САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

По домашней работе №2
Знакомство с ORM Sequelize

Выполнила:

Киреева Марина

Группа: К3333

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

1. Продумать свою собственную модель пользователя

```
marinakireeva@MacBook-Pro-Marina hw2 % npx sequelize init

Sequelize CLI [Node: 18.14.2, CLI: 6.6.0, ORM: 6.29.3]

Created "config/config.json"
Successfully created models folder at "/Users/marinakireeva/Desktop/шарага/ИТМО/6эк/KireevaMarina/hw2/hw2/models".
Successfully created migrations folder at "/Users/marinakireeva/Desktop/шарага/ИТМО/6эк/KireevaMarina/hw2/hw2/migrations".
Successfully created seeders folder at "/Users/marinakireeva/Desktop/шарага/ИТМО/6эк/KireevaMarina/hw2/hw2/seeders".
```

```
marinakireeva@MacBook-Pro-Marina hw2 % npx sequelize-cli model:generate --name User --attributes firstName:string,lastName:string,email:string,passport:string,passwd:string

Sequelize CLI [Node: 18.14.2, CLI: 6.6.0, ORM: 6.29.3]

New model was created at /Users/marinakireeva/Desktop/шарага/ИТМО/6эк/KireevaMarina/hw2/hw2/models/user.js .
New migration was created at /Users/marinakireeva/Desktop/шарага/ИТМО/6эк/KireevaMarina/hw2/hw2/migrations/20230314114421-create-user.js .
```

```
marinakireeva@MacBook-Pro-Marina hw2 % npx sequelize-cli db:migrate

Sequelize CLI [Node: 18.14.2, CLI: 6.6.0, ORM: 6.29.3]

Loaded configuration file "config/config.json".
Using environment "development".
== 20230314114421-create-user: migrating =======
== 20230314114421-create-user: migrated (0.014s)
```

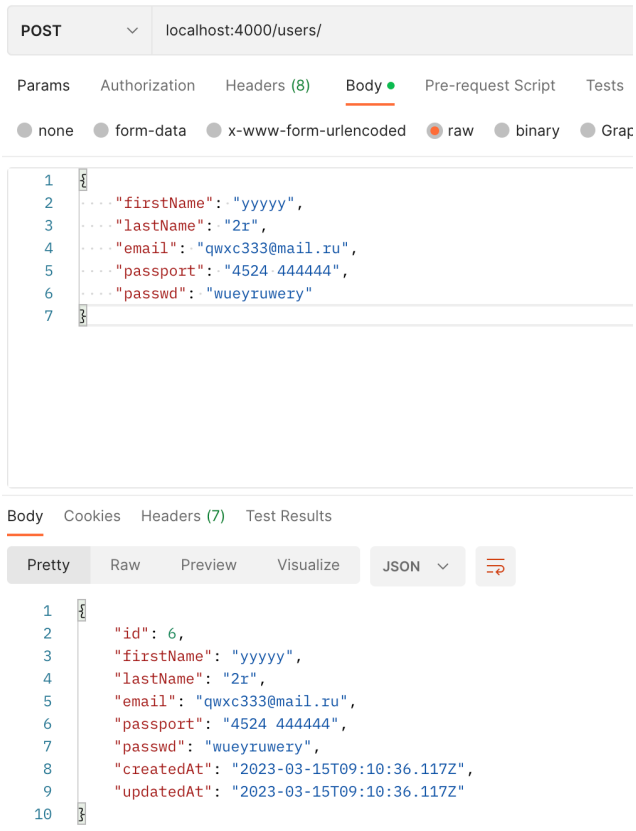
Модель пользователя:

```
'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( attributes: {
    firstName: DataTypes.STRING,
    lastName: DataTypes.STRING,
    email: DataTypes.STRING,
    passport: DataTypes.STRING,
    passwd: DataTypes.STRING
  },  options: {
    sequelize,
    modelName: 'User',
  });
  return User;
};
```

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

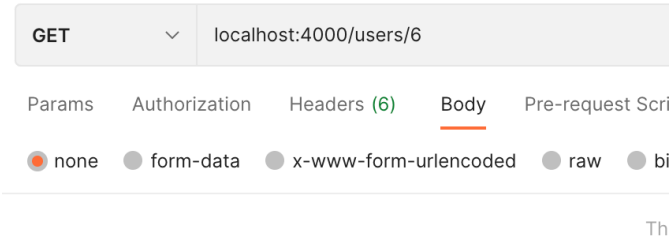
Create (POST)

