

**PRUEBAS DE FUNCIONAMIENTO APLICACIÓN NODE JS**

**ANDRÉS FELIPE FRANCO TÉLLEZ - 20221978031**

**BRAYAN SANTIAGO VALERO ARIAS - 20221978007**

**DOCENTE: NOE ARCOS MUÑOZ**

**PROGRAMACIÓN POR COMPONENTES**

**UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS**

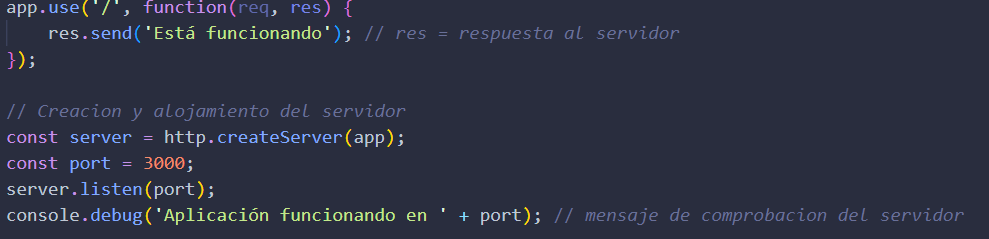
**FACULTAD TECNOLÓGICA**

**SISTEMATIZACIÓN DE DATOS**

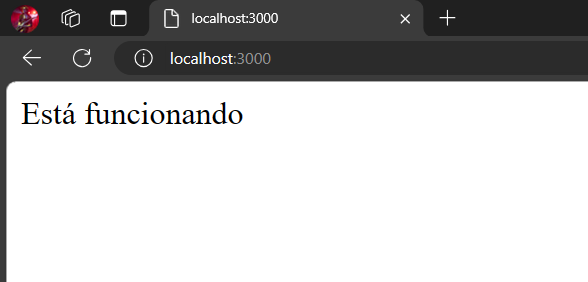
**ABRIL 2024**

**Inicio de la aplicación:**

El primer paso es la creación del servidor pues allí se alojará la aplicación creada.



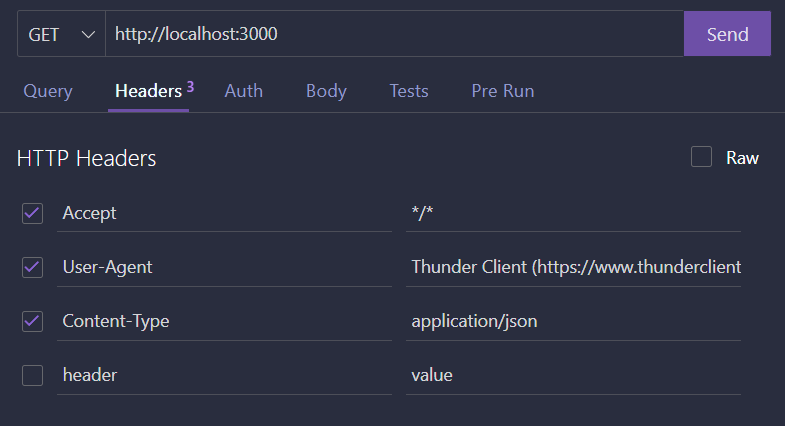
Como se puede observar el servidor será local y se alojará en el puerto 3000. A continuación, se observará el mensaje de respuesta de la página al iniciar el servidor:



**URL:** <http://localhost:3000>

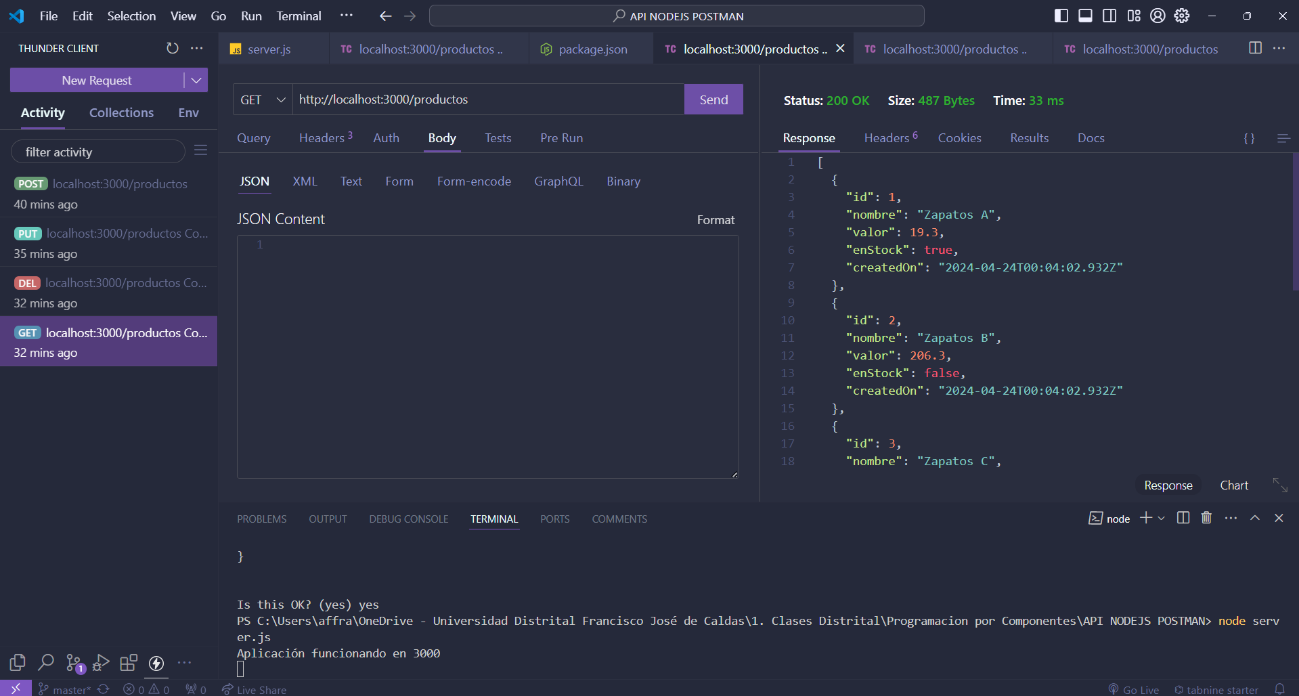
**Thunder Client:**

Para el intercambio de información usando archivos .JSON se deberá incluir la cabecera HTTP JSON en la extensión:



***Método GET:***

El método GET permite mostrar los datos que se encuentran almacenados en este caso en el arreglo “data”. En la extensión de Thunder Client se debe mandar la dirección en donde se encuentra alojada la página junto con el método GET.



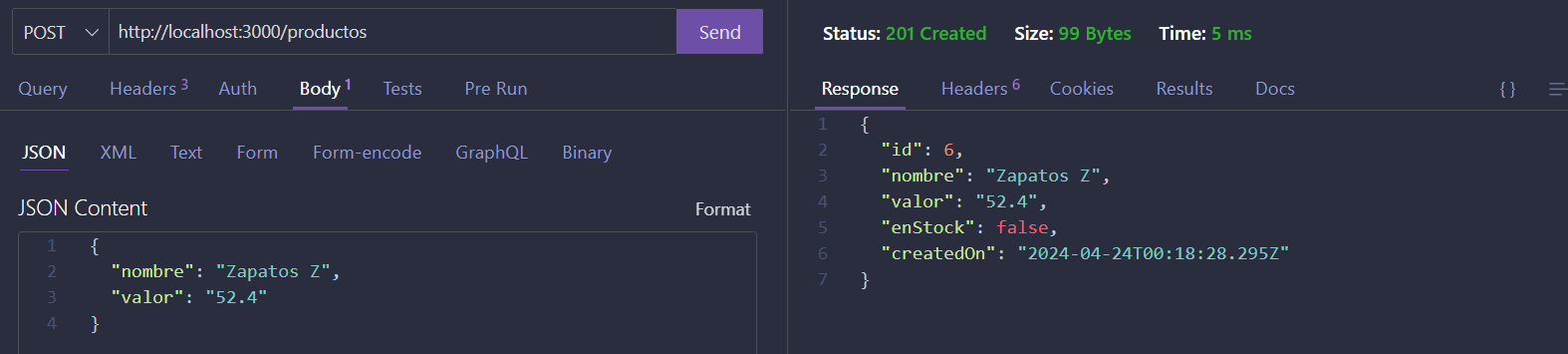
Así se ve la página desde un navegador web:



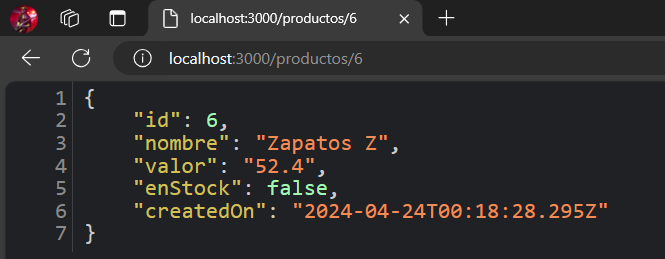
**URL:** <http://localhost:3000/productos>

***Método POST:***

El método POST agrega elementos al arreglo, el programa coge el ultimo id que fue agregado y el siguiente a agregar será el mismo ya existente más uno. Luego en el JSON que será enviado mediante POST tendrá los datos que querrán ser añadidos:



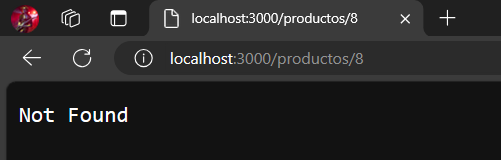
Al ser agregado se podrá consultar en la página de la siguiente manera:



**URL:** <http://localhost:3000/productos/6>

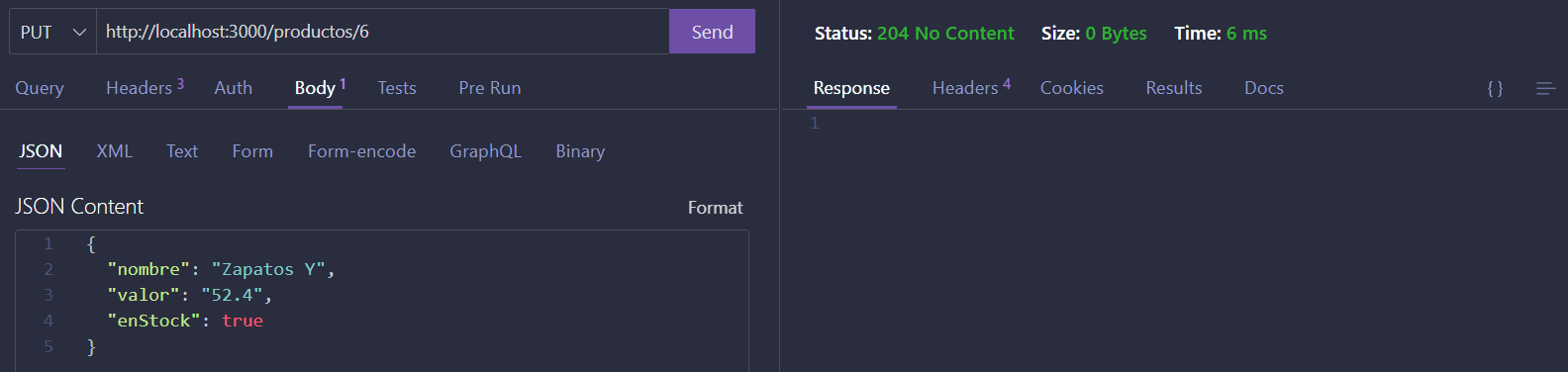
***Método PUT:***

El método de PUT es para actualizar datos de un elemento ya existente por lo cual se deberá indicar en la dirección cual es el id del elemento, si el elemento no existe mandara un código de error 404:

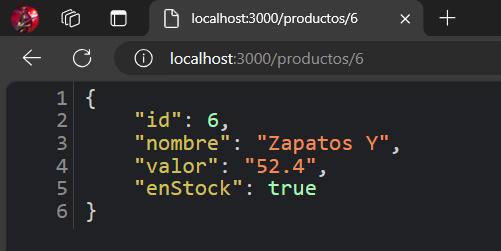


**URL:** <http://localhost:3000/productos/8>

En el caso de que si exista el elemento se mandará en el JSON mediante PUT los datos que se quieren actualizar:



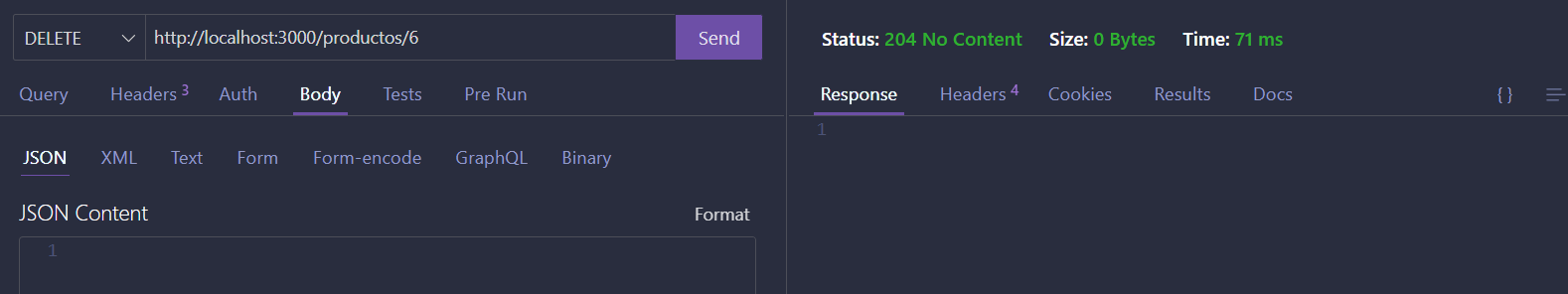
Acá se puede ver como en la página se actualizan los datos proporcionados:



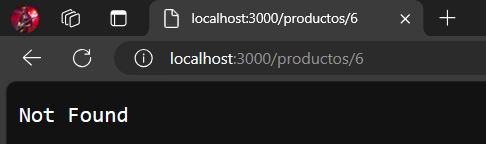
**URL:** <http://localhost:3000/productos/6>

***Método DELETE:***

Por último, el método DELETE eliminará un registro mandando un JSON sin información para que este registro “no exista”:

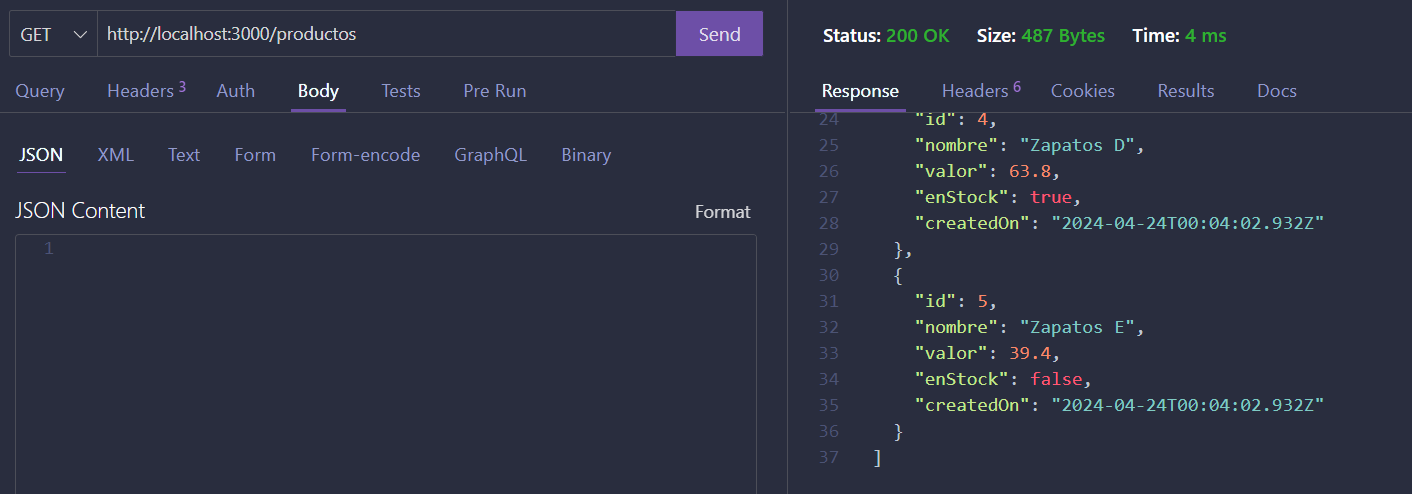


Como se puede ver al buscar el registro no se encontrará:



