

JS



PROFE VIDERMID

Programación en Java Script

(Parte 11)

Facilitador: Ing. Esp. Vidermid Sánchez



@vidermid



+584147106623



@ingenieriadigitalsc



+584147464801



API - REST Y BACK-END

El modulo JSON Server de NodeJS permite crear una REST API con respuestas http desde un Back-End en cuestión de minutos para ser implementada en proyectos Fron – End sin preocuparnos por programar en base de datos.

<https://jsonplaceholder.typicode.com/>

Sitio de prueba

<https://www.npmjs.com/package/json-server>

Paquete npm para instalar

<https://github.com/typicode/json-server>

Repositorio oficial

- 1) Crear una carpeta de trabajo.
- 2) Abrir una consola dentro de VSCode en la carpeta de trabajo. (terminal integrated)
- 3) Crear el archivo package.json integrado al proyecto. (npm init -y)
- 4) Instalar jsonServer. (npm i json-server)
- 5) Crear el archivo db.json. (para simular la base de datos sin necesidad de métodos)



```
{
  "productos":[
    {
      "id":1,
      "descripcion":"Papa Criolla",
      "precio":1850.45,
      "tipoId":1
    },
    {
      "id":2,
      "descripcion":"Apio",
      "precio":2000,
      "tipoId":1
    },
    {
      "id":3,
      "descripcion":"Remolacha",
      "precio":1599.99,
      "tipoId":1
    },
    {
      "id":4,
      "descripcion":"Platano Maduro",
      "precio":3456.99,
      "tipoId":1
    },
    {
      "id":5,
      "descripcion":"Melón",
      "precio":1978.34,
      "tipoId":2
    },
    {
      "id":6,
      "descripcion":"Patilla",
      "precio":2000,
      "tipoId":2
    },
    {
      "id":7,
      "descripcion":"Ajo",
      "precio":890,
      "tipoId":3
    },
    {
      "id":8,
      "descripcion":"Pimienta",
      "precio":999.76,
      "tipoId":3
    }
  ],
}
```

```
"tipos":[
  {
    "id":1,
    "descripcion":"Verduras",
    "temporada":"de enero a diciembre"
  },
  {
    "id":2,
    "descripcion":" Frutas",
    "temporada":"por temporadas especificas"
  },
  {
    "id":3,
    "descripcion":"Condimentos",
    "temporada":"de enero a diciembre"
  }
]
```

Para que el servidor se reinicie y aplique los cambios de la data que se van a realizar en vivo, vamos a modificar el valor y titulo de **test**: en el archivo package.json por lo siguiente :

```
{
  "name": "jsonserver_practica",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "server": "json-server --watch db.json"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

Luego para disparar el server debemos ejecutar la siguiente línea de comandos:

```
npm run server
```

Me devuelve los *end points* que se pueden utilizar, o las url a consultar:

Resources

<http://localhost:3000/productos>

<http://localhost:3000/tipos>

Home

<http://localhost:3000>



JSON Server



JSON Server

♥ GitHub Sponsors

💧 My JSON Server

😊 Supporters

Congrats!

You're successfully running JSON Server
💎.۹(◡`*)۹💎.

Resources

[/productos](#) ^{8x}

[/tipos](#) ^{3x}

To access and modify resources, you can use any HTTP method:

[GET](#) [POST](#) [PUT](#) [PATCH](#) [DELETE](#) [OPTIONS](#)

undefined



Existen varios métodos para realizar peticiones mediante el protocolo HTTP:

GET para solicitar

POST para crear

PUT para actualizar

PATCH para actualizar parcialmente

DELETE para eliminar

OPTIONS para obtener información del envío de datos

Ejemplos de consultas desde la API:

<http://localhost:3000/productos> todos los registros de la tabla productos

<http://localhost:3000/tipos> todos los registros de la tabla tipos

<http://localhost:3000/db> toda la base de datos

<http://localhost:3000/tipos/2> solo tipo con código 2

<http://localhost:3000/tipos/3/productos> solo los productos con tipo 3

http://localhost:3000/productos?_limit=2 solo devuelve 2 registros (_limit)

http://localhost:3000/productos?_page=3&_limit=2 solo dos registros de la pagina tres (_page)

http://localhost:3000/productos?_sort=descripcion ordenado por el campo especifico (_sort)

http://localhost:3000/productos?_sort=descripcion&_order=desc igual al anterior pero descendente (_desc)

JS

Cliente REST

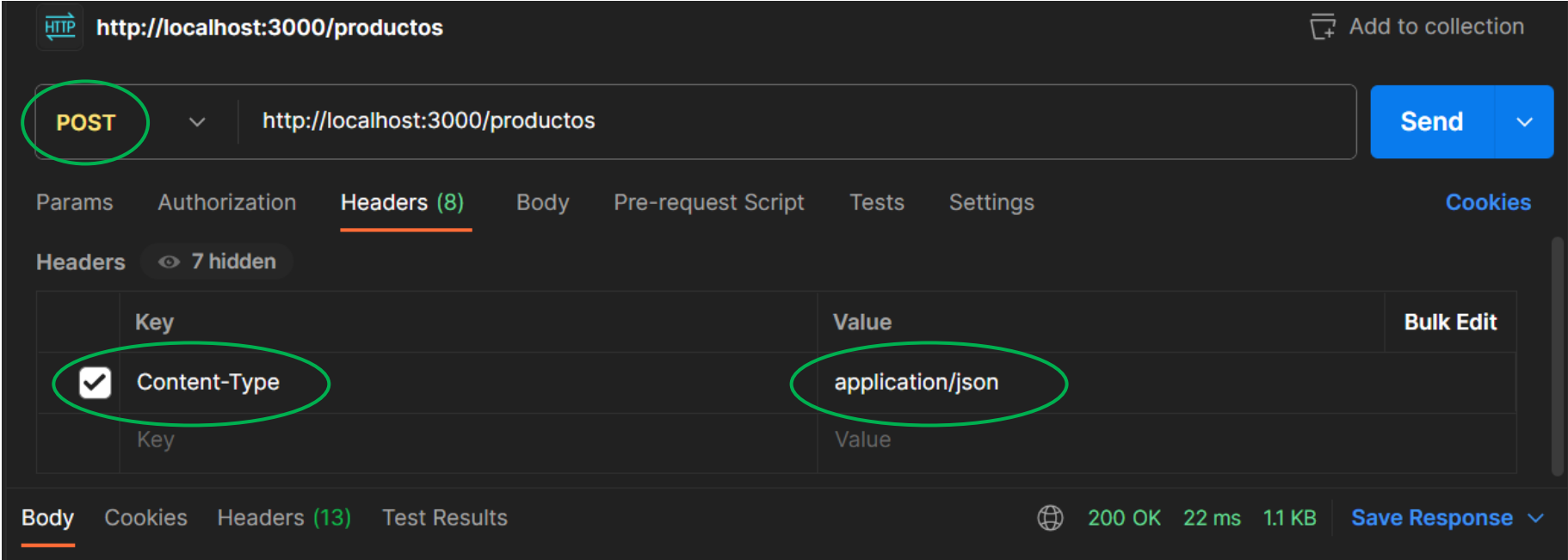
Postman: https://www.postman.com/downloads/?utm_source=postman-home

`http://localhost:3000/productos`

The screenshot displays the Postman application interface. At the top, there's a navigation bar with 'Home', 'Workspaces', and 'Explore' tabs, along with a search bar and 'Sign In'/'Create Account' buttons. The main workspace shows a 'History' panel on the left with a single entry: 'GET http://localhost:3000/productos'. The central panel is configured for a GET request to 'http://localhost:3000/productos'. Below the URL bar, tabs for 'Params', 'Authorization', 'Headers (6)', 'Body', 'Pre-request Script', 'Tests', and 'Settings' are visible. The 'Query Params' section is currently empty. The bottom panel shows the 'Body' tab with a 'Pretty' view of the response. The response status is '200 OK' with a response time of '62 ms' and a size of '1.1 KB'. The JSON response is as follows:

```
1 {
2   "id": 1,
3   "descripcion": "Papa Criolla",
4   "precio": 1850.45,
5   "tipoId": 1
6 },
7 {
8   ...
```

Agregar datos con método POST:



The screenshot shows the REST Client interface for a POST request to `http://localhost:3000/productos`. The **POST** method is selected and circled in green. The **Headers** tab is active, showing a table with headers. The **Content-Type** header is checked and circled in green, with its value `application/json` also circled in green. The status bar at the bottom indicates a successful response: `200 OK`, `22 ms`, and `1.1 KB`.

HTTP `http://localhost:3000/productos` Add to collection

POST `http://localhost:3000/productos` Send

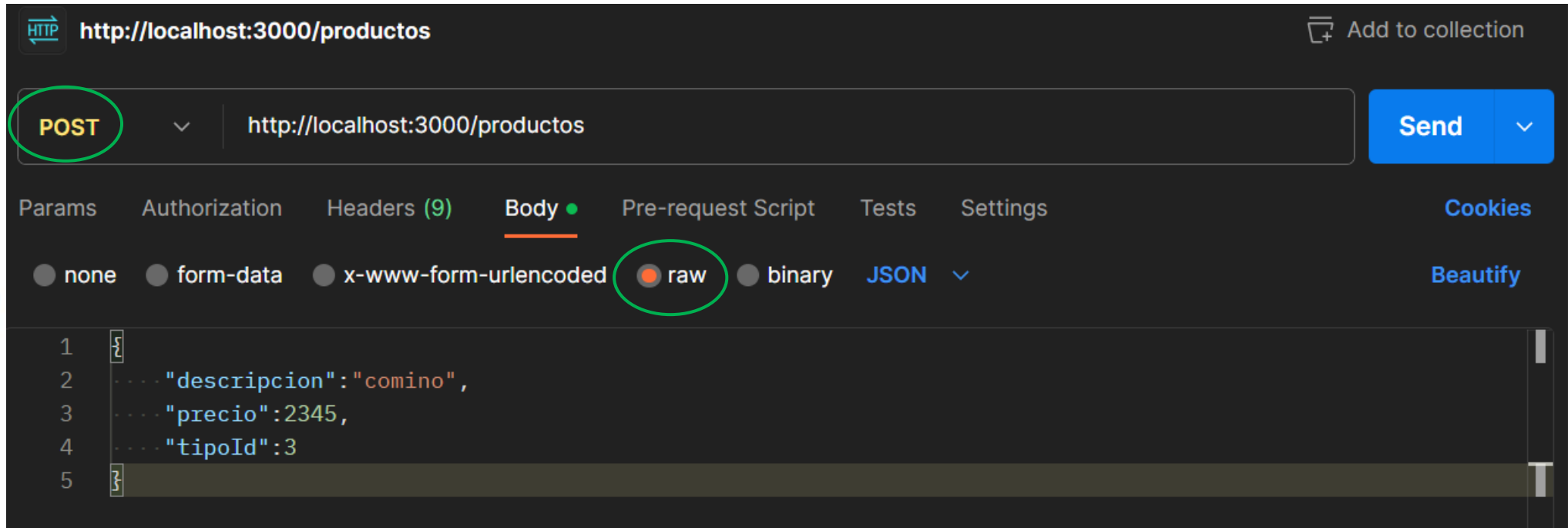
Params Authorization **Headers (8)** Body Pre-request Script Tests Settings Cookies

Headers 7 hidden

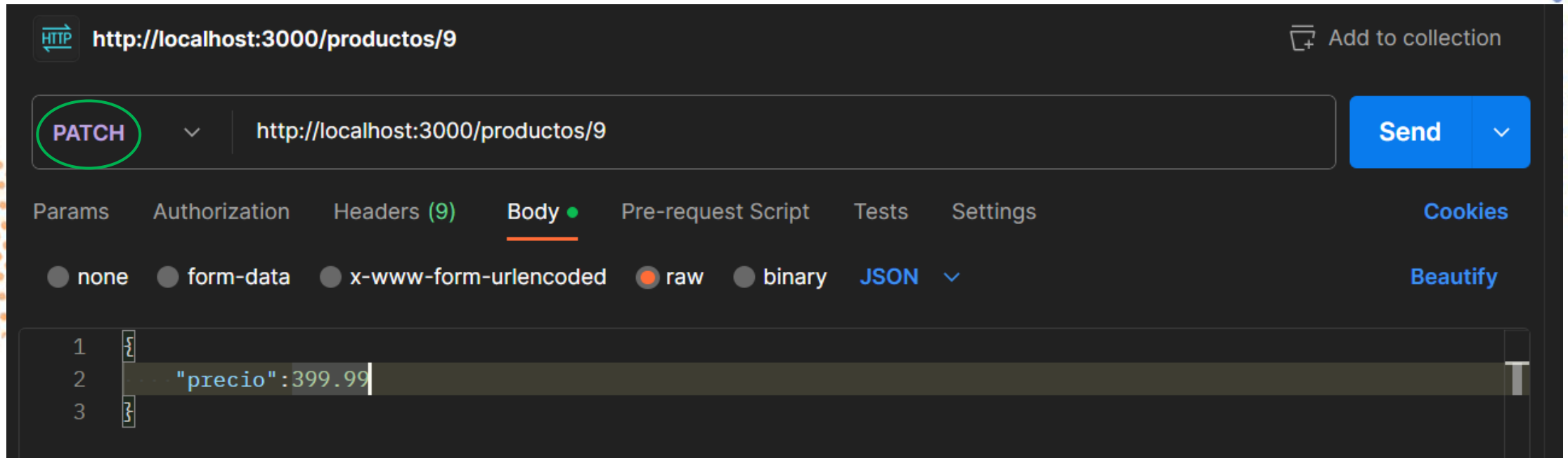
