САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа 2

Выполнил:

Горбатов Дмитрий

Группа
К33402

Проверил:
Добряков Д. И.

Санкт-Петербург
2024 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. Инициализируем модуль:

```
PS C:\Users\Home\Desktop\backend\hw2> npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (hw2)
version: (1.0.0)
description:
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to C:\Users\Home\Desktop\backend\hw2\package.json:

{
  "name": "hw2",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}


Is this OK? (yes)
```

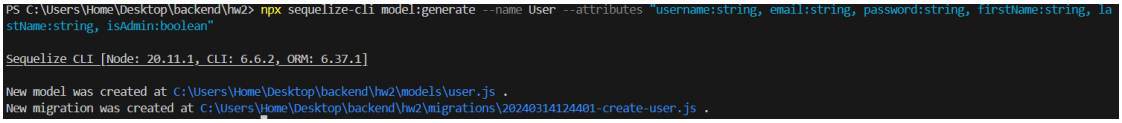
2. Установим зависимости:

```
PS C:\Users\Home\Desktop\backend\hw2> npm i express sequelize postgres sequelize-cli

added 1 package, and audited 344 packages in 2s


added 1 package, and audited 344 packages in 2s

41 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

3. Инициализируем sequelize:

```
PS C:\Users\Home\Desktop\backend\hw2> npx sequelize-cli init

Sequelize CLI [Node: 20.11.1, CLI: 6.6.2, ORM: 6.37.1]

Created "config\config.json"
Successfully created models folder at "C:\Users\Home\Desktop\backend\hw2\models".
Successfully created migrations folder at "C:\Users\Home\Desktop\backend\hw2\migrations".
Successfully created seeders folder at "C:\Users\Home\Desktop\backend\hw2\seeders".
```

4. Сгенерируем модель при помощи sequelize-cli:



5. Проведем миграцию:

```
PS C:\Users\Home\Desktop\backend\hw2> npx sequelize-cli db:migrate

Sequelize CLI [Node: 20.11.1, CLI: 6.6.2, ORM: 6.37.1]

