



Parcial 2

Laura Jhuliana Velasco Gómez

Ingeniería de sistemas, Corporación Universitaria Minuto de Dios,
Sede Cundinamarca, C.R. Zipaquirá

NRC: 60747 Bases de Datos Masivas

Ing. William Matallana Porras

25 abril 2025

1. Inicio creando una carpeta con el nombre **Parcial_2BD**, la cual servirá como base para el desarrollo del proyecto. En ella, comenzaré la creación de la API utilizando Visual Studio Code y Supabase como plataforma de gestión de la base de datos.



2. Para iniciar la ejecución del proyecto, creo un contenedor (container) desde la línea de comandos (CMD), el cual permitirá levantar el entorno necesario para correr la API de forma local.

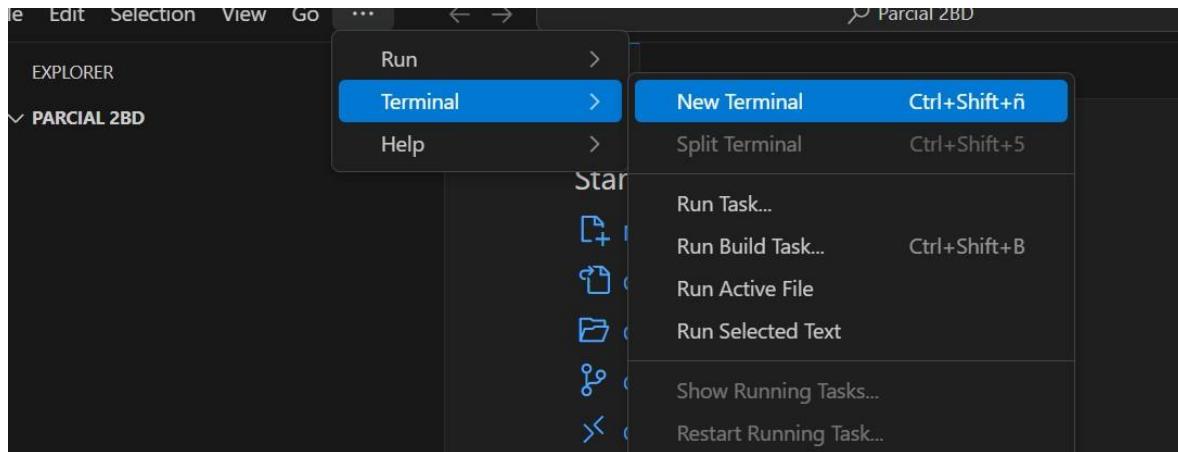
```
C:\Users\Laura Jhuliana velas>docker run -d -p 3308:3306 --name Parcial2BD -e MYSQL_ROOT_PASSWORD=1234 mysql:latest  
ed05df8ea3a4bb304c025a9da7992b1084b1ccda8eb93604178069cce8198197
```

```
C:\Users\Laura Jhuliana velas>
```

```
C:\Users\Laura Jhuliana velas>docker run -d -p 3308:3306 --name Parcial2BD -e MYSQL_ROOT_PASSWORD=1234 mysql:latest  
ed05df8ea3a4bb304c025a9da7992b1084b1ccda8eb93604178069cce8198197  
  
C:\Users\Laura Jhuliana velas>docker exec -it Parcial2BD mysql -u root -p1234  
mysql: [Warning] Using a password on the command line interface can be insecure.  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 9  
Server version: 9.2.0 MySQL Community Server - GPL  
  
Copyright (c) 2000, 2025, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql>
```

<input type="checkbox"/>	 Parcial2BD ed05df8ea3a4... mysql:latest	Running	0.66%	3308:3306	3 minutes ago		
<input type="checkbox"/>	 Tallerexpress be866c5e4c9... mysql:latest	Exited (255)	0%	3308:3306	12 days ago		

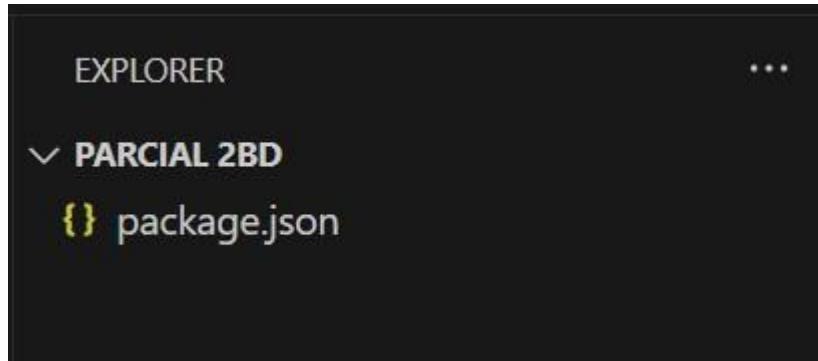
3. Para generar el archivo **package.json**, utilizo el comando **npm init -y** en la terminal, lo cual permite inicializar rápidamente un proyecto Node.js con la configuración predeterminada.



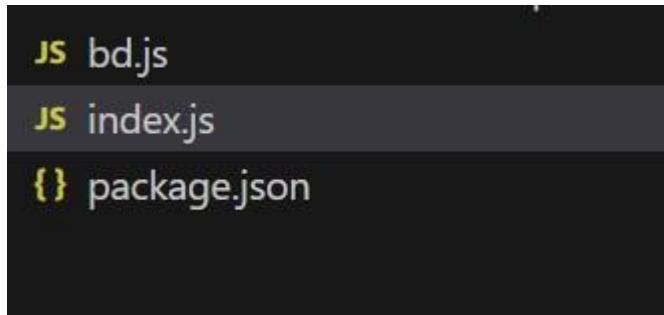
A screenshot of the VS Code terminal window. The tab bar at the top includes 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is underlined, indicating it is active), and 'PORTS'. The terminal output shows:

```
● PS C:\Users\Laura Jhuliana velas\Desktop\Parcial 2BD> npm init -y
Wrote to C:\Users\Laura Jhuliana velas\Desktop\Parcial 2BD\package.json:

{
  "name": "parcial-2bd",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \\"Error: no test specified\\" && exit 1"
  },
  "keywords": [],
  "author": "",
```



4. Además, se crean dos archivos principales: **bd.js**, donde se gestionará la conexión a la base de datos, y **index.js**, que funcionará como el archivo principal del servidor de la API.



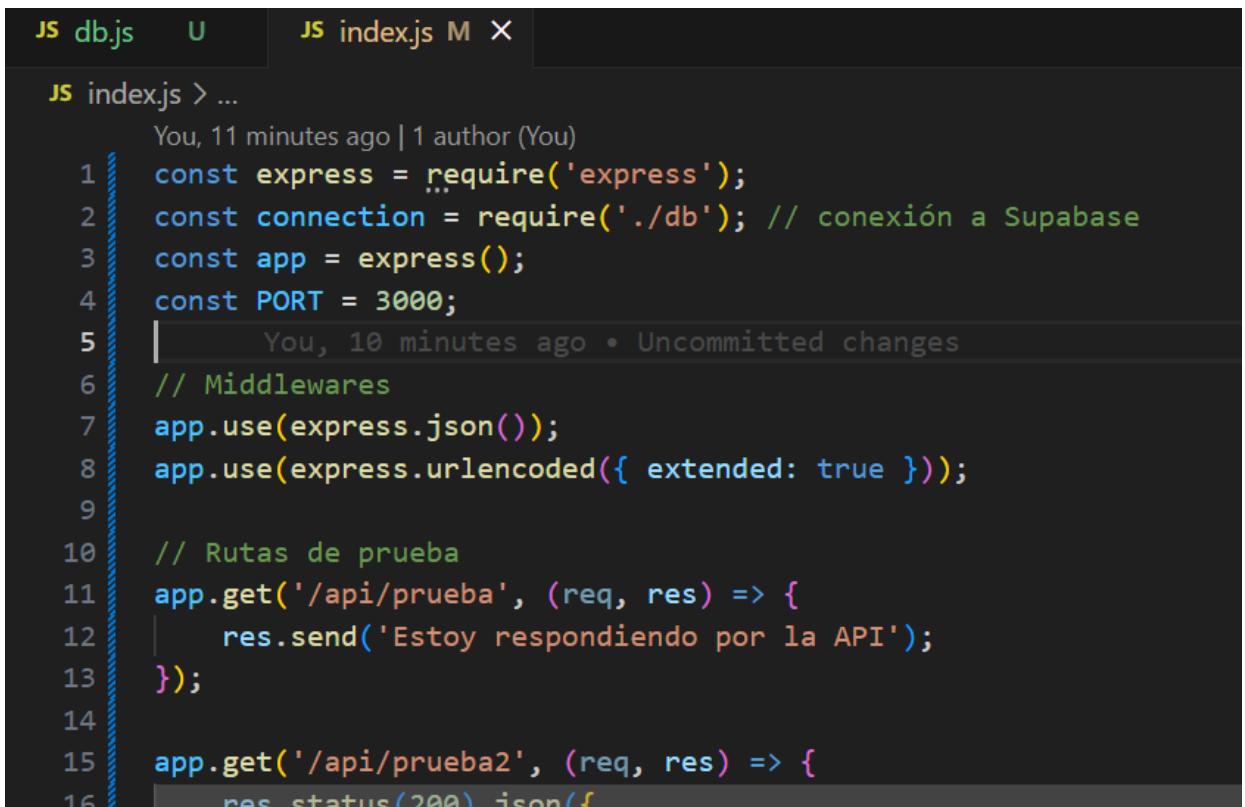
5. En el archivo **bd.js**, configuraré las conexiones a la base de datos utilizando los parámetros obtenidos de pgAdmin, como el **host**, **usuario** y **contraseña**, para establecer una conexión exitosa con PostgreSQL.

The screenshot shows the Visual Studio Code interface with the 'db.js' file open in the editor. The code is as follows:

```
JS db.js U X JS index.js M
JS db.js > ⚡ then() callback
1 const { Pool } = require('pg');
2
3 // Crear una instancia del pool
4 const pool = new Pool({
5   host: 'aws-0-us-east-1.pooler.supabase.com',
6   port: 5432,
7   user: 'postgres.jbxeibcmvpdokmkogwll',
8   password: 'LauraVel-664321',
9   database: 'postgres',
10  ssl: {
11    rejectUnauthorized: false
12  }
13});
14
15 // Verificar la conexión al iniciar (esto es solo para probar)
16 pool.query('SELECT NOW()')
  .then(res => s
```

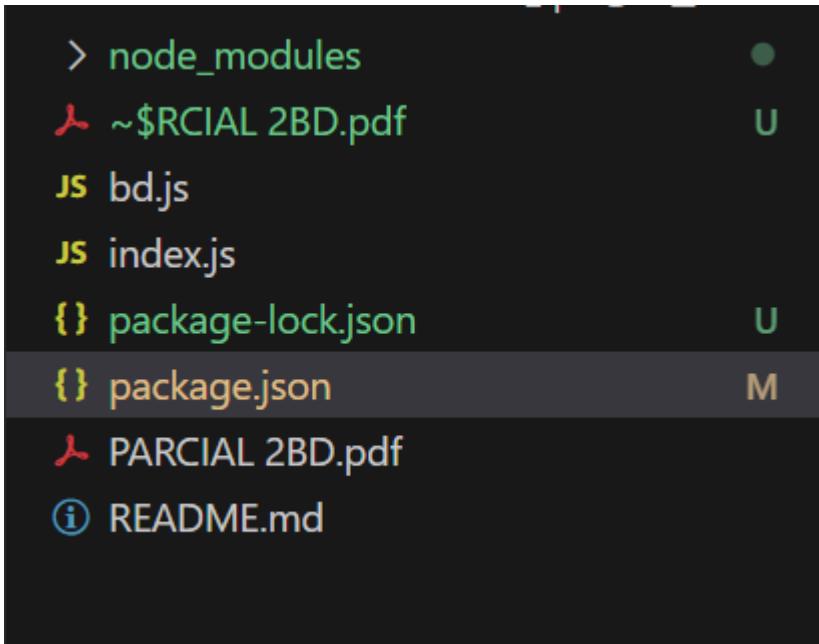
The status bar at the bottom shows tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', 'PORTS', and 'GIT LENS'.

