

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина:** Бэк-энд разработка

Отчет

Домашняя работа №2

Выполнил:  
Вахрушева Ксения

Группа:  
К33412

Проверил:  
Добряков Д. И.

Санкт-Петербург

2023 г.

## 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 задания:

### 1) Модель пользователя:

```
1 //lets says if we have user table in our db, so we are creating user model
2
3
4 const {Model, DataTypes} = require('sequelize');
5 const sequelize = require('./database');
6
7 class User extends Model{}
8
9
10 //here initilizing the user and it takes to two parameters
11 // entities of db and second our database name
12 User.init({
13   username: {
14     type:DataTypes.STRING
15   },
16   password:{
17     type:DataTypes.STRING
18   },
19   email:{
20     type:DataTypes.STRING
21   }
22 }, {
23   sequelize,
24   modelName: 'user', // this is to give name to our db tabel
25   timestamps: false // for not showing timing in our table
26 })
27
28 module.exports = User;
```

## CRUD-методы:

```
1  const express = require("express");
2
3  //for important our instance file we do as below
4  const sequelize = require('./database');//importing the file
5  const User = require('./User');
6  //for running our database we do as below
7  sequelize.sync({force: true}).then(()=> console.log("db is ready"))
8  //force: true. is used here to delete old data from table and create table again every time
9
10 const app = express()
11
12 app.use(express.json()); // this to make express aware what data is coming. unless it will put null in
13 app.post('/users', async (req,res)=>{
14     await User.create(req.body)
15     res.send("User is inserted")
16
17 });
18 //for getting all users in table
19 app.get('/users', async (req, res)=>{
20     const myuser = await User.findAll()
21     res.send(myuser);
22 })
23 //for getting specific user in table
24 app.get('/users/:id',async (req,res)=>{
25     const requestedId = req.params.id;
26     const requestedUser = await User.findOne({where: {id: requestedId} })
27     res.send(requestedUser)
28 })
```

## 2) Создание:

POST ▾ localhost:4000/users

Send ▾

JSON ▾

Auth ▾

Query

Headers 1

Docs

1 {

2   "username": "ксения",

3   "password": "ксепаpоль",

4   "email": "ксения@gmail.com"

5 }

Beautify JSON

200 OK

41.4 ms

16 B

Just Now ▾

Preview ▾

Headers 7

Cookies

Timeline

User is inserted

3) Чтение:

The screenshot shows a web client interface with a dark theme. At the top, the request method is 'GET' and the URL is 'localhost:4000/users'. A 'Send' button is on the right. Below the URL bar, there are tabs for 'Body', 'Auth', 'Query', 'Headers', and 'Docs'. The 'Body' tab is selected, showing a large grey bug icon and the text 'Enter a URL and send to get a response' and 'Select a body type from above to send data in the body of a request'. Below this, a status bar shows '200 OK' in a green box, '5.38 ms' in a grey box, '101 B' in a grey box, and '2 Minutes Ago' with a dropdown arrow. At the bottom, there are tabs for 'Preview', 'Headers', 'Cookies', and 'Timeline'. The 'Preview' tab is selected, displaying a JSON response with line numbers 1 through 8. The JSON contains a single user object with fields 'id', 'username', 'password', and 'email'.

GET ▾ localhost:4000/users Send ▾

Body ▾ Auth ▾ Query Headers Docs

Enter a URL and send to get a response

Select a body type from above to send data in the body of a request

200 OK 5.38 ms 101 B 2 Minutes Ago ▾

Preview ▾ Headers 7 Cookies Timeline

```
1 ▾ [
2 ▾  {
3     "id": 1,
4     "username": "ксения",
5     "password": "ксепароль",
6     "email": "ксения@gmail.com"
7   }
8 ]
```

GET ▼ localhost:4000/users/1 Send ▼

Body ▼ Auth ▼ Query Headers Docs

Enter a URL and send to get a response

Select a body type from above to send data in the body of a request

200 OK 21.1 ms 99 B Just Now ▼

Preview ▼ Headers <sup>7</sup> Cookies Timeline

```
1 {  
2   "id": 1,  
3   "username": "ксения",  
4   "password": "ксепароль",  
5   "email": "ксения@gmail.com"  
6 }
```

4) Изменение:

PUT ▼ localhost:4000/users/1

Send ▼

JSON ▼

Auth ▼

Query

Headers 1

Docs

1 ▼ {

2   "username": "ксения\_1"

