

# **DOCUMENTACION PARCIAL**

MIGUEL ANGEL JIMENEZ PORRAS

ID 834889

BASES DE DATOS MASIVAS

NRC:60747

WILLIAM ALEXANDER MATALLANA PORRAS

BASES DE DATOS MASIVAS

5/04/2025

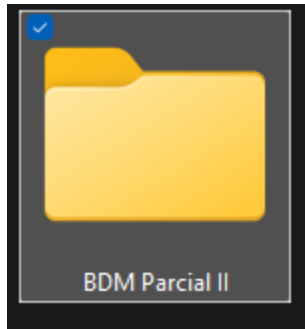
CORPORACIÓN UNIVERSITARIA MINUTO DE DIOS UNIMINUTO

## Contenido

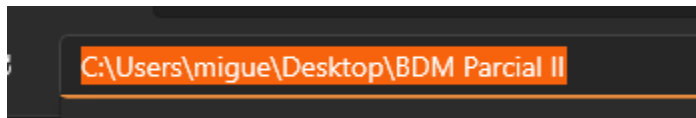
Creación de la carpeta del proyecto.....	3
Creación de base de datos en Supabase .....	4
Asociación de la base de datos con Pgadmin4.....	5
Creación de las tablas con Pgadmin4.....	8
Creación de tabla Restaurante .....	9
Creación de tabla Empleado .....	9
Creación de tabla Producto .....	10
Creación de tabla Pedido .....	10
Creación de tabla DetallePedido .....	11

## Creación de la carpeta del proyecto

Para crear la carpeta del proyecto, que va a ser la que contenga todo el contenido, lo primero, será crear la carpeta, en sí, entonces en la ubicación que deseemos, creamos una nueva carpeta, en mi caso fue en Desktop



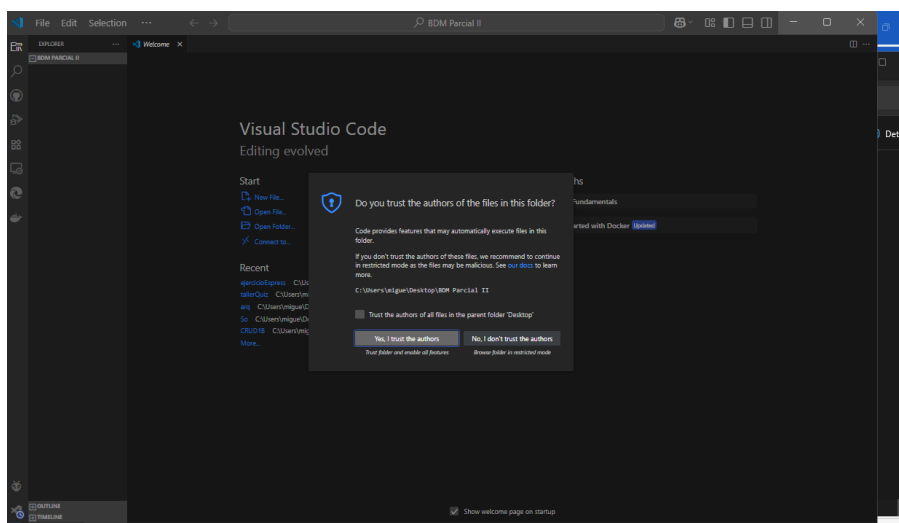
Posteriormente, se copia el path de la carpeta, que nos servirá para que en Visual Studio Code, la peguemos y nos ahorre tiempo buscando la carpeta



Posteriormente, en el entorno de Visual Studio Code, nos dirigimos a la barra de la parte superior izquierda y seleccionamos *open folder*, para empezar a trabajar en el proyecto



