# BASES DE DATOS

Manuel Santiago Malpica Daza

ID: 833776

Ingeniería de Sistemas

Semestre 7

UNIMINUTO

Corporación Universitaria Minuto de Dios

Docente:

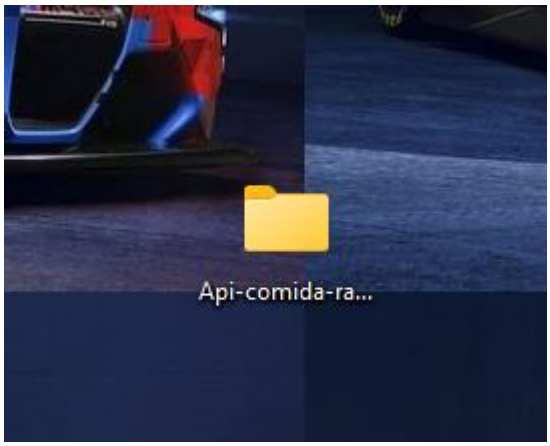
William Alexander Matallana Porras

**DOCUMENTACIÓN DEL PROYECTO API REST - CADENA DE COMIDAS RÁPIDAS**

## 1. Introducción

Este proyecto consiste en el desarrollo de una API REST utilizando Express.js y PostgreSQL (Supabase) para la gestión de la información de una cadena de comidas rápidas. La aplicación permite gestionar restaurantes, empleados, productos, pedidos y ventas.

- Creamos la carpeta en nuestro escritorio




- **Inicializar el proyecto**

Abrir la terminal en VS Code  ejecutar: bash, CopiarEditar

**npm init -y**

Esto crea el archivo package.json.

## 2. Modelo de Datos

- Instalar Dependencias

```
PS C:\Users\PERSONAL\OneDrive\Escritorio\Api-comida-rapida>
PS C:\Users\PERSONAL\OneDrive\Escritorio\Api-comida-rapida> npm install express pg cors dotenv

added 83 packages, and audited 84 packages in 13s

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

found 0 vulnerabilities
PS C:\Users\PERSONAL\OneDrive\Escritorio\Api-comida-rapida>
```

- Dentro de tu carpeta principal, crea la siguiente estructura:

**src/**

**├── controllers/**

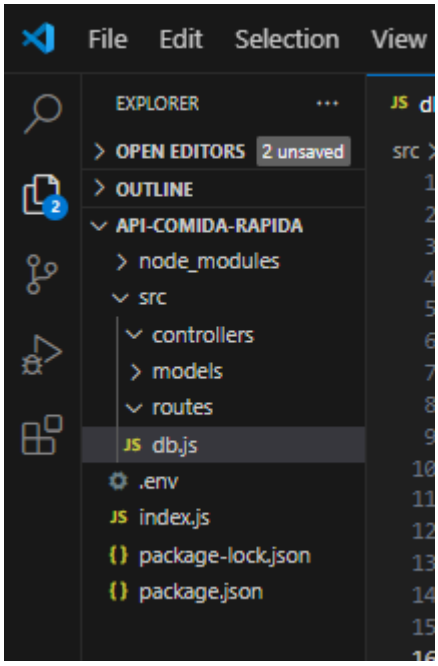
**├── routes/**

**├── models/**

**└── db.js**

**index.js**

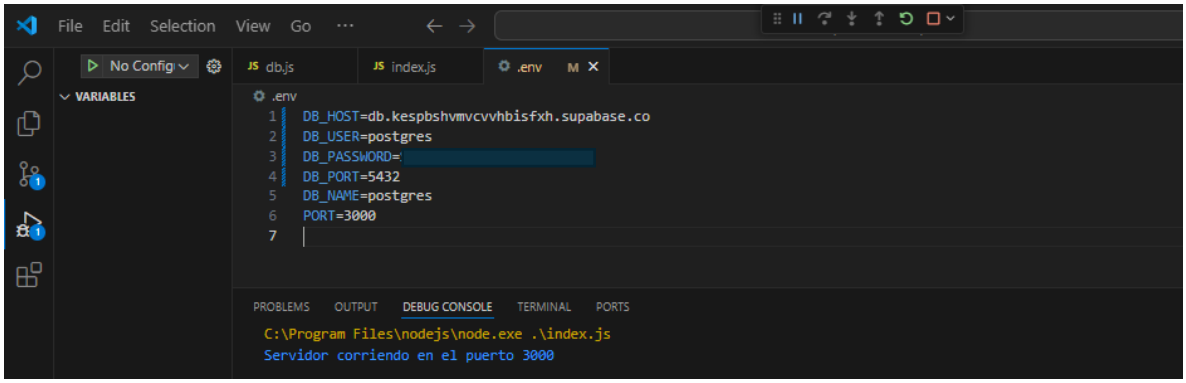
**.env**



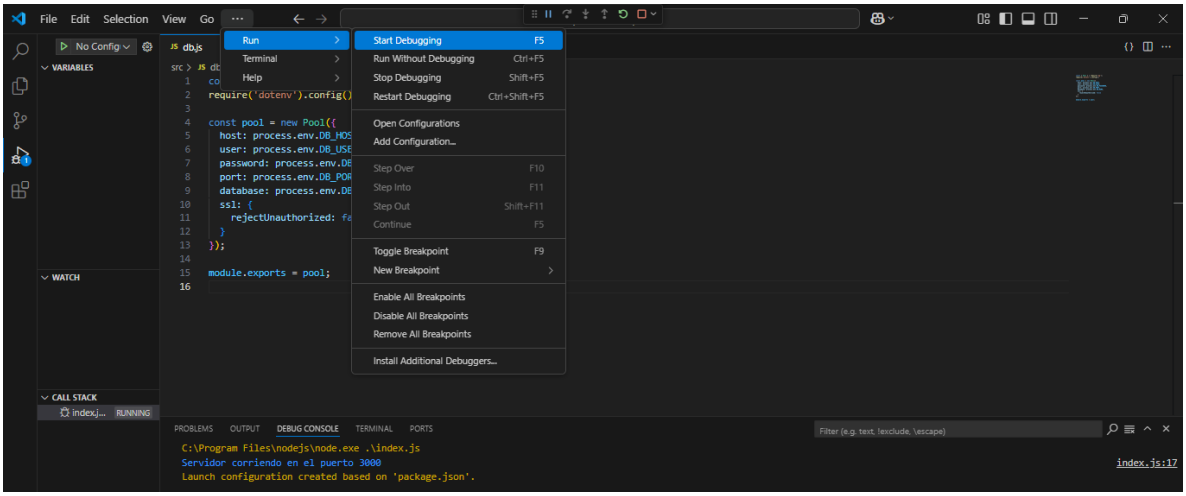
- **Confirguar el db.js, el index y el .env**

```
src > db.js > ...
1   const { Pool } = require('pg');
2   require('dotenv').config();
3
4   const pool = new Pool({
5     host: process.env.DB_HOST,
6     user: process.env.DB_USER,
7     password: process.env.DB_PASSWORD,
8     port: process.env.DB_PORT,
9     database: process.env.DB_NAME,
10    ssl: {
11      rejectUnauthorized: false
12    }
13  });
14
15  module.exports = pool;
16
```

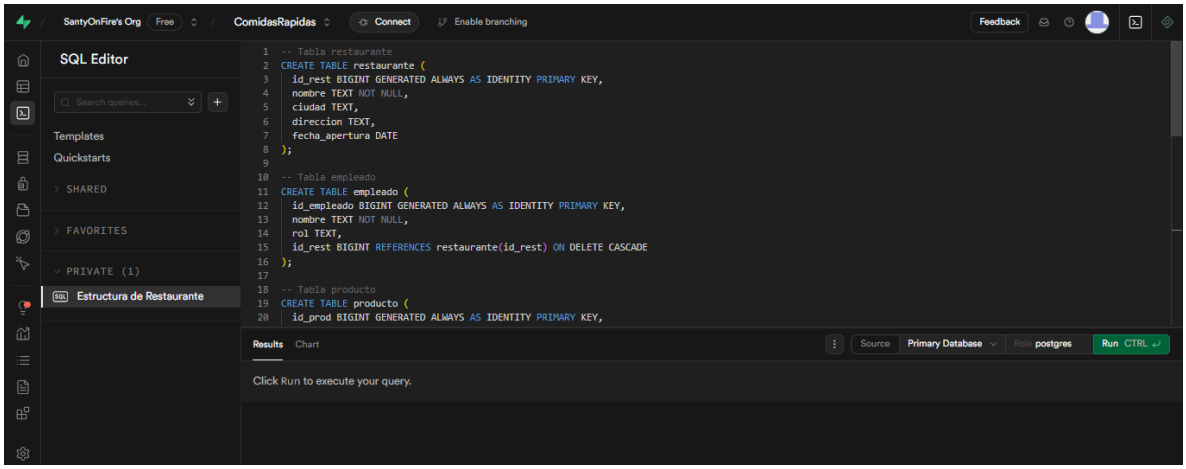
```
index.js > ...
1   const express = require('express');
2   const cors = require('cors');
3   require('dotenv').config();
4
5   const app = express();
6
7   app.use(cors());
8   app.use(express.json());
9   app.get('/', (req, res) => {
10    res.send('API de comidas rápidas funcionando 🚀');
11  });
12
13  const PORT = process.env.PORT || 3000;
14  app.listen(PORT, () => {
15    console.log(`Servidor corriendo en el puerto ${PORT}`);
16  });
```

