

Parcial 2

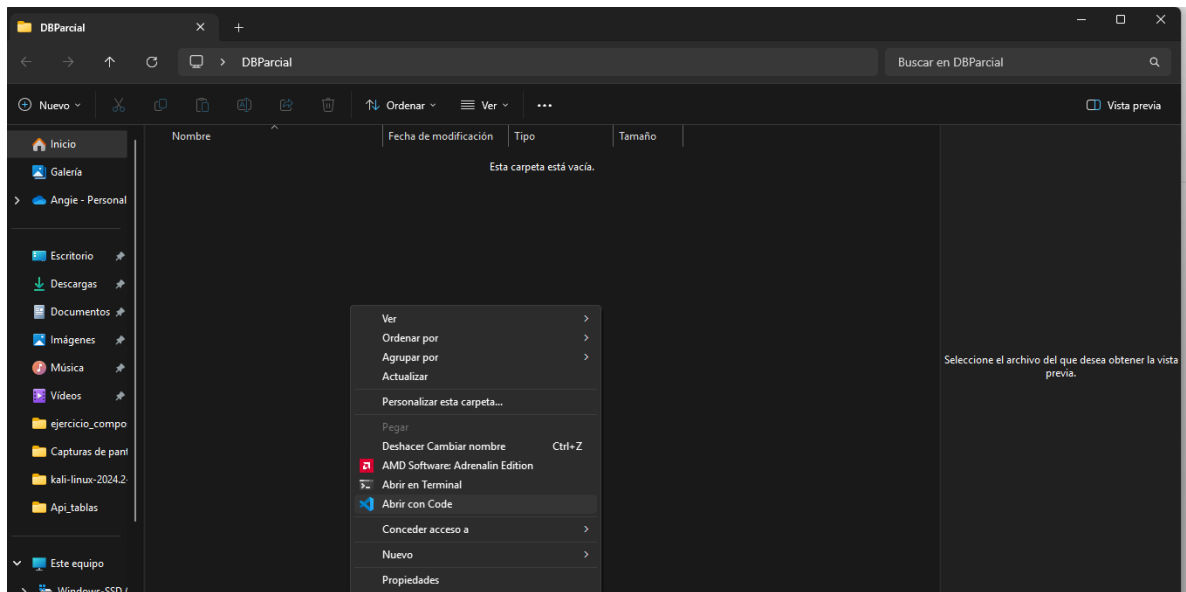
Angie Lorena Jiménez Porras

Ing. Sistemas, UNIMINUTO

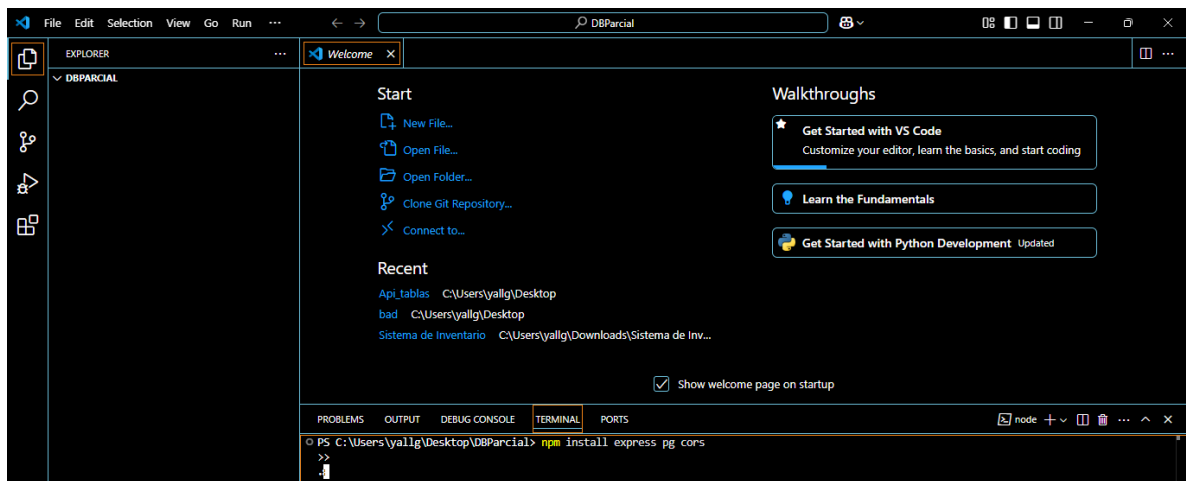
William Alexander Matallana Porras

Bases de datos masivas

25/04/2025



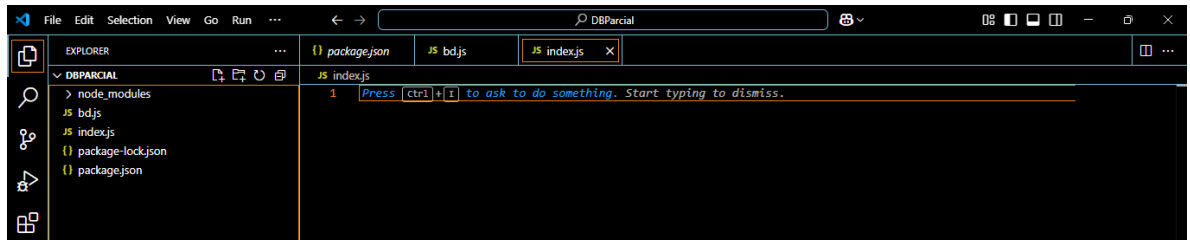
En una carpeta vacía abrimos con visual studio code



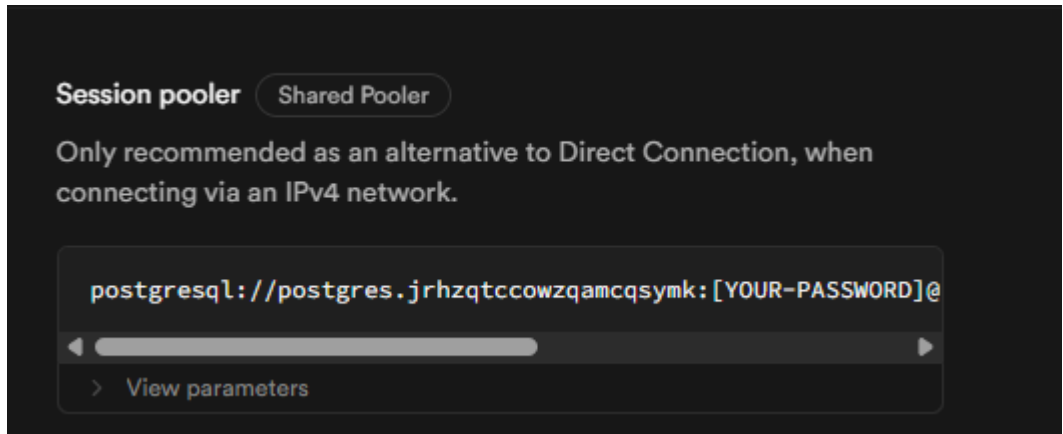
Una vez abierta la carpeta instalamos las extensiones que vayamos a usar.