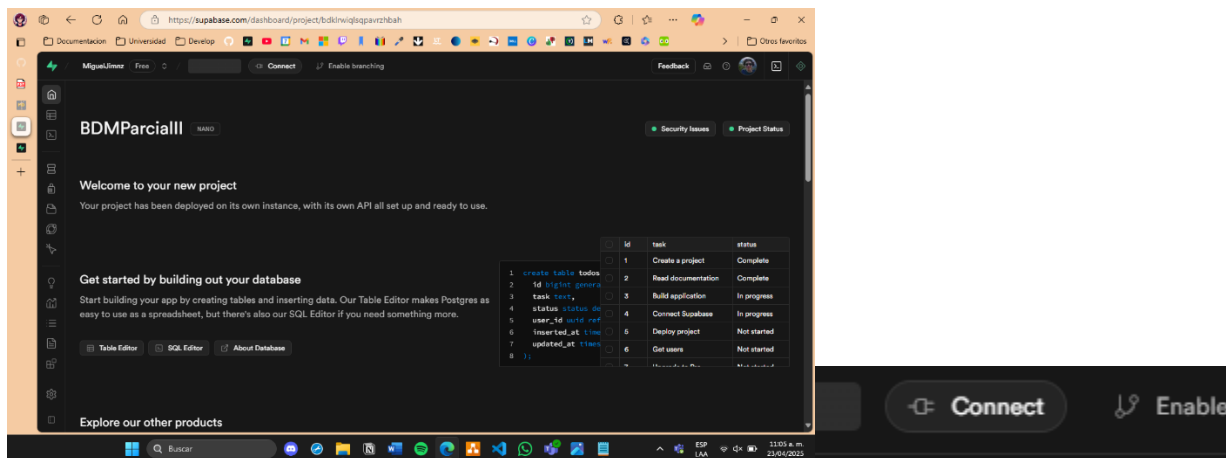
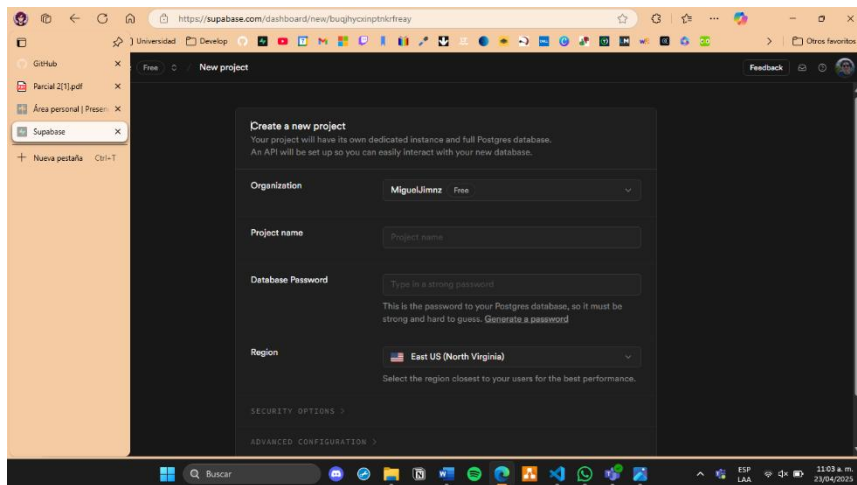
Y en la ventana emergente, seleccionamos *Trust the authors*

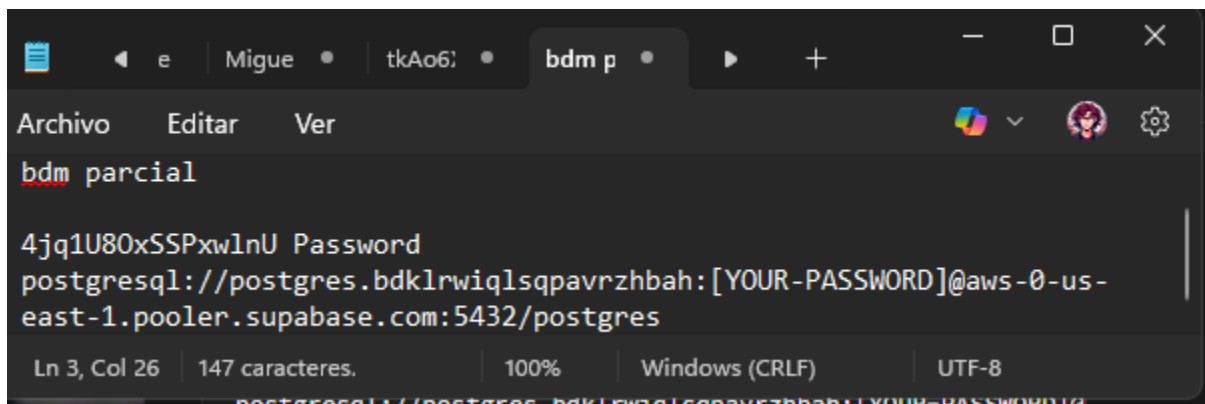
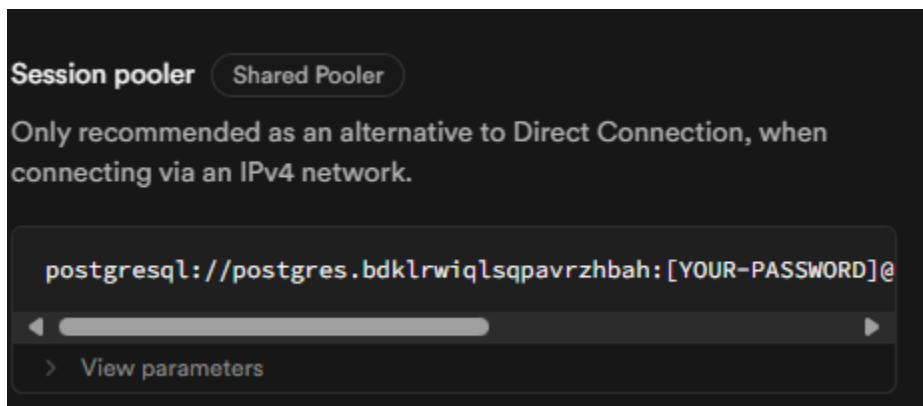


