

## ¿Qué es un GUID?

Un GUID (Globally Unique Identifier), o Identificador Único Global, es un valor numérico de 128 bits que se utiliza para identificar de manera única objetos dentro de sistemas informáticos distribuidos. Su principal ventaja es que garantiza unicidad sin necesidad de una coordinación central entre sistemas, lo que lo hace ideal para bases de datos, APIs y sistemas escalables.

El formato típico de un GUID es el siguiente:

**Por ejemplo:** a1b2c3d4-e5f6-7890-1234-567890abcdef

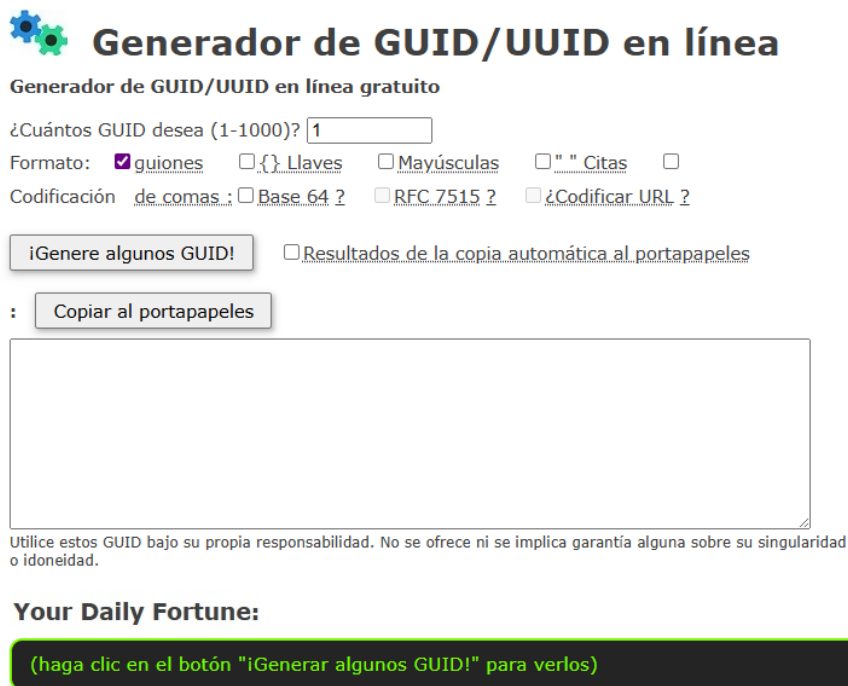
En este trabajo práctico, los GUIDs son utilizados como identificadores únicos para entidades como órdenes y productos, asegurando que cada registro pueda ser referenciado de forma precisa e irrepetible.

## Formas de generar un GUID

Existen múltiples formas de generar GUIDs. A continuación, se detallan tres métodos comunes:

### 1. Generación mediante un sitio web (método utilizado)

Para la realización de este trabajo práctico, se utilizó un generador online disponible en el sitio:



The screenshot shows a web application titled "Generador de GUID/UUID en línea". It includes a sub-header "Generador de GUID/UUID en línea gratuito". The interface has several input fields and checkboxes for customizing the GUID generation. A large empty box is provided for the generated GUIDs. At the bottom, there is a section titled "Your Daily Fortune:" with a button that says "(haga clic en el botón '¡Generar algunos GUID!' para verlos)".

**Generador de GUID/UUID en línea**

Generador de GUID/UUID en línea gratuito

¿Cuántos GUID desea (1-1000)?

Formato: ☒ guiones ☐ { } Llaves ☐ Mayúsculas ☐ " " Citas ☐

Codificación de comas : ☐ Base 64 ? ☐ RFC 7515 ? ☐ ¿Codificar URL ?

☐ Resultados de la copia automática al portapapeles

:

Utilice estos GUID bajo su propia responsabilidad. No se ofrece ni se implica garantía alguna sobre su singularidad o idoneidad.

**Your Daily Fortune:**

(haga clic en el botón "¡Generar algunos GUID!" para verlos)

## 2. Generación mediante PowerShell en Windows

En sistemas operativos Windows, se puede generar un GUID desde la consola de PowerShell utilizando el siguiente comando:

```
powershell
```

```
[guid]::NewGuid()
```

**3. En sistemas basados en Linux (o WSL en Windows), se puede utilizar el comando uuidgen para generar GUIDs. En caso de que no esté instalado, existe una alternativa:**

- uuidgen

O bien:

