

## PARCIAL II CORTE

Base de Datos Masivas

Juan David Moreno Pico

ID:852417

Docente. Alex Matallana

Ingeniería de Sistemas.

Corporación Universitaria Uniminuto.

Zipaquirá

2025

## INICIAMOS EL PROYECTO CON NPM

```
PS C:\Users\juand\OneDrive\Escritorio\PARCIAL2> npm init -y  
Wrote to C:\Users\juand\OneDrive\Escritorio\PARCIAL2\package.json:
```

```
PS C:\Users\juand\OneDrive\Escritorio\PARCIAL2> npm i express pg dotenv cors
```

## VAMOS A SUPABASE Y CREAMOS LA BASE DE DATOS Y LAS TABLAS

```
1 create database parcial;
```

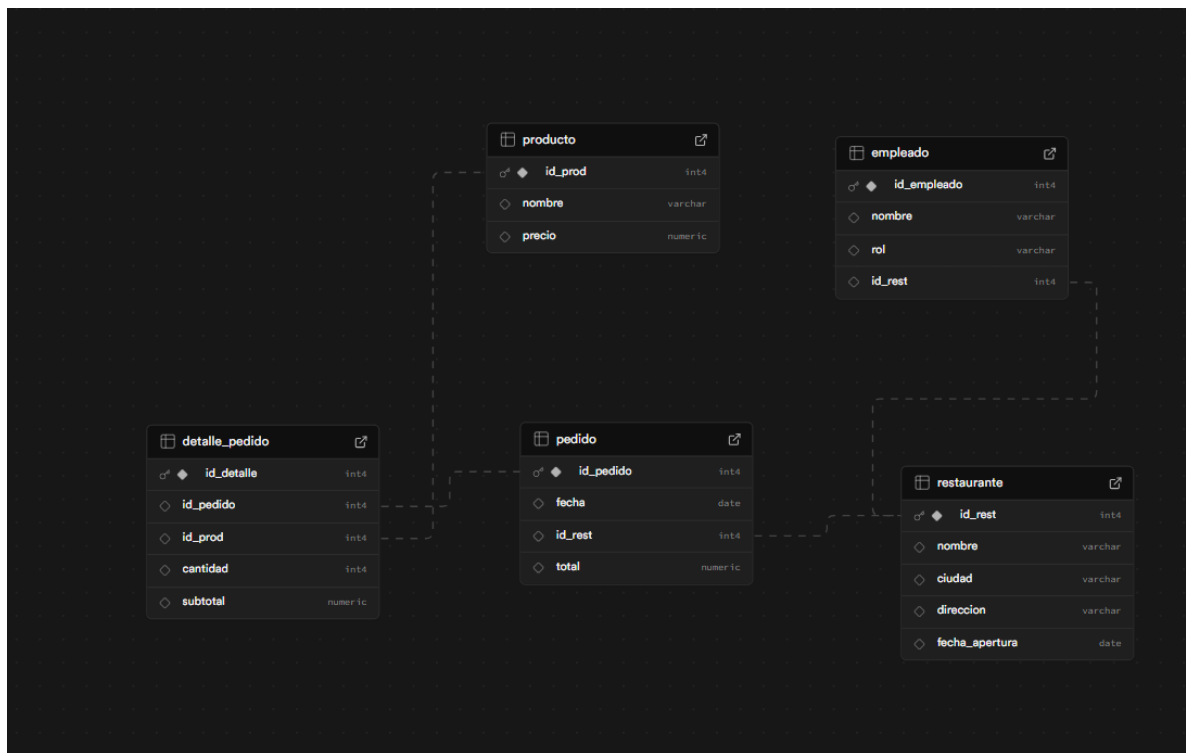
```
CREATE TABLE restaurante (  
  id_rest INT PRIMARY KEY,  
  nombre VARCHAR(100),  
  ciudad VARCHAR(100),  
  direccion VARCHAR(150),  
  fecha_apertura DATE  
);  
  
CREATE TABLE empleado (  
  id_empleado INT PRIMARY KEY,  
  nombre VARCHAR(100),  
  rol VARCHAR(50),  
  id_rest INT,  
  FOREIGN KEY (id_rest) REFERENCES restaurante(id_rest) ON DELETE CASCADE  
);  
  
CREATE TABLE producto (  
  id_prod INT PRIMARY KEY,  
  nombre VARCHAR(100),  
  precio NUMERIC(10,2)  
);
```

```

16 CREATE TABLE producto (
17     id_prod INT PRIMARY KEY,
18     nombre VARCHAR(100),
19     precio NUMERIC(10,2)
20 );
21 CREATE TABLE pedido (
22     id_pedido INT PRIMARY KEY,
23     fecha DATE,
24     id_rest INT,
25     total NUMERIC(10,2),
26     FOREIGN KEY (id_rest) REFERENCES restaurante(id_rest) ON DELETE CASCADE
27 );
28 CREATE TABLE detalle_pedido (
29     id_detalle INT PRIMARY KEY,
30     id_pedido INT,
31     id_prod INT,
32     cantidad INT,
33     subtotal NUMERIC(10,2),
34     FOREIGN KEY (id_pedido) REFERENCES pedido(id_pedido) ON DELETE CASCADE,
35     FOREIGN KEY (id_prod) REFERENCES producto(id_prod) ON DELETE CASCADE
36 );

```

VALIDAMOS QUE ESTE CORRECTA LAS TABLAS Y SUS RELACIONES.