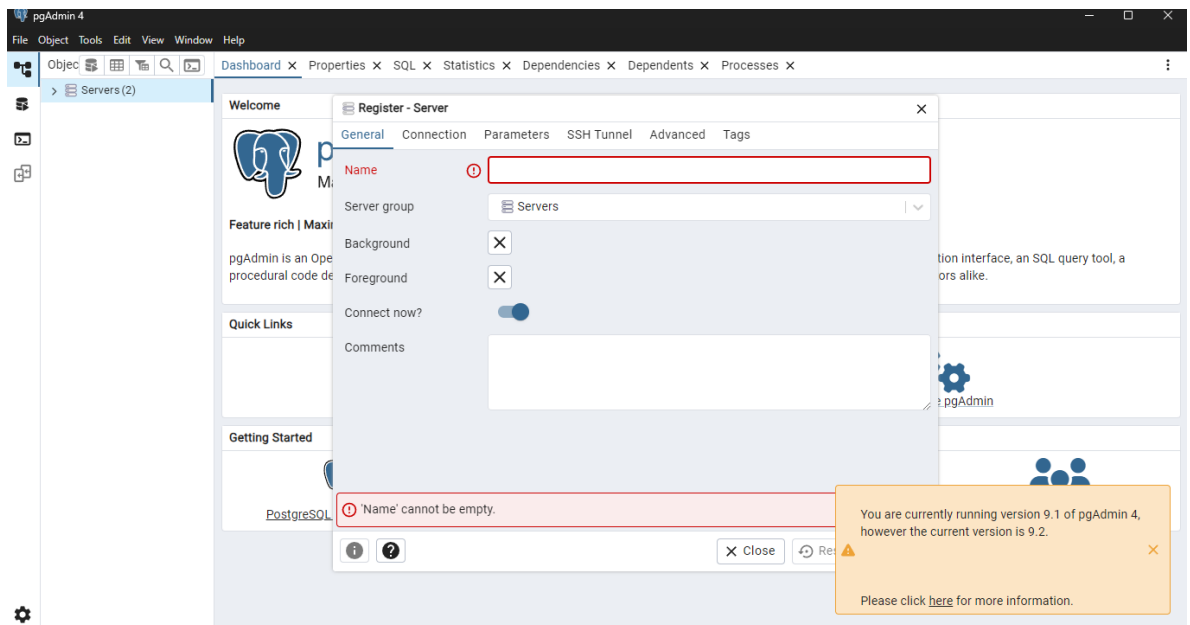
Ahora vamos a tener 2 archivos tipo json



Creamos 2 archivos .js que es bd.js e index.js: esto va a contener nuestra conexión a supabase



Para tener la conexión con Supabase vamos al Connect de un nuevo proyecto y copiaremos lo que nos arroja junto con la contraseña generada



Vamos a usar pg admin que será el cliente que usaremos para conectar a nuestro proyecto de supabase

Register - Server

General

Connection

Parameters

SSH Tunnel

Advanced

Tags

Host name/address

aws-0-us-east-2.pooler.supabase.com

Port

5432

Maintenance database

postgres

Username

postgres.jrhztccowzqamcqsymk

Kerberos authentication?

☐

Password

.....

Save password?

☒

Role

Service

i

?

Close

Reset

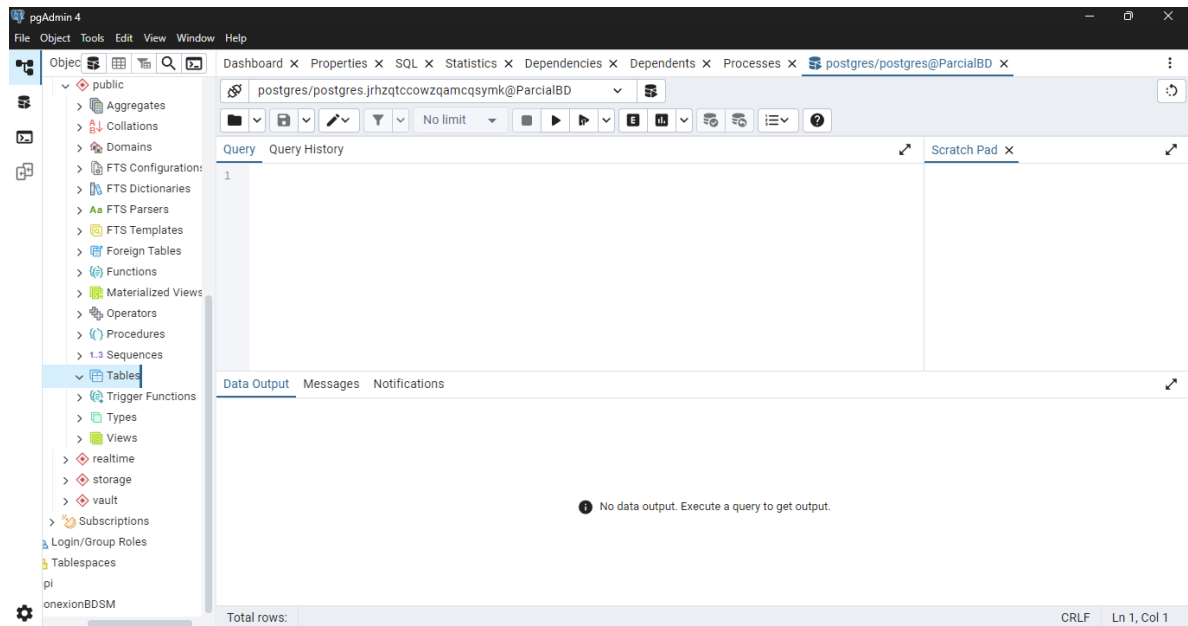
Save

Con los datos proporcionados en SupaBase hacemos la conexión

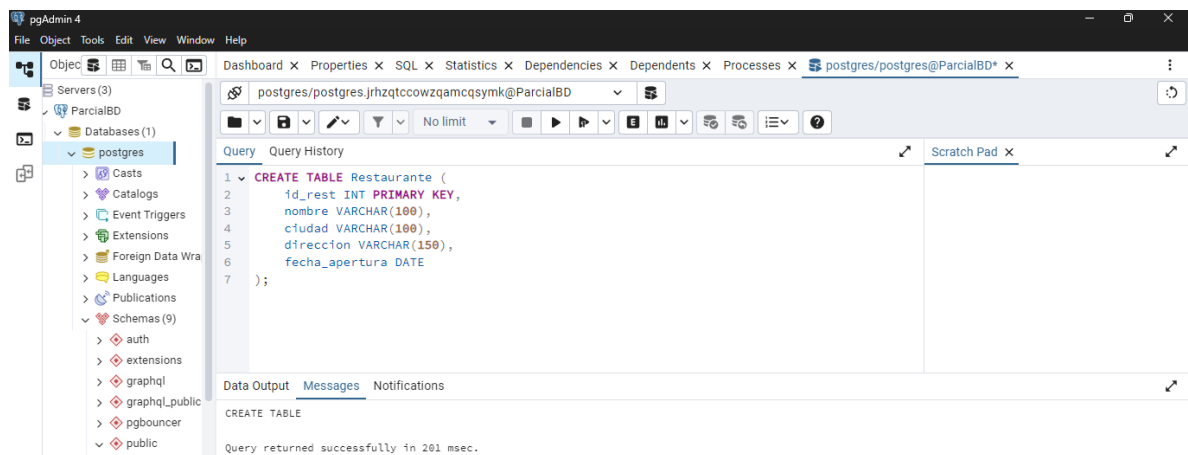
The screenshot shows the pgAdmin 4 interface with the 'Activity' tab selected. The left sidebar shows a tree view of the database structure, including 'Servers (3)', 'Databases (1)', and 'postgres'. The main panel displays several performance metrics and activity graphs:

- Database sessions:** A line graph showing 'Total', 'Active', and 'Idle' sessions over time. The y-axis ranges from 0 to 7.5.
- Transactions per second:** A line graph showing 'Transactions', 'Commits', and 'Rollbacks' over time. The y-axis ranges from 0 to 3.
- Tuples in:** A line graph showing 'Inserts', 'Updates', and 'Deletes' over time. The y-axis ranges from 0 to 100.
- Tuples out:** A line graph showing 'Fetched' and 'Returned' over time. The y-axis ranges from 0 to 200.
- Block I/O:** A line graph showing 'Reads' and 'Hits' over time. The y-axis ranges from 0 to 300.