- cat /proc/sys/kernel/random/uuid

### Pasos a Seguir:

#### 1. Crear una nueva orden:

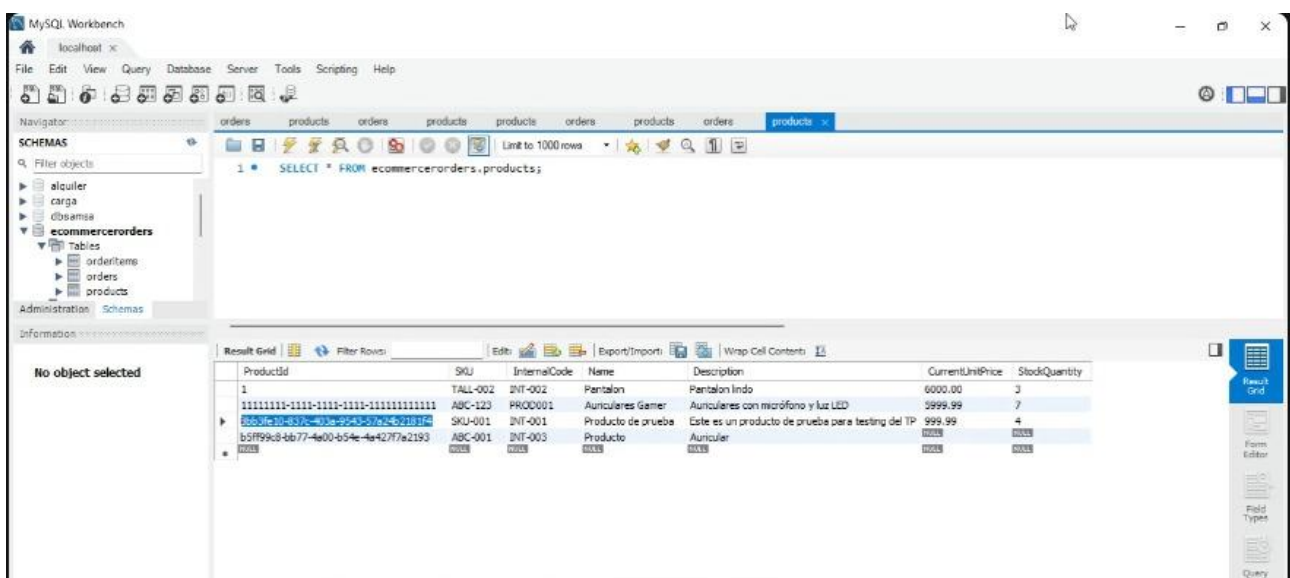
#### 2. Método HTTP: POST

#### 3. Ruta: /api/orders

**4. Descripción:** Permite registrar una nueva orden de compra en el sistema. La orden debe incluir un identificador de cliente (simulado), direcciones de envío y facturación, y una lista de ítems que componen la orden (productos con sus cantidades y precios unitarios al momento de la compra). Antes de crear la orden, se debe verificar que haya suficiente stock disponible para cada producto solicitado. Si el stock es insuficiente para algún producto, la orden no debe crearse y se debe retornar un error. Si la orden se crea exitosamente, el stock de los productos involucrados debe ser decrementado. No se debe implementar ninguna lógica de pago.

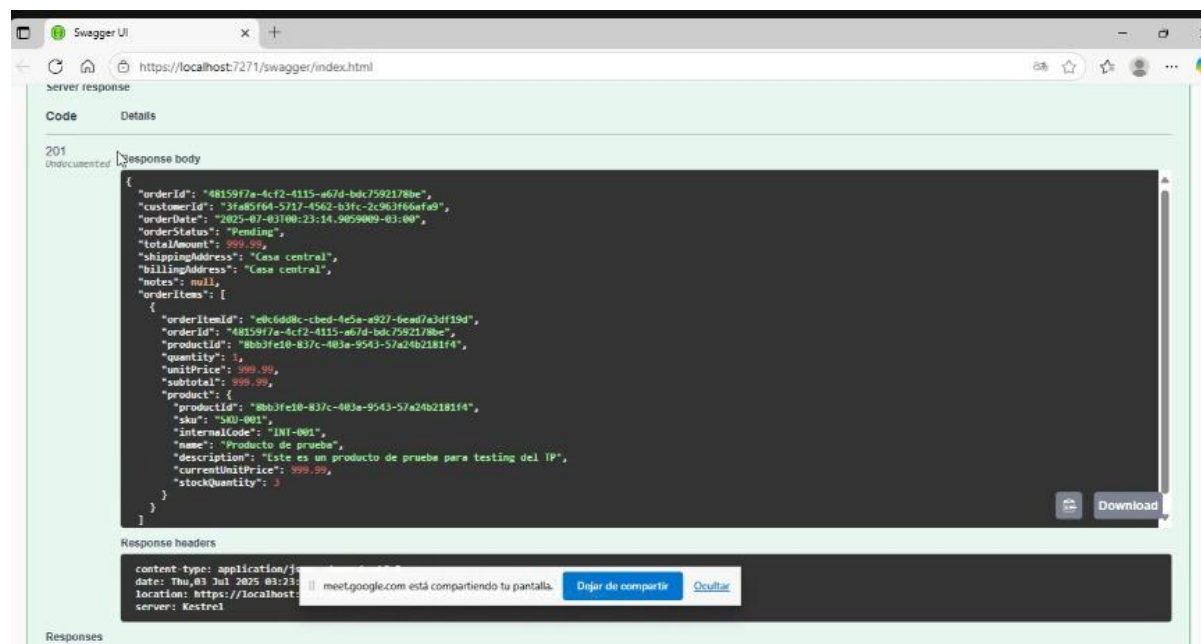
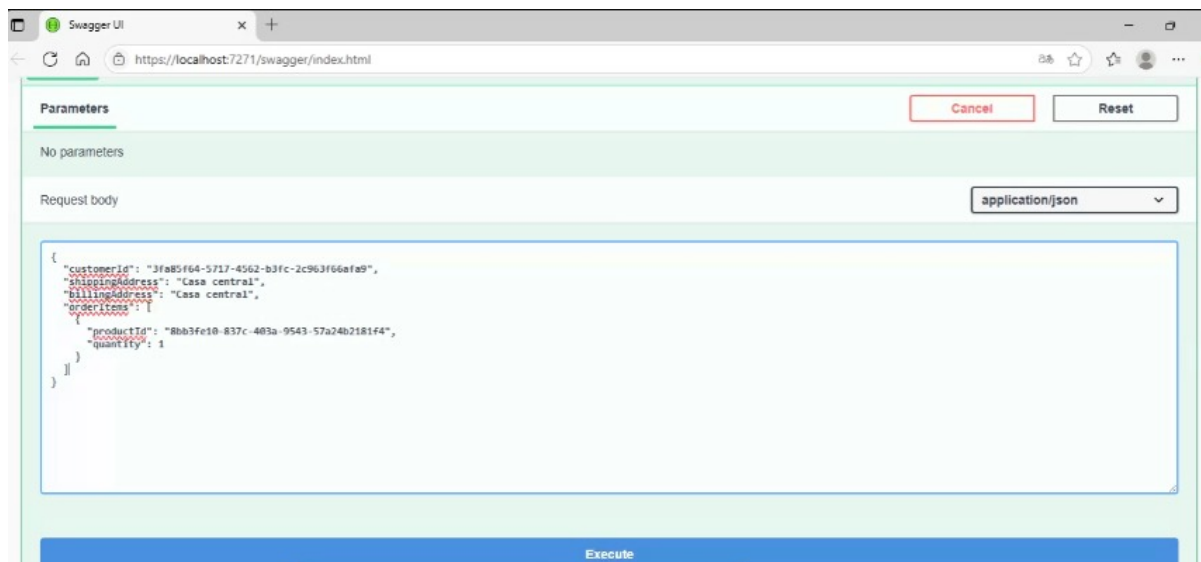
### Precondiciones

- Los productos enviados en la orden existen en la base de datos.
- Los productos tienen suficiente stock ( $\text{StockQuantity} \geq \text{quantity}$ ).
- El formato del JSON es válido y cumple con las validaciones del backend.

