

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение высшего  
образования  
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**  
Факультет инфокоммуникационных технологий

**ОТЧЕТ О ДОМАШНЕЙ РАБОТЕ № 2**  
по теме: Знакомство с ORM Sequelize  
по дисциплине: Бэк-энд разработка

Специальность:

09.03.03 Мобильные и сетевые технологии

Выполнил:

Самаров И.И., К33402

Проверил:

Добряков Д.И.

Санкт-Петербург,

2024

## Задание.

- Придумать 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.

После установки необходимых библиотек была сгенерирована модель пользователя следующего вида:

- username
- email
- role

с помощью команды

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

```
PS D:\hw2\src> npx sequelize-cli model:generate --name User --attributes username:string,email:string,role:string  
Sequelize CLI [Node: 20.10.0, CLI: 6.6.2, ORM: 6.37.1]  
  
New model was created at D:\hw2\src\models\user.js .  
New migration was created at D:\hw2\src\migrations\20240330012512-create-user.js .  
PS D:\hw2\src>
```

## Результат:

```
'use strict';  
const {  
  Model  
} = require('sequelize');  
module.exports = (sequelize, DataTypes) => {  
  class User extends Model {  
    /**  
     * Helper method for defining associations.  
     * This method is not a part of Sequelize lifecycle.  
     * The `models/index` file will call this method automatically.  
     */  
    static associate(models) {  
      // define association here  
    }  
  }  
  User.init({  
    username: DataTypes.STRING,  
    email: DataTypes.STRING,  
    role: DataTypes.STRING  
  }, {  
    sequelize,  
    modelName: 'User',  
  });  
  return User;  
};
```

Далее был реализован контроллер, вот то, что нас наиболее интересует:

createUser:

```
const UserController = {
  createUser: async (req, res) => {
    try {
      const newUser = await User.create(req.body);
      res.status(201).json(newUser);
    } catch (error) {
      res.status(400).json({ message: error.message });
    }
  },
};
```

getUserBy Id/Email:

```
getUserById: async (req, res) => {
  const { id } = req.params;
  try {
    const user = await User.findPk(id);
    if (!user) {
      return res.status(404).json({ message: "User not found" });
    }
    res.json(user);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
},

getUserByEmail: async (req, res) => {
  const { email } = req.params;
  try {
    const user = await User.findOne({ where: { email } });
    if (!user) {
      return res.status(404).json({ message: "User not found" });
    }
    res.json(user);
  } catch (error) {
    res.status(500).json({ message: error.message });
  }
},
};
```

Ну и, разумеется, роутер:

```
const express = require("express");
const router = express.Router();
const UserController = require("../controllers/user");

router.post("/create", UserController.createUser);

router.get("/", UserController.getAllUsers);

router.get("/:id", UserController.getUserById);

router.put("/:id", UserController.updateUser);

router.delete("/:id", UserController.deleteUser);

router.get("/email/:email", UserController.getUserByEmail);

module.exports = router;
```

Проверим нашу модель на практике при помощи Postman

Post-запрос:

The screenshot shows the Postman interface for a POST request. The URL bar shows `http://localhost:3000/users/create` and the method is set to `POST`. The `Body` tab is selected, showing a JSON body with the following content:

