

# **Documento Parcial S2**

**Luis Alejandro Alarcón Chaves**

**ID 792825**

**ingeniería en sistemas, corporación universitaria el minuto de Dios**

**Bases de Datos Masivas**

**NRC-10-60747**

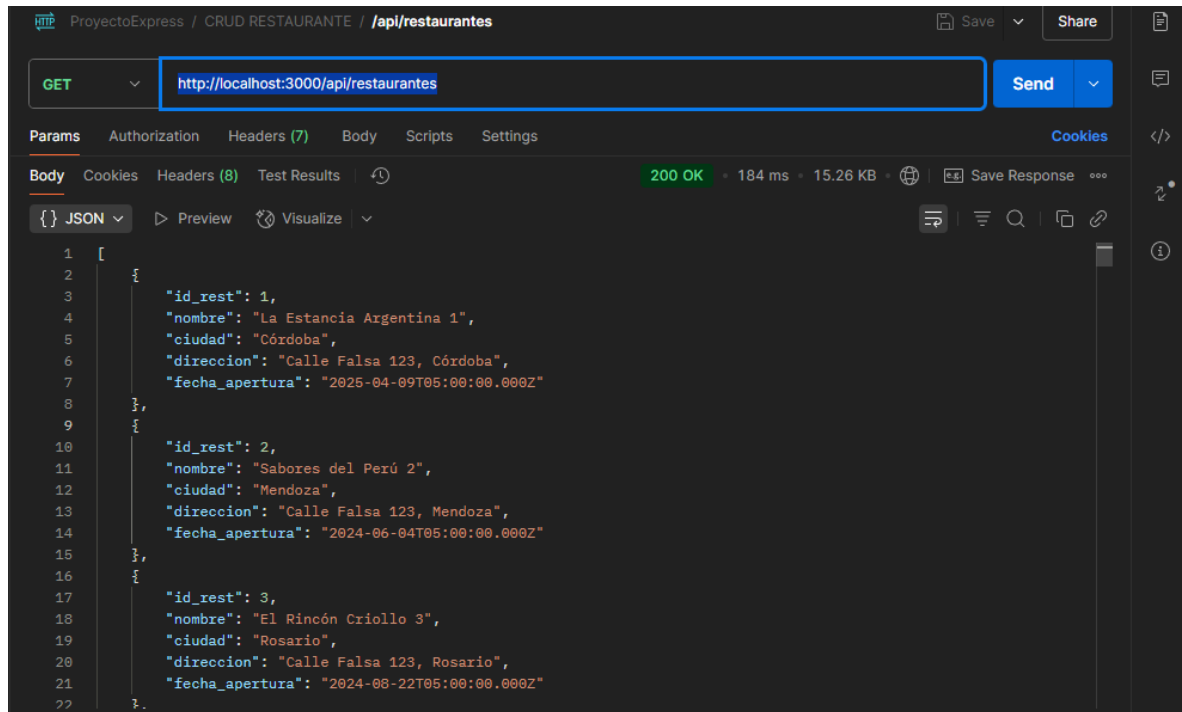
**Prof. William Alexander Matallana Porras**

**Abril 24, 2025.**

- Entregar capturas de pantalla Mostrar la solicitud con su método, parámetros y resultado.

## CRUD RESTAURANTE

GET <http://localhost:3000/api/restaurantes>



ProjectoExpress / CRUD RESTAURANTE / /api/restaurantes

GET <http://localhost:3000/api/restaurantes> Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Body Cookies Headers (8) Test Results 200 OK • 184 ms • 15.26 KB Save Response

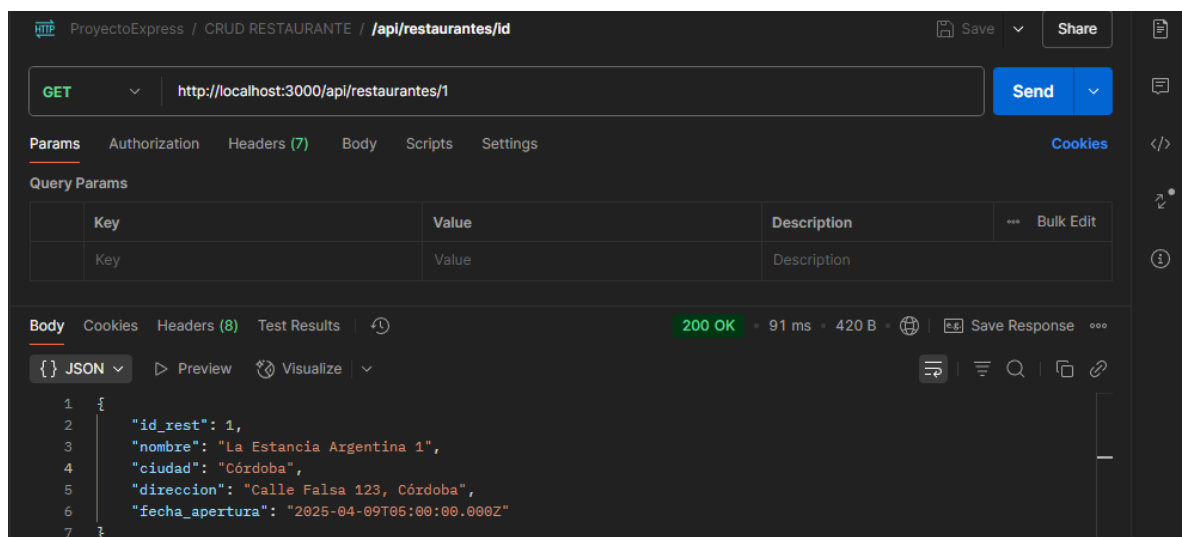
{ } JSON Preview Visualize

```

1  [
2    {
3      "id_rest": 1,
4      "nombre": "La Estancia Argentina 1",
5      "ciudad": "Córdoba",
6      "direccion": "Calle Falsa 123, Córdoba",
7      "fecha_apertura": "2025-04-09T05:00:00.000Z"
8    },
9    {
10     "id_rest": 2,
11     "nombre": "Sabores del Perú 2",
12     "ciudad": "Mendoza",
13     "direccion": "Calle Falsa 123, Mendoza",
14     "fecha_apertura": "2024-06-04T05:00:00.000Z"
15   },
16   {
17     "id_rest": 3,
18     "nombre": "El Rincón Criollo 3",
19     "ciudad": "Rosario",
20     "direccion": "Calle Falsa 123, Rosario",
21     "fecha_apertura": "2024-08-22T05:00:00.000Z"
22   }
23 ]

```

GET <http://localhost:3000/api/restaurantes/id>



ProjectoExpress / CRUD RESTAURANTE / /api/restaurantes/id

GET <http://localhost:3000/api/restaurantes/1> Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (8) Test Results 200 OK • 91 ms • 420 B Save Response

{ } JSON Preview Visualize

```

1  {
2    "id_rest": 1,
3    "nombre": "La Estancia Argentina 1",
4    "ciudad": "Córdoba",
5    "direccion": "Calle Falsa 123, Córdoba",
6    "fecha_apertura": "2025-04-09T05:00:00.000Z"
7  }

```

POST <http://localhost:3000/api/restaurantes>

ProjectoExpress / CRUD RESTAURANTE / [/api/restaurantes](#)

POST [http://localhost:3000/api/restaurantes](#) [Send](#)

Params Authorization Headers (9) **Body** Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL ☐ JSON [Beautify](#)

```
1 {
2   "nombre": "La Fonda",
3   "ciudad": "Guanajato",
4   "direccion": "Barrio Los Prados",
5   "fecha_apertura": "2025-04-25"
6 }
```

**Body** Cookies Headers (8) Test Results [201 Created](#) • 104 ms • 405 B [Save Response](#)

[JSON](#) [Preview](#) [Visualize](#)

```
1 {
2   "id_rest": 101,
3   "nombre": "La Fonda",
4   "ciudad": "Guanajato",
5   "direccion": "Barrio Los Prados",
6   "fecha_apertura": "2025-04-25T05:00:00.000Z"
7 }
```

PUT <http://localhost:3000/api/restaurantes/id>

ProjectoExpress / CRUD RESTAURANTE / [/api/restaurantes/](#)

PUT [http://localhost:3000/api/restaurantes/101](#) [Send](#)

Params Authorization Headers (9) **Body** Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL ☐ JSON [Beautify](#)

```
1 {
2   "nombre": "La Fonda MX",
3   "ciudad": "Chihuahua",
4   "direccion": "Calle Falsa 265, Chihuahua",
5   "fecha_apertura": "2024-06-13"
6 }
```

