PARCIAL API REST CON NODE.JS – EXPRESS - SUPABASE

JORGE ANDRES CASTRO PACHON

ID:827833

BASES DE DATOS MASIVAS

DOCENTE:
WILLIAM ALEXANDER MATALLANA

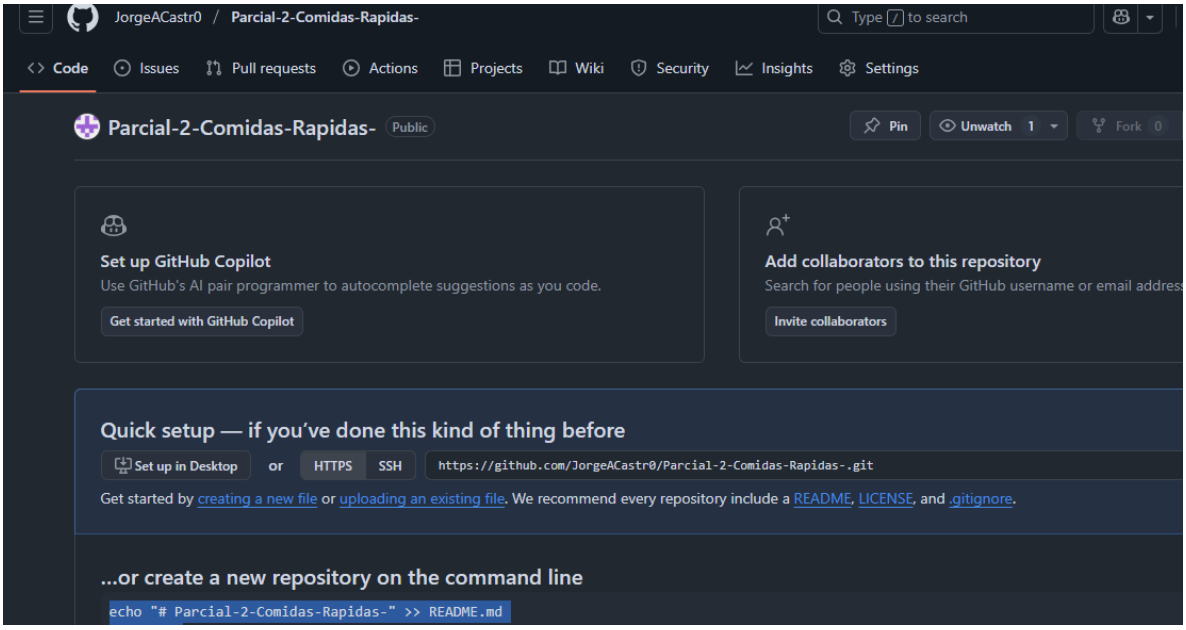
CORPORACIÓN UNIVERSITARIA MINUTO DE

DIOS

INGENERIA DE SISTEMAS

ZIPQUIRÁ, CUNDINAMARCA 2025

1. Creamos repositorio en github



2. Enlazamos la carpeta con GIT

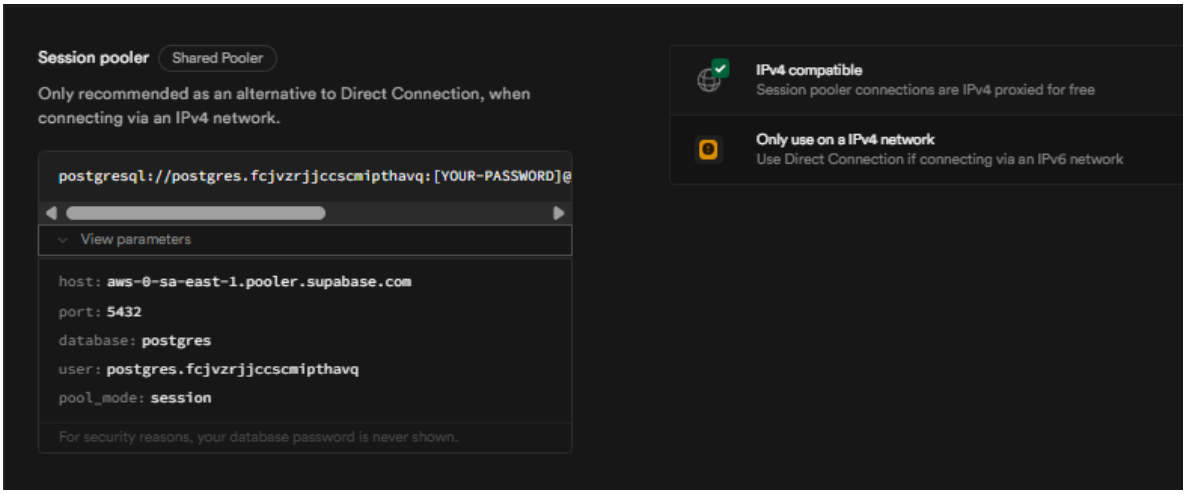
```
SDA-48-239@LAPTOP-IPD2JEJG MINGW64 ~/OneDrive - uniminuto.edu/6to Semestre/Bases
 de datos Masivas/Parcial 2do Corte
$ echo "# Parcial-2-Comidas-Rapidas-" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/JorgeACastr0/Parcial-2-Comidas-Rapidas-
.git
git push -u origin main
Initialized empty Git repository in C:/Users/SDA-48-239/OneDrive - uniminuto.edu
/6to Semestre/Bases de datos Masivas/Parcial 2do Corte/.git/
warning: in the working copy of 'README.md', LF will be replaced by CRLF the nex
t time Git touches it
[master (root-commit) c349166] first commit
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 242 bytes | 121.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/JorgeACastr0/Parcial-2-Comidas-Rapidas-.git
 * [new branch]      main -> main
```

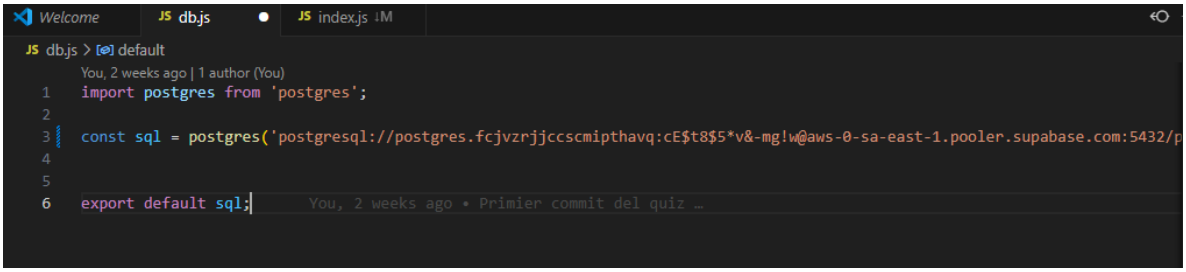
3. Creamos archivos e instalamos dependencias

```
PS C:\Users\SDA-48-239\OneDrive - uniminuto.edu\6to Semestre\Bases de datos Masivas\Parcial 2do Corte> npm install express postgre dotenv cors

added 84 packages in 12s

15 packages are looking for funding
  run `npm fund` for details
```

```
PS C:\Users\SDA-48-239\OneDrive - uniminuto.edu\6to Semestre\Bases de datos Masivas\Parcial 2do Corte> npm init -y
Wrote to C:\Users\SDA-48-239\OneDrive - uniminuto.edu\6to Semestre\Bases de datos Masivas\Parcial 2do Corte\package.json:

{
  "dependencies": {
    "cors": "^2.8.5",
    "dotenv": "^16.5.0",
    "express": "^5.1.0",
    "postgre": "^0.1.8"
  },
  "name": "parcial-2do-corte",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "devDependencies": {},
  "scripts": {
```

4. Creamos el proyecto en supabase y enlazamos la base de datos



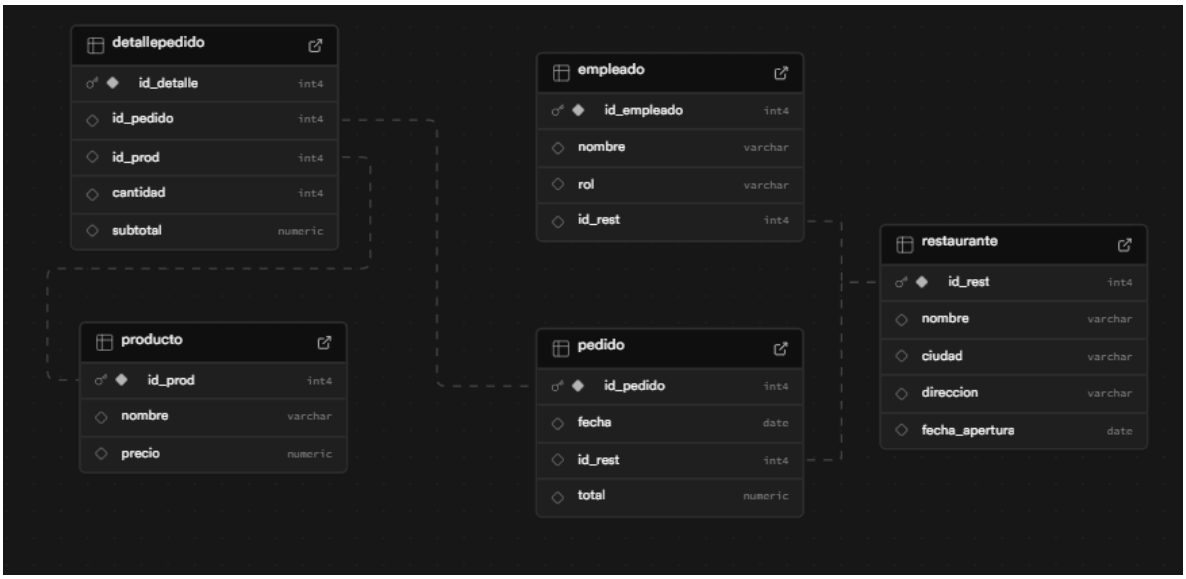


5. Ahora crearemos las tablas de la base de datos según la indicación

```sql
CREATE TABLE Restaurante (
    id_rest SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    ciudad VARCHAR(100),
    direccion VARCHAR(150),
    fecha_apertura DATE
);

CREATE TABLE Producto (
    id_prod SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    precio NUMERIC(10,2)
);

CREATE TABLE Empleado (
    id_empleado SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    rol VARCHAR(50),
    id_rest INT REFERENCES Restaurante(id_rest)
);
```

```sql
CREATE TABLE Pedido (
    id_pedido SERIAL PRIMARY KEY,
    fecha DATE,
    id_rest INT REFERENCES Restaurante(id_rest),
    total NUMERIC(10,2)
);

CREATE TABLE DetallePedido (
    id_detalle SERIAL PRIMARY KEY,
    id_pedido INT REFERENCES Pedido(id_pedido),
    id_prod INT REFERENCES Producto(id_prod),
    cantidad INT,
    subtotal NUMERIC(10,2)
);
```

Diagrama ER:



6. Ahora insertamos registros en la base de datos

Tabla restaurante:

```sql
INSERT INTO Restaurante (nombre, ciudad, direccion, fecha_apertura) VALUES
('Delicias Rápidas', 'Bogotá', 'Calle 123 #45-67', '2020-01-15'),
('Sabor Criollo', 'Medellín', 'Carrera 10 #20-30', '2021-03-12'),
('Burgertown', 'Cali', 'Avenida 3N #45-12', '2019-06-20'),
('Comida Express', 'Barranquilla', 'Calle 30 #15-25', '2022-02-01'),
('La Parrilla', 'Cartagena', 'Cra 7 #54-88', '2018-11-05'),
('Arepas Ya', 'Manizales', 'Calle 50 #22-13', '2020-08-17'),
('Tostones Grill', 'Pereira', 'Cra 4 #32-20', '2021-09-25'),
('Papas y Algo Más', 'Bucaramanga', 'Av. Santander #19-20', '2019-03-30'),
('Comida Urbana', 'Cúcuta', 'Cra 9 #11-11', '2022-07-10'),
('Rico al Paso', 'Ibagué', 'Calle 21 #14-17', '2023-01-05');

```

Tabla Empleado:

```sql
INSERT INTO Empleado (nombre, rol, id_rest) VALUES
-- Restaurante 1
('Carlos Pérez', 'Cocinero', 1),
('Ana Gómez', 'Cajero', 1),
('Luis Rojas', 'Mesero', 1),
('Paola Ruiz', 'Administrador', 1),
('David Torres', 'Domiciliario', 1),
-- Restaurante 2
('Camila Rodríguez', 'Cocinero', 2),
('Juan Esteban', 'Cajero', 2),
('Marta Jiménez', 'Mesero', 2),
('Diego Morales', 'Administrador', 2),
('Lina Campos', 'Domiciliario', 2),
-- Restaurante 3
('José Herrera', 'Cocinero', 3),
('Claudia Arias', 'Cajero', 3),
('Ricardo Lozano', 'Mesero', 3),
('Valentina Rico', 'Administrador', 3),
('Sebastián Quintero', 'Domiciliario', 3),
-- Restaurante 4
('Sandra Beltrán', 'Cocinero', 4),
('Julián Ortiz', 'Cajero', 4),
('Adriana Luna', 'Mesero', 4),
('Andrés Suárez', 'Administrador', 4),
('Diana Cortés', 'Domiciliario', 4),
-- Restaurante 5
('Nicolás Vargas', 'Cocinero', 5),
('María Cárdenas', 'Cajero', 5),
('Fernando Salas', 'Mesero', 5),
('Isabel Naranjo', 'Administrador', 5),
```