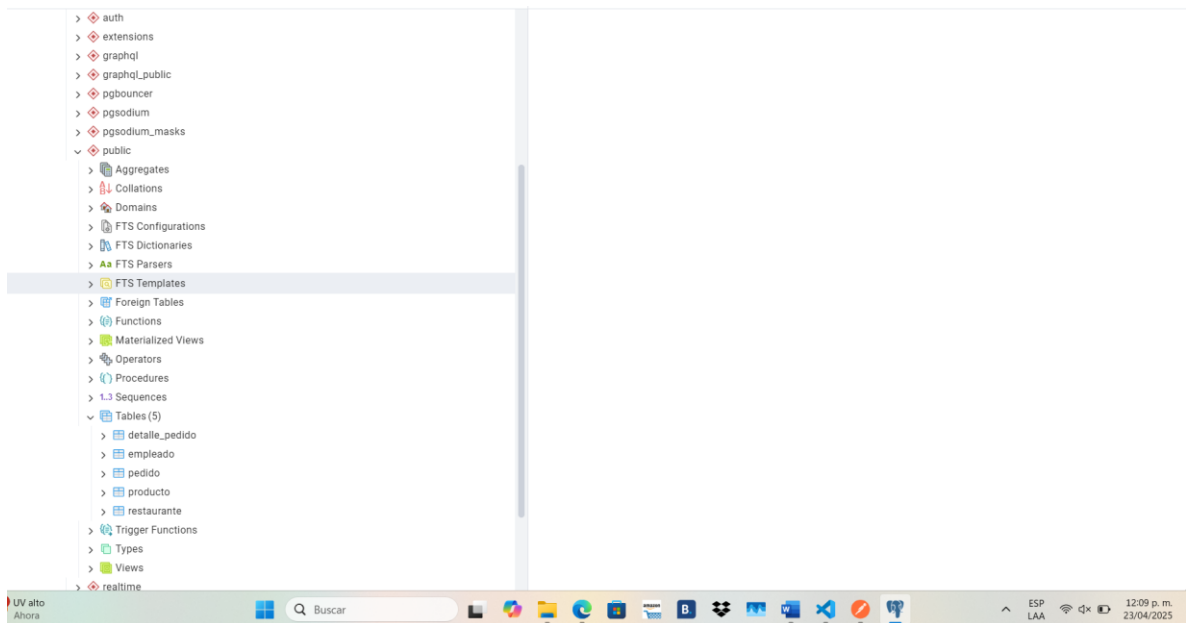
## INSERTAMOS REGISTROS

```
1 INSERT INTO producto (id_prod, nombre, precio) VALUES
2 (1, 'Hamburguesa Clásica', 8.99),
3 (2, 'Pizza Margarita', 12.50),
4 (3, 'Ensalada César', 6.75),
5 (4, 'Sopa de Tomate', 4.99),
6 (5, 'Pasta Alfredo', 10.25),
7 (6, 'Tacos de Pollo', 7.50),
8 (7, 'Sándwich de Pavo', 5.99),
9 (8, 'Helado de Vainilla', 3.25),
10 (9, 'Refresco de Cola', 2.00),
11 (10, 'Agua Mineral', 1.50),
12 (11, 'Cerveza Artesanal', 5.75),
13 (12, 'Tarta de Manzana', 4.50),
14 (13, 'Café Americano', 2.25),
15 (14, 'Batido de Fresa', 3.75),
16 (15, 'Alitas Picantes', 9.99),
17 (16, 'Nachos con Queso', 6.25),
18 (17, 'Sushi Roll', 11.50),
19 (18, 'Pollo a la Parrilla', 14.99),
20 (19, 'Lasagna', 10.75),
21 (20, 'Tiramisu', 5.25),
22 (21, 'Burrito de Carne', 8.50),
23 (22, 'Pan de Azúcar', 3.99)
```

