

# Manual de uso de la API REST

## 1. Diseño del sistema y decisiones arquitectónicas

Durante el desarrollo de esta API RESTful, se tomaron decisiones clave para garantizar una arquitectura modular, escalable y mantenible. A continuación, se describen las principales elecciones de diseño y sus motivaciones:

### b. Framework y estructura del proyecto

El servidor está construido sobre el framework Express.js, complementado con una arquitectura basada en controladores, servicios y repositorios. Se implementó un sistema de decoradores personalizados (@Controller, @Route, @VerifyToken, @authorize) para estructurar las rutas de forma declarativa y mantener el código más limpio y legible.

Esta arquitectura permite:

- Separación clara de responsabilidades (controlador vs lógica de negocio vs acceso a datos).
- Escalabilidad modular al agregar nuevas entidades o funcionalidades.
- Reutilización de componentes como middlewares, validaciones y servicios.

### c. Sistema de base de datos

Se optó por PostgreSQL como sistema de base de datos relacional por su solidez y confiabilidad, así como por su excelente soporte para garantizar la integridad de los datos a través de restricciones referenciales.

### d. ORM y acceso a datos

Se utilizó Prisma ORM por su estrecha integración con TypeScript, lo que permite un tipado fuerte y autocompletado en las consultas. Además, ofrece herramientas para generar clientes de base de datos de forma automática y gestionar migraciones de manera sencilla.

La elección de Prisma también respondió a un reto personal, ya que no lo había utilizado previamente y representaba una oportunidad de aprendizaje práctico dentro del proyecto.

### e. Validación y DTOs

Para validar la entrada del usuario se empleó class-validator junto con DTOs definidos por clase. Esto permite aplicar decoradores como @IsUUID,

@IsOptional, @IsDateString, entre otros, asegurando que la información ingresada cumpla con el formato esperado.

## **f. Seguridad y control de acceso**

Se incorporó autenticación mediante JWT (JSON Web Tokens) y un sistema de roles (user, seller, admin) que permite aplicar autorización basada en roles en cada endpoint mediante decoradores. Esto refuerza la seguridad y restringe el acceso a funcionalidades sensibles.

## **2. API REST**

### **2.1. Autenticación y Autorización**

La API requiere autenticación mediante tokens JWT. Cada solicitud debe incluir un token en el encabezado de autorización de la siguiente manera:

Authorization: Bearer <token>

Los usuarios deben tener roles específicos para acceder a ciertos endpoints. Los roles se definen como parte del payload del token JWT.

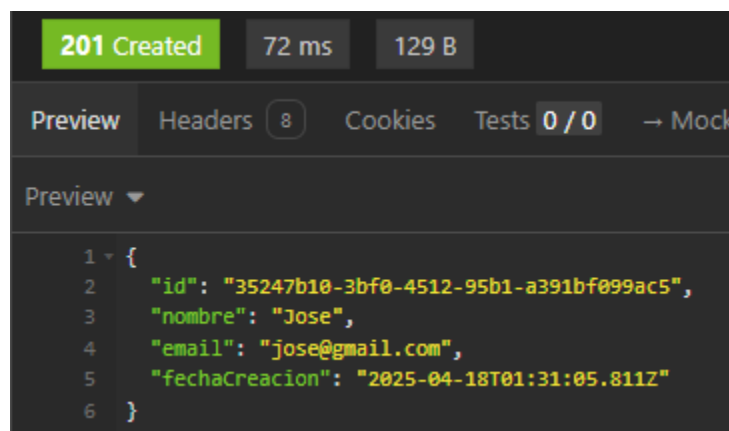
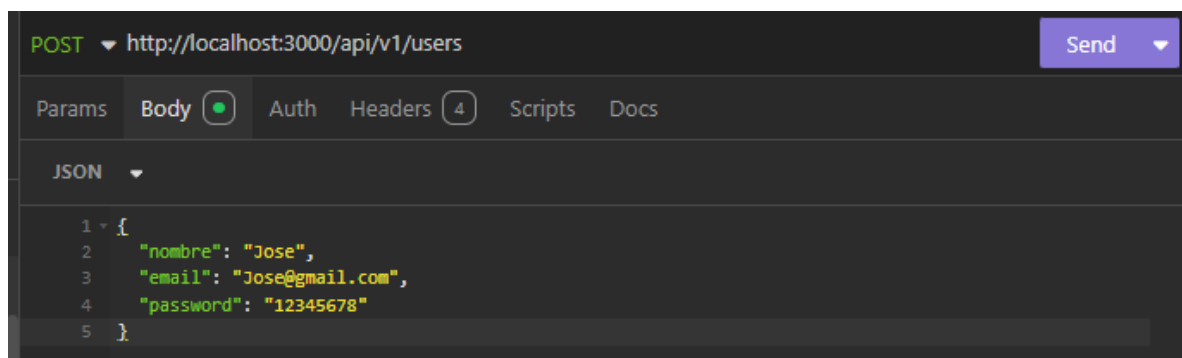
## 2.2. Endpoints

### Post /api/v1/users

- Descripción: Crea un nuevo usuario en el sistema. Se debe enviar un cuerpo con los datos del nuevo usuario.
- Autenticación: No requiere autenticación.
- Cuerpo de la solicitud (JSON):

```
{
  "nombre": "John Doe",
  "email": "johndoe@example.com",
  "password": "secreto123"
}
```

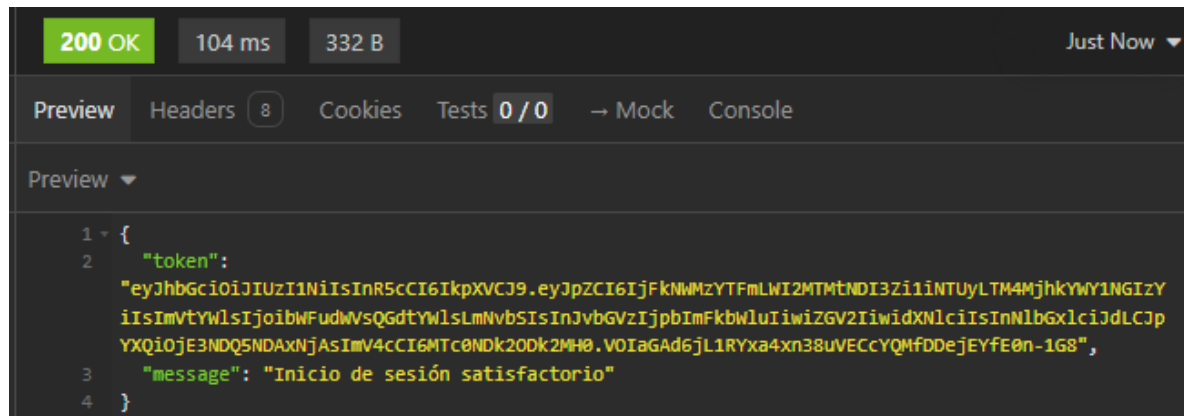
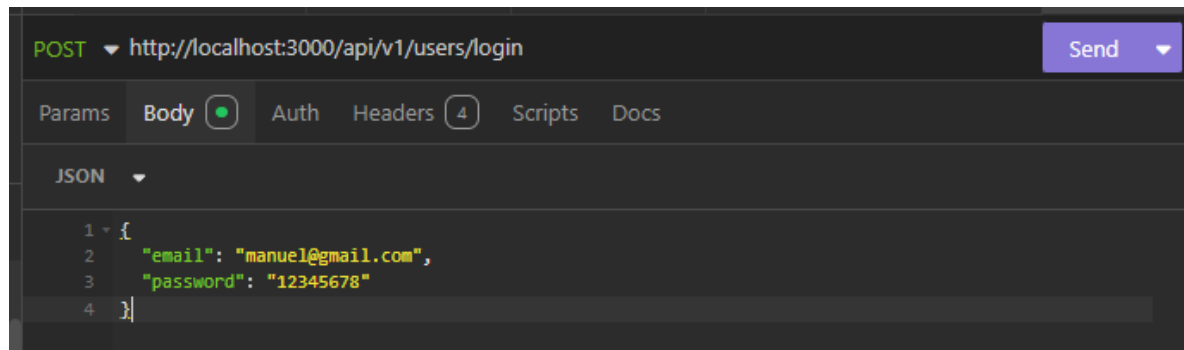
Ejemplo:



## Post /api/v1/users/login

- Descripción: Inicia sesión con las credenciales de usuario. Retorna un token de autenticación si las credenciales son correctas.
- Autenticación: No requiere autenticación.
- Cuerpo de la solicitud (JSON):

Ejemplo:



## Put /api/v1/users/:id

- Método: PUT
- Ruta: /api/v1/users/:id
- Descripción: Actualiza los datos de un usuario existente. Requiere que el usuario esté autenticado.
- Autenticación: Requiere token JWT válido.
- Cuerpo de la solicitud (JSON):

Ejemplo:

The image shows a REST client interface with two panels. The top panel displays a PUT request to the URL `http://localhost:3000/api/v1/users/1d5c3a1f-b613-427f-b552-3828daf54b3b`. The request body is a JSON object: `{ "nombre": "Pedro" }`. The bottom panel shows the response status `201 Created` with a response time of `14 ms` and a response size of `132 B`. The response body is a JSON object: `{ "id": "1d5c3a1f-b613-427f-b552-3828daf54b3b", "nombre": "Pedro", "email": "manuel@gmail.com", "fechaCreacion": "2025-04-17T06:08:41.536Z" }`.

