

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Практическая работа 2

Выполнила:

Афанасьева Ирина Максимовна

Группа:

К33402

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 работы

1. Создана модель пользователя, включающая в себя следующие поля:

- Фамилия
- Имя
- Отчество
- Email
- Дата рождения

Создание модели:

```
PS C:\Users\Irina\Desktop\Семинары\Бекендеки\Нw2> npx sequelize-cli model:generate --name User --attributes firstName:string,lastName:string,fatherName:string,email:string,dateOfBirth:date

Sequelize CLI [Node: 18.15.0, CLI: 6.6.0, ORM: 6.31.1]

New model was created at C:\Users\Irina\Desktop\Семинары\Бекендеки\Нw2\models\user.js .
New migration was created at C:\Users\Irina\Desktop\Семинары\Бекендеки\Нw2\migrations\20230528175557-create-user.js .
```

Модель пользователя:

```
User.init({
  firstName: DataTypes.STRING,
  lastName: DataTypes.STRING,
  fatherName: DataTypes.STRING,
  email: DataTypes.STRING,
  dateOfBirth: DataTypes.DATE
}, {
  sequelize,
  modelName: 'User',
});
return User;
```

2. Реализованы запросы CRUD методов с помощью фреймворка Express и Sequelize.
 - Get/users – список всех пользователей

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

Body Cookies Headers (7) Test Results

Pretty

Raw

Preview

Visualize

JSON

⌵

```
1  [
2    {
3      "id": 1,
4      "firstName": "Irina",
5      "lastName": "Afanasieva",
6      "fatherName": "Maksimovna",
7      "email": "multifructina@gmail.com",
8      "dateOfBirth": "2005-12-15T00:00:00.000Z",
9      "createdAt": "2023-05-28T20:37:59.763Z",
10     "updatedAt": "2023-05-28T21:28:15.127Z"
11   },
12   {
13     "id": 2,
14     "firstName": "Ksenia",
15     "lastName": "Afanasieva",
16     "fatherName": "Maksimovna",
17     "email": "nena@gmail.com",
18     "dateOfBirth": "2001-04-10T00:00:00.000Z",
19     "createdAt": "2023-05-28T20:53:35.565Z",
20     "updatedAt": "2023-05-28T21:28:44.248Z"
21   }
22 ]
```

- Post/users – создание пользователя

```
app.post('/users', async (req, res) => {
  const user = await db.User.create(req.body)
  res.send(user.toJSON())
})
```

POST

⌵

http://127.0.0.1:3000/users/

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

JSON ⌵

```
1  {
2    "firstName": "Gelya",
3    "lastName": "Ivanova",
4    "fatherName": "Nikolaevna",
5    "email": "loli@gmail.com",
6    "dateOfBirth": "2001-11-06"
7  }
```

- Get/users/:id – пользователь по id

```
app.get('/users/:id', async (req, res) => {
  const user = await db.User.findByPk(req.params.id)
  if (user) {
    res.send(user.toJSON())
  } else {
    res.status(404).send({ 'msg': 'user not found' })
  }
})
```

The screenshot shows a REST client interface. At the top, a GET request is defined for the URL `http://127.0.0.1:3000/users/9`. Below the URL bar, there are tabs for Params, Authorization, Headers (8), Body, and Pre-request Scripts. The 'Body' tab is selected, and within it, there are sub-tabs for Body, Cookies, Headers (7), and Test Results. The 'Body' sub-tab is active, showing a JSON response in 'Pretty' format. The response is a JSON object with the following fields: `id` (9), `firstName` (Gelya), `lastName` (Ivanova), `fatherName` (Nikolaevna), `email` (loli@gmail.com), `dateOfBirth` (2001-11-06T00:00:00.000Z), `createdAt` (2023-05-28T21:31:32.218Z), and `updatedAt` (2023-05-28T21:31:32.218Z).

```
GET http://127.0.0.1:3000/users/9
```

Params Authorization Headers (8) Body ● Pre-request Scripts

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON ↕

```
1 {
2   "id": 9,
3   "firstName": "Gelya",
4   "lastName": "Ivanova",
5   "fatherName": "Nikolaevna",
6   "email": "loli@gmail.com",
7   "dateOfBirth": "2001-11-06T00:00:00.000Z",
8   "createdAt": "2023-05-28T21:31:32.218Z",
9   "updatedAt": "2023-05-28T21:31:32.218Z"
10 }
```

- Put/users/:id – редактирование пользователя по id

```
app.put('/users/:id', async (req, res) => {
  const num = await db.User.update(req.body, { where: { id: req.params.id } })
  if (num == 1) {
    res.send({ 'msg': 'user has been updated' })
  } else {
    res.status(404).send({ 'msg': 'user not found' })
  }
})
```

PUT http://127.0.0.1:3000/users/9

Params Authorization Headers (8) **Body** Pre-reqs

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

1 {

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

1 {
2 "msg": "user has been updated"
3 }

- Delete/users/:id – удаление пользователя по id

```
app.delete('/users/:id', async (req, res) => {  
  const user = await db.User.destroy({ where: { id: req.params.id } })  
  if (user) {  
    res.send({ 'msg': 'user deleted' })  
  } else {  
    res.status(404).send({ 'msg': 'user not found' })  
  }  
})
```

DELETE http://127.0.0.1:3000/users/9

Params Authorization Headers (8) **Body**

☐ none ☐ form-data ☐ x-www-form-urlencoded

1 {

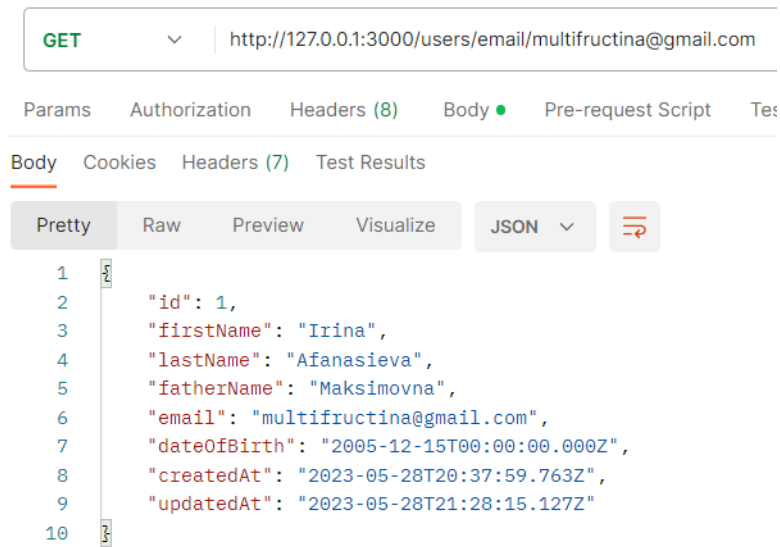
Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

1 {
2 "msg": "user deleted"
3 }

3. Реализован метод получения пользователя по email.

```
app.get('/users/email/:email', async (req, res) => {
  const user = await db.User.findOne({ where: { email: req.params.email } })
  if (user) {
    res.send(user.toJSON())
  } else {
    res.status(404).send({ 'msg': 'user not found' })
  }
})
```



Вывод

В ходе работы был освоен фреймворк Express и ORM Sequelize, основы работы с моделями, создание CRUD-методов и фильтров.