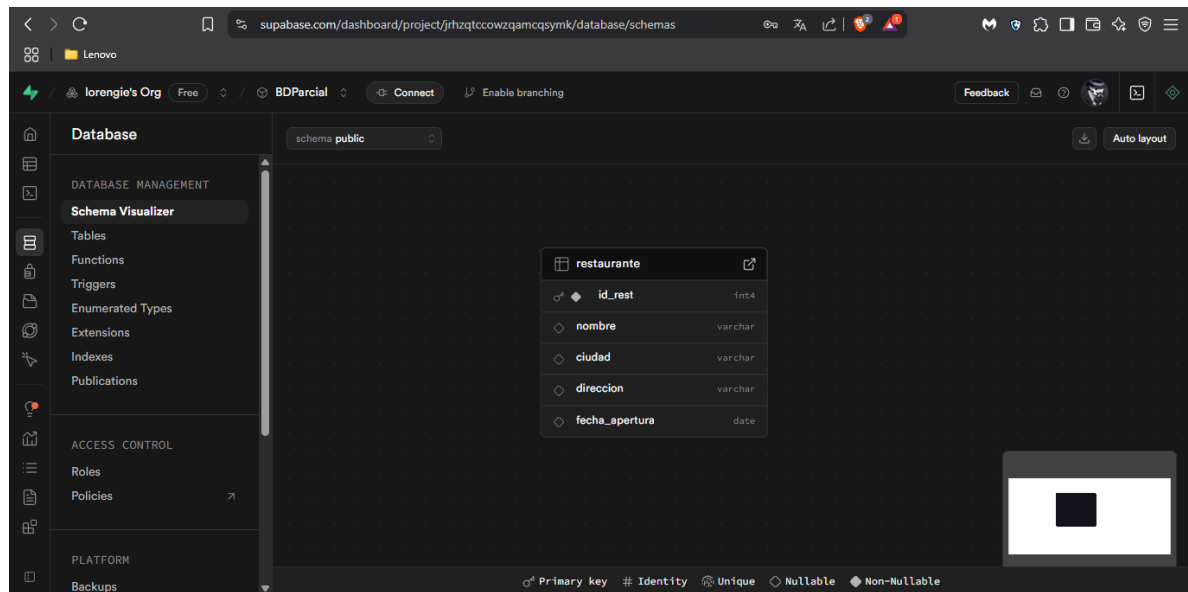
Una vez carga la conexión tendremos una base de datos llamada postgres, aquí van a aparecer todas las tablas que vamos a crear



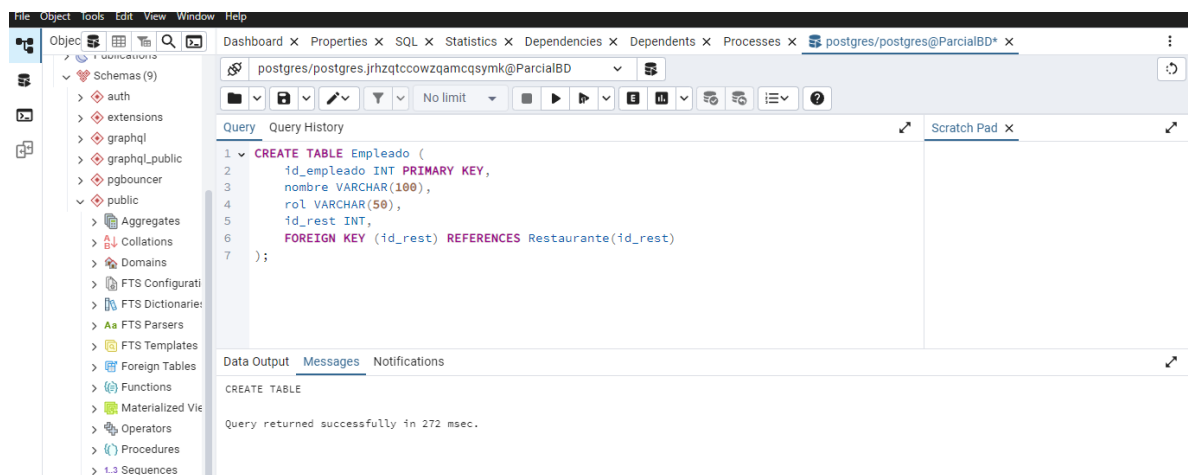
Ahora usaremos la herramienta de Query tools para generar nuestras tablas según los requerimientos dados



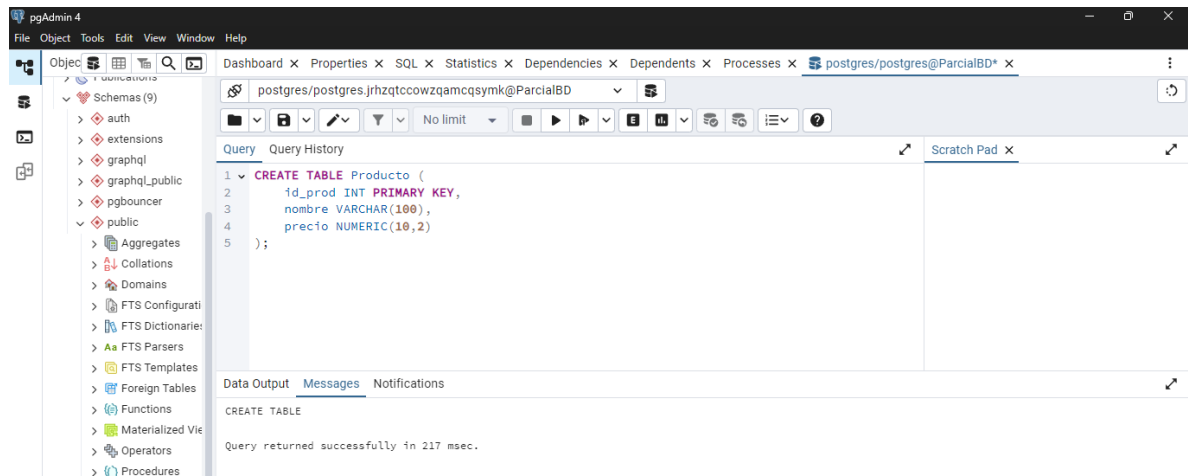
Iniciamos creando la tabla del restaurante según las llaves y campos requeridos