Loaded configuration file "config\config.json".
Using environment "development".
== 20240314124401-create-user: migrating =======
== 20240314124401-create-user: migrated (0.022s)
```

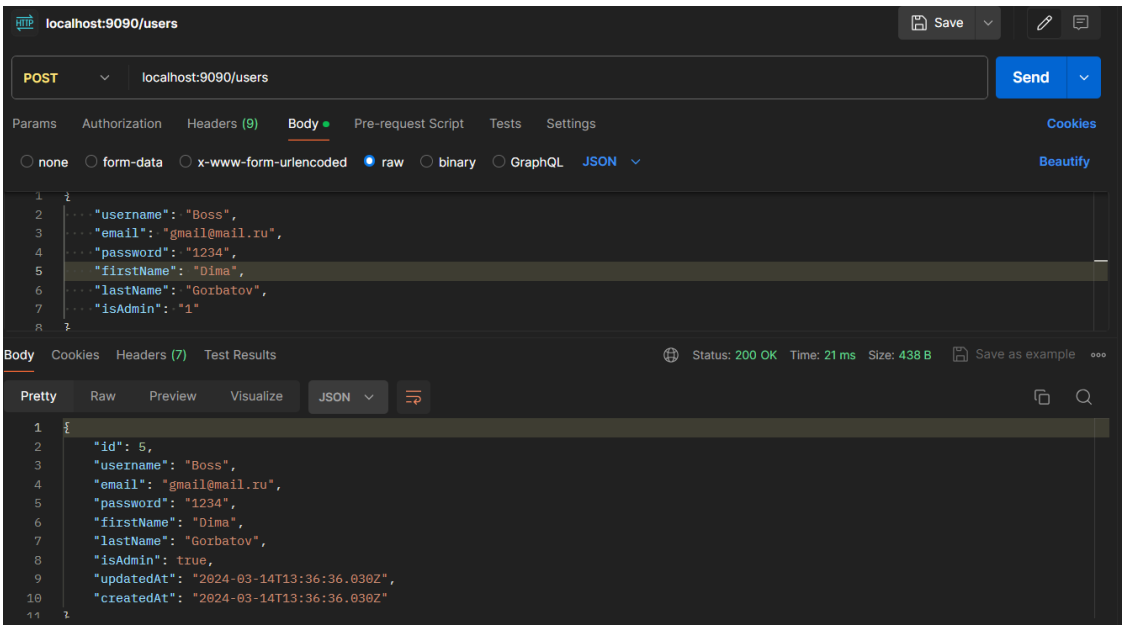
6. Создаем эндпоинты:

hw2 > JS index.js > ...

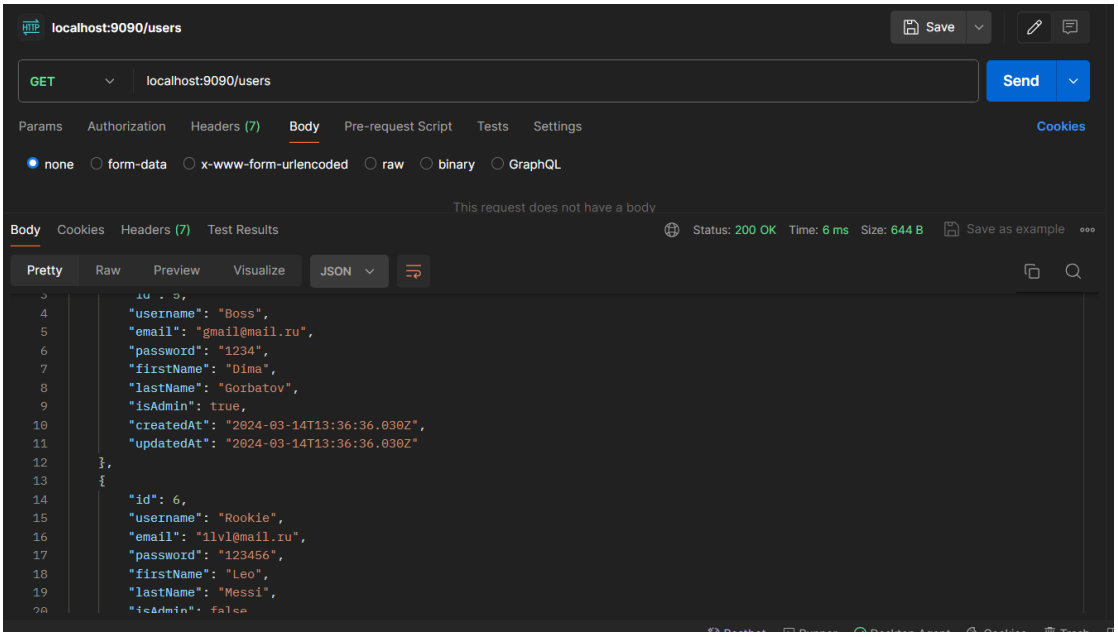
```js
const express = require('express')
const db = require('./models')
const app = express()

app.use(express.json())

// Create a new user
app.post('/users', async (req, res) => {
  try {
    const user = await db.User.create(req.body)
    res.status(200).json(user)
  } catch (error) {
    res.status(400).json({ error: error.message })
  }
})

// Get all users
app.get('/users', async (req, res) => {
  try {
    res.json(await db.User.findAll())
  } catch (error) {
    res.status(500).json({ error: error.message })
  }
})

// Get user by ID
app.get('/users/:id', async (req, res) => {
  try {
    const user = await db.User.findByPk(req.params.id)
    if (!user) {
      res.status(404).json({ error: 'User not found' })
    } else {
      res.json(user)
    }
  } catch (error) {
    res.status(500).json({ error: error.message })
  }
```