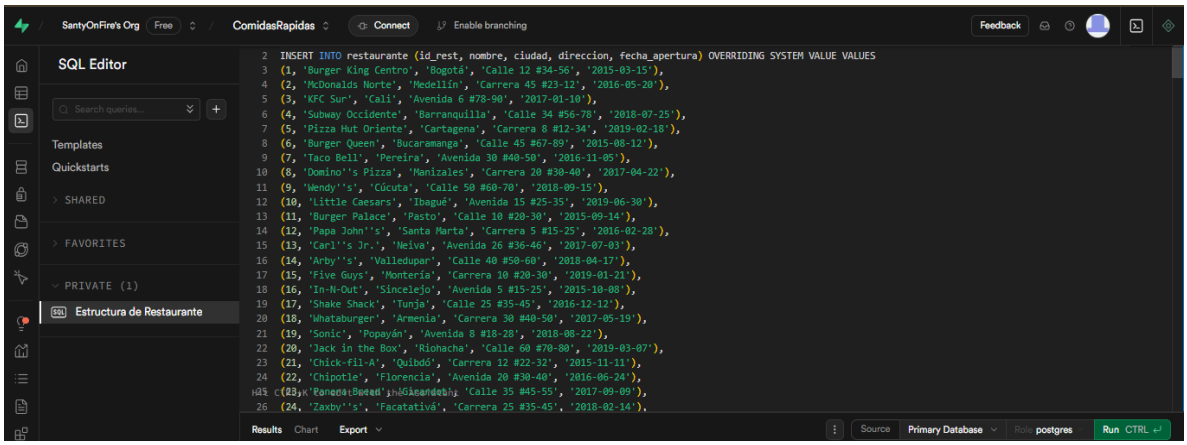


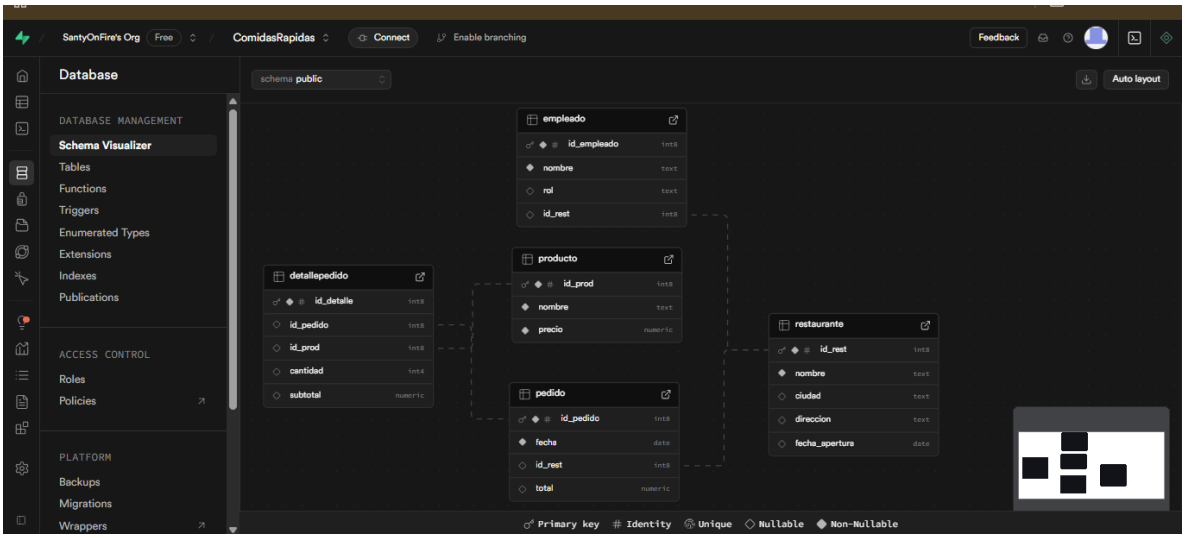
- Ahora probamos que el servidor arranque Ejecutamos en la terminal: **node index.js**




