САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Практическая работа

Выполнил:

Буданцев Артём

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. Продумать свою собственную модель пользователя

```
artem@Debian:~/Projects/ITMO/ITMO-ICT-Backend-2023/homeworks/K3333/HW2$ npx sequelize init

Sequelize CLI [Node: 16.19.1, CLI: 6.6.0, ORM: 6.29.3]

Created "config/config.json"
Successfully created models folder at "/home/artem/Projects/ITMO/ITMO-ICT-Backend-2023/homeworks/K3333/HW2/models".
Successfully created migrations folder at "/home/artem/Projects/ITMO/ITMO-ICT-Backend-2023/homeworks/K3333/HW2/migrations".
Successfully created seeders folder at "/home/artem/Projects/ITMO/ITMO-ICT-Backend-2023/homeworks/K3333/HW2/seeders".
```

```
artem@Debian:~/Projects/ITMO/ITMO-ICT-Backend-2023/homeworks/K3333/HW2$ npx sequelize-cli model:generate --name User --attributes firstName:str
ng,lastName:string,email:string,password:string,phone:string

Sequelize CLI [Node: 16.19.1, CLI: 6.6.0, ORM: 6.29.3]

New model was created at /home/artem/Projects/ITMO/ITMO-ICT-Backend-2023/homeworks/K3333/HW2/models/user.js .
New migration was created at /home/artem/Projects/ITMO/ITMO-ICT-Backend-2023/homeworks/K3333/HW2/migrations/20230311133225-create-user.js .
```

```
artem@Debian:~/Projects/ITMO/ITMO-ICT-Backend-2023/homeworks/K3333/HW2$ npx sequelize db:migrate

Sequelize CLI [Node: 16.19.1, CLI: 6.6.0, ORM: 6.29.3]

Loaded configuration file "config/config.json".
Using environment "development".
== 20230311133225-create-user: migrating =======
== 20230311133225-create-user: migrated (0.026s)
```

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

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

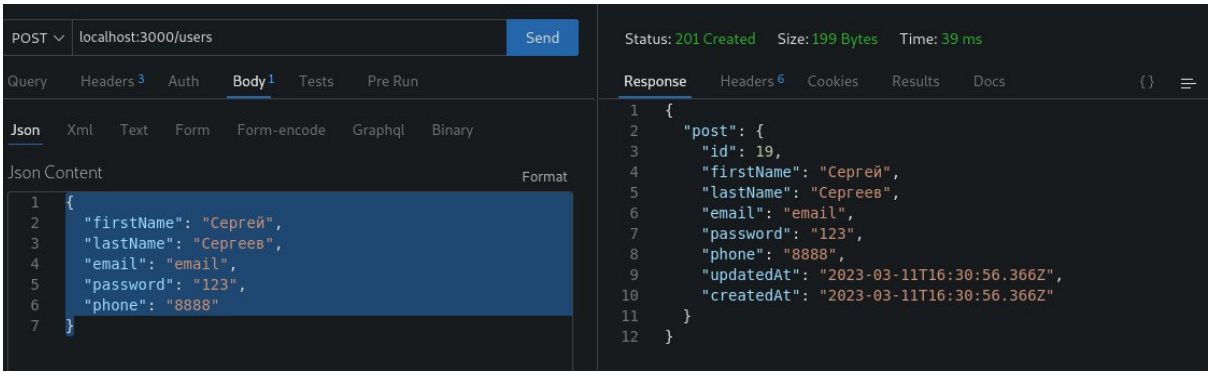
READ

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

  console.log('user is', users)

  if (users) {
    return res.send(users)
  }

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


```
[
  {
    "id": 1,
    "firstName": "Иван",
    "lastName": "Иванов",
    "email": "ivanov@mail.com",
    "password": "123",
    "phone": "123-123",
    "createdAt": "2023-03-11T14:14:57.050Z",
    "updatedAt": "2023-03-11T14:14:57.050Z"
  },
  {
    "id": 2,
    "firstName": "Иван",
    "lastName": "Иванов",
    "email": "ivanov@mail.com",
    "password": "123",
    "phone": "123-123",
    "createdAt": "2023-03-11T14:16:12.294Z",
    "updatedAt": "2023-03-11T14:16:12.294Z"
  },
  {
    "id": 3,
    "firstName": null,
    "lastName": null,
    "email": null,
    "password": null,
    "phone": null,
    "createdAt": "2023-03-11T15:00:23.011Z",
    "updatedAt": "2023-03-11T15:00:23.011Z"
  },
```