6. Así mismo realizo con el index llamando a la bd.js



```
js db.js U js index.js M X
js index.js > ...
You, 11 minutes ago | 1 author (You)
1 const express = require('express');
2 const connection = require('./db'); // conexión a Supabase
3 const app = express();
4 const PORT = 3000;
| You, 10 minutes ago • Uncommitted changes
5
6 // Middlewares
7 app.use(express.json());
8 app.use(express.urlencoded({ extended: true }));
9
10 // Rutas de prueba
11 app.get('/api/prueba', (req, res) => {
12   res.send('Estoy respondiendo por la API');
13 });
14
15 app.get('/api/prueba2', (req, res) => {
16   res.status(200).json({
```

7. Para instalar el paquete de Node.js se realiza con el siguiente comando **npm i express pg dotenv cors**



8. Y para correr el servidor lo realice con un node – watch index.js

```
Restarting 'index.js'
El servidor está corriendo en http://localhost:3000
✓ Conectado a PostgreSQL. Hora actual: 2025-04-25T11:43:57.318Z
```

9. Me dirijo a realizar la conexión del supabase al pgadmin, para ello ingreso al supabase para así mismo asignar un nombre específico para identificarlo y se establece una clave de seguridad

supabase [Producto](#) [Desarrolladores](#) [Empresa](#) [Precios](#) [Documentos](#) [Blog](#) [81.3 mil](#) [Panel](#)

[Semana de lanzamiento 14](#) [Los 10 mejores lanzamientos](#) →

Construir en un fin de semana Escala a millones

Supabase es una alternativa de código abierto a Firebase. Inicia tu proyecto con una base de datos Postgres, autenticación, API instantáneas, funciones Edge, suscripciones en tiempo real, almacenamiento e incrustaciones vectoriales.

Crear una nueva organización

Esta es su organización dentro de Supabase.
Por ejemplo, puede utilizar el nombre de su empresa o departamento.

Nombre ¿Cómo se llama tu empresa o equipo?

Tipo ¿Qué describiría mejor su organización?

Plan El Plan se aplica a su nueva organización.

Precios [↑](#)

[Cancelar](#) Puedes cambiar el nombre de tu organización más tarde [Crear organización](#)

Crear un nuevo proyecto

Tu proyecto tendrá su propia instancia dedicada y una base de datos Postgres completa. Se configurará una API para que puedas interactuar fácilmente con tu nueva base de datos.

Organización

API parcial Gratis

Nombre del proyecto

LauVelasco's Project

Contraseña de la base de datos

.....

Copiar

Esta es la contraseña de su base de datos de Postgres, por lo que debe ser segura y difícil de adivinar. [Genere una contraseña.](#)

Región

Este de EE. UU. (Virginia del Norte)

Seleccione la región más cercana a sus usuarios para obtener el mejor rendimiento.

10. Para ello le doy en connect y me dirijo al session pooler, para así mismo realizar la conexión correspondiente del pgAdmin al supabase

Transaction pooler Shared Pooler

Ideal for stateless applications like serverless functions where each interaction with Postgres is brief and isolated.

postgresql://postgres.jbxeibcmvpdokmkogwll:[YOUR-PASSWORD]@

◀ ▶ Does not support PREPARE statements

> View parameters



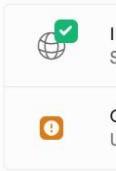
Session pooler Shared Pooler

Only recommended as an alternative to Direct Connection, when connecting via an IPv4 network.

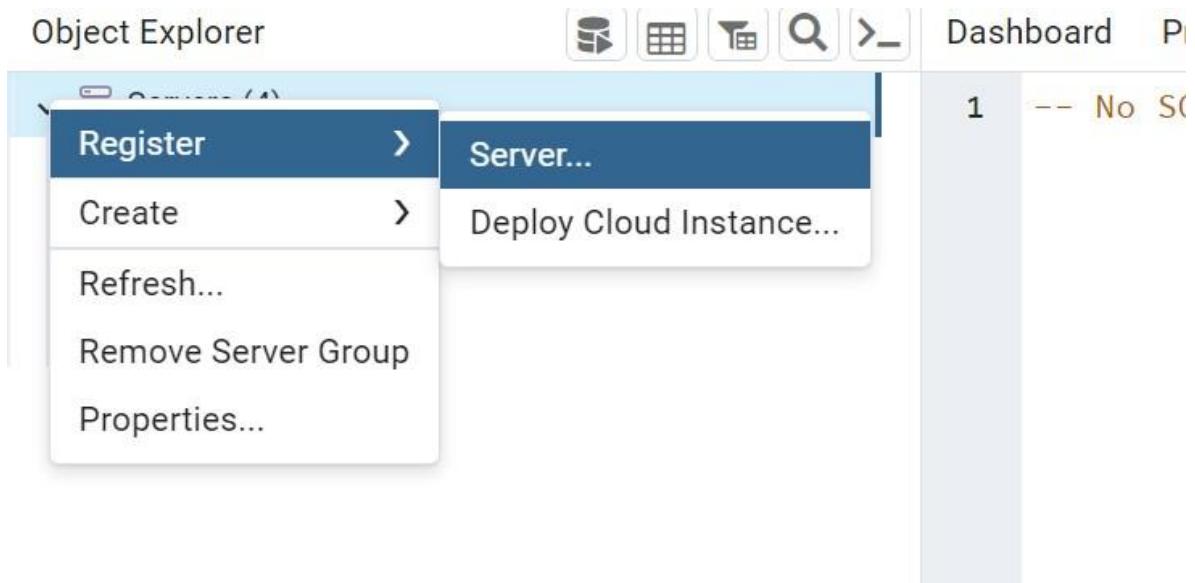
PASSWORD]@aws-0-us-east-1.pooler.supabase.com:5432/postgres

◀ ▶

> View parameters



11. Me dirijo al pg Admin para así mismo continuar con la conexión al supabase



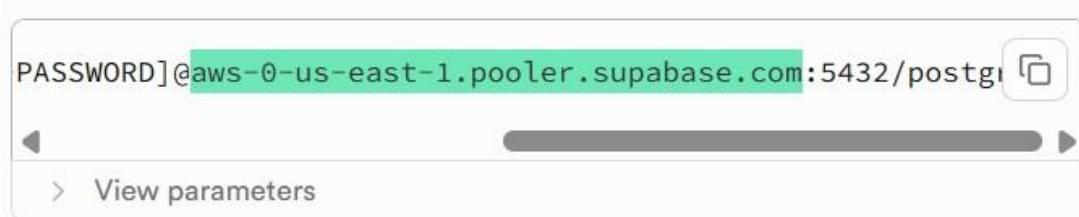
12. Para hacer la conexión tomo esta parte del enlace y la pego en username y así mismo hago con la clave de seguridad que coloque en el supabase

A screenshot of the pgAdmin 4 configuration interface for a session pooler. At the top, there are two tabs: 'Session pooler' and 'Shared Pooler', with 'Shared Pooler' being the active tab. Below the tabs, a note says 'Only recommended as an alternative to Direct Connection, when connecting via an IPv4 network.' In the main area, there is a text input field containing the connection string: 'postgresql://postgres.jbxeibcmvpdokmkogwll:[YOUR-PASS]'. Below the input field is a progress bar and a link labeled 'View parameters'.

13. Para configurar correctamente la conexión en pgAdmin 4 utilizando el Session Pooler de Supabase , es necesario extraer y definir la dirección del servidor en el campo correspondiente.Para ello se toma el enlace desede **aws** hasta **.com** y se copia en el host name.

Session pooler Shared Pooler

Only recommended as an alternative to Direct Connection, when connecting via an IPv4 network.

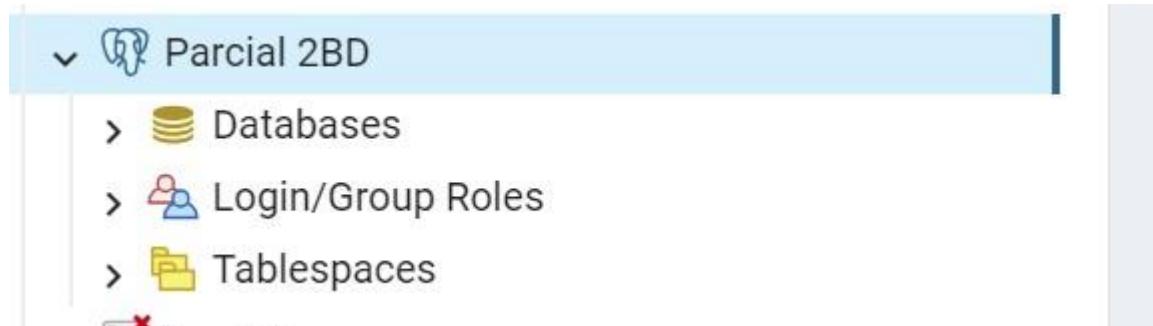
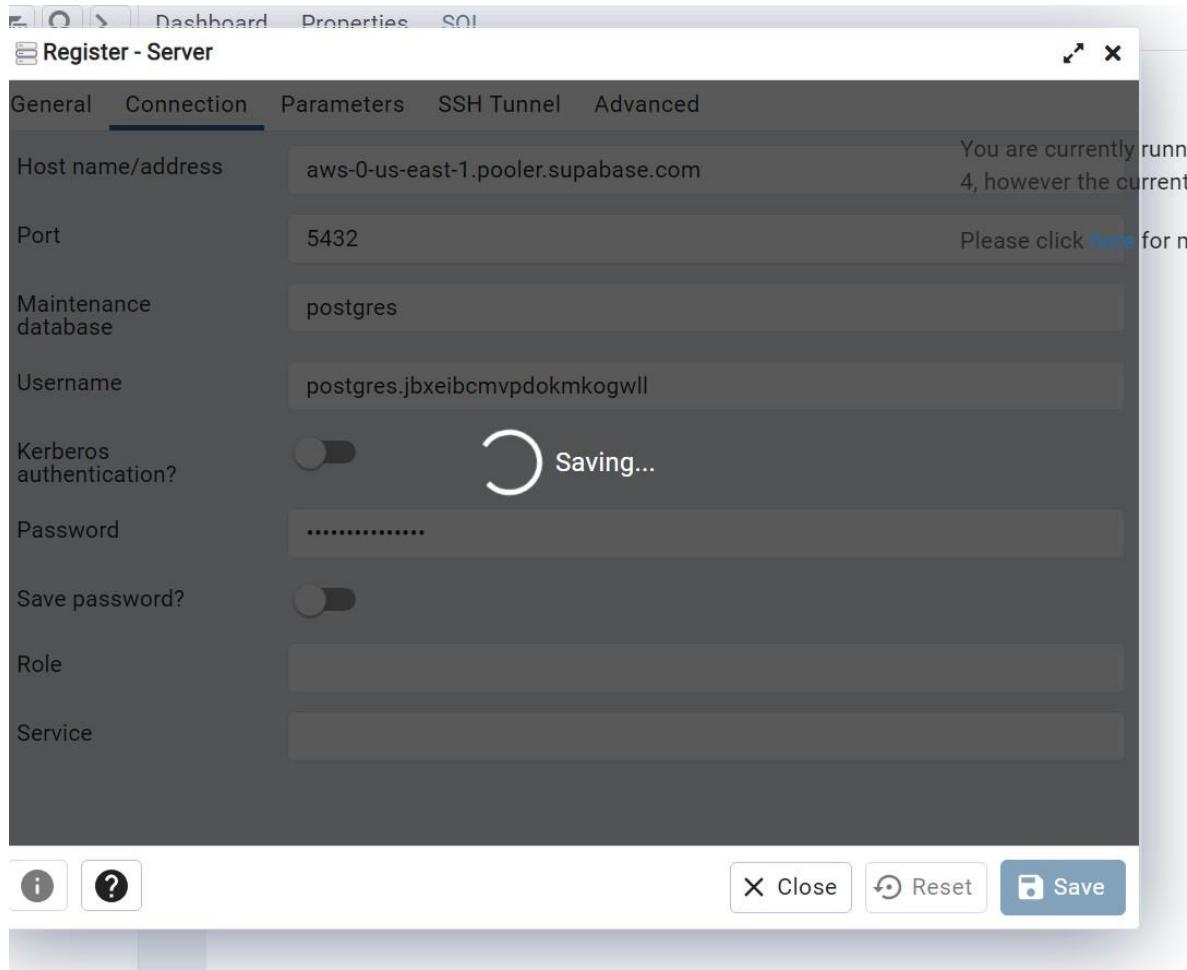


