

Información General

API REST para la gestión de una tienda de tecnología desarrollada con Express.js y Supabase.

Base URL:

`http://localhost:3000/rest/v1`

Stack Tecnológico:

- Node.js v18+
- Express 5.1.0
- Supabase (PostgreSQL)
- Bcrypt 6.0.0
- Dotenv 16.6.1

Configuración Inicial

Variables de Entorno

Archivo .env:

`PORT=3000`

`SUPABASE_URL=tu_url_de_supabase`

`SUPABASE_KEY=tu_clave_de_supabase`

Pasos para correr el proyecto

1. Configuración del frontend (Flutter)

- Clonar el repositorio

Desde tu cuenta de GitHub, copia la URL del repositorio y ejecuta el siguiente comando en tu terminal:

```
git clone <URL_DEL_REPO>
```

2. Instalar dependencias

- Accede a la carpeta del proyecto clonado y ejecuta:

```
flutter pub get
```

Esto descargará e instalará todas las dependencias necesarias para el proyecto

3. Ejecutar la aplicación

- Finalmente, ejecuta la aplicación en tu dispositivo o emulador:

```
flutter run
```

2. Configuración del backend (Node.js)

1. Clonar el repositorio

- Al igual que con el frontend, clona el repositorio del backend:

```
git clone <URL_DEL_REPOITORIO_BACKEND>
```

2. Instalar dependencias

- Ingresa a la carpeta del backend y ejecuta:

```
npm install
```

3. Ejecutar el servidor

- Para levantar el servidor, ejecuta:

```
npm start
```

Nota: Para que la aplicación funcione correctamente, el frontend y el backend deben estar corriendo simultáneamente.

Autenticación

POST /auth/register

Descripción: Registra un nuevo usuario en el sistema de autenticación.

The screenshot shows a Postman interface with the following details:

- Method:** POST
- URL:** http://localhost:3000/rest/v1/users
- Body:** JSON (selected tab)
- JSON Content:**

```
1  {
2    "nombre": "Test",
3    "apellido": "test",
4    "correo": "test@example.com",
5    "telefono": "1234567890",
6    "password": "123456"
7 }
```
- Status:** 201 Created
- Size:** 111 Bytes
- Time:** 1.46 s
- Response:** Headers (6), Cookies, Results, Docs
- Response Body:**

```
1  {
2    "message": "Usuario creado",
3    "data": {
4      "error": null,
5      "data": null,
6      "count": null,
7      "status": 201,
8      "statusText": "Created"
9    }
10 }
```

Respuesta:

201: Usuario creado exitosamente

400: Error en los datos enviado

POST /auth/login

Descripción: Permite a un usuario iniciar sesión validando sus credenciales.

The screenshot shows a POST request to `http://localhost:3000/rest/v1/auth/login`. The request body is JSON, containing:

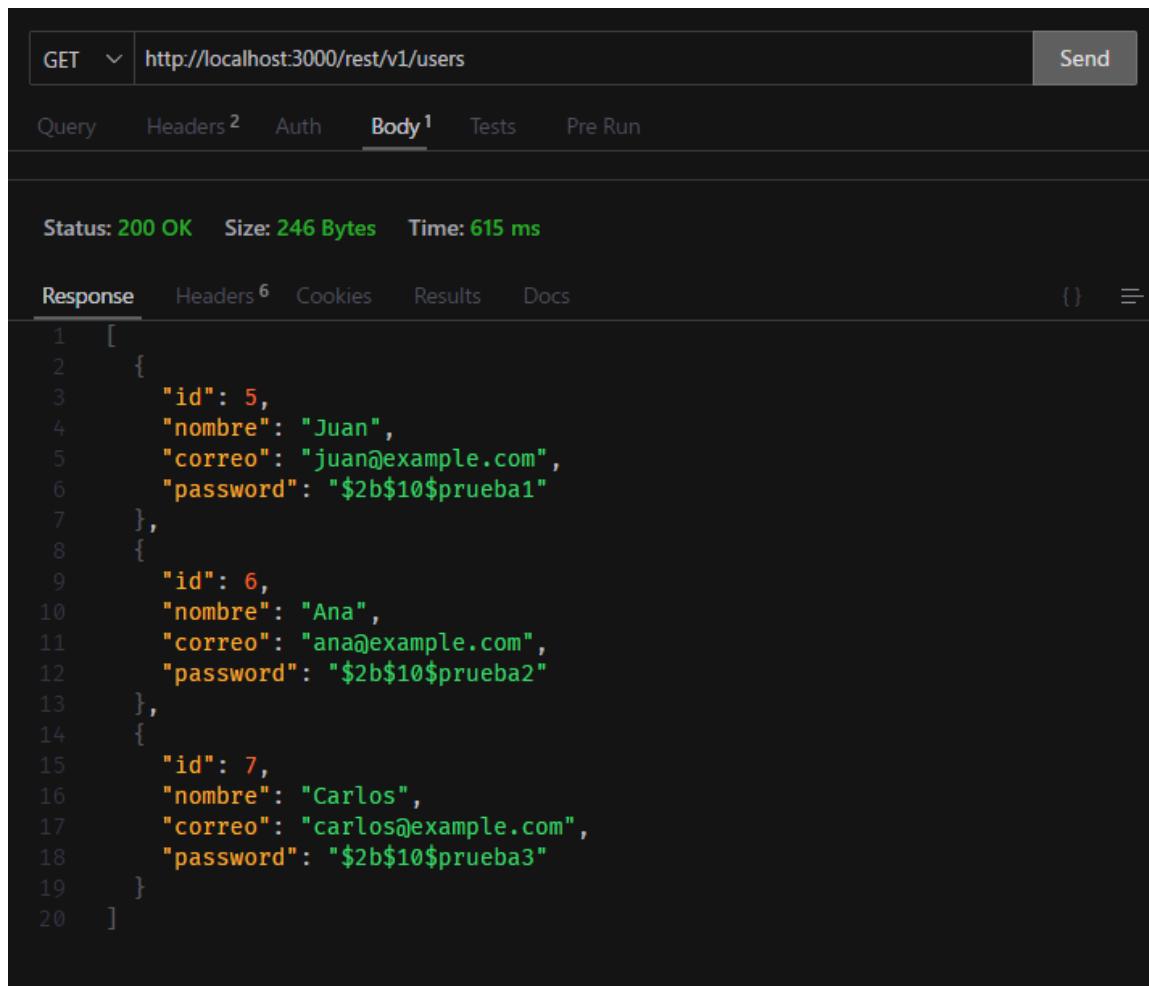
```
1 {  
2   "correo": "test@exampleact.com",  
3   "password": "123456"  
4 }
```

The response status is **200 OK**, size is **140 Bytes**, and time is **1.59 s**. The response body is:

```
1 {  
2   "message": "Inicio de sesión exitoso",  
3   "user": {  
4     "id": 18,  
5     "correo": "test@exampleact.com",  
6     "nombre": "Test Actualizado",  
7     "apellido": "test actual"  
8   }  
9 }
```

GET /auth/users

Descripción: Obtiene la lista de usuarios registrados en el sistema.