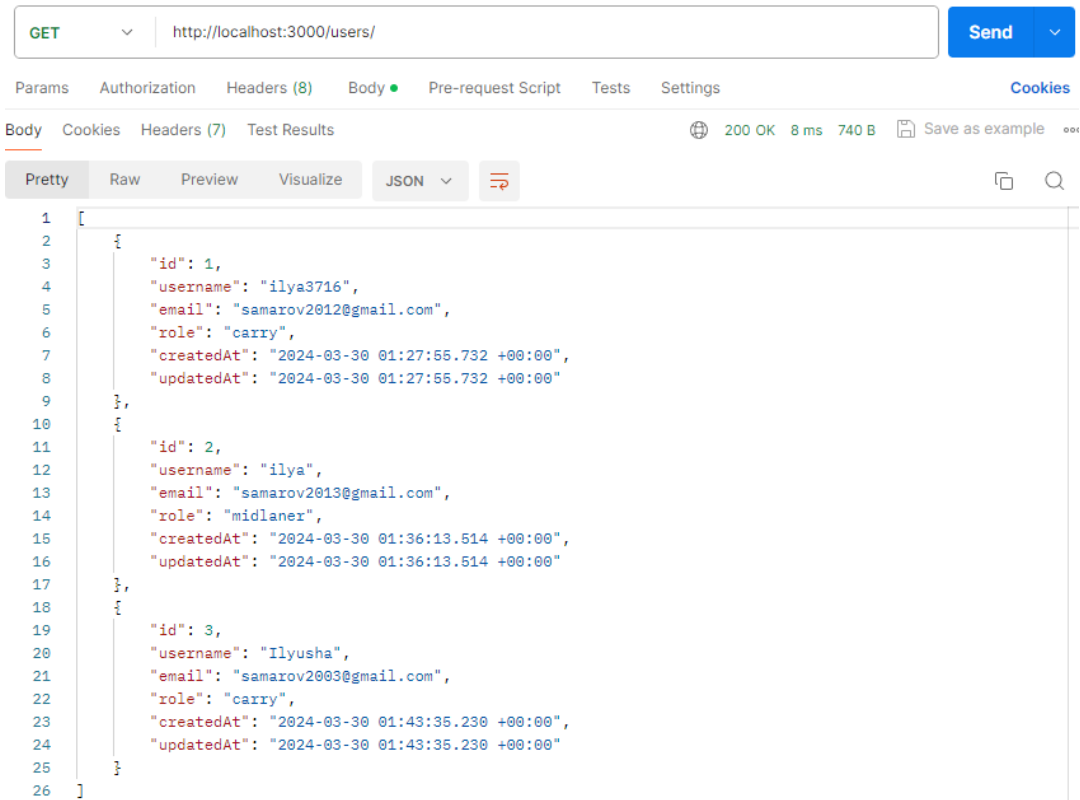
```
1 {
2   "username": "Ilyusha",
3   "email": "samarov2003@gmail.com",
4   "role": "carry"
5 }
```

Below the request body, the `Body` tab of the response is selected, showing the JSON response in `Pretty` format:

```
1 {
2   "id": 3,
3   "username": "Ilyusha",
4   "email": "samarov2003@gmail.com",
5   "role": "carry",
6   "updatedAt": "2024-03-30T01:43:35.230Z",
7   "createdAt": "2024-03-30T01:43:35.230Z"
8 }
```

At the bottom of the interface, status information is displayed: `201 Created`, `91 ms`, and `395 B`. There is also a `Save as example` button.

## Get-запрос по всем пользователям:



GET http://localhost:3000/users/ Send

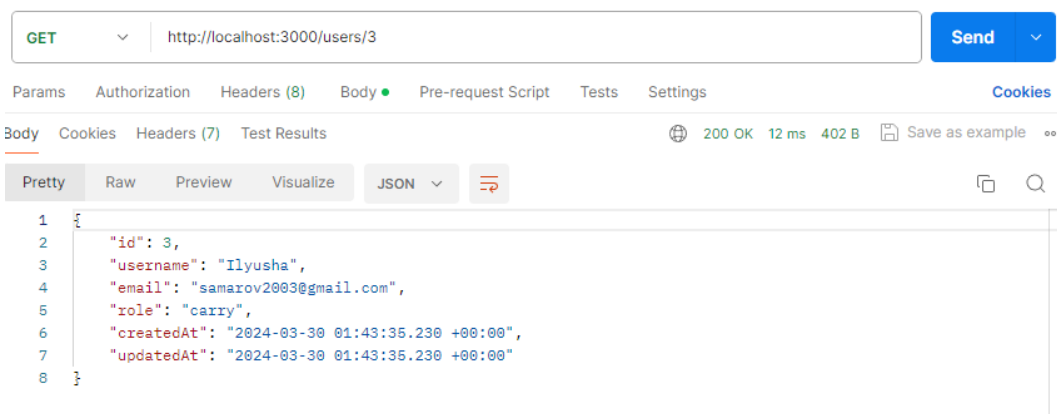
Params Authorization Headers (8) Body • Pre-request Script Tests Settings Cookies