Con esto, ya estaremos en el entorno para trabajar.

## Creación de base de datos en Supabase

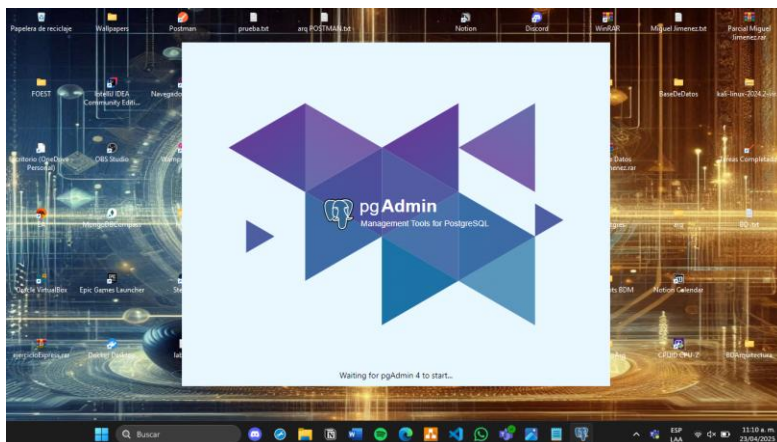
Para crear la base de datos en Supabase, lo primero será crear un nuevo proyecto de base de datos, una vez creado, ir a la sección de *Connect*, en la parte superior, y después en la parte de *Session Pooler*, y los datos registrados los guardamos para posteriormente conectarnos a la base de datos