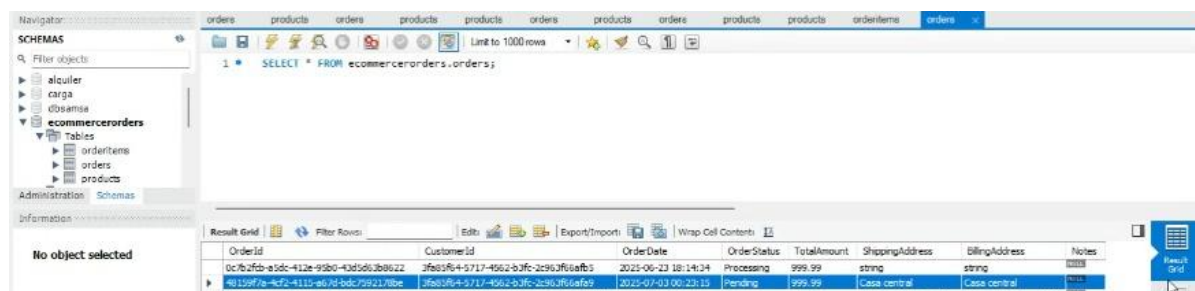


The screenshot shows the MySQL Workbench interface. The 'Navigator' pane on the left displays the database schema, including the 'ecommerceorders' database with tables 'orderitems', 'orders', and 'products'. The 'Query' pane shows a SQL query: `SELECT * FROM ecommerceorders.products;`. The 'Result Grid' pane displays the query results in a table format.

ProductId	SKU	InternalCode	Name	Description	CurrentUnitPrice	StockQuantity
1	TALL-002	ZNT-002	Pantalón	Pantalón lino	6000.00	3
11111111-1111-1111-1111-111111111111	ABC-123	PROD001	Auriculares Gamer	Auriculares con micrófono y luz LED	3999.99	7
b5b7e10d-837c-403a-9243-5762c471817c	SKU-001	ZNT-001	Producto de prueba	Este es un producto de prueba para testing del TP	999.99	4
b5f99c8-bb77-4e00-b54e-4a4277e2193	ABC-001	ZNT-003	Producto	Auricular	0000	0000

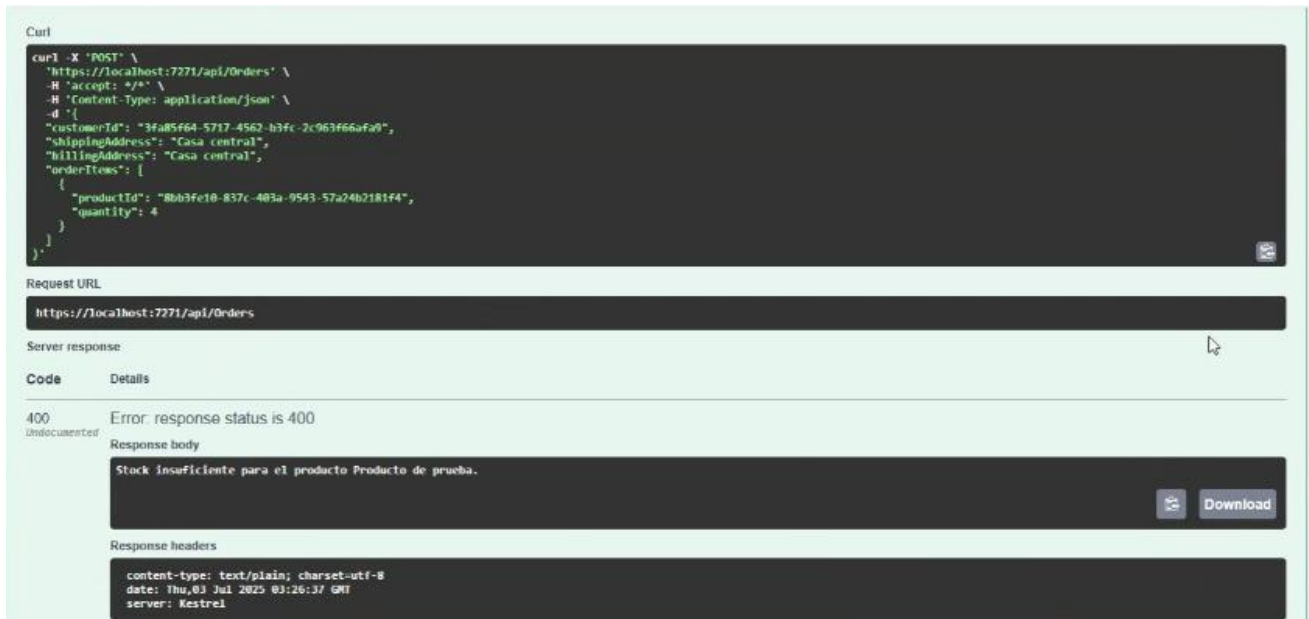


## Caso de error del primer endpoint



## Vista del Código





## 1.Verificaciones

- Se crea un nuevo registro en la tabla Orders con los datos enviados.
- Se crean registros asociados en OrderItems.
- Se actualiza la columna StockQuantity en la tabla Products reduciendo las cantidades correspondientes.
- Se calcula correctamente el totalAmount de la orden.
- El estado inicial de la orden es "Pending"

## 2. Obtener todas las órdenes:

- **Método HTTP:** GET
- **Ruta:** /api/orders
- **Descripción:** Retorna una lista paginada de todas las órdenes registradas en el sistema.