The screenshot shows a Postman interface with the following details:

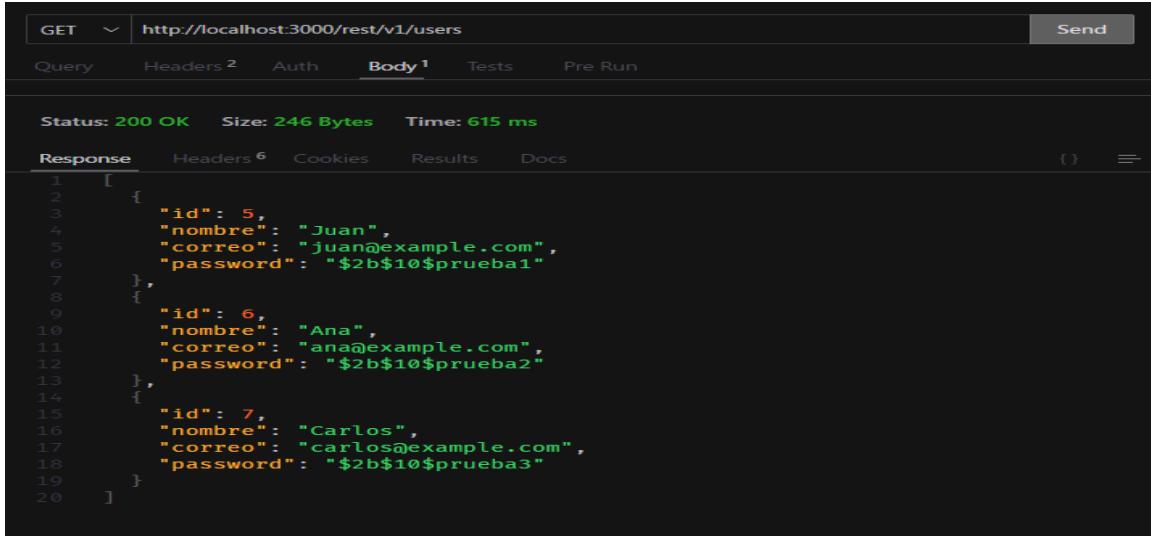
- Method: GET
- URL: http://localhost:3000/rest/v1/users
- Body tab is selected.
- Status: 200 OK
- Size: 246 Bytes
- Time: 615 ms
- Response tab shows the JSON data returned:

```
1 [  
2 {  
3   "id": 5,  
4   "nombre": "Juan",  
5   "correo": "juan@example.com",  
6   "password": "$2b$10$prueba1"  
7 },  
8 {  
9   "id": 6,  
10  "nombre": "Ana",  
11  "correo": "ana@example.com",  
12  "password": "$2b$10$prueba2"  
13 },  
14 {  
15  "id": 7,  
16  "nombre": "Carlos",  
17  "correo": "carlos@example.com",  
18  "password": "$2b$10$prueba3"  
19 }  
20 ]
```

Gestión de Usuarios

GET /users

Descripción: Obtiene la lista de todos los usuarios.



GET <http://localhost:3000/rest/v1/users> Send

Query Headers² Auth Body¹ Tests Pre Run

Status: 200 OK Size: 246 Bytes Time: 615 ms

Response Headers⁶ Cookies Results Docs () ≡

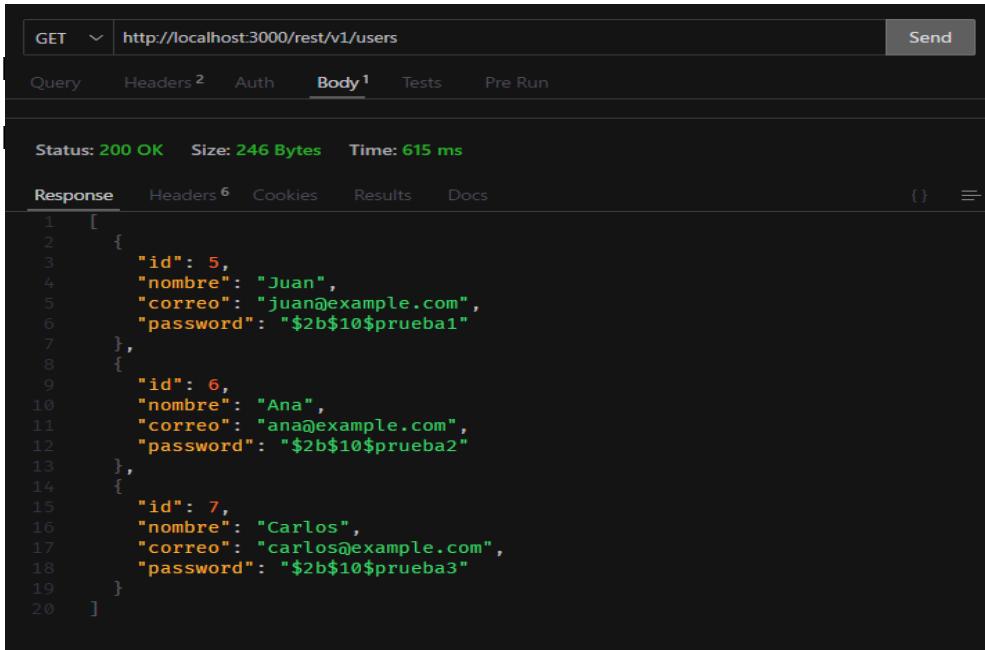
```
1 [  
2 {  
3   "id": 5,  
4   "nombre": "Juan",  
5   "correo": "juan@example.com",  
6   "password": "$2b$10$prueba1"  
7 },  
8 {  
9   "id": 6,  
10  "nombre": "Ana",  
11  "correo": "ana@example.com",  
12  "password": "$2b$10$prueba2"  
13 },  
14 {  
15  "id": 7,  
16  "nombre": "Carlos",  
17  "correo": "carlos@example.com",  
18  "password": "$2b$10$prueba3"  
19 }  
20 ]
```

GET /users/:id

Descripción: Obtiene un usuario por su ID.

