## Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования

## «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

## Отчет

по домашней работе № 2 «Знакомство с ORM Sequelize» по дисциплине «Бэкенд-разработка»

Автор: Власов М. И.

Факультет: ИКТ

Группа: К33402

Преподаватель: Добряков Д. И.

Дата: 18.03.22



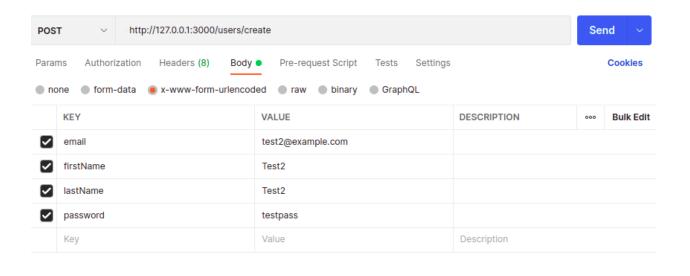
1. Продумать свою собственную модель пользователя

```
matvey@matvey-Aspire-A315-55KG:-/ITMO-ICT-Backend-2022/homeworks/K33402/Vlasov_Matvey/HW2$ npx sequelize-cli model:generate --name User --attributes e
mail:string,firstName:string,lastName:string,password:string
Sequelize CLI [Node: 16.13.0, CLI: 6.4.1, ORM: 6.17.0]
New model was created at /home/matvey/ITMO-ICT-Backend-2022/homeworks/K33402/Vlasov_Matvey/HW2/models/user.js
New migration was created at /home/matvey/ITMO-ICT-Backend-2022/homeworks/K33402/Vlasov_Matvey/HW2/migrations/20220318070515-create-user.js
```

```
'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({
    email: DataTypes.STRING,
    firstName: DataTypes.STRING,
    lastName: DataTypes.STRING,
    password: DataTypes.STRING
  }, {
    sequelize,
    modelName: 'User',
  });
```

2. Реализовать набор из CRUD-методов для работы с пользователями средствами Express + Sequelize

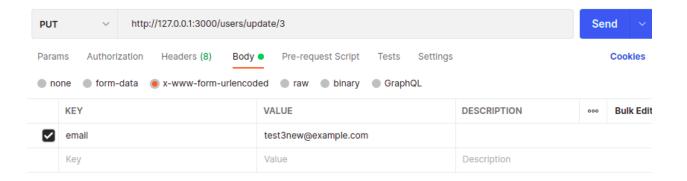
```
app.post('/users/create', async (req, res) => {
    if(!req.body.email || !req.body.firstName || !req.body.lastName || !req.body.password) return res.sendStatus[400]
    const user = await db.User.create(req.body)
    res.send(user.toJSON())
})
```



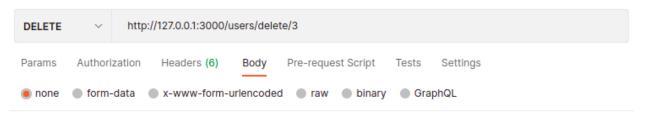
```
Body Cookies Headers (7) Test Results
                                                                            200 OK 61 ms 389 B Save Response >
                             Visualize
                                                                                                         □ Q
 Pretty
                  Preview
                                                                                                                  I
   2
           "id": 2,
   3
           "email": "test2@example.com",
   4
           "firstName": "Test2",
           "lastName": "Test2",
   5
   6
           "updatedAt": "2022-03-18T08:08:05.908Z",
```

```
app.put('/users/update/:id', async (req, res) => {
   const user = await db.User.findByPk(req.params.id)

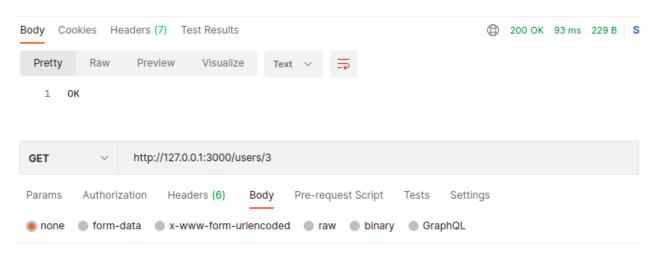
if (user) {
   await user.update(req.body)
   res.send(user.toJSON())
} else {
   res.send({"error": "user is not found"})
}
```



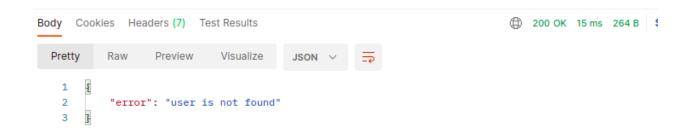
```
Body Cookies Headers (7) Test Results
                                                      200 OK 66 ms 392 B Save Response V
 Pretty
       Raw
            Preview
                   Visualize
                                                                          □ Q
                                                                                 П
        "id": 3,
  2
       "firstName": "test3",
  4
        "lastName": "test3",
  5
        "email": "test3new@example.com",
        "createdAt": "2022-03-18T08:09:08.161Z",
 app.delete('/users/delete/:id', async (req, res) => {
      const user = await db.User.findByPk(req.params.id)
      if (user) {
           await user.destroy()
           res.sendStatus(200)
        else {
           res.send({"error": "user is not found"})
```



This request does not have a body



This request does not have a body



## 3. Написать запрос для получения пользователя по id/email

```
app.get('/users/:id', async (req, res) => {
   const user = await db.User.findByPk(req.params.id)

if (user) {
   res.send(user.toJSON())
} else {
   res.send({"error": "user is not found"})
}

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.send({"error": "user is not found"})
}
})
```

Вывод: в ходе работы мы познакомились с ORM Sequelize и создали модель пользователя, а также набор CRUD-методов для работы с ним.