**Body** Cookies Headers (8) Test Results [200 OK](#) • 102 ms • 412 B [Save Response](#)

[JSON](#) [Preview](#) [Visualize](#)

```
1 {
2   "id_rest": 101,
3   "nombre": "La Fonda MX",
4   "ciudad": "Chihuahua",
5   "direccion": "Calle Falsa 265, Chihuahua",
6   "fecha_apertura": "2024-06-13T05:00:00.000Z"
7 }
```

DELETE <http://localhost:3000/api/restaurantes/101>

ProjectoExpress / CRUD RESTAURANTE / [/api/restaurantes/id](#)

DELETE [http://localhost:3000/api/restaurantes/101](#) [Send](#)

Params Authorization Headers (7) **Body** Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
-----	-------	-------------	-----------

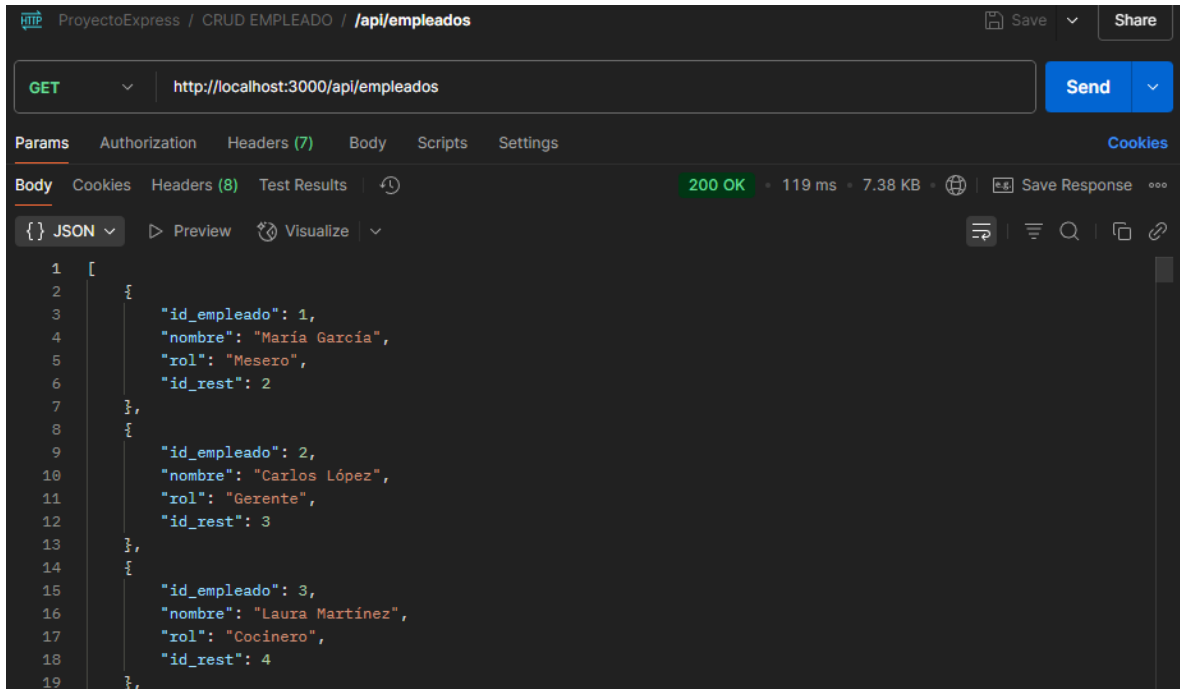
**Body** Cookies Headers (8) Test Results [200 OK](#) • 99 ms • 304 B [Save Response](#)

[JSON](#) [Preview](#) [Visualize](#)

```
1 {
2   "message": "Eliminado correctamente"
3 }
```

## CRUD EMPLEADO

GET <http://localhost:3000/api/empleados>



ProjectoExpress / CRUD EMPLEADO / /api/empleados

GET <http://localhost:3000/api/empleados> Send

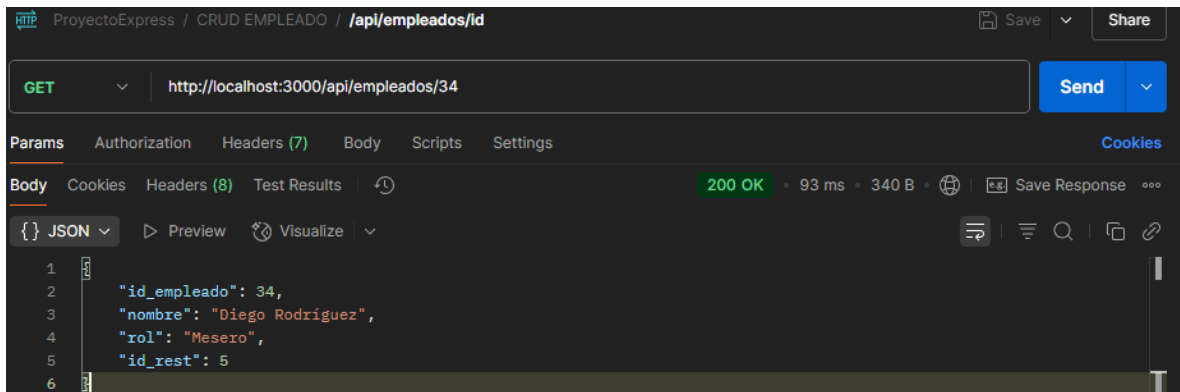
Params Authorization Headers (7) Body Scripts Settings Cookies

Body Cookies Headers (8) Test Results 200 OK • 119 ms • 7.38 KB Save Response

{ } JSON Preview Visualize

```
1 [
2   {
3     "id_empleado": 1,
4     "nombre": "Maria García",
5     "rol": "Mesero",
6     "id_rest": 2
7   },
8   {
9     "id_empleado": 2,
10    "nombre": "Carlos López",
11    "rol": "Gerente",
12    "id_rest": 3
13  },
14  {
15    "id_empleado": 3,
16    "nombre": "Laura Martínez",
17    "rol": "Cocinero",
18    "id_rest": 4
19  },
20 ]
```

GET <http://localhost:3000/api/empleados/id>



ProjectoExpress / CRUD EMPLEADO / /api/empleados/id

GET <http://localhost:3000/api/empleados/34> Send

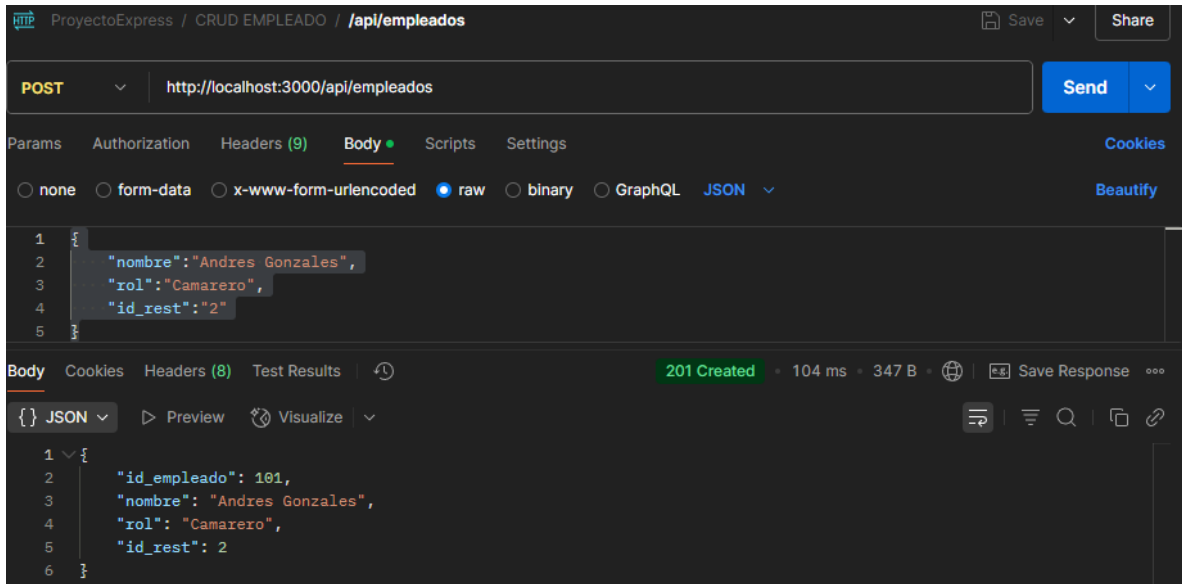
Params Authorization Headers (7) Body Scripts Settings Cookies

