САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №2: Знакомство с ORM Sequelize

Выполнила:

Балдина Дарья

Группа К33401

Проверил:
Добряков Д. И.

Санкт-Петербург

2023 г.

Задача

- 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. **Модель пользователя состоит из 5 полей:**
    - имя
    - фамилия
    - ник(username)
    - email
    - пароль

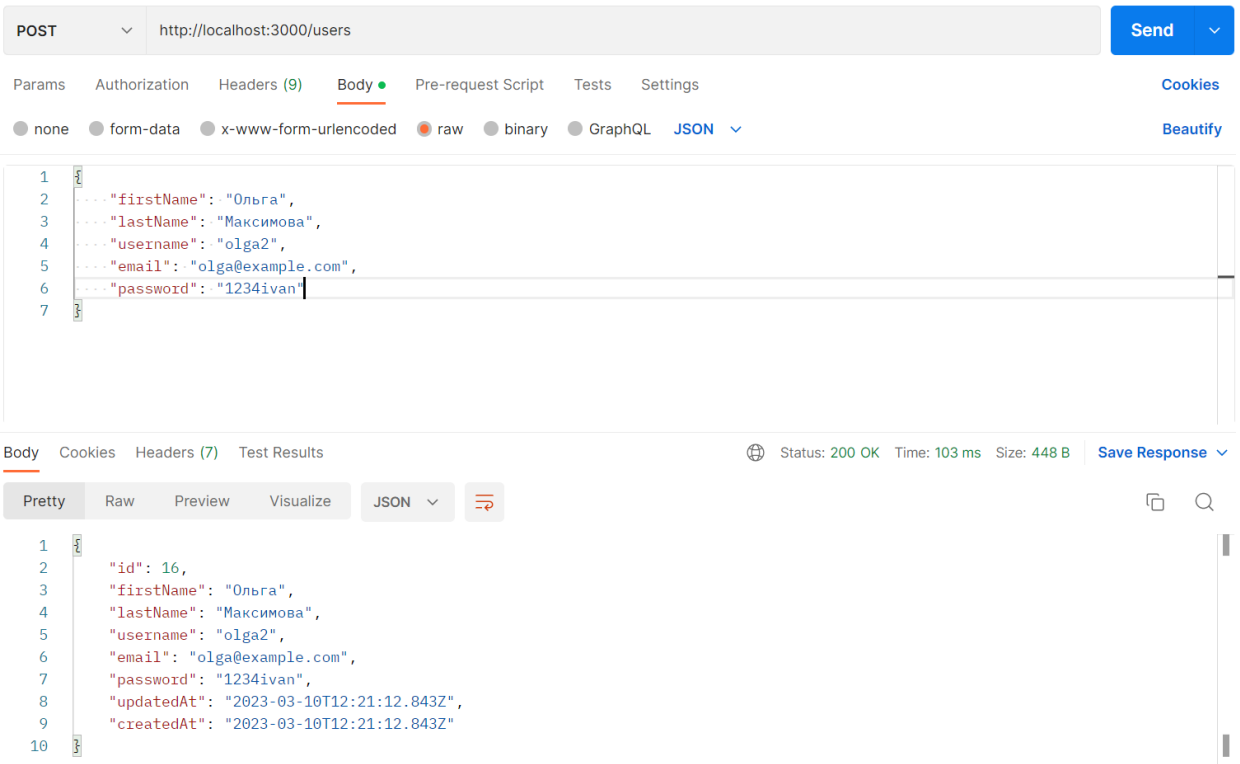
```
'use strict';
const {
  Model
} = require('sequelize');