Register - Server

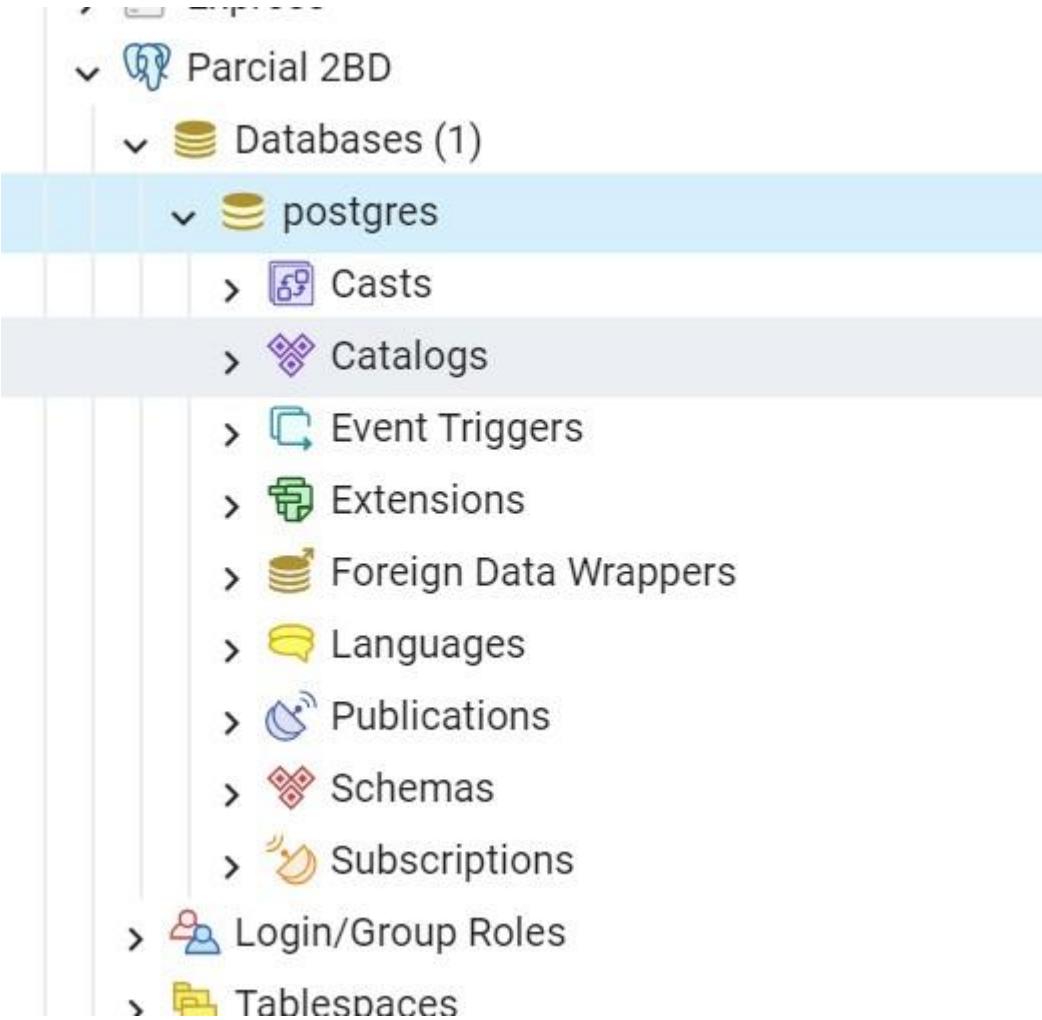
General Connection Parameters SSH Tunnel Advanced

Host name/address	aws-0-us-east-1.pooler.supabase.com	You are currently running pgAdmin 4, however the current connection is using pgAdmin 3. Please click here for more information.
Port	5432	Please click here for more information.
Maintenance database	postgres	
Username	postgres.jbxeibcmvpdokmkogwll	
Kerberos authentication?	<input type="checkbox"/>	
Password	
Save password?	<input type="checkbox"/>	

14. Para finalizar la configuración de la conexión en **pgAdmin 4**, es necesario guardar los cambios para que la conexión con la base de datos se establezca de manera correcta y persistente.



15. Durante el proceso de conexión entre pgAdmin 4 y Supabase, se genera automáticamente una base de datos predeterminada denominada **postgres**.



16. Creo la tabla restaurante

Query Query History Execute/Refresh [F5]

```
1 CREATE TABLE restaurante (
2     id_rest INT PRIMARY KEY,
3     nombre VARCHAR(100),
4     ciudad VARCHAR(100),
5     direccion VARCHAR(150),
6     fecha_apertura DATE
7 );
```

4, however the current version is 7.8. Scratch Pad x
Please click [here](#) for more information.

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 411 msec.

Total rows: 0 of 0 Query complete 00:00:00.411 ✓ Query returned successfully in 411 ms

17. Creo la tabla empleados

postgres/postgres.jbxeibcmvpdokmkogwl@Parcial 2BD Execute/Refresh [F5]

You are currently running version 7.8 of pgAdmin 4, however the current version is 9.2. Scratch Pad x
Please click [here](#) for more information.

Query Query History Execute/Refresh [F5]

```
1 CREATE TABLE empleado (
2     id_empleado INT PRIMARY KEY,
3     nombre VARCHAR(100),
4     rol VARCHAR(50),
5     id_rest INT,
6     FOREIGN KEY (id_rest) REFERENCES restaurante(id_rest)
7 );
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 306 msec.

18. Creo la tabla producto

The screenshot shows a PostgreSQL database client interface. The connection is set to 'postgres/postgres.jbxeibcmvpdokmkogwll@Parcial 2BD'. The toolbar includes standard icons for file, copy, paste, search, and refresh. A dropdown menu shows 'No limit'. The main area has tabs for 'Query' (selected) and 'Query History'. A button labeled 'Execute/Refresh' with the F5 key is highlighted with a callout bubble. The query text is:

```
1 CREATE TABLE producto (
2     id_prod INT PRIMARY KEY,
3     nombre VARCHAR(100),
4     precio NUMERIC(10,2)
5 );
```

Below the query, the status bar shows 'Data Output' and 'Messages' (selected). The message pane displays the command 'CREATE TABLE'.

19. Creo la tabla pedido

The screenshot shows a PostgreSQL database client interface. The connection is set to 'postgres/postgres.jbxeibcmvpdokmkogwll@Parcial 2BD'. The toolbar includes standard icons for file, copy, paste, search, and refresh. A dropdown menu shows 'No limit'. The main area has tabs for 'Query' (selected) and 'Query History'. A button labeled 'Execute/Refresh' with the F5 key is highlighted with a callout bubble. The query text is:

```
1 CREATE TABLE pedido (
2     id_pedido INT PRIMARY KEY,
3     fecha DATE,
4     id_rest INT,
5     total NUMERIC(10,2),
6     FOREIGN KEY (id_rest) REFERENCES restaurante(id_rest)
7 );
8
```

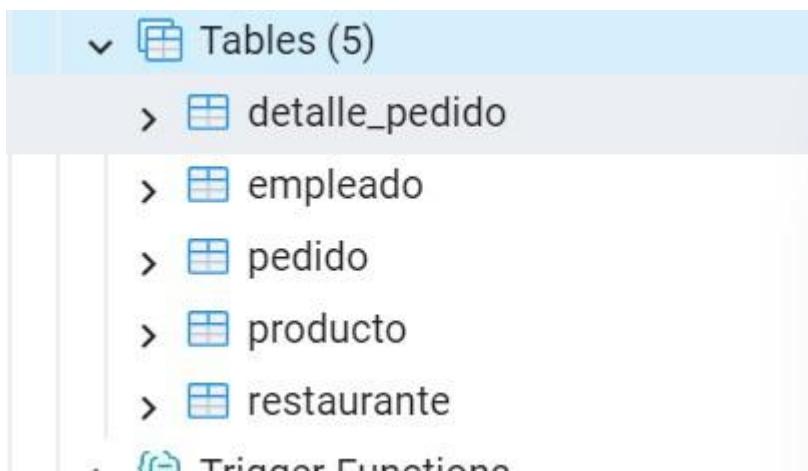
20. Crear la tabla detalle_pedido

```

CREATE TABLE detalle_pedido (
    id_detalle INT PRIMARY KEY,
    id_pedido INT,
    id_prod INT,
    cantidad INT,
    subtotal NUMERIC(10,2),
    FOREIGN KEY (id_pedido) REFERENCES pedido(id_pedido),
    FOREIGN KEY (id_prod) REFERENCES producto(id_prod)
);

```

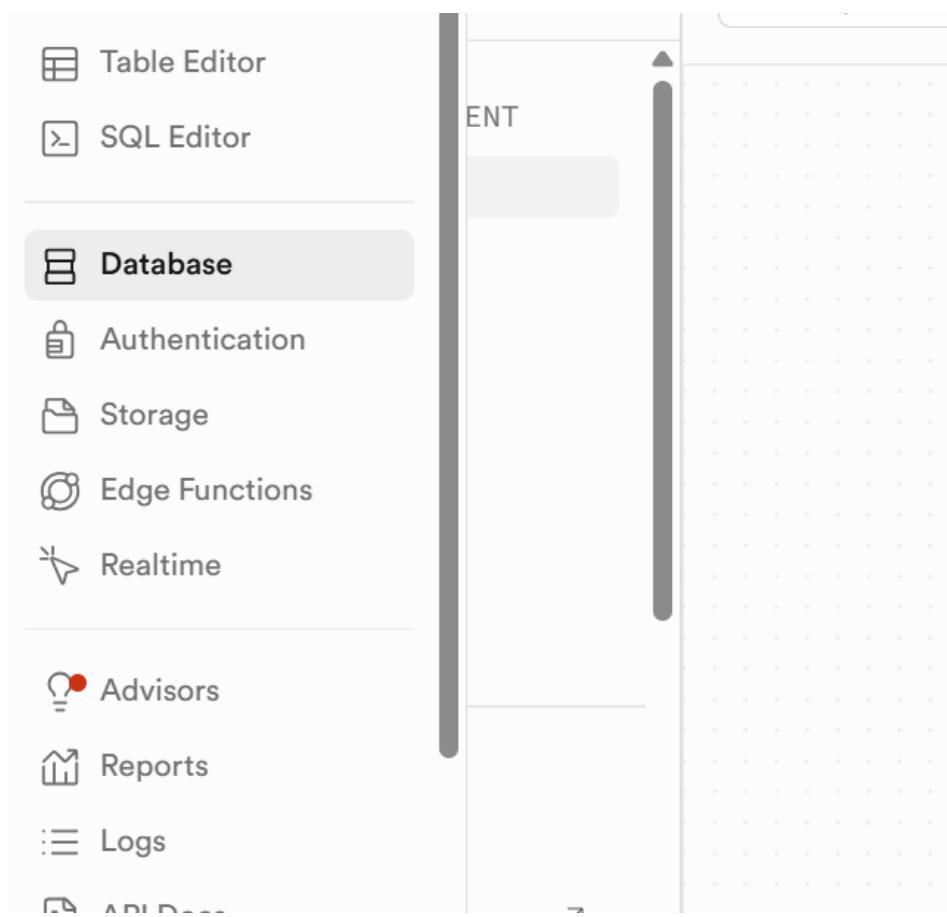
Please click



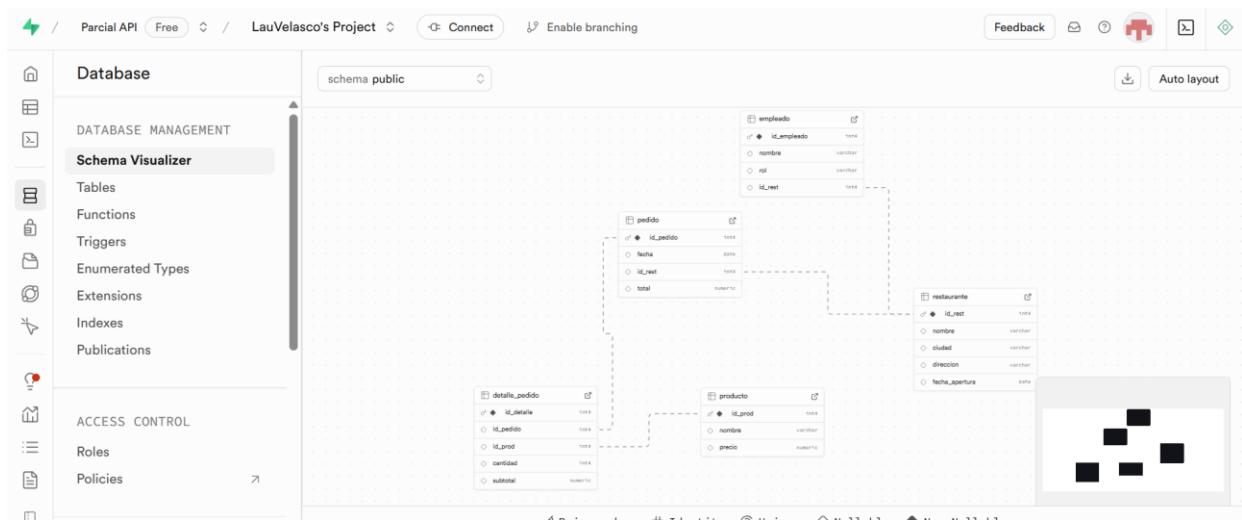
21. Me dirijo al supabase para así mismo sacar el modelo relacional y mirar como quedaron con las llaves foráneas que se le asignaron en cada tabla

	id	task
1	Create a	Read do
2	Build app	Connect
3	Deploy p	Get user
4	Connect	Upgraded
5	Deploy p	Upgraded
6	Get user	Upgraded
7	Upgraded	Upgraded

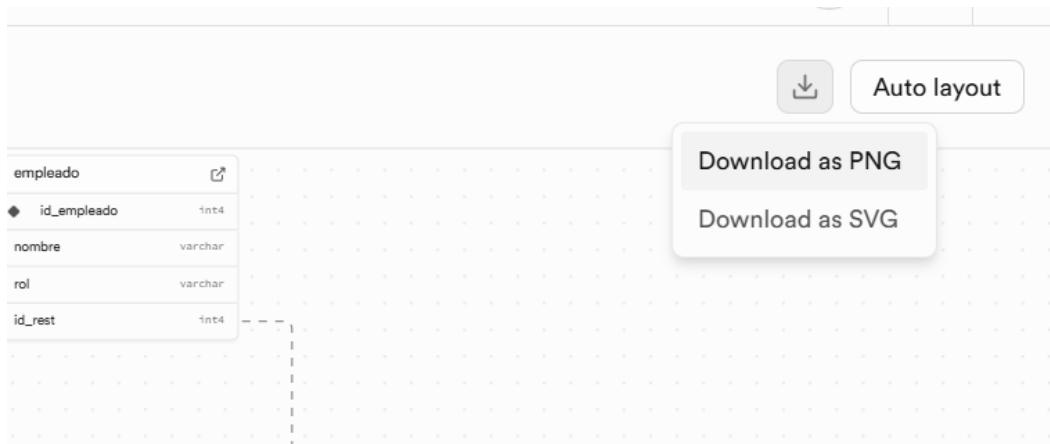
22. Para ello me dirijo a la parte de Database



23. Se visualiza como quedo el modelo relacional



24. Para descargarlo, me dirijo a la parte superior derecha donde se encuentra una flechita de descargar, le doy clic para que así mismo lo genere.



25. Para ello inicio a colocarle datos a cada tabla creada, para ello se necesitan insertar 50 datos por tabla, en el cual lo realizare en el pgadmin, para ello lo realizo con el siguiente codigo.