**PUT** `http://localhost:3000/api/v1/users/1d5c3a1f-b613-427f-b552-3828daf54b3b` **Send**

Params Body Auth Headers 3 Scripts Docs

JSON

```
1 {  
2   "nombre": "Pedro"  
3 }
```

Params Body Auth Headers 3 Scripts Docs

Bearer Token

ENABLED ☒

TOKEN `.....`

PREFIX

**201 Created** 14 ms 132 B

Preview Headers 8 Cookies Tests 0 / 0 → Mock

Preview

```
1 {  
2   "id": "1d5c3a1f-b613-427f-b552-3828daf54b3b",  
3   "nombre": "Pedro",  
4   "email": "manuel@gmail.com",  
5   "fechaCreacion": "2025-04-17T06:08:41.536Z"  
6 }
```

## Get /api/v1/users/table

- Descripción: Obtiene una lista de usuarios con opciones de paginación. Solo accesible para usuarios con rol de administrador, con filtros por nombre y correo por parámetros query.
- Autenticación: Se requiere un token de autenticación válido
- Roles permitidos: admin
- Parámetros de consulta:
  - page: Página de resultados (opcional, predeterminado: 1)
  - size: Número de resultados por página (opcional, predeterminado: 10)
  - nombre: filtro por nombre
  - email: filtro por correo
  - Ejemplo de URL: /api/v1/users/table?page=2&size=5

Ejemplo:

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

- Method and URL:** GET `http://localhost:3000/api/v1/users/table`
- Buttons:** Send, Params (3), Body, Auth, Headers (3), Scripts, Docs
- URL Preview:** `http://localhost:3000/api/v1/users/table?page=1&size=10&email=%40`
- Query Parameters Section:**
  - Buttons: + Add, Delete all, Description
  - Table of parameters:

QUERY PARAMETERS			
page	1		
size	10		
nombre	pe		
email	@		

```

{
  "data": [
    {
      "id": "1d5c3a1f-b613-427f-b552-3828daf54b3b",
      "nombre": "Pedro",
      "email": "manuel@gmail.com",
      "password": "$2b$10$cU0oCK5sVg0JUpwbAYhevOy9tqSnNonima40sZ3EKN3zo0gQlCBXq",
      "fechaCreacion": "2025-04-17T06:08:41.536Z"
    },
    {
      "id": "35247b10-3bf0-4512-95b1-a391bf099ac5",
      "nombre": "Jose",
      "email": "jose@gmail.com",
      "password": "$2b$10$akeyqq/oFBNTwwFGVfQ55.W.yiJfDAFbLAicYYSjqovqKRdlUxQzw",
      "fechaCreacion": "2025-04-18T01:31:05.811Z"
    },
    {
      "id": "904ccb56-2cee-443c-ae6d-c005a8b031df",
      "nombre": "Pedro",
      "email": "pedro@gmail.com",
      "password": "$2b$10$JiDOhFW5mrJ0jVXRcsmPR.dFLgxzc0/sE5v9SxYfbUqx0oke.qJN2",
      "fechaCreacion": "2025-04-18T01:28:53.324Z"
    }
  ],
  "count": 3
}

```

Filtro por email igual a "JO"

email JO

```

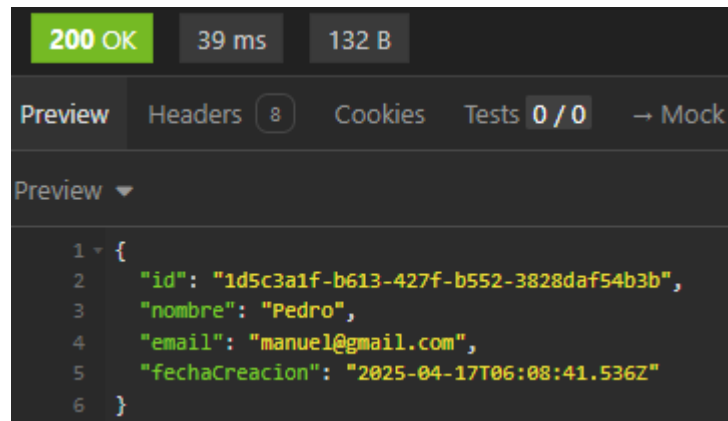
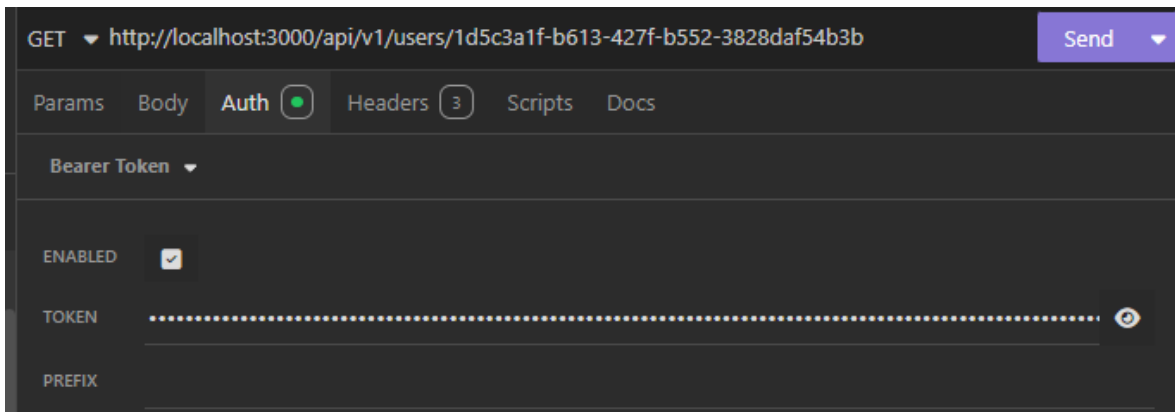
{
  "data": [
    {
      "id": "35247b10-3bf0-4512-95b1-a391bf099ac5",
      "nombre": "Jose",
      "email": "jose@gmail.com",
      "password": "$2b$10$akeyqq/oFBNTwwFGVfQ55.W.yiJfDAFbLAicYYSjqovqKRdlUxQzw",
      "fechaCreacion": "2025-04-18T01:31:05.811Z"
    }
  ],
  "count": 1
}

```

## Get /api/v1/users/:id

- Descripción: Recupera los detalles de un usuario por su ID. Requiere que el usuario esté autenticado.
- Autenticación: Se requiere un token de autenticación válido.

Ejemplo:





## Delete /api/v1/users/:id

- Descripción: Elimina un usuario por su ID. Solo accesible para usuarios con rol de administrador.
- Autenticación: Se requiere un token de autenticación válido
- Roles permitidos: admin

Ejemplo:

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

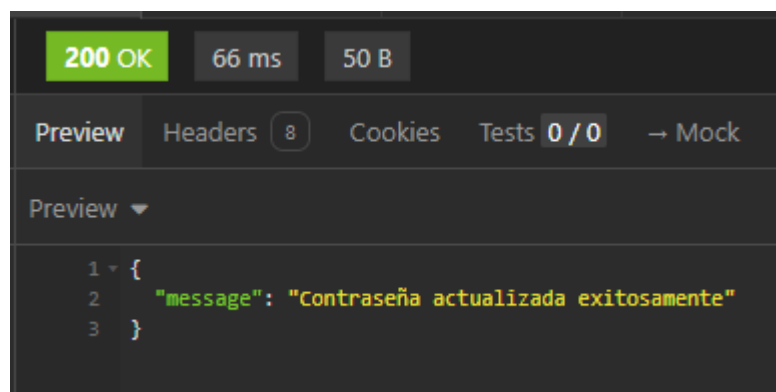
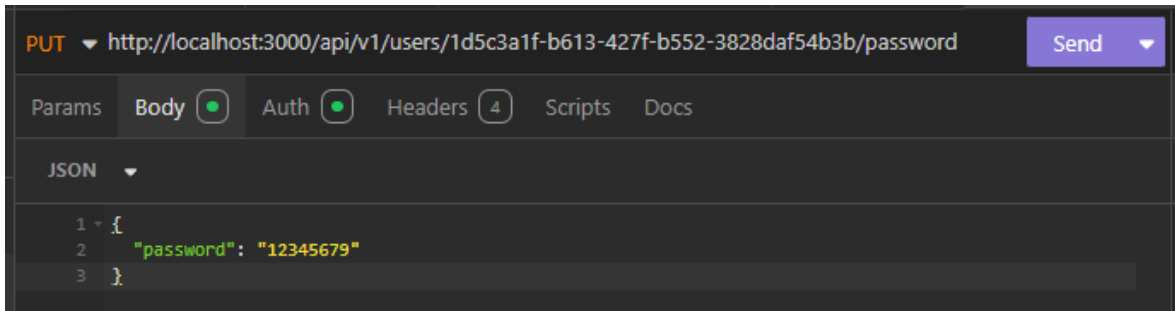
- Method:** DELETE
- URL:** http://localhost:3000/api/v1/users/904ccb56-2cee-443c-ae6d-c005a8b031df
- Buttons:** Params, Body, Auth (selected), Headers (3), Scripts, Docs
- Auth Section:**
  - Type: Bearer Token
  - ENABLED: ☒
  - TOKEN: A field containing a masked token (dots) with an eye icon to toggle visibility.
  - PREFIX: An empty text field.
- Action:** Send button

The screenshot shows the response section of the REST client with the following details:

- Status:** 204 No Content (highlighted in green)
- Time:** 17 ms
- Size:** 0 B
- Buttons:** Preview (selected), Headers (5), Cookies, Tests (0 / 0), → Mock, Console
- Preview Section:**
  - Preview dropdown
  - Text: No body returned for response