module.exports = (sequelize, DataTypes) => {
  class User extends Model {
    /**
     * Helper method for defining associations.
     * This method is not a part of Sequelize lifecycle.
     * The `models/index` file will call this method automatically.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

Представления:

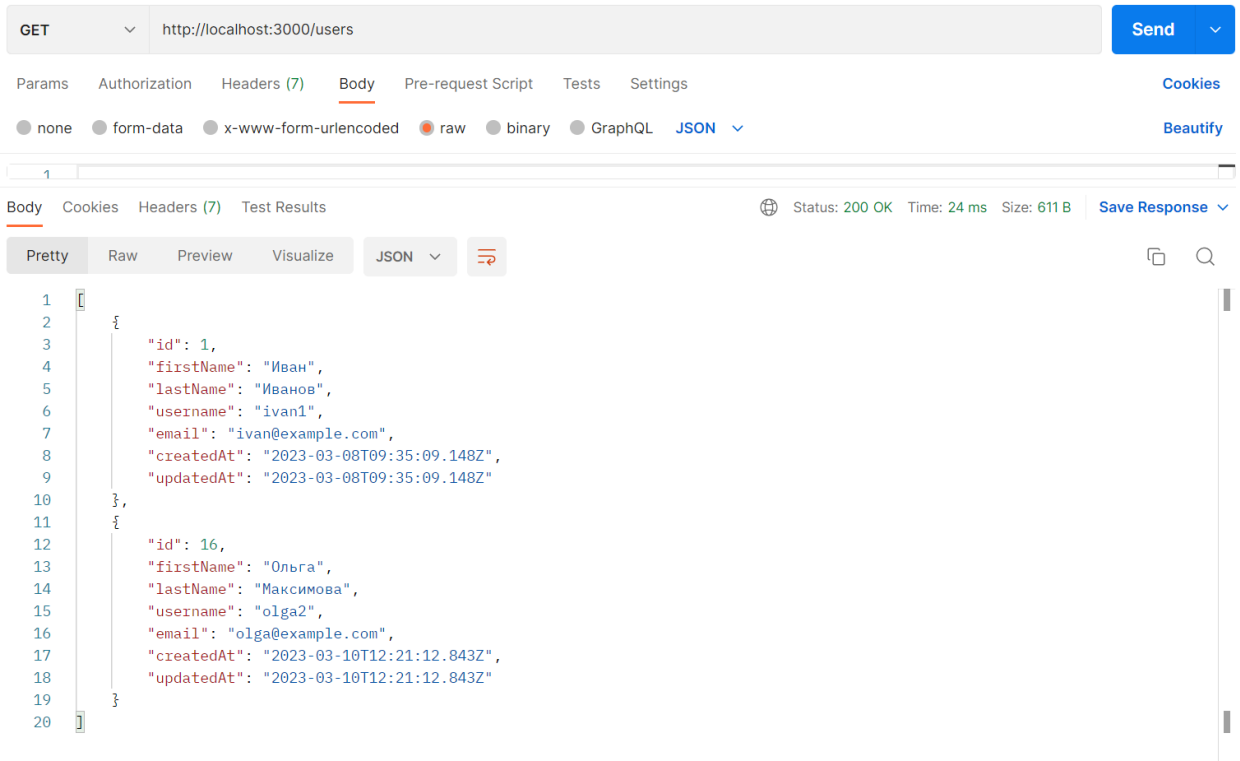
```
const express = require('express')
const db = require('./models')