Body Cookies Headers (8) Test Results 200 OK • 93 ms • 340 B Save Response

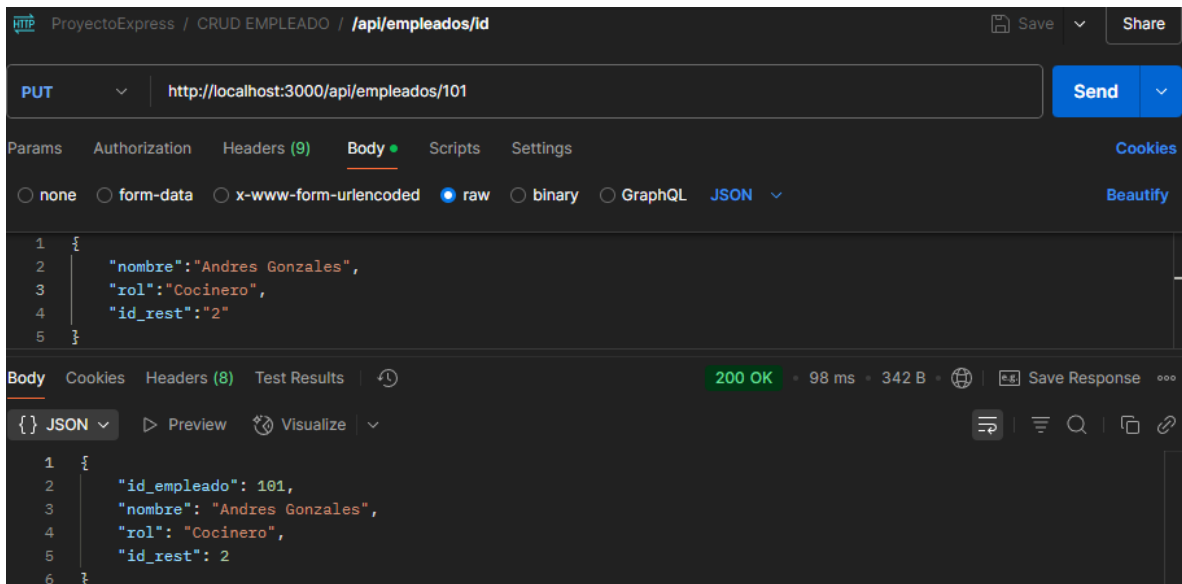
{ } JSON Preview Visualize

```
1 {
2   "id_empleado": 34,
3   "nombre": "Diego Rodríguez",
4   "rol": "Mesero",
5   "id_rest": 5
6 }
```

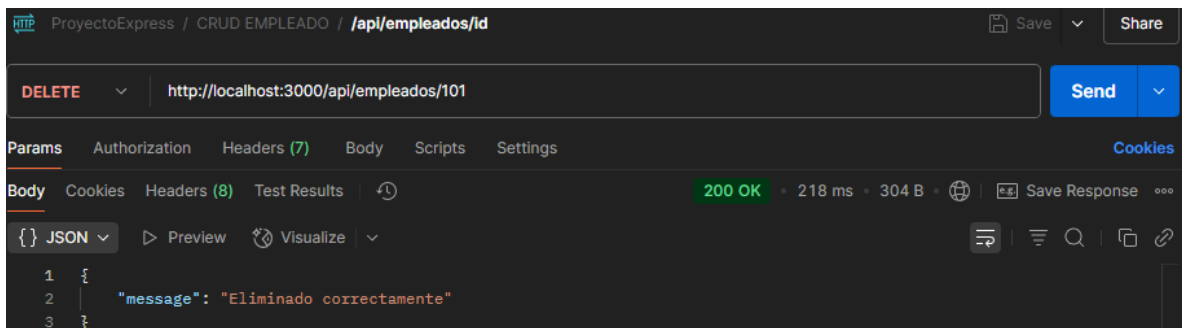
POST <http://localhost:3000/api/empleados>



**PUT** <http://localhost:3000/api/empleados/id>



**DELETE** <http://localhost:3000/api/empleados/id>



**CRUD PRODUCTO**

GET <http://localhost:3000/api/productos>

ProjectoExpress / CRUD PRODU... / **/api/productos** Save Share

GET **http://localhost:3000/api/productos** Send

Params Auth Headers (7) Body Scripts Settings Cookies

Body 200 OK • 322 ms • 5.88 KB

JSON Preview Visualize

```
1  [
2    {
3      "id_prod": 1,
4      "nombre": "Asado Argentino",
5      "precio": "11.12"
6    },
7    {
8      "id_prod": 2,
9      "nombre": "Locro",
10     "precio": "14.83"
11   },
12   {
13     "id_prod": 3,
14     "nombre": "Milanesa Napolitana",
15     "precio": "8.50"
16   },
17 ]
```

GET <http://localhost:3000/api/productos/id>

ProjectoExpress / CRUD PRO... / **/api/productos/id** Save Share

GET **http://localhost:3000/api/productos/4** Send

Params Auth Headers (7) Body Scripts Settings Cookies

Body 200 OK • 315 ms • 325 B

JSON Preview Visualize

```
1  {
2    "id_prod": 4,
3    "nombre": "Empanada Criolla",
4    "precio": "12.22"
5  }
```

POST <http://localhost:3000/api/productos>

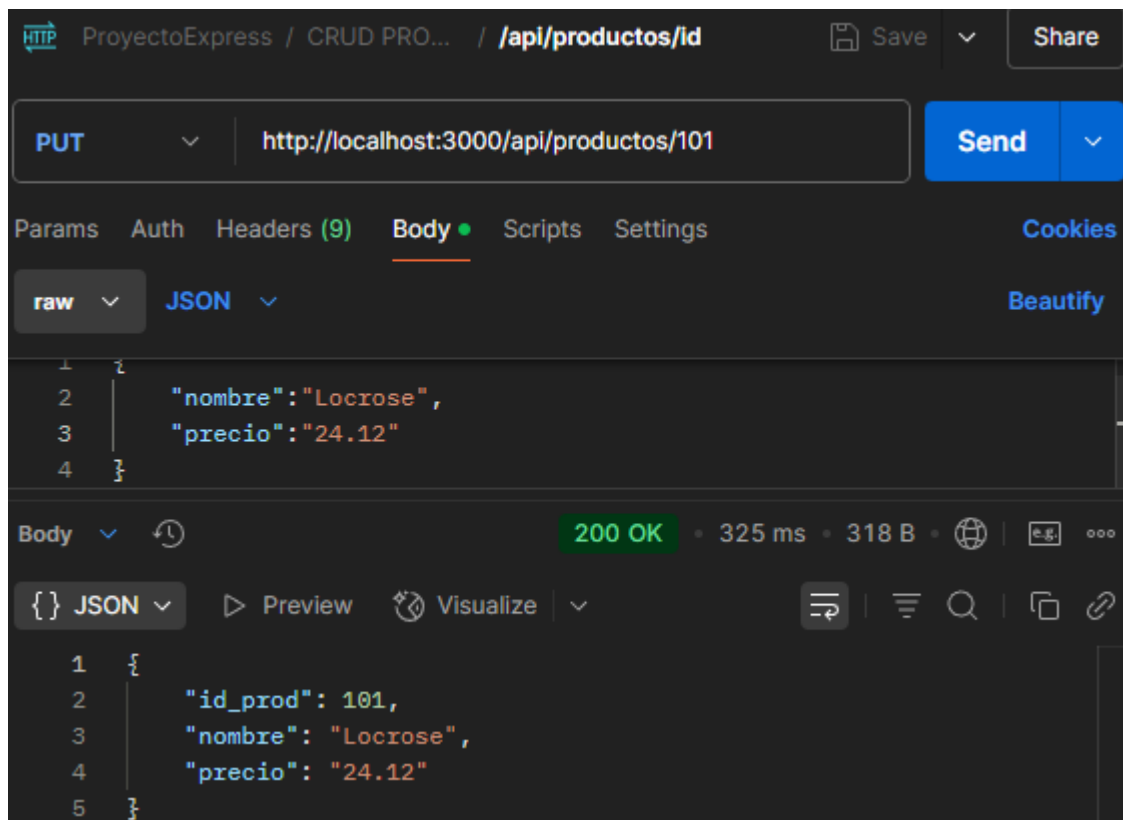
The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** <http://localhost:3000/api/productos>
- Body (raw):**