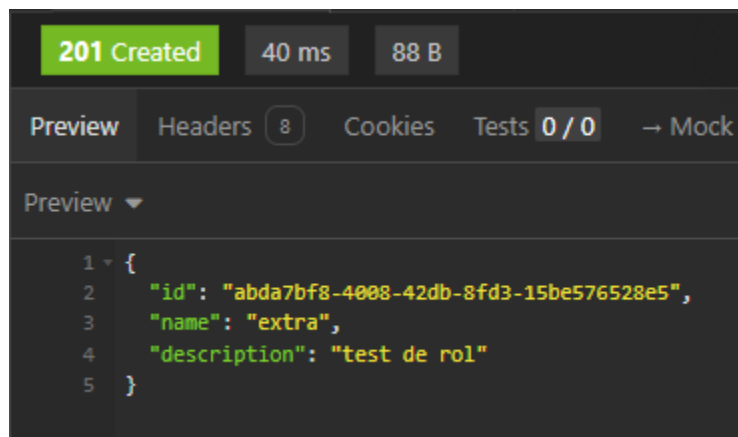
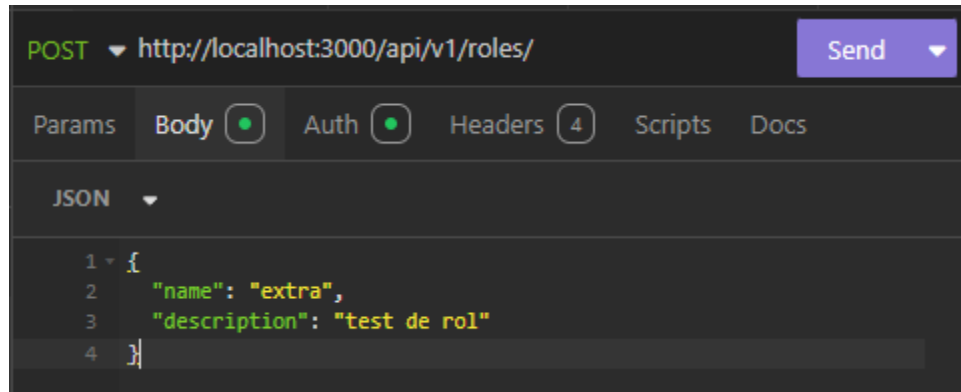
## Put /api/v1/users/:id/password

- Descripción: Cambia la contraseña de un usuario. Requiere que el usuario esté autenticado.
- Requisitos de Autenticación: Se requiere un token de autenticación válido.
- Cuerpo de la solicitud (JSON):



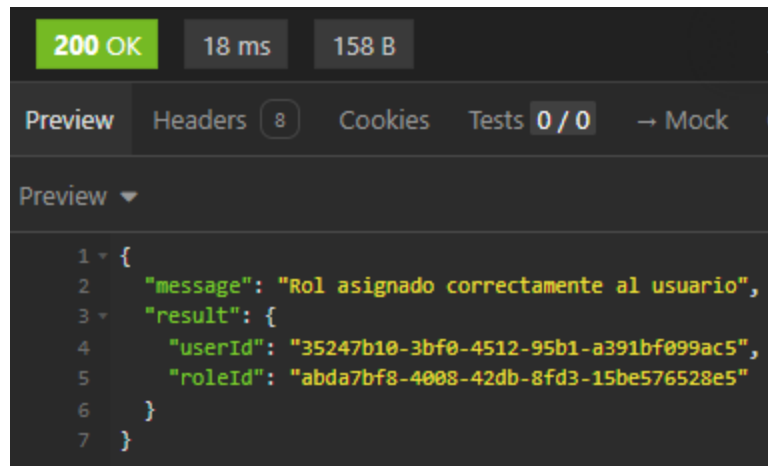
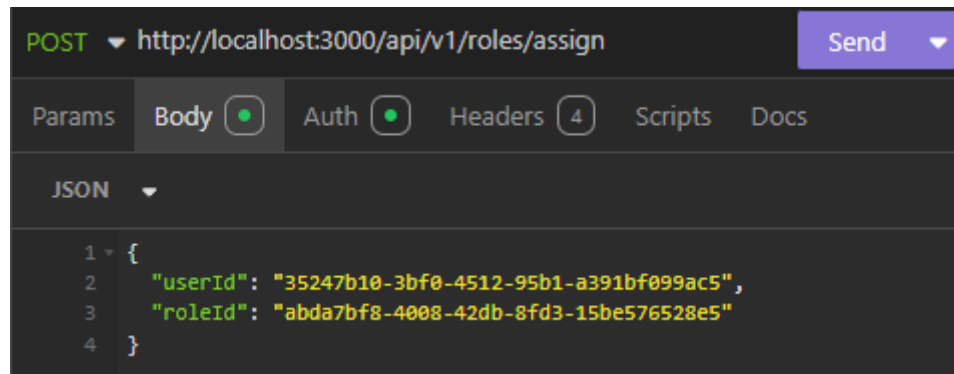
## Post /api/v1/roles

- Descripción: Crea un nuevo rol en el sistema.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: dev
- Cuerpo de la solicitud (JSON):



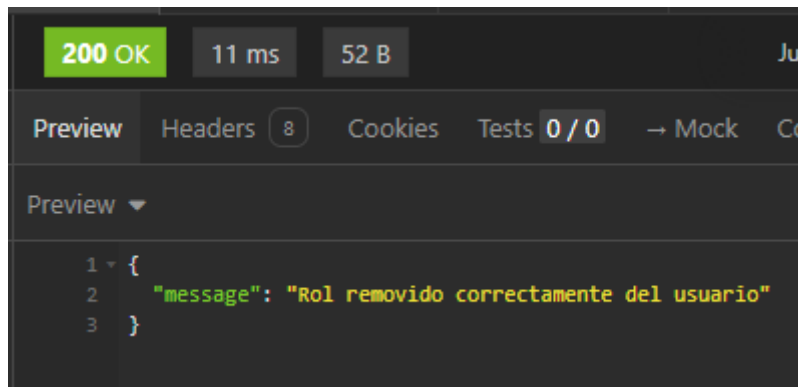
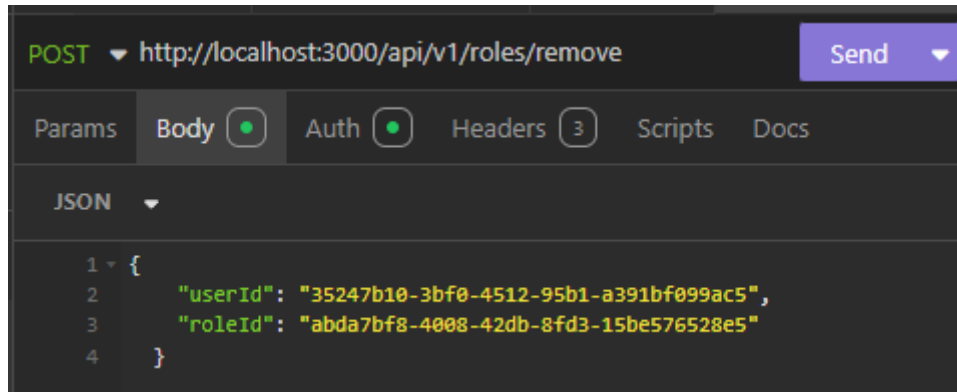
## Post /api/v1/roles/assign

- Descripción: Asigna un rol a un usuario existente.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: admin
- Cuerpo de la solicitud (JSON):



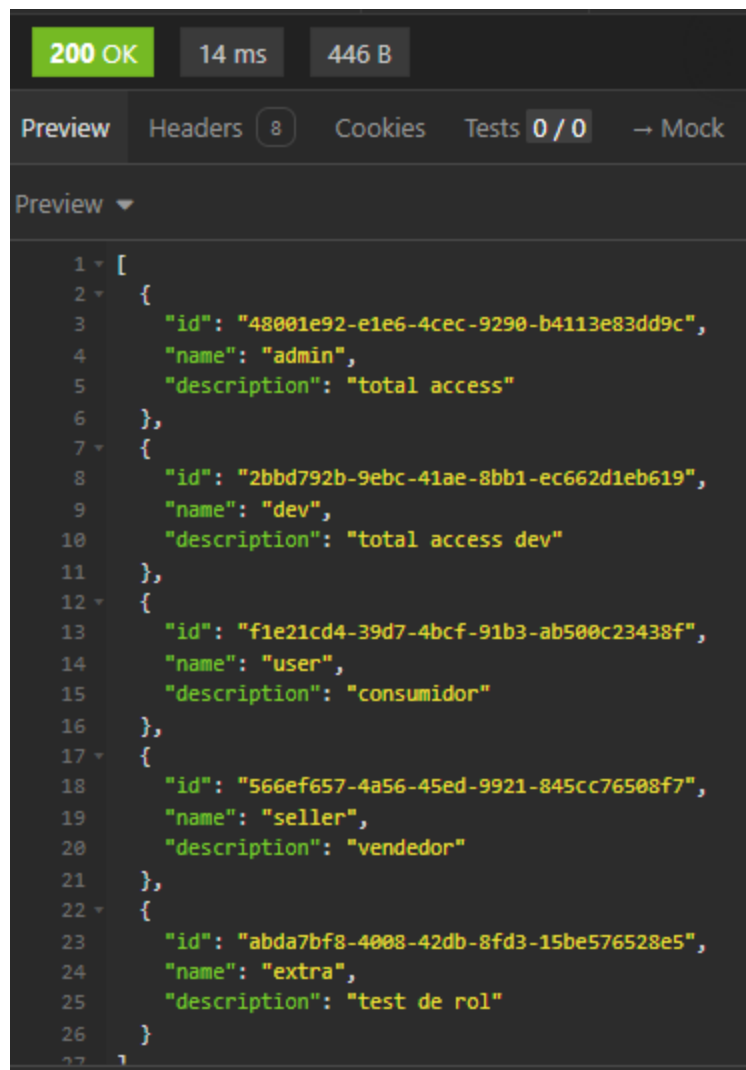
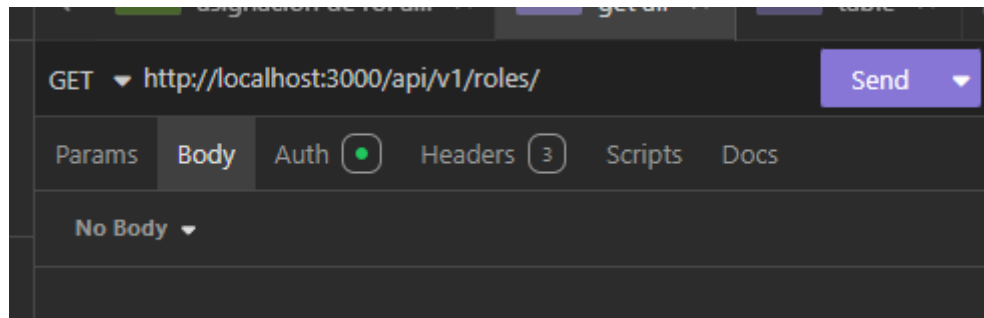
## Post /api/v1/roles/remove