const app = express()
const port = 3000


app.use(express.json());

app.get('/', (req, res) => {
    res.send('Homework 2')
})

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

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

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

})

app.post('/users', async (req, res) => {
  try {
      const user = await db.User.create(req.body);
      res.send(user.toJSON());
  } catch (e) {
      res.status(400).send({"detail": "Failed to create user"});
  }
})
app.put('/users/:id', async (req, res) => {
  try {
    const user = await db.User.update(req.body, {where: {id: req.params.id}});
    res.send({'status_code': 'User updated'});
  } catch (err) {
    res.send(err);
  }
})

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

app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

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

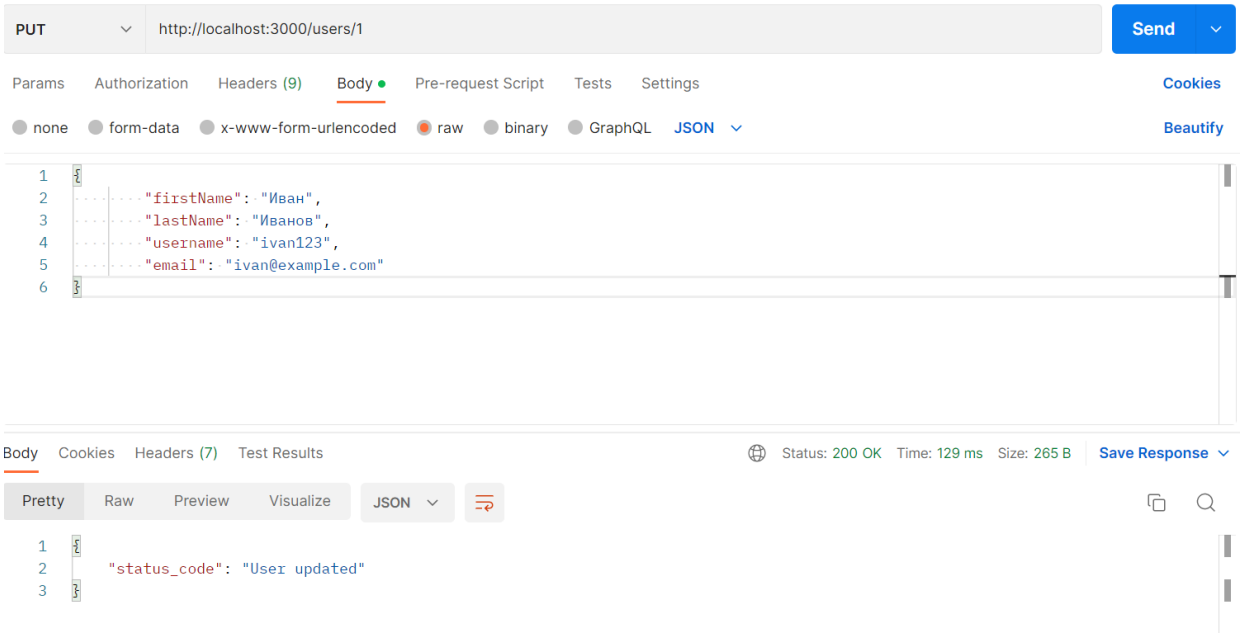
CREATE (POST):



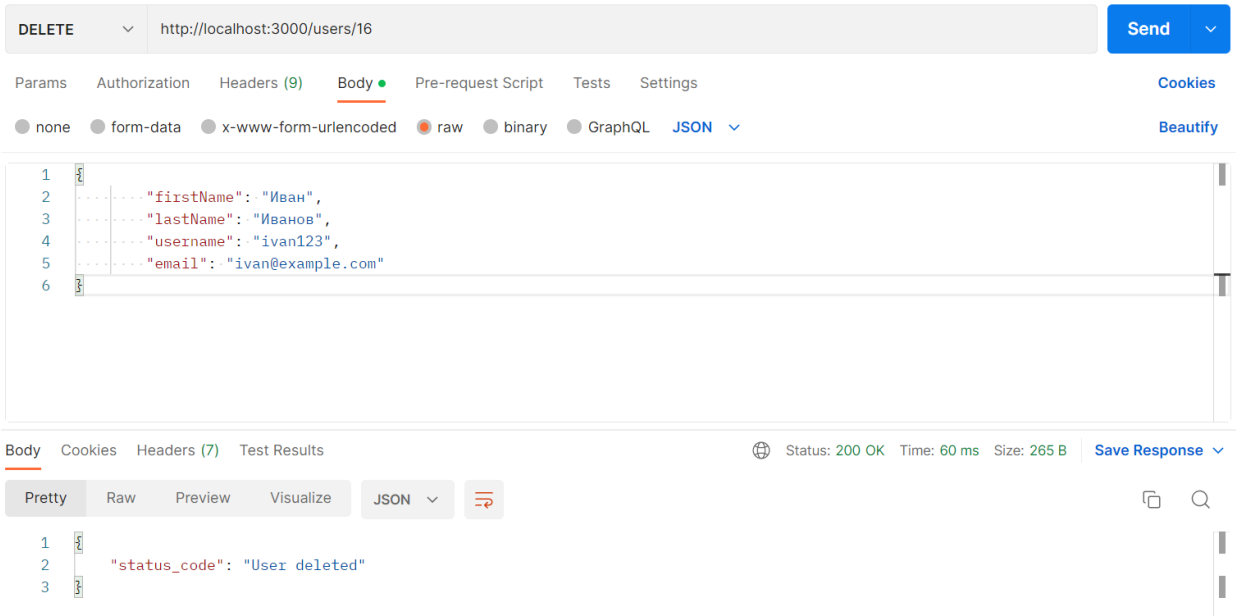
READ (GET):