- **Creación de las tablas y inserción de registros**






**Tablas utilizadas:**

### 2.1 Restaurante

- id_rest (INT, PK)
- nombre (VARCHAR)
- ciudad (VARCHAR)
- direccion (VARCHAR)
- fecha_apertura (DATE)

### 2.2 Empleado

- id_empleado (INT, PK)
- nombre (VARCHAR)
- rol (VARCHAR)
- id_rest (INT, FK)

### 2.3 Producto

- id_prod (INT, PK)
- nombre (VARCHAR)
- precio (NUMERIC)

### 2.4 Pedido

- id_pedido (INT, PK)
- fecha (DATE)
- id_rest (INT, FK)
- total (NUMERIC)

**2.5 DetallePedido**

- id_detalle (INT, PK)
- id_pedido (INT, FK)
- id_prod (INT, FK)
- cantidad (INT)
- subtotal (NUMERIC)

**Relaciones:**

- Un restaurante tiene muchos empleados y pedidos
- Un pedido puede tener muchos productos (DetallePedido)
- Un producto puede estar en muchos pedidos

**3. Configuración del Proyecto**

- Node.js y Express.js
- PostgreSQL (Supabase)
- Librerías: express, pg, dotenv, cors

**.env**

DB_HOST=...

DB_USER=...

DB_PASSWORD=...

DB_PORT=5432

DB_NAME=...

**4. Rutas y CRUD**

**4.1 Restaurante**

- GET /restaurantes
- POST /restaurantes
- PUT /restaurantes/:id
- DELETE /restaurantes/:id

**4.2 Empleado**

- GET /empleados
- POST /empleados
- PUT /empleados/:id
- DELETE /empleados/:id

**4.3 Producto**

- GET /productos
- POST /productos
- PUT /productos/:id
- DELETE /productos/:id

**4.4 Pedido**

- GET /pedidos
- POST /pedidos
- PUT /pedidos/:id
- DELETE /pedidos/:id

**4.5 DetallePedido**

- GET /detalles
- POST /detalles
- PUT /detalles/:id
- DELETE /detalles/:id

## 5. Consultas Nativas y Rutas

**5.1 Productos de un pedido específico** GET /pedido/:id/productos

**5.2 Productos más vendidos** GET /productos/populares?min=X

**5.3 Total de ventas por restaurante** GET /ventas/restaurante

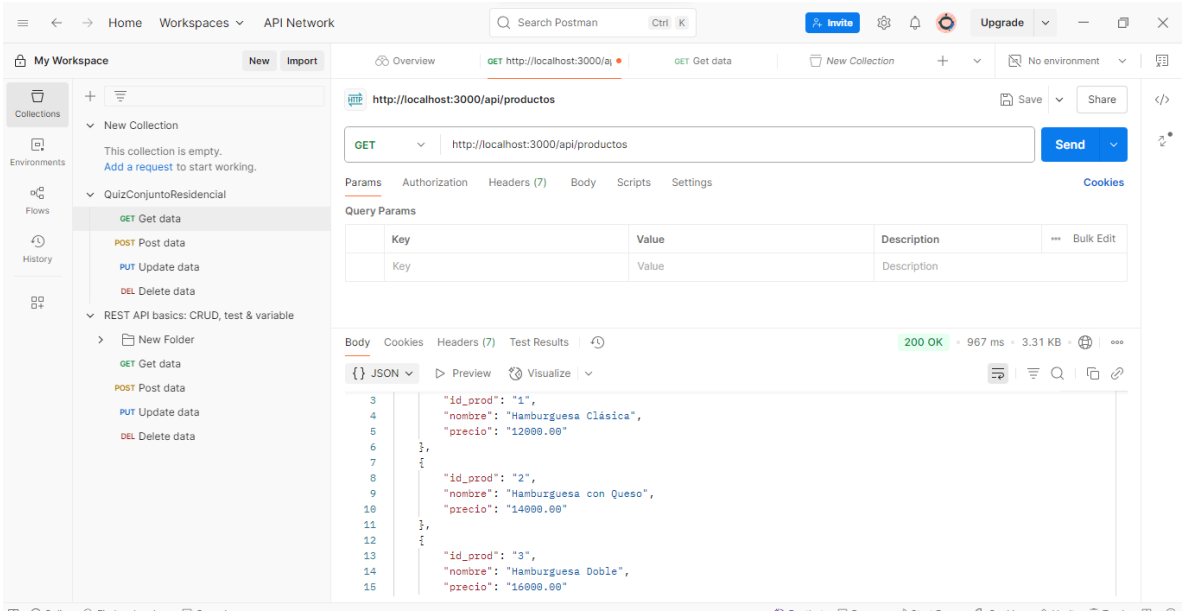
**5.4 Pedidos en una fecha específica** GET /pedidos/fecha/:fecha

