SERVIDOR API-RESTful

Una vez instalado el **SO**, necesitamos un **IDE** de programación que nos facilite la escritura de nuestra aplicación que soporte tecnologías actuales, como **NodeJs**, **Express**, con lenguajes como **JavaScript** o **TypeScript** y tecnologías para la parte Front como **Angular**.

Para ello necesitamos instalar **Visual Studio Code(VSC)** como **IDE** de desarrollo, esto lo podemos hacer fácilmente abriendo la terminal y ejecutando el siguiente comando:

```
sudo snap install --classic code
```

Gracias a este **snap** no necesitamos buscar el **IDE** en un explorador.

Lo siguiente que necesitamos es **Instalar** en nuestra maquina es **NodeJs**, esto es igual de fácil gracias al comando:

```
sudo apt install nodejs
```

Comprobamos que se ha instalado la versión correcta con:

```
nodejs -v
```

Si el paquete de los repositorios se ajusta a las necesidades, será todo lo que necesita para configurar **Node.js**. En la mayoría de los casos, también convendrá instalar **NPM**, el administrador de paquetes de **Node.js**. Puede hacer esto instalando el paquete **npm** con:

```
sudo apt install npm
```

Para mayor facilidad se instalara utilidades para mantener las versiones de **Node** y la versión estable de este mismo con:

```
$ sudo npm clean -f
$ sudo npm i -g n
$ sudo n stable
```

Para el manejo de versiones es recomendable usar **GitHub** o **BitBucket**, en mi caso, este primero. Para instalarlo en el sistema usaremos el comando:

```
sudo apt install git
```

Una vez instalado necesitaremos configurarlo con :

```
git config --global user.name NombreUsuario
git config --global credential.helper store
```

Elegiremos la ubicación del proyecto, podemos hacerlo desde la terminal con **cd ubicación y mkdir *nombreArchivo*** o haciendolo con Click derecho en la ubicación deseada y crear nueva carpeta.

Una vez creada una carpeta con el nombre, dentro crearemos otra con el nombre de **api-rest** donde crearemos el servidor y las operaciones **CRUD**

Una vez creado, lo subiremos a nuestro repositorio en **GitHub** con el comando '**git add .**' (añade los archivos nuevos/modificaciones del directorio actual) o '**git add ***' (todos los archivos del repositorio) y haremos un **commit** con:
git commit -m "nombre commit"

para subirlo, usaremos el comando

git push

pero al ser la primera vez necesitaremos indicar el origen con

git remote add origin <nombre><url>
y luego un push a git push <nombre>

Si había algo en ese repositorio podemos usar el comando **git fetch** para traerlo.
Ahora podemos empezar un proyecto con **NPM INIT**
Le ponemos un nombre al **package**, si no queremos poner una versión, descripción, entryPoint, testCommand… pulsamos ENTER
Esto se podría modificar mas adelante en el "package.json"

Si todo esta **por defecto**, necesitaremos un archivo **index.js** para escribir el código necesario para poner en marcha el servidor.

mijs.js > http.createServer() callback

```
1  var http = require('http');
2
3  http.createServer( (request, response) => {
4      response.writeHead(200, {'Content-Type': 'text/plain'});
5      response.end('Creando Nuestro Servidor\n');
6  }).listen(8080);
7
8  console.log('Servidor ejecutándose en puerto 8080...');
```

Una vez puesto y guardado esto. Podemos ejecutarlo gracias a Node con el comando:
Node index.js (en mi caso mijs.js)

Ahora instalemos unas bibliotecas para facilitar el trabajo como **Express** que nos facilitara el control de recursos HTTP y gestión de métodos.

Morgan para el registro de 'logs'.

Nodemon para evitar que tengamos que estar constantemente reiniciando nuestra aplicación con cada cambio en el código.

Todo esto lo podemos instalar con :

```
npm i -S express
npm i -S morgan
npm i -D nodemon
```

También para mayor facilidad iremos al **Package.json** y añadiremos en "**scripts**" el comando **start** con "nodemon mijs.js"
A continuación veremos una imagen de como queda una vez instalado todo y añadido el script de inicio.