Parámetros URL:

- id (uuid)



GET <http://localhost:3000/rest/v1/users> Send

Query Headers² Auth Body¹ Tests Pre Run

Status: 200 OK Size: 246 Bytes Time: 615 ms

Response Headers⁶ Cookies Results Docs () ≡

```
1 [  
2 {  
3   "id": 5,  
4   "nombre": "Juan",  
5   "correo": "juan@example.com",  
6   "password": "$2b$10$prueba1"  
7 },  
8 {  
9   "id": 6,  
10  "nombre": "Ana",  
11  "correo": "ana@example.com",  
12  "password": "$2b$10$prueba2"  
13 },  
14 {  
15  "id": 7,  
16  "nombre": "Carlos",  
17  "correo": "carlos@example.com",  
18  "password": "$2b$10$prueba3"  
19 }  
20 ]
```

POST <http://localhost:3000/rest/v1/users>

Status: 201 Created Size: 111 Bytes Time: 1.42 s

Response	Headers 6	Cookies	Results	Docs
<pre> 1 { 2 "message": "Usuario creado", 3 "data": { 4 "error": null, 5 "data": null, 6 "count": null, 7 "status": 201, 8 "statusText": "Created" 9 } 10 </pre>				

PUT /users/:id

Descripción: Actualiza un usuario existente.

Parámetros URL:

- id (uuid)

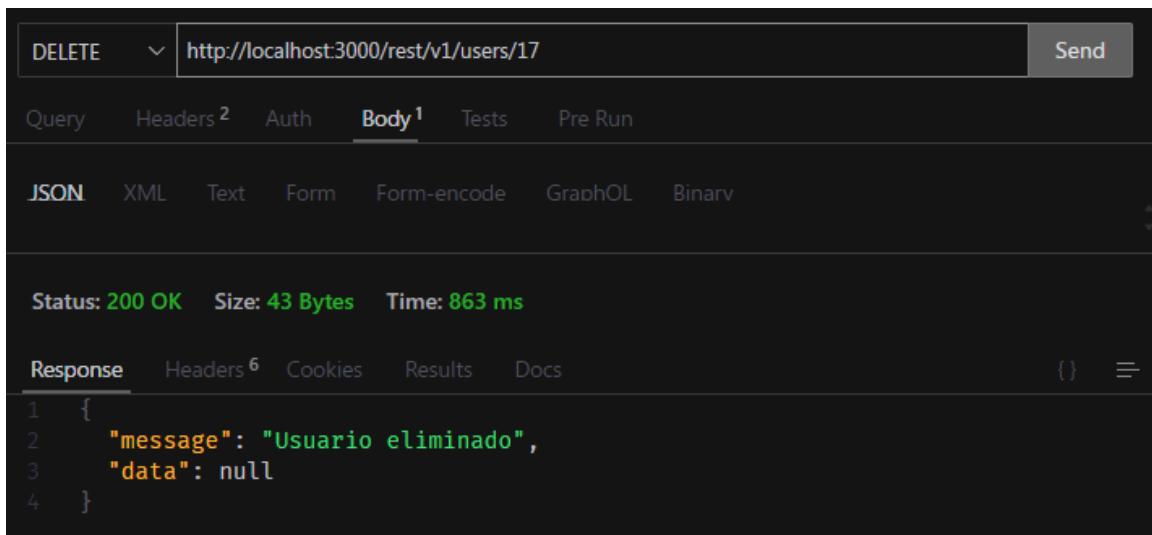
PUT <http://localhost:3000/rest/v1/users/17>

Status: 200 OK Size: 45 Bytes Time: 534 ms

Response	Headers 6	Cookies	Results	Docs
<pre> 1 { 2 "message": "Usuario actualizado", 3 "data": null 4 } </pre>				

DELETE /users/:id

Elimina un usuario del sistema según su ID.



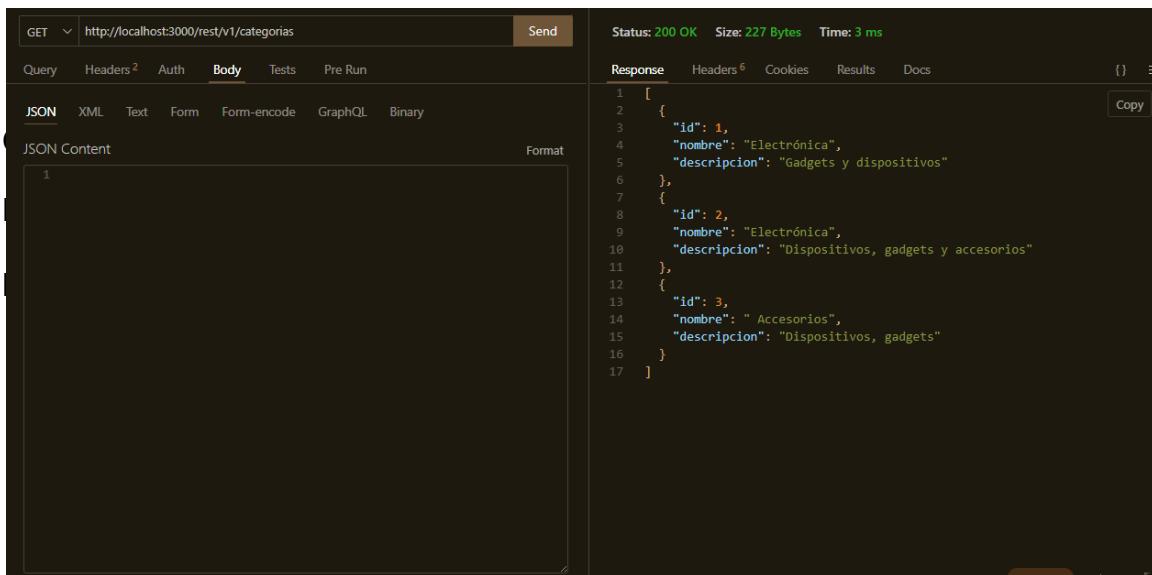
The screenshot shows a Postman request for a DELETE operation. The URL is `http://localhost:3000/rest/v1/users/17`. The response status is **200 OK**, size is **43 Bytes**, and time is **863 ms**. The response body is:

```
1  {
2    "message": "Usuario eliminado",
3    "data": null
4 }
```

Categorías

GET /categorias

Descripción: Obtiene la lista de categorías disponibles.



The screenshot shows a Postman request for a GET operation. The URL is `http://localhost:3000/rest/v1/categorias`. The response status is **200 OK**, size is **227 Bytes**, and time is **3 ms**. The response body is:

```
1  [
2    {
3      "id": 1,
4      "nombre": "Electrónica",
5      "descripcion": "Gadgets y dispositivos"
6    },
7    {
8      "id": 2,
9      "nombre": "Electrónica",
10     "descripcion": "Dispositivos, gadgets y accesorios"
11   },
12   {
13     "id": 3,
14     "nombre": "Accesorios",
15     "descripcion": "Dispositivos, gadgets"
16   }
17 ]
```

The screenshot shows a REST client interface with a dark theme. On the left, there's a configuration panel for a GET request to `http://localhost:3000/rest/v1/categorias/1`. The 'Query' tab is selected. In the 'Query Parameters' section, there is a single entry: a checkbox labeled 'parameter' next to a text input field labeled 'value'. On the right, the response panel shows a status of **200 OK**, size of **71 Bytes**, and time of **3 ms**. The 'Response' tab is selected, displaying the JSON data:

```
1 {  
2   "id": 1,  
3   "nombre": "Electrónica",  
4   "descripcion": "Gadgets y dispositivos"  
5 }
```

Casos de error

The screenshot shows a REST client interface with a dark theme. On the left, there's a configuration panel for a GET request to `http://localhost:3000/rest/v1/categorias/8`. The 'Query' tab is selected. In the 'Query Parameters' section, there is a single entry: a checkbox labeled 'parameter' next to a text input field labeled 'value'. On the right, the response panel shows a status of **404 Not Found**, size of **36 Bytes**, and time of **4 ms**. The 'Response' tab is selected, displaying the JSON data:

```
1 {  
2   "error": "Categoría no encontrada"  
3 }
```

A 'Copy' button is visible next to the response data.