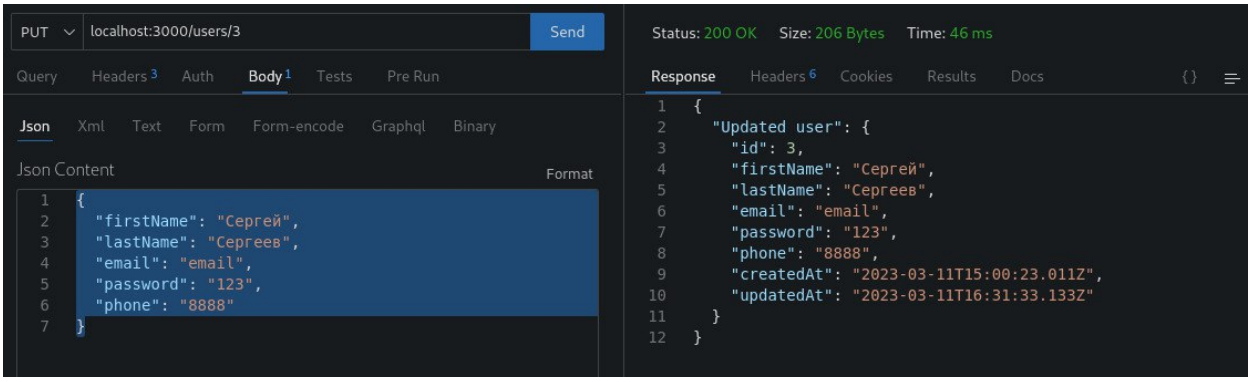
CREATE

```
app.post('/users', async (req, res) => {
  try {
    const post = await db.User.create(req.body)
    return res.status(201).json({post})
  } catch (error) {
    return res.status(500).json({error: error.message})
  }
})
```



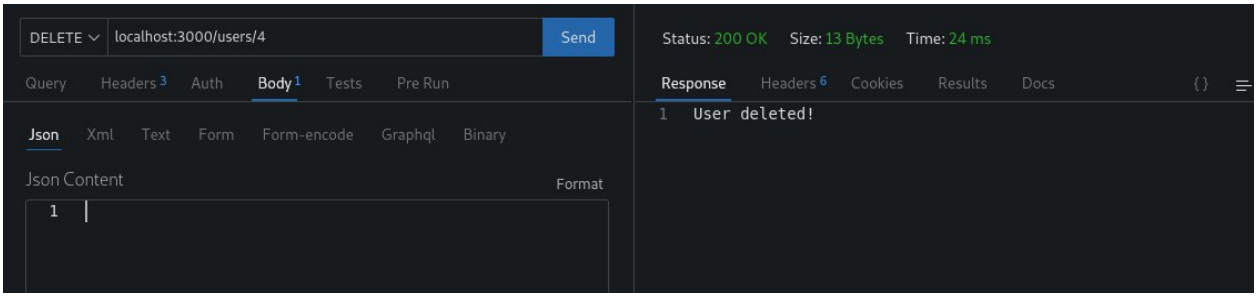
UPDATE

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

    if(check) {
      const updated = await db.User.update(req.body, {where: {id: req.params.id}})

      if(updated) {
        const user = await db.User.findByPk(req.params.id)

        res.send({'Updated user': user})
      }
    }
    else {
      res.send("User didn't find.")
    }

