1. Base de datos de usuarios con los parametros:

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

src > models > JS user.js > ...

```
const mongoose = require('mongoose');
const autoIncrement = require("mongoose-auto-increment");
const Schema = mongoose.Schema;

const userSchema = new Schema({

    _id: {
        type: Number,
    },
    name: {
        type: String,
        required: true
    },
    last_name: {
        type: String,
        required: true
    },
    age: {
        type: Number,
        required: true
    },
    email: {
        type: String,
        required: true
    },
    active: {
        type: Boolean,
        required: true
    }
});

autoIncrement.initialize(mongoose.connection);
userSchema.plugin(autoIncrement.plugin, {
    model: "users",
    field: "_id",
    startAt: 1,
    incrementBy: 1
  });
module.exports = mongoose.model('users', userSchema);
```

2. CRUD:

src > routes > JS users.js > ...

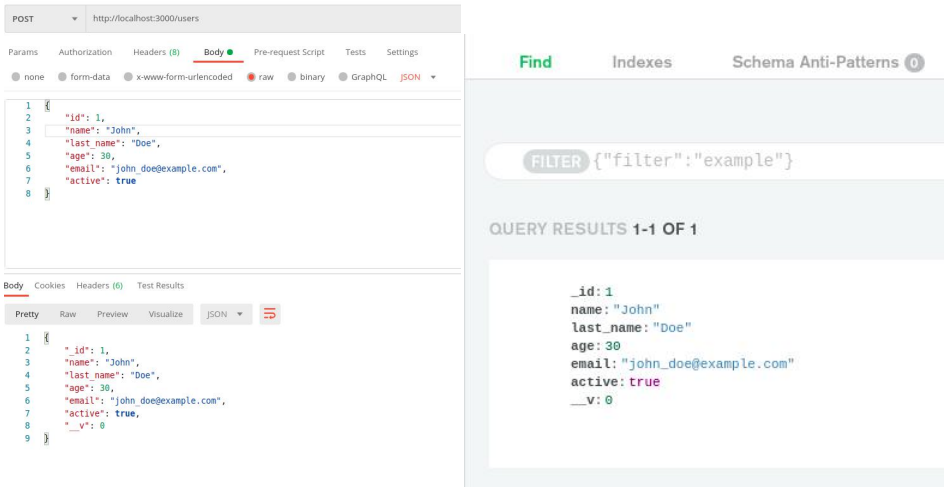
```
const express = require('express');
const router = express.Router();
const User = require('../models/user');

// GET
router.get('/', async (req, res) => {
    try{
        const users = await User.find();
        res.json(users);
    }catch(err){
        res.json({message: err});
    }
});

// GET BY ID
router.get('/:userId', async (req, res) => {
    try{
        const userById = await User.findById(req.params.userId);
        res.json(userById);
    }
    catch (err) {
        res.json({message: err});
    }
});
```

```
// POST
router.post('/', async (req, res) => {
    const post = new User({
        _id: req.body.id,
        name: req.body.name,
        last_name: req.body.last_name,
        age: req.body.age,
        email: req.body.email,
        active: req.body.active
    });
    try{
        const savedUser = await post.save();
        res.json(savedUser);

    } catch (err){
        res.json({message: err});
    }
});
```

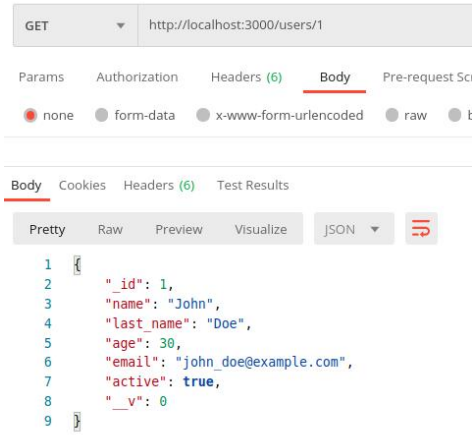
```
// UPDATE
router.patch('/:userId', async (req, res) => {
    try{
        const updatedById = await User.updateOne(
            {_id: req.params.userId},
            {
                $set: {
                    name: req.body.name,
                    last_name: req.body.last_name,
                    age: req.body.age,
                    email: req.body.email,
                    active: req.body.active
                }
            }
            );
        res.json(updatedById);
    }
    catch (err) {
        res.json({message: err});
    }
});
```

```
// DELETE
router.delete('/:userId', async (req, res) => {
    try{
        const removedById = await User.remove({_id: req.params.userId});
        res.json(removedById);
    }
    catch (err) {
        res.json({message: err});
    }
});

module.exports = router;
```