POST /categorias

Descripción: Crea una nueva categoría de productos.

Body:

```
{  
  "nombre": "string"  
}
```

POST <http://localhost:3000/rest/v1/categorias>

Body¹

```

1 {
2   "nombre": "Wearables",
3   "descripcion": "Relojes inteligentes, pulseras de actividad y
4   accesorios conectados"
5 }
```

Status: 201 Created Size: 113 Bytes Time: 3 ms

PUT /categorias/:id

Descripción: Actualiza el nombre de una categoría existente.

PUT <http://localhost:3000/rest/v1/categorias/4>

Body¹

```

1 {
2   "nombre": "Wearables",
3   "descripcion": "Relojes inteligentes, accesorios conectados"
4 }
```

Status: 200 OK Size: 89 Bytes Time: 3 ms

DELETE /categorias/:id

Descripción: Elimina una categoría siempre que no tenga productos asociados.

DELETE <http://localhost:3000/rest/v1/categorias/1>

Query Parameters

<input type="checkbox"/> parameter	value
------------------------------------	-------

Status: 404 Not Found Size: 36 Bytes Time: 15 ms

Casos de error

The screenshot shows a Postman interface with a request to `http://localhost:3000/rest/v1/categorias/1`. The status bar indicates **ERROR**, **Size: 0 B**, and **Time: 0 ms**. The response pane displays the message **Connection was refused by the server.**

Productos

GET /productos

Descripción: Obtiene la lista de productos disponibles en la tienda.

The screenshot shows a Postman interface with a request to `http://localhost:3000/rest/v1/productos`. The status bar indicates **200 OK**, **Size: 631 Bytes**, and **Time: 4 ms**. The response pane displays the following JSON data:

```
[{"id": 1, "nombre": "Altavoz Bluetooth Port\u00e1til", "descripcion": "Resistente al agua, 12 horas de reproducci\u00f3n", "precio": 49, "stock": 25, "imagen_url": "", "categoria_id": 1}, {"id": 2, "nombre": "Power Bank 20000mAh", "descripcion": "Bater\u00eda externa con carga r\u00e1pida USB-C", "precio": 39.5, "stock": 50, "imagen_url": "", "categoria_id": 1}, {"id": 3, "nombre": "Cargador inal\u00e1mbrico 15W", "descripcion": "Base de carga r\u00e1pida compatible Qi", "precio": 24.99, "stock": 40, "imagen_url": "", "categoria_id": 1}]
```

GET /productos/:id

Descripción: Obtiene la información detallada de un producto.

Parámetros URL:

- id (number)

The screenshot shows a REST client interface with two main sections: 'Query' and 'Response'.

Query Section:

- Method: GET
- URL: http://localhost:3000/rest/v1/productos/1
- Headers: Headers², Auth, Body¹, Tests, Pre Run
- Send button

Query Parameters:

parameter	value

Response Section:

- Status: 200 OK, Size: 165 Bytes, Time: 4 ms
- Headers: Headers⁶, Cookies, Results, Docs
- JSON Response:

```
1 {
2   "id": 1,
3   "nombre": "Altavoz Bluetooth Portátil",
4   "descripcion": "Resistente al agua, 12 horas de reproducción",
5   "precio": 45,
6   "stock": 25,
7   "imagen_url": "",
8   "categoria_id": 1
9 }
```

POST /productos

Descripción: Crea un nuevo producto y lo asocia a una categoría.

Body:

```
{  
  "nombre": "string",  
  "precio": "number",  
  "categoria_id": "number"  
  "stock": "number"
```

```
}
```

POST <http://localhost:3000/rest/v1/productos> Send

Query Headers² Auth Body¹ Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "nombre": "Auriculares Bluetooth",
3   "descripcion": "Inalámbricos, cancelación de ruido, color negro",
4   "precio": 59.99,
5   "categoria_id": 1,
6   "stock": 30
7 }
```

Status: 201 Created Size: 166 Bytes Time: 3 ms

Response Headers⁶ Cookies Results Docs { } ≡

```
1 {
2   "id": 5,
3   "nombre": "Auriculares Bluetooth",
4   "descripcion": "Inalámbricos, cancelación de ruido, color negro",
5   "precio": 59.99,
6   "stock": 30,
7   "imagen_url": "",
8   "categoria_id": 1
9 }
```

PUT /productos/:id

Descripción: Actualiza los datos de un producto existente.

PUT <http://localhost:3000/rest/v1/productos/5> Send

Query Headers² Auth Body¹ Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "precio": 44.99,
3   "stock": 25
4 }
```

Status: 200 OK Size: 166 Bytes Time: 2 ms

Response Headers⁶ Cookies Results Docs { } ≡

```
1 {
2   "id": 5,
3   "nombre": "Auriculares Bluetooth",
4   "descripcion": "Inalámbricos, cancelación de ruido, color negro",
5   "precio": 44.99,
6   "stock": 25,
7   "imagen_url": "",
8   "categoria_id": 1
9 }
```

DELETE /productos/:id

Descripción: Elimina un producto del sistema.

DELETE http://localhost:3000/rest/v1/productos/4

Query Headers ² Auth Body ¹ Tests Pre Run

Query Parameters

<input type="checkbox"/> parameter	value
------------------------------------	-------

Status: 200 OK Size: 151 Bytes Time: 3 ms

Response Headers ⁶ Cookies Results Docs

```

1  {
2    "id": 4,
3    "nombre": "Ratón Gaming RGB",
4    "descripcion": "Sensor 16000 DPI, botones programables",
5    "precio": 69.99,
6    "stock": 12,
7    "imagen_url": "",
8    "categoria_id": 1
9  }

```

DELETE http://localhost:3000/rest/v1/productos/4

Query Headers ² Auth Body ¹ Tests Pre Run

Query Parameters

<input type="checkbox"/> parameter	value
------------------------------------	-------

Status: 404 Not Found Size: 34 Bytes Time: 2 ms

Response Headers ⁶ Cookies Results Docs

```

1  {
2    "error": "Producto no encontrado"
3  }

```

Carrito

GET /carrito

Descripción: Obtiene la lista de carritos registrados.

