САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа №2

Выполнила:

Кормановская Дарина

Группа K33401

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

Установим все необходимое

```
npm init
```

```
npm i express -S
```

```
npm i sequelize -S
```

```
npm install sqlite3 -S
```

```
npm install --save-dev sequelize-cli
```

```
npx sequelize init
```

Модель пользователя

username	String	Никнейм пользователя
email	String	Электронный адрес
password	String	Пароль
news	Boolean	Получение новостной рассылки

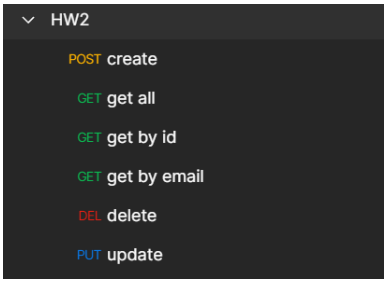
```
npx sequelize-cli model:generate --name User --attributes
username:string,email:string,password:string,news:boolean
```

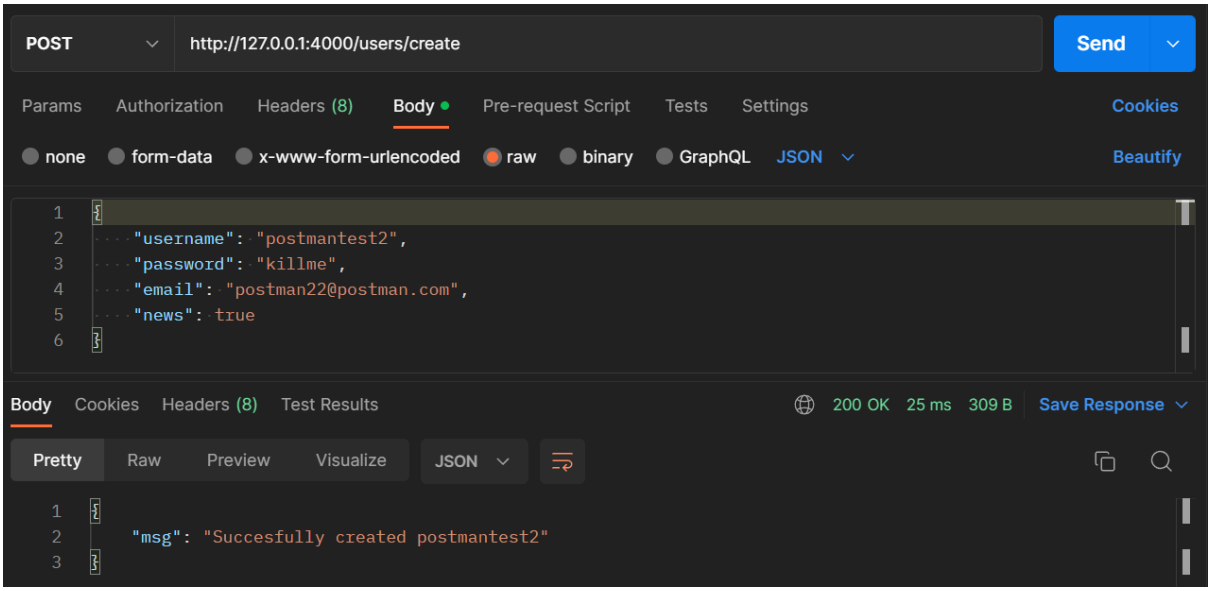
```
npx sequelize-cli db:migrate
```

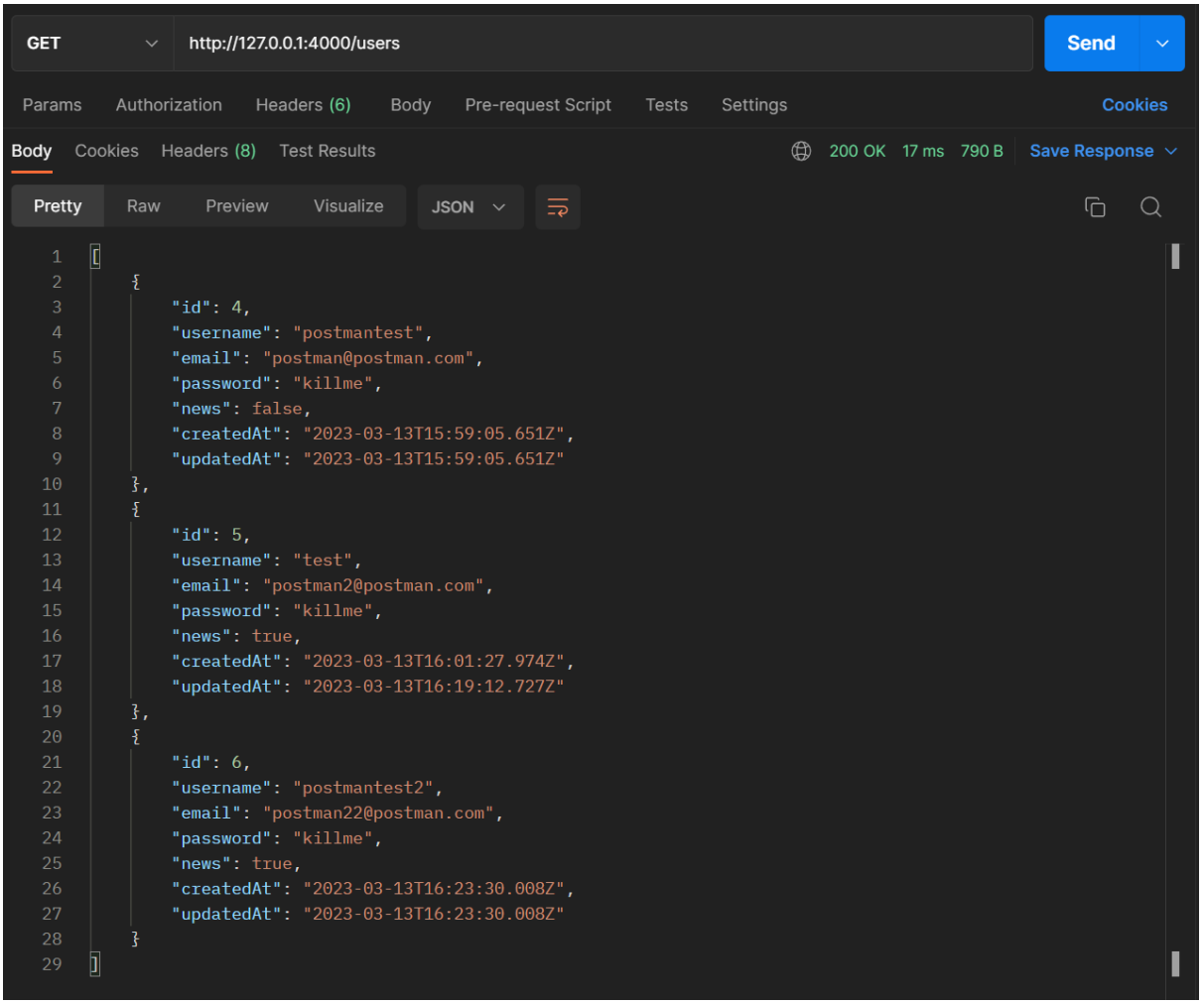
Получившиеся пути и методы

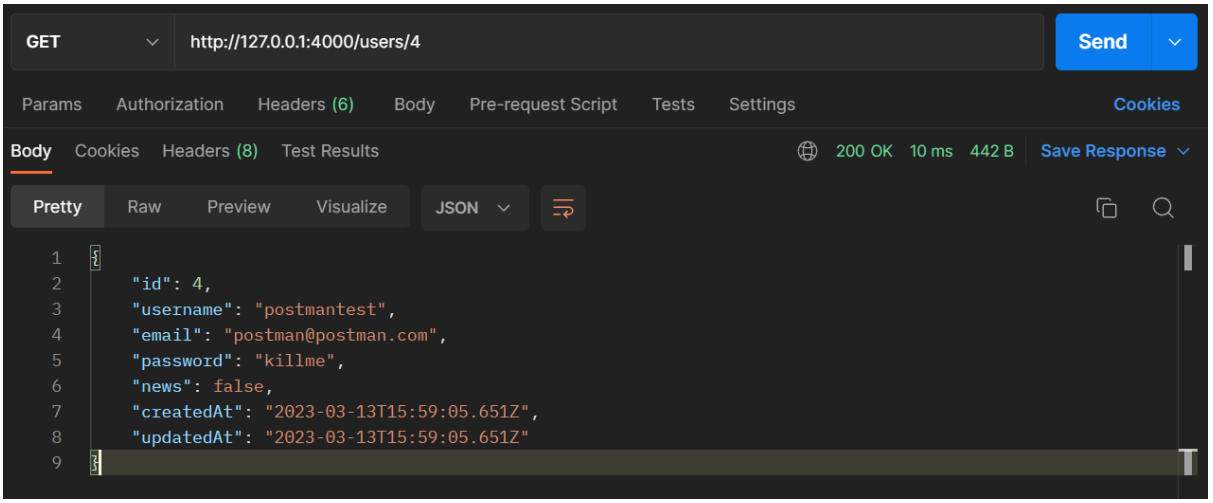
```
app.get('/users/:id', async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) => {
    const user = await db.User.findByPk(req.params.id)
    if (user) {
        return res.send(user.toJSON())
    }
    return res.send( body: {"msg": "User is not found"})
})

app.get('/users', async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) => {
    let users = []
    if (req.query && req.query.email) {
        users = await db.User.findAll( options: {where: {email: req.query.email}})
    } else {
        users = await db.User.findAll()
    }
    console.log(users)
    return res.send(users)
})

app.post( path: '/users/create', handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) => {
    try {
        const user = await db.User.create(req.body);
        return res.send( body: {"msg": "Succesfully created " + user.username})
    } catch (e) {
        return res.send( body: {"msg": e})
    }
})

app.delete( path: '/users/:id', handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) => {
    const user = await db.User.destroy({where: {id: req.params.id}})
    if (user) {
        return res.send( body: {"msg": "User deleted"})
    }
    return res.send( body: {"msg": "User is not found"})
})

app.put( path: '/users/:id', handlers: async (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) => {
    const user = await db.User.findByPk(req.params.id);
    if (user) {
        try {
            user.update(req.body, {where: {id: req.params.id}});
            return res.send( body: {"msg": "Successfully updated!"})
        } catch (e) {
            return res.send( body: {"msg": e})
        }
    }
    return res.send( body: {"msg": "User is not found"})
})
```