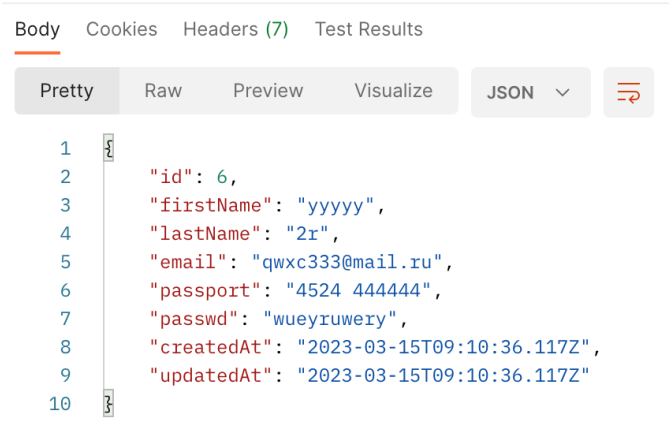
```
app.post ('/users/', async (req, res) => {
    try {
        const user = await db.User.create(req.body)
        await user.reload()
        res.send(user.toJSON())
    }
    catch (error) {
        res.send({error:error.msg})
    }
})
```



Read (GET) по id:

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

    console.log('user is', user)

    if (user) {
        return res.send(user.toJSON())
    }

    return res.send({"msg": "user is not found"})
})
```



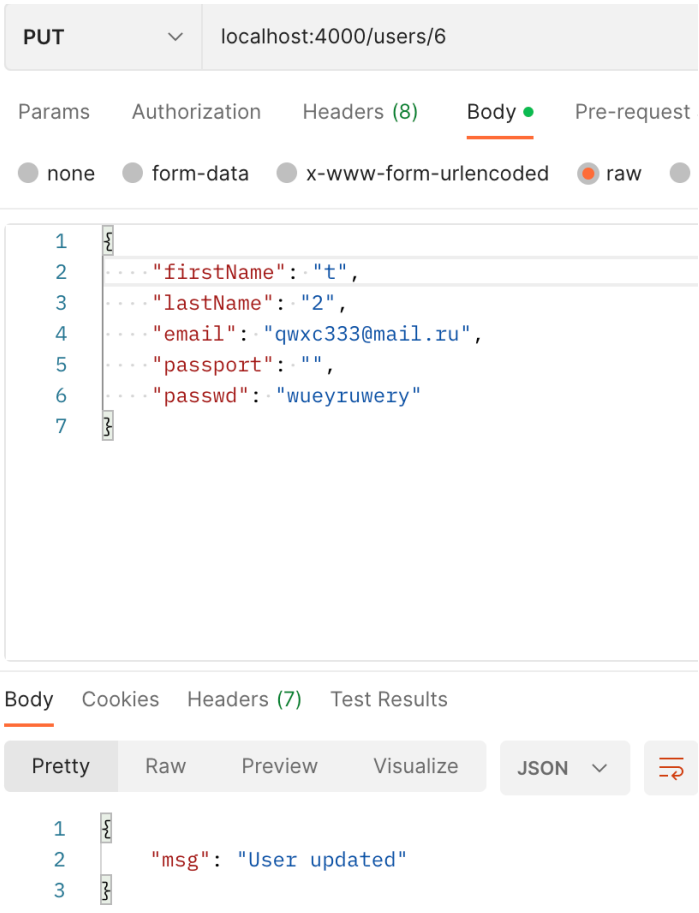


Read (GET) по email:

```
app.get('/users/:email', async (req, res) => {
    const user = await db.User.findAll({where:{email:req.params.email}})

    console.log('user is', user)

    if (user) {
        return res.send(user.toJSON())
    }

    return res.send({"msg": "user is not found"})
})
```

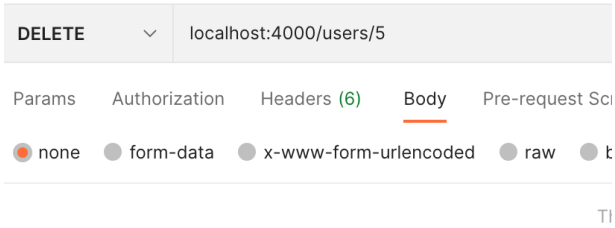
Update (PUT)

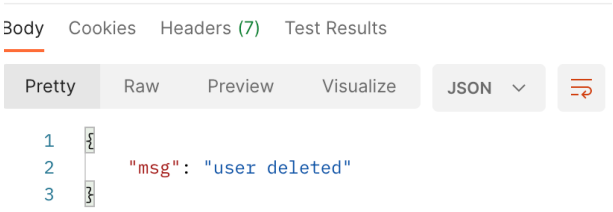
```
app.put ('/users/:id',async (req,res) =>{
    try {
        const user = await db.User.update(req.body, {where:{id:req.params.id}})
        if (user) {
            return res.send({"msg":"User updated"})
        }
            else {
                return res.send({"msg": "user is not found"})
            }
        }
    catch (error) {
        res.send({error:error.msg})
    }
})
```



DELETE

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





Вывод

В ходе выполнения домашнего задания была продумана и создана собственная модель пользователя средствами ORM Sequelize, и реализован набор CRUD-методов для работы с моделью при помощи Express.