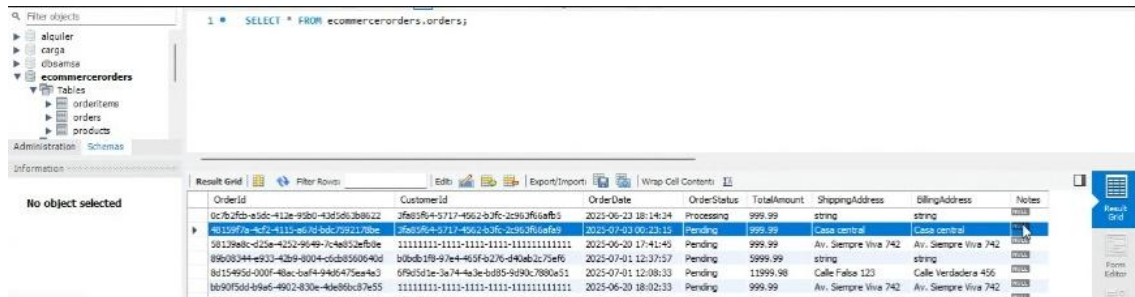
## 3.Parámetros de Consulta (Opcional):

- **status:** Permite filtrar las órdenes por su estado (ej. Pending, Processing, Shipped, Delivered, Cancelled).
- **customerId:** Permite filtrar las órdenes por un ID de cliente específico.
- **pageNumber:** Número de página para paginación (ej. 1).
- **pageSize:** Cantidad de elementos por página (ej. 10).
- **Respuesta Exitosa:** 200 OK con un array de objetos de órdenes.
- **Respuesta de Error:** 500 Internal Server Error en caso de un fallo inesperado del servidor.

#### 4.Precondiciones

- La base de datos contiene al menos una orden registrada.
- El backend está correctamente conectado a la base de datos mediante Entity Framework Core.
- Si se usan filtros (status, customerId), deben existir registros que coincidan con dichos filtros.
- Los valores de pageNumber y pageSize deben ser numéricos y mayores a 0.

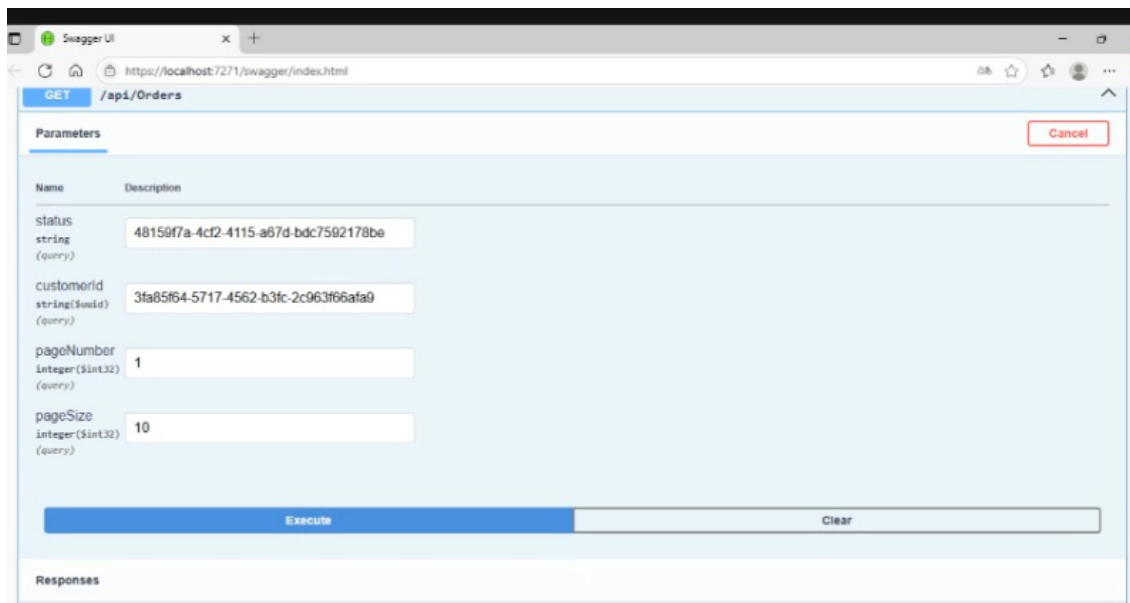
#### Caso de éxito del segundo endpoint



The screenshot shows the SQL Server Enterprise Manager interface. On the left, the 'ecommerceorders' database is expanded, showing tables like 'ordersItems', 'orders', and 'products'. The main pane displays a query: `SELECT * FROM ecommerceorders.orders;`. Below the query, a 'Result Grid' shows the data. The table has columns: OrderId, CustomerId, OrderDate, OrderStatus, TotalAmount, ShippingAddress, BillingAddress, and Notes. The data is paginated, showing 10 rows per page. The first row is highlighted.

OrderId	CustomerId	OrderDate	OrderStatus	TotalAmount	ShippingAddress	BillingAddress	Notes
0c7b7d8-a5d6-412e-9980-43d5d63b8022	3fa85f64-5717-4562-b3fc-2c963f66afa9	2025-06-23 18:14:34	Processing	999.99	string	string	
4c1b079e-529411a-c010-b0c0002170c0e	3fa85f64-5717-4562-b3fc-2c963f66afa9	2025-07-03 09:23:12	Pending	999.99	string	string	
981298dc-d22e-4252-9649-7c4e852aef0e	11111111-1111-1111-1111-111111111111	2025-06-20 17:41:40	Pending	999.99	Av. Siempre Viva 742	Av. Siempre Viva 742	
89600344-e933-42b9-8004-c5d8550640d	b0bd19f-97e4-465f-b276-d40ab2c75ef6	2025-07-01 12:37:57	Pending	9999.99	string	string	
8d15495d-000f-48ec-baf4-94d6475ea4e3	6f9d5d1e-3a74-4a3e-bd85-9d90c7880e51	2025-07-01 12:08:33	Pending	11999.98	Calle Falsa 123	Calle Verdadera 456	
b690f5dd-b9a6-4602-830e-4de86bc87e55	11111111-1111-1111-1111-111111111111	2025-06-20 18:02:33	Pending	999.99	Av. Siempre Viva 742	Av. Siempre Viva 742	

#### Vista en la API



The screenshot shows the Swagger UI for the endpoint `GET /api/Orders`. The parameters section lists four query parameters: `status` (string), `customerId` (string), `pageNumber` (integer), and `pageSize` (integer). The values entered are: `status=48159f7a-4cf2-4115-a67d-bdc7592178be`, `customerId=3fa85f64-5717-4562-b3fc-2c963f66afa9`, `pageNumber=1`, and `pageSize=10`. There are 'Execute' and 'Clear' buttons at the bottom of the parameters section.