Results Chart Export ✓ ⋮ Source Primary Database Role postgres Run CTRL ↵

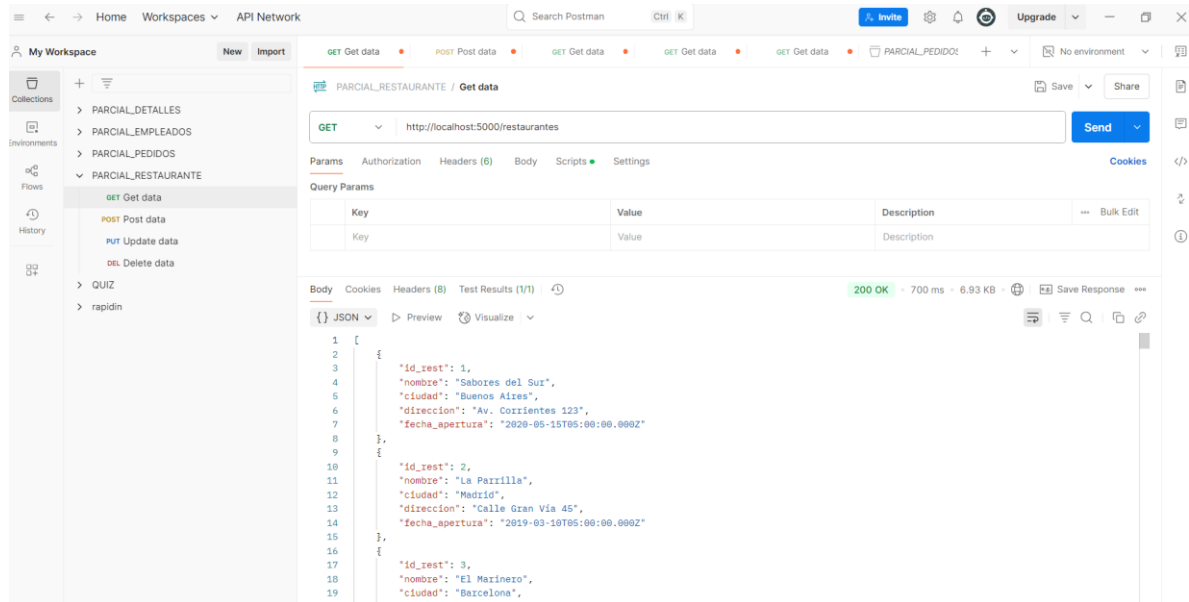
Success. No rows returned

## CONEXIÓN CON PGADMIN4

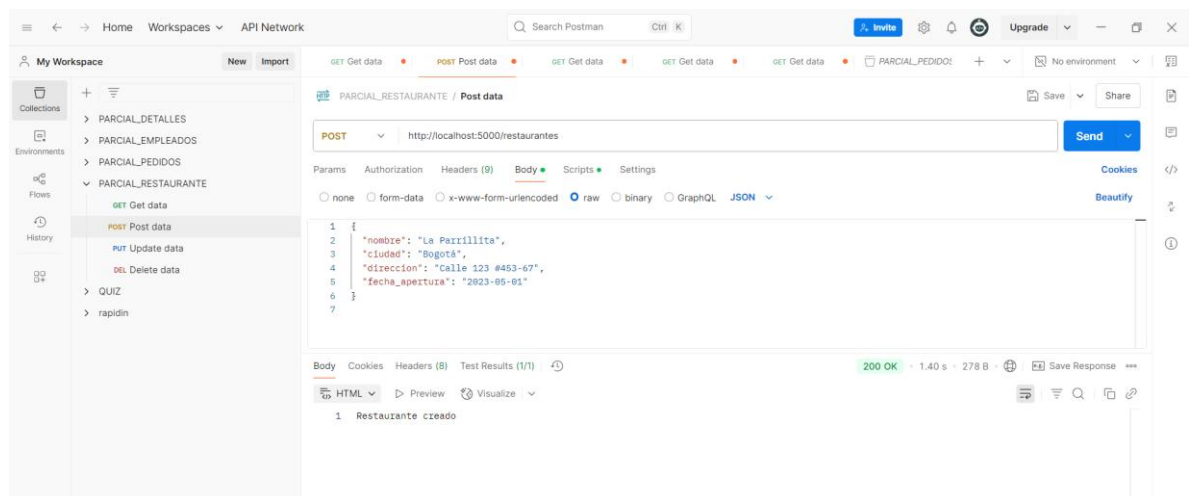


# PROBAMOS LAS APIS EN POSTMAN

## METODO GET EN TABLA RESTAURANTE:



## METODO POST EN RESTAURANTE



## METODO DELETE EN RESTAURANTE

Postman interface showing a DELETE request to `http://localhost:5000/restaurantes/50`. The request is configured with the following headers:

Key	Value	Description
<input checked="" type="checkbox"/> Content-Type	application/json	
Key	Value	Description

The response is a 200 OK status, indicating the restaurant was successfully deleted. The response body is:

```
1 Restaurante eliminado
```

## METODO PUT EN RESTAURANTE

Postman interface showing a PUT request to `http://localhost:5000/restaurantes/50`. The request is configured with the following body (JSON):

```
1 {
2   "nombre": "Elpepe",
3   "direccion": "CALLE #3-56-67, CHIA",
4   "telefono": "3142324484"
5 }
```

The response is a 200 OK status, indicating the restaurant was successfully updated. The response body is:

```
1 Restaurante actualizado
```

## METODO GET EN TABLA PEDIDO

Postman interface showing a GET request to `http://localhost:5000/pedidos`. The response is a JSON array of 4 objects, each representing a pedido (order) with fields: `fecha`, `id_rest`, `id_pedido`, and `total`.

```
4 {
5   "fecha": "2023-01-05T05:00:00.000Z",
6   "id_rest": 1,
7   "total": "45.75"
8 },
9 {
10  "id_pedido": 2,
11  "fecha": "2023-01-06T05:00:00.000Z",
12  "id_rest": 2,
13  "total": "32.50"
14 },
15 {
16  "id_pedido": 3,
17  "fecha": "2023-01-07T05:00:00.000Z",
18  "id_rest": 3,
19  "total": "28.99"
20 },
21 {
22  "id_pedido": 4,
```

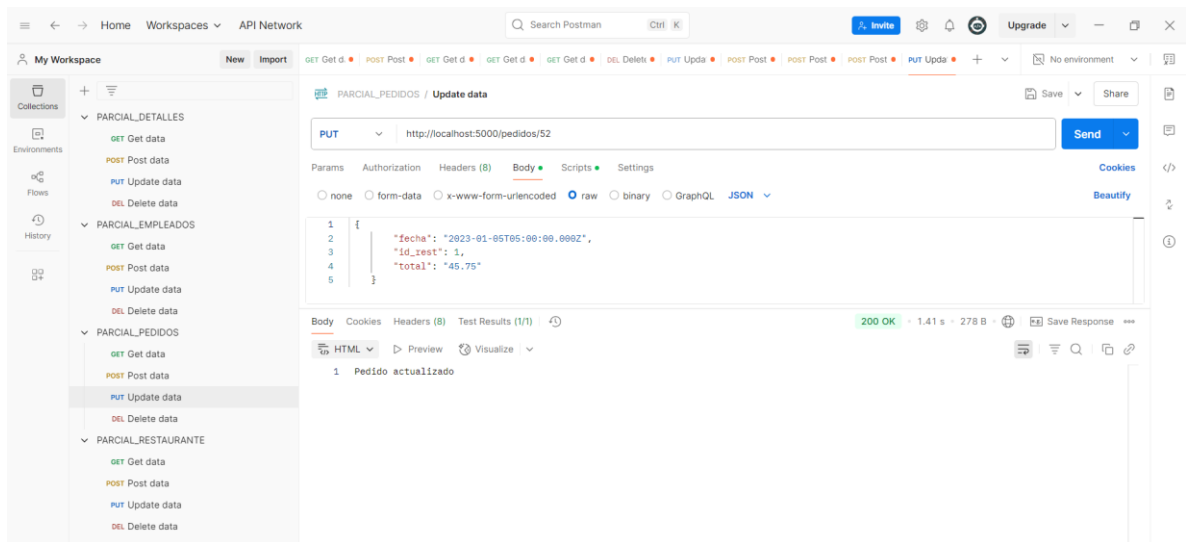
## METODO POST EN TABLA PEDIDO

Postman interface showing a POST request to `http://localhost:5000/pedidos`. The request body is a JSON object with fields: `fecha`, `id_rest`, and `total`. The response is a 200 OK status with the message "Pedido creado".