```
INSERT INTO restaurantes (id_rest, nombre, ciudad, direccion, fecha_apertura)
VALUES
(1, 'Restaurante El Buen Sabor', 'Bogotá', 'Calle 123 #45-67', '2020-03-15'),
```

The screenshot shows a pgAdmin 4 interface. The title bar indicates the connection is to 'postgres/postgres.jbxeibcmvpdokmkogwll@Parcial 2BD'. The toolbar has various icons for file operations, search, and database management. Below the toolbar, there are tabs for 'Query' (which is selected) and 'Query History'. The main area contains a numbered list of SQL statements:

```
1  INTO restaurante (id_rest, nombre, ciudad, direccion, fecha_apertura)
2
3  restaurante El Buen Sabor', 'Bogotá', 'Calle 123 #45-67', '2020-03-15'),
4  ushi House', 'Medellín', 'Carrera 25 #34-56', '2018-07-10'),
5  a Parrilla del Chef', 'Cali', 'Avenida 5 #56-78', '2021-11-02'),
6  izza Rápida', 'Barranquilla', 'Calle 12 #34-89', '2019-09-08'),
7  aco Fiesta', 'Cartagena', 'Avenida 1 #78-90', '2022-01-25'),
8  amburguesas y Más', 'Cúcuta', 'Calle 14 #23-45', '2020-05-30'),
9  l Rincón del Sabor', 'Bucaramanga', 'Carrera 8 #12-34', '2021-08-19'),
10 afé del Mar', 'Santa Marta', 'Calle 21 #56-90', '2017-12-05')
```

Below the query editor, there are tabs for 'Data Output', 'Messages' (which is selected), and 'Notifications'.

26. Para insertarlos me dirijo al triangulo que aparece en la parte superior para así mismo insertarlos

This screenshot is similar to the previous one, showing the pgAdmin 4 interface with the same query editor content. The difference is that the 'Execute/Refresh' button (represented by a triangle icon and labeled 'F5') is highlighted with a black box, drawing attention to it.

```
1  INTO restaurante (id_rest, nombre, ciudad, direccion, fecha_apertura)
2
3  restaurante El Buen Sabor', 'Bogotá', 'Calle 123 #45-67', '2020-03-15'),
4  ushi House', 'Medellín', 'Carrera 25 #34-56', '2018-07-10'),
5  a Parrilla del Chef', 'Cali', 'Avenida 5 #56-78', '2021-11-02'),
6  izza Rápida', 'Barranquilla', 'Calle 12 #34-89', '2019-09-08'),
7  aco Fiesta', 'Cartagena', 'Avenida 1 #78-90', '2022-01-25'),
8  amburguesas y Más', 'Cúcuta', 'Calle 14 #23-45', '2020-05-30'),
9  l Rincón del Sabor', 'Bucaramanga', 'Carrera 8 #12-34', '2021-08-19'),
10 afé del Mar', 'Santa Marta', 'Calle 21 #56-90', '2017-12-05')
```

Below the query editor, there are tabs for 'Data Output', 'Messages' (which is selected), and 'Notifications'.

File Object Tools Help

Object Explorer Dashboard Properties SQL postgres/postgres@Parcial 2BD*

Query History Execute/Refresh

```

45 (43, 'Restaurante El Árbol', 'Santafé', 'Avenida 5 #12-45', '2020-12-01')
46 (44, 'Bistro Italiano', 'Pereira', 'Carrera 8 #23-67', '2021-10-16'),
47 (45, 'Carnes del Río', 'Neiva', 'Calle 7 #34-89', '2020-07-30'),
48 (46, 'La Picantería', 'Bogotá', 'Calle 50 #12-34', '2019-02-10'),
49 (47, 'Burgers & Fries', 'Medellín', 'Carrera 15 #34-56', '2021-06-03'),
50 (48, 'La Cocina de Andrea', 'Cali', 'Calle 11 #67-23', '2020-09-14'),
51 (49, 'Sushi & Wok', 'Barranquilla', 'Avenida 9 #45-12', '2021-03-18'),
52 (50, 'La Taberna de los Sabores', 'Cartagena', 'Calle 6 #23-78', '2019-05-26')
53

```

Data Output Messages Notifications

INSERT 0 50

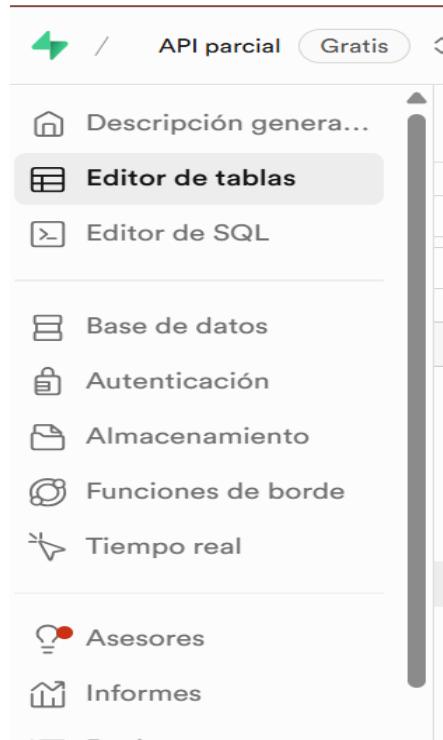
Query returned successfully in 243 msec.

Total rows: 0 of 0 Query complete 00:00:00.243

✓ Query returned successfully in 243 msec. X

Ln 53, Col 1

27. Me dirijo al supabase para así mismo revisar que los datos quedaron insertados correctamente en la tabla restaurante, así mismo me dirijo a la parte de edito de tablas



The screenshot shows the DBeaver interface with the 'restaurante' table selected in the left sidebar. The main area displays the table structure and data. The columns are: id_rest, nombre, ciudad, dirección, and fec. The data rows are:

	id_rest	nombre	ciudad	dirección	fec
	40	Comida China Express	Cartagena	Calle 13 #45-23	22
	41	La Casa del Curry	Cúcuta	Carrera 20 #34-56	5 d
	42	Pescados y Mariscos El Golfo	Bucaramanga	Calle 15 #67-89	25
	43	Restaurante El Árbol	Santa Marta	Avenida 5 #12-45	1 d
	44	Bistro Italiano	Pereira	Carrera 8 #23-67	16
	45	Carnes del Río	Neiva	Calle 7 #34-89	30
	46	La Picantería	Bogotá	Calle 50 #12-34	10
	47	Hamburguesas y papas fritas	Medellín	Carrera 15 #34-56	03
	48	La Cocina de Andrea	Cali	Calle 11 #67-23	14
	49	Sushi y wok	Barranquilla	Avenida 9 #45-12	18
	50	La Taberna de los Sabores	Cartagena	Calle 6 #23-78	20

28. Ahora inserto los datos de la tabla empleado con el mismo comando anteriormente mencionado: `INSERT INTO empleado (id_empleado, nombre, rol, id_rest) VALUES (1, 'Carlos González', 'Cocinero', 1)`...

```

1 INSERT INTO empleado (id_empleado, nombre, rol, id_rest)
2 VALUES
3 (1, 'Carlos González', 'Cocinero', 1),
4 (2, 'Ana Martínez', 'Cajero', 2),
5 (3, 'Luis Ramírez', 'Cocinero', 3),
6 (4, 'Marta Fernández', 'Cajero', 4),
7 (5, 'Pedro García', 'Mesero', 5),
8 (6, 'Sofía López', 'Cocinera', 6),
9 (7, 'José Pérez', 'Cajero', 7),
10 (8, 'Elena Díaz', 'Mesera', 8).

```

The screenshot shows the pgAdmin interface with a query editor window. The connection is set to 'postgres/postgres.jbxeibcmvpdokmkogwli@Parcial 2BD'. The query is:

```

1 INSERT INTO empleado (id_empleado, nombre, rol, id_rest)
2 VALUES
3 (1, 'Carlos González', 'Cocinero', 1),
4 (2, 'Ana Martínez', 'Cajero', 2),
5 (3, 'Luis Ramírez', 'Cocinero', 3),
6 (4, 'Marta Fernández', 'Cajero', 4),
7 (5, 'Pedro García', 'Mesero', 5),
8 (6, 'Sofía López', 'Cocinera', 6),
9 (7, 'José Pérez', 'Cajero', 7),
10 (8, 'Elena Díaz', 'Mesera', 8).

```

The 'Execute/Refresh' button is highlighted. Below the query, the 'Data Output' tab is selected, showing the message: 'INSERT 0 50'. The status bar at the bottom right indicates: '✓ Query returned successfully in 235 msec.'

29. Verifico en el supabase que los datos que inserte se hallan insertado correctamente.

The screenshot shows the Supabase Table Editor for the 'empleado' table. The table has columns: id_empleado, nombre, rol, and id_rest. The data is as follows:

	id_empleado	nombre	rol	id_rest
38	Patricia Martínez	Cocina	8	
39	Roberto Torres	Cajero	9	
40	Verónica Ruiz	Mesera	10	
41	Eduardo Álvarez	Cocinero	11	
42	Isabel Hernández	Cajera	12	
43	Tomás Jiménez	Mesero	13	
44	Cristina González	Cocina	14	
45	Antonio García	Cajero	15	
46	Tomás Pérez	Supervisor	1	
47	Jessica López	Cajera	2	
48	Manuel Rodríguez	Cocinero	3	
49	Natalia Jiménez	Mesera	4	

30. Ahora realizo lo mismo con la tabla productos :
INSERT INTO producto
(id_prod, nombre, precio) VALUES (1, 'Hamburguesa Clásica', 15000.00)...

The screenshot shows the pgAdmin 4 interface. At the top, there are tabs for Dashboard, Properties, SQL, and three connection status indicators. Below the tabs is a toolbar with various icons. The main area is a query editor with a "Query History" tab open. The history contains several numbered entries, with entry 44 highlighted. An "Execute/Refresh" button is positioned above the history list. Below the history is a "Data Output" tab, which displays the message "Query returned successfully in 135 msec." A green success message box at the bottom right also states "Query returned successfully in 135 msec.".