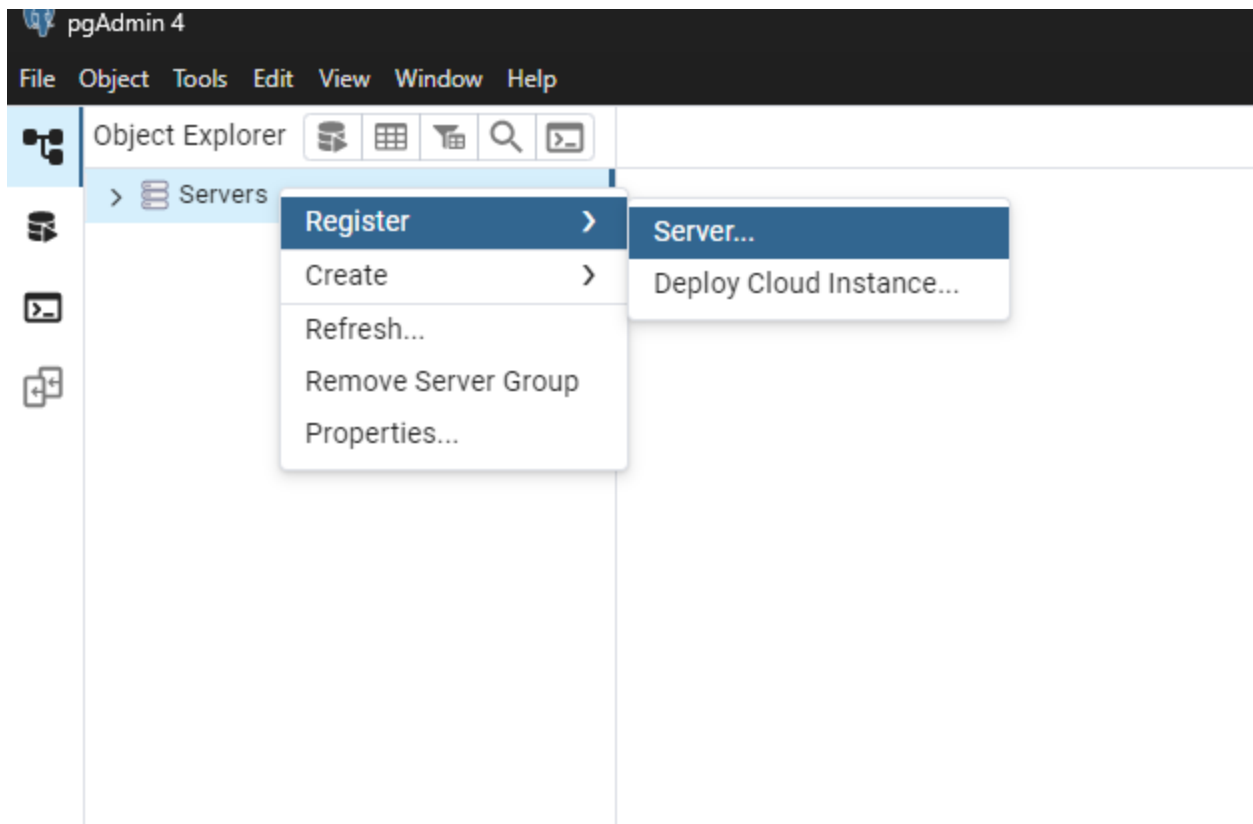




## Asociación de la base de datos con Pgadmin4

Para que Pgadmin4, sea nuestra herramienta de trabajo para la base de datos, es necesario que, primero que todo, abriendo el programa y con los datos de conexión, previamente, guardados, en el menú *Servers*, seleccionar con el click derecho -> Register-> Server





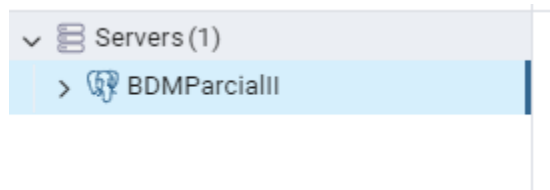
Y en la interfaz de *connection*, establecer los valores, previamente gurdados, el enlace, user, password.

A screenshot of the 'Register - Server' dialog box in pgAdmin 4. The 'Connection' tab is selected. The form contains the following fields and controls:

- Host name/address: aws-0-us-east-1.pooler.supabase.com
- Port: 5432
- Maintenance database: postgres
- Username: postgres.bdklrwiqlsqpavrzhhah
- Kerberos authentication?: ☐
- Password:
- Save password?: ☒
- Role:
- Service:

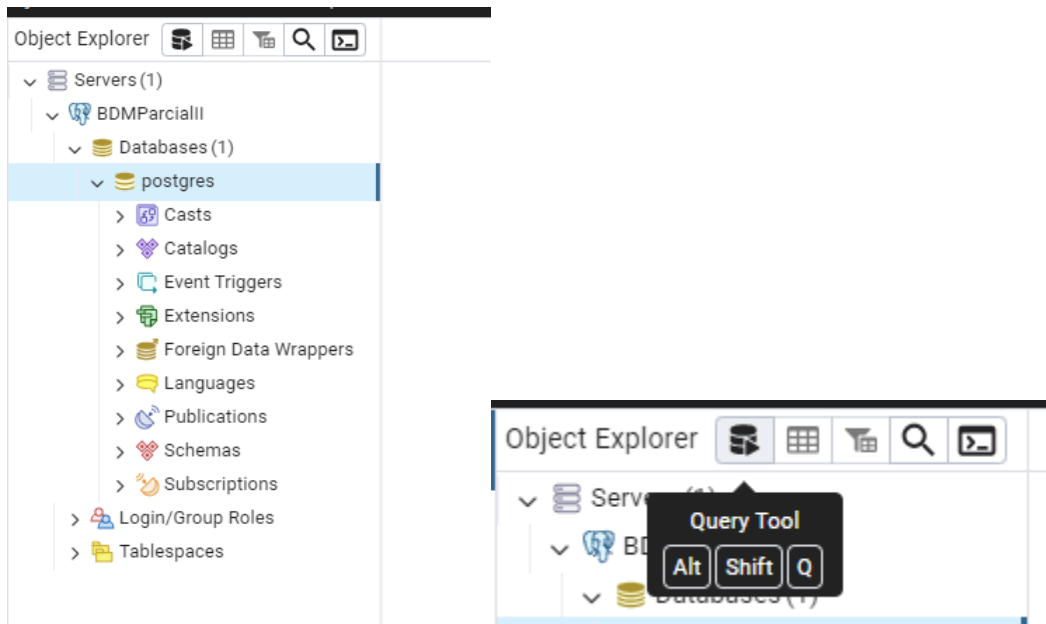
At the bottom, there are buttons for 'Close', 'Reset', and 'Save'.