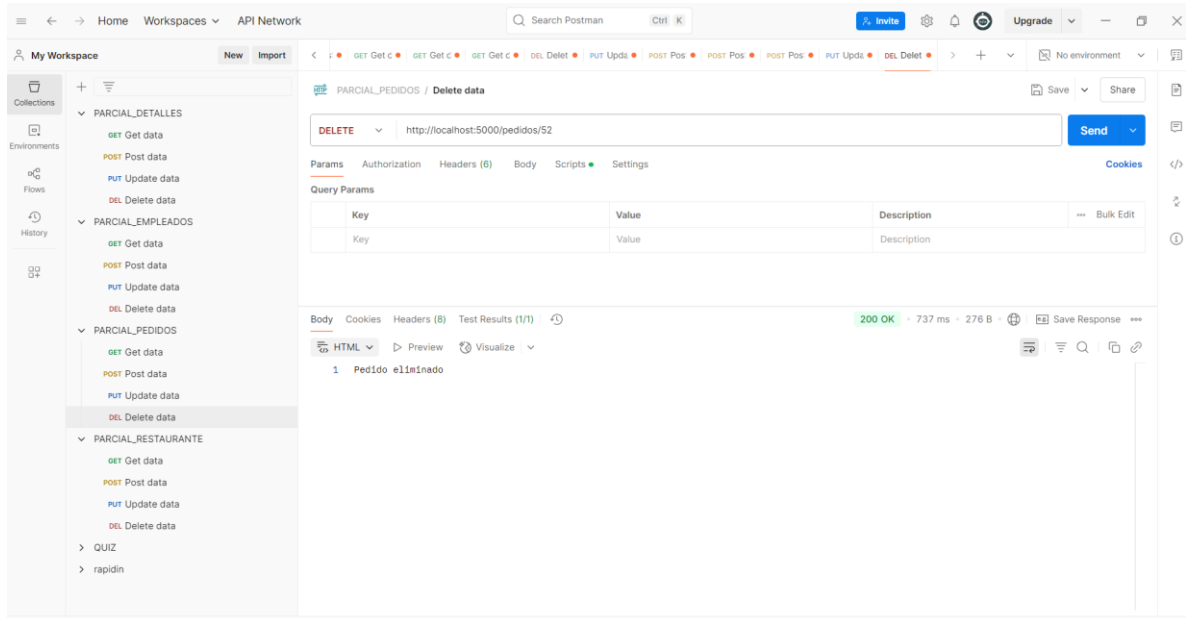
```
1 {
2   "fecha": "2019-01-05T05:00:00.000Z",
3   "id_rest": 52,
4   "total": "85.75"
5 }
6 }
```