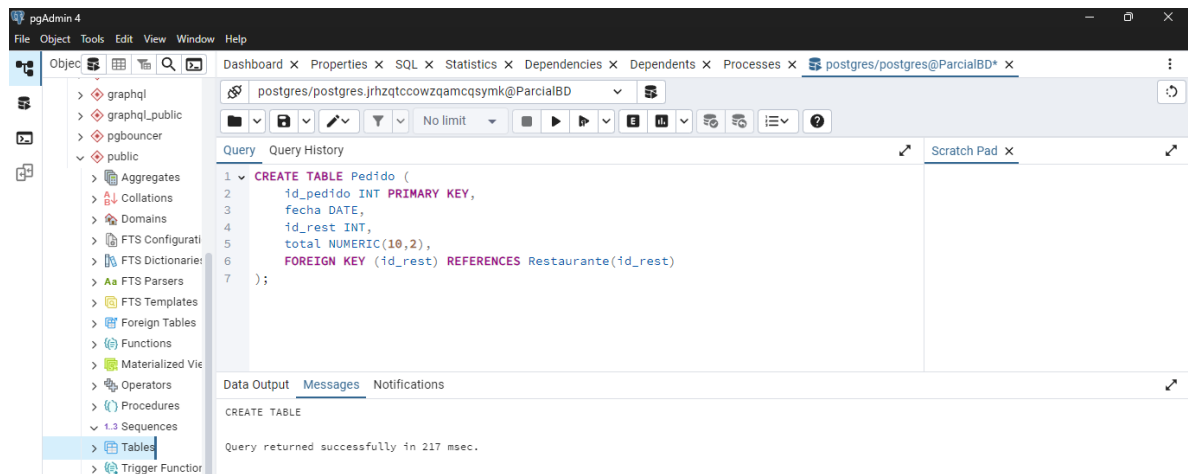
Con supabase revisamos que se haya generado la tabla esto con el
de verificar que la conexión haya quedado bien y para verificar que todo
haya quedado sincronizado



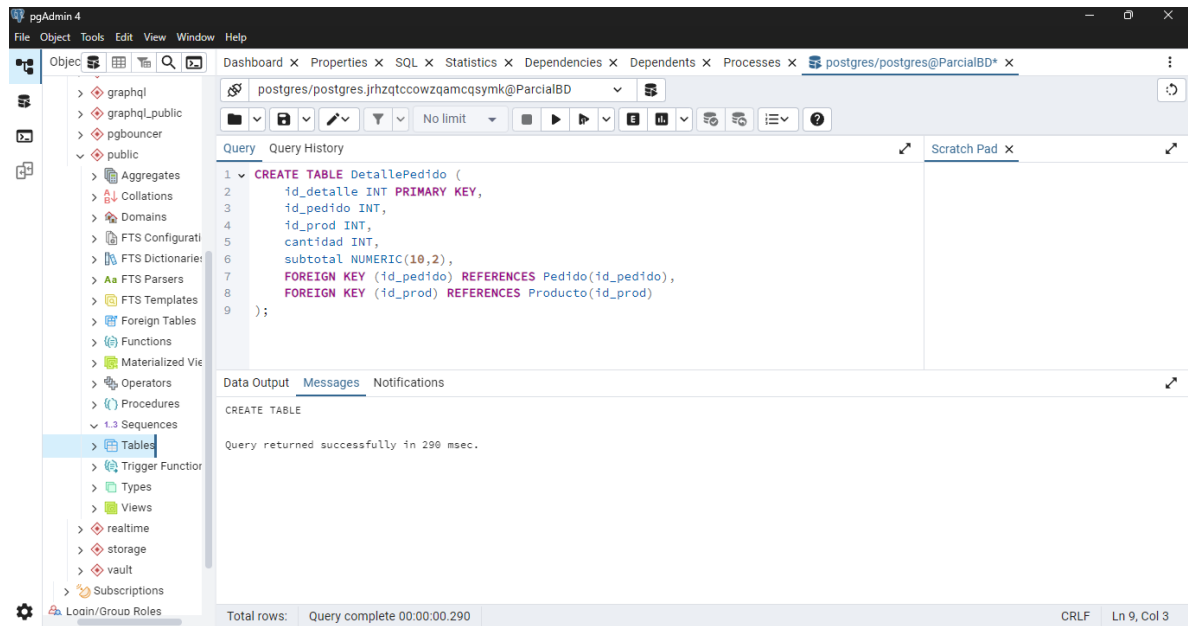
Como vimos que toda la conexión quedo bien hecha y responde
procedemos a crear la tabla de empleado



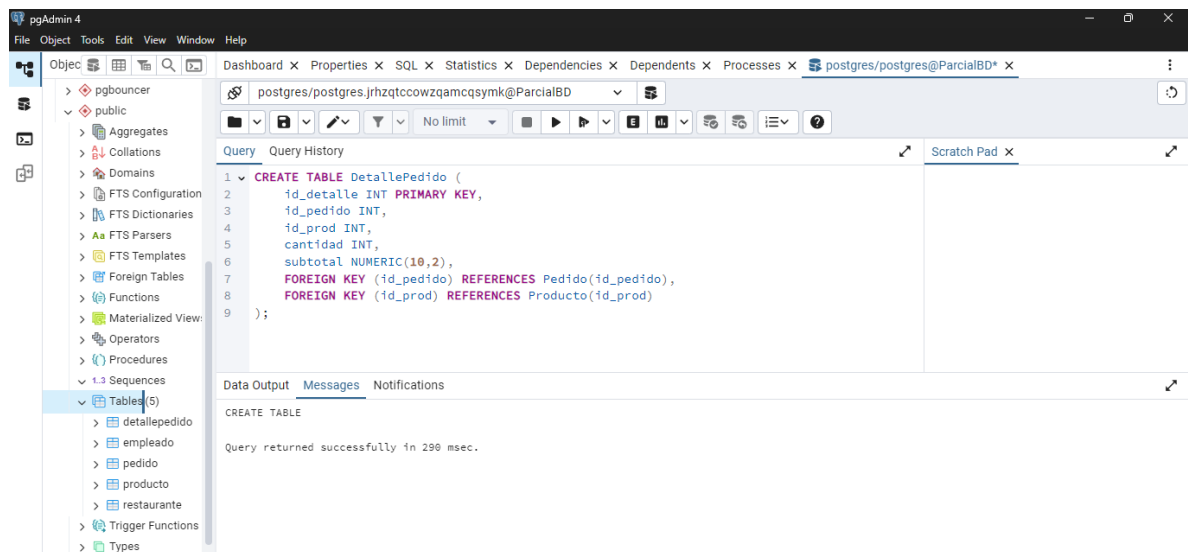
Continuamos generando la tabla de Producto según los campos
requeridos