También para todo esto necesitaremos una base de Datos, y aquí entra en juego **MongoDB** que será nuestra **BBDD** no relacional con:

```
sudo apt update
sudo apt install -y mongodb
```

Para decirle tu computadora que deje el a mongo encendido cuando encendamos el ordenador usaremos el comando:

```
sudo systemctl start mongo
```

y podemos comprobar que esta en funcionamiento con:

mongo –eval db.runCommand({ connectionStatus: 1 })'

```
magic@magic-GL62-6QF:~/Escritorio/SD/api-rest$ mongo --eval 'db.runCommand({ c
onnectionStatus: 1 })'
MongoDB shell version v3.6.8
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("a753840f-c1de-4e5d-88c6-d168074a8c78")
}
MongoDB server version: 3.6.8
{
        "authInfo" : {
                "authenticatedUsers" : [ ],
                "authenticatedUserRoles" : [ ]
        },
        "ok" : 1
}
magic@magic-GL62-6QF:~/Escritorio/SD/api-rest$
```

$mongo
>show dbs

```
magiic@magiic-GL62-6QF:~/Escritorio/SD/api-rest$ mongo
MongoDB shell version v3.6.8
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("08842e76-17cc-4ed6-8bb7-e3691cabb1a2") }
MongoDB server version: 3.6.8
Server has startup warnings:
2022-03-03T22:12:26.862+0100 I STORAGE  [initandlisten]
2022-03-03T22:12:26.862+0100 I STORAGE  [initandlisten] ** WARNING: Using the XFS f
2022-03-03T22:12:26.862+0100 I STORAGE  [initandlisten] **          See http://doch
2022-03-03T22:12:30.653+0100 I CONTROL  [initandlisten]
2022-03-03T22:12:30.653+0100 I CONTROL  [initandlisten] ** WARNING: Access control
2022-03-03T22:12:30.653+0100 I CONTROL  [initandlisten] **          Read and write
2022-03-03T22:12:30.653+0100 I CONTROL  [initandlisten]
> show dbs
SD      0.000GB
admin   0.000GB
config  0.000GB
local   0.000GB
>
```

Para enlazar **MongoDB** con nuestro proyecto usaremos las bibliotecas de de mongo descargandolas con **NODE** en nuestra api-rest:

npm i -S mongodb
npm i -S mongojs

```
{
  "name": "api-rest",
  "version": "1.0.0",
  "description": "Proyecto de API RESTfil con Node y Express",
  "main": "mijs.js",
  ▷ Debug
  "scripts": {
    "start": "nodemon mijs.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "CRUD",
    "REST",
    "node",
    "express",
    "mongo"
  ],
  "author": "Vladislav",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.2",
    "mongodb": "^4.4.0",
    "mongojs": "^3.1.0",
    "morgan": "^1.10.0"
  },
  "devDependencies": {
    "nodemon": "^2.0.15"
  }
}
```

Mejoramos un poco nuestro código del servidor añadiendo estas funciones de **EXPRESS, MORGAN y MongoDB.** Cambiamos el puerto al 3000, usado para desarrollo. Y quedará algo así.

```
3   'use strict'
4
5   const port    = process.env.port || 3000;
6   const express = require('express');
7   const logger  = require('morgan');
8   const mongojs = require('mongojs');
9   const { request, response } = express;
10
11  var db = mongojs("SD");
12  var id = mongojs.ObjectID;
13  //var db = mongojs('username:password@example.com/SD');
14
15
16  const app = express();
17
18
19  // Declaramos los middleware
20  app.use(logger('dev'));
21  app.use(express.urlencoded({ extended: false }));
22  app.use(express.json());
```

Podemos probar esto en el navegador con
"http//localhost:3000/api/product"
Solo falta añadir el resto de operaciones **CRUD** (get, post, put, delete).

```
app.param("coleccion", (request, response, next, coleccion)=>{
    console.log('param/api/:coleccion');
    console.log('coleccion: ',coleccion);

    request.coleccion = db.collection(coleccion);
    return next();
});

// el servicio se puede llamar a una funcion o crearlo directamente

app.listen(port , () => {
    console.log(` API RESTFul CRUD ejecutandose desde http//localhost:${port}/api/:coleccion:id`);
});
//GET
app.get( '/api', (request, response, next) =>{
    console.log(request.params);
    console.log(request.collection);

    db.getCollectionNames((err, colecciones) => {
       if (err) return next(err);
       response.json(colecciones);
    });

});
 app.get('/api/:coleccion', (request, response, next) => {
    request.coleccion.find((err, coleccion) => {
       if (err) return next(err);
       response.json(coleccion);
   });
 });
app.get( '/api/:coleccion/:id', (request, response, next) =>{
    console.log(request.params);
    console.log(request.collection);

    request.coleccion.findOne({_id: id(request.params.id)}, (err, elemento) => {
        if(err) return next(err);
        response.json(elemento);
    });
});
app.post( '/api/:coleccion', (request, response, next) =>{
    console.log(request.body);
    const elemento = request.body;

   if (!elemento.nombre) {
    response.status(400).json ({
     error: 'Bad data',
     description: 'Se precisa al menos un campo <nombre>'
       });
   } else {
    request.coleccion.save(elemento, (err, coleccionGuardada) => {
           if(err) return next(err);
           response.json(coleccionGuardada);
      });
   }
 })
//pasamos el ID por valor
//PUT
app.put('/api/:coleccion/:id', (request, response, next) =>{
    let elementoId = request.params.id;
    let elementoNuevo = request.body;
    request.coleccion.update({_id: id(elementoId)},
           {$set: elementoNuevo}, {safe: true, multi: false}, (err, elementoModif) => {
       if (err) return next(err);
       response.json(elementoModif);
   });
 });


//borramos por id,
//DELETE
app.delete('/api/:coleccion/:id', (request, response, next) => {
   let elementoId = request.params.id;

   request.coleccion.remove({_id: id(elementoId)}, (err, resultado) => {
       if (err) return next(err);
       response.json(resultado);
    });
});
```