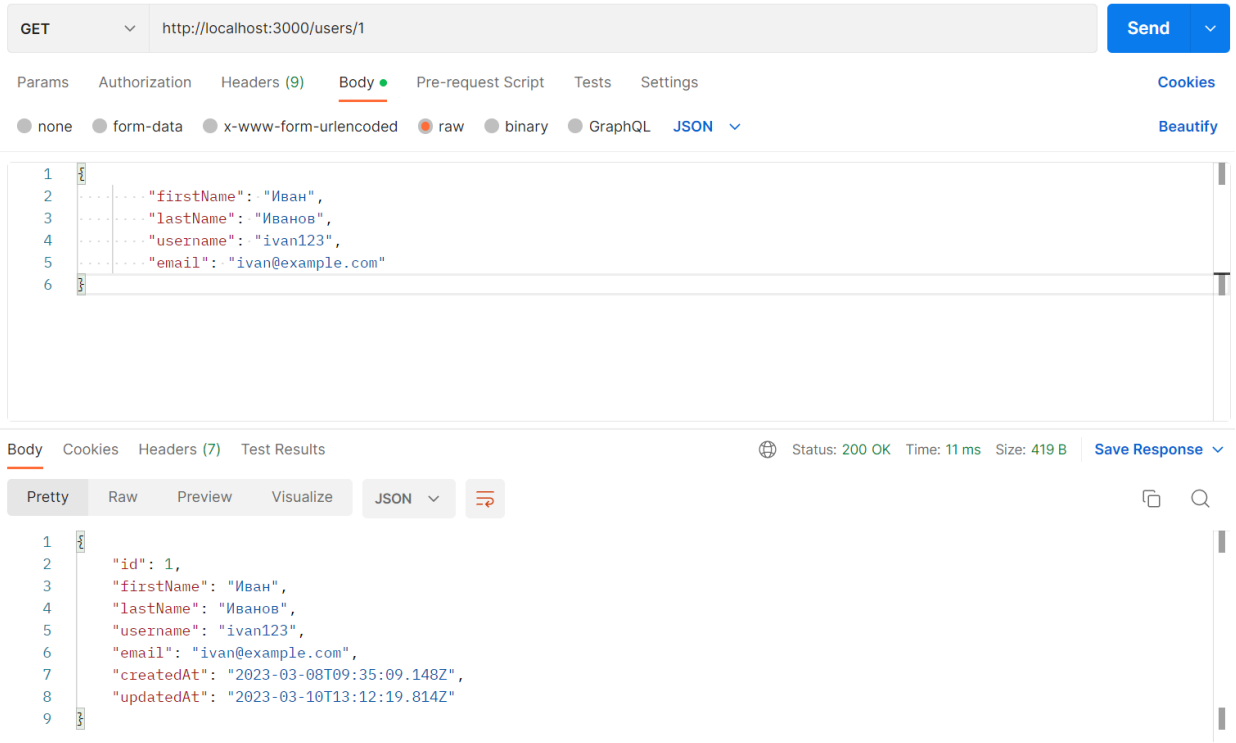
UPDATE (PUT):



DELETE:



3. Запрос для получения пользователя по id



Вывод

В ходе выполнения домашнего задания была придумана и создана собственная модель пользователя средствами ORM Sequelize, а также реализован набор CRUD-методов для работы с моделью юзера с помощью Express.