Tabla productos:

```sql
INSERT INTO Producto (nombre, precio) VALUES
('Hamburguesa Clásica', 12000),
('Hamburguesa Doble', 15000),
('Hamburguesa BBQ', 14500),
('Perro Caliente', 9000),
('Salchipapa', 10000),
('Arepa Rellena', 8000),
('Tacos Mixtos', 13000),
('Chorizo Santarrosano', 7000),
('Papas Criollas', 5000),
('Nachos con Queso', 9500),
('Choripan', 10500),
('Empanadas x3', 6000),
('Alitas BBQ x5', 11000),
('Tostones', 7500),
('Arepa con Chorizo', 8500),
('Bebida Gaseosa 350ml', 3000),
('Jugos Naturales', 4000),
('Agua en Botella', 2500),
('Cerveza Nacional', 5000),
('Cerveza Importada', 8000),
('Postre de Tres Leches', 6500),
('Brownie con Helado', 7000),
('Helado en Copa', 5000),
('Pizza Personal', 12000),
('Pizza Mediana', 18000),
('Pizza Grande', 25000),
('Combo Hamburguesa', 18000),
('Combo Perro Caliente', 15000),
```

Tabla pedidos:

```sql
INSERT INTO Pedido (fecha, id_rest, total) VALUES
('2024-04-01', 1, 45000),
('2024-04-01', 2, 32000),
('2024-04-02', 3, 25000),
('2024-04-02', 4, 54000),
('2024-04-03', 5, 39000),
('2024-04-03', 6, 46000),
('2024-04-04', 7, 31000),
('2024-04-04', 8, 52000),
('2024-04-05', 9, 48000),
('2024-04-05', 10, 36000),
('2024-04-06', 1, 37000),
('2024-04-06', 2, 41000),
('2024-04-07', 3, 29000),
('2024-04-07', 4, 47000),
('2024-04-08', 5, 50000),
('2024-04-08', 6, 28000),
('2024-04-09', 7, 33000),
('2024-04-09', 8, 31000),
('2024-04-10', 9, 49000),
('2024-04-10', 10, 35000),
('2024-04-11', 1, 31000),
('2024-04-11', 2, 27000),
('2024-04-12', 3, 51000),
('2024-04-12', 4, 34000),
('2024-04-13', 5, 46000),
('2024-04-13', 6, 42000),
('2024-04-14', 7, 39000),
('2024-04-14', 8, 37000),
('2024-04-15', 9, 53000),
```

Tabla detallepedido:

```sql
INSERT INTO DetallePedido (id_pedido, id_prod, cantidad, subtotal) VALUES
(1, 1, 2, 24000),
(2, 5, 1, 10000),
(3, 12, 3, 18000),
(4, 3, 2, 29000),
(5, 20, 4, 20000),
(6, 7, 2, 26000),
(7, 8, 3, 21000),
(8, 15, 2, 17000),
(9, 23, 1, 5000),
(10, 27, 2, 36000),
(11, 2, 2, 30000),
(12, 30, 3, 27000),
(13, 11, 2, 21000),
(14, 13, 3, 33000),
(15, 18, 2, 5000),
(16, 25, 1, 25000),
(17, 6, 2, 16000),
(18, 4, 2, 18000),
(19, 10, 2, 19000),
(20, 9, 2, 10000),
(21, 21, 2, 13000),
(22, 16, 2, 6000),
(23, 22, 1, 5000),
(24, 24, 2, 24000),
(25, 26, 1, 25000),
```

7. Ahora configuraremos el archivo index.js para que escuche por el puerto 3000 y traiga las demás dependencias

```
JS index.js > ...
1  import express from 'express';
2  import sql from './db.js';
3
4  const app = express();
5
6  app.use(express.json());
7
8  app.use(express.urlencoded({extended:true}));
9
10 app.get('/api/prueba' , (req, res) => {
11
12     res.send('LA API FUNCIONA');
13 });
14
15 //Crear puerto de conexion del servidor
16 const PORT = 3000;
17
18 //La conexion la va a escuchar por el puerto 3000 y si
19 app.listen(PORT, ()=>{
20     console.log('El servidor esta corriendo');
21
22 });
```

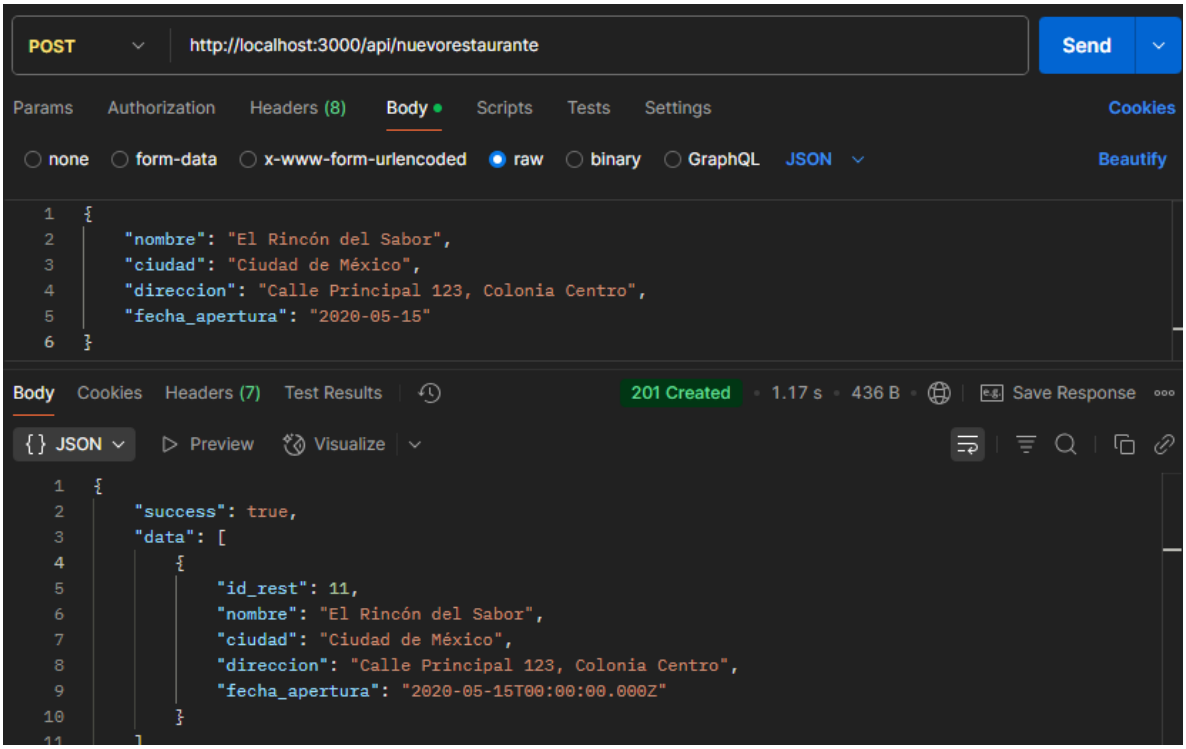
```
El servidor esta corriendo
```

8. Ahora creamos las APIS para hacer el CRUD

Aquí creamos insertamos los restaurantes

```
app.post('/api/nuevorestaurante', async (req, res) => {
    try {
      const { nombre, ciudad, direccion, fecha_apertura } = req.body;
      const result = await sql`
        INSERT INTO Restaurante (nombre, ciudad, direccion, fecha_apertura)
        VALUES (${nombre}, ${ciudad}, ${direccion}, ${fecha_apertura})
        RETURNING *;
      `;
      res.status(201).json({ success: true, data: result });
    } catch (error) {
      res.status(500).json({ success: false, message: 'Error al crear restaurante', details: error.message });
    }
  });
```

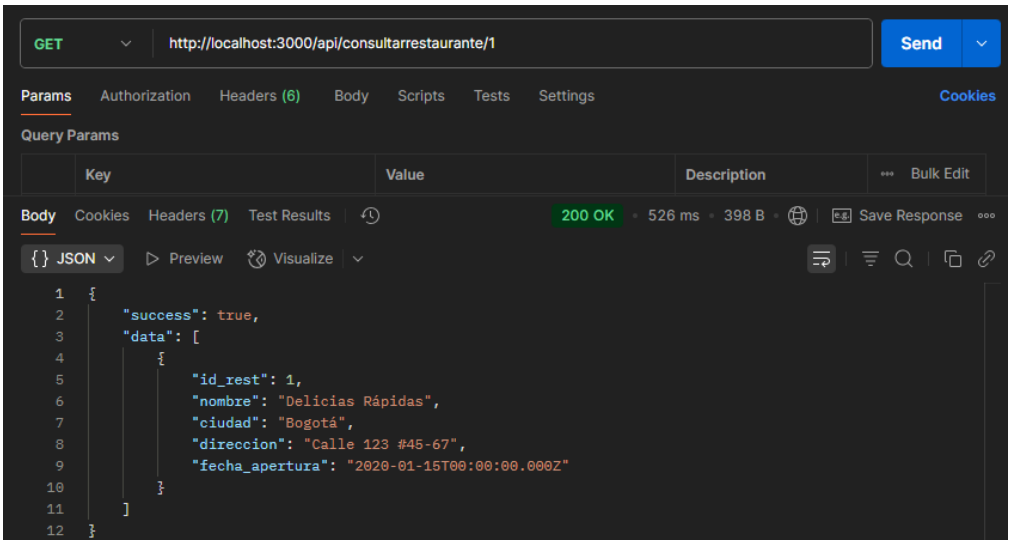
Y la probamos en postman



Leemos por id de restaurante:

```
//AQUI SE CONSULTA UN RESTURANTE POR ID
app.get('/api/consultarrestaurante/:id', async (req, res) => {
    try {
      const result = await sql`SELECT * FROM Restaurante WHERE id_rest = ${req.params.id};`;
      res.status(200).json({ success: true, data: result }); //Devuelve el resultado
    } catch (error) {
      res.status(500).json({ success: false, message: 'Error consultando restaurante', details: error.message });
    }
  });       You, 2 days ago • Proyecto con postman  y scrip db
```

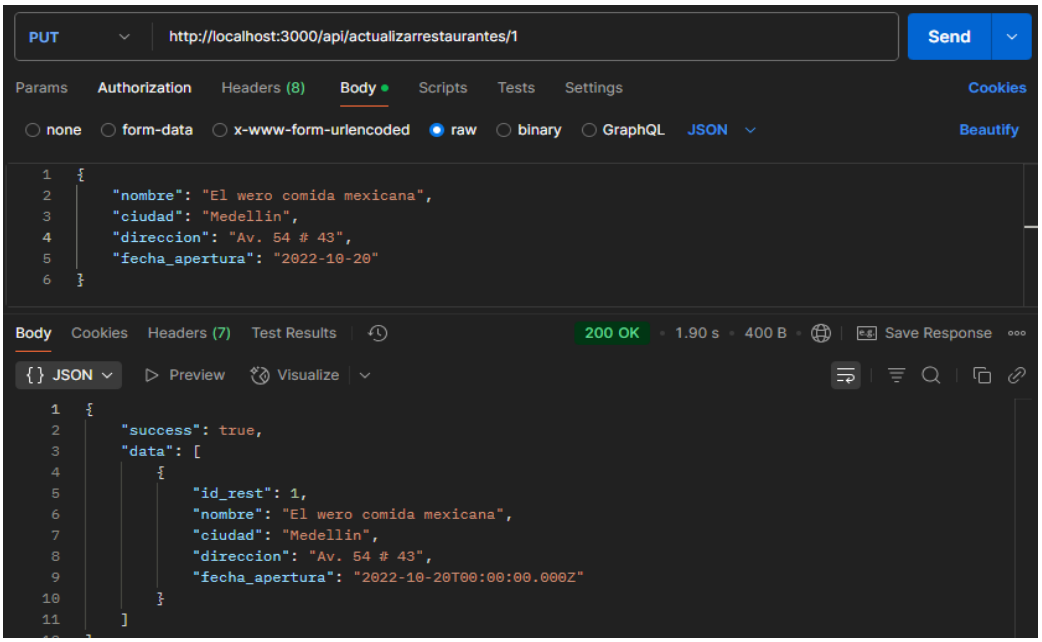
Probamos en postman



Actualizar registros

```
//Actualizar restaurante
app.put('/api/actualizarrestaurantes/:id', async (req, res) => {
  try {
    const { nombre, ciudad, direccion, fecha_apertura } = req.body;
    const result = await sql`
      UPDATE Restaurante
      SET nombre = ${nombre}, ciudad = ${ciudad}, direccion = ${direccion}, fecha_apertura = ${fecha_apertura}
      WHERE id_rest = ${req.params.id}
      RETURNING *;
    `;
    res.status(200).json({ success: true, data: result });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error actualizando restaurante', details: error.message });
  }
});
```