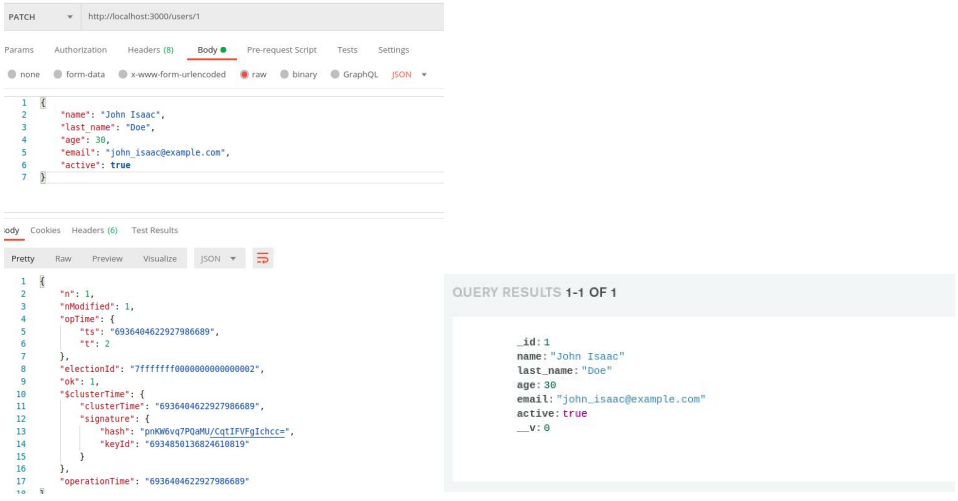
3. Creación de usuario a tráves de API



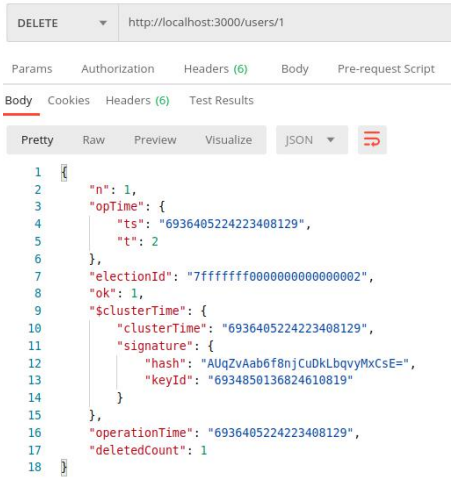
4. Consulta de usuario creado

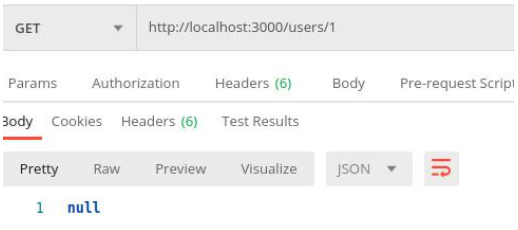


5. Modificación de usuario



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
{
"n": 1,
"opTime": {
"ts": "6936405224223408129",
"t": 2
},
"electionId": "7fffffff0000000000000002",
"ok": 1,
"$clusterTime": {
"clusterTime": "6936405224223408129",
"signature": {
"hash": "AUqZvAab6f8njCuDkLbqvyMxCsE=",
"keyId": "6934850136824610819"
}
},
"operationTime": "6936405224223408129",
"deletedCount": 1
}


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