```
44 (42, 'Tostadas Francesas', 9500
45 (43, 'Jugo Natural de Fresa', 6000.00), -- cambiado
46 (44, 'Empanadas de Pollo (3 unidades)', 7500.00), -- cambiado
47 (45, 'Sándwich Vegetariano', 11000.00), -- cambiado
48 (46, 'Churros con Chocolate', 8000.00),
49 (47, 'Coca-Cola 350ml', 3500.00),
50 (48, 'Waffles con Frutas', 10000.00),
51 (49, 'Tamal Santandereano', 12000.00),
52 (50, 'Ensalada César con Pollo', 15000.00);
53
```

Data Output Messages Notifications

Query returned successfully in 135 msec.

✓ Query returned successfully in 135 msec. ✘

31. Así mismo verifico en el supabase que los datos que coloque quedaron insertados correctamente

The screenshot shows the Supabase Studio interface, specifically the "Products" table view. The table has columns for id, nombre, and precio. The data includes the newly inserted row (id 1, nombre 'Hamburguesa Clásica', precio 15000.00) and other existing items like Pizza de pepperoni, Taco mexicano, etc. The table includes standard database navigation buttons at the bottom.

	id_...	nombre	pre...
	1	Hamburguesa Clásica	15000.00
	2	Pizza de pepperoni	23000.00
	3	Taco mexicano	8000.00
	4	Rollo de sushi California	28000.00
	5	Ensalada César	12000.00
	6	Pollo asado	18000.00
	7	Papas fritas grandes	7000.00
	8	Jugo natural de mango	5000.00
	9	Limonada Natural	4500.00
	10	Brownie con helado	9000.00
	11	Café Expreso	4000.00

Página 1 de 1 100 filas 50 registros

32. Ahora inserto los datos de la tabla pedidos: INSERT INTO pedido (id_pedido, fecha, id_rest, total) VALUES (1, '2025-04-01', 1, 45000.00)...

The screenshot shows the pgAdmin 4 interface with a query editor window. The query is:

```
44 (42, '2025-04-21', 2, 39000.25),
45 (43, '2025-04-22', 3, 56000.00),
46 (44, '2025-04-22', 4, 78500.75),
47 (45, '2025-04-23', 5, 43000.00),
48 (46, '2025-04-23', 1, 34000.50),
49 (47, '2025-04-24', 2, 92000.00),
50 (48, '2025-04-24', 3, 25500.00),
51 (49, '2025-04-25', 4, 31500.99),
52 (50, '2025-04-25', 5, 61000.00);
53
```

Below the query, the "Data Output" tab is selected, showing the message:

```
INSERT 0 50
```

And the status message:

```
Query returned successfully in 1 secs 475 msec.
```

A green success message is displayed at the bottom right:

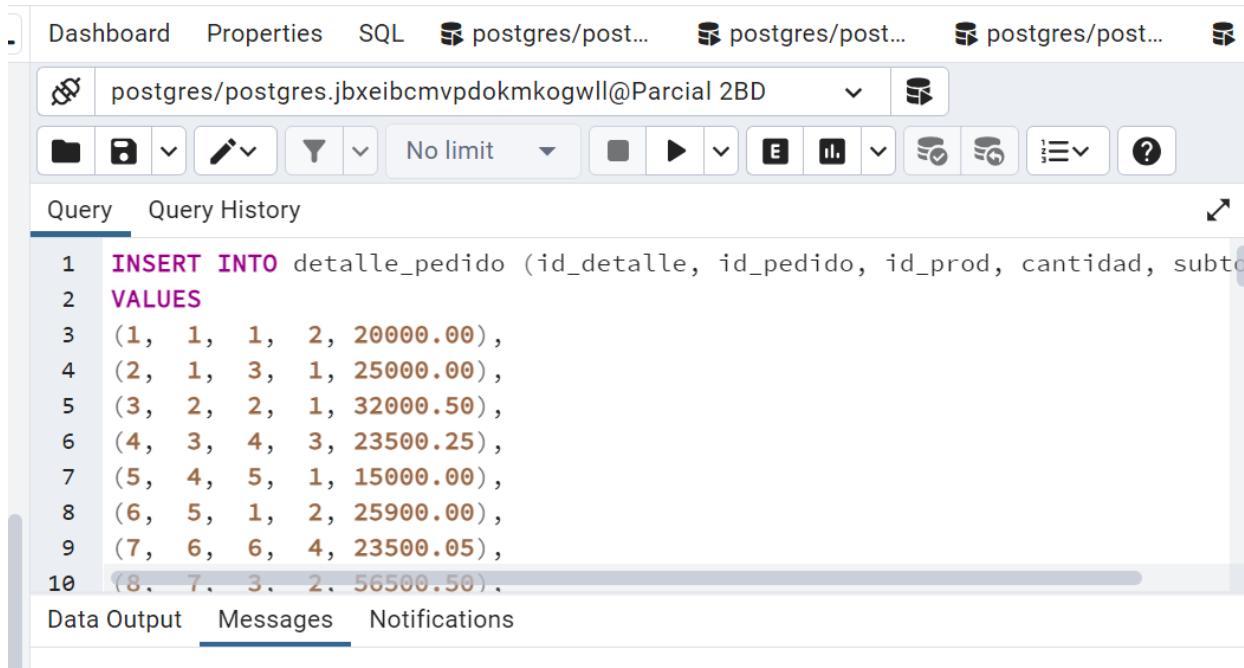
```
✓ Query returned successfully in 1 secs 475 msec. X
```

33. Verifico en el supabase si los datos de la tabla pedidos que inserte se encuentran allí.

The screenshot shows the Supabase Studio interface with a table view of the "pedidos" table. The table has columns: id_pedido, fecha, id_rest, and total. The data matches the inserted values in the previous step:

	id_pedido	fecha	id_rest	total
	40	2025-04-20	5	81000.00
	41	21 de abril de 2025	1	47000.00
	42	21 de abril de 2025	2	39000.25
	43	22 de abril de 2025	3	56000.00
	44	22 de abril de 2025	4	78500.75
	45	23 de abril de 2025	5	43000.00
	46	23 de abril de 2025	1	34000.50
	47	24 de abril de 2025	2	92000.00
	48	24 de abril de 2025	3	25500.00
	49	25 de abril de 2025	4	31500.99
	50	25 de abril de 2025	5	61000.00

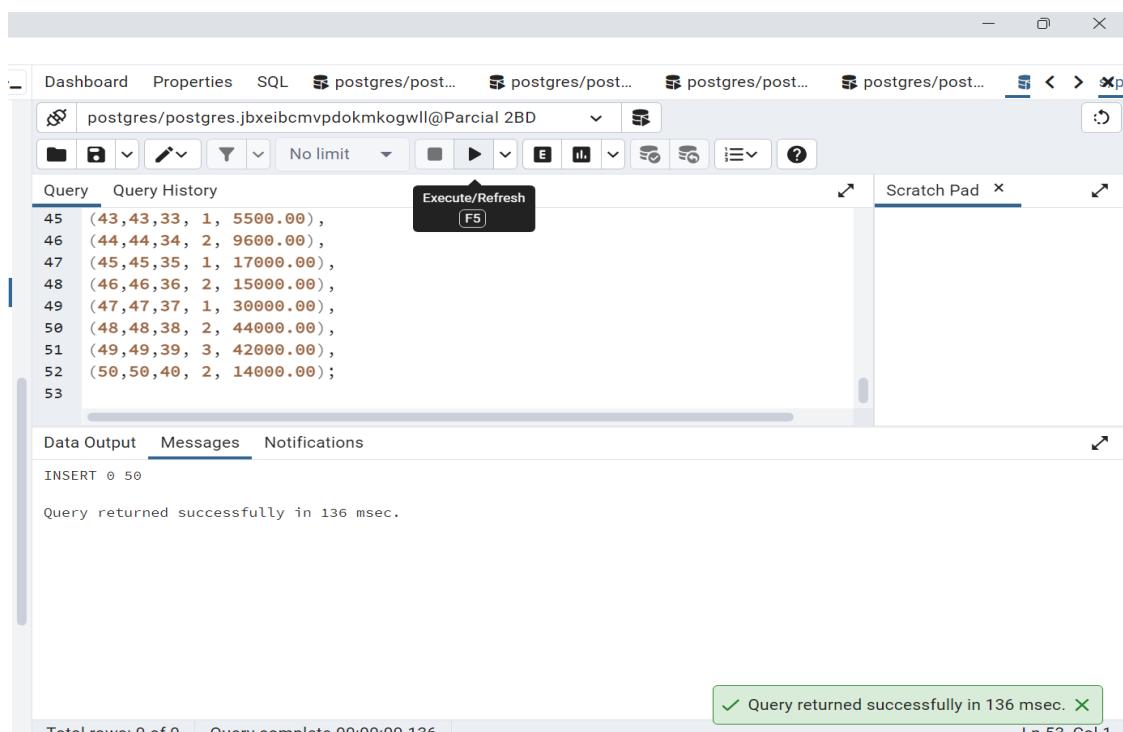
34. Inserto datos en la ultima tabla de DetallePedido INSERT INTO detalle_pedido (id_detalle, id_pedido, id_prod, cantidad, subtotal) VALUES (1, 1, 1, 1, 2, 20000.00)...



The screenshot shows the pgAdmin 4 interface with the SQL tab selected. The query editor contains the following SQL code:

```
1 INSERT INTO detalle_pedido (id_detalle, id_pedido, id_prod, cantidad, subtotal)
2 VALUES
3 (1, 1, 1, 1, 2, 20000.00),
4 (2, 1, 3, 1, 25000.00),
5 (3, 2, 2, 1, 32000.50),
6 (4, 3, 4, 3, 23500.25),
7 (5, 4, 5, 1, 15000.00),
8 (6, 5, 1, 2, 25900.00),
9 (7, 6, 6, 4, 23500.05),
10 (8, 7, 3, 2, 56500.50).
```

The status bar at the bottom indicates "Total rows: 0 of 0 Query complete 00:00:00 136".

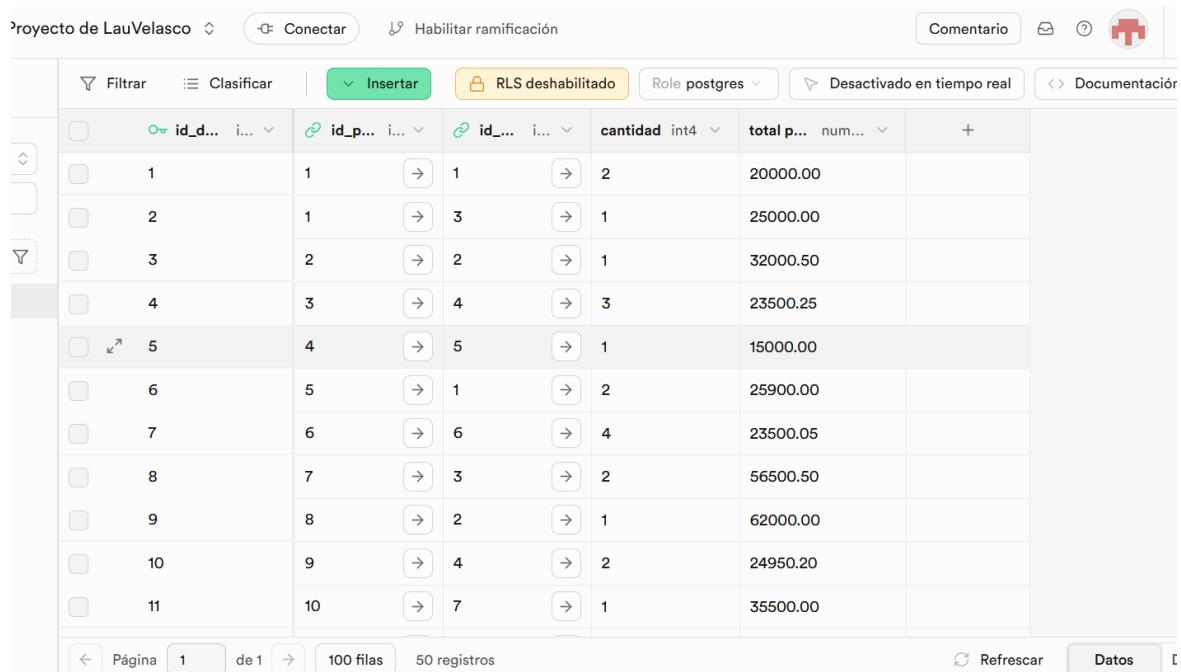


The screenshot shows the pgAdmin 4 interface with the SQL tab selected. The query editor contains the following SQL code:

```
45 (43,43,33, 1, 5500.00),
46 (44,44,34, 2, 9600.00),
47 (45,45,35, 1, 17000.00),
48 (46,46,36, 2, 15000.00),
49 (47,47,37, 1, 30000.00),
50 (48,48,38, 2, 44000.00),
51 (49,49,39, 3, 42000.00),
52 (50,50,40, 2, 14000.00);
53
```

The "Execute/Refresh" button is highlighted with a black box. The status bar at the bottom indicates "Total rows: 0 of 0 Query complete 00:00:00 136". A green message box at the bottom right says "Query returned successfully in 136 msec. X".

35. Me dirijo nuevamente al supabase para así mismo verificar.



The screenshot shows a data table in Supabase Data Studio. The table has columns: id_d... (text), id_p... (text), id_... (text), cantidad (int4), total p... (num...), and an unnamed column. The data consists of 11 rows:

	id_d...	i...	id_p...	i...	id_...	i...	cantidad	int4	total p...	num...	+
	1		1	→	1	→	2		20000.00		
	2		1	→	3	→	1		25000.00		
	3		2	→	2	→	1		32000.50		
	4		3	→	4	→	3		23500.25		
	5	↖	4	→	5	→	1		15000.00		
	6		5	→	1	→	2		25900.00		
	7		6	→	6	→	4		23500.05		
	8		7	→	3	→	2		56500.50		
	9		8	→	2	→	1		62000.00		
	10		9	→	4	→	2		24950.20		
	11		10	→	7	→	1		35500.00		

36. Ahora inicio creando cada api para cada tabla en el mismo index de la siguiente manera:

RESTAURANTE

Get