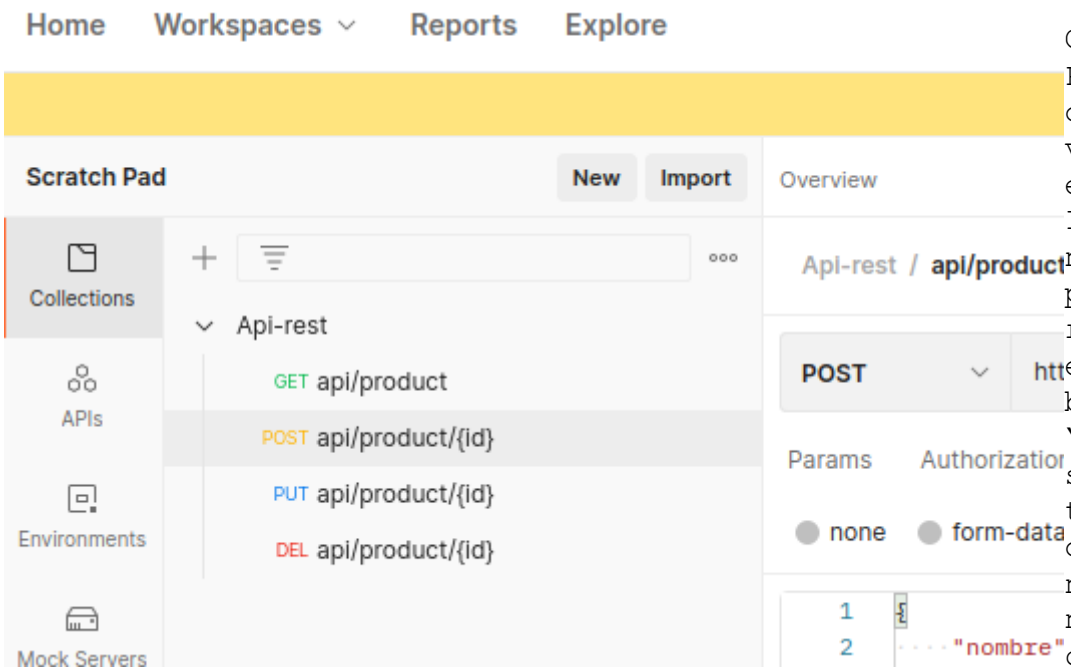
Recuerda ir guardando el repositorio con
git add *, git commit -m "", git push

Puedes poner **tags** para las versiones y tengas un mejor control con **git tags <version>1.0.1**

El navegador solo nos permite ver peticiones **GET**, para comprobar que el resto de operaciones **CRUD** funcionan correctamente (post, put, delete) podemos usar **POSTMAN**, una aplicación que procesa las request. Podemos descargarlo con