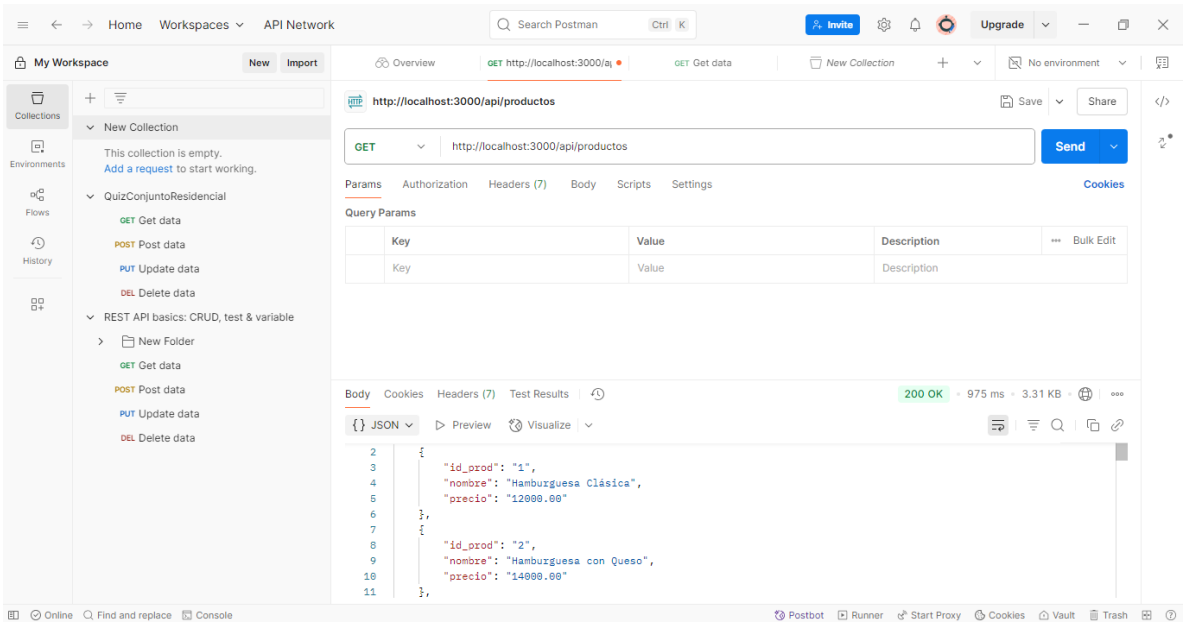
**5.5 Empleados por rol en un restaurante** GET /empleados/rol/:rol/restaurante/:id

## 6. Pruebas con Postman

Para cada ruta CRUD y consulta nativa:

- Captura del request (método, headers, body)
- Captura de la respuesta (status code, json)






- Obtener todos los productos

Método: `GET`
URL: `http://localhost:3000/api/productos`

**1. Obtener todos los productos de un pedido específico**

**Objetivo:** Ver qué productos están en un pedido determinado.

**Ruta ejemplo (GET):**

GET http://localhost:3000/api/pedidos/5/productos

**Descripción:**
Devuelve una lista de los productos incluidos en el pedido con ID 5.
Necesita hacer un JOIN entre detallePedido, producto y pedido.

**2. Obtener los productos más vendidos (más de X unidades)**

**Objetivo:** Saber qué productos han sido vendidos en más cantidad.

**Ruta ejemplo (GET):**

GET http://localhost:3000/api/productos/mas-vendidos?min=10

**Descripción:**
Devuelve los productos cuyo total de unidades vendidas es mayor que X (por ejemplo, 10).
Agrupa por id_producto en la tabla detallePedido y suma las cantidades.

### 3. Obtener el total de ventas por restaurante

**Ruta ejemplo (GET):**

GET http://localhost:3000/api/restaurantes/ventas-totales

**Descripción:**
Agrupa los pedidos (pedido) por id_rest y suma el campo total.
Devuelve el total de ventas por restaurante.

### 🗓️ 4. Obtener los pedidos realizados en una fecha específica

**Objetivo:** Listar todos los pedidos hechos en una fecha dada.

**Ruta ejemplo (GET):**

GET http://localhost:3000/api/pedidos/fecha/2025-04-25

## 7. Conclusiones

Este proyecto permite comprender la integración entre backend y base de datos utilizando tecnologías modernas. Se desarrollaron consultas nativas, se configuró Supabase y se documentó el uso de la API mediante Postman.