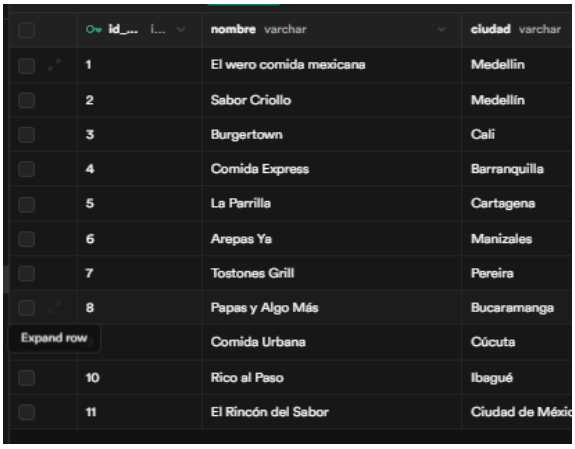
Probamos con postman



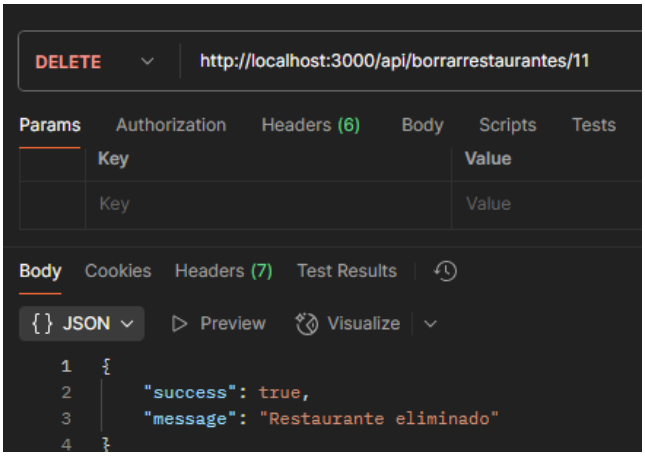
Borrar registros restaurante

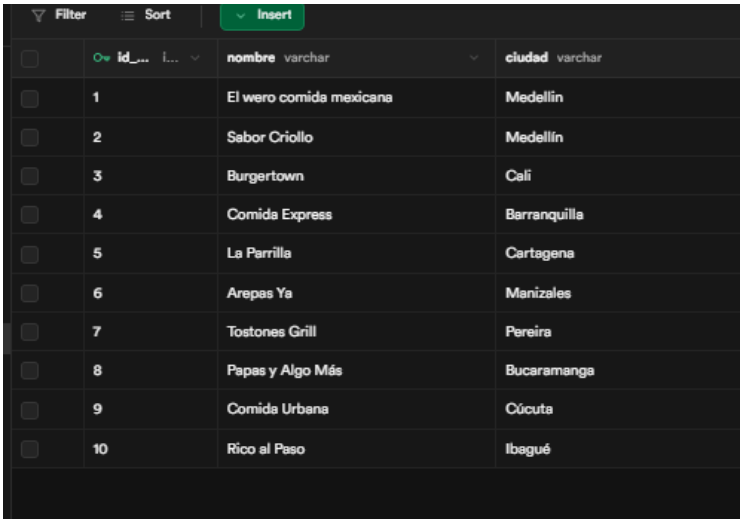
```
//borrar restaurante
app.delete('/api/borrarrestaurantes/:id', async (req, res) => {
  try {
    await sql`DELETE FROM Restaurante WHERE id_rest = ${req.params.id};`;
    res.status(200).json({ success: true, message: 'Restaurante eliminado' });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error eliminando restaurante', details: error.message });
  }
});
```

Probamos con psotman



id_... integer	nombre varchar	ciudad varchar
1	El wero comida mexicana	Medellín
2	Sabor Criollo	Medellín
3	Burgertown	Cali
4	Comida Express	Barranquilla
5	La Parrilla	Cartagena
6	Arepas Ya	Manizales
7	Tostones Grill	Pereira
8	Papas y Algo Más	Bucaramanga
Expand row	Comida Urbana	Cúcuta
10	Rico al Paso	Ibagué
11	El Rincón del Sabor	Ciudad de Méxic





Filter Sort Insert

id_... integer	nombre varchar	ciudad varchar
1	El wero comida mexicana	Medellín
2	Sabor Criollo	Medellín
3	Burgertown	Cali
4	Comida Express	Barranquilla
5	La Parrilla	Cartagena
6	Arepas Ya	Manizales
7	Tostones Grill	Pereira
8	Papas y Algo Más	Bucaramanga
9	Comida Urbana	Cúcuta
10	Rico al Paso	Ibagué

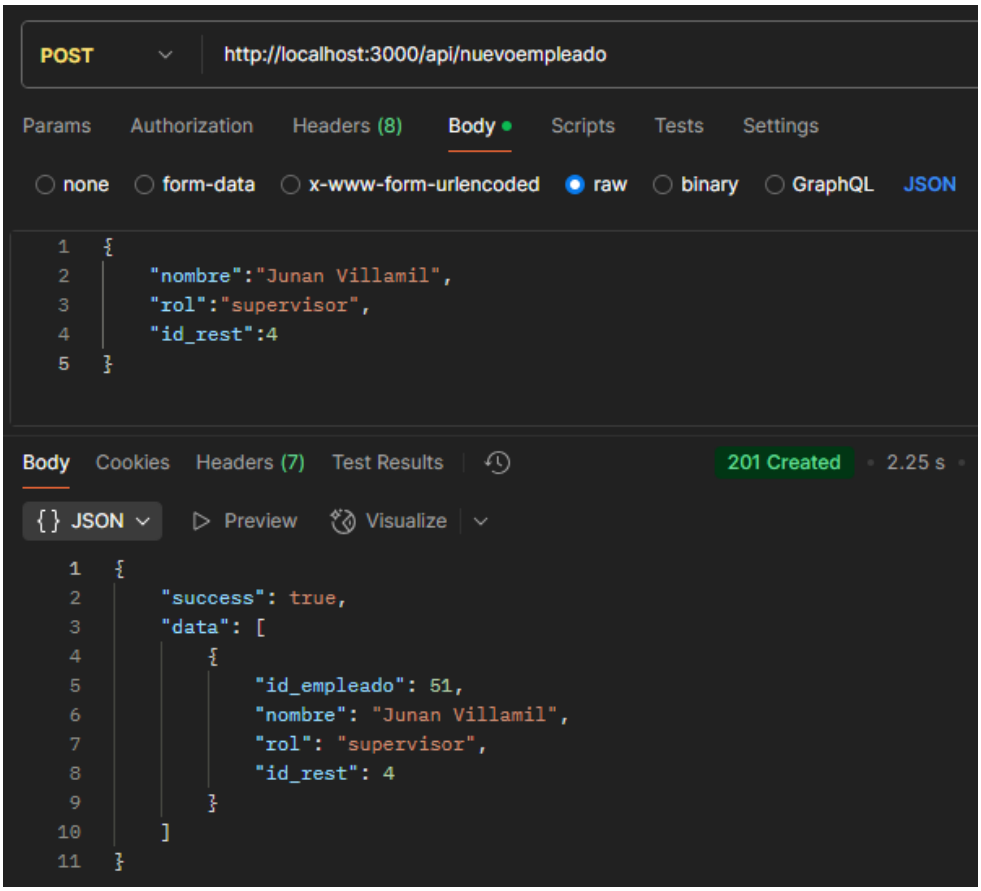
9. Ahora vamos a hacer lo mismo con cada tabla

Empleado

Insertar empleado

```
app.post('/api/nuevoempleado', async (req, res) => {      You, 1 second ago • Uncommitted changes
  try {
    const { nombre, rol, id_rest } = req.body;
    const result = await sql`
      INSERT INTO Empleado (nombre, rol, id_rest)
      VALUES (${nombre}, ${rol}, ${id_rest})
      RETURNING *;
    `;
    res.status(201).json({ success: true, data: result });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error creando empleado', details: error.message });
  }
});
```

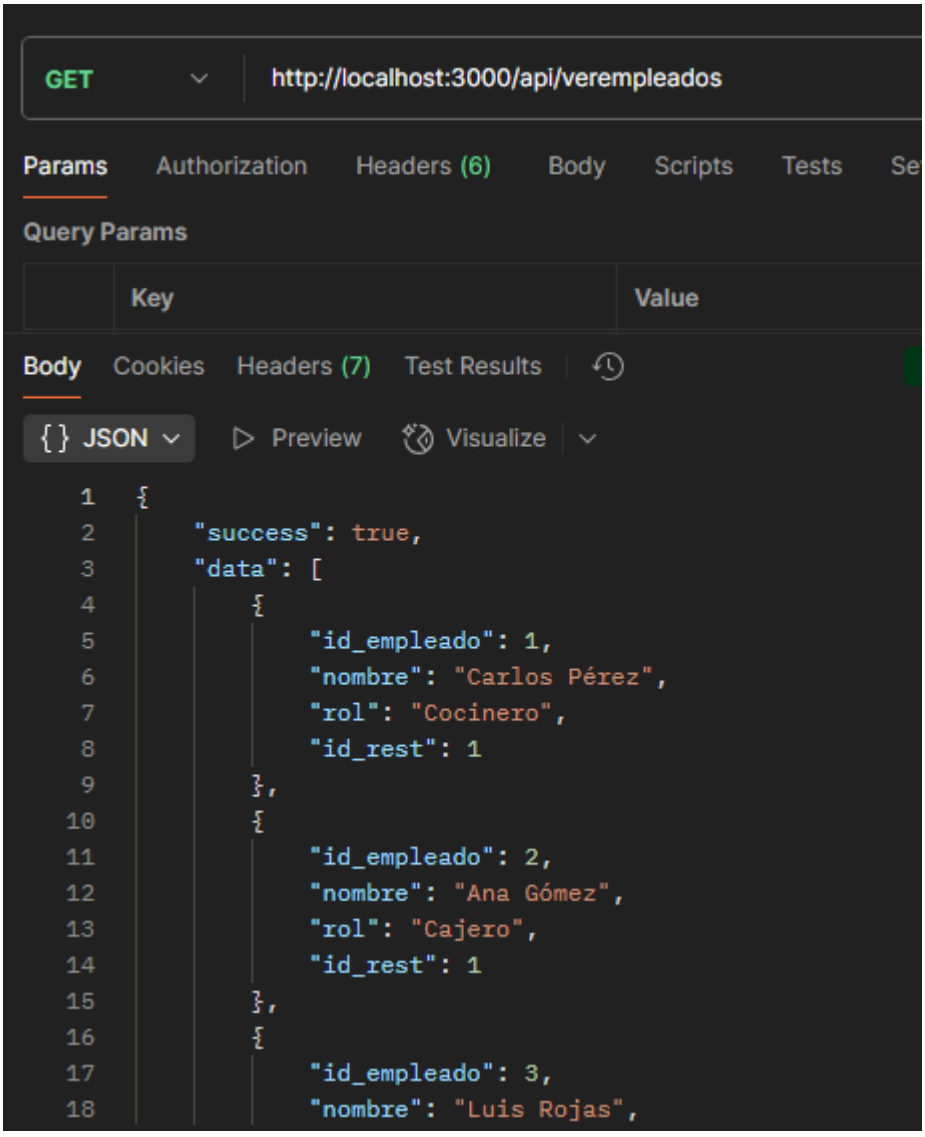
Probar postman



Ver empleados

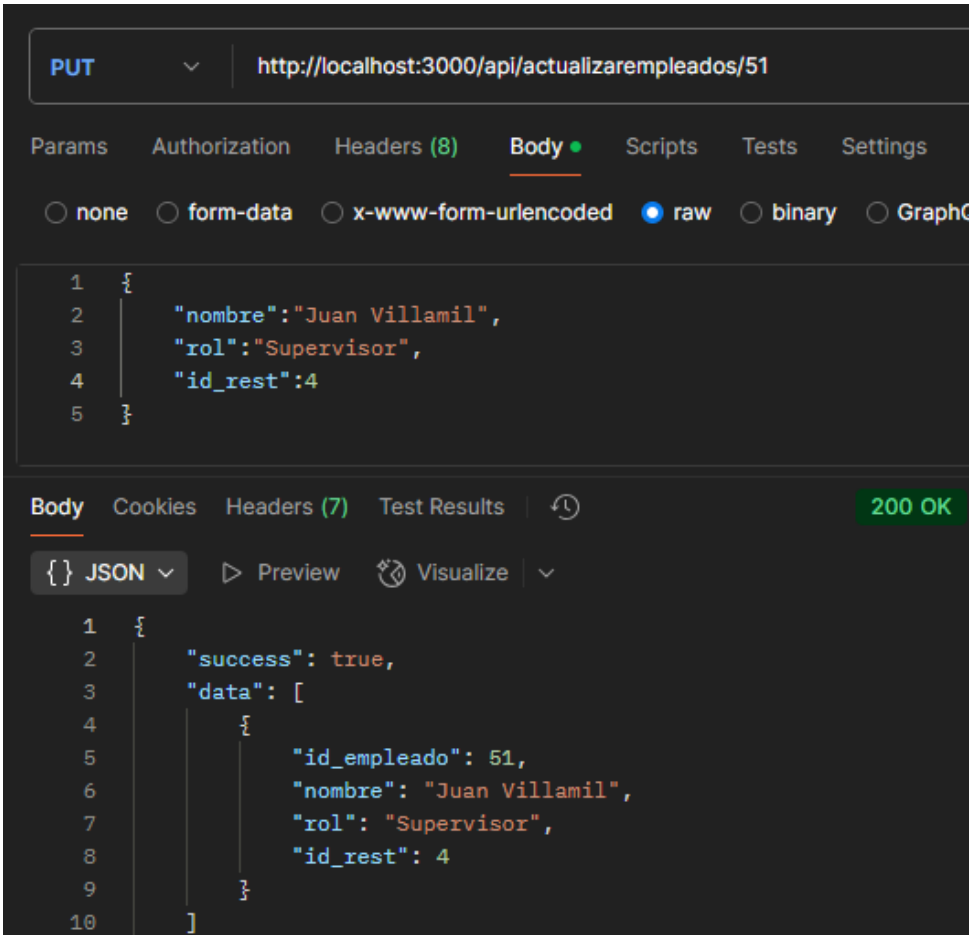
```
//Listar empleados
app.get('/api/verempleados', async (req, res) => {
  try {
    const result = await sql`SELECT * FROM Empleado;`;
    res.json({ success: true, data: result });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error obteniendo empleados', details: error.message });
  }
```

