

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Практическая работа №1

Выполнила:

Мухина Юлия

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

Создала проект и установила нужные пакеты, воспользовавшись командами

```
npm init
```

```
npm install express
```

```
npm install sequelize
```

```
npm i sqlite3 --S
```

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

```
npx sequelize init
```

В моей модели пользователя есть поля username, email, password, все они типа STRING. Код для этой модели генерирует команда:

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

Для проведения миграции после изменения структуры базы данных выполняем данную команду

```
npx sequelize db:migrate
```

Написанный API:

```

app.route("/users/").get(async (req, res) => {
  // Get all Users from db.
  // Supports query param `email`, for example /users/?email=example@example.com
  if (req.query && req.query.email) {
    const users = await db.User.findAll(
      {
        where: {email: req.query.email}
      }
    )
    return res.send(users)
  } else {
    const users = await db.User.findAll()
    return res.send(users)
  }
})

```

```

app.route("/users/").post(async (req, res) => {
  // Create new User
  try {
    const user = await db.User.create(req.body);
    return res.send("Success")
  } catch (e) {
    return res.send(e)
  }
})

```

```

app.route("/users/:id").get(async (req, res) => {
  // Get User by id
  const user = await db.User.findByPk(req.params.id)
  if (user) {
    return res.send(user.toJSON())
  } else {
    return res.send("User does not exist")
  }
})

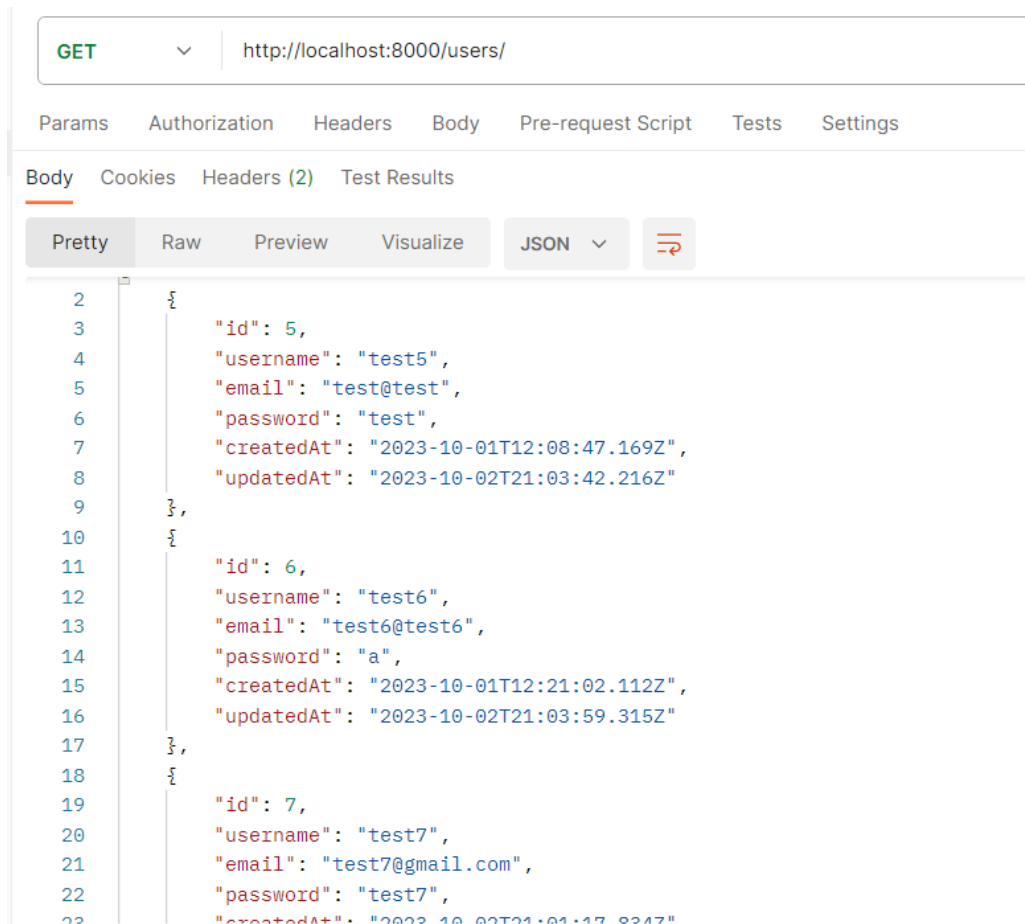
```

```
app.route("/users/:id").delete(async (req, res) => {
  // Delete User by id
  const user = await db.User.destroy({where: {id: req.params.id}})
  if (user) {
    return res.send("Success")
  } else {
    return res.send("User does not exist")
  }
})

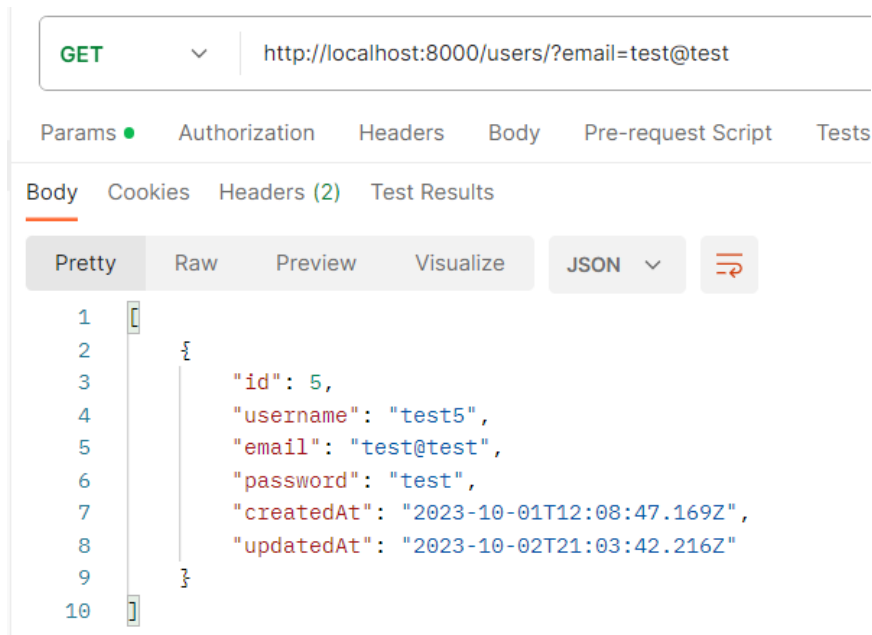
app.route("/users/:id").put(async (req, res) => {
  // Update User by id
  const user = await db.User.findByPk(req.params.id);
  if (user) {
    try {
      user.update(req.body, {where: {id: req.params.id}});
      return res.send("Success")
    } catch (e) {
      return res.send(e)
    }
  } else {
    return res.send("User does not exist")
  }
})
```