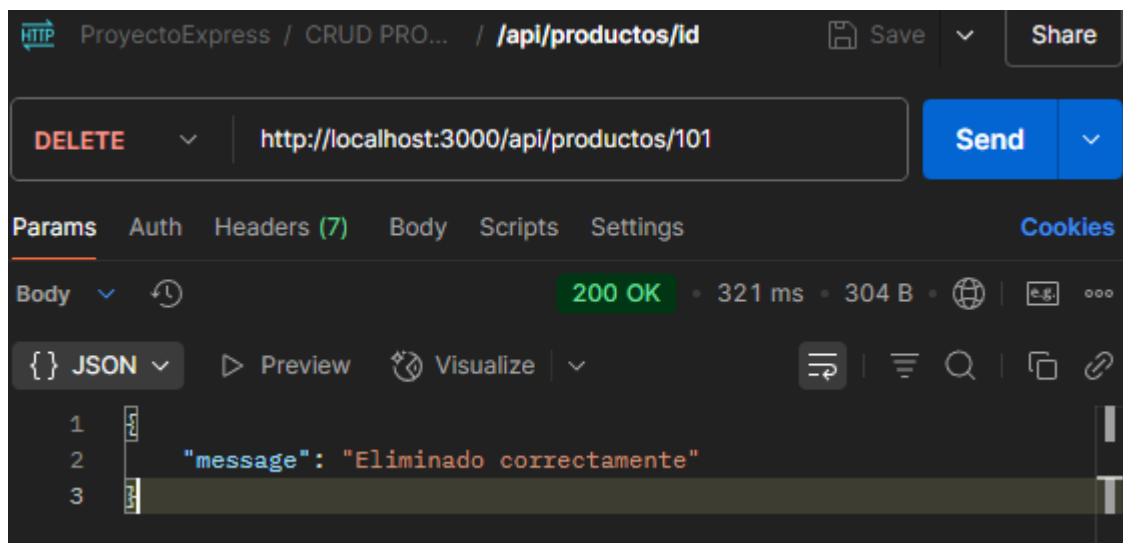
```
{  "nombre": "Locro",  "precio": "20.12"}
```
- Status:** 201 Created
- Response Time:** 144 ms
- Response Size:** 321 B
- Response Body (JSON):**

```
{  "id_prod": 101,  "nombre": "Locro",  "precio": "20.12"}
```

PUT <http://localhost:3000/api/productos/id>



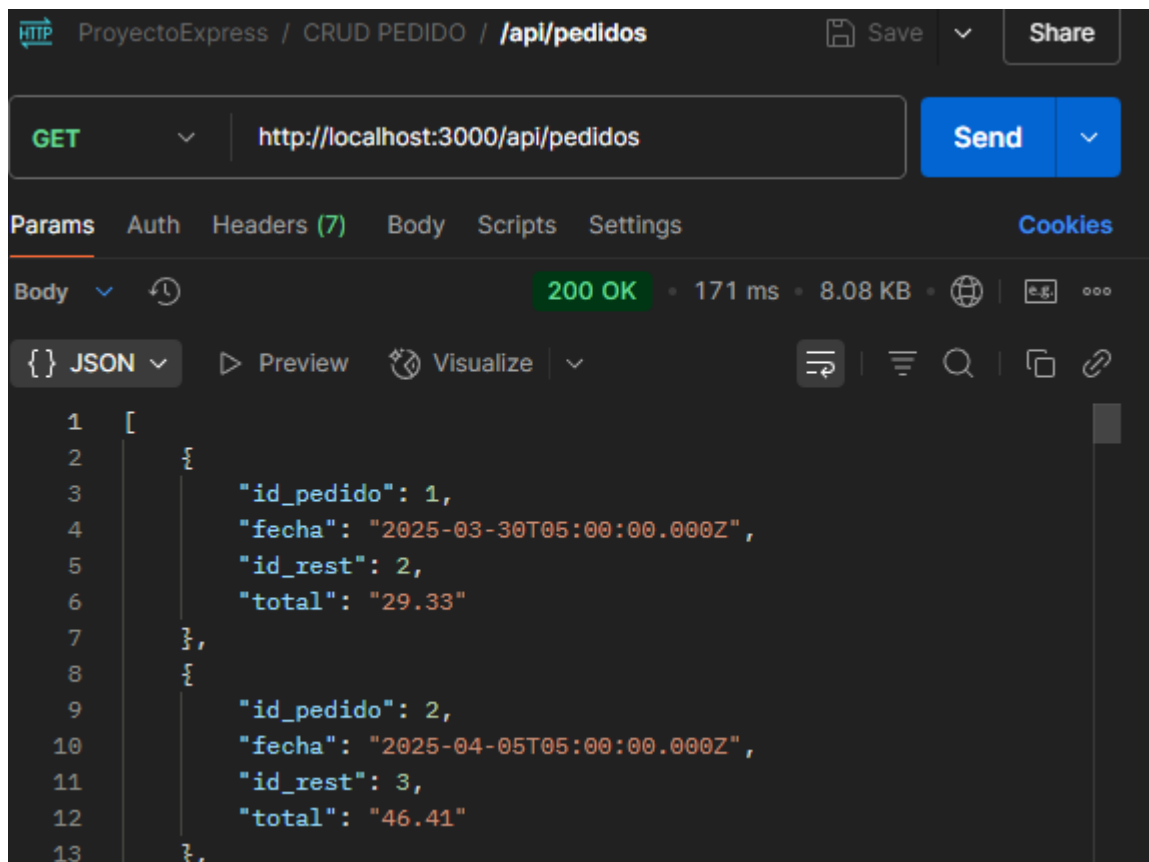
**DELETE** <http://localhost:3000/api/productos/id>



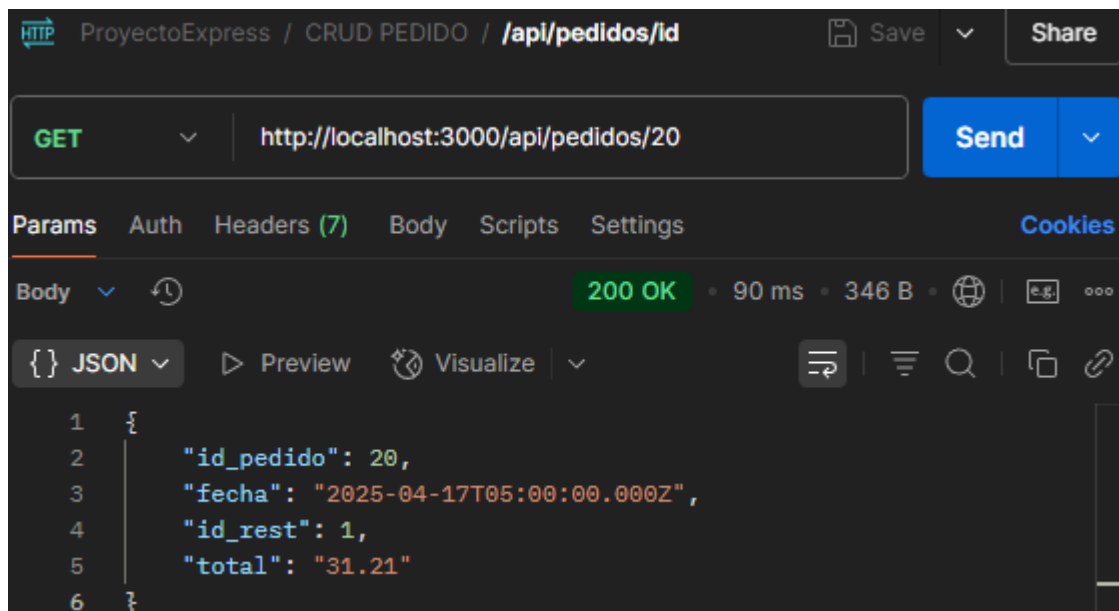
## CRUD PEDIDO

**GET** <http://localhost:3000/api/pedidos>





GET <http://localhost:3000/api/pedidos/id>



POST <http://localhost:3000/api/pedidos>

ProyectoExpress / CRUD PEDIDO / [/api/pedidos](#) Save Share

POST http://localhost:3000/api/pedidos Send

Params Auth Headers (9) Body Scripts Settings Cookies Beautify

raw JSON

```
1 {  
2   "fecha": "2024-08-18",  
3   "id_rest": "4",  
4   "total": "30.12"  
5 }
```

Body 201 Created • 306 ms • 352 B • Preview Visualize

{ } JSON Preview Visualize

```
1 {  
2   "id_pedido": 101,  
3   "fecha": "2024-08-18T05:00:00.000Z",  
4   "id_rest": 4,  
5   "total": "30.12"  
6 }
```

PUT <http://localhost:3000/api/pedidos/id>

ProyectoExpress / CRUD PEDIDO / [/api/pedidos/id](#) Save Share

PUT [http://localhost:3000/api/pedidos/101](#) Send

Params Auth Headers (9) **Body** Scripts Settings Cookies

raw JSON Beautify

```
1 {
2   "fecha": "2024-09-12",
3   "id_rest": "5",
4   "total": "32.12"
5 }
```

Body 200 OK • 99 ms • 347 B

{ } JSON Preview Visualize

```
1 {
2   "id_pedido": 101,
3   "fecha": "2024-09-12T05:00:00.000Z",
4   "id_rest": 5,
5   "total": "32.12"
6 }
```