```
38  })
39
40  // Update user by ID
41  app.patch('/users/:id', async (req, res) => {
42    try {
43      const user = await db.User.findByPk(req.params.id)
44      if (!user) {
45        res.status(404).json({ error: 'User not found' })
46      } else {
47        await user.update(req.body)
48        res.json(user)
49      }
50    } catch (error) {
51      res.status(500).json({ error: error.message })
52    }
53  })
54
55  // Delete user by ID
56  app.delete('/users/:id', async (req, res) => {
57    try {
58      const user = await db.User.findByPk(req.params.id)
59      if (!user) {
60        res.status(404).json({ error: 'User not found' })
61      } else {
62        await user.destroy()
63        res.status(200).send()
64      }
65    } catch (error) {
66      res.status(500).json({ error: error.message })
67    }
68  })
69
70  app.listen(9090, () => {
71    console.log('listening on port 9090')
72  })
```

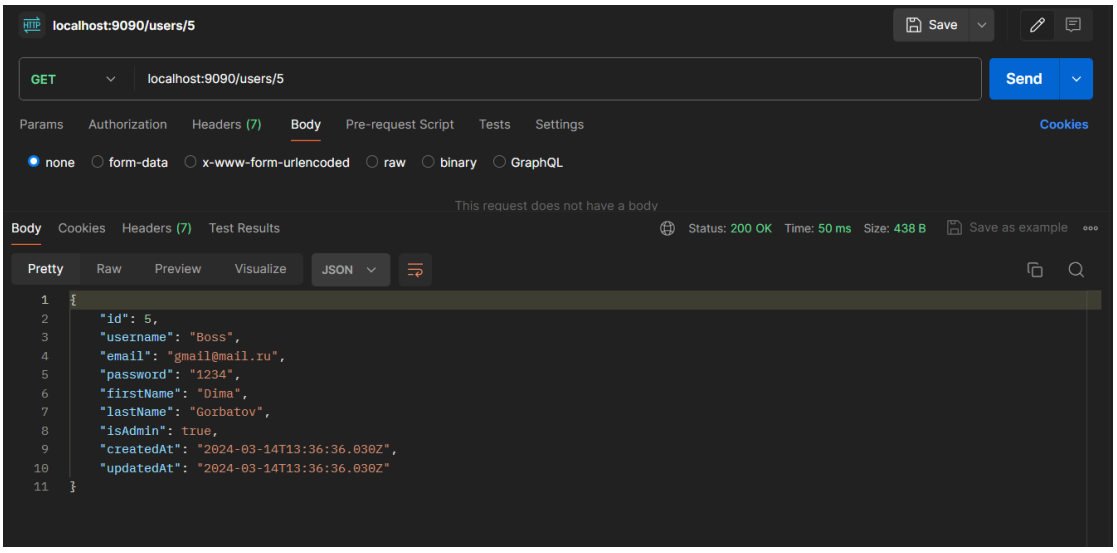
7. POST запрос:



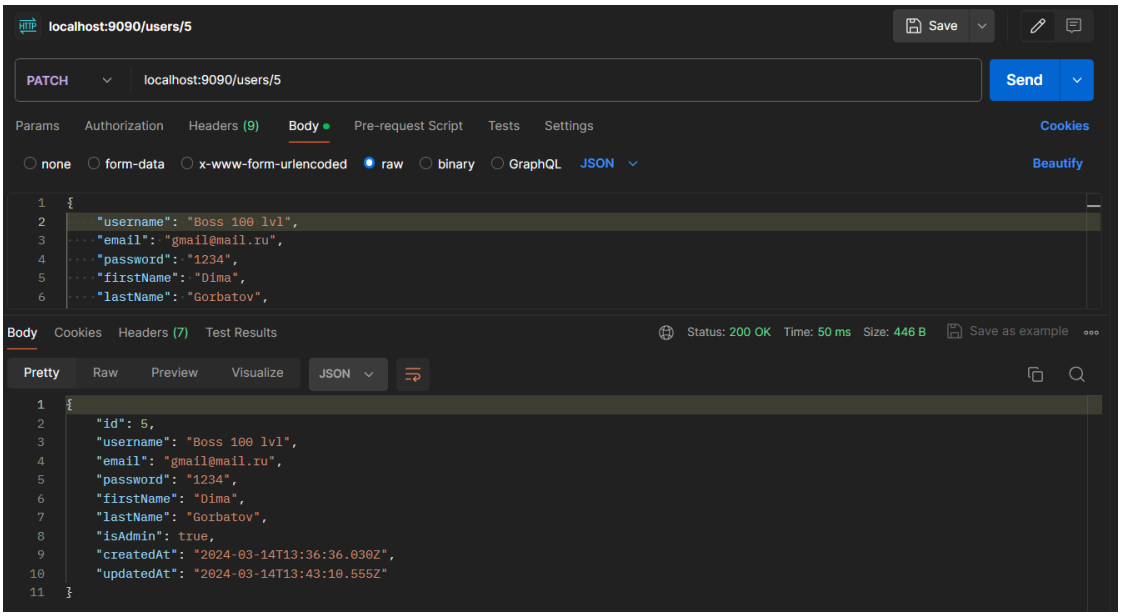
8. GET запрос:



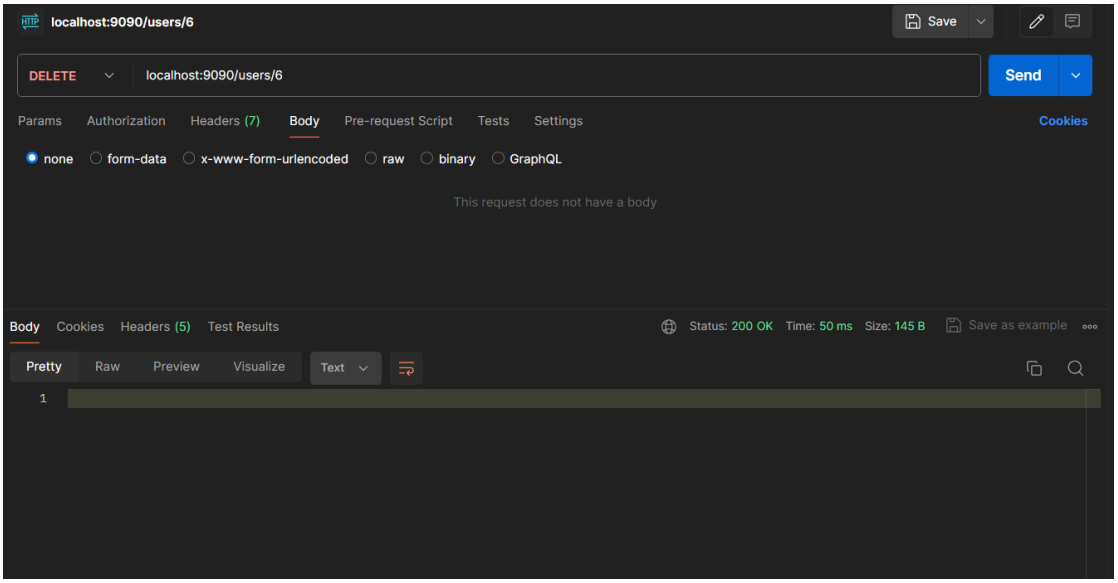
9. GET запрос определенного юзера:

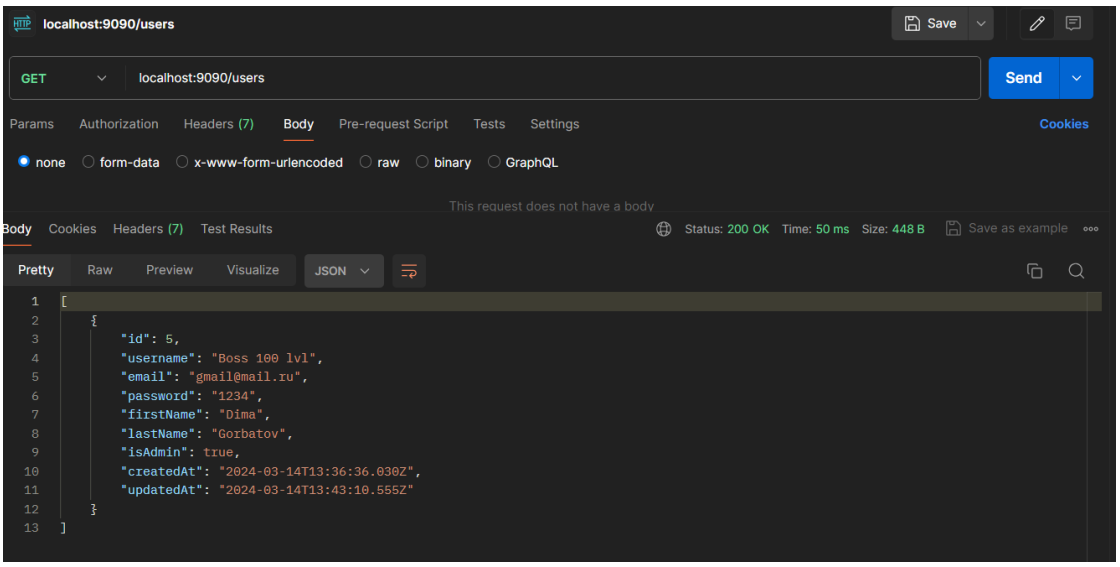


10. PATCH запрос:



11. DELETE запрос:





Вывод:

В данной домашней работе удалось написать HTTP сервер, который обрабатывает запросы для CRUD-операций над пользователем при помощи библиотеки express и sequelize для работы с базой данных.