Проверка посредством Postman

Получить всех пользователей



Получить пользователя по email'у



Создать пользователя

POST

http://localhost:8000/users/

ParamsAuthorizationHeadersBodyPre-request ScriptTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

1

2

3

4

5

```
{
  "username": "test8",
  "email": "test8@gmail.com",
  "password": "test8"
}
```

BodyCookiesHeaders (2)Test Results

Pretty

Raw

Preview

Visualize

HTML

1 Success

GET

http://localhost:8000/users/9

ParamsAuthorizationHeadersBodyPre-request ScriptTestsSettings

Query Params

	Key	Value
--	-----	-------

BodyCookiesHeaders (2)Test Results

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

```
{
  "id": 9,
  "username": "test8",
  "email": "test8@gmail.com",
  "password": "test8",
  "createdAt": "2023-10-04T12:29:55.409Z",
  "updatedAt": "2023-10-04T12:29:55.409Z"
}
```

Изменить пользователя

PUT

▼

http://localhost:8000/users/9

ParamsAuthorizationHeadersBody ●Pre-request ScriptTestsSettings

● none

● form-data

● x-www-form-urlencoded

● raw

● binary

● GraphQL

JSON ▼

1

2

3

5

2

5

BodyCookiesHeaders (2)Test Results

PrettyRawPreviewVisualizeHTML ▼

1 Success

Удалить пользователя

DELETE

▼

http://localhost:8000/users/9

ParamsAuthorizationHeadersBody ●Pre-request ScriptTestsS

Query Params

	Key	Value
	Key	Value

BodyCookiesHeaders (2)Test Results

PrettyRawPreviewVisualizeHTML ▼

1 Success

GET

▼

http://localhost:8000/users/9

ParamsAuthorizationHeadersBodyPre-request ScriptTestsS

Query Params

	Key	Value
--	-----	-------

BodyCookiesHeaders (2)Test Results

PrettyRawPreviewVisualizeHTML ▼

1 User does not exist

Вывод

Была изучена работа Express и Sequelize, продумана и реализована собственная модель пользователя, реализован набор из CRUD-методов для работы с пользователями, написан запрос для получения пользователя по id и email.