Probamos con postman



Actualizar un empleado

```
//Actualizar un empleado
app.put('/api/actualizarempleados/:id', async (req, res) => {
  try {
    const { nombre, rol, id_rest } = req.body;
    const result = await sql`
      UPDATE Empleado
      SET nombre = ${nombre}, rol = ${rol}, id_rest = ${id_rest}
      WHERE id_empleado = ${req.params.id}
      RETURNING *;
    `;
    res.json({ success: true, data: result });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error actualizando empleado', details: error.message });
  }
});
```



Eliminar empleado

```
//Eliminar empleado
app.delete('/api/borrarempleados/:id', async (req, res) => {      You, 1 second ago • Uncommitted changes
  try {
    await sql`DELETE FROM Empleado WHERE id_empleado = ${req.params.id};`;
    res.json({ success: true, message: 'Empleado eliminado' });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error eliminando empleado', details: error.message });
  }
});
```

47	Nelson Franco	Cajero	10	
48	Luisa Herrera	Mesero	10	
49	Samuel Ríos	Administrador	10	
50	Yesenia Fuentes	Domiciliario	10	
51	Juan Villamil	Supervisor	4	

Page 1 of 1 | 100 rows | 51 records

DELETE http://localhost:3000/api/borrarempleados/51

Params | Authorization | Headers (6) | Body | Scripts | Te

Query Params

Key	Value
Key	Value

Body | Cookies | Headers (7) | Test Results

{ } JSON | Preview | Visualize

```
1  {
2      "success": true,
3      "message": "Empleado eliminado"
4  }
```

48	Luisa Herrera	Mesero	
49	Samuel Ríos	Administrador	
50	Yesenia Fuentes	Domiciliario	

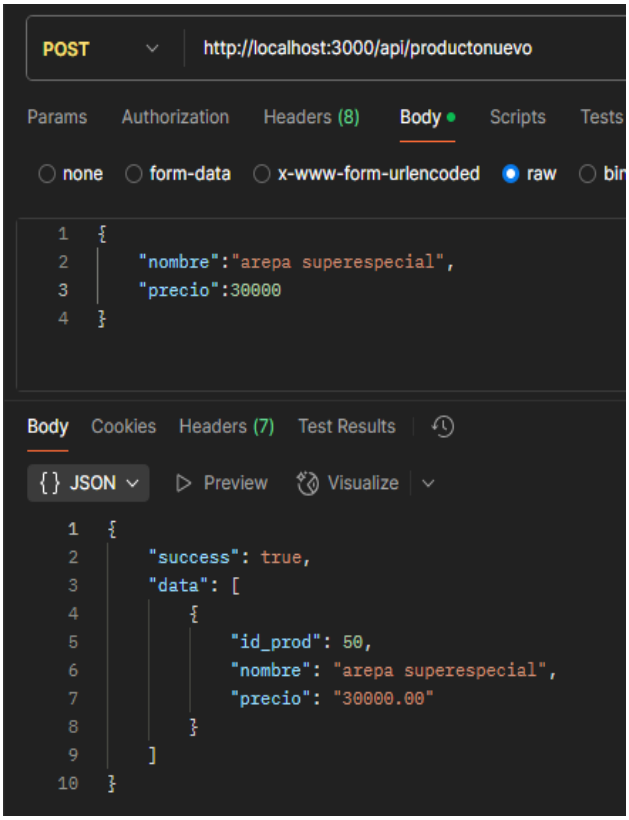
Page 1 of 1 | 100 rows | 50 records

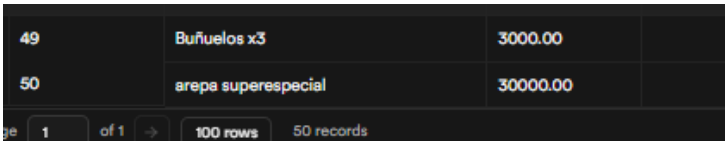
Tabla Producto

Crear producto

```
//Crear  producto nuevo
app.post('/api/productonuevo', async (req, res) => {
  try {
    const { nombre, precio } = req.body;
    const result = await sql`
      INSERT INTO Producto (nombre, precio)
      VALUES (${nombre}, ${precio})
      RETURNING *;      You, 1 second ago • Uncommitted changes
    `;
    res.status(201).json({ success: true, data: result });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error creando producto', details: error.message });
  }
});
```

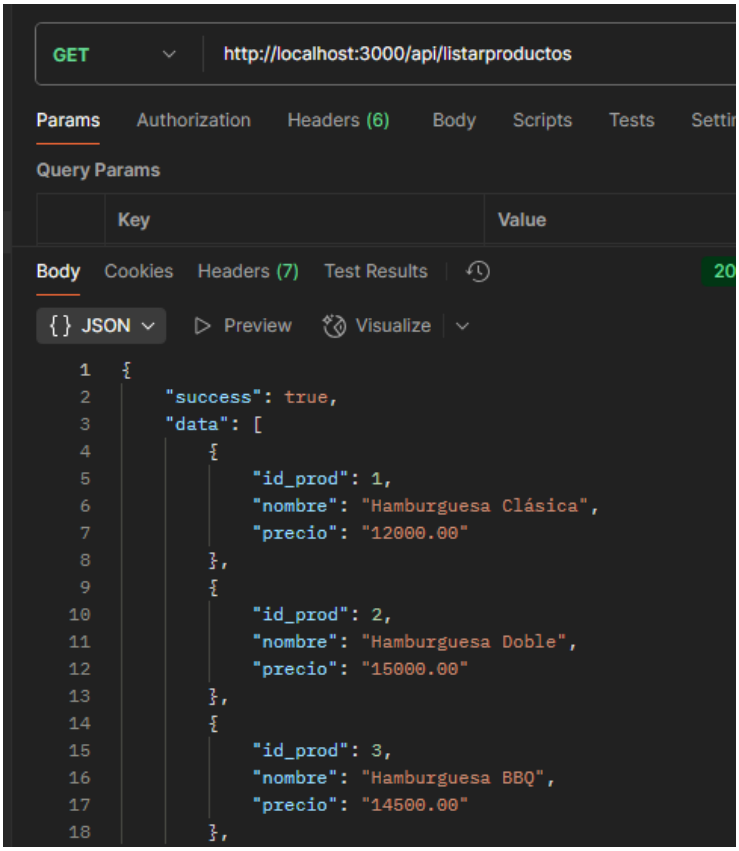
Probamos con postman





Listar productos

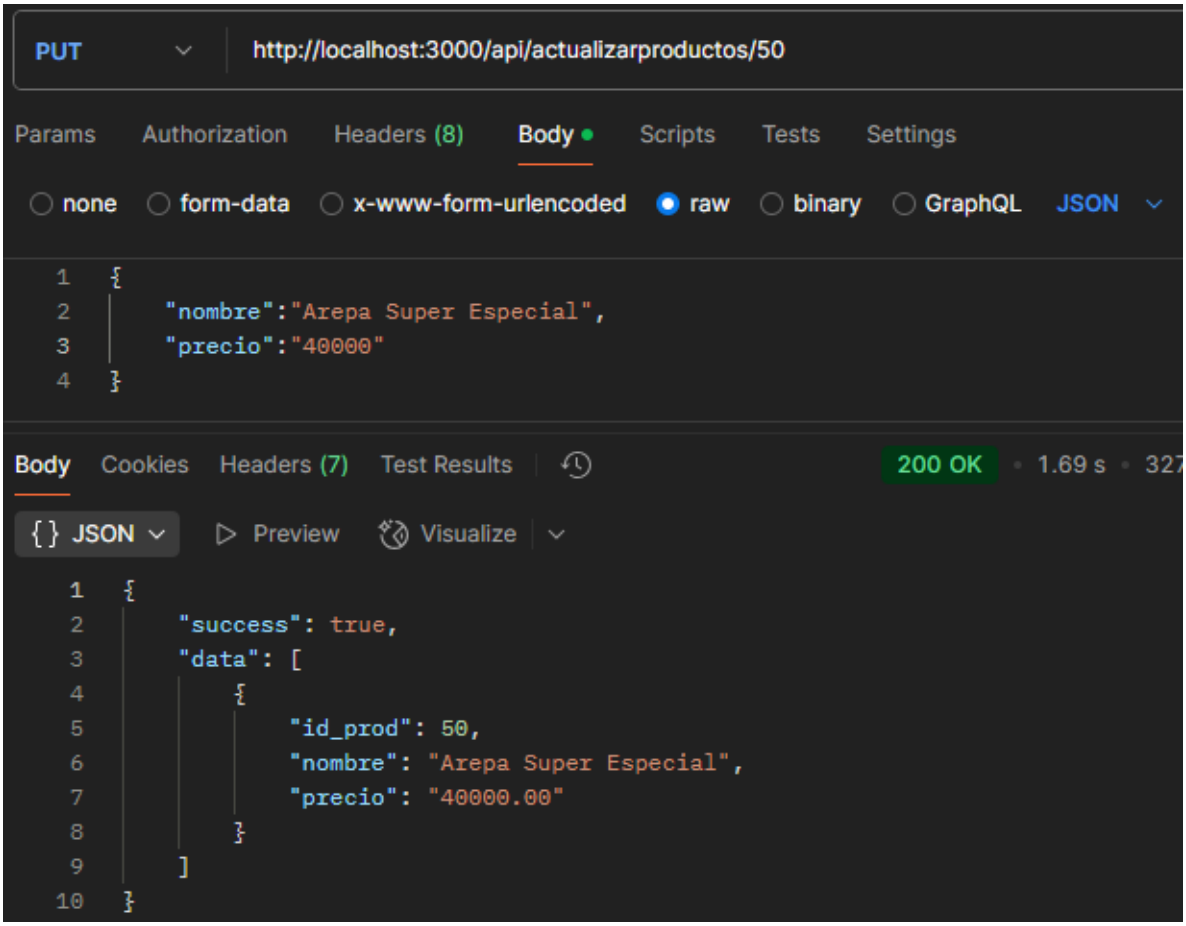
Probar con postman



Actualizar producto

```
//Actualizar producto
app.put('/api/actualizarproductos/:id', async (req, res) => {      You, 1 second ago • Uncommitted changes
try {
    const { nombre, precio } = req.body;
    const result = await sql`
      UPDATE Producto
      SET nombre = ${nombre}, precio = ${precio}
      WHERE id_prod = ${req.params.id}
      RETURNING *;
    `;
    res.json({ success: true, data: result });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error actualizando producto', details: error.message });
  }
});
```

