

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Домашняя работа

Выполнил:

Халимов Тимур

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

```
@Table ({modelName: 'Users'...})
class Users extends Model {
  @PrimaryKey
  @AutoIncrement
  @Column

  id: number;

  @Column

  username: string;

  @Column

  name: string;

  @Column({
    type: DataTypes.DATEONLY
  })

  birthday: Date;

  @Column

  password: string

  @Column

  email: string
}

export { Users };
```

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

READ

```
app.get('/users', async (req, res) => {  
  const users = await Users.findAll();  
  if (users) {  
    return res.send(users);  
  }  
  return res.send({"msg" : "Users not found"});  
})
```

GET localhost:3333/users/

Params • Authorization Headers (6) Body Pre-request Scr

Query Params

KEY

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON ⌵

```
1  [  
2    {  
3      "id": 1,  
4      "username": "username3",  
5      "name": "name3",  
6      "birthday": "2002-01-10",  
7      "password": "passwd3",  
8      "email": null,  
9      "createdAt": "2023-03-13T18:53:22.459Z",  
10     "updatedAt": "2023-03-13T18:53:22.459Z"  
11   },  
12   {  
13     "id": 2,  
14     "username": "username2",  
15     "name": "name2",  
16     "birthday": "2002-01-20",  
17     "password": "passwd2",  
18     "email": "email2@gmail.com",  
19     "createdAt": "2023-03-13T19:02:25.538Z",  
20     "updatedAt": "2023-03-13T19:02:25.538Z"  
21   }  
22 ]
```

CREATE

```
app.post('/users', (req, res) => {  
  const created = Users.create(req.query);  
  if (created) {  
    res.send("User was created!")  
  }  
})
```

POST

localhost:3333/users/?username=username2&email=email2@gmail.com&password=pa

Params **●** Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE
<input checked="" type="checkbox"/>	username	username2
<input checked="" type="checkbox"/>	email	email2@gmail.com
<input checked="" type="checkbox"/>	password	passwd2
<input checked="" type="checkbox"/>	birthday	2002-01-20
<input checked="" type="checkbox"/>	name	name2

Database Structure Browse Data Edit Pragmas Execute SQL

Table:

Users

	id	username	name	birthday	password	email	createdAt	updatedAt
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	2	username2	name2	2002-01-20	passwd2	email2@gm...	2023-03-13 ...	2023-03-13 ...

UPDATE

```
app.put('/users', async (req, res) => {
  try
  {
    console.log(req)
    const has_user = await Users.findByPk(req.query.id);
    if (has_user) {
      const updated_user = await Users.update(req.query, {
        where: {
          id: req.query.id
        }
      })
      if (updated_user) {
        res.send('user updated')
      }
    }
    else {
      res.send('User not found')
    }
  }
  catch(error) {
    res.send({"error": error.message})
  }
})
```

PUT localhost:3333/users/?username=edited_username2&email=email2@gmail.com&pass

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE
<input checked="" type="checkbox"/>	username	edited_username2
<input checked="" type="checkbox"/>	email	email2@gmail.com
<input checked="" type="checkbox"/>	password	passwd2
<input checked="" type="checkbox"/>	birthday	2002-01-20
<input checked="" type="checkbox"/>	name	name2
<input checked="" type="checkbox"/>	id	2

Database Structure Browse Data Edit Pragmas Execute SQL

Table: Users

	id	username	name	birthday	password	email	createdAt	updatedAt
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	username3	name3	2002-01-10	passwd3	NULL	2023-03-13 ...	2023-03-13 ...
2	2	edited_username2	name2	2002-01-20	passwd2	email2@gm...	2023-03-13 ...	2023-03-13 ...

DELETE

```
app.delete('/users', async (req, res) => {
  try
  {
    const has_user = await Users.findByPk(req.query.id);
    if (has_user) {
      const deleted_user = await Users.destroy({
        where: {
          id: req.query.id
        }
      })
      if (deleted_user) {
        res.send("User deleted!")
      }
    }
    else {
      res.send('User not found')
    }
  }
  catch(error) {
    res.send({"error": error.message})
  }
})
```

DELETE



localhost:3333/users/?id=2

Params



Authorization

Headers (6)


Body

Pre-request Script

Tests

Settings

Query Params

	KEY	VALUE
 <input checked="" type="checkbox"/>	id	2

Database Structure

Browse Data

Edit Pragmas

Execute SQL

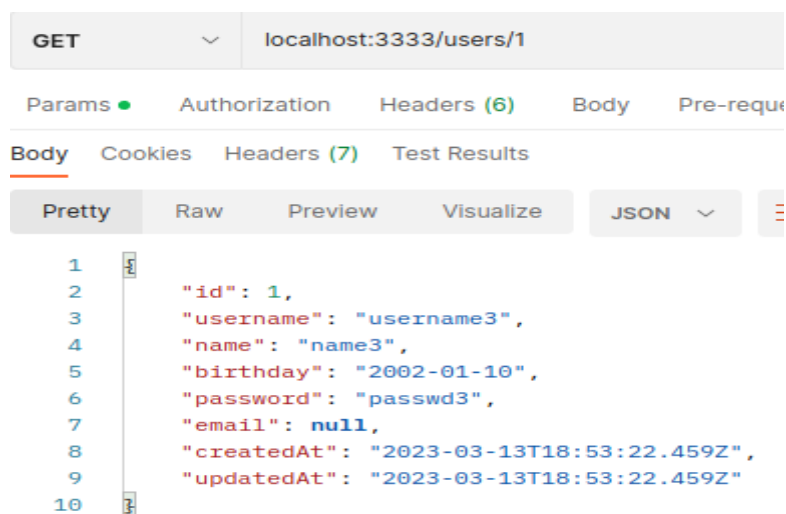
Table: Users



	id	username	name	birthday	password
	Filter	Filter	Filter	Filter	Filter
1	1	username3	name3	2002-01-10	passwd3

3. Получение пользователя по id:

```
app.get('/users/:id', async (req, res) => {  
  const user = await Users.findByPk(req.params.id);  
  if (user) {  
    return res.send(user);  
  }  
  return res.send('User not found')  
})
```



Вывод

В ходе выполнения работы была продумана модель пользователя, были написаны CRUD - методы, а также был реализован метод для получения пользователя по id.