Body Cookies Headers (7) Test Results 200 OK 8 ms 740 B Save as example

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "username": "ilya3716",
5     "email": "samarov2012@gmail.com",
6     "role": "carry",
7     "createdAt": "2024-03-30 01:27:55.732 +00:00",
8     "updatedAt": "2024-03-30 01:27:55.732 +00:00"
9   },
10  {
11    "id": 2,
12    "username": "ilya",
13    "email": "samarov2013@gmail.com",
14    "role": "midlaner",
15    "createdAt": "2024-03-30 01:36:13.514 +00:00",
16    "updatedAt": "2024-03-30 01:36:13.514 +00:00"
17  },
18  {
19    "id": 3,
20    "username": "Ilyusha",
21    "email": "samarov2003@gmail.com",
22    "role": "carry",
23    "createdAt": "2024-03-30 01:43:35.230 +00:00",
24    "updatedAt": "2024-03-30 01:43:35.230 +00:00"
25  }
26 ]
```

## Get-запрос по id:



GET http://localhost:3000/users/3 Send

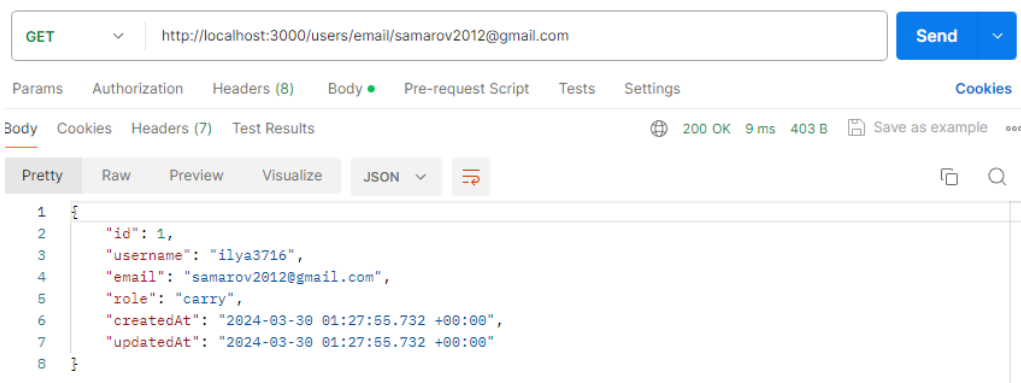
Params Authorization Headers (8) Body • Pre-request Script Tests Settings Cookies

Body Cookies Headers (7) Test Results 200 OK 12 ms 402 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 3,
3   "username": "Ilyusha",
4   "email": "samarov2003@gmail.com",
5   "role": "carry",
6   "createdAt": "2024-03-30 01:43:35.230 +00:00",
7   "updatedAt": "2024-03-30 01:43:35.230 +00:00"
8 }
```

## Get-запрос по email:



GET http://localhost:3000/users/email/samarov2012@gmail.com Send

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

Body Cookies Headers (7) Test Results 200 OK 9 ms 403 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "username": "ilya3716",
4   "email": "samarov2012@gmail.com",
5   "role": "carry",
6   "createdAt": "2024-03-30 01:27:55.732 +00:00",
7   "updatedAt": "2024-03-30 01:27:55.732 +00:00"
8 }
```

## **Выводы.**

В результате работы, я научился работать с express и sequelize и писать пользовательские обработчики запросов.