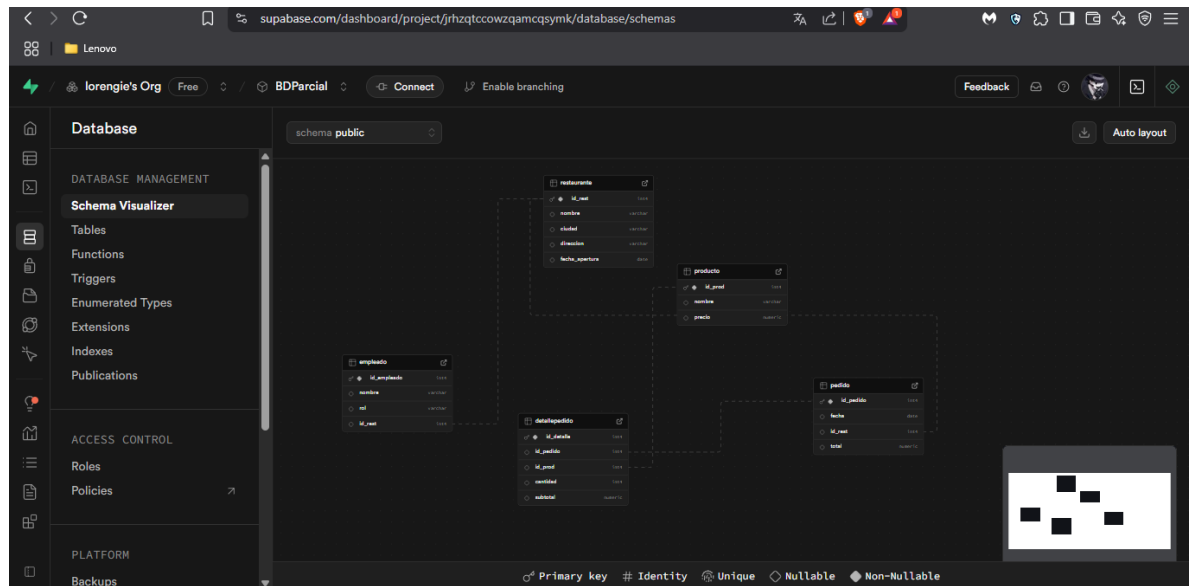
Creamos la tabla de Pedido



Finalmente creamos la tabla de DetallePedido según las llaves
foráneas, llave primaria y campos



Mediante pgadmin verificamos que todas las tablas se hayan creado
de manera correcta



Revisamos el esquema en supabase. Con esto finalizamos la creación de los modelos de la base de datos usando SupaBase y PgAdmin4

Creación de las API

```
package.json JS bd.js JS index.js M
JS bd.js > ...
1 C:\Users\yallg\Desktop\DBParcial\bd.js ('pg');
2
3 const client = new client({
4   host: 'aws-0-us-east-2.pooler.supabase.com',
5   port: 5432,
6   user: 'postgres.jrhztccowzqamcqsymk',
7   password: 'GtlV0zS7p5jLicCF',
8   database: 'postgres'
9 });
10
11 client.connect()
12   .then(() => console.log('Conectado a Supabase PostgreSQL'))
13   .catch(err => console.error('Error conectando a Supabase:', err.stack));
14
15 module.exports = client;
```

Iniciamos creando la conexión con supabase en el db.js

```
{ } package.json JS bd.js JS index.js M X
JS index.js > ...
1  const express = require('express');
2  const { Pool } = require('pg');
3  const app = express();
4
5  app.use(express.json());
6
7  const pool = new Pool({
8    user: 'postgres.jrhztccowzqamcqsymk',
9    host: 'aws-0-us-east-2.pooler.supabase.com',
10   database: 'postgres',
11   password: 'GTlV0zS7p5jLicCF',
12   port: 5432,
13 });
```

En el index vamos a poner la conexión, el puerto y los datos que generó supabase

API CRUD

```
10 app.post('/api/restaurantes', (req, res) => {
11   const data = req.body;
12   res.json({ message: 'Restaurante creado', data });
13 });
14
15 app.get('/api/restaurantes', async (req, res) => {
16   try {
17     const result = await pool.query('SELECT * FROM restaurante');
18     res.json(result.rows);
19   } catch (error) {
20     console.error('Error al obtener restaurantes:', error.message);
21     res.status(500).json({ error: 'Error al obtener restaurantes' });
22   }
23 });
```

Para hacer el crud CREAR: de restaurantes se inicia con un app.post para usarlo posteriormente en postman, va a requerir los datos de los campos de la tabla que hayamos configurado

Crud LEER: usamos .get, el cual va a hacer una consulta según los datos de la tabla y va a seleccionar todos los datos, los visualizaremos con el GET de postman

```

25 ✓ app.get('/api/restaurantes/:id', async (req, res) => {
26   const id = req.params.id;
27   try {
28     const result = await pool.query('SELECT * FROM restaurante WHERE id_restaurante = $1', [id]);
29
30     if (result.rows.length === 0) {
31       return res.status(404).json({ message: 'Restaurante no encontrado' });
32     }
33
34     res.json(result.rows[0]);
35   } catch (error) {
36     console.error('Error al obtener restaurante:', error.message);
37     res.status(500).json({ error: 'Error al obtener restaurante' });
38   }
39 });

```

LEER según el id del restaurante, usamos :id para en postman
daremos el id del que queremos consultar

```

41 app.put('/api/restaurantes/:id', (req, res) => {
42   const id = req.params.id;
43   const data = req.body;
44   res.json({ message: `Restaurante ${id} actualizado`, data });
45 });
46 app.delete('/api/restaurantes/:id', (req, res) => {
47   const id = req.params.id;
48   res.json({ message: `Restaurante ${id} eliminado` });
49 });

```

ACTUALIZAR: usamos .put y el id de lo que vayamos a utilizar
BORRAR: usamos .delete para eliminar según el id que pondremos
en postman

CRUD empleados

```

52 app.post('/api/empleados', (req, res) => {
53   const data = req.body;
54   res.json({ message: 'Empleado creado', data });
55 });
56
57 app.get('/api/empleados', async (req, res) => {
58   try {
59     const result = await pool.query('SELECT * FROM empleado');
60     res.json(result.rows);
61   } catch (error) {
62     console.error('Error al obtener empleados:', error.message);
63     res.status(500).json({ error: 'Error al obtener empleados' });
64   }
65 });

```

```

66
67 app.get('/api/empleados/:id', async (req, res) => {
68   const id = req.params.id;
69   try {
70     const result = await pool.query('SELECT * FROM empleado WHERE id_empleado = $1', [id]);
71
72     if (result.rows.length === 0) {
73       return res.status(404).json({ message: 'Empleado no encontrado' });
74     }
75
76     res.json(result.rows[0]);
77   } catch (error) {
78     console.error('Error al obtener empleado:', error.message);
79     res.status(500).json({ error: 'Error al obtener empleado' });
80   }
81 });

```

```

83 ✓ app.put('/api/empleados/:id', (req, res) => {
84   const id = req.params.id;
85   const data = req.body;
86   res.json({ message: `Empleado ${id} actualizado`, data });
87 });
88
89 ✓ app.delete('/api/empleados/:id', (req, res) => {
90   const id = req.params.id;
91   res.json({ message: `Empleado ${id} eliminado` });
92 });
93

```

CRUD Productos

```

95  ✓ app.post('/api/productos', (req, res) => {
96      const data = req.body;
97      res.json({ message: 'Producto creado', data });
98  });
99
100  ✓ app.get('/api/productos', async (req, res) => {
101      ✓ try {
102          const result = await pool.query('SELECT * FROM producto');
103          res.json(result.rows);
104      } catch (error) {
105          console.error('Error al obtener productos:', error.message);
106          res.status(500).json({ error: 'Error al obtener productos' });
107      }
108  });

```

```

110  ✓ app.get('/api/productos/:id', async (req, res) => {
111      const id = req.params.id;
112      ✓ try {
113          const result = await pool.query('SELECT * FROM producto WHERE id_producto = $1', [id]);
114
115          ✓ if (result.rows.length === 0) {
116              return res.status(404).json({ message: 'Producto no encontrado' });
117          }
118
119          res.json(result.rows[0]);
120      } catch (error) {
121          console.error('Error al obtener producto:', error.message);
122          res.status(500).json({ error: 'Error al obtener producto' });
123      }
124  });