- Descripción: Remueve un rol previamente asignado a un usuario.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: admin
- Cuerpo de la solicitud (JSON):



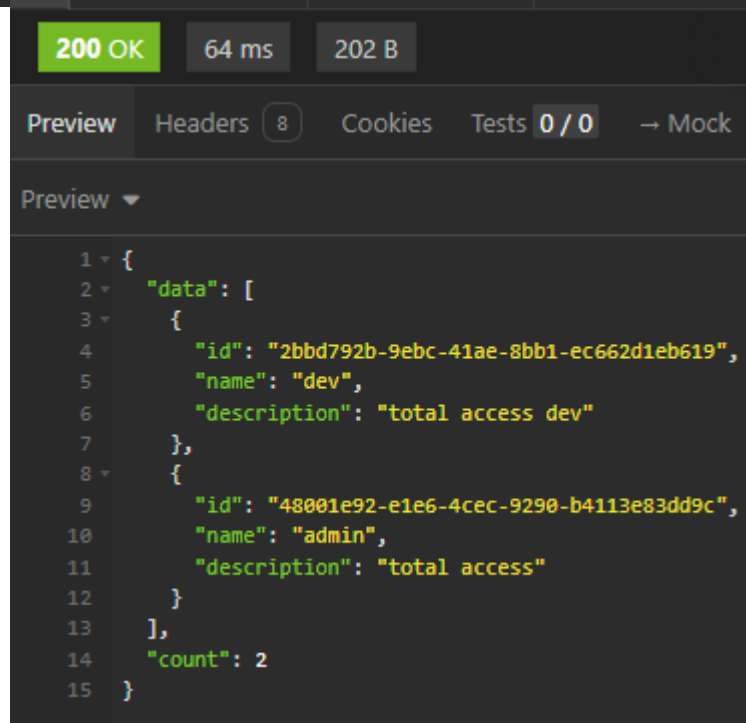
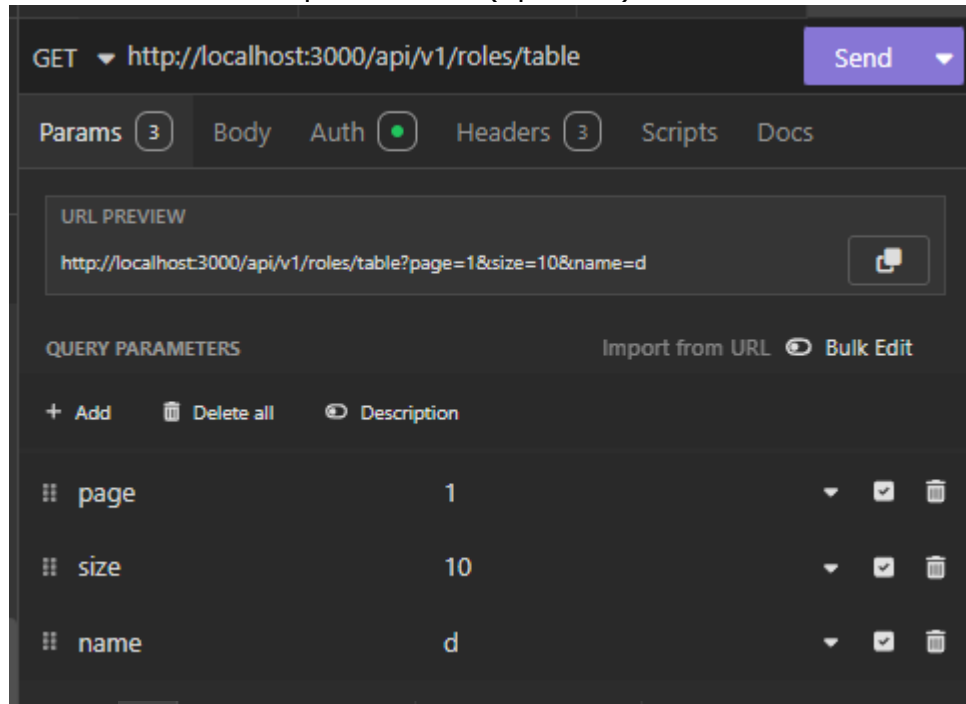
## Get /api/v1/roles

- Descripción: Obtiene la lista completa de roles disponibles en el sistema.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: admin
- Parámetros: Ninguno



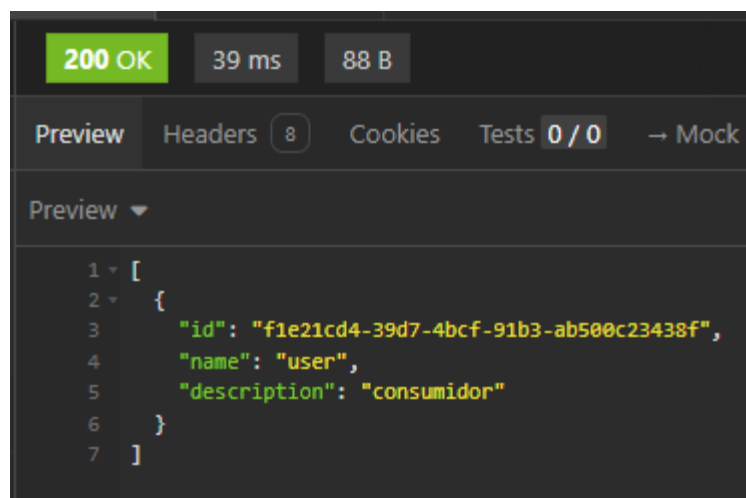
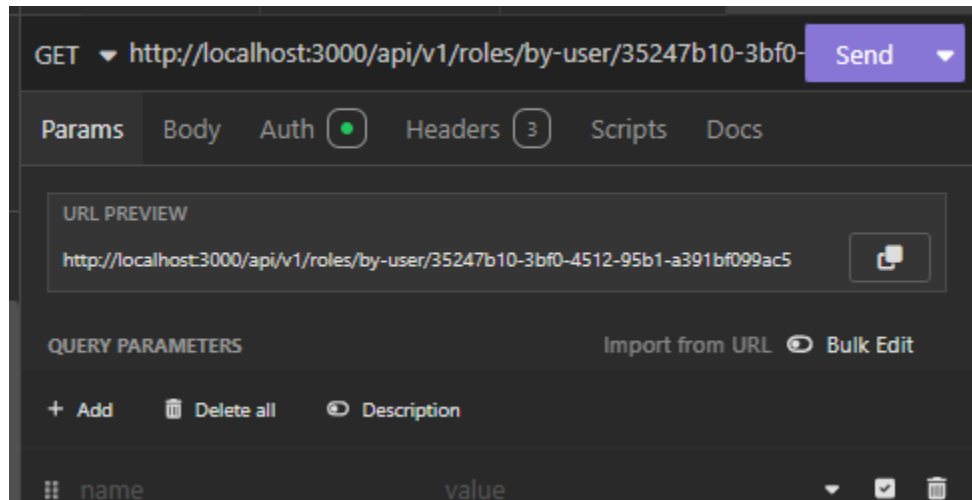
## Get /api/v1/roles/table

- Descripción: Obtiene una lista paginada de roles con opción de aplicar filtros.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: admin
- Parámetros de consulta:
  - page – Número de página (por defecto: 1)
  - size – Tamaño de página (por defecto: 10)
  - name – Filtro por nombre (opcional)



## Get /api/v1/roles/by-user/:userId

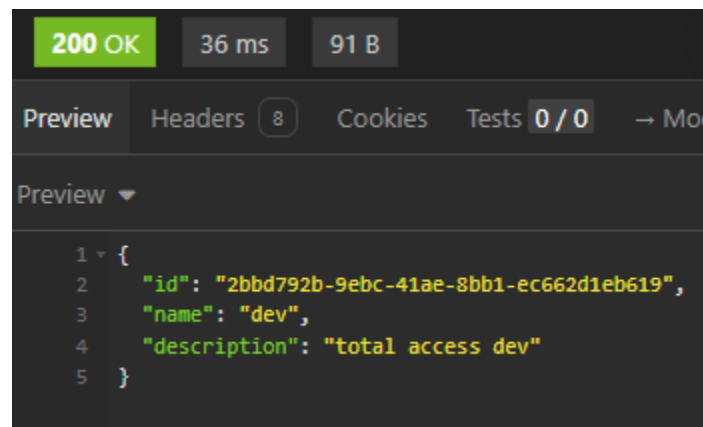
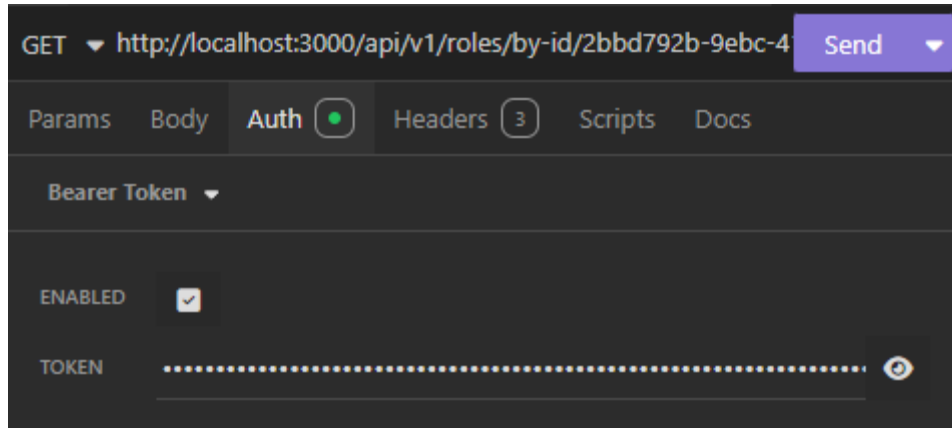
- Descripción: Obtiene todos los roles asignados a un usuario específico.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: admin
- Parámetros de ruta:
  - userId – UUID del usuario