  } catch(error) {
    res.send({error: error.message})
  }
})
```



DELETE

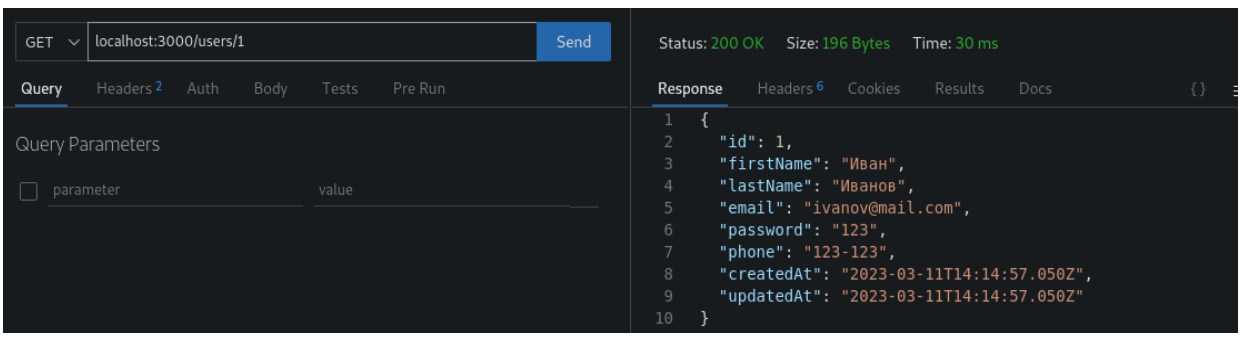
```
app.delete('/users/:id', async (req, res) => {
  try {
    const check = await db.User.findByPk(req.params.id)

    if(check) {
      const deleted = await db.User.destroy({where: {id: req.params.id}})

      if(deleted) {
        res.send("User deleted!")
      }
    }
    else {
      res.send("User didn't find.")
    }


  } catch(error) {
    res.send({error: error.message})
  }
})
```



3. Написать запрос для получения пользователя по id/email

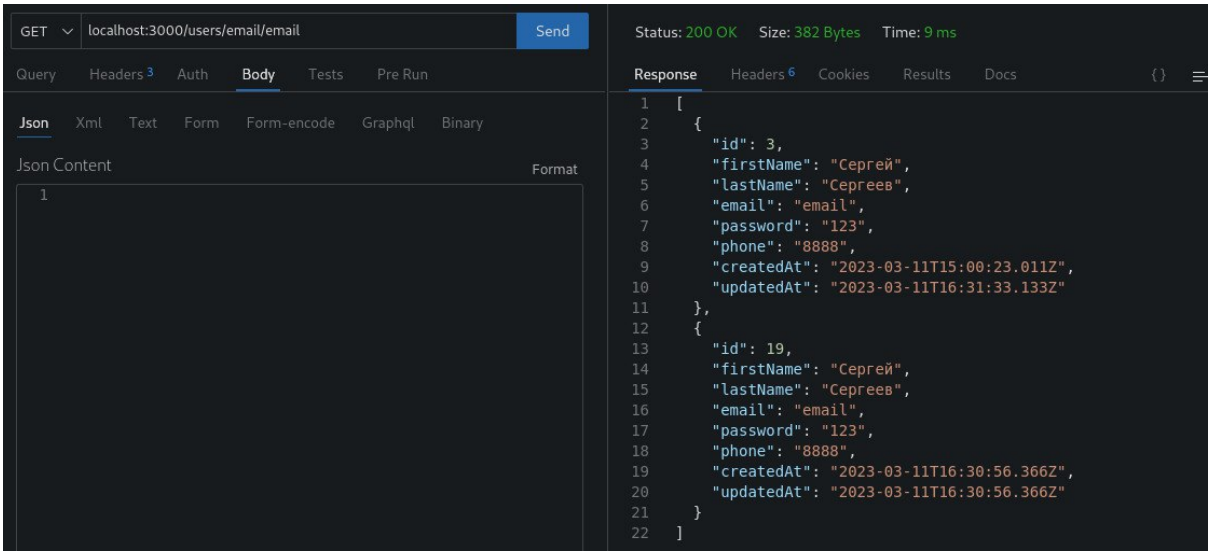
По id:

```
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"})
})
```



По email:

```
app.get('/users/email/: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)
  }

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



Вывод

В ходе работы были реализована модель пользователя, освоен инструмент миграций, созданы для неё CRUD-операции и освоена работа с ORM Sequelize.