3   }

Beautify JSON

200 OK

30.9 ms

7 B

Just Now ▼

Preview ▼

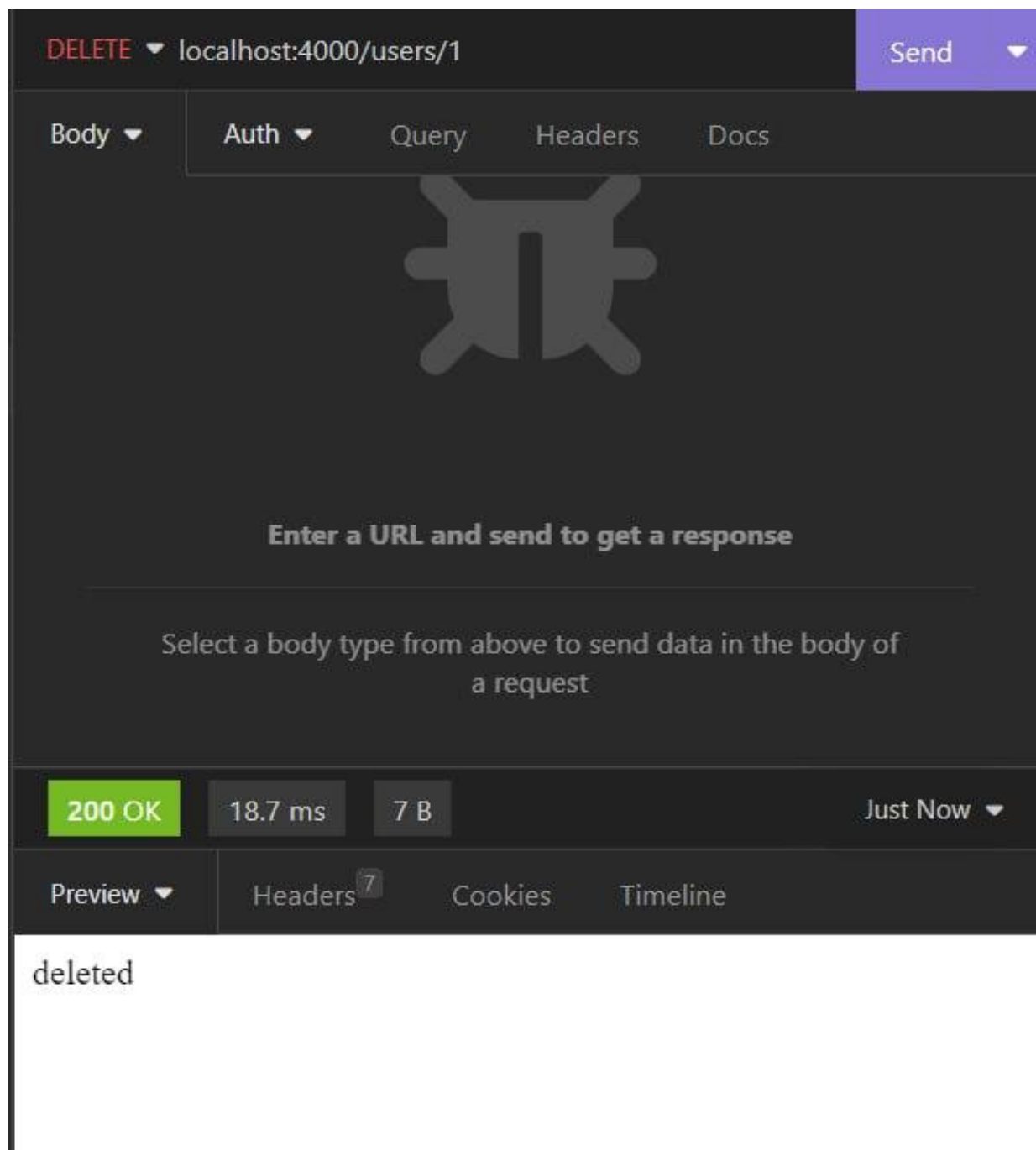
Headers 7

Cookies

Timeline

Updated

5) Удаление:



### Вывод:

В ходе выполнения задания была придумана собственная модель пользователя, реализован набор из CRUD-методов для работы с пользователями средствами Express и Sequelize, написан запрос для получения пользователя по id/email.