**DELETE** <http://localhost:3000/api/pedidos/id>

ProyectoExpress / CRUD PEDIDO / [/api/pedidos/id](#) Save Share

DELETE [http://localhost:3000/api/pedidos/101](#) Send

Params Auth Headers (7) **Body** Scripts Settings Cookies

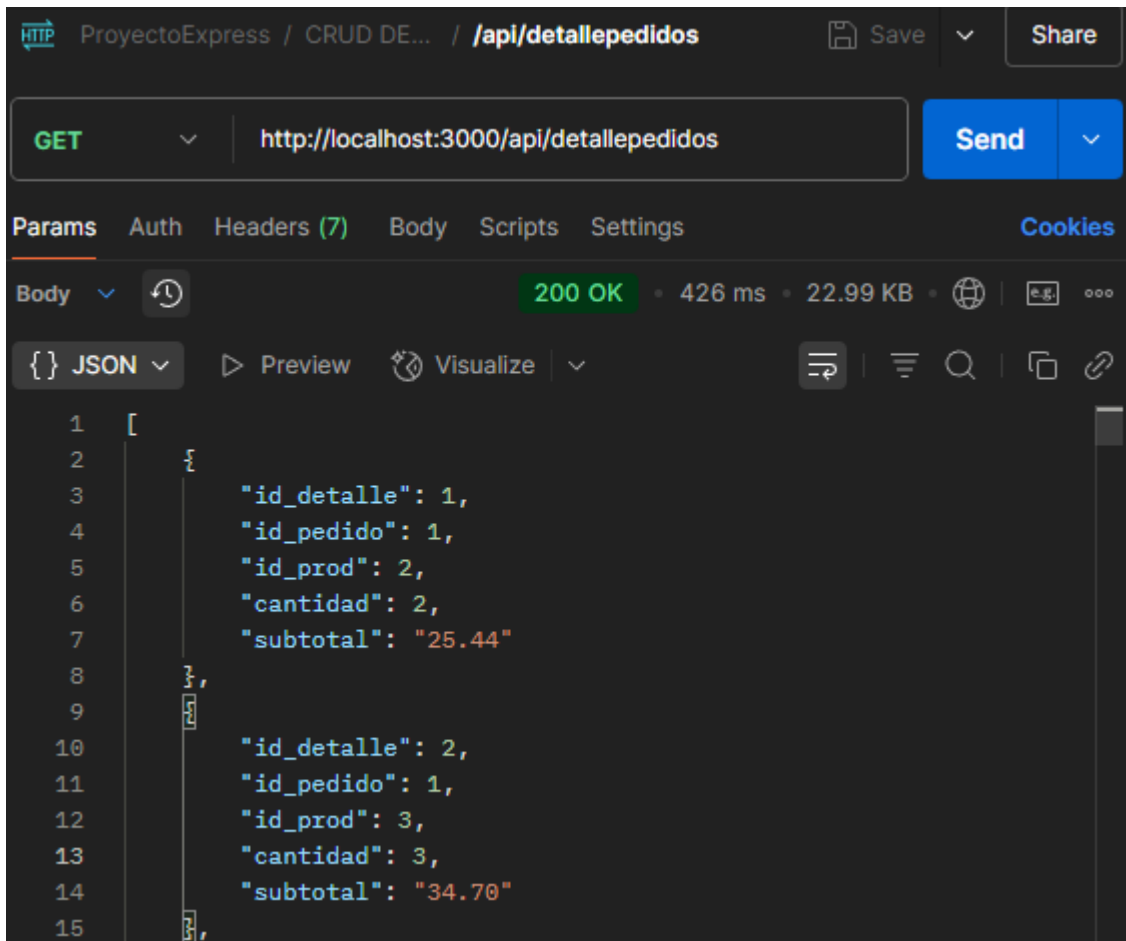
Body 200 OK • 102 ms • 304 B

{ } JSON Preview Visualize

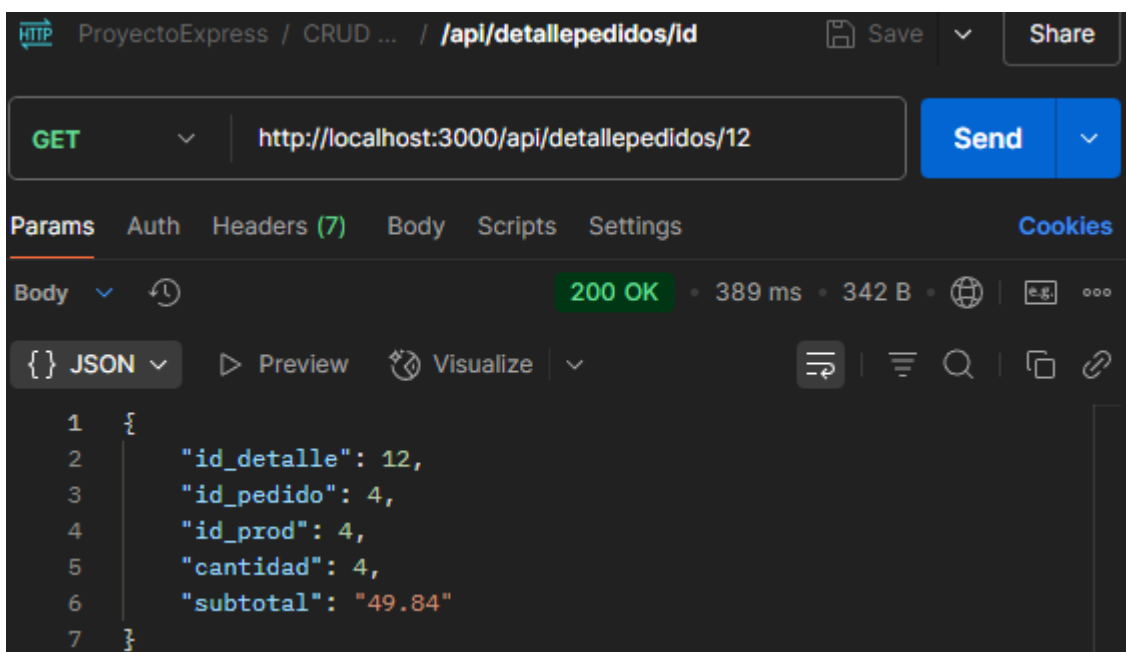
```
1 {
2   "message": "Eliminado correctamente"
3 }
```

## CRUD DETALLE PEDIDO

**GET** <http://localhost:3000/api/detallepedidos>



GET <http://localhost:3000/api/detallepedidos/id>



POST <http://localhost:3000/api/detallepedidos>

HTTP ProyectoExpress / CRUD DE... / **/api/detallepedidos** Save Share

**POST** http://localhost:3000/api/detallepedidos Send

Params Auth Headers (9) **Body** Scripts Settings Cookies

raw JSON Beautify

```
2   "id_pedido": "20",
3   "id_prod": "2",
4   "cantidad": "4",
5   "subtotal": "45.12"
6 }
```

Body 201 Created • 262 ms • 349 B • Preview Visualize

```
1 {
2   "id_detalle": 301,
3   "id_pedido": 20,
4   "id_prod": 2,
5   "cantidad": 4,
6   "subtotal": "45.12"
7 }
```

PUT <http://localhost:3000/api/detallepedidos/id>

HTTP ProyectoExpress / CRUD ... / **/api/detallepedidos/id** Save Share

**PUT** http://localhost:3000/api/detallepedidos/301 Send

Params Auth Headers (9) **Body** Scripts Settings Cookies

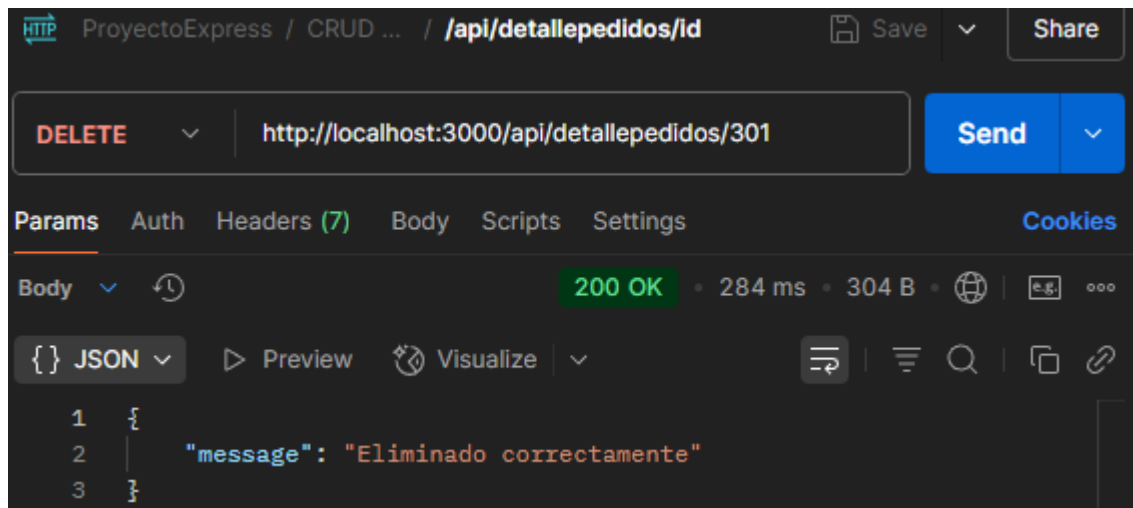
raw JSON Beautify

```
2   "id_pedido": "20",
3   "id_prod": "4",
4   "cantidad": "8",
5   "subtotal": "95.24"
6 }
```