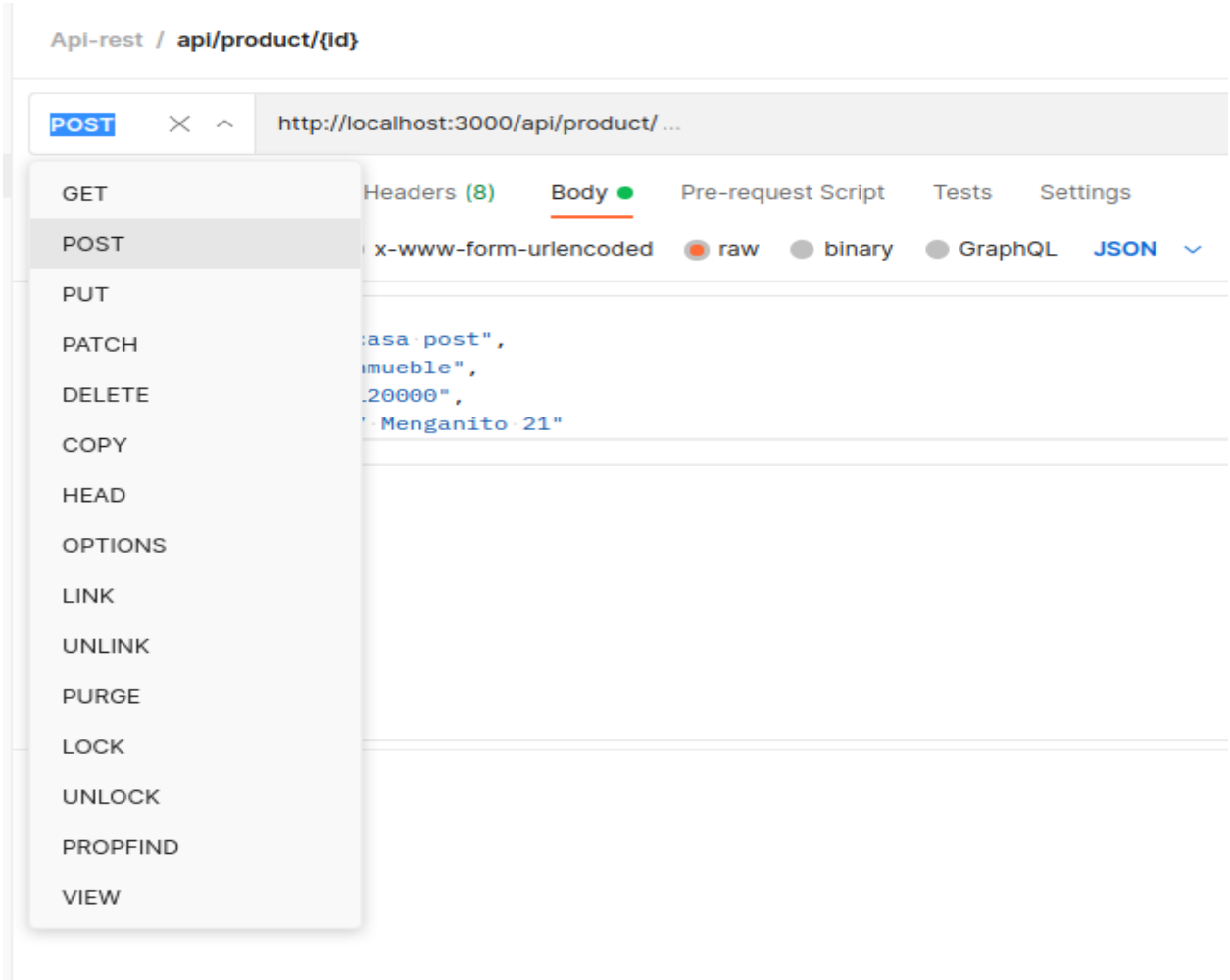
sudo snap install postman

Una vez descargado, para ejecutarlo solo necesitaremos escribir postman en la terminal o buscarlo en Aplicaciones.



Como Podemos ver en la Imagen pulsaremos el botón "***NEW***" si no tenemos ninguna colección o **request**. Una vez creada una **Colección** podemos ir guardando **requests** como se puede apreciar de **GET**, **POST**, **PUT**, **DELETE**

Seleccionamos La **Resquest** que deseamos y ponemos al lado la **URL** a la que queremos **lanzar la petición**



