

1. Base de datos de usuarios con los parametros:

- id
- name
- last_name
- age
- email
- active (true, false)

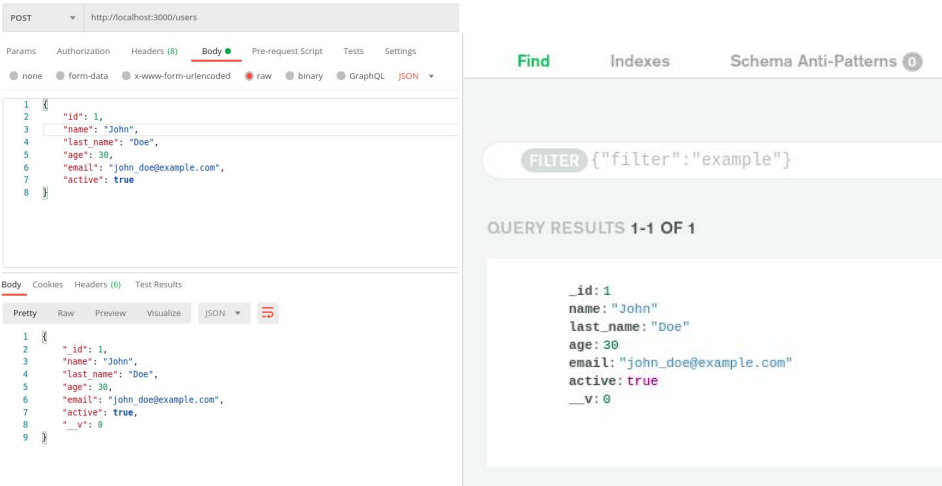
```
src > models > .# users.js > ...
1 const mongoose = require('mongoose');
2 const autoIncrement = require('mongoose-auto-increment');
3 const Schema = mongoose.Schema;
4
5 const userSchema = new Schema({
6
7   id: {
8     type: Number,
9   },
10  name: {
11    type: String,
12    required: true
13  },
14  last_name: {
15    type: String,
16    required: true
17  },
18  age: {
19    type: Number,
20    required: true
21  },
22  email: {
23    type: String,
24    required: true
25  },
26  active: {
27    type: Boolean,
28    required: true
29  }
30 });
31
32 autoIncrement.initialize(mongoose.connection);
33 userSchema.plugin(autoIncrement.plugin, {
34   model: "users",
35   field: "id",
36   startAt: 1,
37   incrementBy: 1
38 });
39 module.exports = mongoose.model('users', userSchema);
```

2. CRUD:

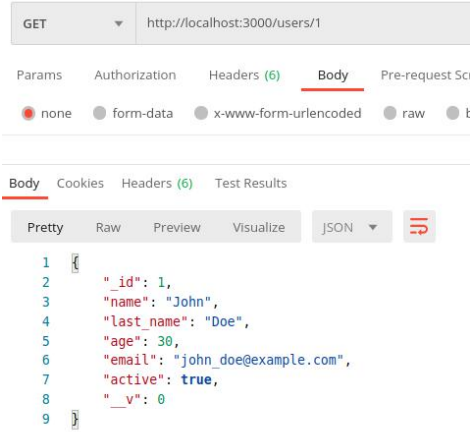
```
src > routes > .# users.js > ...
1 const express = require('express');
2 const router = express.Router();
3 const User = require('../models/user');
4
5 // GET
6 router.get('/', async (req, res) => {
7   try{
8     const users = await User.find();
9     res.json(users);
10  } catch(err){
11    res.json({message: err});
12  }
13 });
14
15 // GET BY ID
16 router.get('/:userId', async (req, res) => {
17   try{
18     const userById = await User.findById(req.params.userId);
19     res.json(userById);
20   }
21   catch (err) {
22     res.json({message: err});
23   }
24 });
25
26 // UPDATE
27 router.patch('/:userId', async (req, res) => {
28   try{
29     const updatedById = await User.updateOne(
30       { _id: req.params.userId },
31       {
32         $set: {
33           name: req.body.name,
34           last_name: req.body.last_name,
35           age: req.body.age,
36           email: req.body.email,
37           active: req.body.active
38         }
39       }
40     );
41     res.json(updatedById);
42   }
43   catch (err) {
44     res.json({message: err});
45   }
46 });
47
48 // DELETE
49 router.delete('/:userId', async (req, res) => {
50   try{
51     const removedById = await User.remove({ _id: req.params.userId });
52     res.json(removedById);
53   }
54   catch (err) {
55     res.json({message: err});
56   }
57 });
58
59 module.exports = router;
```

```
26 // POST
27 router.post('/', async (req, res) => {
28   const post = new User({
29     _id: req.body.id,
30     name: req.body.name,
31     last_name: req.body.last_name,
32     age: req.body.age,
33     email: req.body.email,
34     active: req.body.active
35   });
36   try{
37     const savedUser = await post.save();
38     res.json(savedUser);
39   } catch (err){
40     res.json({message: err});
41   }
42 });
43
44 // DELETE
45 router.delete('/:userId', async (req, res) => {
46   try{
47     const removedById = await User.remove({ _id: req.params.userId });
48     res.json(removedById);
49   }
50   catch (err) {
51     res.json({message: err});
52   }
53 });
54
55 module.exports = router;
```