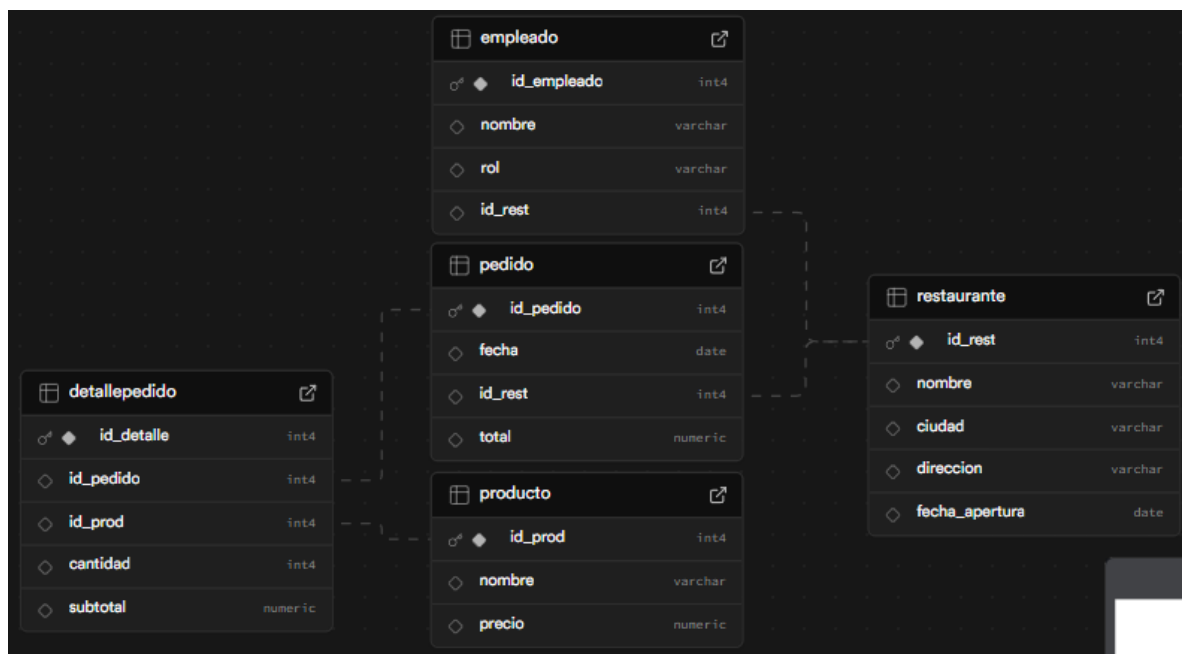
Body 200 OK • 215 ms • 344 B • Preview Visualize

```
1 {
2   "id_detalle": 301,
3   "id_pedido": 20,
4   "id_prod": 4,
5   "cantidad": 8,
6   "subtotal": "95.24"
7 }
```

**DELETE** <http://localhost:3000/api/detallepedidos/id>



- Exportar la colección de postman en formato .json
- Entregar modelo de datos de supabase



- Documentar las consultas nativas con las respectivas peticiones en Postman
- **.Obtener todos los productos de un pedido específico**

**Endpoint:**

**GET /api/pedidos/:id/productos**

**Descripción:**

Este endpoint permite obtener todos los productos asociados a un pedido específico, identificado por su id.

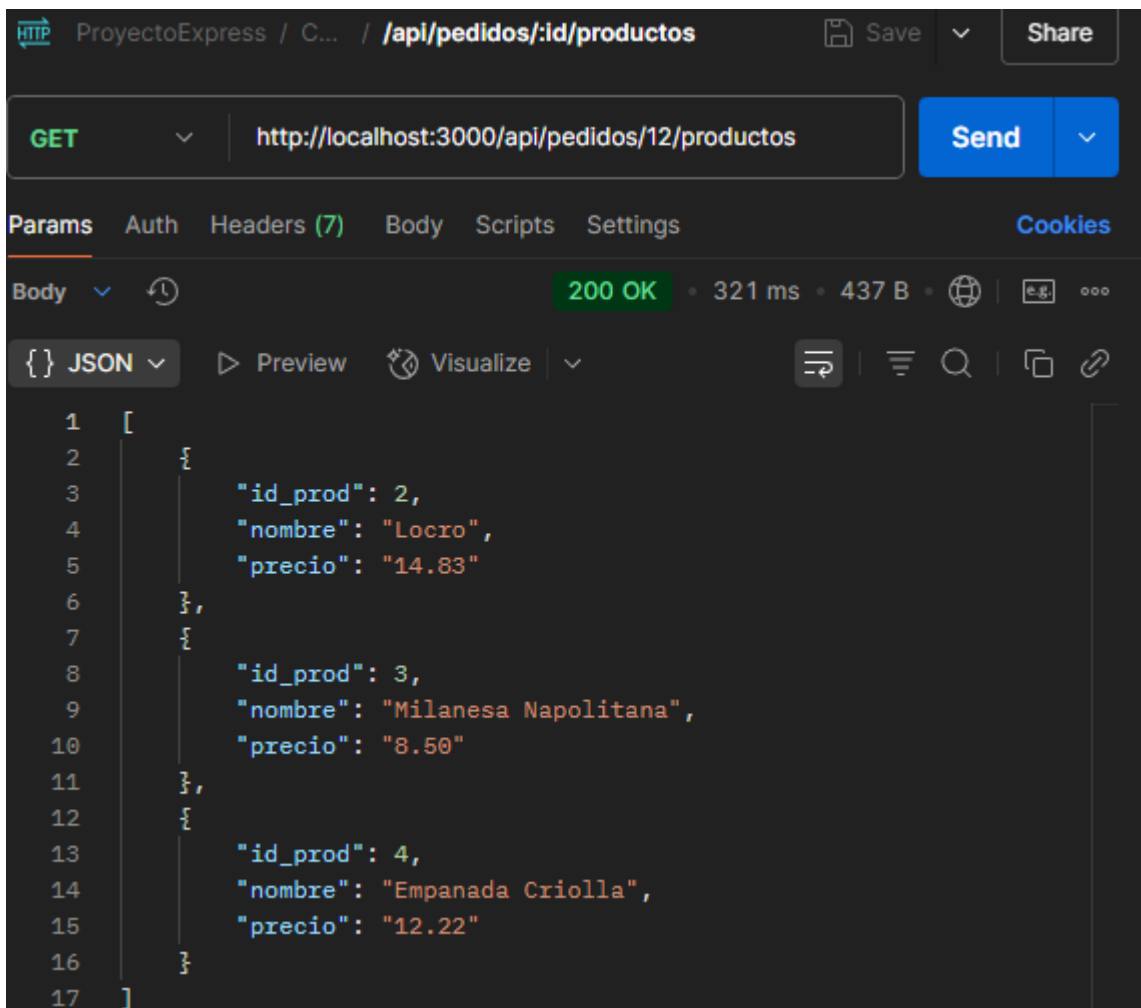
Parámetros:- id (URL param): ID del pedido.

### Consulta SQL:

```
SELECT p.*FROM producto p JOIN detallepedido dp ON p.id_prod = dp.id_prod WHERE  
dp.id_pedido = $1
```

### Proceso de la consulta SQL

- Se seleccionan todos los campos (\*) de la tabla producto.
- Se realiza un JOIN con la tabla detallepedido para relacionar los productos con los pedidos.
- Se filtra el resultado para que solo incluya los productos correspondientes al id\_pedido especificado.



- Obtener productos más vendidos (más de X unidades)

Endpoint:

GET /api/productos/mas-vendidos/:cantidad

### Descripción:

Este endpoint devuelve una lista de productos que han sido vendidos en una cantidad superior a la indicada en el parámetro cantidad. Es útil para identificar los productos más populares o con mayor rotación en el sistema.