Y si los datos ingresados fueron correctos, nos abrirá la conexión

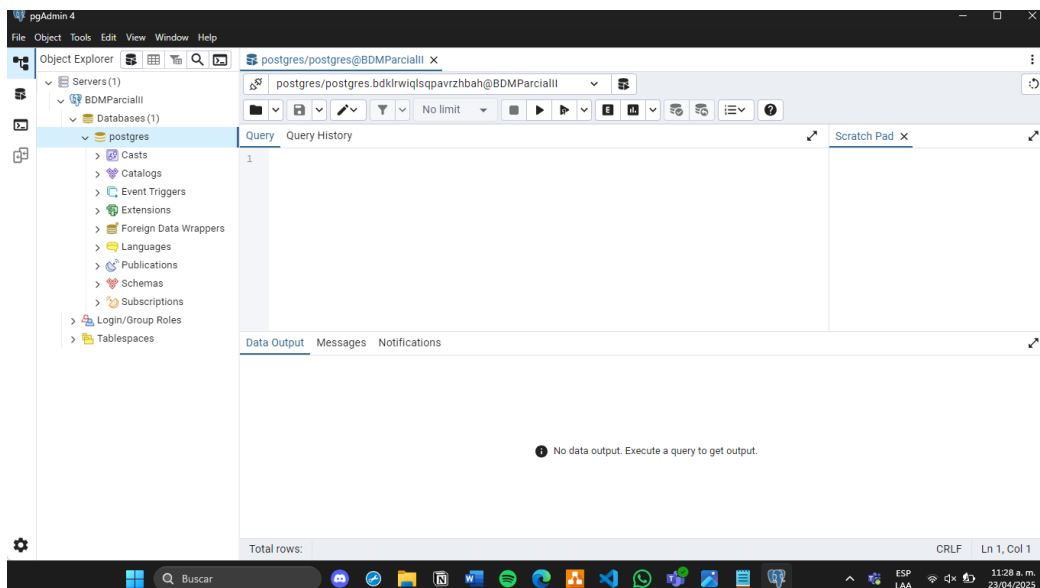


## Creación de las tablas con Pgadmin4.

Para crear las tablas del proyecto, lo primero será en el apartado de la conexión, desplegarlo, después *Databases -> postgres*, para que nos habilite el apartado de *query tool*



Y en el apartado de *query tool*, se procede a escribir los comandos de cada tabla.





## Creación de tabla Restaurante

The screenshot shows the pgAdmin 4 interface. On the left, the 'Servers' tree is expanded to 'BDMParcialII' > 'postgres'. The 'Query' tab is active, displaying the following SQL code:

```
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 );
```

Below the query editor, the 'Messages' tab shows the execution result:

```
CREATE TABLE  
  
Query returned successfully in 2 secs 70 msec.
```

The status bar at the bottom indicates 'Total rows: Query complete 00:00:02.074' and 'Ln 7, Col 3'.