```

```

125
126  app.put('/api/productos/:id', (req, res) => {
127      const id = req.params.id;
128      const data = req.body;
129      res.json({ message: `Producto ${id} actualizado`, data });
130  });
131
132  app.delete('/api/productos/:id', (req, res) => {
133      const id = req.params.id;
134      res.json({ message: `Producto ${id} eliminado` });
135  });

```

CRUD pedido

```

138 app.post('/api/pedidos', (req, res) => {
139   const data = req.body;
140   res.json({ message: 'Pedido creado', data });
141 });
142
143 app.get('/api/pedidos', async (req, res) => {
144   try {
145     const result = await pool.query('SELECT * FROM pedido');
146     res.json(result.rows);
147   } catch (error) {
148     console.error('Error al obtener pedidos:', error.message);
149     res.status(500).json({ error: 'Error al obtener pedidos' });
150   }
151 });

```

```

152
153 app.get('/api/pedidos/:id', async (req, res) => {
154   const id = req.params.id;
155   try {
156     const result = await pool.query('SELECT * FROM pedido WHERE id_pedido = $1', [id]);
157
158     if (result.rows.length === 0) {
159       return res.status(404).json({ message: 'Pedido no encontrado' });
160     }
161
162     res.json(result.rows[0]);
163   } catch (error) {
164     console.error('Error al obtener pedido:', error.message);
165     res.status(500).json({ error: 'Error al obtener pedido' });
166   }
167 });

```

```

169
170 app.put('/api/pedidos/:id', (req, res) => {
171   const id = req.params.id;
172   const data = req.body;
173   res.json({ message: `Pedido ${id} actualizado`, data });
174 });
175
176 app.delete('/api/pedidos/:id', (req, res) => {
177   const id = req.params.id;
178   res.json({ message: `Pedido ${id} eliminado` });
179 });

```

CRUD detallepedido

```
182 app.post('/api/detallepedidos', (req, res) => {
183   const data = req.body;
184   res.json({ message: 'Detalle de pedido creado', data });
185 });
186
187 app.get('/api/detallepedidos', async (req, res) => {
188   try {
189     const result = await pool.query('SELECT * FROM detallepedido');
190     res.json(result.rows);
191   } catch (error) {
192     console.error('Error al obtener detallepedidos:', error.message);
193     res.status(500).json({ error: 'Error al obtener detallepedidos' });
194   }
195 });
```

```
197 app.get('/api/detallepedidos/:id', async (req, res) => {
198   const id = req.params.id;
199   try {
200     const result = await pool.query('SELECT * FROM detallepedido WHERE id_detalle = $1', [id]);
201
202     if (result.rows.length === 0) {
203       return res.status(404).json({ message: 'DetallePedido no encontrado' });
204     }
205
206     res.json(result.rows[0]);
207   } catch (error) {
208     console.error('Error al obtener detallepedido:', error.message);
209     res.status(500).json({ error: 'Error al obtener detallepedido' });
210   }
211 });
```

```
212
213 app.put('/api/detallepedidos/:id', (req, res) => {
214   const id = req.params.id;
215   const data = req.body;
216   res.json({ message: `DetallePedido ${id} actualizado`, data });
217 });
218
219 app.delete('/api/detallepedidos/:id', (req, res) => {
220   const id = req.params.id;
221   res.json({ message: `DetallePedido ${id} eliminado` });
222 });
```

Consultas nativas

1 Productos de un pedido específico

```
227 app.get('/consultas/productos-pedido/:id_pedido', async (req, res) => {
228   const { id_pedido } = req.params;
229   const result = await pool.query(`
230     SELECT p.nombre, dp.cantidad, dp.subtotal
231     FROM DetallePedido dp
232     JOIN Producto p ON dp.id_prod = p.id_prod
233     WHERE dp.id_pedido = $1
234   `, [id_pedido]);
235   res.json(result.rows);
236 });
```

Este api pedirá el id dentro de un pedido para mostrar la información; con req.params se extraerá el id desde la url; hace una unión entre la tabla de producto y detallepedido usando un join donde solo se muestre el id del pedido que tenga ese id haciendo un filtro de búsqueda según el id del pedido. Con res.json se devolverá un array tipo json con los detalles de la consulta

2 Productos más vendidos

```
239 app.get('/consultas/productos-mas-vendidos/:cantidad_min', async (req, res) => {
240   const { cantidad_min } = req.params;
241   const result = await pool.query(`
242     SELECT p.nombre, SUM(dp.cantidad) AS total_vendido
243     FROM DetallePedido dp
244     JOIN Producto p ON dp.id_prod = p.id_prod
245     GROUP BY p.nombre
246     HAVING SUM(dp.cantidad) > $1
247   `, [cantidad_min]);
248   res.json(result.rows);
249 });
```

Para hacer este api se hace una búsqueda según un filtro de ventas que solo mostrará los productos con esa mínima cantidad de ventas donde: join junta detalle pedido con producto, agrupa según el nombre, con sum se sumá las unidades vendidas, having filtra solo los que sumen el total lo

mínimo que se pasó como parámetro para la consulta. Finalmente mostrara en un formato json los resultados

3Total de ventas por restaurante

```
252 app.get('/consultas/ventas-por-restaurante', async (req, res) => {
253   const result = await pool.query(`
254     SELECT r.nombre, SUM(p.total) AS total_ventas
255     FROM Pedido p
256     JOIN Restaurante r ON p.id_rest = r.id_rest
257     GROUP BY r.nombre
258   `);
259   res.json(result.rows);
260 });
```

No se reciben parámetros específicos, se hace mediante la unión de pedido con restaurante usando con join y agrupando según el nombre

4Pedidos por fecha

```
263 app.get('/consultas/pedidos-por-fecha/:fecha', async (req, res) => {
264   const { fecha } = req.params;
265   const result = await pool.query(`SELECT * FROM Pedido WHERE fecha = $1`, [fecha]);
266   res.json(result.rows);
267 });
268
```

Se recibe un dato de tipo fecha para hacer la consulta, se mostrará solo los que tengan la fecha que se pasó y coincida con el valor, en caso de no tener ninguno de esa fecha se devolverá un json en blanco

5Empleados por rol en un restaurante

```
270 app.get('/consultas/empleados-por-rol/:id_rest/:rol', async (req, res) => {
271   const { id_rest, rol } = req.params;
272   const result = await pool.query(`
273     SELECT nombre FROM Empleado
274     WHERE id_rest = $1 AND rol = $2
275   `, [id_rest, rol]);
276   res.json(result.rows);
277 });
```

Esta api pedirá el id del empleado y el rol dado, hará una consulta en la tabla empleado donde el id y el rol sean los que se pasó

Prueba con las API creadas usando Postman

CRUD Productos

REST API basics: CRUD, test & variable / CRUD producto / GET productos

GET <http://localhost:3000/api/productos> [Send](#)

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (7) Test Results [200 OK](#) • 252 ms • 6.32 KB [Save Response](#)

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

```
1 [
2   {
3     "id_prod": 1,
4     "nombre": "Facilis especial",
5     "precio": "12776.71"
6   },
7   {
8     "id_prod": 2,
9     "nombre": "Quam especial",
10    "precio": "15997.84"
11  },
12  {
13    "id_prod": 3,
14    "nombre": "Nisi especial",
15    "precio": "14963.24"
16  }
17 ]
```

Postbot Runner Start Proxy Cookies Vault Trash

REST API basics: CRUD, test & variable / CRUD producto / POST producto

POST <http://localhost:3000/api/productos> [Send](#)