```
app.get('/api/restaurantes', (req, res) => {
  connection.query('SELECT * FROM restaurante', (err, results) => {
    if (err) return res.status(500).json({ error: err.message });
    res.json(results.rows);
  });
});
```

Post

```

app.post('/api/restaurantes', (req, res) => {
  const { nombre, ciudad, direccion, fecha_apertura } = req.body;
  connection.query(
    'INSERT INTO restaurante (nombre, ciudad, direccion, fecha_apertura) VALUES ($1, $2, $3, $4) RETURNING [nombre, ciudad, direccion, fecha_apertura]',
    (err, results) => {
      if (err) return res.status(500).json({ error: err.message });
      res.status(201).json(results.rows[0]);
    }
  );
});

```

Put

```

app.put('/api/restaurantes/:id', (req, res) => {
  const id = req.params.id;
  const { nombre, ciudad, direccion, fecha_apertura } = req.body;
  connection.query(
    'UPDATE restaurante SET nombre=$1, ciudad=$2, direccion=$3, fecha_apertura=$4 WHERE id_rest=$5 RETURNING [nombre, ciudad, direccion, fecha_apertura, id]',
    (err, results) => {
      if (err) return res.status(500).json({ error: err.message });
      res.json(results.rows[0]);
    }
  );
});

```

Delete

```

app.delete('/api/restaurantes/:id', (req, res) => {
  const id = req.params.id;
  connection.query('DELETE FROM restaurante WHERE id_rest=$1', [id], (err) => {
    if (err) return res.status(500).json({ error: err.message });
    res.status(204).send();
  });
});

```

EMPLEADOS

Get

```

app.get('/api/empleados', (req, res) => {
  connection.query('SELECT * FROM empleado', (err, results) => {
    if (err) return res.status(500).json({ error: err.message });
    res.json(results.rows);
  });
});

```

Post

```
app.post('/api/empleados', (req, res) => {
  const { nombre, rol, id_rest } = req.body;
  connection.query(
    'INSERT INTO empleado (nombre, rol, id_rest) VALUES ($1, $2, $3) RETURNING *',
    [nombre, rol, id_rest],
    (err, results) => {
      if (err) return res.status(500).json({ error: err.message });
      res.status(201).json(results.rows[0]);
    }
  );
});
```

Put

```
app.put('/api/empleados/:id', (req, res) => {
  const id = req.params.id;
  const { nombre, rol, id_rest } = req.body;
  connection.query(
    'UPDATE empleado SET nombre=$1, rol=$2, id_rest=$3 WHERE id_empleado=$4 RETURNING *',
    [nombre, rol, id_rest, id],
    (err, results) => {
      if (err) return res.status(500).json({ error: err.message });
      res.json(results.rows[0]);
    }
  );
});
```

Delete

```
app.delete('/api/empleados/:id', (req, res) => {
  const id = req.params.id;
  connection.query('DELETE FROM empleado WHERE id_empleado=$1', [id], (err) => {
    if (err) return res.status(500).json({ error: err.message });
    res.status(204).send();
  });
});
```

PRODUCTOS

Get

```
app.get('/api/productos', (req, res) => {
  connection.query('SELECT * FROM producto', (err, results) => {
    if (err) return res.status(500).json({ error: err.message });
    res.json(results.rows);
  });
});
```

Post

```
120 app.post('/api/productos', (req, res) => {
121   const { nombre, precio } = req.body;
122   connection.query(
123     'INSERT INTO producto (nombre, precio) VALUES ($1, $2) RETURNING *',
124     [nombre, precio],
125     (err, results) => {
126       if (err) return res.status(500).json({ error: err.message });
127       res.status(201).json(results.rows[0]);
128     }
129   );
130 });
131 );
```

Put

```
132 app.put('/api/productos/:id', (req, res) => {
133   const id = req.params.id;
134   const { nombre, precio } = req.body;
135   connection.query(
136     'UPDATE producto SET nombre=$1, precio=$2 WHERE id_prod=$3 RETURNING *',
137     [nombre, precio, id],
138     (err, results) => {
139       if (err) return res.status(500).json({ error: err.message });
140       res.json(results.rows[0]);
141     }
142   );
143 });
144 );
```

Delete

```

145 app.delete('/api/productos/:id', (req, res) => {
146   const id = req.params.id;
147   connection.query('DELETE FROM producto WHERE id_prod=$1', [id], (err) => {
148     if (err) return res.status(500).json({ error: err.message });
149     res.status(204).send();
150   });
151 });
152

```

PEDIDOS

Get

```

156 app.get('/api/pedidos', (req, res) => {
157   connection.query('SELECT * FROM pedido', (err, results) => {
158     if (err) return res.status(500).json({ error: err.message });
159     res.json(results.rows);
160   });
161 });
162

```

Post

```

163 app.post('/api/pedidos', (req, res) => {
164   const { fecha, id_rest, total } = req.body;
165   connection.query(
166     'INSERT INTO pedido (fecha, id_rest, total) VALUES ($1, $2, $3) RETURNING *',
167     [fecha, id_rest, total],
168     (err, results) => {
169       if (err) return res.status(500).json({ error: err.message });
170       res.status(201).json(results.rows[0]);
171     }
172   );
173 });
174

```

Put

```

175 app.put('/api/pedidos/:id', (req, res) => {
176   const id = req.params.id;
177   const { fecha, id_rest, total } = req.body;
178   connection.query(
179     'UPDATE pedido SET fecha=$1, id_rest=$2, total=$3 WHERE id_pedido=$4 RETURNING *',
180     [fecha, id_rest, total, id],
181     (err, results) => {
182       if (err) return res.status(500).json({ error: err.message });
183       res.json(results.rows[0]);
184     }
185   );
186 });
187

```

Delete

```
.88 app.delete('/api/pedidos/:id', (req, res) => {
.89   const id = req.params.id;
.90   connection.query('DELETE FROM pedido WHERE id_pedido=$1', [id], (err) => {
.91     if (err) return res.status(500).json({ error: err.message });
.92     res.status(204).send();
.93   });
.94 });
.95
```

DETALLEPEDIDO

Get

```
app.get('/api/detalles', (req, res) => {
  connection.query('SELECT * FROM detallepedido', (err, results) => {
    if (err) return res.status(500).json({ error: err.message });
    res.json(results.rows);
  });
});
```

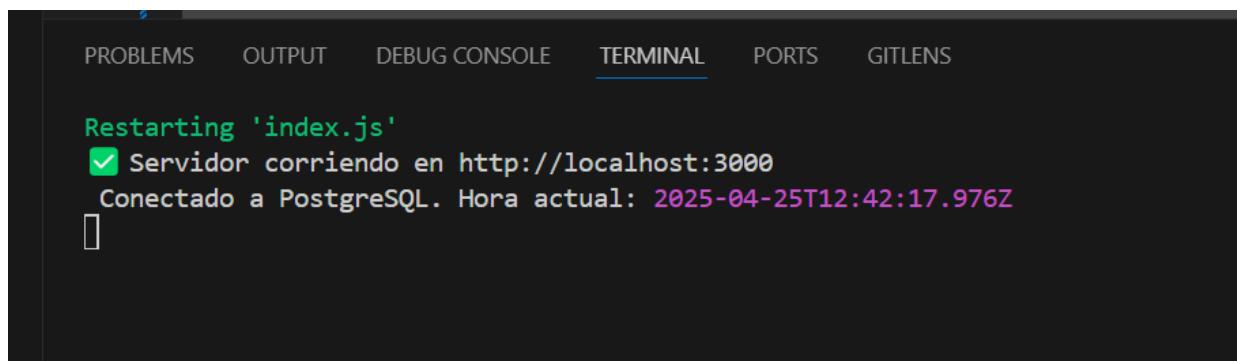
Post

```
206 app.post('/api/detalles', (req, res) => {
207   const { id_pedido, id_prod, cantidad, subtotal } = req.body;
208   connection.query(
209     'INSERT INTO detallepedido (id_pedido, id_prod, cantidad, subtotal) VALUES ($1, $2, $3, $4) RETURNING [id_pedido, id_prod, cantidad, subtotal]',
210     (err, results) => {
211       if (err) return res.status(500).json({ error: err.message });
212       res.status(201).json(results.rows[0]);
213     }
214   );
215 });
216
```

Put

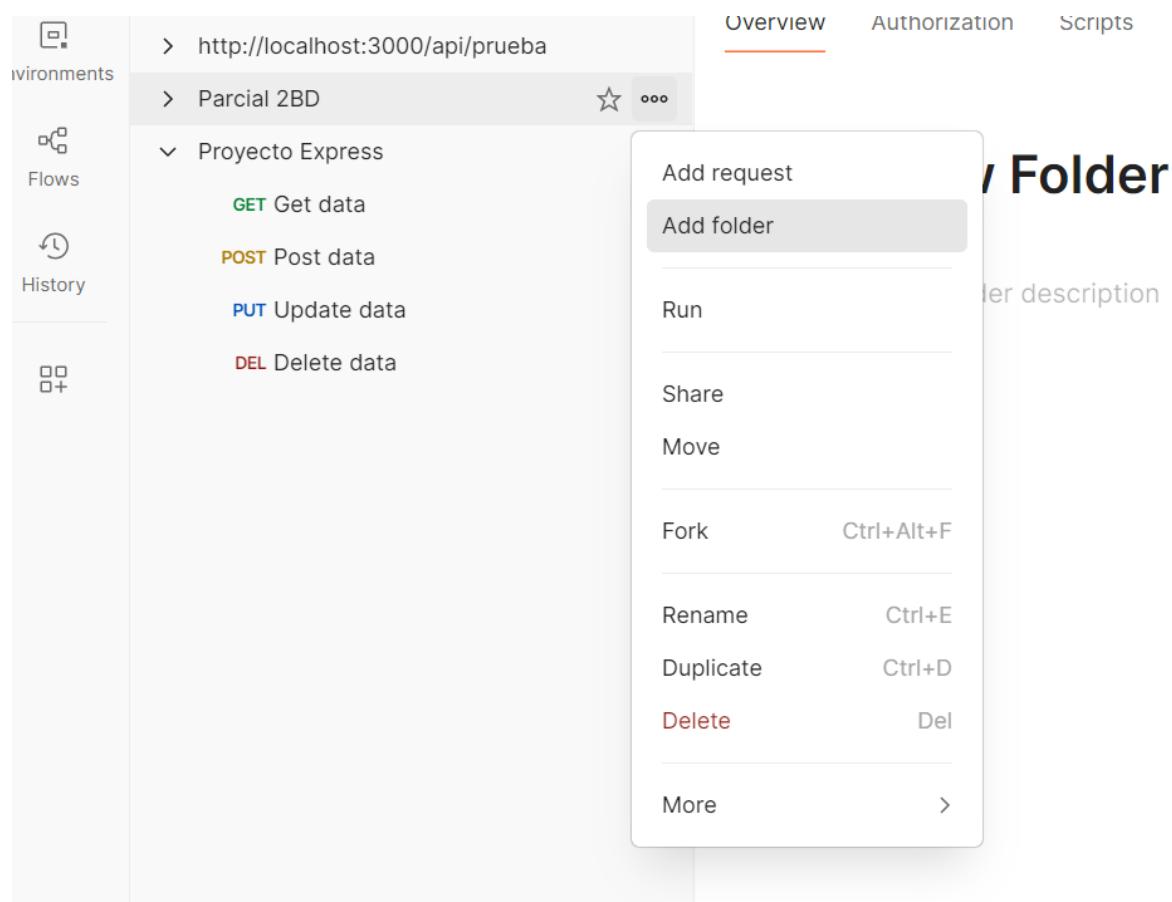
```
218 app.put('/api/detalles/:id', (req, res) => {
219   const id = req.params.id;
220   const { id_pedido, id_prod, cantidad, subtotal } = req.body;
221   connection.query(
222     'UPDATE detallepedido SET id_pedido=$1, id_prod=$2, cantidad=$3, subtotal=$4 WHERE id_detalle=$5',
223     [id_pedido, id_prod, cantidad, subtotal, id],
224     (err, results) => {
225       if (err) return res.status(500).json({ error: err.message });
226       res.json(results.rows[0]);
227     }
228   );
229 });
230
```

37. De igual manera tiene que salir que el servidor está corriendo

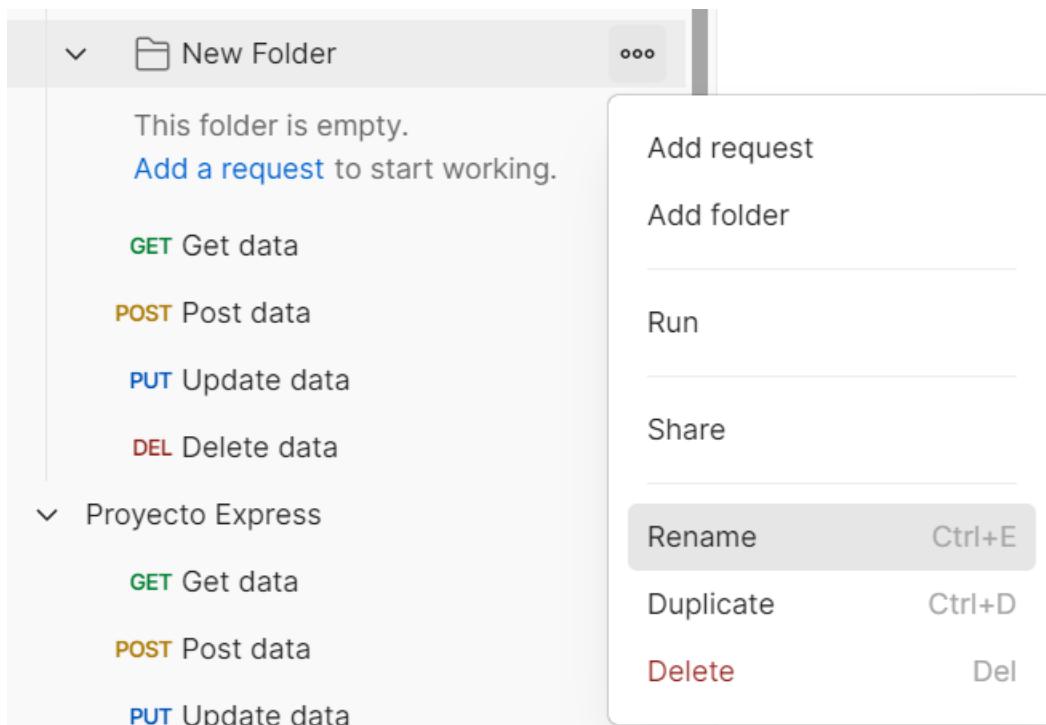