Body: 1 Pedido creado

## METODO PUT EN TABLA PEDIDO

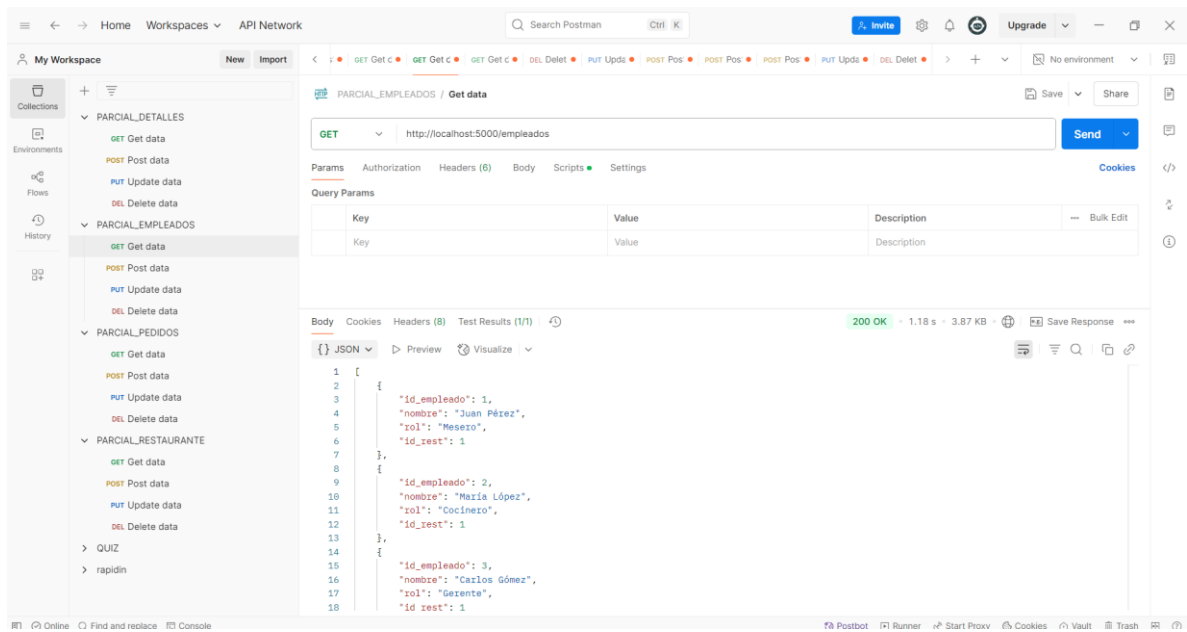


## METODO DELETE EN TABLA PEDIDO

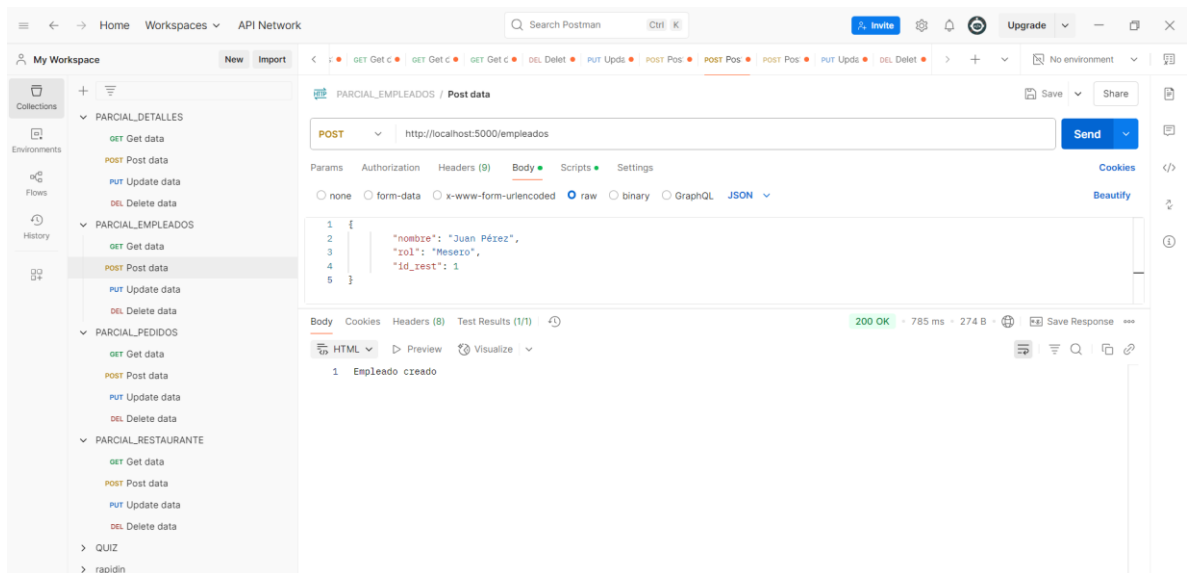


## METODO GET EN TABLA EMPLEADO

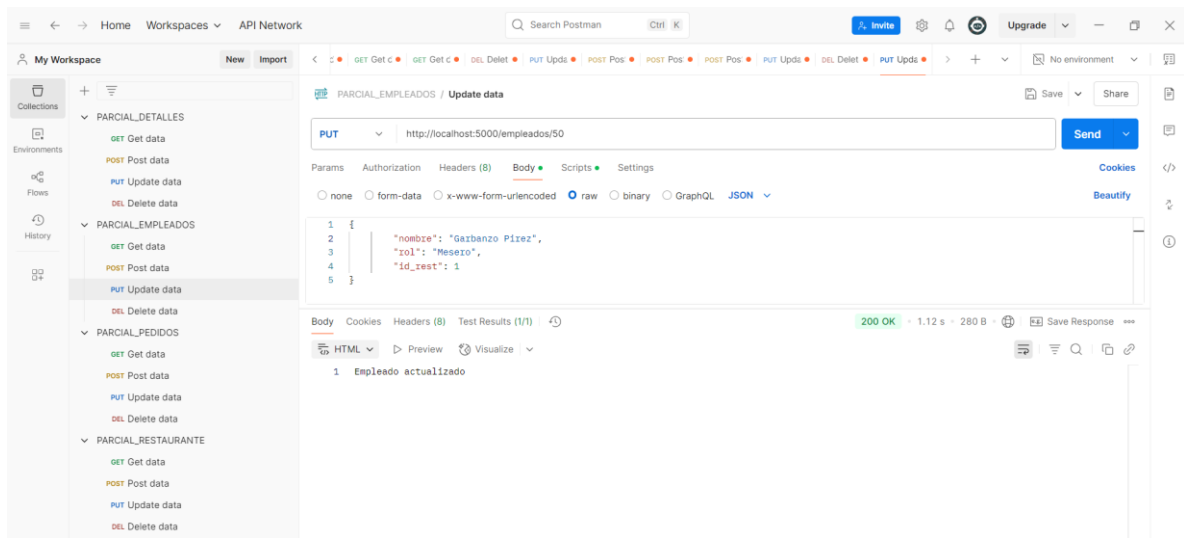




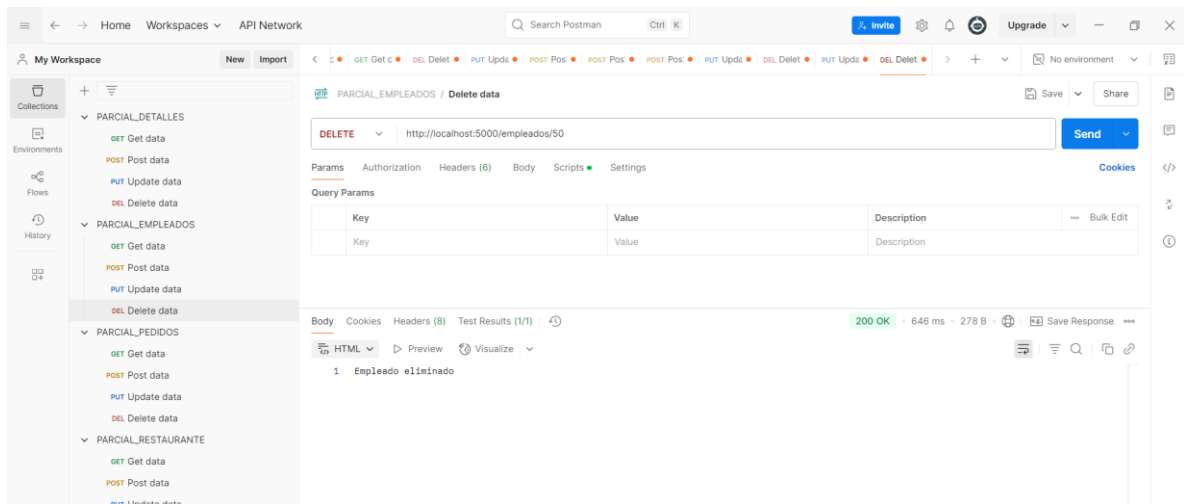
## METODO POST EN TABLA EMPLEADO



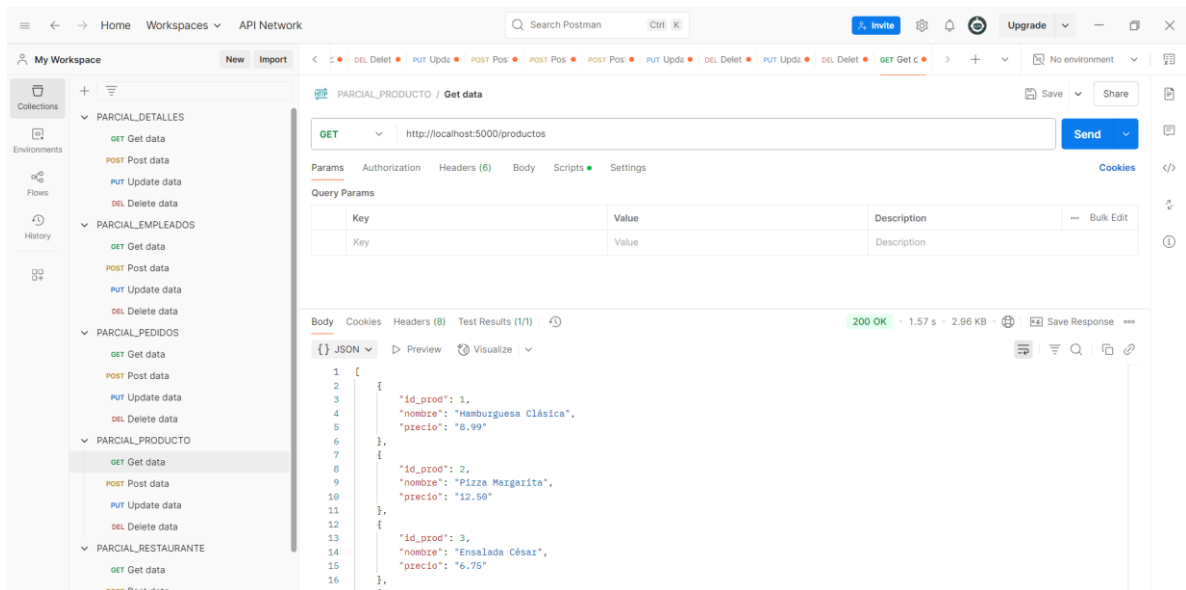
## METODO PUT EN TABLA EMPLEADO



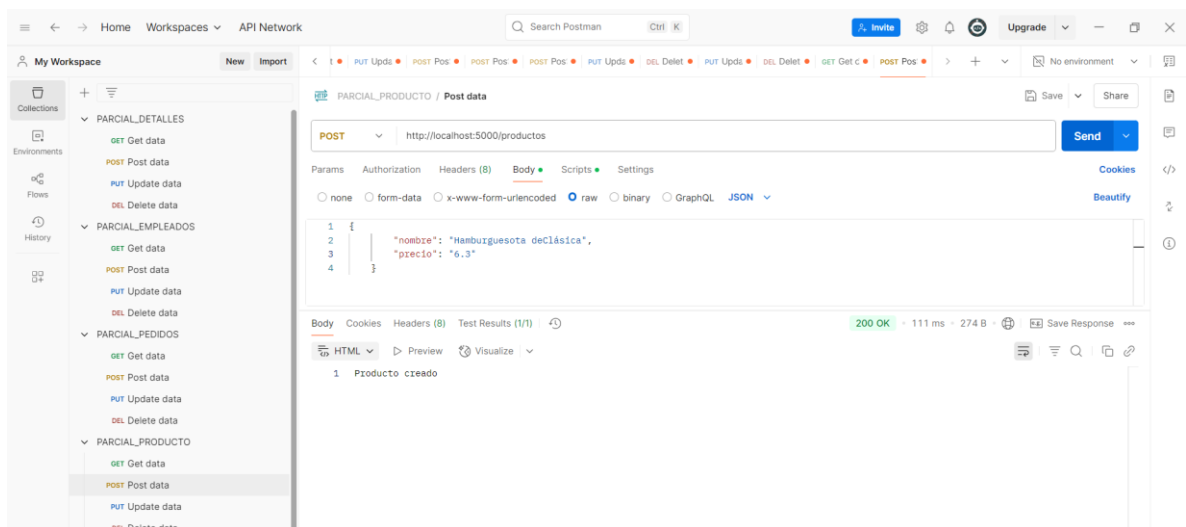