The screenshot shows the 'Responses' section of the Swagger UI. It displays a curl command and the response for the `GET /api/Orders` endpoint. The response is a 200 status code with a JSON body. The JSON body contains a 'total' field, a 'page' field, a 'pageSize' field, and a 'data' field. The response headers show the content type as 'application/json' and the server as 'Nestlé'.

```
curl -X 'GET' \
  'https://localhost:7271/api/Orders?status=48159f7a-4cf2-4115-a67d-bdc7592178be&customerId=3fa85f64-5717-4562-b3fc-2c963f66afa9&pageNumber=1&pageSize=10' \
  -H 'accept: */*'

Request URL
https://localhost:7271/api/Orders?status=48159f7a-4cf2-4115-a67d-bdc7592178be&customerId=3fa85f64-5717-4562-b3fc-2c963f66afa9&pageNumber=1&pageSize=10

Server response
Code Details
200
Response body
{
  "total": 0,
  "page": 1,
  "pageSize": 10,
  "data": []
}
Response headers
content-type: application/json; charset=utf-8
date: Thu, 03 Jul 2025 03:31:07 GMT
server: Nestlé

Responses
Code Description Links
200 OK No links
```

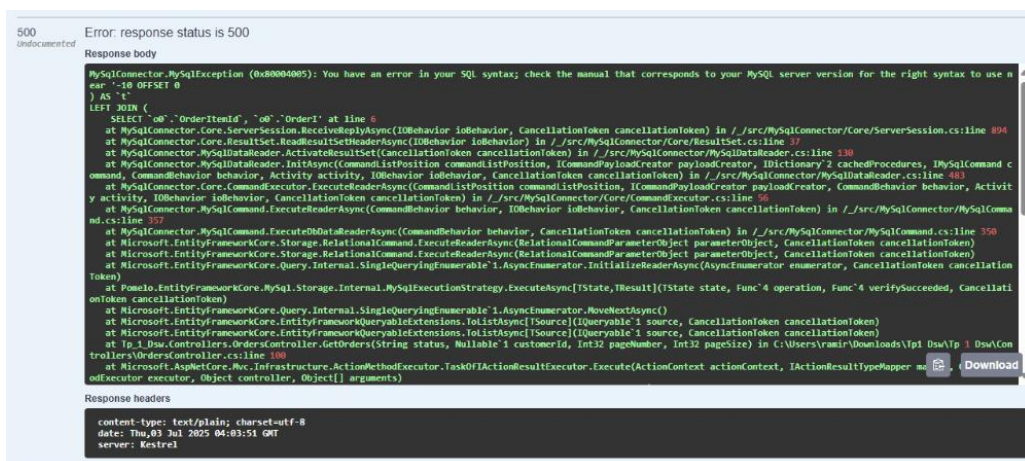
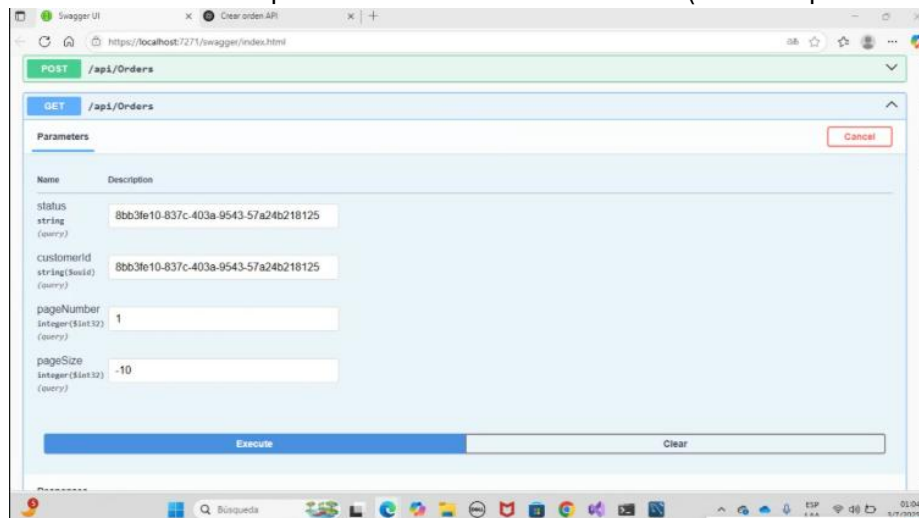
## 5.Verificaciones

- La respuesta contiene un arreglo de objetos representando órdenes válidas.
- Cada objeto incluye los campos esenciales: orderId, customerId, orderDate, orderStatus, totalAmount, shippingAddress y billingAddress.
- Si se pasan parámetros de filtrado (status, customerId), todas las órdenes devueltas cumplen con dichos filtros.
- La cantidad de elementos devueltos no excede el pageSize especificado.
- Si no hay órdenes en la base o no hay coincidencias con los filtros, se devuelve un arreglo vacío (no error).
- El formato de fecha (orderDate) y los montos (totalAmount) son consistentes y correctos.

### Caso de error 500

#### 1.Precondiciones

- El backend tiene un error activo, como:
- Fallo en la conexión a la base de datos.
- Migraciones no aplicadas correctamente.
- Lógica con errores en el controlador o servicio correspondiente.
- Error no manejado (ej. NullReferenceException, SQLException, etc.).
- No se implementó correctamente el manejo de excepciones globales.
- El cliente realiza una petición correctamente formada (no es culpa del cliente).