**URL:** <http://localhost:3000/productos/6>

Del mismo modo mediante el método GET no se encontrará el elemento:





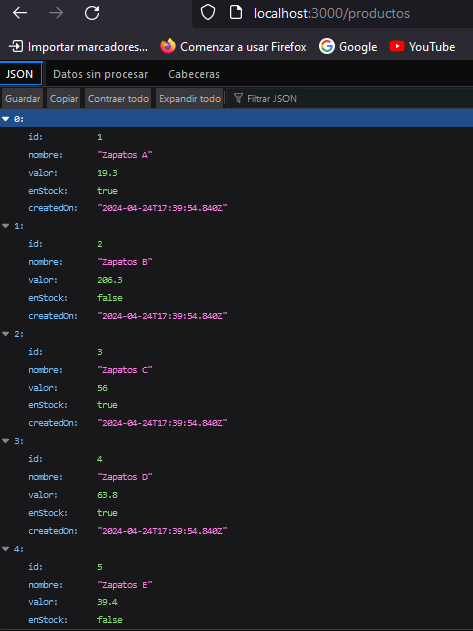
**URL:** <http://localhost:3000/productos>

Al igual que el método PUT, cada elemento deberá buscarse mediante su id, por lo que en la dirección habrá que especificarlo. Si el id no existe directamente se mostrará el código de error 404.