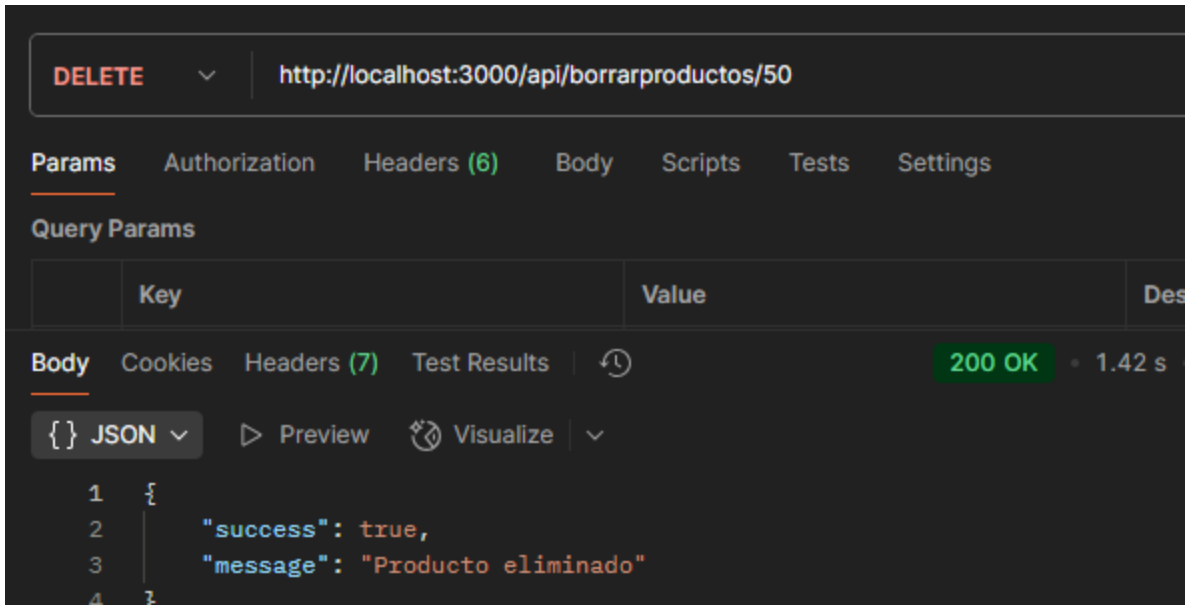
Probamos con postman



Borrar producto

```
//Borrar Producto
app.delete('/api/borrarproductos/:id', async (req, res) => {
  try {
    await sql`DELETE FROM Producto WHERE id_prod = ${req.params.id};`;
    res.json({ success: true, message: 'Producto eliminado' });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error eliminando producto', details: error.message });
  }
});
```

Probamso con postman



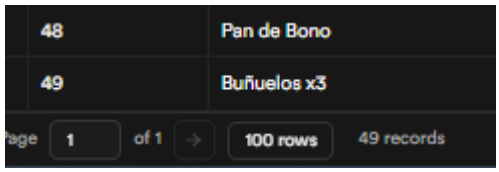
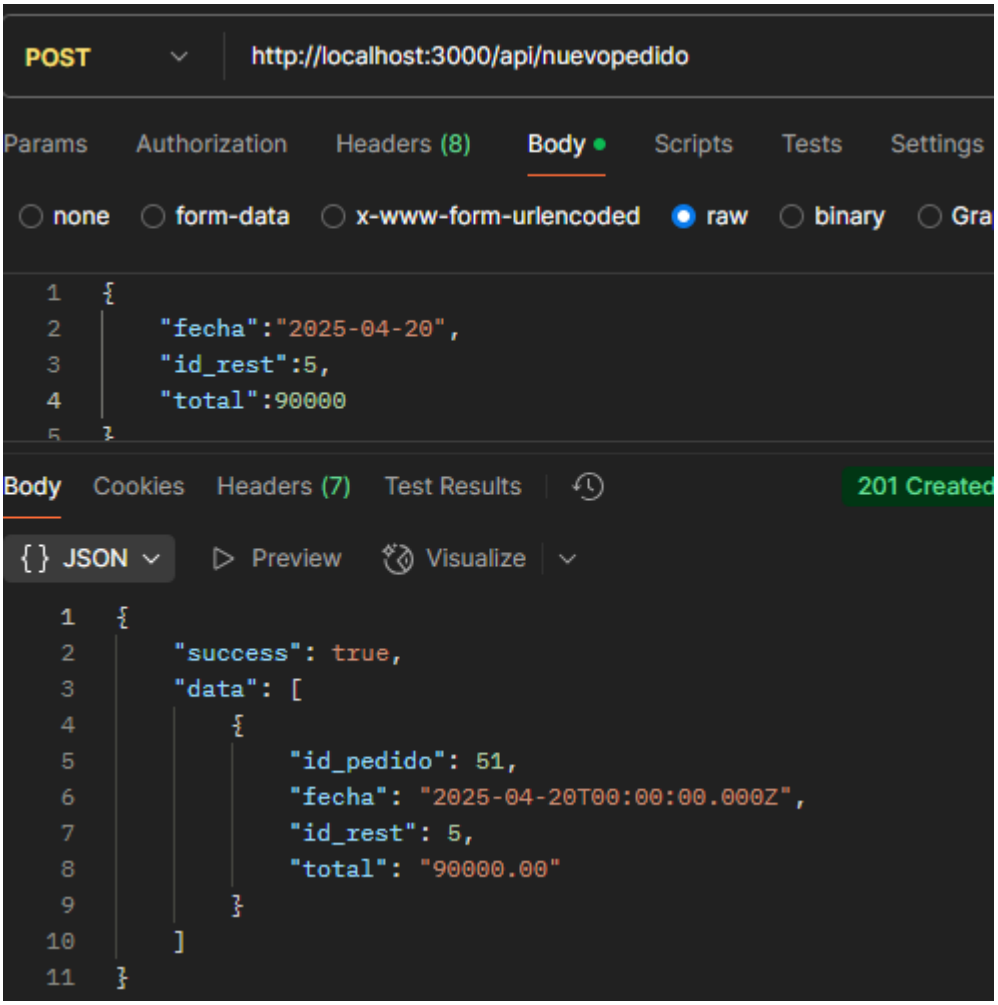


Tabla Pedido

Crear nuevo pedido

```
//Nuevo pedido
app.post('/api/nuevopedido', async (req, res) => {
  try {
    const { fecha, id_rest, total } = req.body;
    const result = await sql`
      INSERT INTO Pedido (fecha, id_rest, total)
      VALUES (${fecha}, ${id_rest}, ${total})
      RETURNING *;
    `;
    res.status(201).json({ success: true, data: result });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error creando pedido', details: error.message });
  }
});
```

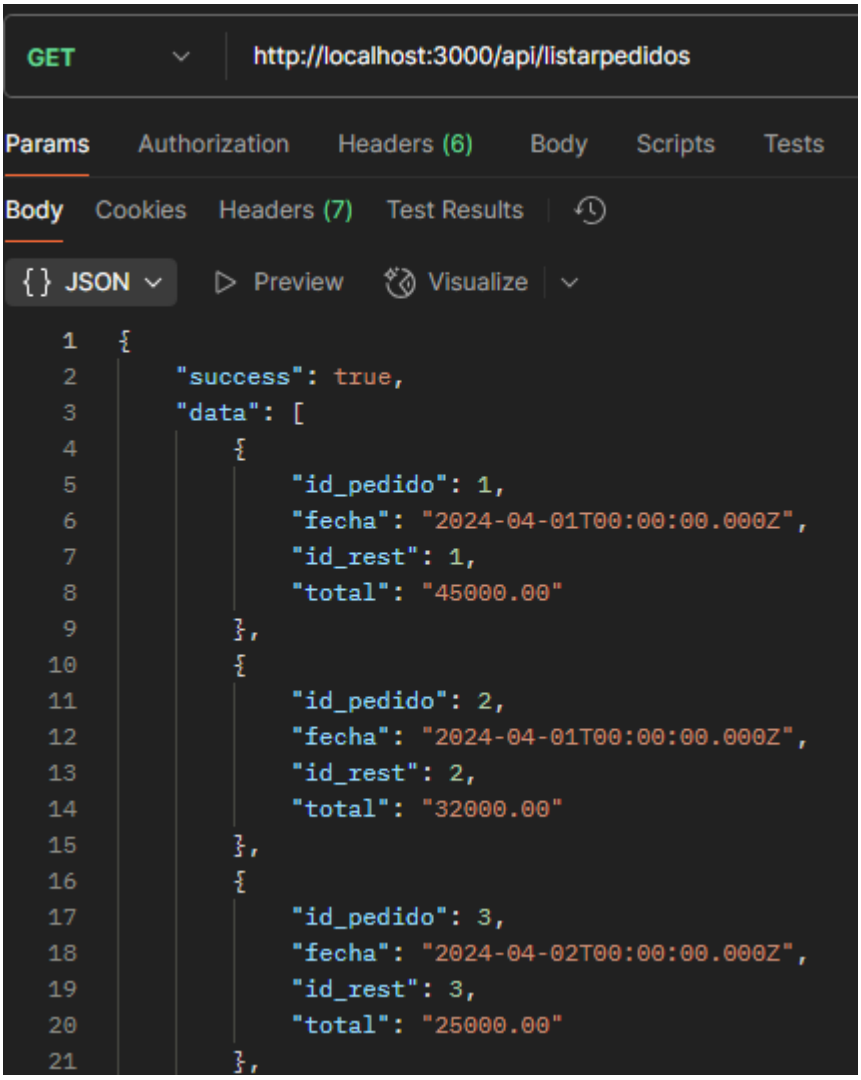
Probamos con postman



Listar los pedidos

```
//listar pedidos
app.get('/api/listarpedidos', async (req, res) => {
  try {
    const result = await sql`SELECT * FROM Pedido;`;
    res.json({ success: true, data: result });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error obteniendo pedidos', details: error.message });
  }
});
```

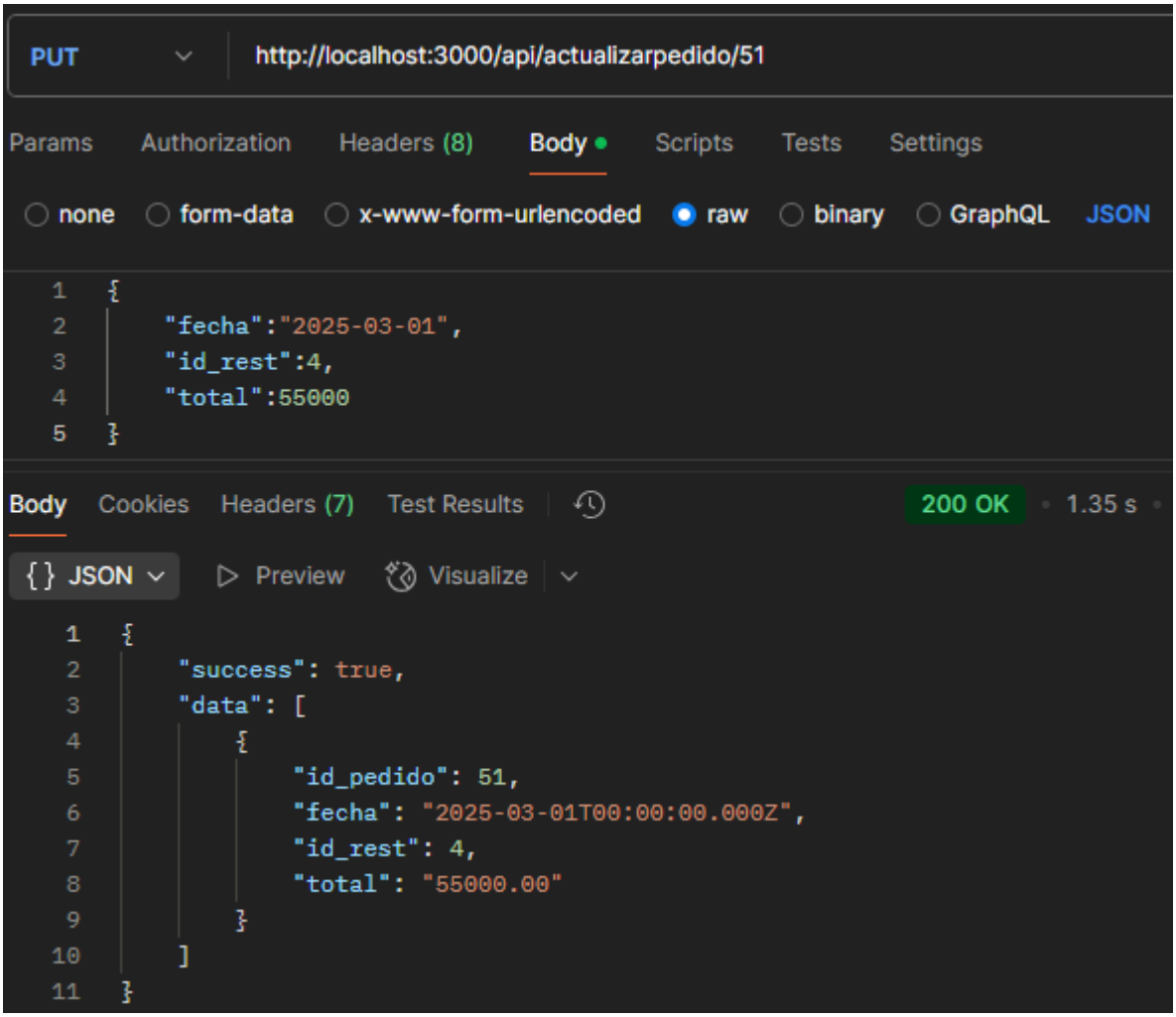
Probamos con postman



Actualizar pedido

```
app.put('/api/actualizarpedido/:id', async (req, res) => {
  try {
    const { fecha, id_rest, total } = req.body;
    const result = await sql`
      UPDATE Pedido
      SET fecha = ${fecha}, id_rest = ${id_rest}, total = ${total}
      WHERE id_pedido = ${req.params.id}
      RETURNING *;
    `;
    res.json({ success: true, data: result });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error actualizando pedido', details: error.message });
  }
});
```