## 2.Verificaciones

- El backend responde con el código 500 ante errores no controlados.
- El mensaje de error es genérico y no expone detalles internos del sistema (por seguridad).
- Se registra el detalle del error en el log del servidor para su análisis por parte del equipo de desarrollo.
- El error no se debe a una mala solicitud del cliente (el input es correcto).
- Si hay un middleware de manejo de errores (ej. `ExceptionHandler` o `ExceptionHandlerMiddleware`), se asegura que capture y formatee apropiadamente la excepción.

## 3. Obtener una orden por ID:

- **Método HTTP:** GET
- **Ruta:** `/api/orders/{id}`
- **Descripción:** Retorna los detalles completos de una orden específica, incluyendo todos sus ítems.
- **Respuesta Exitosa:** 200 OK con el objeto de la orden solicitada.
- **Respuesta de Error:** 404 Not Found si no se encuentra una orden con el ID proporcionado.

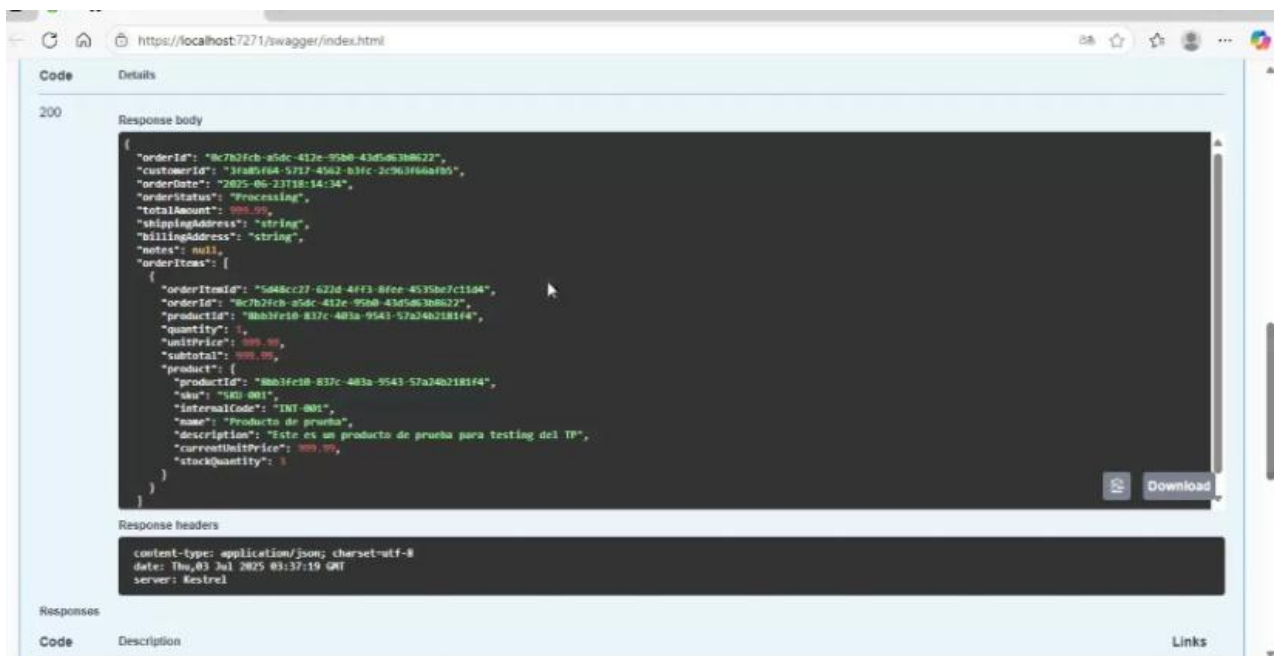
## 4.Precondiciones

- El `orderId` especificado existe en la base de datos.
- La orden correspondiente tiene ítems asociados en la tabla `OrderItems`.
- La conexión a la base de datos está activa y la consulta no produce errores.
- El ID está en un formato válido de GUID.

## Caso de éxito del tercer endpoint

The screenshot displays the Swagger UI for a web application. The top navigation bar includes links for 'Administration' and 'Schemas'. The main content area shows a 'Result Grid' with a table of orders. The first row is selected, showing details for an order with ID '0c7b2fcb-a5dc-412e-95b0-43d5d63b8622'. The table columns include OrderId, CustomerId, OrderDate, OrderStatus, TotalAmount, ShippingAddress, BillingAddress, and Notes. The order status is 'Processing' and the total amount is '999.99'. Below the table, there is a 'Parameters' section for the GET endpoint `/api/orders/{id}`. The 'id' parameter is marked as required and has a value of '0c7b2fcb-a5dc-412e-95b0-43d5d63b8622' entered. The 'Execute' button is highlighted in blue, indicating a successful request. The 'Responses' section is visible at the bottom.

OrderId	CustomerId	OrderDate	OrderStatus	TotalAmount	ShippingAddress	BillingAddress	Notes
0c7b2fcb-a5dc-412e-95b0-43d5d63b8622	3fa85041-3717-4662-b3fc-2c963f66afa9	2025-06-23 18:14:34	Processing	999.99	string	string	
48109f7e-4cf2-4115-a61d-bdc7992178be	3fa85041-3717-4662-b3fc-2c963f66afa9	2025-07-03 00:23:15	Pending	999.99	Casa central	Casa central	
58139ebc-d25e-4252-9649-7c4e032efb0e	11111111-1111-1111-1111-111111111111	2025-06-20 17:41:40	Pending	999.99	Av. Siempre Viva 742	Av. Siempre Viva 742	
89608344-e933-42b9-8004-c0cb8560640d	b0b0b1f8-97e4-465f-6276-d40ab2c75ef6	2025-07-01 12:37:57	Pending	9999.99	string	string	
8d15495d-000f-48ec-baf4-94d6475ea4e3	6f9d5d1e-3e74-4a3e-bd85-9d90c7880e51	2025-07-01 12:08:33	Pending	11999.98	Calle Falsa 123	Calle Verdadera 456	
bb90f5dd-b9e6-4902-830e-4de86bc87e55	11111111-1111-1111-1111-111111111111	2025-06-20 18:02:33	Pending	999.99	Av. Siempre Viva 742	Av. Siempre Viva 742	
dbb1155a-b2c8-4fd4-bc24-81550878d47c	8bb3fe10-837c-403a-9543-5762462181f4	2025-06-30 23:54:02	Pending	999.99	Juan X	Juan	