## METODO DELETE EN TABLA EMPLEADO



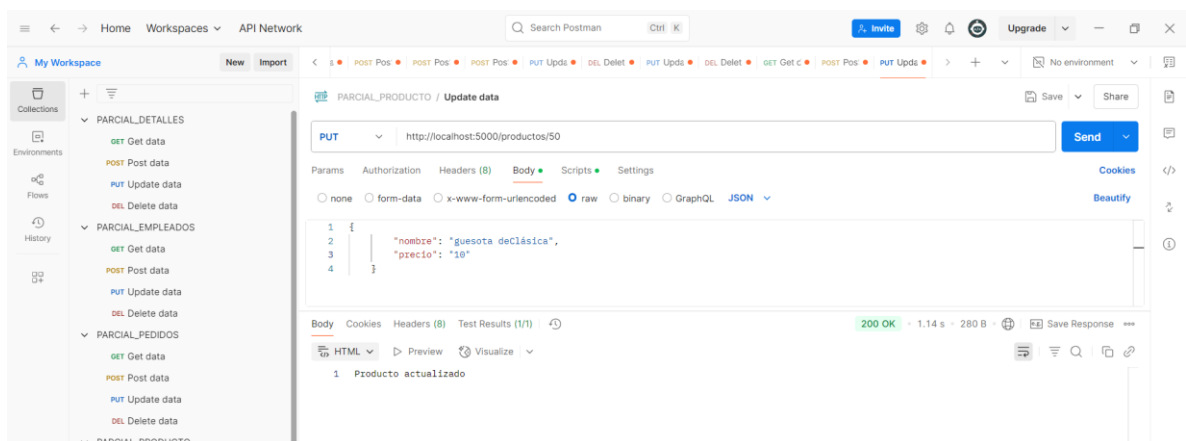
## METODO GET EN TABLA PRODUCTO



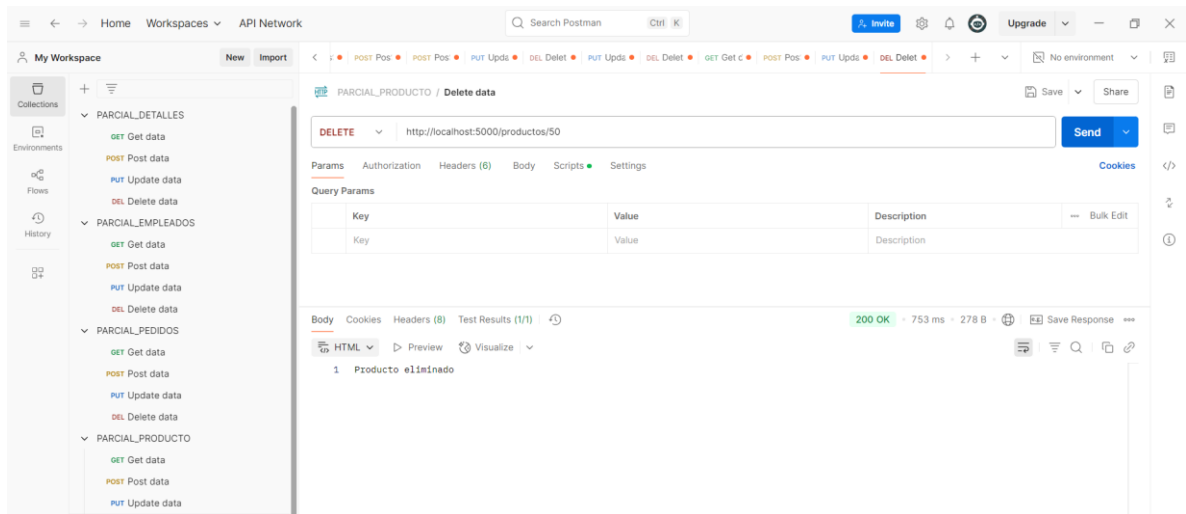
## METODO POST EN TABLA PRODUCTO



## METODO PUT EN TABLA PRODUCTO



## METODO DELETE EN TABLA PRODUCTO



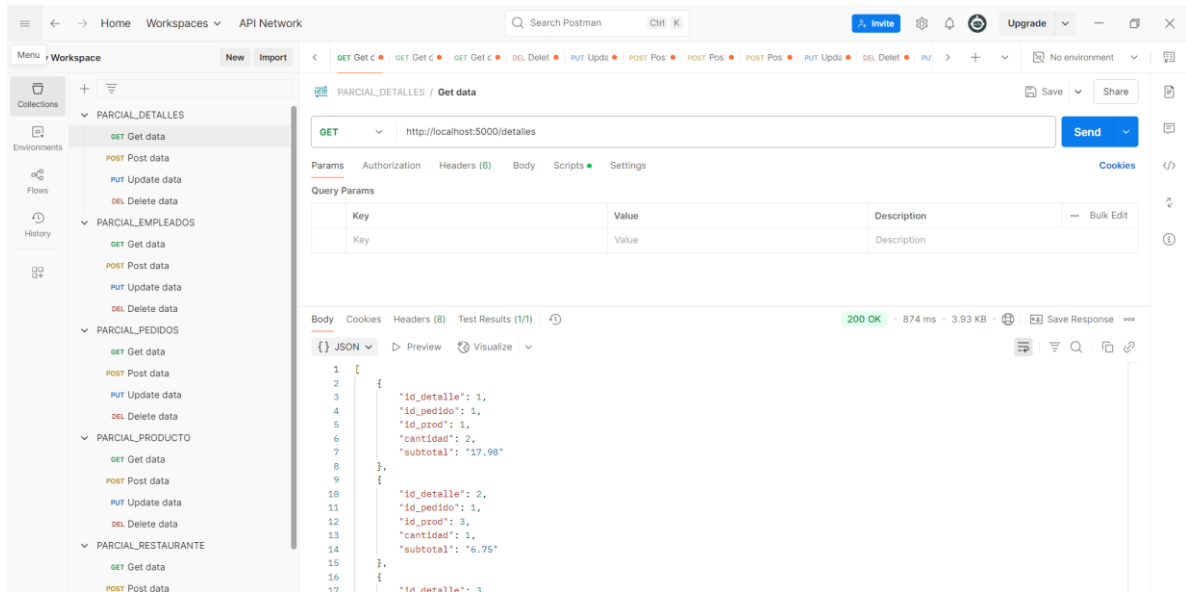
The screenshot shows the Postman interface with a DELETE request to `http://localhost:5000/productos/50`. The response is a 200 OK