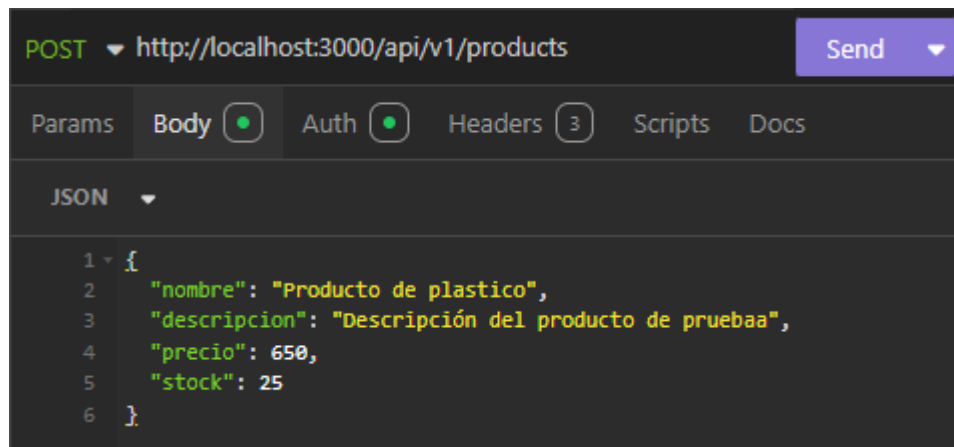
## Get /api/v1/roles/by-id/:id

- Descripción: Obtiene un rol específico por su ID.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: admin
- Parámetros de ruta:
  - id – UUID del rol



## Post /api/v1/products

- Descripción: Crea un nuevo producto y lo asocia al usuario autenticado con rol de vendedor.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: seller
- Cuerpo de la solicitud (JSON):

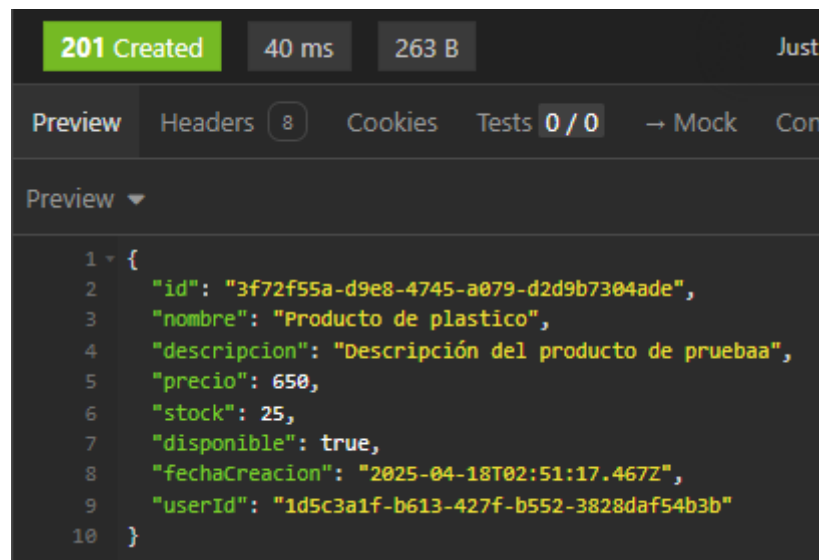


POST http://localhost:3000/api/v1/products

Params Body Auth Headers 3 Scripts Docs

JSON

```
1 {  
2   "nombre": "Producto de plastico",  
3   "descripcion": "Descripción del producto de pruebaa",  
4   "precio": 650,  
5   "stock": 25  
6 }
```



201 Created 40 ms 263 B Just

Preview Headers 8 Cookies Tests 0 / 0 → Mock Con

Preview

```
1 {  
2   "id": "3f72f55a-d9e8-4745-a079-d2d9b7304ade",  
3   "nombre": "Producto de plastico",  
4   "descripcion": "Descripción del producto de pruebaa",  
5   "precio": 650,  
6   "stock": 25,  
7   "disponible": true,  
8   "fechaCreacion": "2025-04-18T02:51:17.467Z",  
9   "userId": "1d5c3a1f-b613-427f-b552-3828daf54b3b"  
10 }
```

## Get /api/v1/products/table

- Descripción: Obtiene una lista paginada de productos con posibilidad de aplicar filtros dinámicos.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: user, seller
- Parámetros de consulta:
  - page – Número de página (por defecto: 1)
  - size – Tamaño de página (por defecto: 10)
  - Otros filtros dinámicos posibles como: "nombre", "descripcion", "disponible", "userId"

The screenshot displays a REST client interface with the following components:

- Request Bar:** Method: GET, URL: `http://localhost:3000/api/v1/products/table`, Send button.
- Params Tab:** 5 parameters listed in the QUERY PARAMETERS section.
- URL PREVIEW:** `http://localhost:3000/api/v1/products/table?page=1&size=10&userId=1d5c3a1f-b613-427f-b552-3828daf54b3b&disponible=true&descripcion=producto`
- QUERY PARAMETERS Table:**

+	Add	🗑️ Delete all	🔍 Description
⋮	page	1	▼ ✓ 🗑️
⋮	size	10	▼ ✓ 🗑️
⋮	userId	1d5c3a1f-b613-427f-b552-382	▼ ✓ 🗑️
⋮	disponible	true	▼ ✓ 🗑️
⋮	descripcion	producto	▼ ✓ 🗑️

**Response Status:** 200 OK, 40 ms, 808 B, Just Now

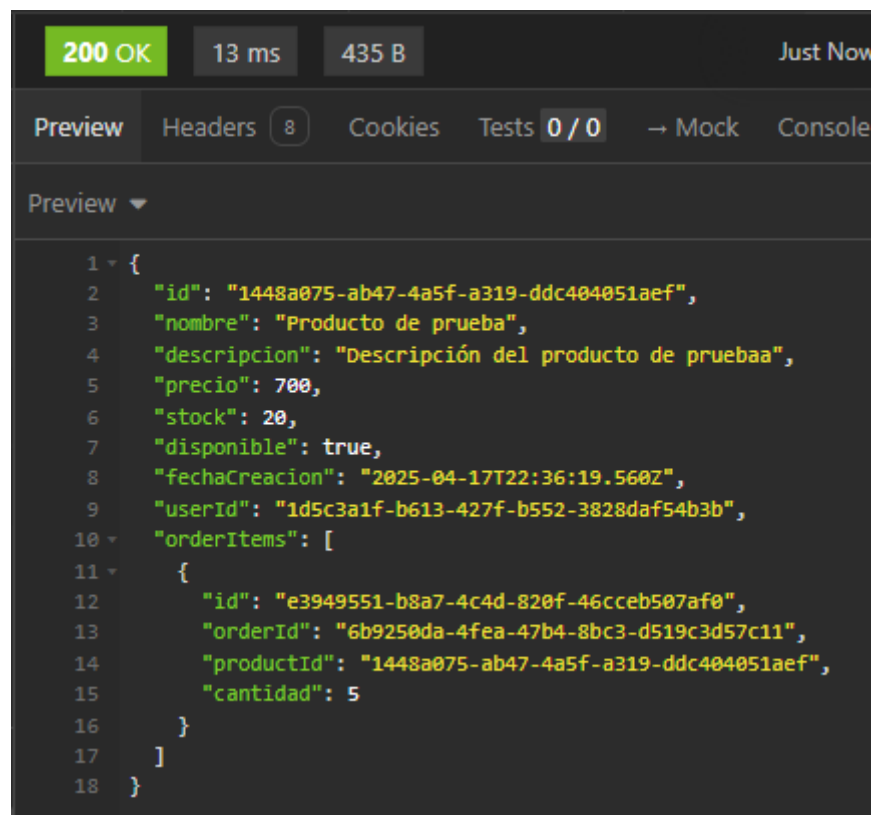
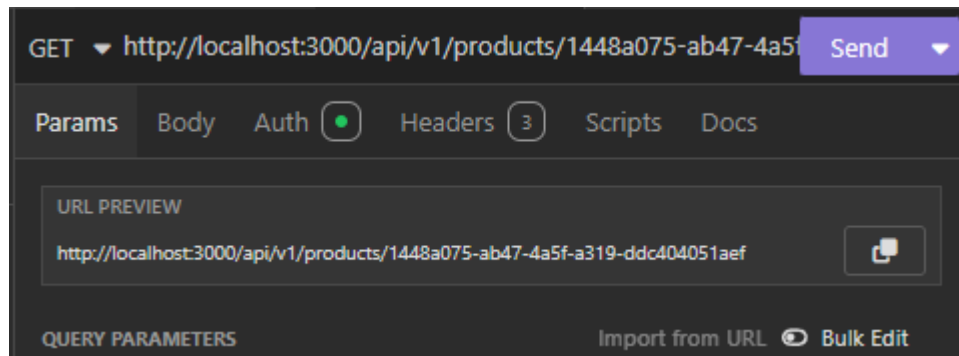
**Preview Tab:** Headers (8), Cookies, Tests (0/0), Mock, Console

**Preview:**

```
1 {
2   "data": [
3     {
4       "id": "1448a075-ab47-4a5f-a319-ddc404051aef",
5       "nombre": "Producto de prueba",
6       "descripcion": "Descripción del producto de pruebaa",
7       "precio": 700,
8       "stock": 20,
9       "disponible": true,
10      "fechaCreacion": "2025-04-17T22:36:19.560Z",
11      "userId": "1d5c3a1f-b613-427f-b552-3828daf54b3b"
12    },
13  ]
14 }
```