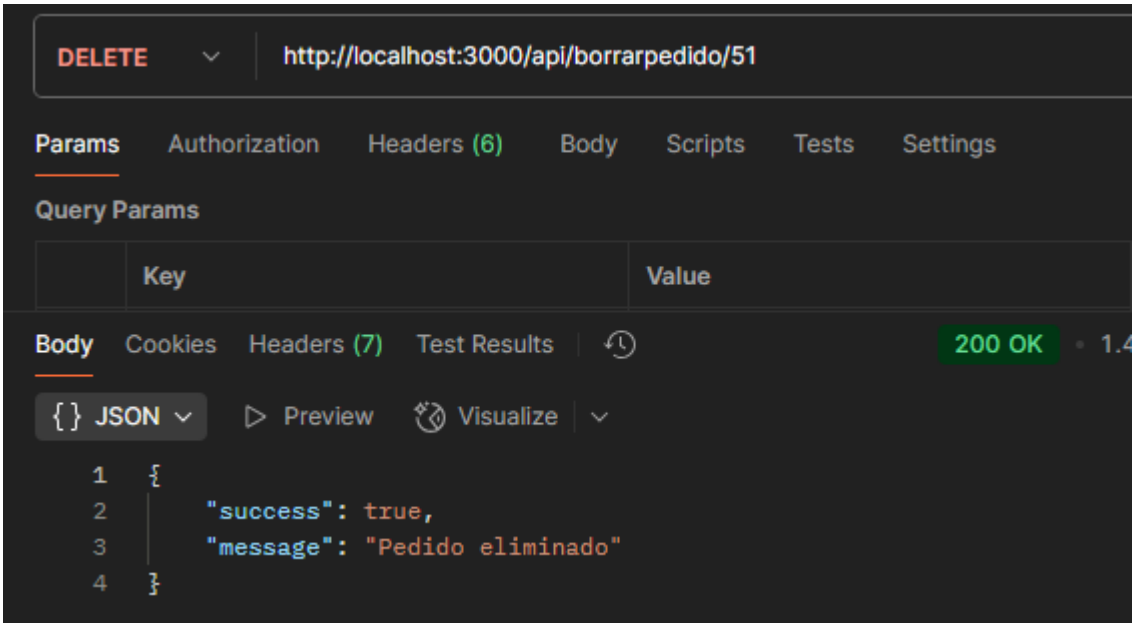
Probamos con postman



Borrar pedido

```
//Borrar pedido
app.delete('/api/borrarpedido/:id', async (req, res) => {
  try {
    await sql`DELETE FROM Pedido WHERE id_pedido = ${req.params.id};`;
    res.json({ success: true, message: 'Pedido eliminado' });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error eliminando pedido', details: error.message });
  }
});
```

Probamos con postman



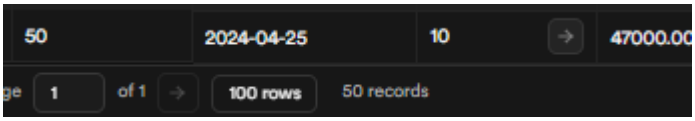
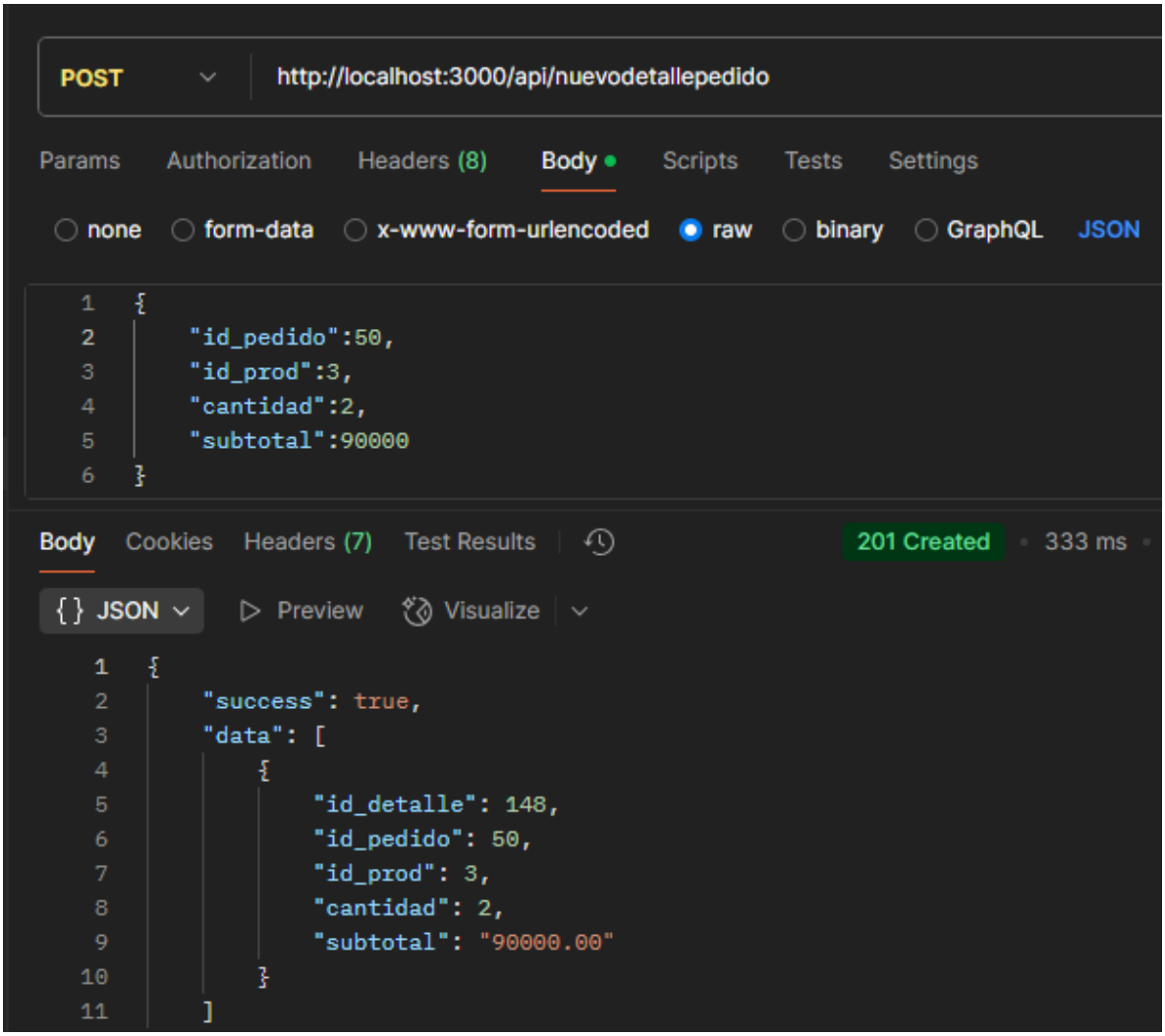


Tabla DetallePedido

Crear detalle de pedido nuevo

```
//Nuevo detallepedido
app.post('/api/nuevodetallepedido', async (req, res) => {
  try {
    const { id_pedido, id_prod, cantidad, subtotal } = req.body;
    const result = await sql`
      INSERT INTO DetallePedido (id_pedido, id_prod, cantidad, subtotal)
      VALUES (${id_pedido}, ${id_prod}, ${cantidad}, ${subtotal})
      RETURNING *;
    `;
    res.status(201).json({ success: true, data: result });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error creando detalle', details: error.message });
  }
});
```

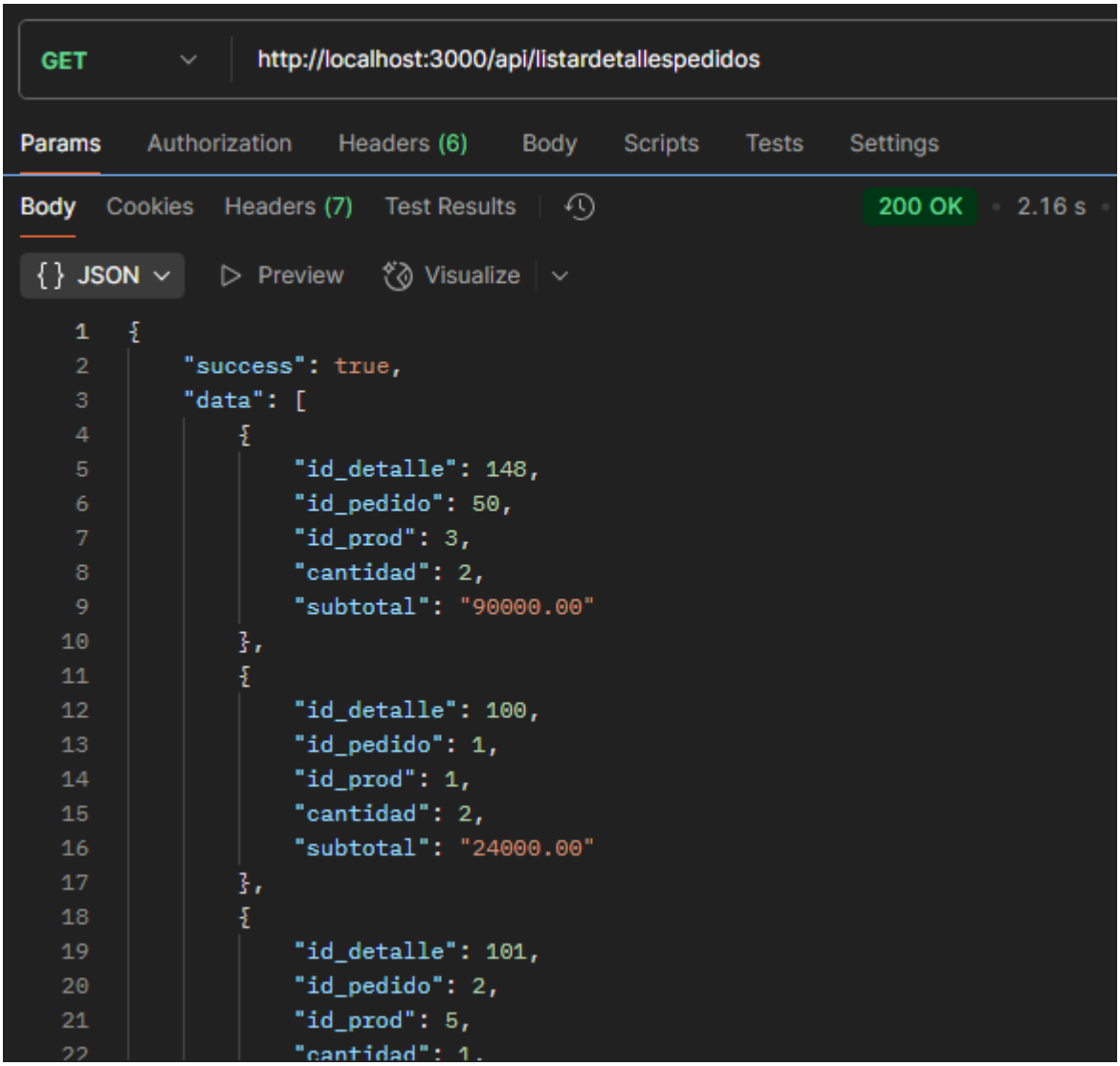
Probamos en postman



Listar los detalles de los pedidos

```
//Listar detallesPedido
app.get('/api/listardetallespedidos', async (req, res) => {
  try {
    const result = await sql`SELECT * FROM DetallePedido;`;
    res.json({ success: true, data: result });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error obteniendo detalles', details: error.message });
  }
});
```