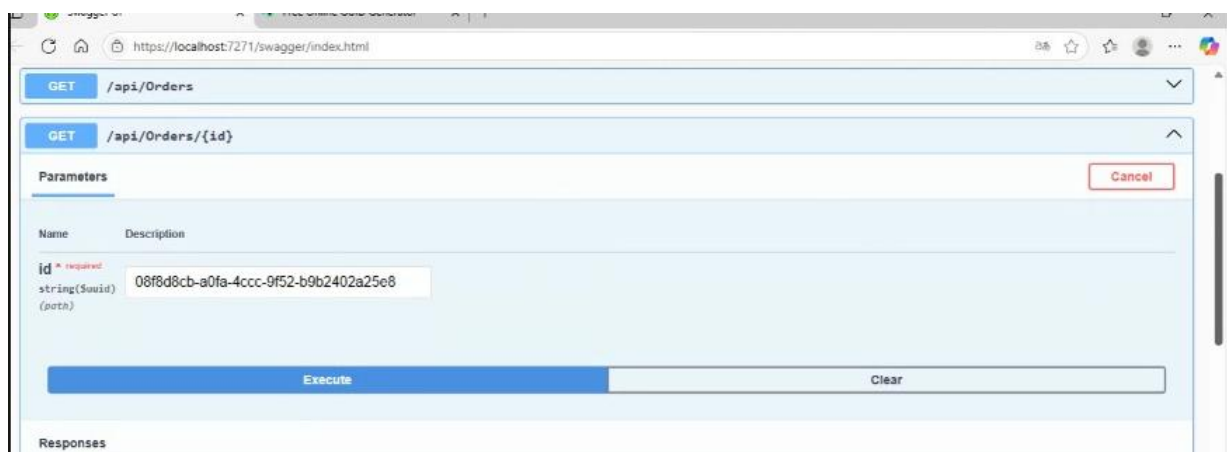
## 1.Verificaciones

- La respuesta contiene el objeto completo de la orden con todos sus campos esperados.
- Se incluyen correctamente todos los ítems relacionados con la orden (orderItems).
- El totalAmount corresponde a la suma de los subtotales de los ítems.
- El formato del campo orderDate es correcto (ISO 8601).
- Si el ID corresponde a una orden válida, se garantiza una respuesta 200 OK.

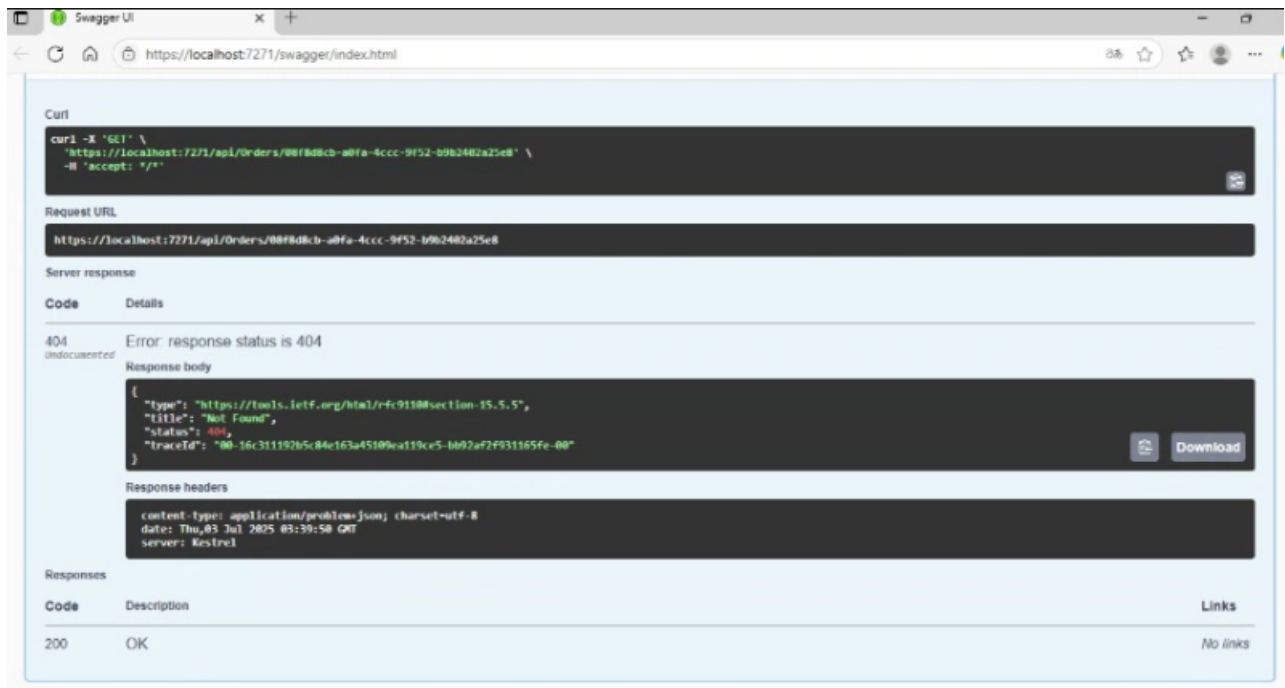
## 2.Precondiciones

- El orderId especificado **no existe** en la tabla Orders.
- El GUID está correctamente formado (sintaxis válida), pero no coincide con ningún registro.
- El backend tiene implementado el control de errores para retornar 404 si no encuentra la entidad.
- La base de datos está funcionando correctamente y no hay errores de conexión.

### Caso de error del tercer endpoint







## 1.Verificaciones

- El sistema no lanza excepciones ni errores internos al no encontrar la orden.
- El código HTTP de la respuesta es exactamente 404.
- El mensaje devuelto es claro y no expone detalles técnicos innecesarios.
- Si el orderId es inválido (por formato), se debe retornar 400 Bad Request en lugar de 404

## 2. Actualizar el estado de una orden

- **Método HTTP:** PUT
- **Ruta:** /api/orders/{id}/status
- **Descripción:** Permite cambiar el estado de una orden existente. Este endpoint debe ser idempotente y solo modificar el estado de la orden.