Params Authorization Headers (9) Body Scripts Settings Cookies

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

```
1 {
2   "nombre": "Pizza Margarita especial",
3   "precio": 15000.50
4 }
```

Body Cookies Headers (7) Test Results [200 OK](#) • 127 ms • 326 B [Save Response](#)

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

```
1 {
2   "message": "Producto creado",
3   "data": {
4     "nombre": "Pizza Margarita especial",
5     "precio": 15000.5
6   }
7 }
```

REST API basics: CRUD, test & variable / CRUD producto / **PUT producto**

PUT **Send**

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

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON **Beautify**

```
1 {
2   "nombre": "Pizza Margarita especial",
3   "precio": 15000.50
4 }
```

Body Cookies Headers (7) Test Results **200 OK** · 31 ms · 333 B Save Response

JSON Preview Visualize

```
1 {
2   "message": "Producto 7 actualizado",
3   "data": {
4     "nombre": "Pizza Margarita especial",
5     "precio": 15000.5
6   }
7 }
```

REST API basics: CRUD, test & variable / CRUD producto / **DELETE producto**

DELETE **Send**

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

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (7) Test Results **200 OK** · 13 ms · 269 B Save Response

JSON Preview Visualize

```
1 {
2   "message": "Producto 3 eliminado"
3 }
```

CRUD empleados

REST API basics: CRUD, test & variable / CRUD-empleados / GET empleados

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

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (7) Test Results 200 OK 128 ms 8.14 KB Save Response

{ } JSON Preview Visualize

```
1  [
2    {
3      "id_empleado": 1,
4      "nombre": "Victorino Dueñas Barón",
5      "rol": "cocinero",
6      "id_rest": 65
7    },
8    {
9      "id_empleado": 2,
10     "nombre": "Ernesto Calvet Benet",
11     "rol": "gerente",
12     "id_rest": 65
13   },
14   {
15     "id_empleado": 3,
```

REST API basics: CRUD, test & variable / CRUD-empleados / POST empleados

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

Params Authorization Headers (9) Body Scripts Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Beautify

```
1  {
2    "id_empleado": 101,
3    "nombre": "Juan Pérez",
4    "rol": "Cocinero"
```

Body Cookies Headers (7) Test Results 200 OK 83 ms 344 B Save Response

{ } JSON Preview Visualize

```
1  {
2    "message": "Empleado creado",
3    "data": {
4      "id_empleado": 101,
5      "nombre": "Juan Pérez",
6      "rol": "Cocinero",
7      "id_rest": 1
8    }
9  }
```

REST API basics: CRUD, test & variable / CRUD empleados / **PUT empleado**

PUT ▼ http://localhost:3000/api/empleados/4 Send ▼

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

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▼ Beautify

```
2  "nombre": "Juan Pérez Actualizado",
3  "rol": "Chef",
4  "id_rest": 1
```

Body Cookies Headers (7) Test Results ↺

{} JSON ▼ ▶ Preview 🔗 Visualize ▼ 🔍 📄 🔗

200 OK • 39 ms • 341 B • 🌐 📄 Save Response ⋮

```
1  {
2    "message": "Empleado 4 actualizado",
3    "data": {
4      "nombre": "Juan Pérez Actualizado",
5      "rol": "Chef",
6      "id_rest": 1
7    }
8  }
```

REST API basics: CRUD, test & variable / CRUD empleados / **DELETE empleado**

DELETE ▼ http://localhost:3000/api/empleados/4 Send ▼

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

Query Params

	Key	Value	Description	⋮	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (7) Test Results ↺

{} JSON ▼ ▶ Preview 🔗 Visualize ▼ 🔍 📄 🔗

200 OK • 16 ms • 269 B • 🌐 📄 Save Response ⋮

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

CRUD restaurantes

REST API basics: CRUD, test & variable / restaurante / **POST restaurante**

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

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

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Beautify

```
1 {
2   "id_rest": 6666,
3   "nombre": "Restaurante El Buen Sabor",
4   "ciudad": "Bogotá"
```

Body Cookies Headers (7) Test Results 200 OK • 14 ms • 409 B Save Response

JSON Preview Visualize

```
1 {
2   "message": "Restaurante creado",
3   "data": {
4     "id_rest": 6666,
5     "nombre": "Restaurante El Buen Sabor",
6     "ciudad": "Bogotá",
7     "direccion": "Calle 123 #45-67",
8     "fecha_apertura": "2023-01-15"
9   }
10 }
```

REST API basics: CRUD, test & variable / restaurante / **GET restaurante**

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

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

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

This request does not have a body

Body Cookies Headers (7) Test Results 200 OK • 145 ms • 17.19 KB Save Response

JSON Preview Visualize

```
1 [
2   {
3     "id_rest": 1,
4     "nombre": "Pereira-Aguiló",
5     "ciudad": "Burgos",
6     "direccion": "Callejón de Pía Rubio 59 Puerta 6 \nVizcaya, 02111",
7     "fecha_apertura": "2023-01-11T05:00:00.000Z"
8   },
9   {
10    "id_rest": 2,
11    "nombre": "Bueno-Borja",
12    "ciudad": "Melilla",
13    "direccion": "Pasadizo de Águeda Barba 12 Puerta 7 \nAlmería, 93997",
14    "fecha_apertura": "2018-11-11T05:00:00.000Z"
15  },
16 ]
```

REST API basics: CRUD, test & variable / restaurante / **PUT restaurante** Save Share

PUT Send ▼ http://localhost:3000/api/restaurantes/1

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

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```
1 {
2   "nombre": "Big Threat Soul",
3   "ciudad": "Medellin",
4   "direccion": "Carrera 80 #10-45"
```

Body Cookies Headers (7) Test Results 🕒 200 OK 7 ms 395 B 🌐 📄 Save Response ⋮

{} **JSON** ▼ ▶ Preview 🔍 Visualize ▼ 🔍 📄 🔗

```
1 {
2   "message": "Restaurante 1 actualizado",
3   "data": {
4     "nombre": "Big Threat Soul",
5     "ciudad": "Medellin",
6     "direccion": "Carrera 80 #10-45",
7     "fecha_apertura": "2023-06-15"
8   }
9 }
```

REST API basics: CRUD, test & variable / restaurante / **DELETE restaurante** Save Share

DELETE Send ▼ http://localhost:3000/api/restaurantes/3

Params Authorization Headers (7) **Body** Scripts Settings Cookies </>

Query Params

	Key	Value	Description	⋮ Bulk Edit
	Key	Value	Description	

Body Cookies Headers (7) Test Results 🕒 200 OK 18 ms 272 B 🌐 📄 Save Response ⋮

{} **JSON** ▼ ▶ Preview 🔍 Visualize ▼ 🔍 📄 🔗

```
1 {
2   "message": "Restaurante 3 eliminado"
3 }
```

CRUD pedidos

REST API basics: CRUD, test & variable / CRUD pedidos / GET pedidos

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

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (7) Test Results 200 OK • 259 ms • 8.58 KB Save Response

{ } JSON Preview Visualize

