

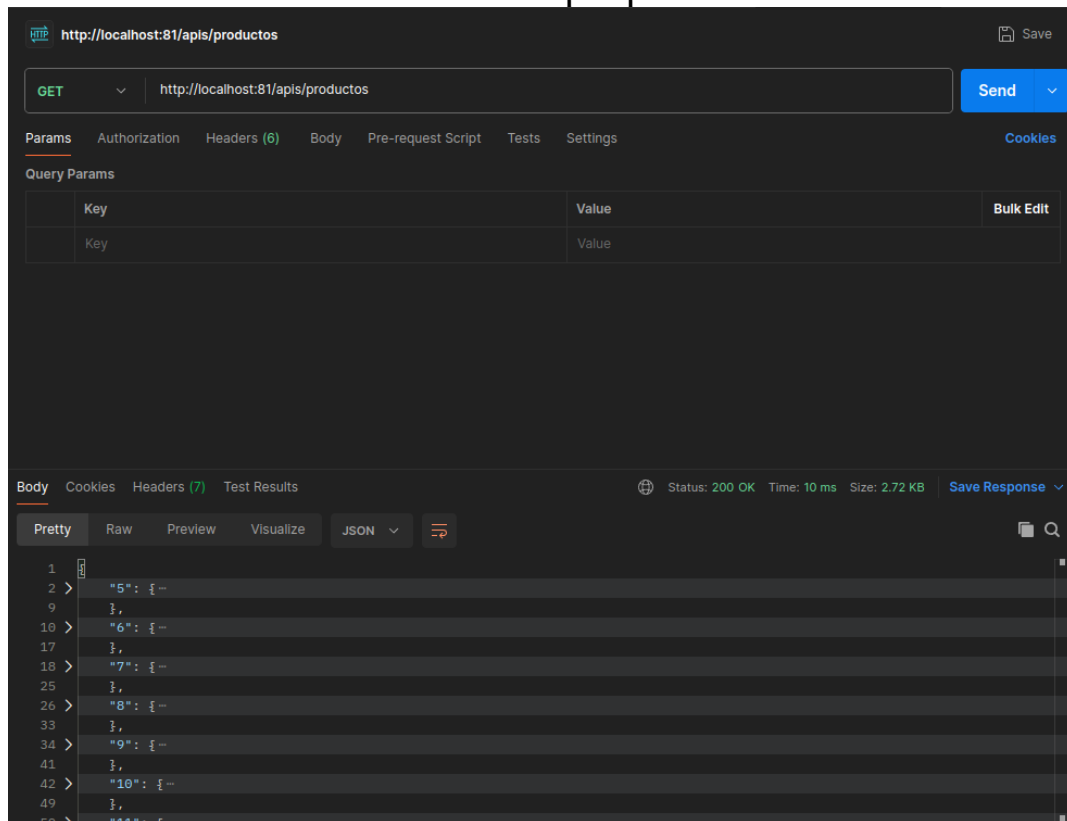
# **Documentación API REST Productos**

**Hecho por: José Carlos Pérez González**

## Método GET

Devuelve todos los productos o el producto con el id especificado en un objeto con formato JSON

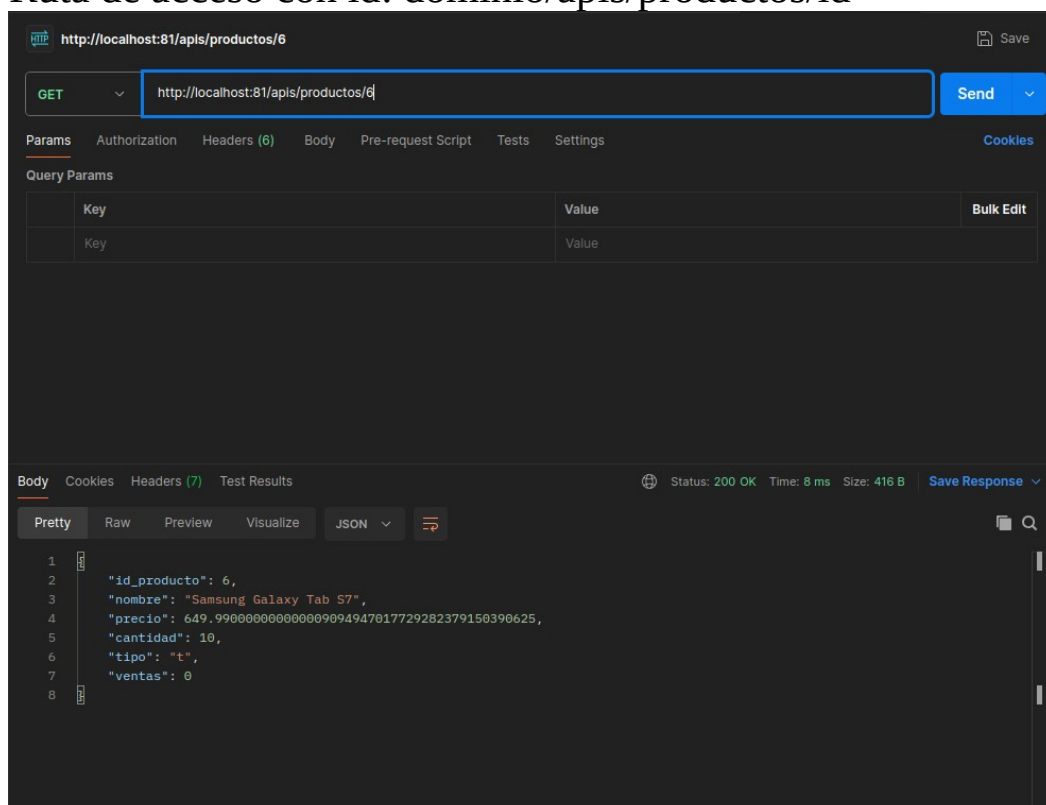
Ruta de acceso sin id: dominio/apis/productos



