

DOCUMENTACIÓN DE API RESTFUL - SERVIDOR DE PRODUCTOS

En esta documentación se detalla los endpoints RESTful para el servicio de gestión de productos

1. LISTAR PRODUCTOS

Devuelve una lista de todos los productos que se encuentran en la base de datos.

- **Método HTTP:** GET
 - **URL:** http://localhost:8080/product
 - **Argumentos de Entrada:** Ninguno.
 - **Argumentos de Salida (Respuesta Exitosa):**
 - **Código HTTP:** 200 (OK)
 - **Ejemplos:**

Ejemplo 1: Extraído desde Postman

GET Send

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

Query Params

Key	Value
Key	Value

Bulk Edit

```
 1   [
 2     {
 3       "id": 1,
 4       "name": "Chais",
 5       "unit": "18 boxes x 20 bags",
 6       "price": 59.99,
 7       "supplier": {
 8         "id": 1,
 9         "name": "Exotic Liquid"
10       },
11       "category": {
12         "id": 1,
13         "name": "Beverages"
14       }
15     },
16     {
17       "id": 2,
18       "name": "Chang",
19       "unit": "12 oz bottles",
20       "price": 19,
21       "supplier": {
22         "id": 2,
23         "name": "Kang's"
24       }
25     }
26   ]
27 }
```

Ejemplo 2: Extraído desde un cliente web

Buscar Producto por ID					
Lista Completa de Productos					
ID	Nombre del Producto	Categoría	Proveedor	Precio	Unidad
1	Chais	Beverages	Exotic Liquid	18	10 boxes x 20 bags
2	Chang	Beverages	Exotic Liquid	19	24 - 12 oz bottles
3	Aniseed Syrup	Condiments	Exotic Liquid	10	12 - 550 ml bottles
4	Chef Anton's Cajun Seasoning	Condiments	New Orleans Cajun Delights	22	48 - 6 oz jars
5	Chef Anton's Gumbo Mix	Condiments	New Orleans Cajun Delights	21	36 boxes
6	Grandma's Boysenberry Spread	Condiments	Grandma Kelly's Homestead	25	12 - 8 oz jars
7	Uncle Bob's Organic Dried Pears	Produce	Grandma Kelly's Homestead	30	12 - 1 lb pkgs.
8	Northwoods Cranberry Sauce	Condiments	Grandma Kelly's Homestead	40	12 - 12 oz jars
9	Mishi Kobe Niku	Meat/Poultry	Tokyo Traders	97	18 - 500 g pkgs.
10	Iura	Seafood	Tokyo Traders	31	12 - 200 ml jars
11	Queso Cabrales	Dairy Products	Cooperativa de Quesos 'Las Cabras'	21	1 kg pkgs.
12	Queso Manchego La Pastora	Dairy Products	Cooperativa de Quesos 'Las Cabras'	38	10 - 500 g pkgs.
13	Konbu	Seafood	Mayumi's	6	2 kg box
14	Tofu	Produce	Mayumi's	23	40 - 100 g pkgs.
15	Genen Shouyu	Condiments	Mayumi's	16	24 - 250 ml bottles
16	Pavlova	Confection	Pavlova, Ltd.	17	32 - 500 g boxes
17	Alice Mutton	Meat/Poultry	Pavlova, Ltd.	39	20 - 1 kg tins
18	Francesinha Tuna	Foodstuff	Foodstuff, Ltd.	62	140 km miles

2. OBTENER UN PRODUCTO POR ID

Busca y devuelve un producto específico por su ID.

- **Método HTTP:** GET
- **URL:** http://localhost:8080/product/{id}
- **Argumentos de Entrada:**
 - **id (parámetro de ruta):** El ID del producto a buscar.
- **Argumentos de Salida (Respuesta Exitosa):**
 - **Código HTTP:** 200 (OK)
 - **Ejemplos**

Ejemplo 1: Extraído desde Postman

The screenshot shows the Postman interface with a successful API call. The URL is set to `localhost:8080/product/10`. The response status is 200 OK, time 1502 ms, size 426 B. The response body is a JSON object:

```
1  {
2     "estado": "success",
3     "producto": [
4         {
5             "id": 10,
6             "name": "Ikura",
7             "unit": "12 - 200 ml jars",
8             "price": 31,
9             "supplice": {
10                 "id": 4,
11                 "name": "Tokyo Traders"
12             },
13             "category": {
14                 "id": 6,
15                 "name": "Seafood"
16             }
17         }
18     ]
19 }
```

Ejemplo 2: Extraído desde un cliente web

The screenshot shows a web-based application for searching products by ID. The search bar contains the value `10`, and the search button is labeled `Buscar`. The results section is titled `Producto Encontrado` and displays the following information for product `Ikura`:

ID: 10
Nombre del Producto: Ikura
Categoría: Seafood
Proveedor: Tokyo Traders
Precio: 31
Unidad: 12 - 200 ml jars

- **Argumentos de Salida (Respuesta con error):**
 - **Código HTTP:** 400 (Bad request)
 - **Descripción:** Cuando el producto no existe
 - **Ejemplos:**

Ejemplo 1: Extraído desde Postman

The screenshot shows the Postman interface with a GET request to `localhost:8080/product/100`. The response body is a JSON object with one key-value pair: `"msg": "producto no encontrado"`.