```
Restarting 'index.js'
✓ Servidor corriendo en http://localhost:3000
Conectado a PostgreSQL. Hora actual: 2025-04-25T12:42:17.976Z
```

38. Realizo unas subcarpetas en el postan de cada tabla así mismo coloco cada CRUD, para realizarlas me ubico en la carpeta de parcial 2BD en los tres puntos que se encuentran al lado de la carpeta, me ubico en la parte donde dice add folder y le doy clic



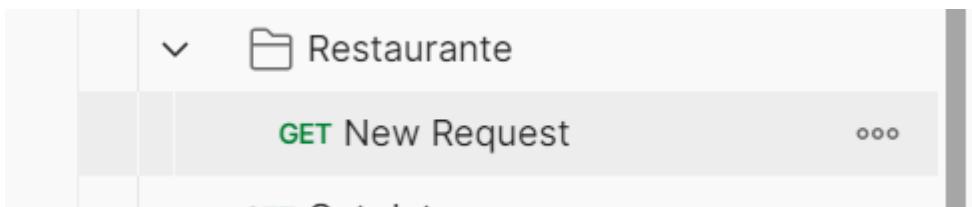
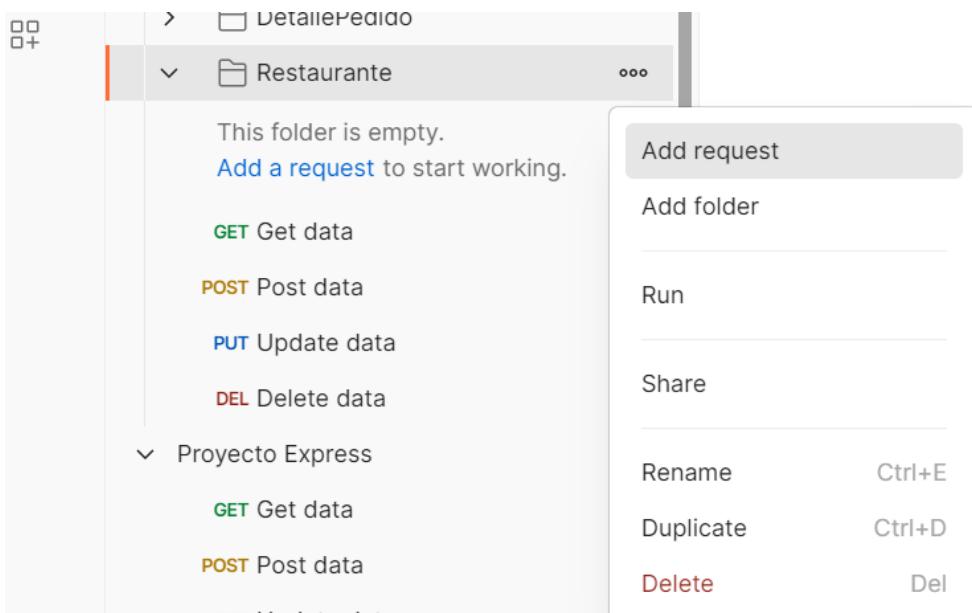
39. Al momento de darle clic me aparecerá nuevo folder, allí lo renombrare y le colocare cada una de las tablas que se crearon anteriormente como lo son: Restaurantes, empleados, productos, pedidos, detallepedido



40. Al momento de renombrar cada carpeta quedara de la siguiente manera

The screenshot shows a file explorer window with a sidebar on the left and a main content area on the right. In the sidebar, under the heading 'Parcial 2BD', there are five folder icons: 'Restaurantes', 'Empleados', 'Producto', 'Pedido', and 'DetallePedido'. Below these, there is a 'GET Get data' item. In the main content area, the word 'Parcial 2E' is prominently displayed in large bold letters. To its right, there is a small rocket icon and the text 'Get sta'. Further down, it says 'This template guid' and 'DELETE), variables'.

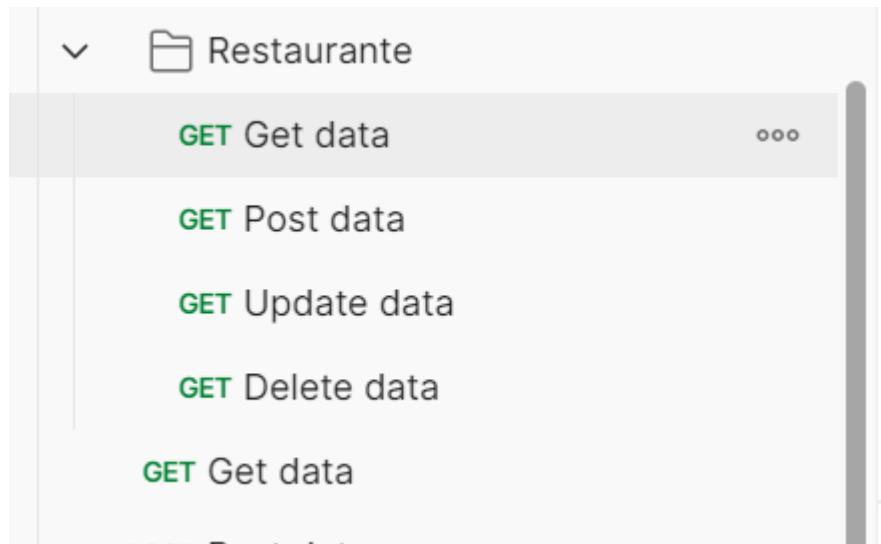
41. Para comenzar a configurar cada una de las peticiones de la API, debo seguir los siguientes pasos: hacer clic en los tres puntos del menú, ubicarme en la opción "Add Request" y, a continuación, se mostrará una nueva solicitud, por defecto del tipo GET, la cual puedo editar según la operación deseada.



42. Para continuar, debo repetir el mismo procedimiento descrito anteriormente, ya que en total se deben crear cuatro solicitudes de tipo GET, cada una destinada a consultar diferentes recursos de la API.



43. Debo comenzar renombrando cada una de las solicitudes GET, asignándoles un nombre que refleje claramente la operación o recurso que consultan, para mantener una mejor organización y comprensión dentro del entorno de pruebas.



44. Una vez renombradas correctamente las solicitudes GET según su función, procedo a ingresar en cada una de ellas para modificar la URL correspondiente. Este paso se realiza ubicándome en el campo donde se coloca el enlace de la petición, y allí inserto la ruta específica proporcionada por el servidor local de la API.

A screenshot of a software interface showing the configuration of a specific API endpoint. On the left, there's a sidebar with sections for 'Collections', 'Environments', 'Flows', and 'History'. The main area shows a list of endpoints under a 'Restaurante' folder. One endpoint, 'GET Update data', is selected and highlighted with a gray background. On the right, a detailed configuration panel is open for this endpoint. It shows the method dropdown set to 'GET'. Below it is a URL input field containing the text 'Parcial 2BD / Restaurante / Update data'. To the right of the URL are tabs for 'Headers (6)', 'Body', and 'Script'. A large list of HTTP methods is visible on the right side of the panel, including GET, POST, PUT, PATCH, DELETE, HEAD, and OPTIONS. At the bottom of the configuration panel, there's a text input field with the placeholder 'Type a new method'.

The screenshot shows a left sidebar with a tree view of API endpoints and a right panel for sending requests.

Left Sidebar (API Endpoints):

- GET Put data
- GET Delete data
- >
- >
- >
- ▽
 - GET Get data
 - GET Post data** (highlighted)
 - GET Update data
 - GET Delete data
 - GET Get data
- POST Post data**

Right Panel (Request Form):

- Method dropdown: **POST** (highlighted with a blue box)
- URL input field: Enter URL or paste text
- Method list:
 - GET
 - POST** (highlighted with a grey box)
 - PUT
 - PATCH
 - DELETE
 - HEAD
 - OPTIONS
- Type a new method input field

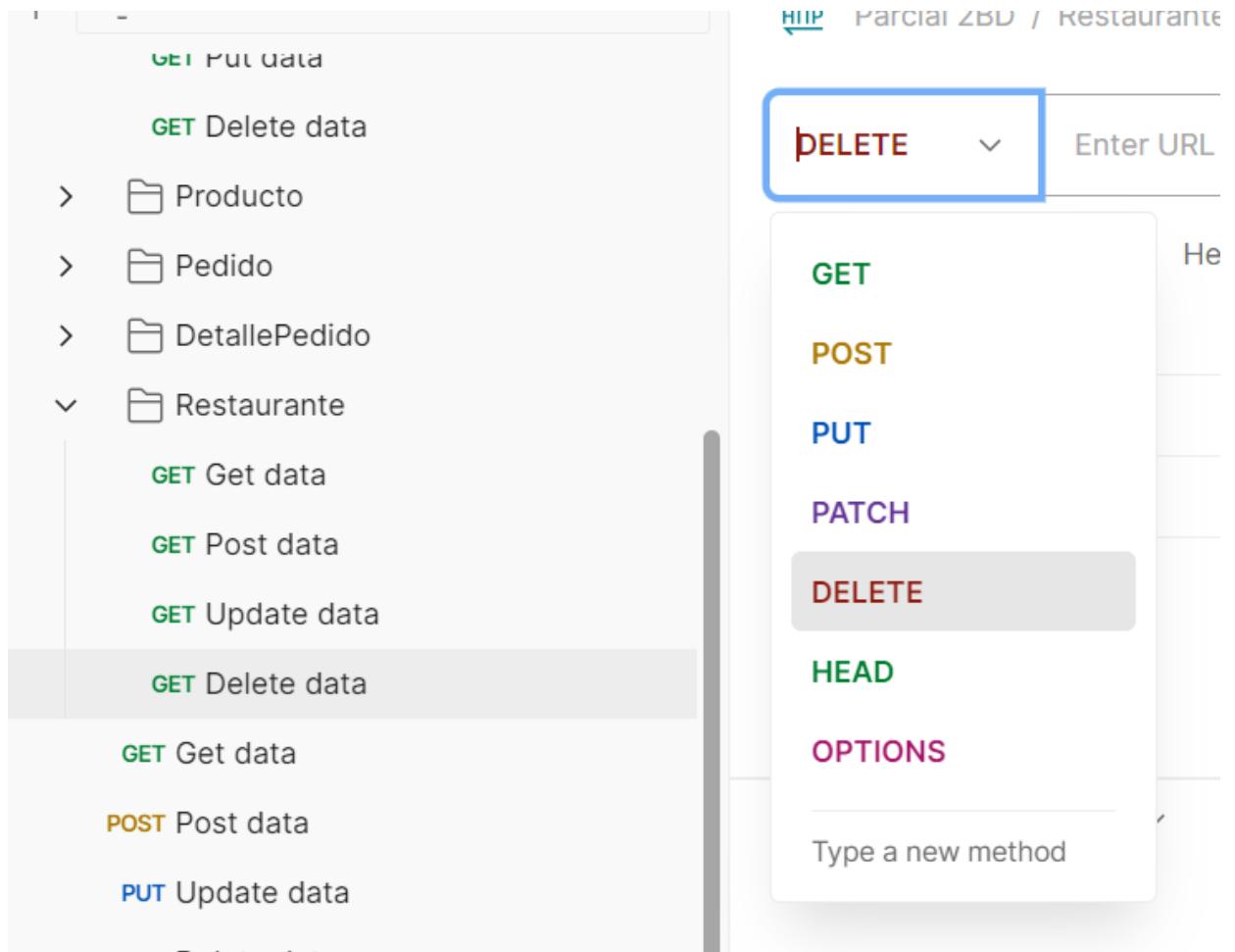
The screenshot shows a left sidebar with a tree view of API endpoints and a right panel for sending requests.

Left Sidebar (API Endpoints):

- GET Put data
- GET Delete data
- >
- >
- >
- ▽
 - GET Get data
 - GET Post data** (highlighted)
 - GET Update data
 - GET Delete data
 - GET Get data
- POST Post data**
- PUT Update data**
- DEL Delete data**

Right Panel (Request Form):

- Method dropdown: **PUT** (highlighted with a blue box)
- URL input field: Enter URL or paste text
- Method list:
 - GET
 - POST** (highlighted with a grey box)
 - PUT** (highlighted with a grey box) (highlighted with a blue box)
 - PATCH
 - DELETE
 - HEAD
 - OPTIONS
- Type a new method input field



45. Tal como se realizó anteriormente, ahora es posible visualizar que cada una de las solicitudes GET que fueron renombradas aparece correctamente con su respectiva petición. Esto permite identificar fácilmente qué recurso consulta cada solicitud, facilitando así el proceso de pruebas y validación de la API.

My Workspace

New Import

POST Pos • PUT Upda • DEL Delet •

Collections

Environments

Flows

History

HTTP Parcial 2BD / Restaurante / Get data

GET

Params Authorization Headers (6)

Query Params

	Key
	Key

Response History

GET Put data

GET Delete data

> Producto

> Pedido

> DetallePedido

< Restaurante

GET Get data

GET Post data

GET Update data

GET Delete data

GET Get data

POST Post data

PUT Update data

DEL Delete data

HTTP Parcial 2BD / Restaurante / Post data

POST

Params Authorization Headers (7) Body

Query Params

	Key	V
	Key	V

GET Put data

GET Delete data

> Producto

> Pedido

> DetallePedido

< Restaurante

GET Get data

GET Post data

GET Update data

GET Delete data

The screenshot shows the Postman interface with the left sidebar titled 'My Workspace' containing collections like 'Collections', 'Environments', 'Flows', and 'History'. The main area displays a 'PUT' request for 'Update data' under the 'Restaurante' collection. The 'Headers' tab is selected, showing 7 items. A green arrow points from the text above to the 'PUT' button.

The screenshot shows the Postman interface with the left sidebar titled 'My Workspace' containing collections like 'Collections', 'Environments', 'Flows', and 'History'. The main area displays a 'DELETE' request for 'Delete data' under the 'Restaurante' collection. The 'Headers' tab is selected, showing 6 items. A green arrow points from the text above to the 'DELETE' button.

46. Del mismo modo, realizo este proceso con cada una de las subcarpetas que he creado para organizar las diferentes peticiones de la API. Dentro de cada subcarpeta, las solicitudes aparecerán claramente clasificadas según su funcionalidad, lo que facilita la navegación y el manejo del entorno de pruebas.

Restaurante

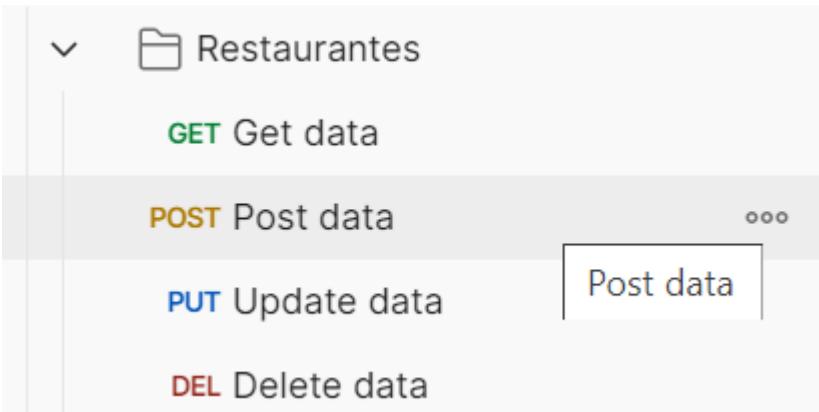
▼ Restaurantes

GET Get data

POST Post data ...

PUT Update data Post data

DEL Delete data



Empleados

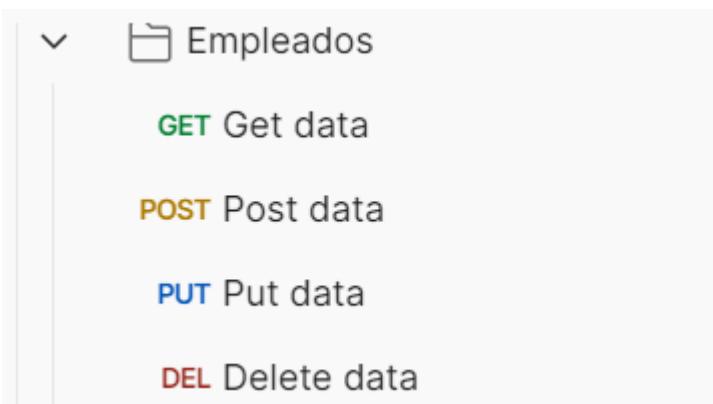
▼ Empleados

GET Get data

POST Post data

PUT Put data

DEL Delete data



Producto

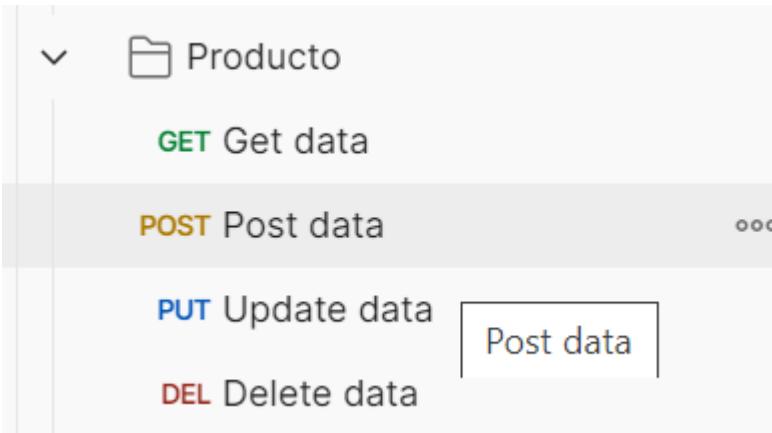
▼ Producto

GET Get data

POST Post data ...

PUT Update data Post data

DEL Delete data



Pedido

The screenshot shows a REST API endpoint for 'Pedido'. It includes a 'GET' method for 'Get data', a 'PUT' method for 'Update data', a 'POST' method for 'Post data', and a 'DEL' method for 'Delete data'.

DetallePedido

The screenshot shows a REST API endpoint for 'DetallePedido'. It includes a 'GET' method for 'Get data', a 'POST' method for 'Post data', a 'PUT' method for 'Update data', and a 'DEL' method for 'Delete data'.

47. Para visualizar correctamente cada petición utilizando el enlace <http://localhost:3000/api/...>, debo acceder individualmente a cada una de ellas y asegurarme de que la URL esté bien configurada según el recurso que se desea consultar.

Restaurante

The screenshot shows a cURL interface with a 'GET' method selected. The URL field contains 'http://localhost:3000/api/restaurantes'. Below the URL, there are tabs for 'Params', 'Authorization', 'Headers (6)', 'Body', 'Scripts', 'Settings', and 'Cookies'. A 'Send' button is located to the right. Under 'Params', there is a table titled 'Query Params' with columns 'Key', 'Value', 'Description', and 'Bulk Edit'. There is one row in the table with the key 'Key' and value 'Value'.

48. Le doy clic en send y de allí me ubico en boy, después de ello me ubico en raw

GET Send

Params Authorization Headers (6) **Body** Scripts Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

HTTP Parcial 2BD / Restaurantes / Get data Save Share

GET Send

Params Authorization Headers (6) **Body** Scripts Settings Cookies

none form-data x-www-form-urlencoded raw **JSON** Beautify

1

49. En este punto, se mostrarán los registros que fueron previamente insertados en la tabla **restaurante**, permitiéndome verificar que la conexión con la base de datos y la consulta a través de la API están funcionando correctamente.