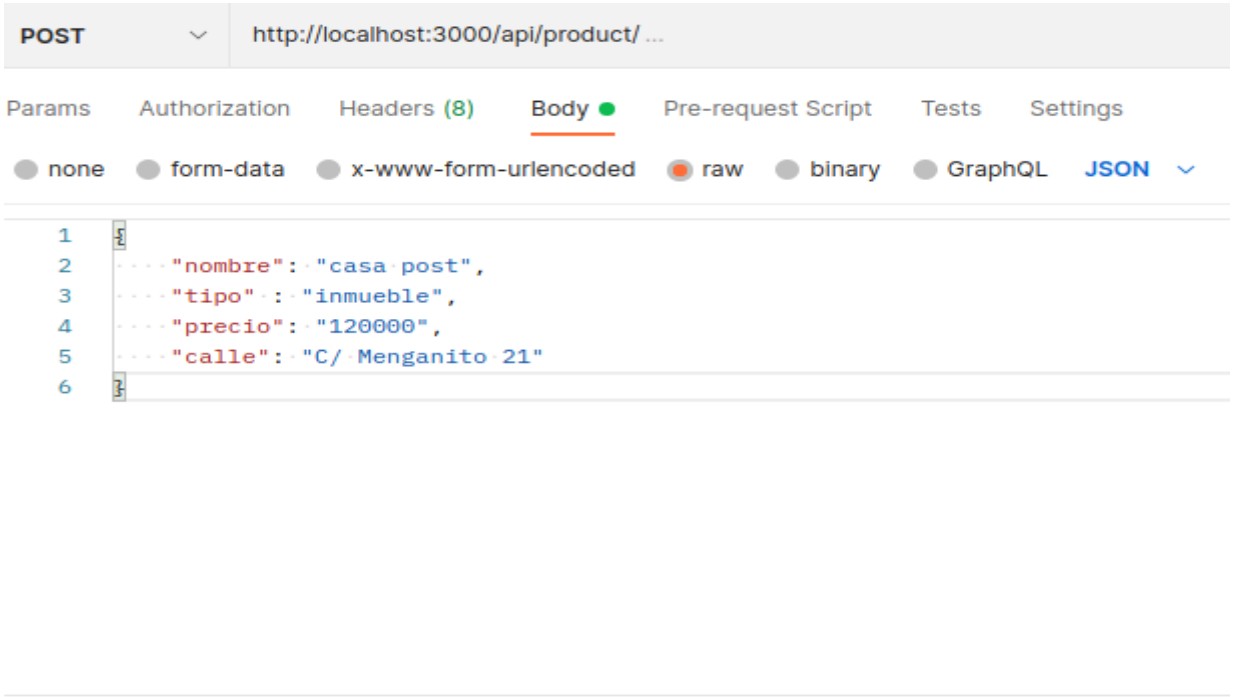
No Olvides tener el servidor encendido antes de mandar peticiones, sera algo similar a:

```
magic@magic-GL62-6QF:~$ cd Escritorio/SD/api-rest/
magic@magic-GL62-6QF:~/Escritorio/SD/api-rest$ npm start

> api-rest@1.0.0 start
> nodemon mijs.js

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node mijs.js`
 API RESTFul CRUD ejecutandose desde http//localhost:3000/api/product
```

Una vez mandada la peticion con un objeto **Json** (si lo mandas en **RAW** hay que seleccionar Json un poco mas a la derecha) quedará asi:



```
magiic@magiic-GL62-6QF:~/Escritorio/SD/api-rest$ npm start

> api-rest@1.0.0 start
> nodemon mijs.js

[nodemon] 2.0.15
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node mijs.js`
 API RESTFul CRUD ejecutandose desde http//localhost:3000/api/product
{
  nombre: 'casa post',
  tipo: 'inmueble',
  precio: '120000',
  calle: 'C/ Menganito 21'
}
POST /api/product/ 200 25.403 ms - 96
```

Y comprobamos que este en mongo mirando el dbs con:
>show dbs
>use SD
>show collections
>db.gatos.find().pretty()

```
magiic@magiic-GL62-6QF:~$ mongo
MongoDB shell version v3.6.8
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID("01209c6b-2ed5-
MongoDB server version: 3.6.8
Server has startup warnings:
2022-03-03T22:12:26.862+0100 I STORAGE  [initandlisten]
2022-03-03T22:12:26.862+0100 I STORAGE  [initandlisten]
2022-03-03T22:12:26.862+0100 I STORAGE  [initandlisten]
2022-03-03T22:12:30.653+0100 I CONTROL  [initandlisten]
2022-03-03T22:12:30.653+0100 I CONTROL  [initandlisten]
2022-03-03T22:12:30.653+0100 I CONTROL  [initandlisten]
2022-03-03T22:12:30.653+0100 I CONTROL  [initandlisten]
> show dbs
SD      0.000GB
admin   0.000GB
config  0.000GB
local   0.000GB
> use SD
switched to db SD
> show collections
casas
familia
gatos
product
> db.gatos.find().pretty()
{
        "_id" : ObjectId("621d1f3233121e17dd41de14"),
        "tipo" : "Maine Coon3",
        "nombre" : "Zarpitas",
        "edad" : 4
}
>
```