**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>**

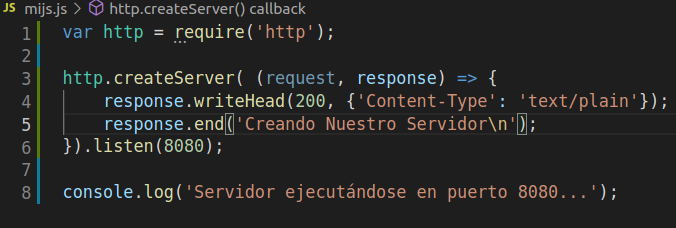
Si había algo en ese repositorio podemos usar el comando **git fetch** para traerlo.

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.



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:

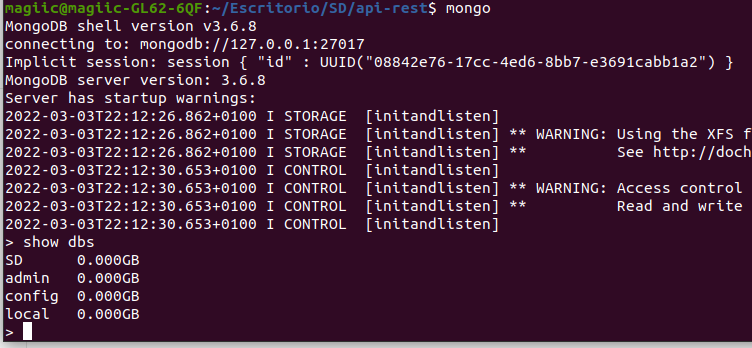
sudo systemctl start mongo

y podemos comprobar que esta en funcionamiento con:

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

****

$mongo  
>show dbs



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

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



Podemos probar esto en el navegador con **“http//localhost:3000/api/product”**

Solo falta añadir el resto de operaciones **CRUD** (get, post, put, delete).

******

Recuerda ir guardando el repositorio con

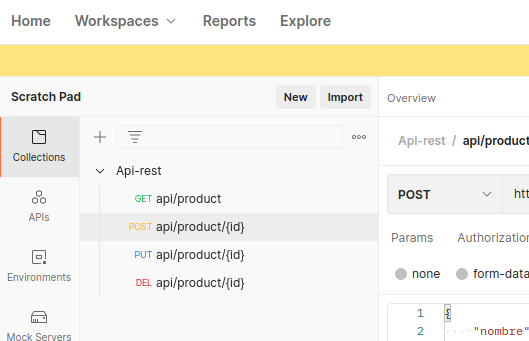
**git add \*, git commit -m “”, git push**

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**

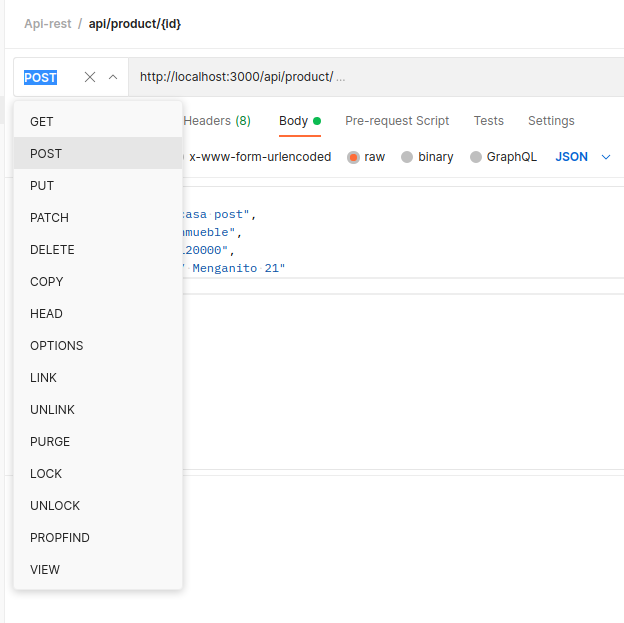
Una vez descargado, para ejecutarlo solo necesitaremos escribir postman en la terminal o buscarlo en Aplicaciones.

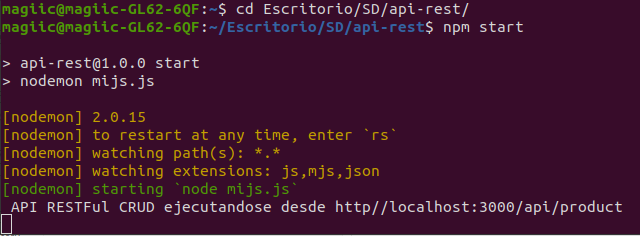


Como Podemos ver en la Imagen pulsaremos el botón “***NEW***” si no tenemos ninguna colección

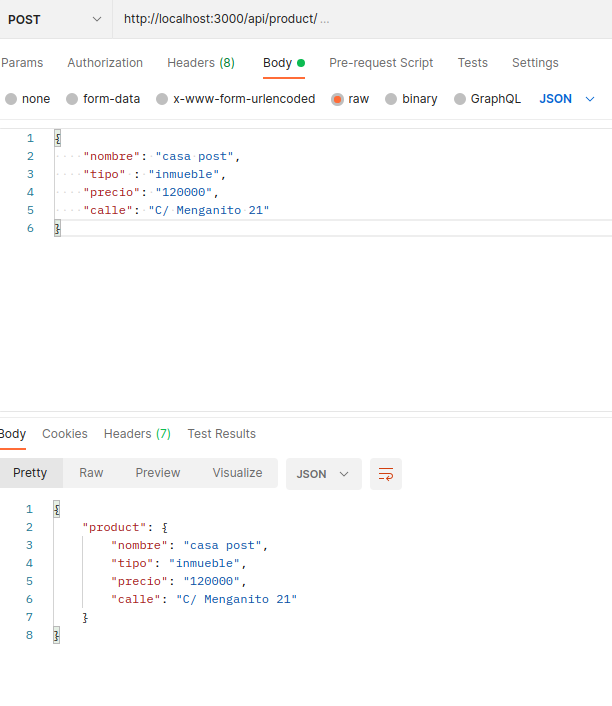
o **request**. Una vez creada una **Colección** podemos ir guardando **requests** como se puede apreciar de **GET, POST, PUT, DELETE**

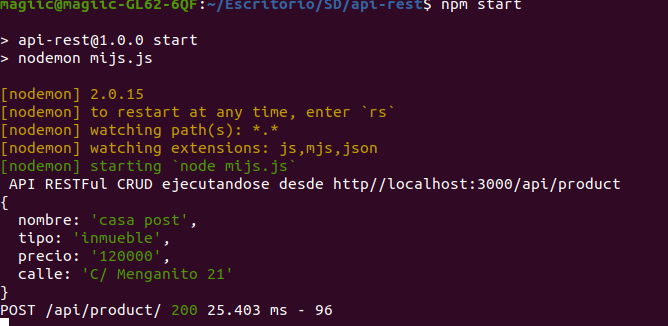
Seleccionamos La **Resquest** que deseamos y ponemos al lado la **URL** a la que queremos **lanzar la petición**

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

****

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:

****

**Y comprobamos que este en mongo mirando el dbs con:**

**>show dbs**

**>use SD  
>show collections  
>db.gatos.find().pretty()**

