

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бек-энд разработка

Отчет

Домашняя работа №2

Выполнил:

Кобелев Л.К.

К33401

Проверил:

Добряков Д. И.

Санкт-Петербург

2023 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

Инициализируем проект:

```
npm init
```

Устанавливаем express, sequelize, sequelize cli и sqlite:

```
npm install express -S
```

```
npm install sequelize -S
```

```
npm install --save-dev sequelize-cli
```

```
npm install sqlite3 -S
```

Инициализируем:

```
npx sequelize init
```

Создадим модель пользователя, которая будет состоять из имени, почты и пароля:

```
npx sequelize-cli model:generate --name User --attributes  
username:string,email:string,password:string
```

Делаем миграцию, предварительно поменяв в конфиге mysql на sqlite и добавив хранилище:

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

Создаем index.js, который будет выступать сервером:

```
const express = require('express')  
const db = require('./models')  
const bodyParser = require('body-parser')  
  
const app = express()  
const port = 8080  
  
app.use(bodyParser.json())  
app.use(express.urlencoded({ extended: true }));  
  
app.listen(port, () => {  
  console.log(`App listening on port ${port}`)  
})
```

Реализуем CRUD операции для модели User:

Код

Create User:

```
app.post('/users/create', async (req, res) => {
  try {
    const user = await db.User.create(req.body);
    return res.send({"msg": "New user: " + user.username});
  } catch (error) {
    return res.send({"msg": error});
  }
});
```

Get User by PK:

```
app.get('/users/:id', async (req, res) => {
  const user = await db.User.findById(req.params.id)
  if (user) {
    return res.send(user.toJSON())
  }
  return res.send({"msg": "User does not exist"})
});
```

Get User by e-mail:

```
app.get('/users', async (req, res) => {
  let users = await db.User.findAll()
  if (req.query.email) {
    users = users.find(user => {
      return user.email === req.query.email
    })
  }
  return res.send(users)
});
```

Проверка

POST http://127.0.0.1:8080/users/create

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

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> username	user01
<input checked="" type="checkbox"/> email	test@mail.ru
<input checked="" type="checkbox"/> password	password

Body Cookies Headers (7) Test Results Status

Pretty Raw Preview Visualize JSON

```
1 {
2   "msg": "New user: user01"
3 }
```

GET http://127.0.0.1:8080/users/6

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

none form-data x-www-form-urlencoded raw binary GraphQL

KEY	VALUE
<input checked="" type="checkbox"/> username	user02
Key	Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 6,
3   "username": "user01",
4   "email": "test@mail.ru",
5   "password": "password",
6   "createdAt": "2023-03-13T18:00:36.357Z",
7   "updatedAt": "2023-03-13T18:00:36.357Z"
8 }
```

GET http://127.0.0.1:8080/users?email=test2@test.com

Params **Authorization** Headers (6) Body Pre-request Script Tests

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 14,
3   "username": "user02",
4   "email": "test2@test.com",
5   "password": "password",
6   "createdAt": "2023-03-13T18:58:14.374Z",
7   "updatedAt": "2023-03-13T18:58:14.374Z"
8 }
```

Get All Users:

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

GET http://127.0.0.1:8080/users

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

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

KEY	VALUE
Body Cookies Headers (7) Test Results	
Pretty Raw Preview Visualize JSON	
<pre>1 [2 { 3 "id": 7, 4 "username": "user01", 5 "email": "test@mail.ru", 6 "password": "password", 7 "createdAt": "2023-03-13T18:04:19.192Z", 8 "updatedAt": "2023-03-13T18:04:19.192Z" 9 }, 10 { 11 "id": 8, 12 "username": "user02", 13 "email": "test@mail.test", 14 "password": "password", 15 "createdAt": "2023-03-13T18:37:13.545Z", 16 "updatedAt": "2023-03-13T18:37:13.545Z" 17 } 18]</pre>	

Update User:

```
app.put('/users/:id', async (req, res) => {
  const user = await db.User.findByPk(req.params.id);
  if (user) {
    try {
      user.update(req.body);
      return res.send({"msg": "User updated: " + user.username});
    } catch (error) {
      return res.send({"msg": error});
    }
  }
  return res.send({"msg": "User does not exist"})
});
```

PUT http://127.0.0.1:8080/users/8

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

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

KEY	VALUE
<input checked="" type="checkbox"/> username	user03

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "msg": "User updated: user03"
3 }
```

Delete User:

```
app.delete('/users/:id', async (req, res) => {
  const user = await db.User.destroy({where: {id: req.params.id}})
  if (user) {
    return res.send({"msg": "User is deleted"})
  }
  return res.send({"msg": "User does not exist"})
});
```

DELETE http://127.0.0.1:8080/users/12

Params Authorization Headers (6) Body Pre-request Script Tests

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

KEY	VALUE
Body Cookies Headers (7) Test Results	
Pretty Raw Preview Visualize JSON	
<pre>1 { 2 "msg": "User is deleted" 3 }</pre>	

Вывод

Были опробованы микрофреймворк Express и ORM Sequelize.