

Documentación del API Restful de Productos:

Esta documentación describe los endpoints disponibles en el servicio web para la gestión de productos.

Url base: <http://localhost:8080>

1. Listar Productos:

Este hace obtener una lista de todos los productos disponibles en la base de datos:

- **Método HTTP & URL:** GET /product
- **Argumentos de Entrada:** Ninguno
- **Argumentos de Salida:**
Arreglo de objetos Product en formato JSON.
Ejemplo de respuesta y estado:

```
[  
  {  
    "productId": 1,  
    "productName": "Chai",  
    "supplier": { "supplierId": 1, "companyName": "Exotic  
Liquids", ... },  
    "category": { "categoryId": 1, "categoryName":  
"Beverages", ... },  
    "quantityPerUnit": "10 boxes x 20 bags",  
    "unitPrice": 18.0000,  
    "unitsInStock": 39,  
    ...  
  }, {  
    "productId": 2,  
    "productName": "Chang",  
    ...  
  }  
]
```

- **Códigos de Estado:** 200 OK, donde la solicitud fue exitosa.

2. Obtener un Producto por ID:

Este hace obtener los detalles de un producto específico con su ID:

- **Método HTTP & URL:** GET /product/{id}
- **Argumentos de Entrada:** La id del producto
Ejemplo: /product/12
- **Argumentos de Salida:**
Un arreglo de objeto Product en formato JSON
Ejemplo de respuesta y estado:

```
{
```

```
        "productId": 1,  
        "productName": "Chai",  
        "supplier": { "supplierId": 1, ... },  
        "category": { "categoryId": 1, ... },  
        ...  
    }  
• Argumentos de Salida: En caso haya error aparecerá:
```

```
{  
    "error": " no encontrado con id: 50"  
}
```

- **Códigos de estado:**
200 OK: producto encontrado
400 Bad Request: El producto no existe o no es un numero valido

3. Crear un Producto:

Registra un nuevo producto en la base de datos:

- **Método HTTP & URL:** POST /product
- **Argumentos de Entrada:**
Cuerpo: un objeto JSON con los datos del producto. Las relaciones deben incluirse también.
Ejemplo de Cuerpo:

```
{  
  
    "productName": "Nuevo Producto",  
  
    "supplier": {  
  
        "supplierId": 1  
  
    },  
  
    "category": {  
  
        "categoryId": 2  
  
    },  
  
    "quantityPerUnit": "12 oz",  
  
    "unitPrice": 25.50,  
  
    "unitsInStock": 100,
```

```

    "unitsOnOrder": 0,
    "reorderLevel": 10,
    "discontinued": false
}

```

- **Argumentos de Salida:**

Un objeto JSON indica el éxito y el ID del nuevo producto.

```
{
  "status": "Producto creado",
  "productId": 78
}
```

- **Códigos de estado:**

201 Created : producto creado

400 Bad Request: Al JSON le falta datos o esta malformado

Ejemplo:

The screenshot shows a Postman interface with the following details:

- Request URL:** http://localhost:8080/product
- Method:** POST
- Body (JSON):**

```

1  {
2   "productName": "Producto de Prueba",
3   "supplier": {
4     "supplierId": 1
5   },
6   "category": {
7     "categoryId": 1
8   },
9   "quantityPerUnit": "10 units",
10  "unitPrice": 25.0,
11  "unitsInStock": 50,
12  "unitsOnOrder": 0,
13  "reorderLevel": 5,
14  "discontinued": false
15 }

```
- Response Status:** 201 Created
- Response Body (JSON):**

```

1  {
2   "productId": 78,
3   "status": "Producto creado"
4 }

```

4. Actualizar un Producto:

Modifica los datos de un producto en la base de datos:

- **Método HTTP & URL:** PUT/product
- **Argumentos de Entrada:**

Cuerpo: un objeto JSON con los datos del producto a actualizar. El productId es obligatorio dentro del JSON:

Ejemplo de cuerpo:

```
{  
    "productId": 78,  
    "productName": "Producto Actualizado",  
    "supplier": { "supplierId": 1 },  
    "category": { "categoryId": 2 },  
    "unitPrice": 28.00,  
    "unitsInStock": 90,  
    ...  
}
```

- **Argumentos de Salida:** En caso de existir parece el objeto Product completo, con los datos actualizados:

Ejemplo de respuesta:

```
{  
    "productId": 78,  
    "productName": "Producto Actualizado",  
    "unitPrice": 28.00,  
    "unitsInStock": 90,  
    ...  
}
```

- **Argumentos de Salida:**

- En caso de error aparecerá:

```
{  
    "error": " Se requiere id"  
}
```

- En caso de no encontrarse:

```
{  
    "error": " El producto no está en la BD"  
}
```

- **Códigos de estado:**

200 OK: producto actualizado

400 Bad Request: Si el productId no lo especificas en el JSON o si no existe su id.

Ejemplo:

The screenshot shows a POSTMAN interface with the following details:

- Method:** PUT
- URL:** http://localhost:8080/product
- Body:** Raw JSON (selected)
- JSON Content:**

```

1 {
2     "productId": 78,
3     "productName": "Producto Actualizado- (PUT)",
4     "supplier": {
5         "supplierId": 1
6     },
7     "category": {
8         "categoryId": 1
9     },
10    "quantityPerUnit": "10 units",
11    "unitPrice": 29.99,
12    "unitsInStock": 40,
13    "unitsOnOrder": 0,
14    "reorderLevel": 5,
15    "discontinued": false
16 }

```
- Response Status:** 200 OK
- Response Headers:** 430 ms, 13.95 KB

5. Borrar un Producto:

Borra un producto de la base de datos:

- **Método HTTP & URL:** DELETE / product/{id}
- **Argumentos de Entrada:** Id es el parámetro de ruta.
Ejemplo de cuerpo: /product/12
- **Argumentos de Salida:**
 - En caso exitoso, devuelve un JSON:

```

{
    "status": " Producto eliminado"
}

```

 - En caso de error:

```

{
    "error": "no se puede eliminar"
}

```
- **Códigos de estado:**
 - 200 OK: Producto eliminado
 - 400 Bad Request: El producto con el id no existe o no se puede eliminar.

Ejemplo:

The screenshot shows the Postman application interface. At the top, there is a header bar with the URL `DEL http://localhost:8080/pr`, a plus icon for creating new requests, and a dropdown for environment selection. Below the header, the main interface shows a request card for `http://localhost:8080/product/78`. The method is set to `DELETE`, and the URL is `http://localhost:8080/product/78`. To the right of the URL are buttons for `Save`, `Share`, and `Send`. The `Send` button is highlighted in blue.

The request card includes tabs for `Params`, `Authorization`, `Headers (6)`, `Body`, `Scripts`, and `Settings`. The `Body` tab is selected, showing options: `none` (selected), `form-data`, `x-www-form-urlencoded`, `raw`, `binary`, and `GraphQL`. A note below says, "This request does not have a body".

At the bottom of the interface, there is a preview section. It shows tabs for `Body`, `Cookies`, `Headers (5)`, and `Test Results`. The `Body` tab is selected, displaying a JSON response. The response is shown in three lines:

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

On the right side of the preview area, there is a status bar showing `200 OK`, `37 ms`, `194 B`, and a copy icon. Below the status bar are icons for copy, search, and refresh.