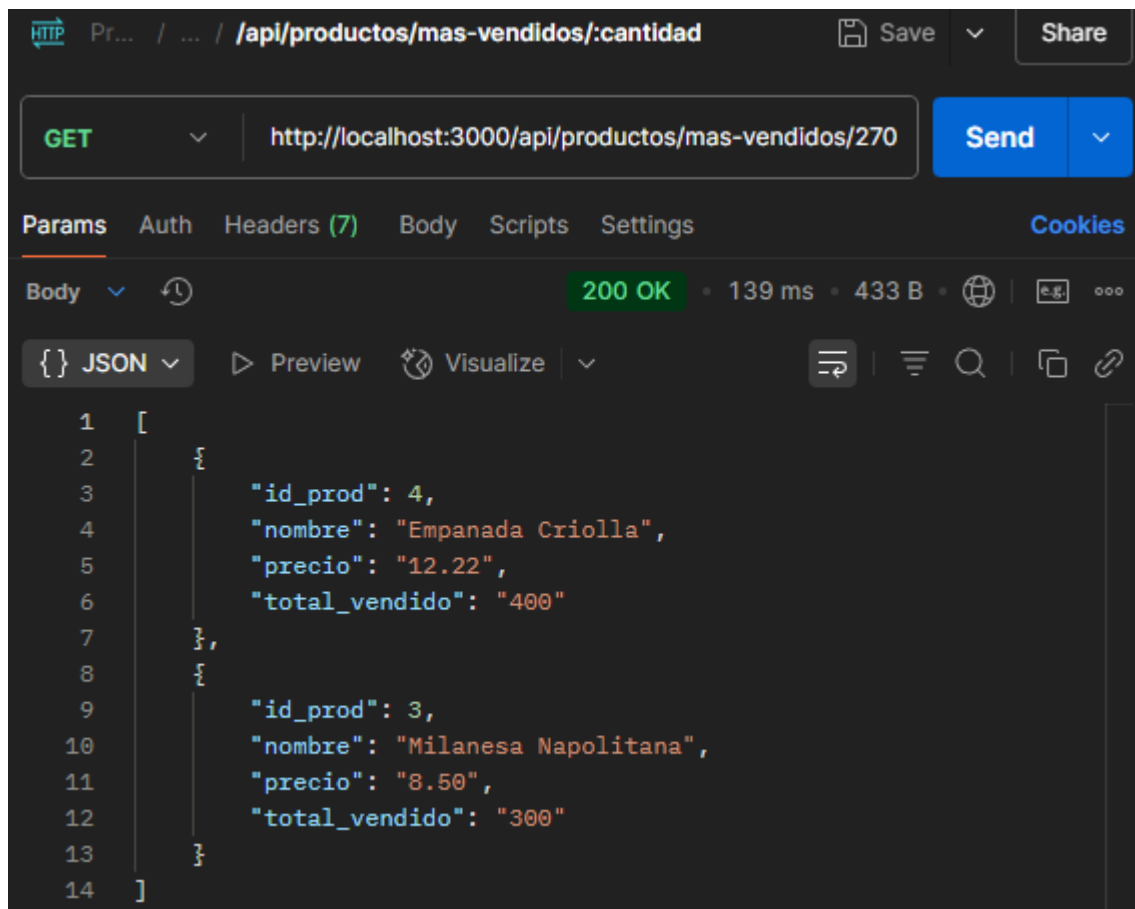
Parámetros:- cantidad (URL param): Número mínimo de unidades vendidas.

### Consulta SQL:

```
SELECT p.*, SUM(dp.cantidad) AS total_vendido FROM producto p JOIN detallepedido dp ON p.id_prod = dp.id_prod GROUP BY p.id_prod HAVING SUM(dp.cantidad) > $1
```

### Proceso de la consulta SQL

- Se seleccionan todos los campos de la tabla producto junto con la suma total de unidades vendidas (SUM(dp.cantidad)) como total\_vendido.
- Se realiza un JOIN con la tabla detallepedido para acceder a las cantidades vendidas por producto.
- Se agrupan los resultados por el ID del producto (GROUP BY p.id\_prod).
- Se filtran los resultados con HAVING SUM(dp.cantidad) > \$1 para incluir solo productos cuyo total vendido supera el valor indicado en el parámetro.



The screenshot shows a REST client interface with a GET request to `http://localhost:3000/api/productos/mas-vendidos/270`. The response is a 200 OK status with a 139 ms response time and 433 B of data. The response body is a JSON array containing two product objects.

```
{
  "id_prod": 4,
  "nombre": "Empanada Criolla",
  "precio": "12.22",
  "total_vendido": "400"
},
{
  "id_prod": 3,
  "nombre": "Milanesa Napolitana",
  "precio": "8.50",
  "total_vendido": "300"
}
```



- **Obtener total de ventas por restaurante**

**Endpoint:**

**GET /api/restaurantes/ventas/id\_rest**

**Descripción:**

Este endpoint devuelve el total de ventas de un restaurante específico, identificado por su id\_rest. El total de ventas se calcula sumando la cantidad vendida de cada producto multiplicada por su precio.

**Consulta SQL:**

```
SELECT r.id_rest, r.nombre, COALESCE(SUM(dp.cantidad * p.precio), 0) AS total_ventas  
FROM restaurante r
```

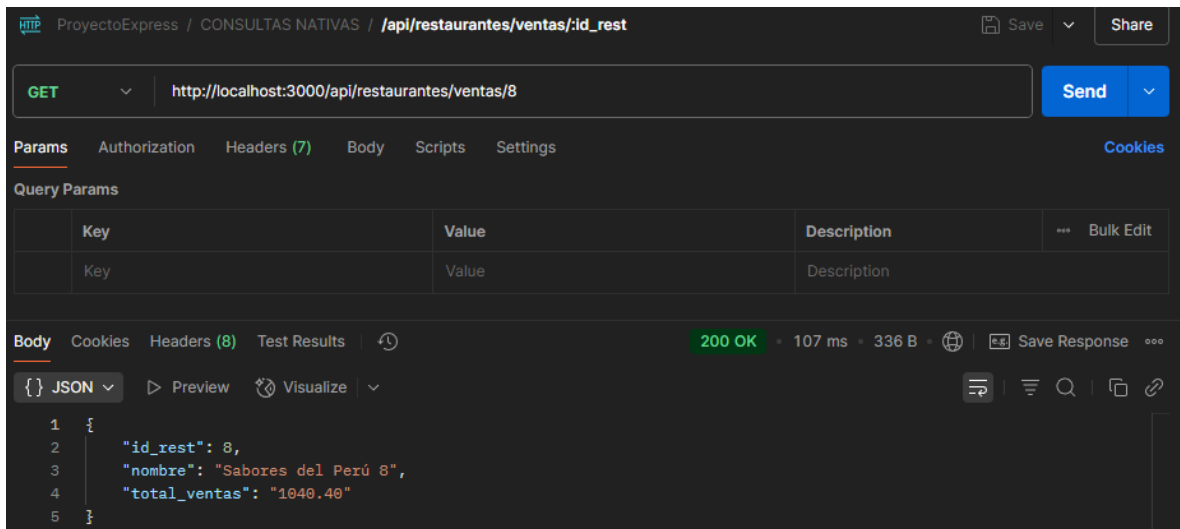
```
LEFT JOIN pedido ped ON r.id_rest = ped.id_rest
```

```
LEFT JOIN detallepedido dp ON ped.id_pedido = dp.id_pedido
```

```
LEFT JOIN producto p ON dp.id_prod = p.id_prod WHERE r.id_rest = $1 GROUP BY r.id_rest,  
r.nombre
```

**Proceso de la consulta SQL**

- Se seleccionan los campos id\_rest y nombre del restaurante.
- Se realiza un LEFT JOIN con la tabla pedido para obtener los pedidos asociados al restaurante.
- Se realiza un LEFT JOIN con la tabla detallepedido para obtener los detalles de cada pedido.
- Se realiza un LEFT JOIN con la tabla producto para obtener el precio de cada producto vendido.
- Se calcula el total de ventas con la suma del producto de la cantidad vendida (dp.cantidad) por el precio (p.precio).
- Se usa COALESCE para que, en caso de que no haya ventas, el total sea 0.
- Se filtra por el restaurante con el id\_rest proporcionado.
- Se agrupa por el ID y nombre del restaurante para consolidar la suma.



- Obtener pedidos realizados en una fecha específica

**Endpoint:**

**GET /api/pedidos/fecha/:fecha**

**Descripción:**

Este endpoint permite obtener todos los pedidos que fueron realizados en una fecha específica. Es útil para consultar las ventas del día o generar reportes por fecha.