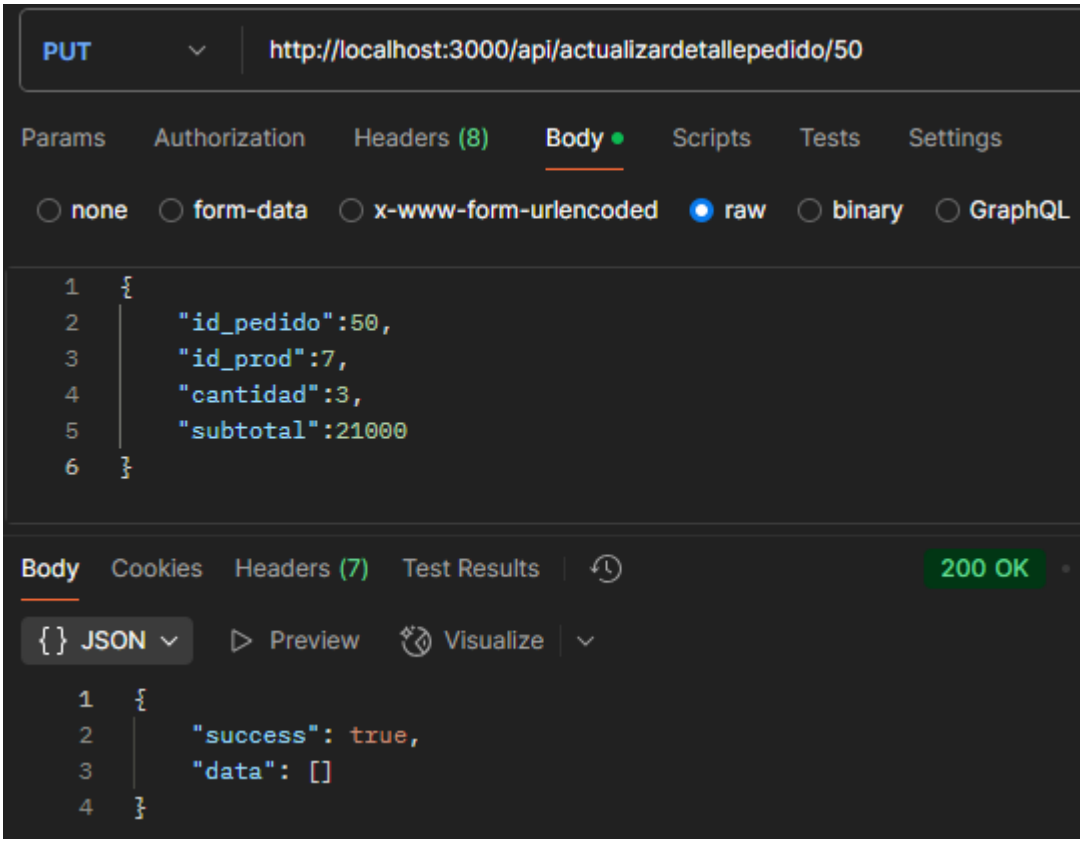
Probamos en postman



Actualizamos un detallepedido

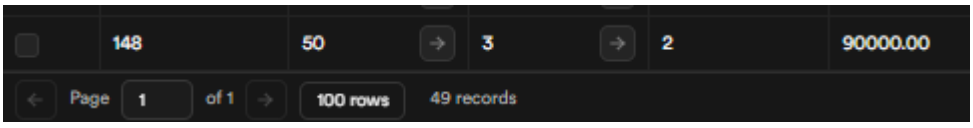
```
//Actualizar un detallepedido
app.put('/api/actualizardetallepedido/:id', async (req, res) => {
  try {
    const { id_pedido, id_prod, cantidad, subtotal } = req.body;
    const result = await sql`
      UPDATE DetallePedido
      SET id_pedido = ${id_pedido}, id_prod = ${id_prod}, cantidad = ${cantidad}, subtotal = ${subtotal}
      WHERE id_detalle = ${req.params.id}
      RETURNING *;
    `;
    res.json({ success: true, data: result });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error actualizando detalle', details: error.message });
  }
});
```

Probamos con postman

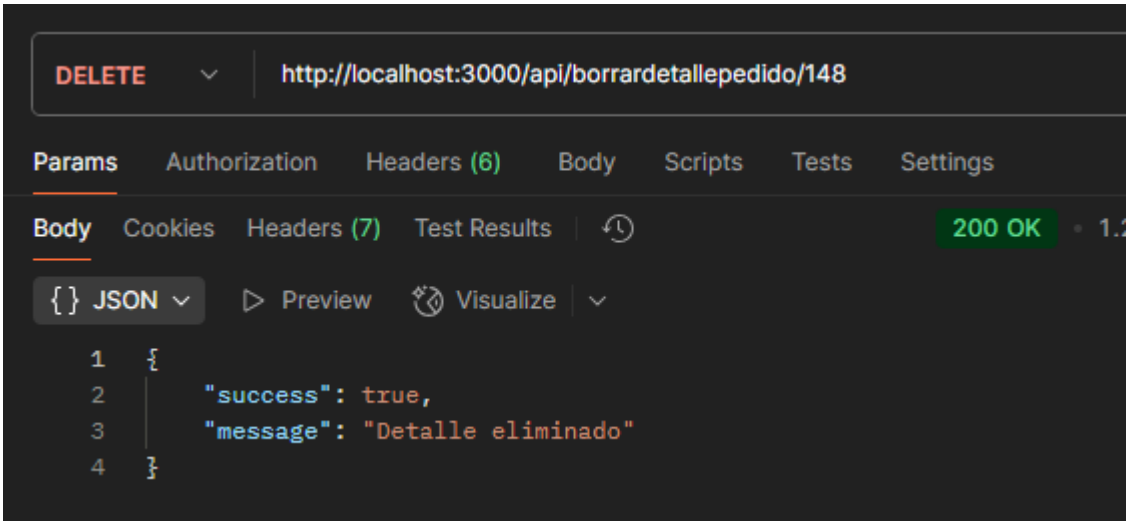


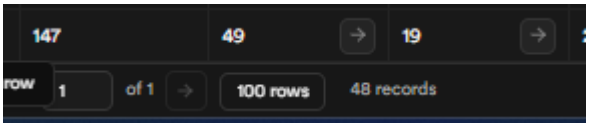
Borrar detalle pedido

```
//Borrar detallepedido
app.delete('/api/borrardetallepedido/:id', async (req, res) => {
  try {
    await sql`DELETE FROM DetallePedido WHERE id_detalle = ${req.params.id};`;
    res.json({ success: true, message: 'Detalle eliminado' });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error eliminando detalle', details: error.message });
  }
});
```



Probamos con postman

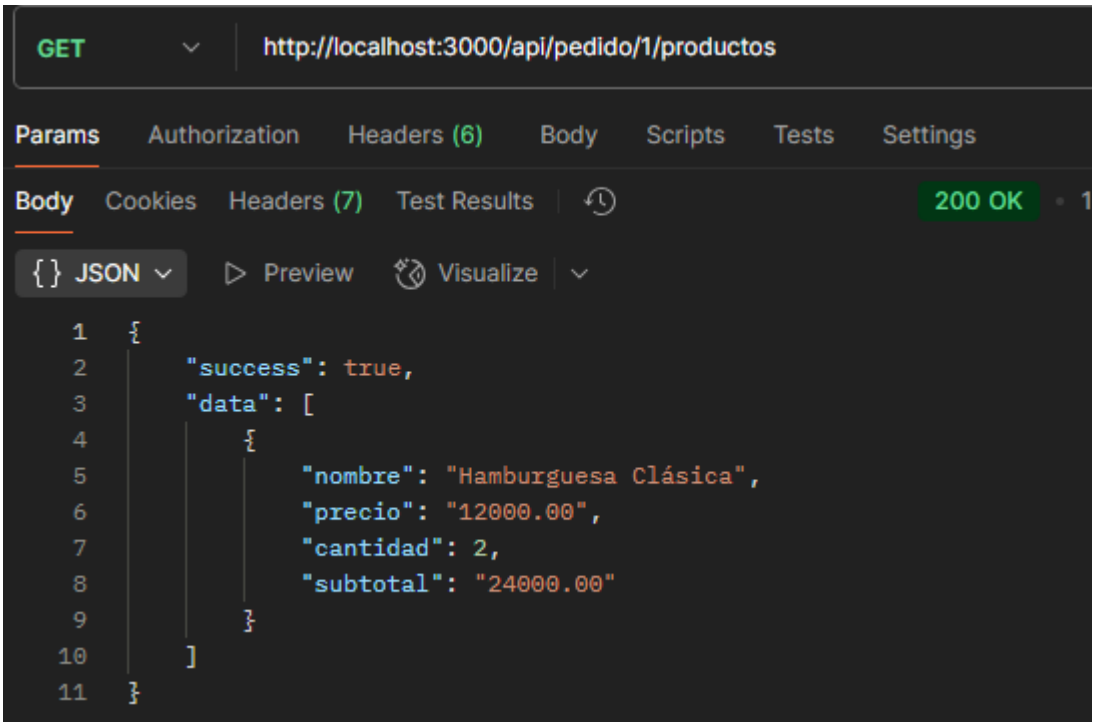




10. Ahora vamos a hacer las consultas

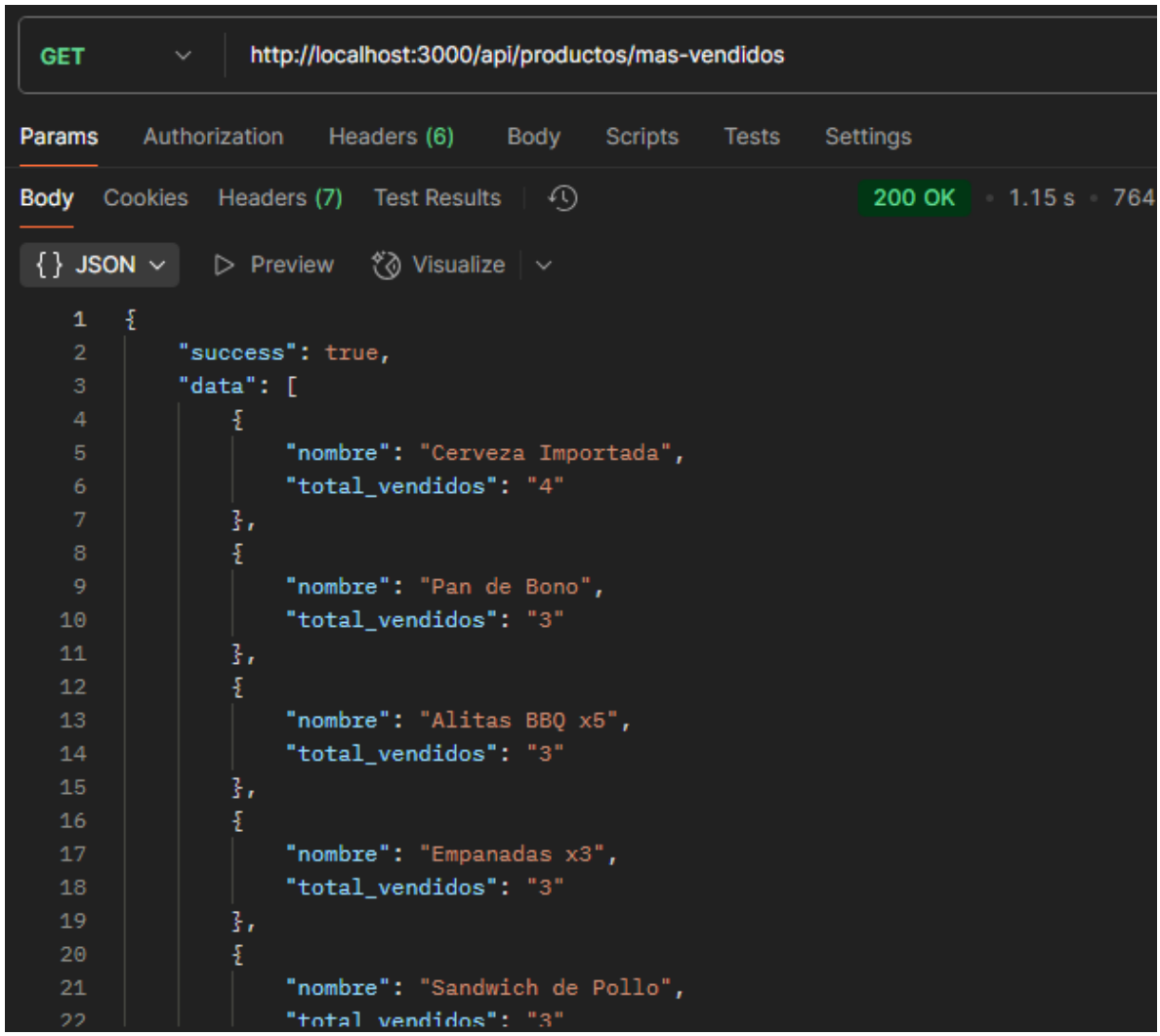
a) Obtener todos los productos de un pedido específico

```
//obtener todos los productos de un pedido especifico
app.get('/api/pedido/:id/productos', async (req, res) => {
  const { id } = req.params;
  try {
    const result = await sql`
      SELECT p.nombre, p.precio, dp.cantidad, dp.subtotal
      FROM DetallePedido dp
      JOIN Producto p ON dp.id_prod = p.id_prod
      WHERE dp.id_pedido = ${id};
    `;
    res.json({ success: true, data: result });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error obteniendo productos del pedido', details: error.message });
  }
});
```