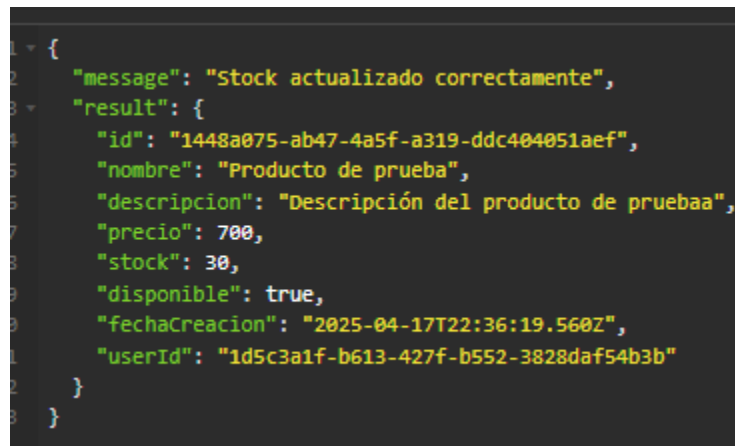
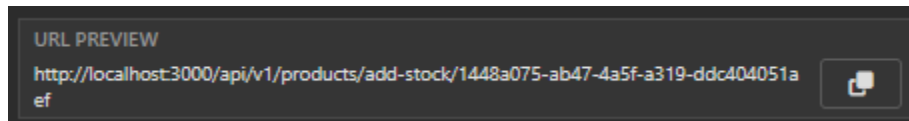
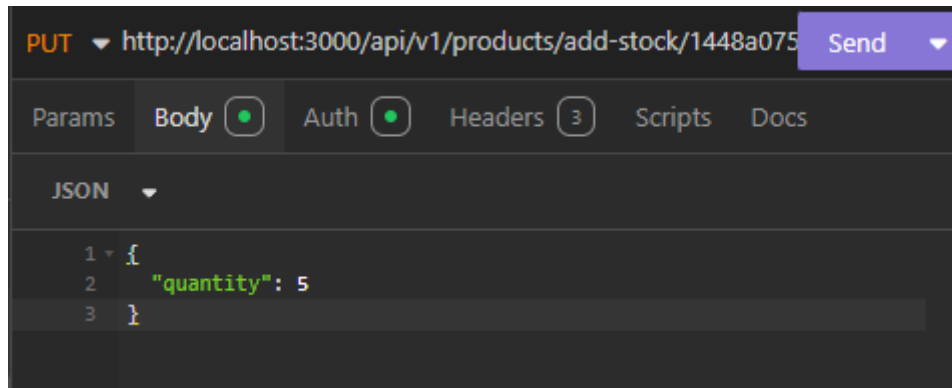
## Get /api/v1/products/:id

- Descripción: Obtiene los detalles de un producto específico por su ID.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: user, seller
- Parámetros de ruta:
  - id – UUID del producto



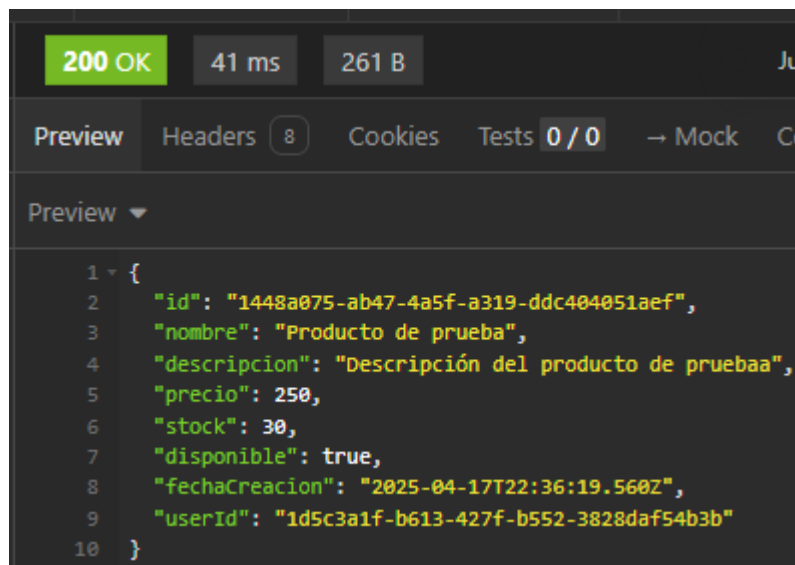
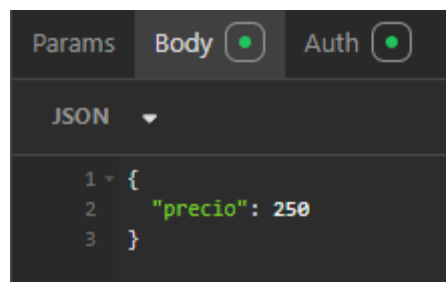
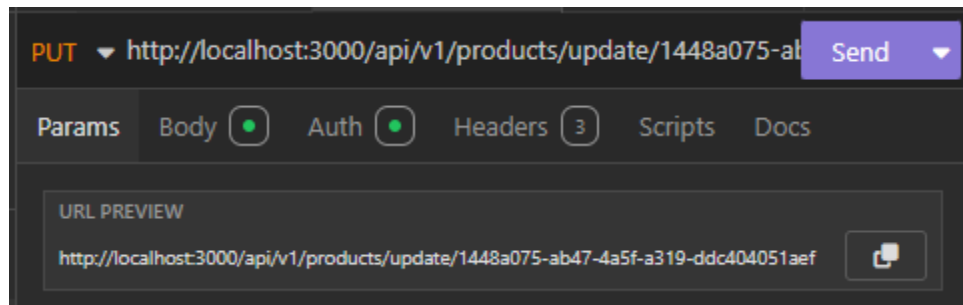
## Put /api/v1/products/add-stock/:id

- Descripción: Agrega una cantidad específica de stock a un producto existente.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: seller
- Parámetros de ruta:
  - id – UUID del producto



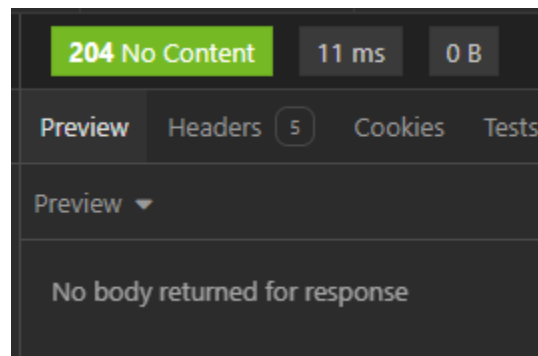
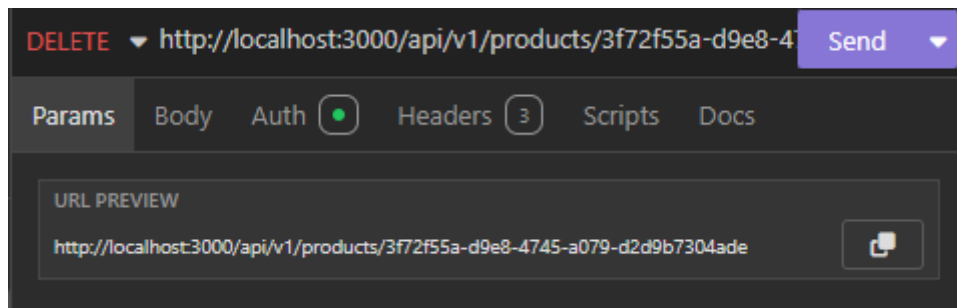
## Put /api/v1/products/update/:id

- Descripción: Actualiza los datos de un producto existente. Los campos son opcionales y se pueden enviar parcialmente.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: seller
- Parámetros de ruta:
  - id – UUID del producto



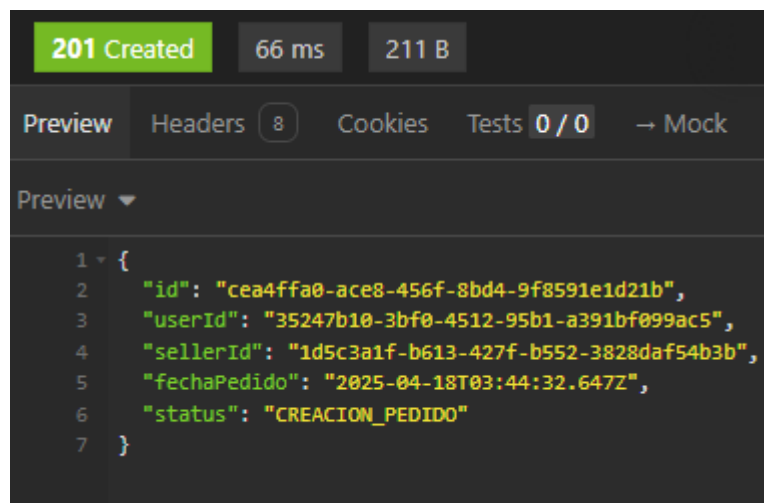
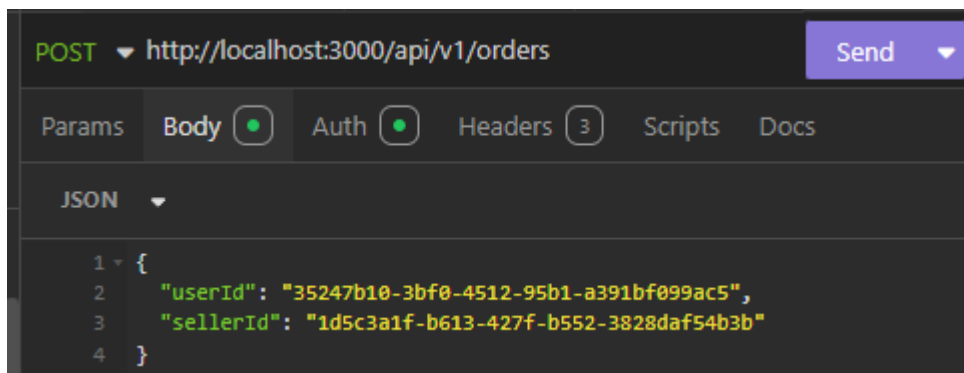
## Delete /api/v1/products/:id

- Descripción: Elimina un producto si cumple con las condiciones establecidas (e.g., si pertenece al usuario autenticado).
- Autenticación: Requiere token JWT válido.
- Roles permitidos: seller
- Parámetros de ruta:
  - id – UUID del producto