status with a response time of 753 ms and a body of "Producto eliminado".

Key	Value	Description
Key	Value	Description

Body: 200 OK - 753 ms - 278 B

1 Producto eliminado

## METODO GET EN TABLA DETALLE\_PEDIDO



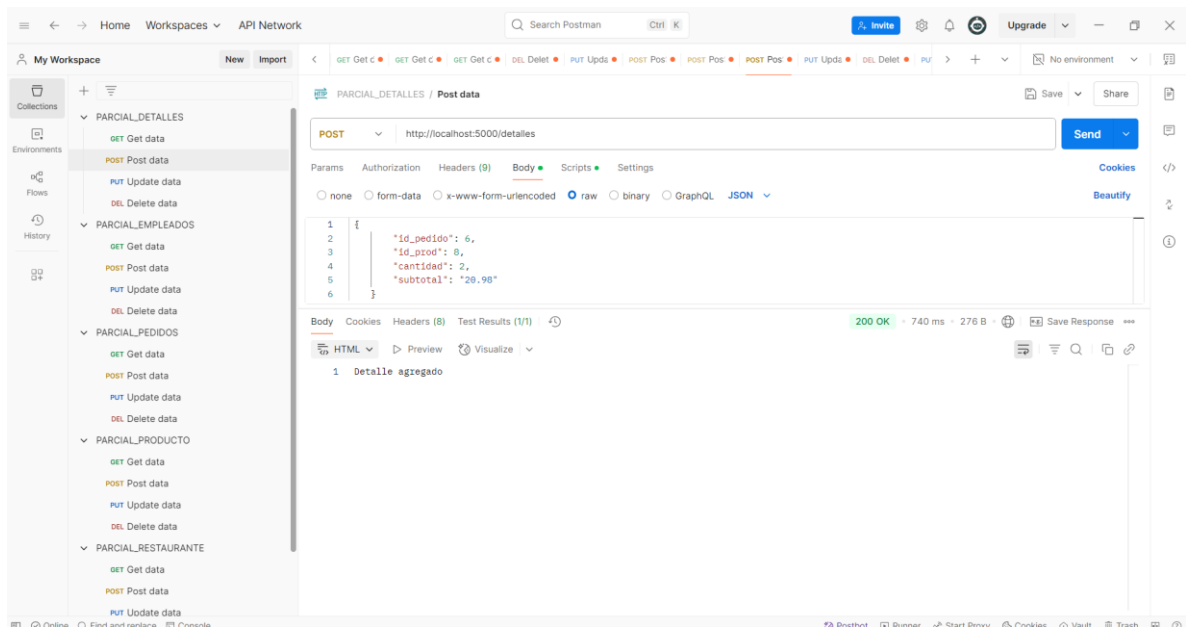
The screenshot shows the Postman interface with a GET request to `http://localhost:5000/detalles`. The response is a 200 OK status with a response time of 874 ms and a body of JSON data.