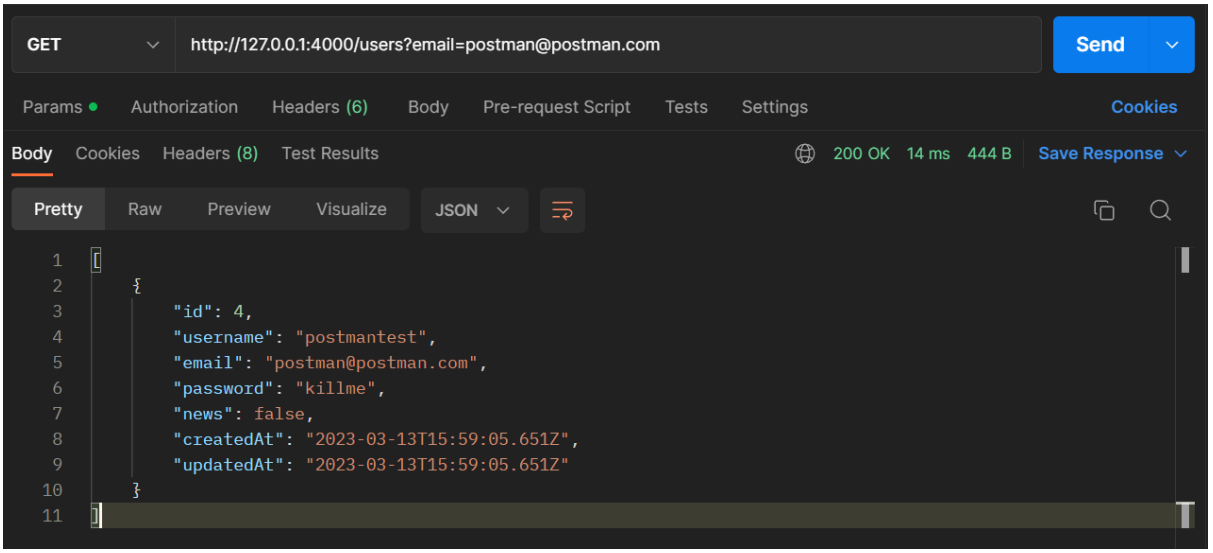
Проверка

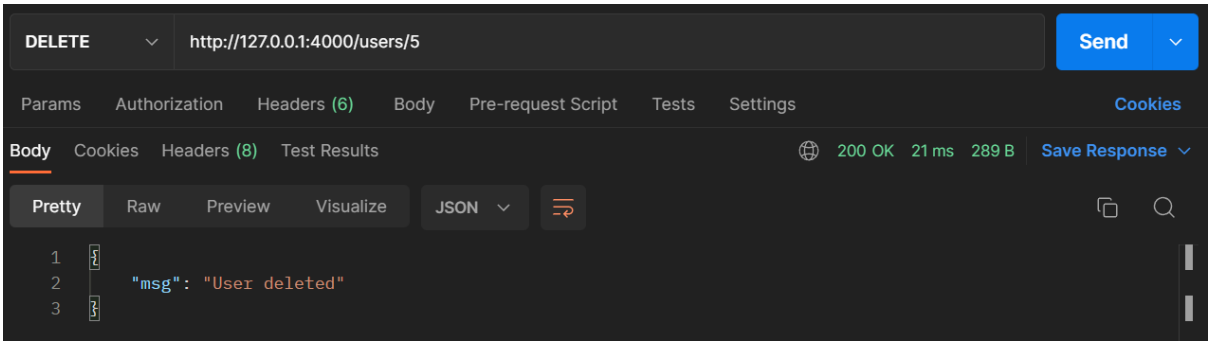


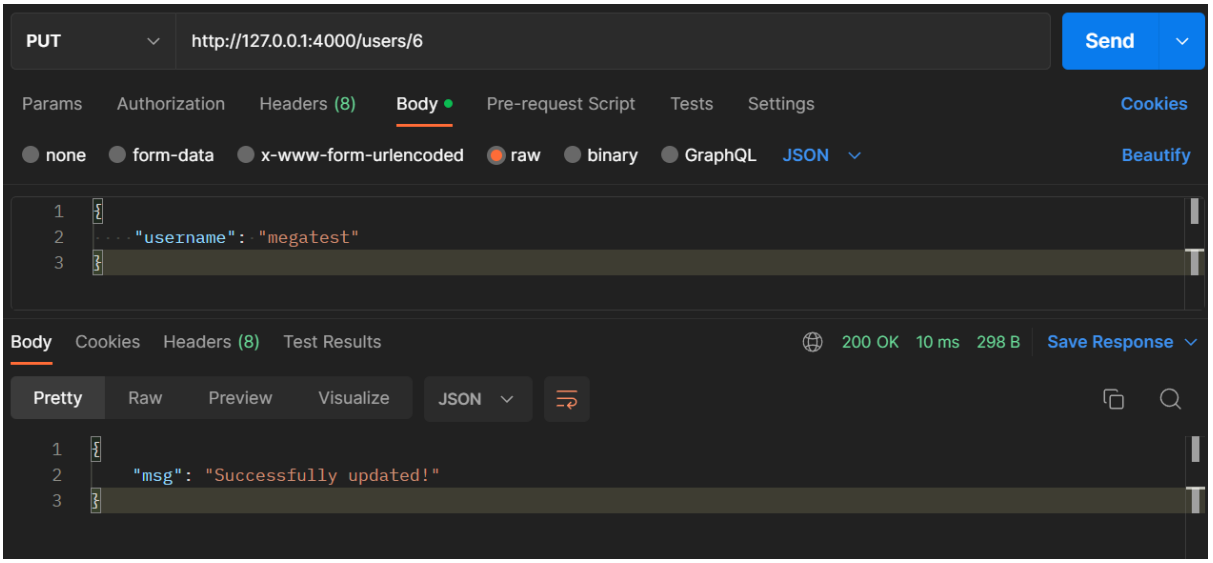

POST
http://127.0.0.1:4000/users/create
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
JSON
Beautify
{
"username": "postmantest2",
"password": "killme",
"email": "postman22@postman.com",
"news": true
}
Body
Cookies
Headers (8)
Test Results
200 OK
25 ms
309 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"msg": "Succesfully created postmantest2"
}