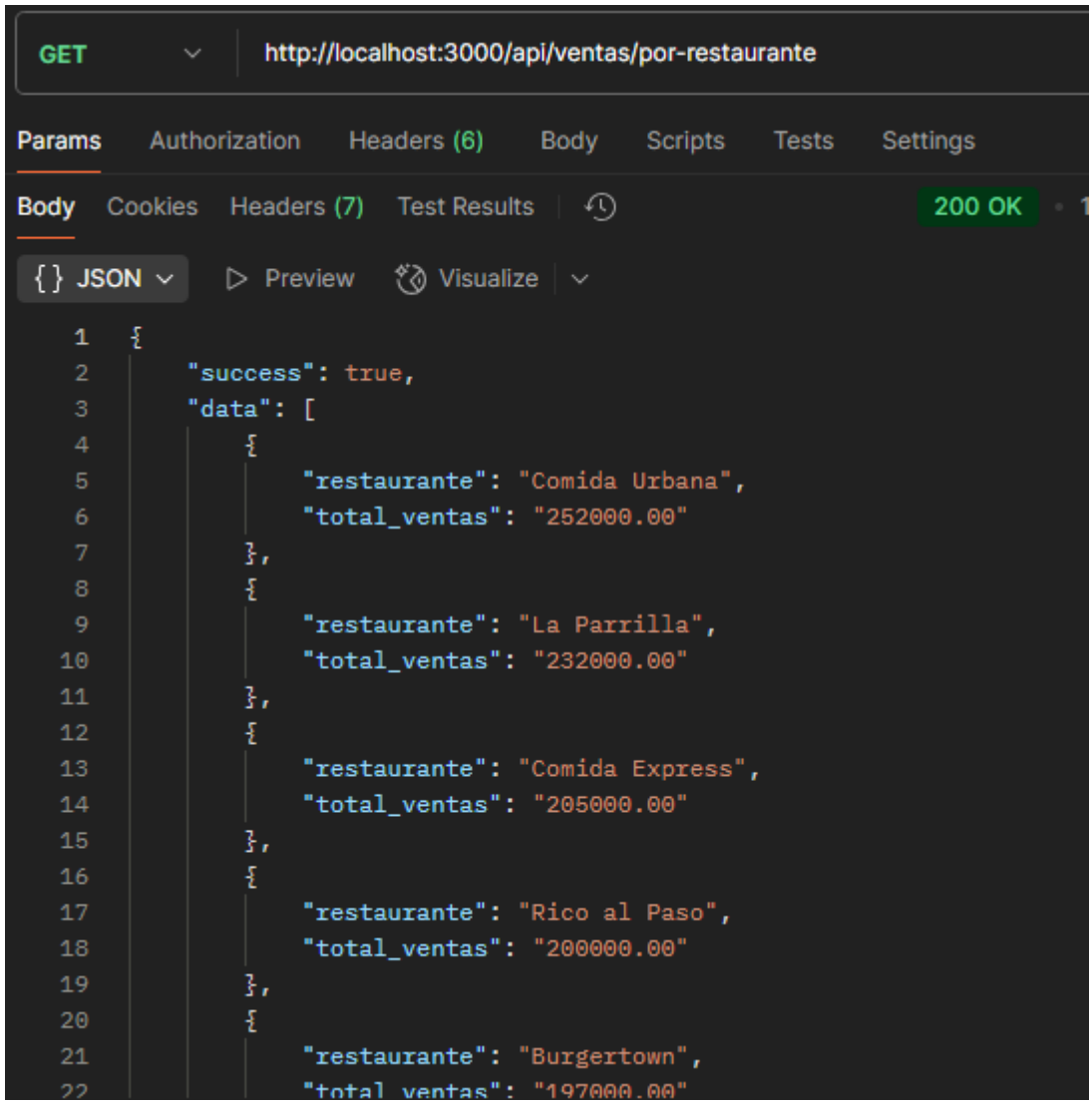
b) Obtener los productos mas vendidos

```
//obtener los productos mas vendidos
app.get('/api/productos/mas-vendidos', async (req, res) => {
  try {
    const result = await sql`
      SELECT
        p.nombre,
        SUM(dp.cantidad) AS total_vendidos
      FROM DetallePedido dp
      JOIN Producto p ON dp.id_prod = p.id_prod
      GROUP BY p.nombre
      ORDER BY total_vendidos DESC
      LIMIT 10;
    `;
    res.json({ success: true, data: result });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error obteniendo productos más vendidos', details: error.message });
  }
});
```



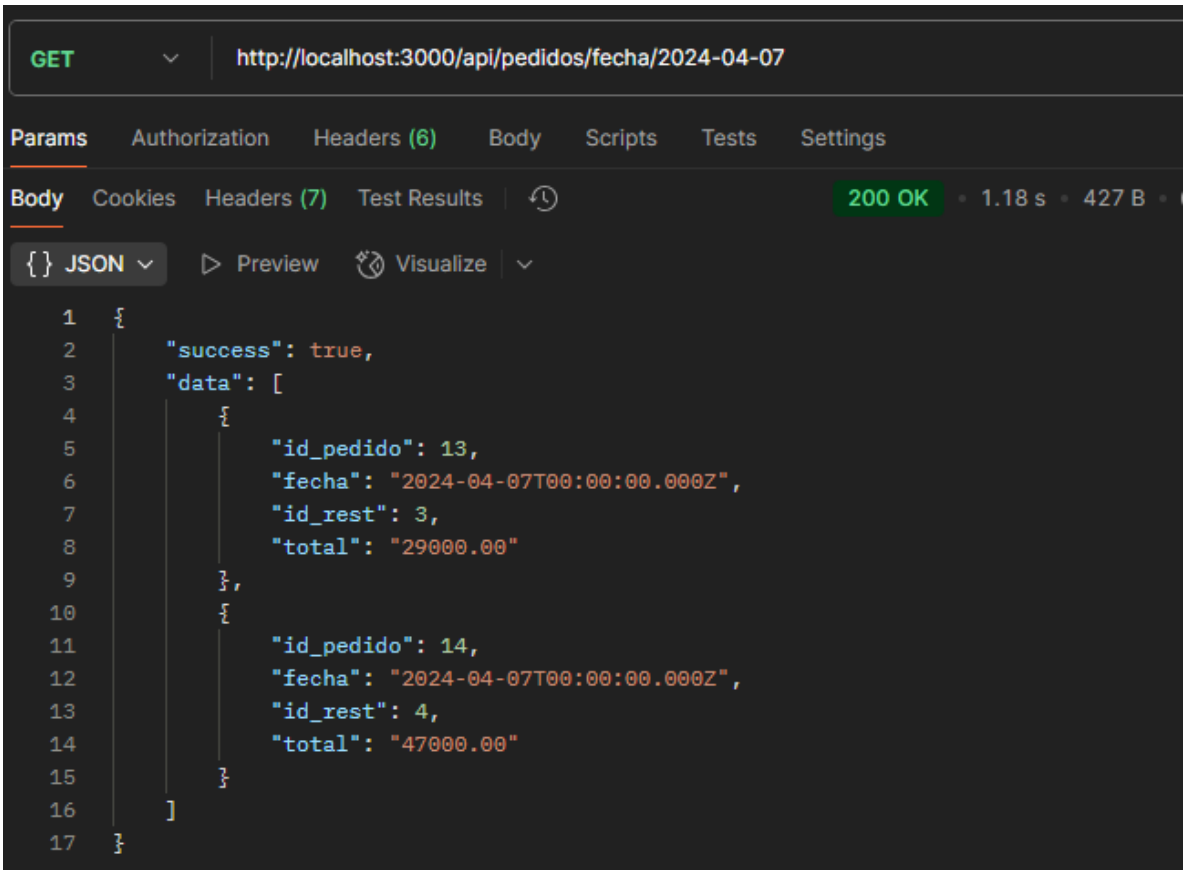
c) Obtener el total de ventas por restaurante

```
//Total de ventas por restaurante
app.get('/api/ventas/por-restaurante', async (req, res) => {
  try {
    const result = await sql`
      SELECT r.nombre AS restaurante, SUM(p.total) AS total_ventas
      FROM Pedido p
      JOIN Restaurante r ON p.id_rest = r.id_rest
      GROUP BY r.nombre
      ORDER BY total_ventas DESC;
    `;
    res.json({ success: true, data: result });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error obteniendo ventas por restaurante', details: error.message });
  }
});
```



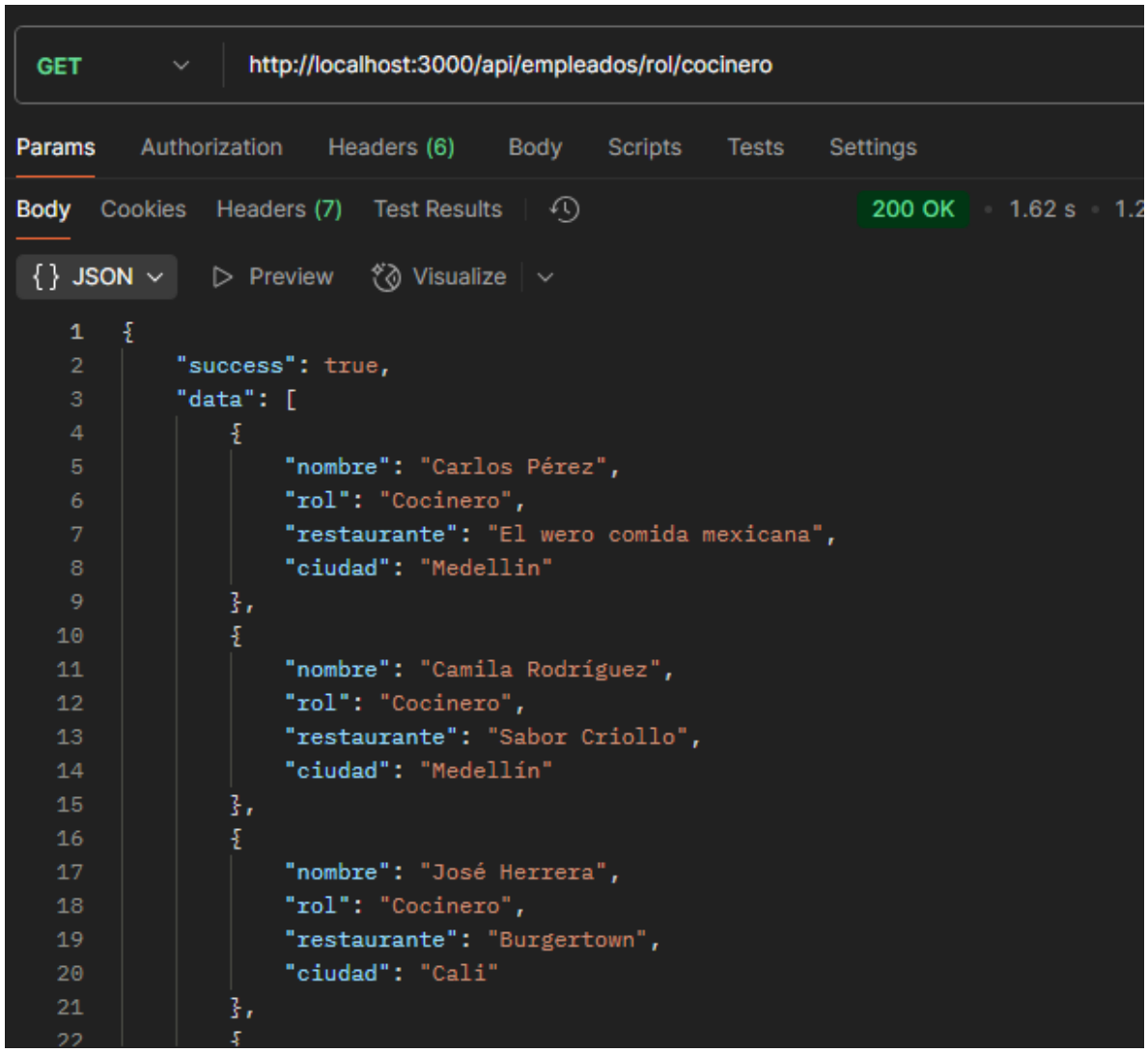
d) Obtener los pedidos realizados en una fecha específica

```
//obtener pedido por fecha

app.get('/api/pedidos/fecha/:fecha', async (req, res) => {
  const { fecha } = req.params;
  try {
    const result = await sql`
      SELECT * FROM Pedido
      WHERE fecha = ${fecha};
    `;
    res.json({ success: true, data: result });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error obteniendo pedidos por fecha', details: error.message });
  }
});
```



e) Obtener empleados por rol

```
//obtener empleados por rol
app.get('/api/empleados/rol/:rol', async (req, res) => {
  const { rol } = req.params;
  try {
    const result = await sql`
      SELECT e.nombre, e.rol, r.nombre AS restaurante, r.ciudad
      FROM Empleado e
      JOIN Restaurante r ON e.id_rest = r.id_rest
      WHERE LOWER(e.rol) = LOWER(${rol});
    `;
    res.json({ success: true, data: result });
  } catch (error) {
    res.status(500).json({ success: false, message: 'Error obteniendo empleados por rol', details: error.message });
  }
});
```


GET
http://localhost:3000/api/empleados/rol/cocinero
Params
Authorization
Headers (6)
Body
Scripts
Tests
Settings
Body
Cookies
Headers (7)
Test Results
200 OK
1.62 s
JSON
Preview
Visualize
{
"success": true,
"data": [
{
"nombre": "Carlos Pérez",
"rol": "Cocinero",
"restaurante": "El wero comida mexicana",
"ciudad": "Medellin"
},
{
"nombre": "Camila Rodríguez",
"rol": "Cocinero",
"restaurante": "Sabor Criollo",
"ciudad": "Medellín"
},
{
"nombre": "José Herrera",
"rol": "Cocinero",
"restaurante": "Burgertown",
"ciudad": "Cali"
},