GET http://localhost:3000/rest/v1/carrito

Query Headers ² Auth Body Tests Pre Run

Query Parameters

<input type="checkbox"/> parameter	value
------------------------------------	-------

Status: 200 OK Size: 89 Bytes Time: 791 ms

Response Headers ⁶ Cookies Results Docs

```

1  [
2    {
3      "id": 5,
4      "usuario_id": 5,
5      "estado": "activo"
6    },
7    {
8      "id": 6,
9      "usuario_id": 6,
10     "estado": "completado"
11   }
12 ]

```

GET /carrito/:id

Descripción: Obtiene la información de un carrito específico.

The screenshot shows a REST client interface with a request panel on the left and a response panel on the right. In the request panel, a GET request is made to `http://localhost:3000/rest/v1/carrito/5`. The response panel displays a status of 200 OK, a size of 41 Bytes, and a time of 696 ms. The response body is a JSON object:

```
1  {
2    "id": 5,
3    "usuario_id": 5,
4    "estado": "activo"
5 }
```

Casos de error

The screenshot shows a REST client interface with a request panel on the left and a response panel on the right. In the request panel, a GET request is made to `http://localhost:3000/rest/v1/carrito/3`. The response panel displays a status of 404 Not Found, a size of 33 Bytes, and a time of 565 ms. The response body is a JSON object:

```
1  {
2    "error": "Carrito no encontrado"
3 }
```

POST /carrito

Descripción: Crea un nuevo carrito asociado a un usuario.

Body:

```
{
  "usuario_id": "uuid"
```

}

A screenshot of a REST client interface showing a successful POST request. The URL is `http://localhost:3000/rest/v1/carrito`. The response status is `201 Created`, size is `40 Bytes`, and time is `1.42 s`. The response body is:

```
1 {
2   "message": "Carrito creado",
3   "data": null
4 }
```

Casos de error

A screenshot of a REST client interface showing an error response. The URL is `http://localhost:3000/rest/v1/carrito`. The response status is `400 Bad Request`, size is `32 Bytes`, and time is `693 ms`. The response body is:

```
1 {
2   "error": "usuario_id no existe"
3 }
```

PUT /carrito/:id

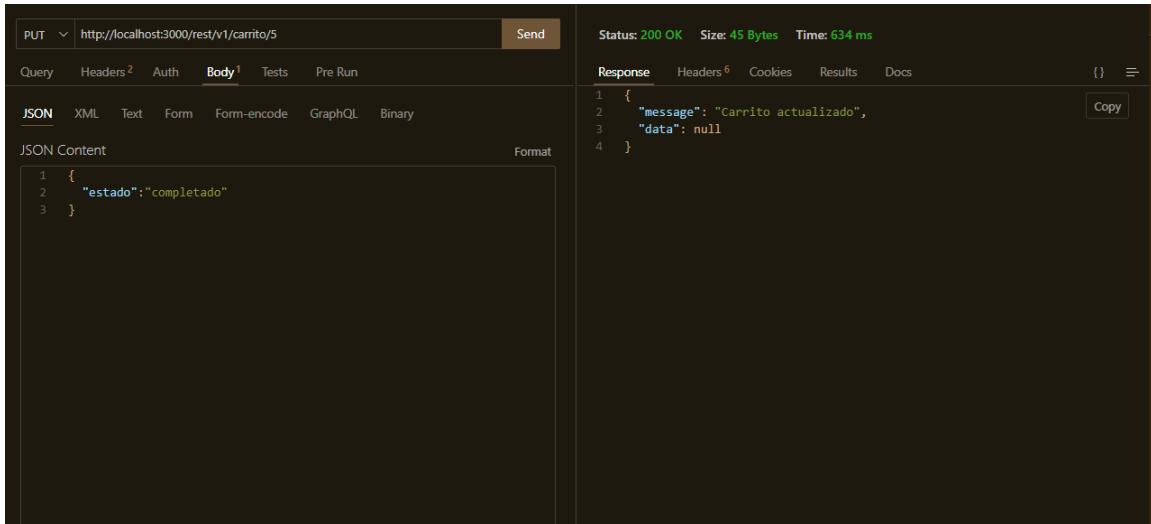
Descripción: Actualiza el estado de un carrito.

Body:

{

"estado": "activo,completado, abandonado"

}



PUT http://localhost:3000/rest/v1/carrito/5

Body (JSON)

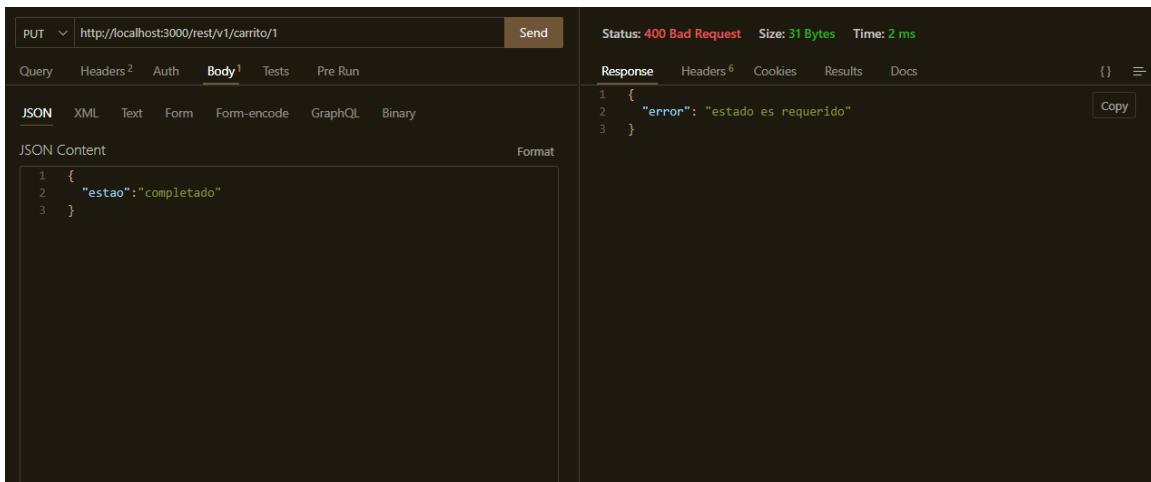
```
1 {
2   "estado": "completado"
3 }
```

Status: 200 OK Size: 45 Bytes Time: 634 ms

Response Headers Cookies Results Docs

```
1 {
2   "message": "Carrito actualizado",
3   "data": null
4 }
```

Casos de error



PUT http://localhost:3000/rest/v1/carrito/1

Body (JSON)

```
1 {
2   "estao": "completado"
3 }
```

Status: 400 Bad Request Size: 31 Bytes Time: 2 ms

Response Headers Cookies Results Docs

```
1 {
2   "error": "estado es requerido"
3 }
```

DELETE /carrito/:id

Descripción: Elimina un carrito del sistema.