## Post /api/v1/orders

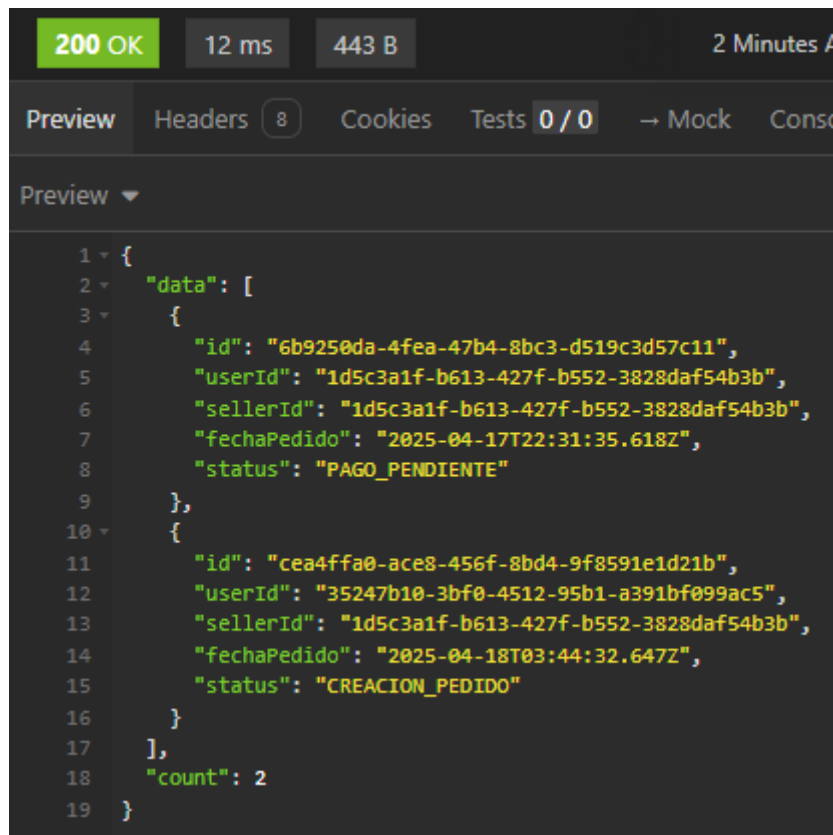
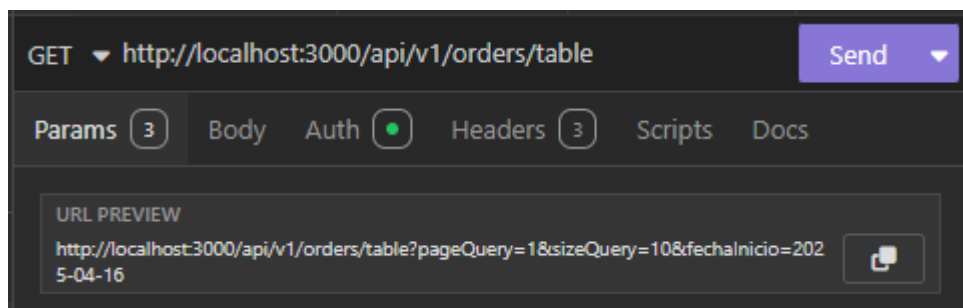
- Descripción: Crea una nueva orden para el usuario autenticado. Requiere proporcionar el ID del vendedor y opcionalmente la fecha del pedido.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: user
- Cuerpo de la solicitud (JSON):





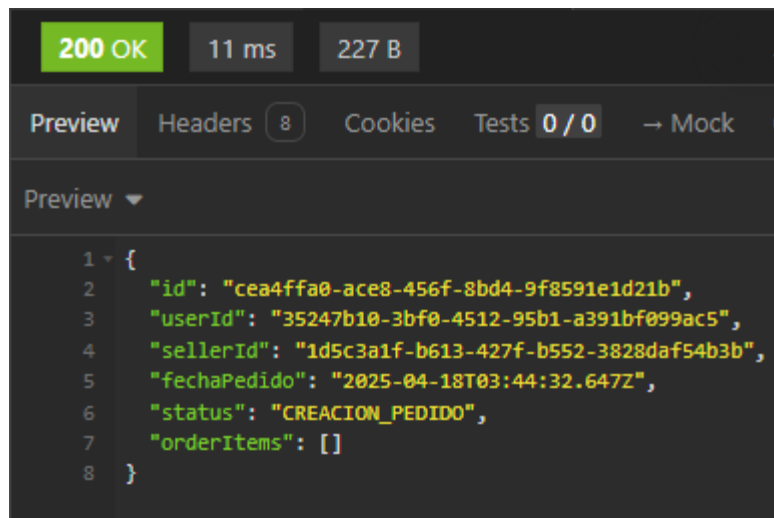
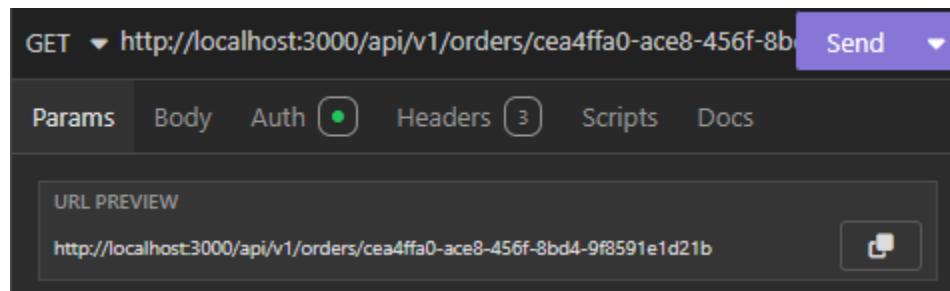
## Get /api/v1/orders/table

- Descripción: Retorna una lista paginada de órdenes, permitiendo aplicar filtros adicionales como fechas u otros campos.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: user, seller, admin
- Parámetros de consulta (opcional):
  - pageQuery – Número de página (por defecto: 1).
  - sizeQuery – Cantidad de resultados por página (por defecto: 10).
  - Otros filtros opcionales según la lógica de negocio ("userId", "status", "fechaInicio", "fechaFin").
  - Donde los campos fecha deben cumplir con el formato "YYYY-MM-DD"



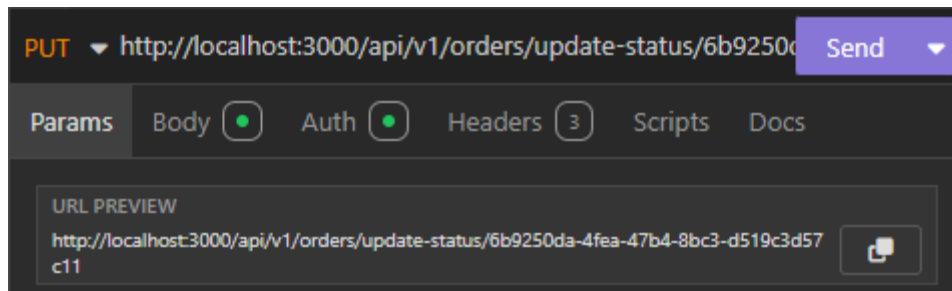
## Get /api/v1/orders/:id

- Descripción: Obtiene los detalles de una orden específica por su ID.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: user, seller, admin
- Parámetros de ruta:
  - id – UUID de la orden a consultar.

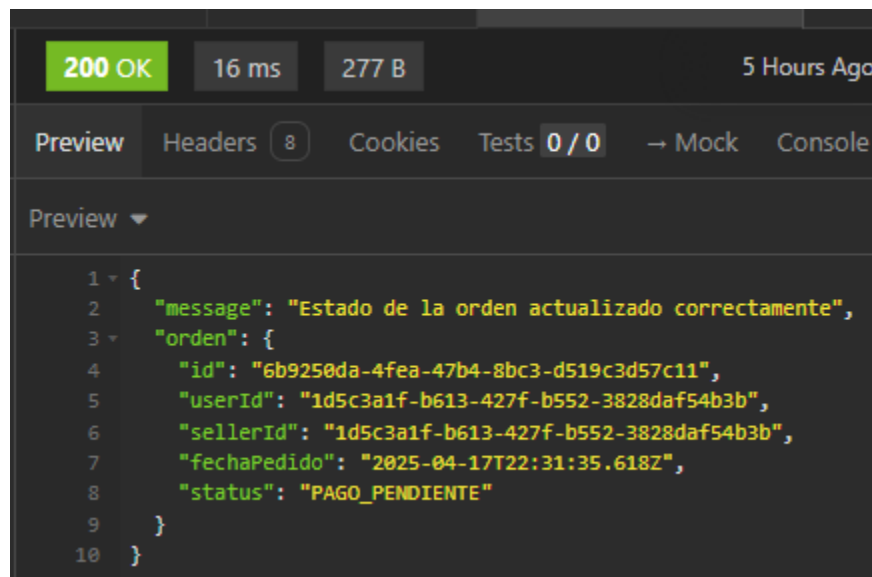
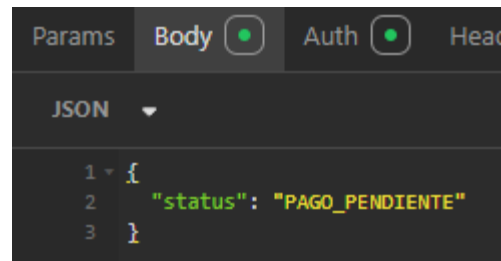


## Put /api/v1/orders/update-status/:id

- Descripción: Cambia el estado de una orden específica. Solo los vendedores pueden modificar el estado, y solo a valores válidos.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: seller
- Parámetros de ruta:
  - id – UUID de la orden.
- Cuerpo de la solicitud:
  - status – Nuevo estado de la orden. Debe ser uno de los definidos en OrderStatus.



```
enum OrderStatus {  
    CREACION_PEDIDO  
    ESPERA_DISPONIBILIDAD  
    CANCELADO_POR_VENDEDOR  
    PAGO_PENDIENTE  
    PROCESANDO  
    ENVIADO  
    RECIBIDO  
}
```