```

1 [ 
2   {
3     "id_rest": 1,
4     "nombre": "Restaurante El Buen Sabor",
5     "ciudad": "Bogotá",
6     "direccion": "Calle 123 #45-67",
7     "fecha_apertura": "2020-03-15T05:00:00.000Z"
8   },
9   {
10    "id_rest": 2,
11    "nombre": "Sushi House",
12    "ciudad": "Medellín",
13    "direccion": "Carrera 25 #34-56",
14    "fecha_apertura": "2018-07-10T05:00:00.000Z"
15  },
16 ]

```

CONSULTAS

50. Obtener todos los productos de un pedido específico

```
// 1. Mostrar los productos de un pedido
app.get('/api/pedido/:id/productos', (req, res) => {
  const idPedido = req.params.id;
  const query = `
    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 = $1
  `;
  connection.query(query, [idPedido], (err, results) => {
    if (err) return res.status(500).json({ error: err.message });
    res.json(results.rows);
  });
});
```

The screenshot shows the Postman interface with a GET request to `http://localhost:3000/api/restaurantes`. The response body is a JSON array with one element:

```
[{"id_rest": 1, "nombre": "Sushi Bar", "ciudad": "Barcelona", "direccion": "Calle Marina 200", "fecha_apertura": "2024-01-01"}]
```

The response status is 200 OK with a duration of 936 ms and a size of 6.99 KB.

51. Obtener los productos más vendidos (más de X unidades)

```
// 2. Empleados por restaurante
app.get('/api/restaurante/:id/empleados', (req, res) => {
  const idRest = req.params.id;
  connection.query('SELECT * FROM empleado WHERE id_rest = $1', [idRest], (err, results) => {
    if (err) return res.status(500).json({ error: err.message });
    res.json(results.rows);
  });
});
```

52. Obtener el total de ventas por restaurante

```
// 3. Total de ventas por restaurante
app.get('/api/ventas/restaurantes', (req, res) => {      You, 31 minutes ago • Uncommitted change
  const query = `SELECT r.nombre, SUM(p.total) AS total_ventas
  FROM restaurante r
  JOIN pedido p ON r.id_rest = p.id_rest
  GROUP BY r.nombre
  ORDER BY total_ventas DESC
`;
  connection.query(query, (err, results) => {
    if (err) return res.status(500).json({ error: err.message });
    res.json(results.rows);
  });
});
```

53. Obtener los pedidos realizados en una fecha específica

```
// 4. Obtener los pedidos realizados en una fecha específica
app.get('/api/pedidos/fecha/:fecha', (req, res) => {
  const fecha = req.params.fecha; // Obtener la fecha del parámetro de la URL
  const query = `SELECT *
  FROM pedido
  WHERE fecha = $1
`;
  connection.query(query, [fecha], (err, results) => {
    if (err) return res.status(500).json({ error: err.message });
    res.json(results.rows); // Devolver los resultados de los pedidos
  });
});
```

54. Obtener los empleados por rol en un restaurante

```
// 5. Obtener los empleados por rol en un restaurante específico
app.get('/api/restaurante/:id/empleados/rol/:rol', (req, res) => {
  const idRest = req.params.id; // Obtener el id del restaurante del parámetro
  const rol = req.params.rol; // Obtener el rol del parámetro
  const query = `
    SELECT *
    FROM empleado
    WHERE id_rest = $1 AND rol = $2
  `;
  connection.query(query, [idRest, rol], (err, results) => {
    if (err) return res.status(500).json({ error: err.message });
    res.json(results.rows); // Devolver los empleados que coinciden con el rol y restaurante
  });
});
```