The screenshot shows a REST client interface with the URL `http://localhost:81/apis/productos` and the method `GET`. The `Send` button is visible. Below the URL bar, there are tabs for `Params`, `Authorization`, `Headers (6)`, `Body`, `Pre-request Script`, `Tests`, and `Settings`. The `Query Params` section is empty. The `Body` tab is selected, showing the response in `JSON` format. The response is a JSON array of 10 product objects, each with an `id`, `nombre`, `precio`, `cantidad`, `tipo`, and `ventas` property.

```
1 {
2   "id": 5,
3   "nombre": "Samsung Galaxy Tab S7",
4   "precio": 649.99,
5   "cantidad": 10,
6   "tipo": "t",
7   "ventas": 0
8 }
```

Ruta de acceso con id: dominio/apis/productos/id



The screenshot shows a REST client interface with the URL `http://localhost:81/apis/productos/6` and the method `GET`. The `Send` button is visible. Below the URL bar, there are tabs for `Params`, `Authorization`, `Headers (6)`, `Body`, `Pre-request Script`, `Tests`, and `Settings`. The `Query Params` section is empty. The `Body` tab is selected, showing the response in `JSON` format. The response is a JSON object representing a single product with an `id`, `nombre`, `precio`, `cantidad`, `tipo`, and `ventas` property.

```
1 {
2   "id_producto": 6,
3   "nombre": "Samsung Galaxy Tab S7",
4   "precio": 649.99,
5   "cantidad": 10,
6   "tipo": "t",
7   "ventas": 0
8 }
```

## Método POST

Añade un producto a la base de datos, siendo necesario pasarle los parámetros:

- Nombre
- Precio (minimo 1)
- Cantidad (minimo 1)
- Ventas (minimo 0)
- Tipo (m, t)

Ruta de acceso: dominio/apis/productos/

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

- URL:** `http://localhost:81/apis/productos/`
- Method:** `POST`
- Body Type:** `x-www-form-urlencoded`
- Body Data:**

Key	Value
<input checked="" type="checkbox"/> nombre	PruebaDocumentacion
<input checked="" type="checkbox"/> precio	10
<input checked="" type="checkbox"/> tipo	m
<input checked="" type="checkbox"/> cantidad	20
<input checked="" type="checkbox"/> ventas	1

**Response:** Status: 201 Created, Time: 21 ms, Size: 439 B

```
1  {
2    "mensaje": "Se ha insertado el producto correctamente",
3    "59": [
4      {
5        "id_producto": 59,
6        "nombre": "PruebaDocumentacion",
7        "precio": 10,
8        "cantidad": 20,
9        "tipo": "m",
10       "ventas": 1
11      }
12    ]
13  }
```

## Método PUT

Actualiza los valores del producto pasado por la misma URL y siendo necesario especificar cada características del producto, siendo estas las mismas que en el método POST y con las mismas condiciones.

Ruta de acceso: dominio/apis/productos/id

PUT <http://localhost:81/apis/productos/59> Send

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

none form-data **x-www-form-urlencoded** raw binary

Key	Value
nombre	EstoAhoraEsUnPut
precio	10
tipo	m
cantidad	20
ventas	1

Body Cookies Headers (7) Test Results Status: 200 OK Time: 19 ms Size: 306 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "mensaje": "Producto Modificado"
3 }
```

GET del producto modificado:

GET <http://localhost:81/apis/productos/59> Send

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

none form-data x-www-form-urlencoded raw binary

Key	Value
nombre	EstoAhoraEsUnPut
precio	10
tipo	m
cantidad	20
ventas	1

Body Cookies Headers (7) Test Results Status: 200 OK Time: 10 ms Size: 367 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id_producto": 59,
3   "nombre": "EstoAhoraEsUnPut",
4   "precio": 10,
5   "cantidad": 20,
6   "tipo": "m",
7   "ventas": 1
8 }
```

## Método DELETE

Borra el producto de la base de datos con el id que coincida con el que se le pasa a la URL

Ruta de acceso: dominio/apis/productos/id

The screenshot shows a REST client interface with the URL `http://localhost:81/apis/productos/` and the method **DELETE**. The `Send` button is highlighted. Below the URL bar, the **Params** tab is active, showing a table with one row: 

Key	Value
Key	Value

. The **Body** tab is also visible. At the bottom, the **Body** tab is selected, showing the response in **Pretty** format: 

```
1 {
2   "mensaje": "Producto eliminado"
3 }
```

. The status bar at the bottom indicates **Status: 200 OK**, **Time: 19 ms**, and **Size: 305 B**.

GET del producto eliminado con el id especificado:

The screenshot shows a REST client interface with the URL `http://localhost:81/apis/productos/` and the method **GET**. The `Send` button is highlighted. Below the URL bar, the **Params** tab is active, showing a table with one row: 

Key	Value
Key	Value

. The **Body** tab is also visible. At the bottom, the **Body** tab is selected, showing the response in **Pretty** format: 

```
1
```

. The status bar at the bottom indicates **Status: 404 Not Found**, **Time: 15 ms**, and **Size: 279 B**.

**DATO: Si se introduce un metodo que sea distinto a uno de los anteriores o distinto a OPTIONS da error 405 Not Allowed**