**POSTMAN:**

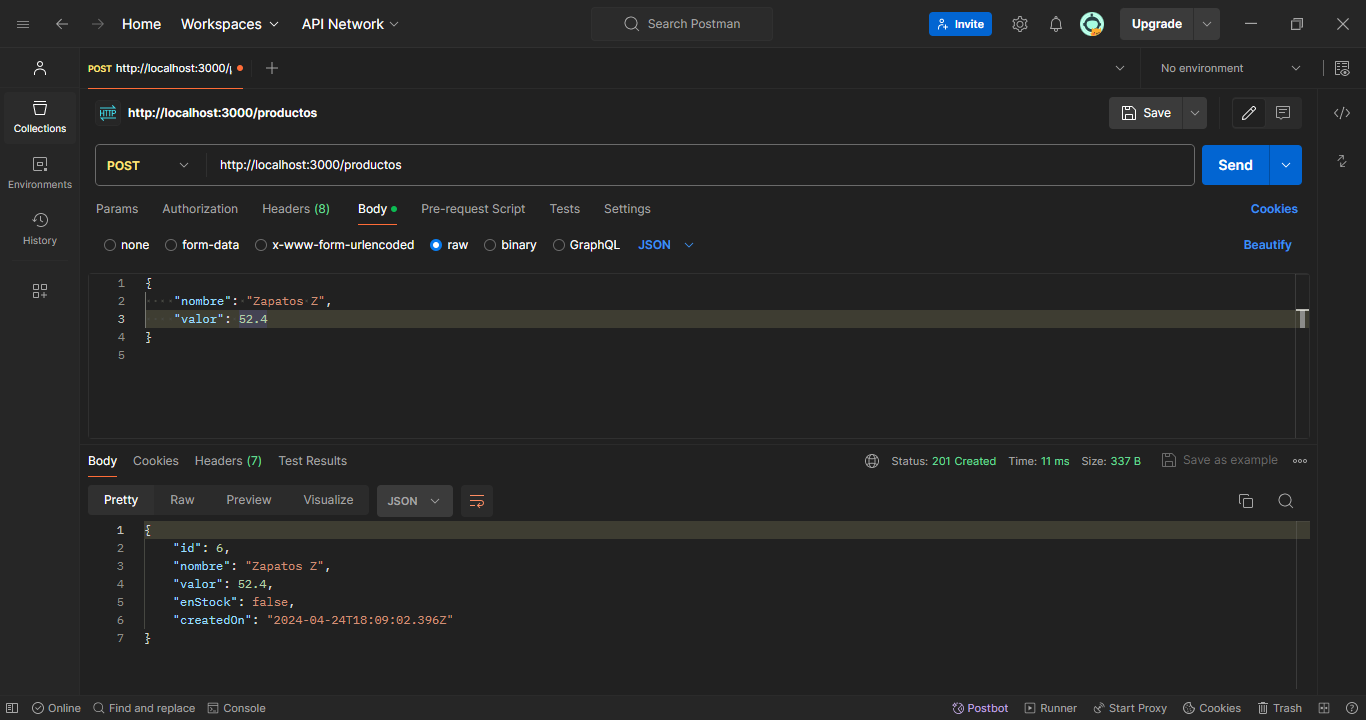
***Método GET:***

El método GET permite mostrar los datos que se encuentran almacenados en este caso en el arreglo “data”. Así se visualiza desde un navegador, usando la siguiente url: <http://localhost:3000/productos>

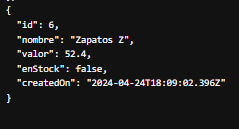


***Método POST:***

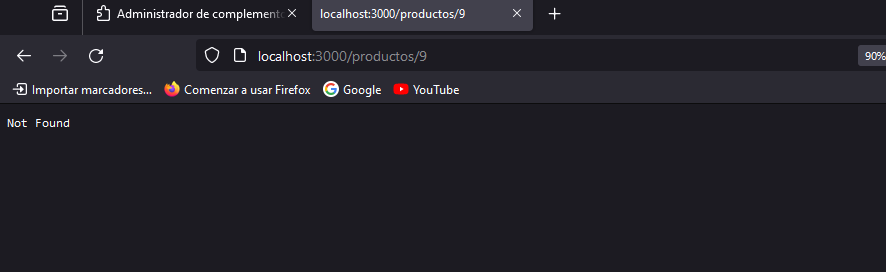
El método POST agrega elementos al arreglo, el programa agarra el ultimo id que fue agregado y el siguiente a agregar será el mismo ya existente más uno, de esta forma el id no se repetirá. Luego en el JSON que será enviado mediante POST tendrá los datos que querrán ser añadidos, este caso lo hacemos desde Postman. Para esto usamos la estructura en donde indicamos que usaremos POST y le proporcionamos la URL, por último, agregamos la información siguiendo la estructura que se visualiza en la siguiente imagen:



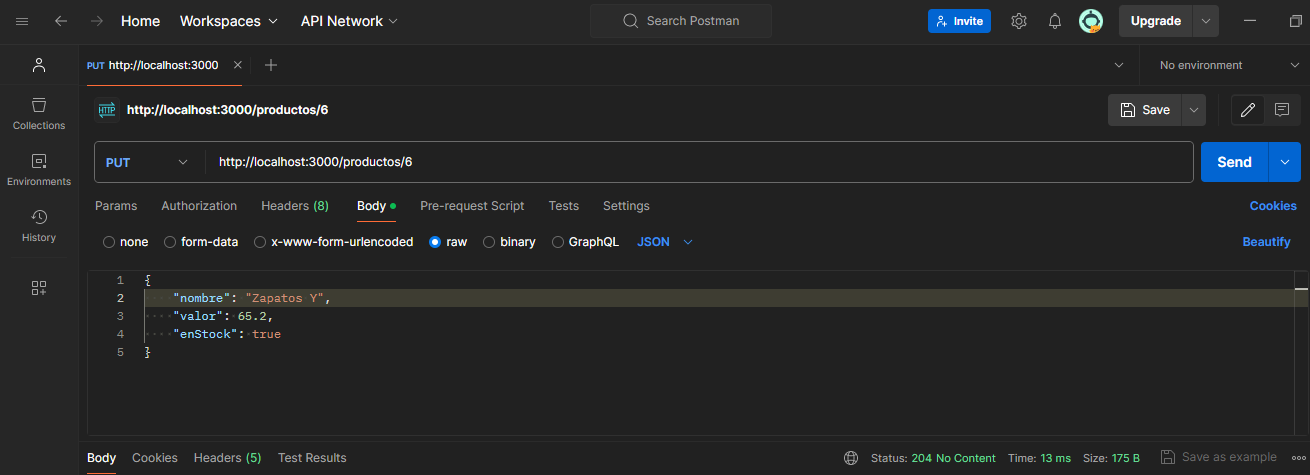
Así se visualiza la información agregada desde el navegador:



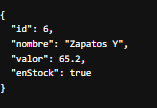
***Método PUT:***

El método de PUT es para actualizar datos de un elemento ya existente por lo cual se deberá indicar en la dirección cual es el id del elemento, si el elemento no existe mandara un código de error 404:

Método PUT desde Postman:

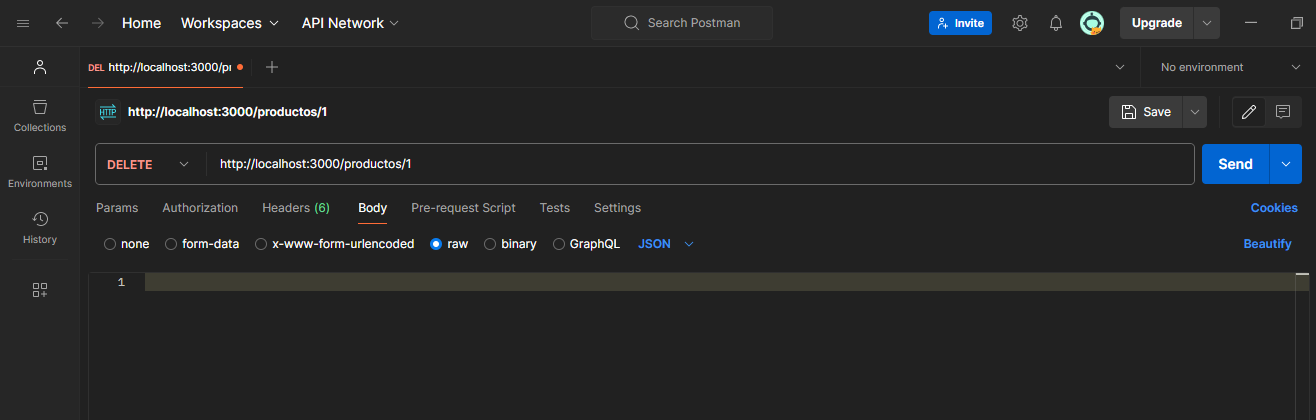


Así se visualiza la actualización de la información desde el navegador:

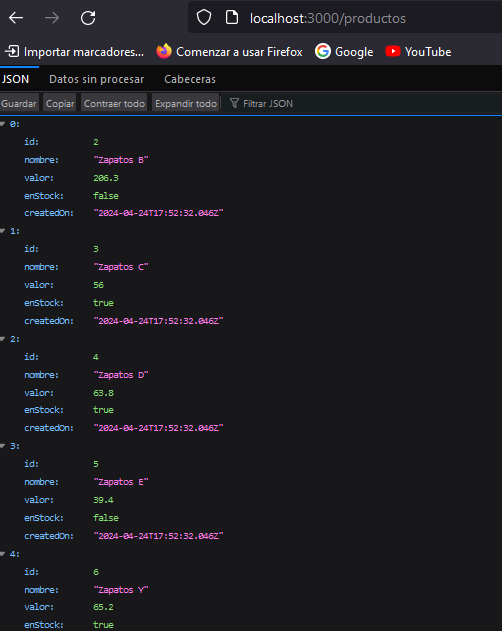


***Método DELETE:***

Por último, el método DELETE eliminará un registro mandando un JSON sin información para que este registro “no exista”. Desde Postaman añadimos a la URL, el ID del producto que queremos eliminar, en la imagen, se añade el 1, por tanto ese es el registro que se elimina:



Así se visualiza el DELETE desde un navegador:



**Conclusiones:**

Usar Postman ofrece una serie de ventajas significativas para los desarrolladores y equipos de desarrollo, ya que esta herramienta facilita enormemente el proceso de probar, depurar y colaborar en el desarrollo de API y servicios web. Con Postman, los equipos pueden crear y ejecutar fácilmente solicitudes HTTP, realizar pruebas automatizadas, documentar API de manera efectiva y colaborar de manera eficiente, su interfaz intuitiva y su amplia gama de funcionalidades, como la gestión de entornos, pruebas de regresión y monitoreo, hacen que sea una herramienta integral para cualquier proyecto de desarrollo de software.

**Repositorio de GitHub:**

<https://github.com/ItsFranco666/Api-con-NodeJS-y-POSTMAN.git>