Ejemplo 2: Extraído desde un cliente web

Buscar Producto por ID

The screenshot shows a search form with the placeholder text `Buscar Producto por ID`. The input field contains the value `100`, and the button is labeled `Buscar`. Below the form, a red box highlights the error message `producto no encontrado`.

- **Argumentos de Salida (Respuesta con error):**
 - **Código HTTP:** 400 (Bad request)
 - **Descripción:** Cuando el ID no es un valor entero
 - **Ejemplos:**

Ejemplo 1: Extraído desde Postman

The screenshot shows the Postman interface with a GET request to `localhost:8080/product/hola`. The response body is a JSON object with one key-value pair: `"msg": "el ID debe ser un valor entero"`.

Ejemplo 2: Extraído desde un cliente web

Buscar Producto por ID

holo

Buscar

El ID debe ser un número entero

3. CREAR UN PRODUCTO

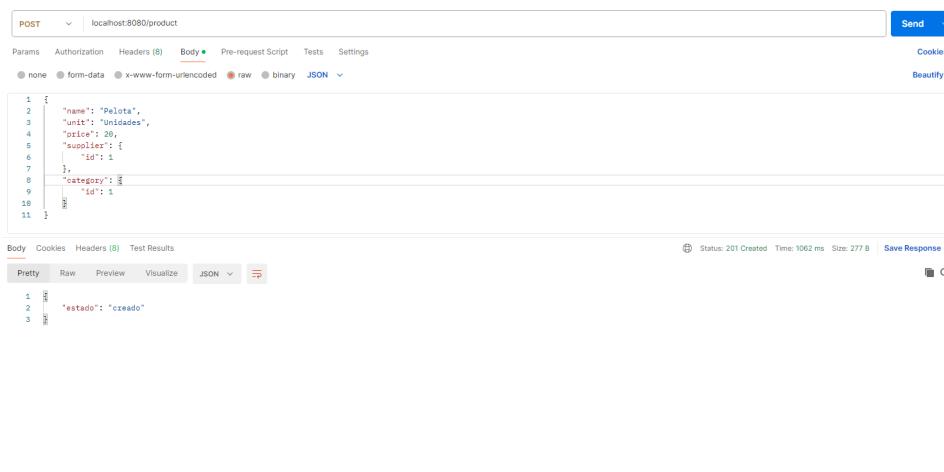
Guarda un nuevo producto en la base de datos.

- **Método HTTP:** POST
- **URL:** http://localhost:8080/product
- **Argumentos de Entrada:**
 - En formato JSON:



```
1 {
2   "name": "Pelota",
3   "unit": "Unidades",
4   "price": 20,
5   "supplier": {
6     "id": 1
7   },
8   "category": {
9     "id": 1
10 }
11 }
```

- **Argumentos de Salida (Respuesta Exitosa):**
 - **Código HTTP:** 201 (Created)
 - **Ejemplo:** (extraído desde Postman)



POST localhost:8080/product

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

Body (Pretty) [Raw] [Preview] [Visualize] [JSON]

```
1 {
2   "name": "Pelota",
3   "unit": "Unidades",
4   "price": 20,
5   "supplier": {
6     "id": 1
7   },
8   "category": {
9     "id": 1
10 }
11 }
```