```
1 [
2   {
3     "id_pedido": 1,
4     "fecha": "2024-11-01T05:00:00.000Z",
5     "id_rest": 97,
6     "total": "272788.43"
7   },
8   {
9     "id_pedido": 2,
10    "fecha": "2023-07-26T05:00:00.000Z",
11    "id_rest": 19,
12    "total": "116509.40"
13  },
14  {
15    "id_pedido": 3,
```

REST API basics: CRUD, test & variable / CRUD pedidos / POST pedidos

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

Params Authorization Headers (9) Body Scripts Settings Cookies Beautify

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {
2   "fecha": "2024-04-20",
3   "id_rest": 10,
4   "total": 48500.00
```

Body Cookies Headers (7) Test Results 200 OK • 7 ms • 319 B Save Response

{ } JSON Preview Visualize

```
1 {
2   "message": "Pedido creado",
3   "data": {
4     "fecha": "2024-04-20",
5     "id_rest": 10,
6     "total": 48500
7   }
8 }
```


REST API basics: CRUD, test & variable / CRUD pedidos / **PUT pedidos**

PUT http://localhost:3000/api/pedidos/66

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

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON

```
1 {
2   "fecha": "2024-04-20",
3   "id_rest": 10,
4   "total": 48500.00
5 }
```

Body Cookies Headers (7) Test Results 200 OK • 13 ms • 327 B Save Response

Preview Visualize

```
1 {
2   "message": "Pedido 66 actualizado",
3   "data": {
4     "fecha": "2024-04-20",
5     "id_rest": 10,
6     "total": 48500
7   }
8 }
```

REST API basics: CRUD, test & variable / CRUD pedidos / **DELETE pedido**

DELETE http://localhost:3000/api/pedidos/77

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

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (7) Test Results 200 OK • 9 ms • 268 B Save Response

Preview Visualize

```
1 {
2   "message": "Pedido 77 eliminado"
3 }
```

CRUD detallepedido

REST API basics: CRUD, test & variable / CRUD detalles-pedido / **GET detallePedido** Save Share

GET ▼ Send ▼

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (7) Test Results ↺ 200 OK · 142 ms · 25.83 KB · 🌐 📄 Save Response ⋮

{} JSON ▼ ▶ Preview 🔗 Visualize ▼ 🔍 📄 🔗

```
1  [
2    {
3      "id_detalle": 1,
4      "id_pedido": 1,
5      "id_prod": 25,
6      "cantidad": 3,
7      "subtotal": "80221.44"
8    },
9    {
10     "id_detalle": 2
```

REST API basics: CRUD, test & variable / CRUD detalles-pedido / **POST detalle-pedido** Save Share 📄

POST ▼ Send ▼ 💬

Params Authorization Headers (9) **Body** ● Scripts Settings Cookies </>

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▼ Beautify 🔗

```
1  {
2    "id_pedido": 1001,
3    "fecha": "2023-10-25",
4    "id_rest": 1
```

Body Cookies Headers (7) Test Results ↺ 200 OK · 28 ms · 347 B · 🌐 📄 Save Response ⋮

{} JSON ▼ ▶ Preview 🔗 Visualize ▼ 🔍 📄 🔗

```
1  {
2    "message": "Detalle de pedido creado",
3    "data": {
4      "id_pedido": 1001,
5      "fecha": "2023-10-25",
6      "id_rest": 1,
7      "total": 29.97
8    }
9  }
```

REST API basics: CRUD, test & variable / CRUD detalles-pedido / **PUT detalle-pedido**

PUT http://localhost:3000/api/detallepedidos/5 Send

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

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Beautify

```
1 {
2   "id_pedido": 1001,
3   "id_prod": 501,
4   "cantidad": 2
5 }
```

Body Cookies Headers (7) Test Results 200 OK 19 ms 347 B Save Response

{ JSON Preview Visualize

```
1 {
2   "message": "DetallePedido 5 actualizado",
3   "data": {
4     "id_pedido": 1001,
5     "id_prod": 501,
6     "cantidad": 2,
7     "subtotal": 19.98
8   }
9 }
```

REST API basics: CRUD, test & variable / CRUD detalles-pedido / **DELETE detalle-pedido**

DELETE http://localhost:3000/api/detallepedidos/66 Send

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

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (7) Test Results 200 OK 35 ms 275 B Save Response

{ JSON Preview Visualize

```
1 {
2   "message": "DetallePedido 66 eliminado"
3 }
```

CONSULTAS NATIVAS

1 Productos de un pedido específico

REST API basics: CRUD, test & variable / Consultas nativas / **productos de un pedido específico**

GET Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (7) Test Results 200 OK • 191 ms • 439 B Save Response

{ } JSON Preview Visualize

```
1  [
2    {
3      "nombre": "Doloremque especial",
4      "cantidad": 3,
5      "subtotal": "88221.44"
6    },
7    {
8      "nombre": "Voluptas especial",
9      "cantidad": 2,
10     "subtotal": "48278.34"
11   },
12   {
13     "nombre": "Voluptate especial",
14     "cantidad": 5,
15     "subtotal": "144296.65"
```

2 Productos más vendidos

REST API basics: CRUD, test & variable / Consultas nativas / **Productos más vendidos**

GET Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

	Key	Value	Description	...	Bulk Edit
	Key	Value	Description		

Body Cookies Headers (7) Test Results 200 OK • 210 ms • 3.38 KB Save Response

{ } JSON Preview Visualize

```
1  [
2    {
3      "nombre": "Quasi especial",
4      "total_vendido": "10"
5    },
6    {
7      "nombre": "Quos especial",
8      "total_vendido": "8"
9    },
10   {
11     "nombre": "Occaecati especial",
12     "total_vendido": "18"
13   },
14   {
15     "nombre": "Quisquam especial",
```

3 Total de ventas por restaurante

REST API basics: CRUD, test & variable / Consultas nativas / Ventas por restaurante

GET http://localhost:3000/consultas/ventas-por-restaurante Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (7) Test Results 200 OK 167 ms 3.8 KB Save Response

JSON Preview Visualize

```
1 [
2   {
3     "nombre": "Checa, León and Verdejo",
4     "total_ventas": "336982.61"
5   },
6   {
7     "nombre": "Linares LLC",
8     "total_ventas": "188603.94"
9   },
10  {
11    "nombre": "Pereira-Aguiló",
12    "total_ventas": "191491.76"
13  },
14  {
15    "nombre": "Ureña Group",
```

4Pedidos por fecha

API_PARCIAL2 / Consultas nativas / pedidos por fecha

GET http://localhost:3000/consultas/pedidos-por-fecha/2024-11-01 Send

Params Authorization Headers (7) Body Scripts Settings Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (7) Test Results 200 OK 157 ms 320 B Save Response

JSON Preview Visualize

```
1 [
2   {
3     "id_pedido": 1,
4     "fecha": "2024-11-01T05:00:00.000Z",
5     "id_rest": 97,
6     "total": "272788.43"
7   }
8 ]
```

5Empleados por rol en un restaurante

REST API basics: CRUD, test & variable / Consultas nativas / Empleados por rol en un restaurante

SaveShare

GET

http://localhost:3000/consultas/empleados-por-rol/1/cocinero

Send

Params

Authorization

Headers (7)

Body

Scripts

Settings

Cookies

Query Params

	Key	Value	Description	⋮ Bulk Edit
	Key	Value	Description	

Body

Cookies

Headers (7)

Test Results

🔄

200 OK

195 ms

277 B

🌐

📄 Save Response

⋮

{ } JSON

▶ Preview

🔗 Visualize

⌵

🔍

📄

🔗

```
1  [
2    {
3      "nombre": "Eva Maria Villena Carvajal"
4    }
5  ]
```