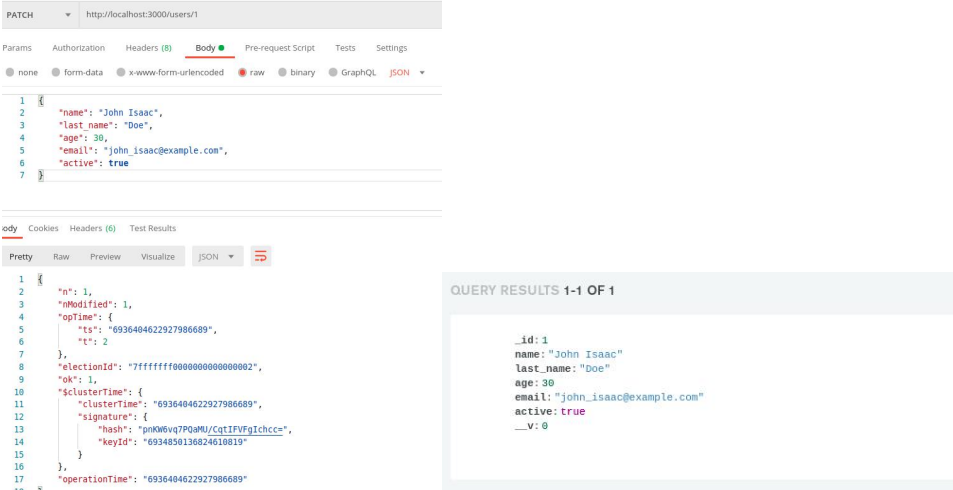
3. Creación de usuario a través de API



4. Consulta de usuario creado



5. Modificación de usuario



6. Eliminar usuario

DELETE

http://localhost:3000/users/1

Params

Authorization

Headers (6)

Body

Pre-request Script

Body

Cookies

Headers (6)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

"n": 1,

2

"opTime": {

3

"ts": "6936405224223408129",

4

"t": 2

5

},

6

"electionId": "7ffffffff00000000000000002",

7

"ok": 1,

8

"\$clusterTime": {

9

"clusterTime": "6936405224223408129",

10

"signature": {

11

"hash": "AUqZvAab6f8nCudKlbqvyMxCsE=",

12

"keyId": "6934850136824610819"

13

}

14

},

15

"operationTime": "6936405224223408129",

16

"deletedCount": 1

17

18

GET

http://localhost:3000/users/1

Params

Authorization

Headers (6)

Body

Pre-request Script

Body

Cookies

Headers (6)

Test Results

Pretty

Raw

Preview

Visualize

JSON

1

null