DELETE http://localhost:3000/rest/v1/carrito/1

Query Parameters

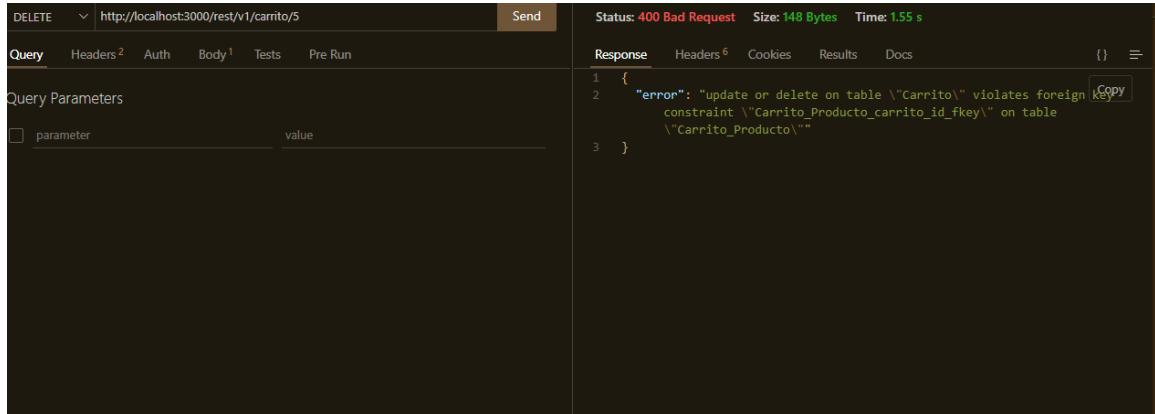
parameter	value

Status: 200 OK Size: 43 Bytes Time: 331 ms

Response Headers Cookies Results Docs

```
1 {
2   "message": "Carrito eliminado",
3   "data": null
4 }
```

Casos de error



DELETE Send

Query Headers² Auth Body¹ Tests Pre Run

Query Parameters

parameter value

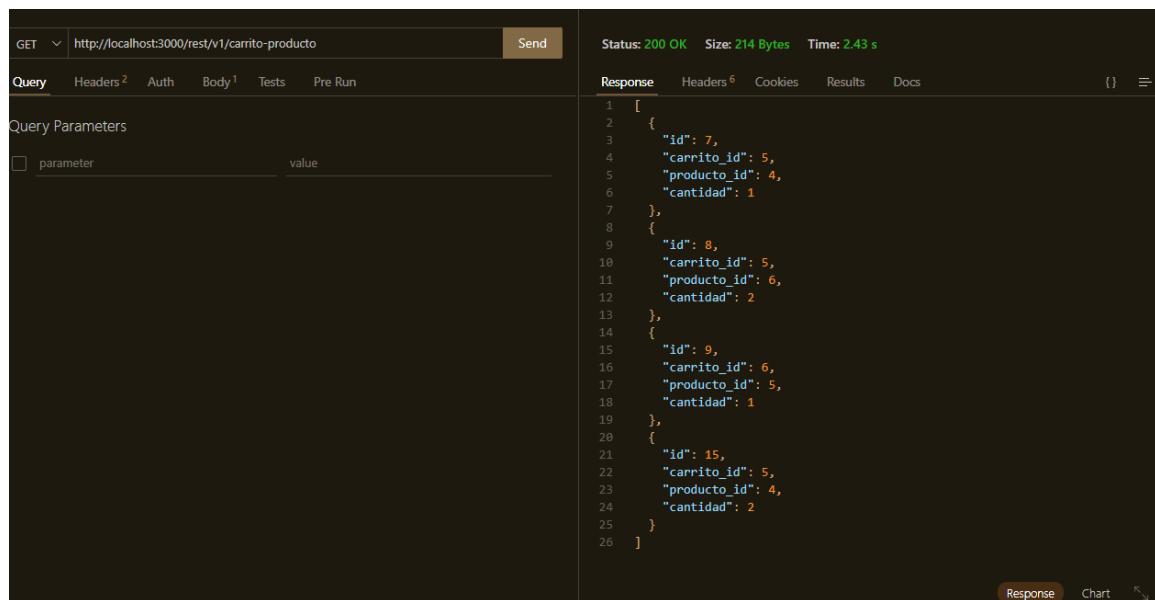
Status: 400 Bad Request Size: 148 Bytes Time: 1.55 s

Response Headers⁶ Cookies Results Docs

```
1  {
2    "error": "update or delete on table \"Carrito\" violates foreign key constraint \"Carrito_Producto_carrito_id_fkey\" on table \"Carrito_Producto\""
3 }
```

GET /carrito-producto

Descripción: Obtiene la lista de productos agregados a los carritos.



GET Send

Query Headers² Auth Body¹ Tests Pre Run

Query Parameters

parameter value

Status: 200 OK Size: 214 Bytes Time: 2.43 s

Response Headers⁶ Cookies Results Docs

```
1 [
2   {
3     "id": 7,
4     "carrito_id": 5,
5     "producto_id": 4,
6     "cantidad": 1
7   },
8   {
9     "id": 8,
10    "carrito_id": 5,
11    "producto_id": 6,
12    "cantidad": 2
13 },
14 {
15   "id": 9,
16   "carrito_id": 6,
17   "producto_id": 5,
18   "cantidad": 1
19 },
20 {
21   "id": 15,
22   "carrito_id": 5,
23   "producto_id": 4,
24   "cantidad": 2
25 }
26 ]
```

Casos de error



GET Send

Query Headers² Auth Body¹ Tests Pre Run

Query Parameters

parameter value

Status: 404 Not Found Size: 162 Bytes Time: 4 ms

Response Headers⁷ Cookies Results Docs

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <title>Error</title>
6   </head>
7   <body>
8     <pre>Cannot GET /rest/v1/carrito-produco</pre>
9   </body>
10  </html>
```

POST /carrito-producto

Descripción: Agrega un producto a un carrito específico.

Body:

```
{  
  "carrito_id": "uuid",  
  "producto_id": "number",  
  "cantidad": "number",  
  "precio_unitario": "number"}  
}
```

The screenshot shows a Postman request configuration and its response. The request is a POST to `http://localhost:3000/rest/v1/carrito-producto`. The body is set to JSON with the following content:

```
{  
  "carrito_id": 5,  
  "producto_id": 4,  
  "cantidad": 3,  
  "precio_unitario": 10000  
}
```

The response status is 201 Created, size is 54 Bytes, and time is 988 ms. The response body is:

```
{  
  "message": "Producto agregado al carrito",  
  "data": null  
}
```

Casos de error

The screenshot shows a Postman request configuration and its response. The request is a POST to `http://localhost:3000/rest/v1/carrito-producto`. The body is set to JSON with the following content:

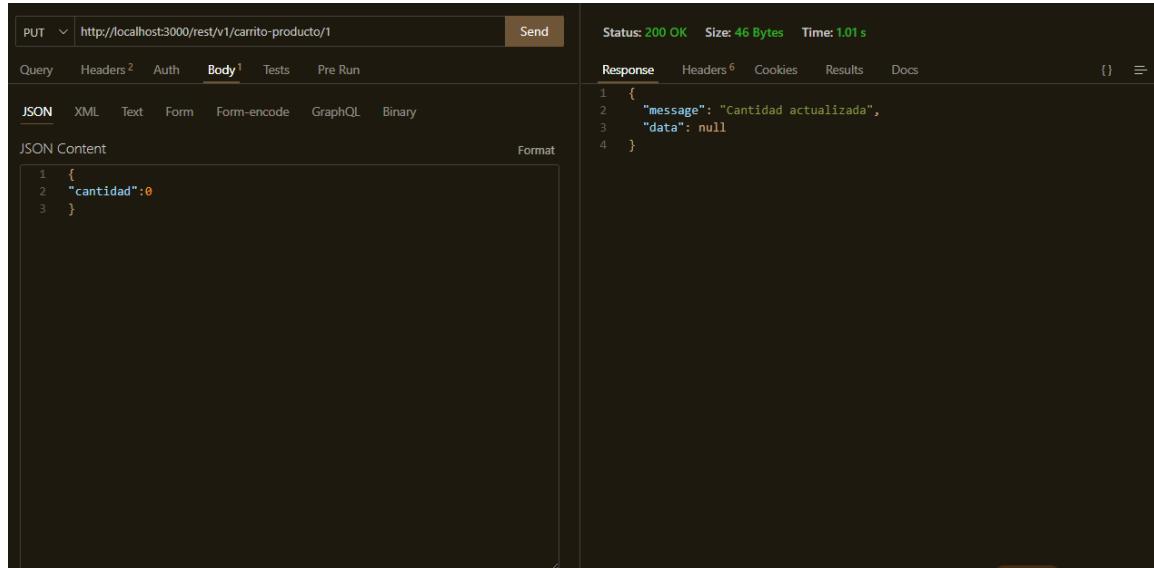
```
{  
  "carrito_id": 5,  
  "producto_id": 2,  
  "cantidad": 3,  
  "precio_unitario": 10000  
}
```

The response status is 400 Bad Request, size is 33 Bytes, and time is 1.17 s. The response body is:

```
{  
  "error": "producto_id no existe"  
}
```

PUT /carrito-producto

Descripción: Actualizar cantidad



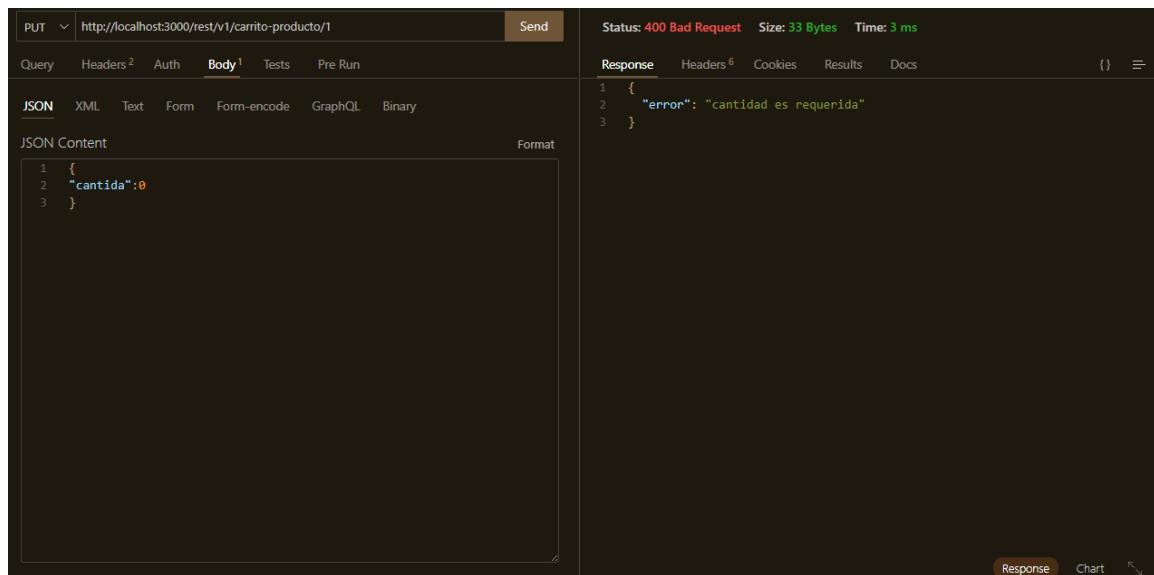
PUT http://localhost:3000/rest/v1/carrito-producto/1 Send

Status: 200 OK Size: 46 Bytes Time: 1.01 s

Response Headers Cookies Results Docs

```
1 {
2   "message": "Cantidad actualizada",
3   "data": null
4 }
```

Casos de error



PUT http://localhost:3000/rest/v1/carrito-producto/1 Send

Status: 400 Bad Request Size: 33 Bytes Time: 3 ms

Response Headers Cookies Results Docs

```
1 {
2   "error": "cantidad es requerida"
3 }
```

DELETE /carrito-producto/:id

Descripción: Elimina un producto del carrito.

The screenshot shows the Postman interface with a successful DELETE request. The URL is `http://localhost:3000/rest/v1/carrito-producto/1`. The response status is 200 OK, size is 56 Bytes, and time is 319 ms. The response body is:

```
1 {
2   "message": "Producto eliminado del carrito",
3   "data": null
4 }
```

Casos de error

The screenshot shows the Postman interface with a 404 Not Found error. The URL is `http://localhost:3000/rest/v1/carrito-producto/`. The response status is 404 Not Found, size is 167 Bytes, and time is 3 ms. The response body is:

```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <title>Error</title>
6   </head>
7   <body>
8     <pre>Cannot DELETE /rest/v1/carrito-producto/</pre>
9   </body>
10  </html>
```

Órdenes

GET /orden

Descripción: Obtiene la lista de órdenes registradas.

GET Send

Query Headers ² Auth Body ¹ Tests Pre Run

Query Parameters

<input type="checkbox"/> parameter	value
------------------------------------	-------

Status: 200 OK Size: 188 Bytes Time: 1.03 s

Response Headers ⁶ Cookies Results Docs

```

1 [
2   {
3     "id": 9,
4     "usuario_id": 5,
5     "fecha": "2025-12-15T16:35:03.945",
6     "estado": "pendiente",
7     "total": 150.5
8   },
9   {
10    "id": 11,
11    "usuario_id": 5,
12    "fecha": "2025-12-15T19:14:53.208",
13    "estado": "pendiente",
14    "total": 150.5
15  }
16 ]

```

Response Chart

GET /orden/:id

Descripción: Obtiene el detalle de una orden específica, incluyendo productos.

GET Send

Query Headers ² Auth Body ¹ Tests Pre Run

Query Parameters

<input type="checkbox"/> parameter	value
------------------------------------	-------

Status: 200 OK Size: 92 Bytes Time: 329 ms

Response Headers ⁶ Cookies Results Docs