	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	Content-Type	application/json	
	Key	Value	

Body Cookies Headers (13) Test Results 200 OK 22 ms 1.1 KB Save Response

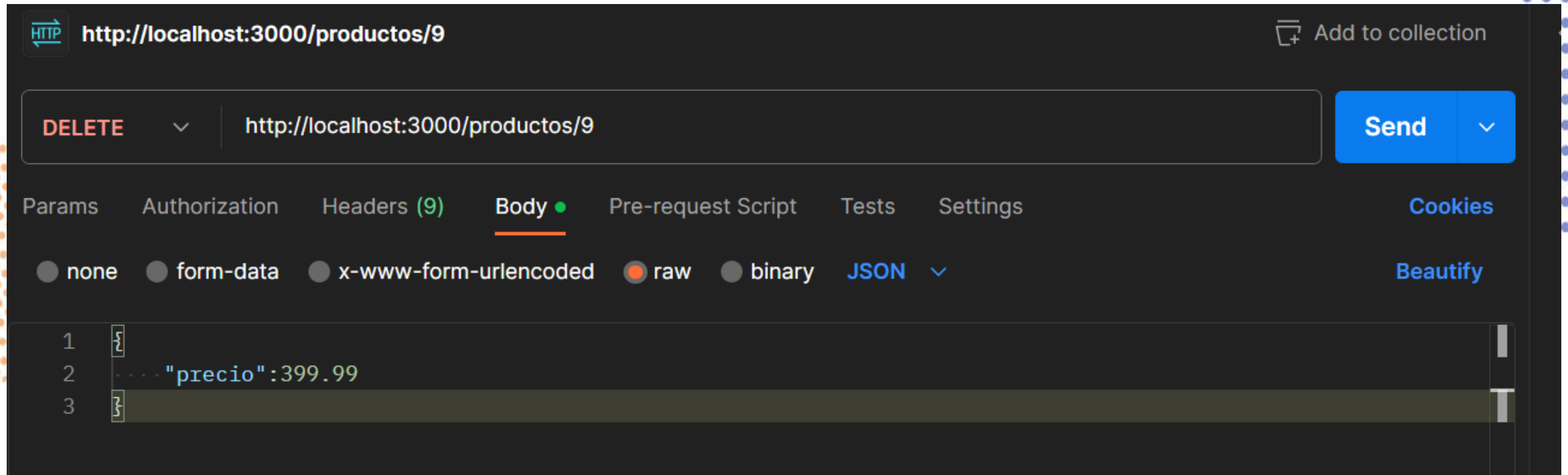
Agregar datos con método POST:



Actualizar datos con método PATCH:



Borrar datos con método POST:



The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:3000/productos/9`
- Method:** `DELETE`
- Body:** `{
 "precio": 399.99
}`
- Format:** `JSON`
- Buttons:** `Send`, `Beautify`, `Add to collection`
- Tabs:** `Params`, `Authorization`, `Headers (9)`, `Body` (selected), `Pre-request Script`, `Tests`, `Settings`, `Cookies`

Consulta avanzadas:

http://localhost:3000/productos?precio_gte=1000

http://localhost:3000/productos?precio_lte=1000

http://localhost:3000/productos?precio_lte=1600&precio_gte=1000

<http://localhost:3000/productos?q=ajo>