Key	Value	Description
Key	Value	Description

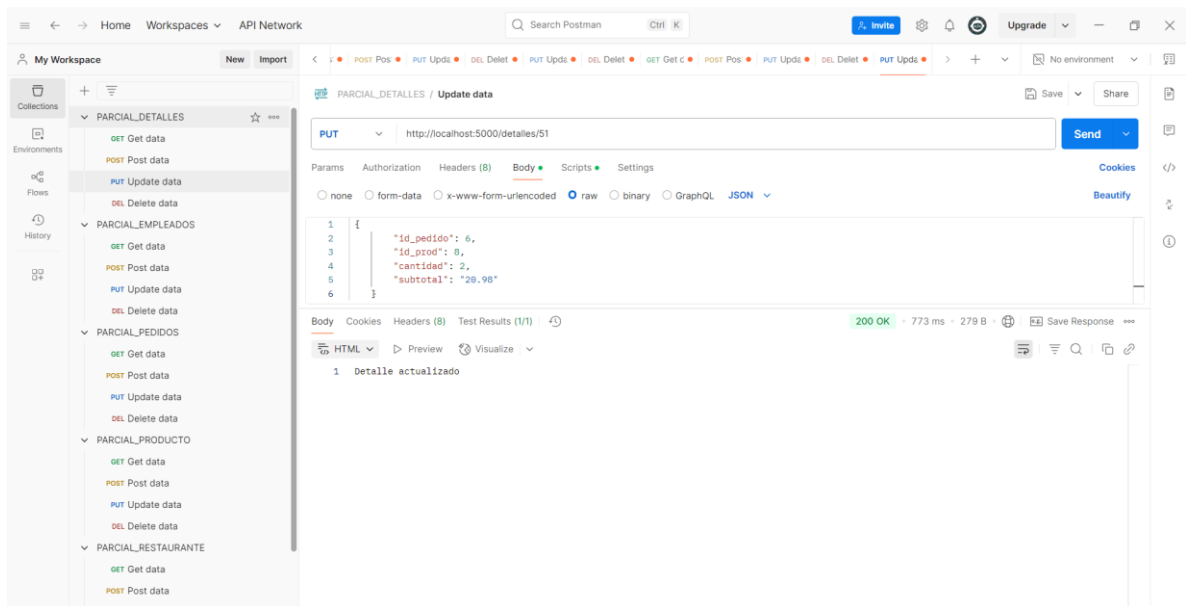
Body: 200 OK - 874 ms - 3.93 KB

```
1 {
2   {
3     "id_detalle": 1,
4     "id_pedido": 1,
5     "id_prod": 1,
6     "cantidad": 2,
7     "subtotal": "17.98"
8   },
9   {
10    "id_detalle": 2,
11    "id_pedido": 1,
12    "id_prod": 3,
13    "cantidad": 1,
14    "subtotal": "6.75"
15  },
16  {
17    "id_detalle": 3,
```

## METODO POST EN TABLA DETALLE\_PEDIDO



## METODO PUT EN TABLA DETALLE\_ PEDIDO



## METODO DELETE EN TABLA DETALLE\_ PEDIDO

Postman interface showing a DELETE request to `http://localhost:5000/detalles/51`. The request is successful, returning a 200 OK status. The response body is HTML, displaying "1 Detalle eliminado".

**Query Params**

Key	Value	Description
Key	Value	Description

**Body**

1 Detalle eliminado

## OBTENER TODOS LOS PRODUCTOS DE UN PEDIDO ESPECÍFICO

Postman interface showing a GET request to `http://localhost:5000/pedidos/3/productos`. The request is successful, returning a 200 OK status. The response body is JSON, displaying a list of products for a specific order.

**Query Params**

Key	Value	Description
Key	Value	Description

**Body**

```
{
  "id_pedido": 3,
  "productos": [
    {
      "id_producto": 4,
      "nombre": "Sopa de Tomate",
      "cantidad": 2,
      "subtotal": "9.90"
    },
    {
      "id_producto": 7,
      "nombre": "Sándwich de Pavo",
      "cantidad": 1,
      "subtotal": "5.99"
    }
  ]
}
```

## OBTENER LOS PRODUCTOS MÁS VENDIDOS (MÁS DE X UNIDADES)

The screenshot shows the Postman interface with a workspace named 'My Workspace'. The left sidebar lists collections, including 'PARCIAL\_CONSULTAS\_ADICIONALES'. The main panel displays a GET request to the URL 'http://localhost:5000/productos/mas-vendidos/3'. The response is a 200 OK status with a response time of 104 ms and a body of 330 B. The response body is a JSON object:

```
{  "id_prod": 9,  "nombre": "Refresco de Cola",  "total_vendido": 4}
```

## OBTENER EL TOTAL DE VENTAS POR RESTAURANTE

The screenshot shows the Postman interface with a workspace named 'My Workspace'. The left sidebar lists collections, including 'PARCIAL\_CONSULTAS\_ADICIONALES'. The main panel displays a GET request to the URL 'http://localhost:5000/ventas/restaurantes'. The response is a 200 OK status with a response time of 1.18 s and a body of 3.33 KB. The response body is a JSON array of objects, each representing a restaurant's sales data:

```
[  {    "id_rest": 52,    "nombre": "La Parrillita",    "total_ventas": 89.75  },  {    "id_rest": 13,    "nombre": "El Fogón",    "total_ventas": 75.25  },  {    "id_rest": 24,    "nombre": "La Trattoria",    "total_ventas": 62.75  }]
```

## OBTENER LOS PEDIDOS REALIZADOS EN UNA FECHA ESPECÍFICA

The screenshot shows the Postman interface with a workspace named 'My Workspace'. The left sidebar displays a collection of API endpoints under the folder 'PARCIAL\_CONSULTAS\_ADICIONALES'. The selected endpoint is 'DELETE data' with a GET method. The URL is 'http://localhost:5000/pedidos/fecha/2023-01-06'. The 'Send' button is visible. The response is a 200 OK status, indicating a successful request. The response body is a JSON array containing one object with the following details:

Key	Value	Description
id_pedido	2	
fecha	"2023-01-06T05:00:00.000Z"	
id_rest	2	
total	"32.58"	

## OBTENER LOS EMPLEADOS POR ROL EN UN RESTAURANTE

The screenshot shows the Postman interface with a workspace named 'My Workspace'. The left sidebar displays a collection of API endpoints under the folder 'PARCIAL\_CONSULTAS\_ADICIONALES'. The selected endpoint is 'New Request' with a GET method. The URL is 'http://localhost:5000/empleados/rol/3/Cocinero'. The 'Send' button is visible. The response is a 200 OK status, indicating a successful request. The response body is a JSON array containing one object with the following details:

Key	Value	Description
id_empleado	7	
nombre	"Pedro Diaz"	
rol	"Cocinero"	