Status: 201 Created Time: 1062 ms Size: 277 B Save Response

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "msg": "creado",
3   "estado": "creado"
}
```

- **Argumentos de Salida (Respuesta con error):**
 - **Código HTTP:** 400 (Bad request)
 - **Descripción:** No se ha enviado ningún parámetro para crear el producto.
 - **Ejemplo:** (extraído desde Postman)

POST | localhost:8080/product

Send | ↴

Params Authorization Headers (7) Body Pre-request Script Tests Settings

None form-data x-www-form-urlencoded raw binary JSON

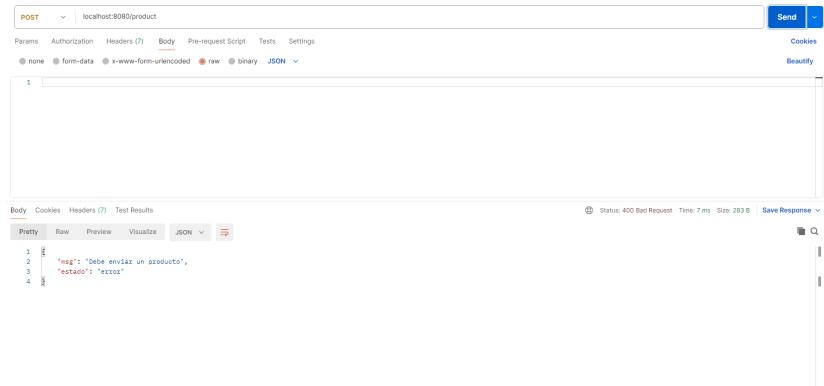
1

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

1 %
2 "msg": "Debe enviar un producto",
3 "estado": "errores"
4 %

Status: 400 Bad Request Time: 7 ms Size: 283 B Save Response



4. ACTUALIZAR PRODUCTO

Actualiza un producto existente. Esta implementación realiza una actualización parcial, pues solo actualiza los campos donde no son nulos.

- **Método HTTP:** PUT
- **URL:** http://localhost:8080/product
- **Argumentos de Entrada:**
 - En formato JSON: Es necesario colocar el id del producto a actualizar.

```
PUT /product
Content-Type: application/json
Accept: application/json

{
  "id": 88,
  "unit": "te cambie"
}
```

- **Argumentos de Salida (Respuesta Exitosa):**
 - **Código HTTP:** 200 (OK)
 - **Ejemplo:** (extraído desde Postman)

The screenshot shows the Postman interface for a PUT request to `localhost:8080/product`. The request body is a JSON object with fields `id` and `unit`. The response status is 200 OK, with a response body indicating the product was updated.

```
PUT /product
Content-Type: application/json
Accept: application/json

{
  "id": 88,
  "unit": "te cambie"
}

{
  "id": 88,
  "unit": "te cambie",
  "estado": "actualizado"
}
```

Status: 200 OK Time: 2.36 s Size: 277 B Save Response

- **Argumentos de Salida (Respuesta con error):**
 - **Código HTTP:** 400 (Bad request)
 - **Descripción:** No se ha enviado ningún ID del producto a actualizar.
 - **Ejemplo:** (extraído desde Postman)

PUT | localhost:8080/product

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

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

```
1
2
3
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1
2
3
4
```

Status: 400 Bad Request Time: 4 ms Size: 295 B Save Response

```
1
2
3
4
```

- **Argumentos de Salida (Respuesta con error):**

- **Código HTTP:** 400 (Bad request)
- **Descripción:** No existe el ID del producto a actualizar.
- **Ejemplo: (extraído desde Postman)**

PUT | localhost:8080/product

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

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

```
1
2
3
4
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1
2
3
4
```

Status: 400 Bad Request Time: 1474 ms Size: 282 B Save Response

```
1
2
3
4
```

5. BORRAR UN PRODUCTO

Elimina un producto en base a su ID.

- **Método HTTP:** DELETE
- **URL:** http://localhost:8080/product/{id}
- **Argumentos de Entrada:**
 - **id (parámetro de ruta):** El ID del producto a eliminar.
- **Argumentos de Salida (Respuesta Exitosa):**
 - **Código HTTP:** 200 (OK)
 - **Ejemplo: (extraído desde Postman)**

The screenshot shows a Postman request configuration for a DELETE method to the URL `localhost:8080/product/80`. The 'Body' tab is selected, showing a JSON response with the key `"msg": "eliminado exitosamente"`. The status bar indicates a 200 OK response with a size of 288 B.

- **Argumentos de Salida (Respuesta con error):**
 - **Código HTTP:** 400 (Bad request)
 - **Descripción:** Cuando el producto no existe
 - **Ejemplo: (extraído desde Postman)**

The screenshot shows a Postman request configuration for a DELETE method to the URL `localhost:8080/product/82`. The 'Body' tab is selected, showing a JSON response with the key `"msg": "No se ha encontrado el producto"`. The status bar indicates a 400 Bad Request response with a size of 291 B.

- **Argumentos de Salida (Respuesta con error):**
 - **Código HTTP:** 400 (Bad request)
 - **Descripción:** Cuando el ID del producto no es un valor entero.
 - **Ejemplo:** (extraído desde Postman)

The screenshot shows a Postman request configuration and its response. The request is a **DELETE** to **localhost:8080/product/hola**. The response status is **400 Bad Request**, with a time of **8 ms** and size of **273 B**. The response body is a JSON object: **{msg: "el ID debe ser un valor entero"}**.