## 3.Precondiciones

- El orderId especificado corresponde a una orden existente en la base de datos.
- El nuevo estado (newStatus) es válido dentro del ciclo de vida de una orden (Pending, Processing, Shipped, Delivered, Cancelled).
- La transición de estado es lógica y permitida (por ejemplo, de Pending a Processing sí, pero no de Delivered a Pending).
- El formato del cuerpo JSON es correcto.
- El sistema tiene correctamente implementada la lógica de actualización e idempotencia (si se reenvía la misma solicitud con el mismo estado, no cambia el resultado).

## Caso de éxito del tercer endpoint

The screenshot shows a REST client interface with a 'Parameters' tab. A required parameter 'id' of type 'string(\$uuid)' with the value '0c7b2fcb-a5dc-412e-95b0-43d5d63b8622' is entered. The 'Request body' is set to 'application/json' and contains a JSON object: 

```
{  "newStatus": "Pending"}
```

The screenshot shows the 'Curl' tab with the following command: 

```
curl -X 'PUT' \
  'https://localhost:7271/api/Orders/0c7b2fcb-a5dc-412e-95b0-43d5d63b8622/status' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "newStatus": "Pending"
  }'
```

The 'Request URL' is `https://localhost:7271/api/Orders/0c7b2fcb-a5dc-412e-95b0-43d5d63b8622/status`.

The 'Server response' shows a 200 status code. The 'Response body' is a JSON object: 

```
{  "orderId": "0c7b2fcb-a5dc-412e-95b0-43d5d63b8622",  "customerId": "3fa85f64-5717-4562-b3fc-2c963f66afb5",  "orderDate": "2025-06-23T18:14:34",  "orderStatus": "Pending",  "totalAmount": 999.99,  "shippingAddress": "string",  "billingAddress": "string",  "notes": null,  "orderItems": []}
```

The 'Response headers' are: 

```
content-type: application/json; charset=utf-8
date: Thu, 03 Jul 2025 03:42:27 GMT
server: Kestrel
```

### 1.Verificaciones

- El estado de la orden se actualiza correctamente en la base de datos.
- La respuesta devuelve el orderId y el nuevo orderStatus.
- La respuesta incluye un mensaje confirmando la acción.
- Si se repite la misma solicitud con el mismo estado, la respuesta sigue siendo 200 OK (comportamiento idempotente).
- No se modifica ningún otro campo de la orden (solo el estado).

## Caso de error 404

### 1.Precondiciones

- El orderId especificado **no existe** en la base de datos.
- El GUID tiene un formato válido, pero no coincide con ningún registro en la tabla Orders.
- El backend implementa un mecanismo para verificar la existencia de la orden antes de intentar actualizar.
- La base de datos está operativa y no hay errores de conexión.

PUT /api/Orders/{id}/status

Parameters

Name	Description
id * required string(\$uuid) (path)	a7b9663e-a4d7-4959-ad03-07d9701ecbb2

Request body

application/json

```
{  "newStatus": "Pending"}
```

Curl

```
curl -X 'PUT' \
  'https://localhost:7271/api/Orders/a7b9663e-a4d7-4959-ad03-07d9701ecbb2/status' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
    "newStatus": "Pending"
  }'
```

Request URL

https://localhost:7271/api/Orders/a7b9663e-a4d7-4959-ad03-07d9701ecbb2/status

Server response

Code	Details
404 Unauthenticated	Error: response status is 404

Response body

```
Orden con ID a7b9663e-a4d7-4959-ad03-07d9701ecbb2 no encontrada.
```

Response headers

```
content-type: text/plain; charset=utf-8
date: Thu, 03 Jul 2025 03:44:36 GMT
server: Kestrel
```

Responses

Code	Description	Links
200	OK	No links

### 2.Verificaciones

- El sistema no intenta modificar registros si la orden no existe.
- Se devuelve estrictamente un código 404 ante esta situación.
- El mensaje de error es claro y no expone información interna.
- La validación de existencia ocurre antes de intentar cualquier lógica de negocio.

## Caso de error 400

### 1.Precondiciones

- El orderId especificado **sí existe**.
- El nuevo estado (newStatus) no forma parte de los estados válidos permitidos (Pending, Processing, Shipped, Delivered, Cancelled) o
- La transición desde el estado actual al nuevo estado no está permitida por la lógica del negocio.
- El formato del JSON es válido (no hay error de sintaxis).

The screenshot shows a REST client interface with a table for parameters:

Name	Description
id * required string(50id) (path)	0c7b2fcb-a5dc-412e-95b0-43d5d0b8622

Below the table, the 'Request body' is set to 'application/json' and contains the following JSON:

```
{  "newStatus": "Pending"}
```

The screenshot shows the 'Responses' section of the REST client. It displays the curl command used for the request:

```
curl -X PUT \  'https://localhost:7271/api/orders/0c7b2fcb-a5dc-412e-95b0-43d5d0b8622/status' \  -H 'accept: */*' \  -H 'Content-Type: application/json' \  -d '{  "newStatus": "Pending"  }'
```

The 'Request URL' is `https://localhost:7271/api/orders/0c7b2fcb-a5dc-412e-95b0-43d5d0b8622/status`.

The 'Server response' section shows a 400 status code with the message: 'Error: response status is 400'.

The 'Response body' is displayed in a dark box with the following text: 'Estado inválido. Estados permitidos: Pending, Processing, Shipped, Delivered, Cancelled'.

The 'Response headers' section shows the following headers: 'content-type: text/plain; charset=utf-8', 'date: Thu, 03 Jul 2025 03:49:35 GMT', and 'server: Kestrel'.

### 2.Verificaciones

- El sistema valida que el nuevo estado esté dentro de los valores permitidos.
- Se controla la lógica de transición de estados (ej. no volver de Delivered a Processing).
- El mensaje de error informa explícitamente el motivo del rechazo.
- El estado de la orden **no se modifica** en ningún caso.
- La respuesta devuelve un 400 en todos los escenarios inválidos de estado.