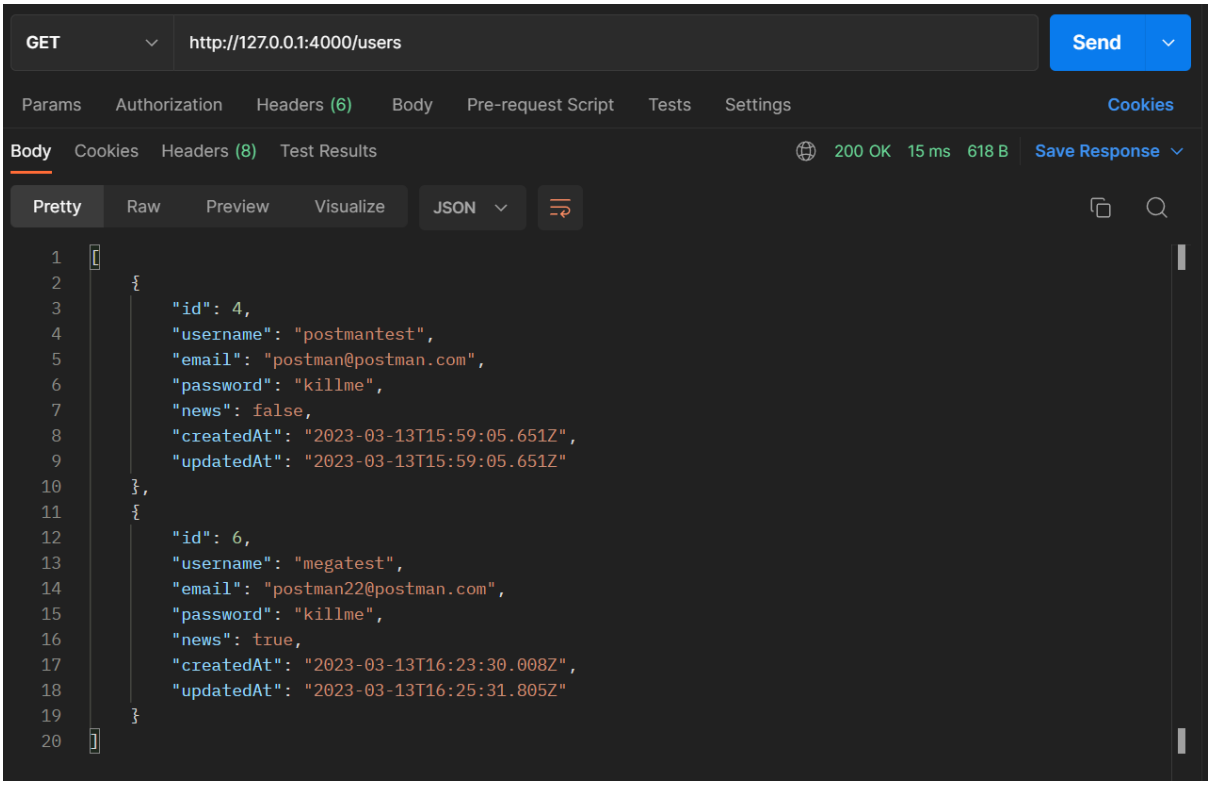

GET
http://127.0.0.1:4000/users
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Body
Cookies
Headers (8)
Test Results
200 OK
17 ms
790 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": 4,
"username": "postmantest",
"email": "postman@postman.com",
"password": "killme",
"news": false,
"createdAt": "2023-03-13T15:59:05.651Z",
"updatedAt": "2023-03-13T15:59:05.651Z"
},
{
"id": 5,
"username": "test",
"email": "postman2@postman.com",
"password": "killme",
"news": true,
"createdAt": "2023-03-13T16:01:27.974Z",
"updatedAt": "2023-03-13T16:19:12.727Z"
},
{
"id": 6,
"username": "postmantest2",
"email": "postman22@postman.com",
"password": "killme",
"news": true,
"createdAt": "2023-03-13T16:23:30.008Z",
"updatedAt": "2023-03-13T16:23:30.008Z"
}
]


GET
http://127.0.0.1:4000/users/4
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Body
Cookies
Headers (8)
Test Results
200 OK
10 ms
442 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": 4,
"username": "postmantest",
"email": "postman@postman.com",
"password": "killme",
"news": false,
"createdAt": "2023-03-13T15:59:05.651Z",
"updatedAt": "2023-03-13T15:59:05.651Z"
}


GET
http://127.0.0.1:4000/users?email=postman@postman.com
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Body
Cookies
Headers (8)
Test Results
200 OK
14 ms
444 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": 4,
"username": "postmantest",
"email": "postman@postman.com",
"password": "killme",
"news": false,
"createdAt": "2023-03-13T15:59:05.651Z",
"updatedAt": "2023-03-13T15:59:05.651Z"
}
]


DELETE
http://127.0.0.1:4000/users/5
Send
Params
Authorization
Headers (6)
Body
Pre-request Script
Tests
Settings
Cookies
Body
Cookies
Headers (8)
Test Results
200 OK
21 ms
289 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"msg": "User deleted"
}





Вывод

В ходе работы изучена работа Express и Sequelize.