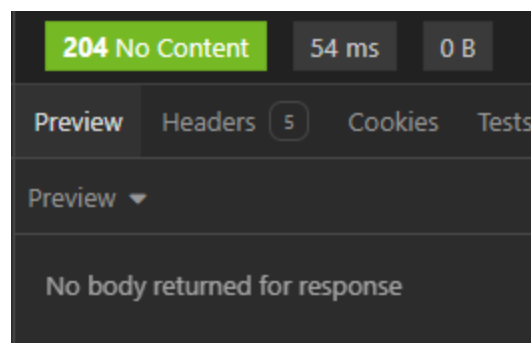
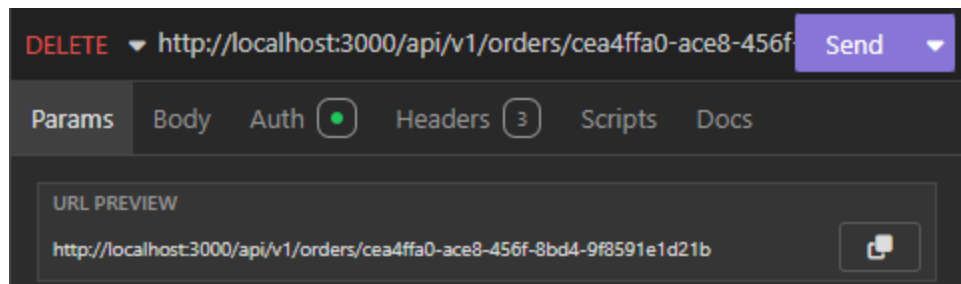


Flujo de status:

```
export const OrderStatusTransitions = {
  [OrderStatus.CREACION_PEDIDO]: [
    OrderStatus.ESPERA_DISPONIBILIDAD,
    OrderStatus.CANCELADO_POR_VENDEDOR,
  ],
  [OrderStatus.ESPERA_DISPONIBILIDAD]: [
    OrderStatus.PAGO_PENDIENTE,
    OrderStatus.CANCELADO_POR_VENDEDOR,
  ],
  [OrderStatus.CANCELADO_POR_VENDEDOR]: [],
  [OrderStatus.PAGO_PENDIENTE]: [OrderStatus.PROCESANDO],
  [OrderStatus.PROCESANDO]: [OrderStatus.ENVIADO],
  [OrderStatus.ENVIADO]: [OrderStatus.RECIBIDO],
  [OrderStatus.RECIBIDO]: [],
};
```

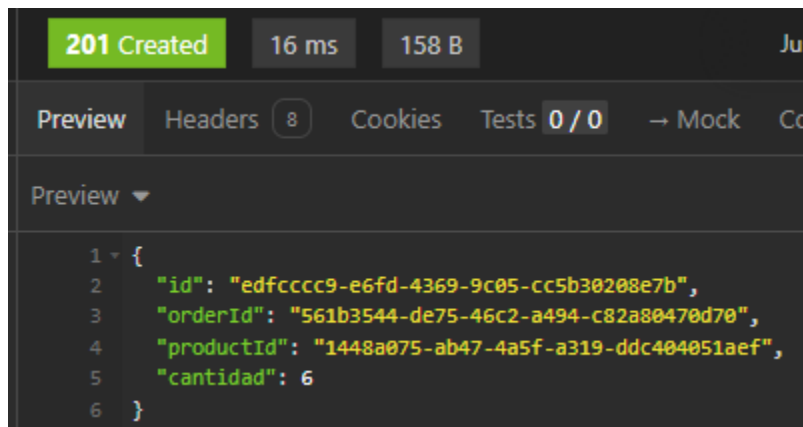
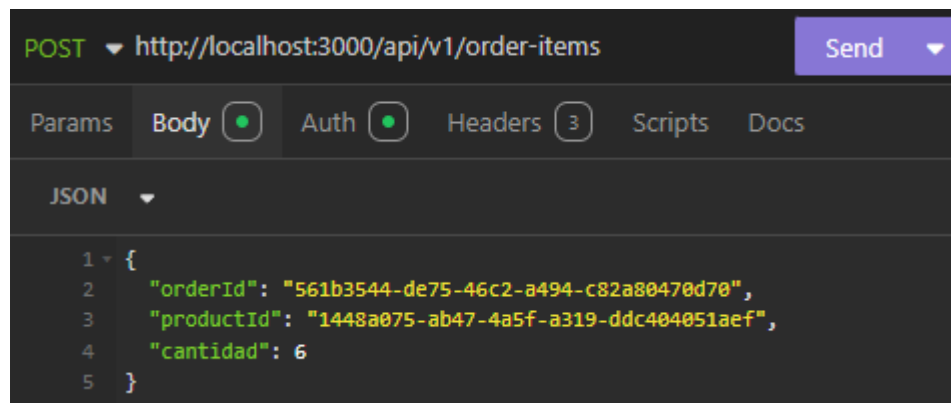
## Delete /api/v1/orders/:id

- Descripción: Elimina una orden si el usuario autenticado tiene permisos y es vendedor asociado a la orden.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: seller
- Parámetros de ruta:
  - id – UUID de la orden a eliminar.



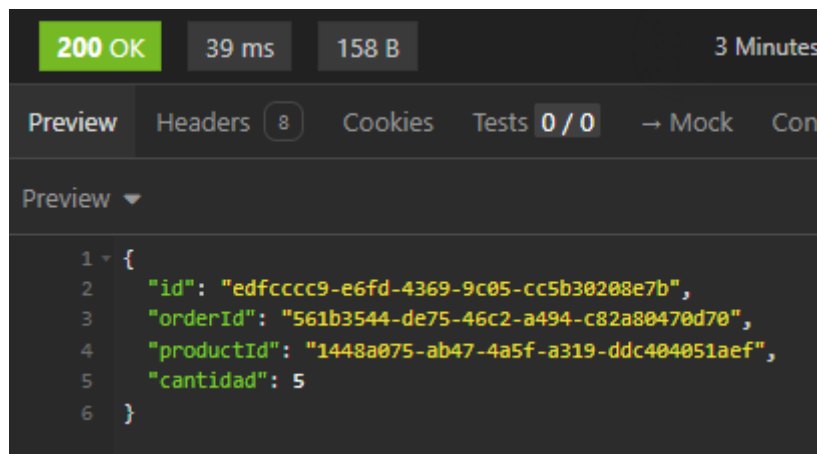
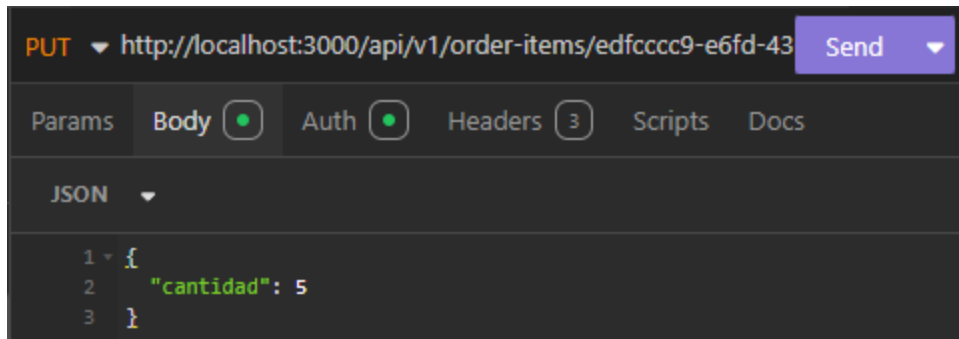
## Post /api/v1/order-items

- Descripción: Crea un nuevo ítem dentro de una orden existente. El producto debe estar disponible y la cantidad solicitada debe ser válida.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: user
- Cuerpo de la solicitud (JSON):



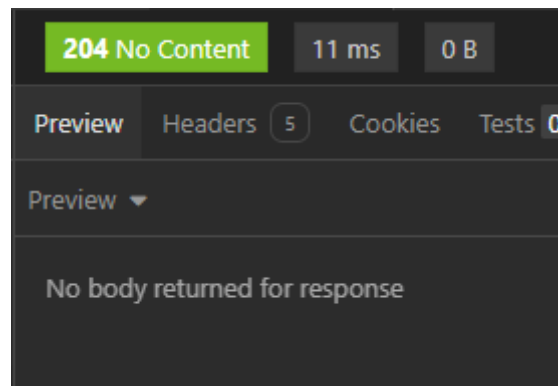
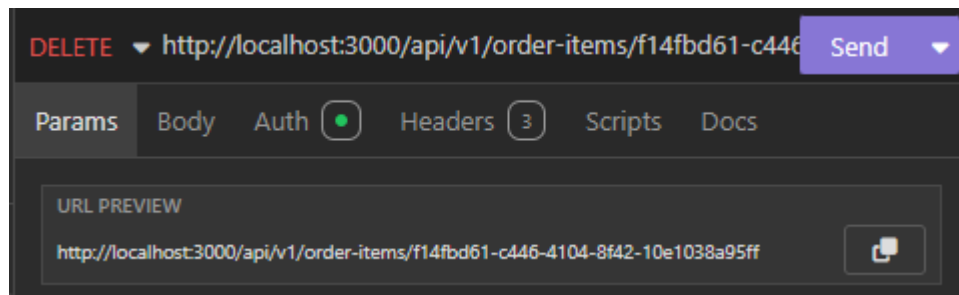
## PUT /api/v1/order-items/:id

- Descripción: Actualiza la cantidad de un ítem de orden previamente creado.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: user, seller
- Parámetros de ruta:
  - id – UUID del ítem de orden a actualizar.
- Cuerpo de la solicitud:
  - cantidad – Número entero (mínimo 1) con la nueva cantidad deseada.



## DELETE /api/v1/order-items/:id

- Descripción: Elimina un ítem de una orden existente, siempre que el usuario autenticado sea el propietario de la orden.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: user
- Parámetros de ruta:
  - id – UUID del ítem de orden a eliminar.



## GET /api/v1/order-items/:orderId/table

- Descripción: Obtiene una lista paginada de ítems pertenecientes a una orden específica. Se puede aplicar paginación y filtros adicionales si se desea.
- Autenticación: Requiere token JWT válido.
- Roles permitidos: user, seller, admin
- Parámetros de ruta:
  - orderId – UUID de la orden a consultar.
- Parámetros de consulta (opcional):
  - pageQuery – Número de página (por defecto: 1).
  - sizeQuery – Tamaño de página (por defecto: 10).

