.User.findOne({ where: {id: req.params.id}}) ...
47   .then(record => {
48     if (!record) {
49       throw new Error('No record found')
50     }
51     let values = {
52       firstName: req.body.firstName,
53       lastName: req.body.lastName,
54       email: req.body.email,
55     }
56
57     record.update(values).then( updatedRecord => {
58       // log into your DB and confirm update
59     })
60
61   }) Promise<T>
62   .catch((error) => {
63     //error
64     throw new Error(error)
65   })
66   console.log('user was update')
67   res.send('user was update')
68 }
```

```
68
69
70
71 app.post ('/create/', async (req,res) => {
72   await db.User.create(req.body)
73   console.log('user was create')
74   res.send('user was create')
75 })
76
77
78 app.listen(port, () => {
79   console.log('Example app listening on port ${port}')
80 })
```

2. createUser.js

```
models\index.js HW2\index.js createUser.js package.json config.json user.js
1 const db = require('./models')
2
3 async function main() {
4   await db.User.create({
5     firstName: 'Test',
6     lastName: 'Testov',
7     email: 'test@test.com'
8   })
9
10 }
11
12 main()
```

3. Создание пользователя:

The screenshot shows a REST client interface with a POST request to `http://127.0.0.1:3000/create/`. The request body is a JSON object: `{ "firstName": "Sasha", "lastName": "Kholodov", "email": "test@aleksandr.com" }`. The response status is 200 OK, with a response time of 31 ms and a body size of 242 B. The response body is displayed in the 'Body' tab, showing the text `user was create`.

```
1 {
2   "firstName": "Sasha",
3   "lastName": "Kholodov",
4   "email": "test@aleksandr.com"
5 }
```

200 OK 31 ms 242 B Save Response

1 user was create

4. Получение пользователя по id:

The screenshot shows a REST client interface with a GET request to `http://127.0.0.1:3000/users/4`. The response status is 200 OK, with a response time of 9 ms and a body size of 393 B. The response body is displayed in the 'Body' tab, showing a JSON object with user details.

```
1 {
2   "id": 4,
3   "firstName": "Sasha",
4   "lastName": "Kholodov",
5   "email": "test@aleksandr.com",
6   "createdAt": "2023-03-14T14:11:41.772Z",
7   "updatedAt": "2023-03-14T14:11:41.772Z"
8 }
```

200 OK 9 ms 393 B Save Response

5. Обновление данных пользователя по id:

The screenshot shows a REST client interface with a PUT request to `http://127.0.0.1:3000/users/update/4`. The request body is a JSON object: `{ "firstName": "Aleksandr", "lastName": "Kholodov", "email": "test@aleksandr.com" }`. The response is a 200 OK status with a response time of 8 ms and a body of 242 B. The response body is displayed as `1 user was update`.

```
PUT http://127.0.0.1:3000/users/update/4
```

```
{
  "firstName": "Aleksandr",
  "lastName": "Kholodov",
  "email": "test@aleksandr.com"
}
```

200 OK 8 ms 242 B

```
1 user was update
```

6. Удаление пользователя по id:

The screenshot shows a REST client interface with a GET request to `http://127.0.0.1:3000/users/delete/4`. The request body is a JSON object: `{ "firstName": "Aleksandr", "lastName": "Kholodov", "email": "test@aleksandr.com" }`. The response is a 200 OK status with a response time of 27 ms and a body of 245 B. The response body is displayed as `1 user was deleted!`.

```
GET http://127.0.0.1:3000/users/delete/4
```

```
{
  "firstName": "Aleksandr",
  "lastName": "Kholodov",
  "email": "test@aleksandr.com"
}
```

200 OK 27 ms 245 B

```
1 user was deleted!
```

Вывод

В ходе выполнения домашнего задания я научился создавать, получать, обновлять и удалять пользователей с помощью Node.js, используя express и sequelize.