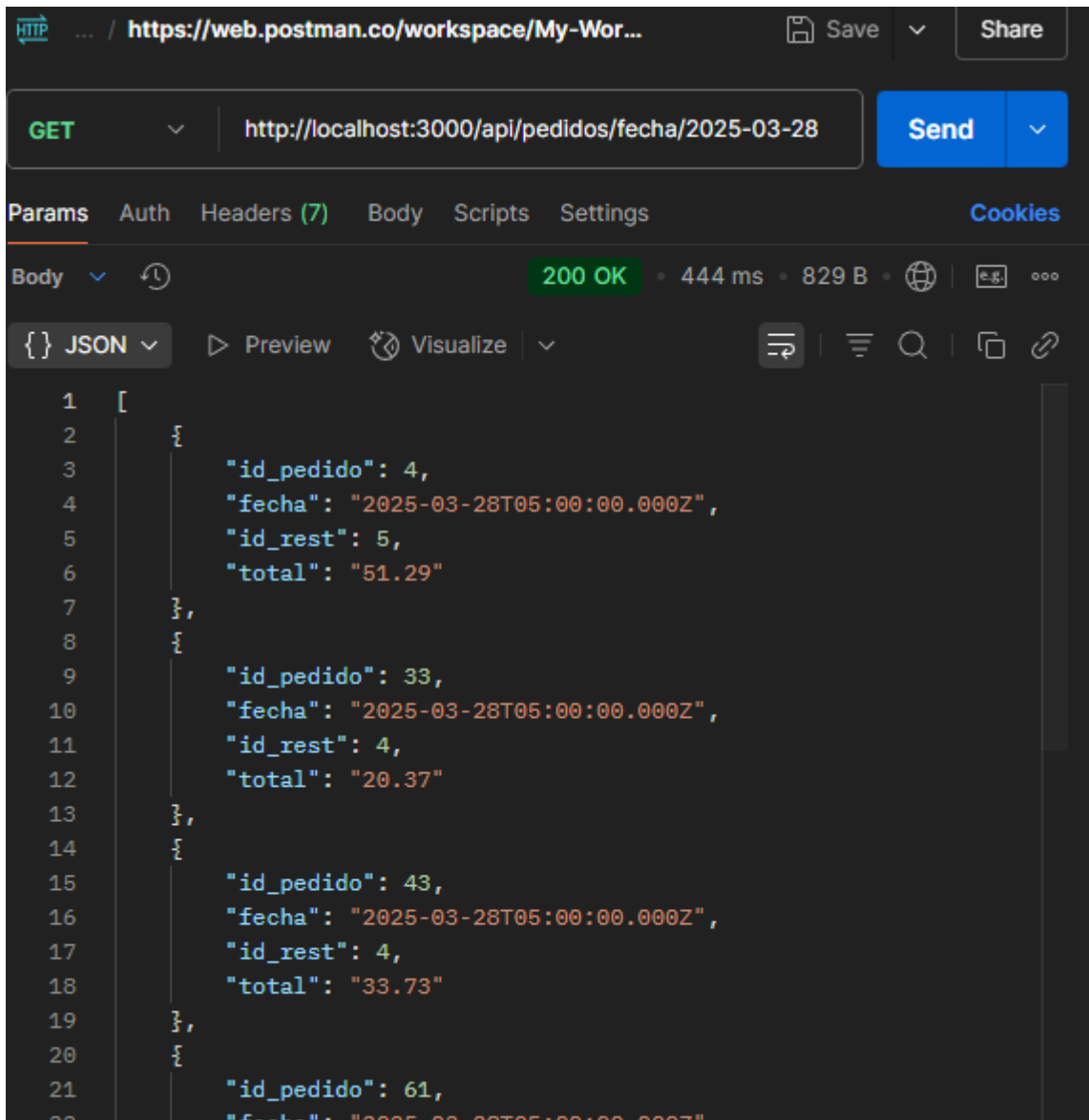
Parámetros:- fecha (URL param): Fecha en formato YYYY-MM-DD.

**Consulta SQL:**

SELECT \*FROM pedido WHERE DATE(fecha) = \$1

**Proceso de la consulta SQL**

- Se seleccionan todos los campos de la tabla pedido.
- Se utiliza la función DATE(fecha) para comparar únicamente la parte de fecha (ignorando la hora si existiera).
- Se filtran los resultados para que coincidan exactamente con la fecha proporcionada.



- Obtener empleados por rol en un restaurante

**Endpoint:**

**GET** `http://localhost:3000/api/empleados/:id_rest/:rol`

**Descripción:**

Este endpoint permite obtener todos los empleados que trabajan en un restaurante específico y que cumplen un rol determinado. Es útil para filtrar empleados según su función dentro de un restaurante (como mesero, cocinero, administrador, etc.).

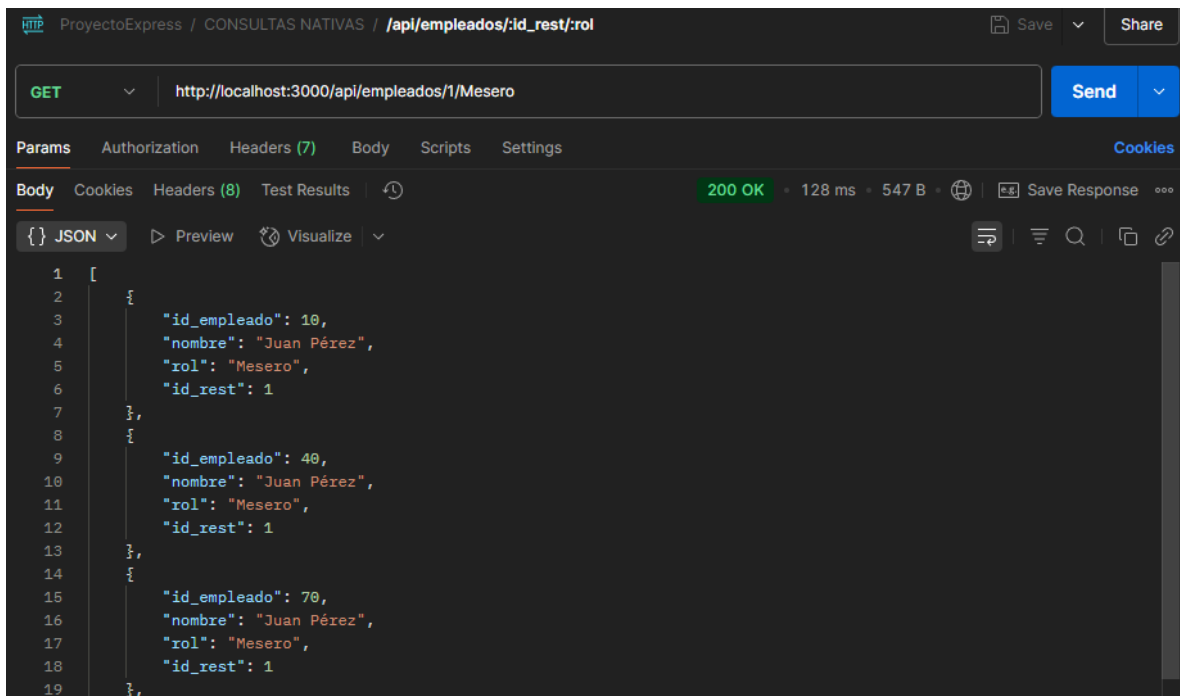
Parámetros (query string):- `id_rest`: ID del restaurante.- `rol`: Rol del empleado (ej: "chef", "mesero", "admin").

**Consulta SQL:**

SELECT \*FROM empleado WHERE id\_rest = CAST(\$1 AS INT) AND LOWER(rol) = LOWER(\$2);

### Proceso de la consulta SQL

- Se seleccionan todos los campos de la tabla empleado.
- Se filtra por el id\_rest especificado.
- Se convierte el campo rol a minúsculas con LOWER() para comparar sin distinción de mayúsculas/minúsculas.
- Se usa CAST(\$1 AS INT) para asegurar que id\_rest sea tratado como entero.



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

- URL:** `http://localhost:3000/api/empleados/1/Mesero`
- Method:** GET
- Status:** 200 OK
- Response Body (JSON):**

```
[
  {
    "id_empleado": 10,
    "nombre": "Juan Pérez",
    "rol": "Mesero",
    "id_rest": 1
  },
  {
    "id_empleado": 40,
    "nombre": "Juan Pérez",
    "rol": "Mesero",
    "id_rest": 1
  },
  {
    "id_empleado": 70,
    "nombre": "Juan Pérez",
    "rol": "Mesero",
    "id_rest": 1
  }
]
```