```

1 {
2   "id": 9,
3   "usuario_id": 5,
4   "fecha": "2025-12-15T16:35:03.945",
5   "estado": "pendiente",
6   "total": 150.5
7 }

```

Casos de error

GET Send

Query Headers ² Auth Body ¹ Tests Pre Run

Query Parameters

<input type="checkbox"/> parameter	value
------------------------------------	-------

Status: 404 Not Found Size: 60 Bytes Time: 384 ms

Response Headers ⁶ Cookies Results Docs

```

1 [
2   {
3     "error": "Cannot coerce the result to a single JSON object"
4   }
5 ]

```

POST /orden

Descripción: Crea una nueva orden asociada a un usuario.

Body:

```
{  
  "user_id": "number"  
}
```

The screenshot shows a Postman request to `http://localhost:3000/rest/v1/orden`. The request method is `POST`. The response status is `201 Created`, size is `157 Bytes`, and time is `497 ms`. The response body is a JSON object:

```
{  
  "message": "Orden creada correctamente",  
  "orden_id": 12,  
  "data": [  
    {  
      "id": 12,  
      "usuario_id": 5,  
      "fecha": "2025-12-15T21:18:17.303",  
      "estado": "pendiente",  
      "total": 150.5  
    }  
  ]  
}
```

Casos de error

The screenshot shows a Postman request to `http://localhost:3000/rest/v1/orden`. The request method is `POST`. The response status is `400 Bad Request`, size is `1.13 KB`, and time is `4 ms`. The response body is a JSON object with an error message:

```
<!DOCTYPE html>  
<html lang="en">  
  <head>  
    <meta charset="utf-8">  
    <title>Error</title>  
  </head>  
  <body>  
    <pre>SyntaxError: Unexpected token '}'"total"  
    :  
    <br>&quot; is not valid JSON<br>  at JSON.parse (<anonymous>:1:16)<br>  at parse (C:\Users\usuario\backend\ecommerce_backend\node_modules\body-parser\lib\types\Json.js:7:19)<br>  at C:\Users\usuario\backend\ecommerce_backend\node_modules\body-parser\lib\read.js:123:18<br>  at AsyncResource.runInAsyncScope (node:async_hooks:214:14)<br>  invokeCallback (C:\Users\usuario\backend\ecommerce_backend\node_modules\raw-body\index.js:238:16)<br>  at done (C:\Users\usuario\backend\ecommerce_backend\node_modules\raw-body\index.js:227:7)<br>  at IncomingMessage.onEnd (C:\Users\usuario\backend\ecommerce_backend\node_modules\raw-body\index.js:287:7)<br>  at IncomingMessage.emit (node:events:508:28)<br>  at endReadableNT (node:internal/streams/readable:1701:12)<br>  at processTicksAndRejections (node:internal/process/task_queues:89:21)</pre>
```

PUT /orden/:id

Descripción: Actualiza el estado o total de una orden.

Body:

```
{
```

```
  "estado": "creada, pendiente, enviada, entregada, cancelada",
```

```
  "total": "number"
```

```
}
```

PUT http://localhost:3000/rest/v1/orden/1

Status: 200 OK Size: 45 Bytes Time: 986 ms

Response Headers Cookies Results Docs

```
1 {  
2   "message": "Orden actualizada correctamente"  
3 }
```

Query Headers² Auth Body¹ Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {  
2   "estado": "completada",  
3   "total": 180.75  
4 }
```

Casos de error

PUT http://localhost:3000/rest/v1/orden/

Status: 404 Not Found Size: 153 Bytes Time: 4 ms

Response Headers⁷ Cookies Results Docs

```
1 <!DOCTYPE html>  
2 <html lang="en">  
3   <head>  
4     <meta charset="utf-8">  
5     <title>Error</title>  
6   </head>  
7   <body>  
8     <pre>Cannot PUT /rest/v1/orden/</pre>  
9   </body>  
10  </html>
```

Query Headers² Auth Body¹ Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {  
2   "estado": "completada",  
3   "total": 180.75  
4 }
```

DELETE /orden/:id

Descripción: Elimina una orden del sistema.

```

Status: 200 OK  Size: 60 Bytes  Time: 754 ms
Response Headers 6 Cookies Results Docs
1 {
2   "message": "Orden y sus productos eliminados correctamente"
3 }

```

GET /orden-producto

Descripción: Obtiene los productos asociados a las órdenes.

```

Status: 200 OK  Size: 77 Bytes  Time: 308 ms
Response Headers 6 Cookies Results Docs
1 [
2   {
3     "id": 19,
4     "orden_id": 9,
5     "producto_id": 5,
6     "cantidad": 2,
7     "precio_unitario": 25.99
8   }
9 ]

```

POST /orden-producto

Descripción: Agrega un producto a una orden.

```

Status: 201 Created  Size: 135 Bytes  Time: 495 ms
Response Headers 6 Cookies Results Docs
1 {
2   "message": "Producto agregado a la orden",
3   "id": 21,
4   "data": [
5     {
6       "id": 21,
7       "orden_id": 9,
8       "producto_id": 5,
9       "cantidad": 2,
10      "precio_unitario": 25.99
11    }
12  ]
13 }

```

Casos de error

The screenshot shows a POST request to `http://localhost:3000/rest/v1/orden-producto`. The request body is a JSON object:

```
1 {
2   "orden_id": 6,
3   "producto_id": 5,
4   "cantidad": 2,
5   "precio_unitario": 25.99
6 }
```

The response status is **400 Bad Request**, size is **121 Bytes**, and time is **311 ms**. The response body is:

```
1 {
2   "error": "insert or update on table \"Orden_Producto\" violates foreign
            key constraint \"Orden_Producto_orden_id_fkey\""
3 }
```

PUT /orden-producto/:id

Descripción: Actualiza un producto dentro de una orden.

The screenshot shows a PUT request to `http://localhost:3000/rest/v1/orden-producto/1`. The request body is a JSON object:

```
1 {
2   "cantidad": 3,
3   "precio_unitario": 27.50
4 }
```

The response status is **200 OK**, size is **46 Bytes**, and time is **408 ms**. The response body is:

```
1 {
2   "message": "Producto de la orden actualizado"
3 }
```

DELETE /orden-producto/:id

Descripción: Elimina un producto de una orden.

The screenshot shows a DELETE request to `http://localhost:3000/rest/v1/orden-producto/1`. There are no parameters in the query string.

The response status is **200 OK**, size is **44 Bytes**, and time is **331 ms**. The response body is:

```
1 {
2   "message": "Producto eliminado de la orden"
3 }
```

Códigos de Estado HTTP

Código	Descripción
200	OK
201	Created
400	Bad Request
401	Unauthorized
404	Not Found
409	Conflict
500	Internal Server Error