## Creación de tabla Empleado

The screenshot shows the pgAdmin 4 interface. On the left, the 'Servers' tree is expanded to 'BDMParcialII' > 'postgres'. The 'Query' tab is active, displaying the following SQL code:

```
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 );
```

Below the query editor, the 'Messages' tab shows the execution result:

```
CREATE TABLE  
  
Query returned successfully in 183 msec.
```

## Creación de tabla Producto

The screenshot shows a database management interface with a sidebar on the left containing a tree view of database objects. The main area is titled 'Query' and 'Query History'. The SQL query being executed is:

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

Below the query editor, there are tabs for 'Data Output', 'Messages', and 'Notifications'. The 'Messages' tab is selected, showing the output of the query execution.

## Creación de tabla Pedido

The screenshot shows the same database management interface as the previous one, but with a different SQL query being executed. The query is:

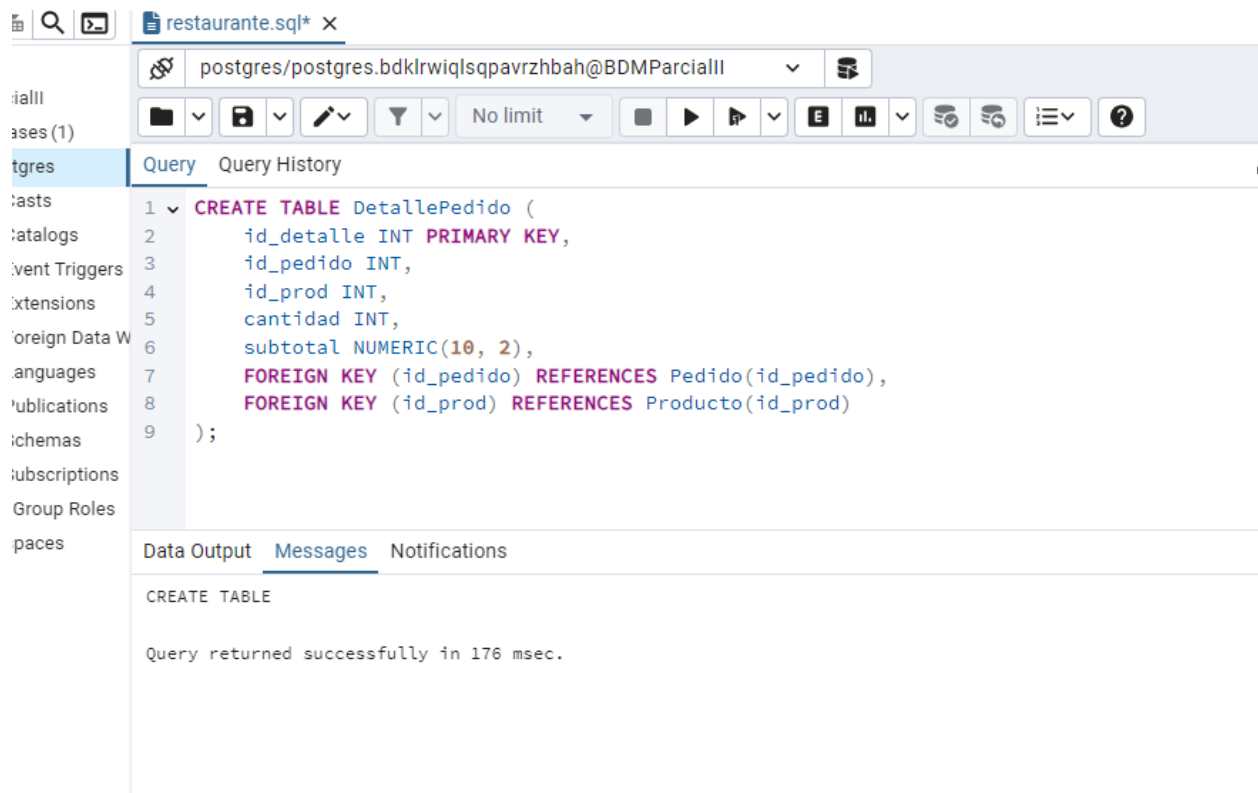
```
1 CREATE TABLE Pedido (  
2     id_pedido INT PRIMARY KEY,  
3     fecha DATE,  
4     id_rest INT,  
5     total NUMERIC(10, 2),  
6     FOREIGN KEY (id_rest) REFERENCES Restaurante(id_rest)  
7 );
```

Below the query editor, the 'Messages' tab is selected, showing the output of the query execution:

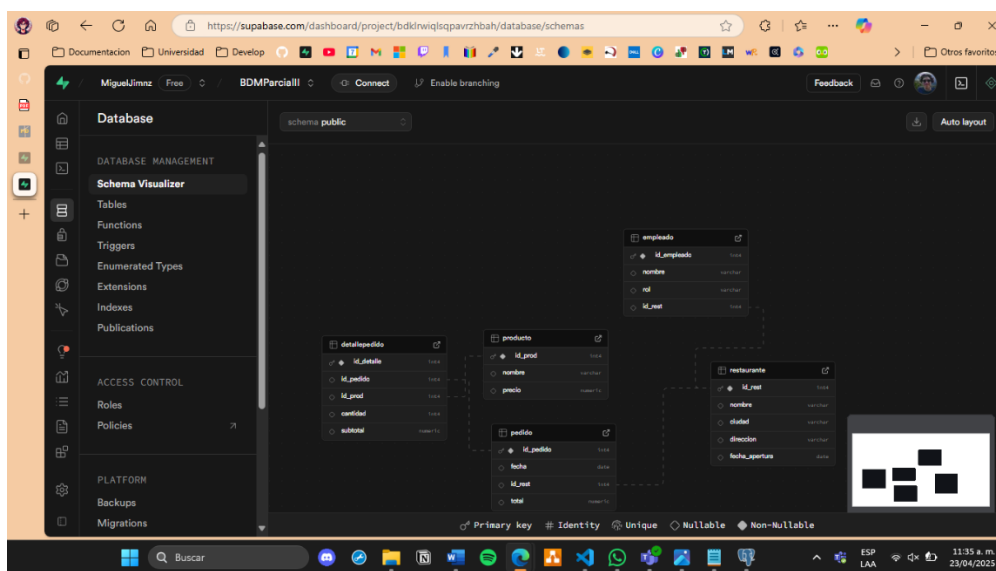
```
CREATE TABLE
```

Query returned successfully in 172 msec.

## Creación de tabla DetallePedido



Una vez, con las tablas hechas, solo falta revisar en la conexión de supabase, el modelo de las tablas.



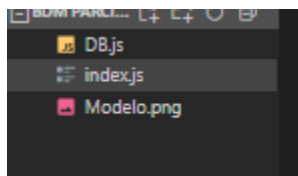
Y con el modelo, lo podemos descargar



## Conexión de la base de datos en la carpeta

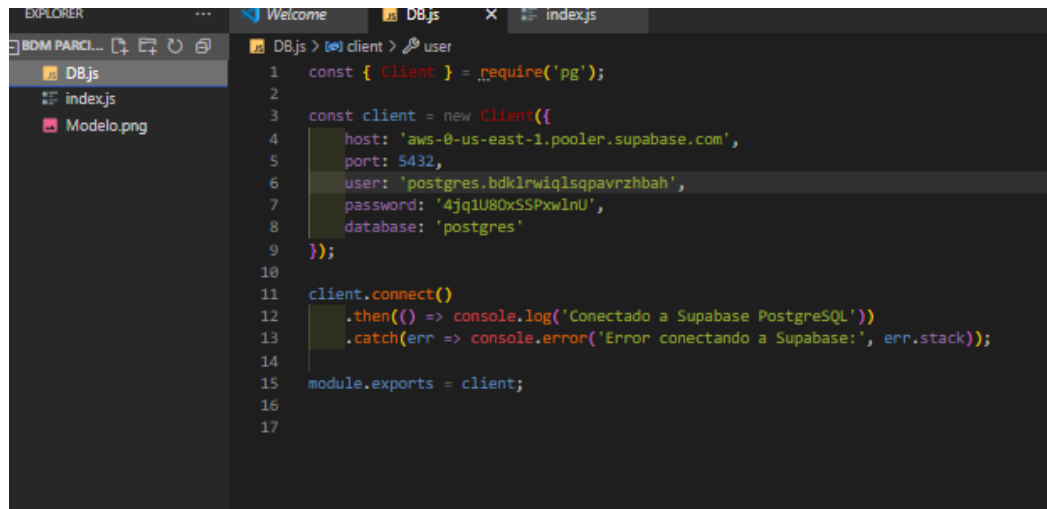
Para ello, lo primero es crear los documentos

1. DB.js
2. Index.js



Una vez creado, se establece la conexión para la base de datos y se verifica su conexión

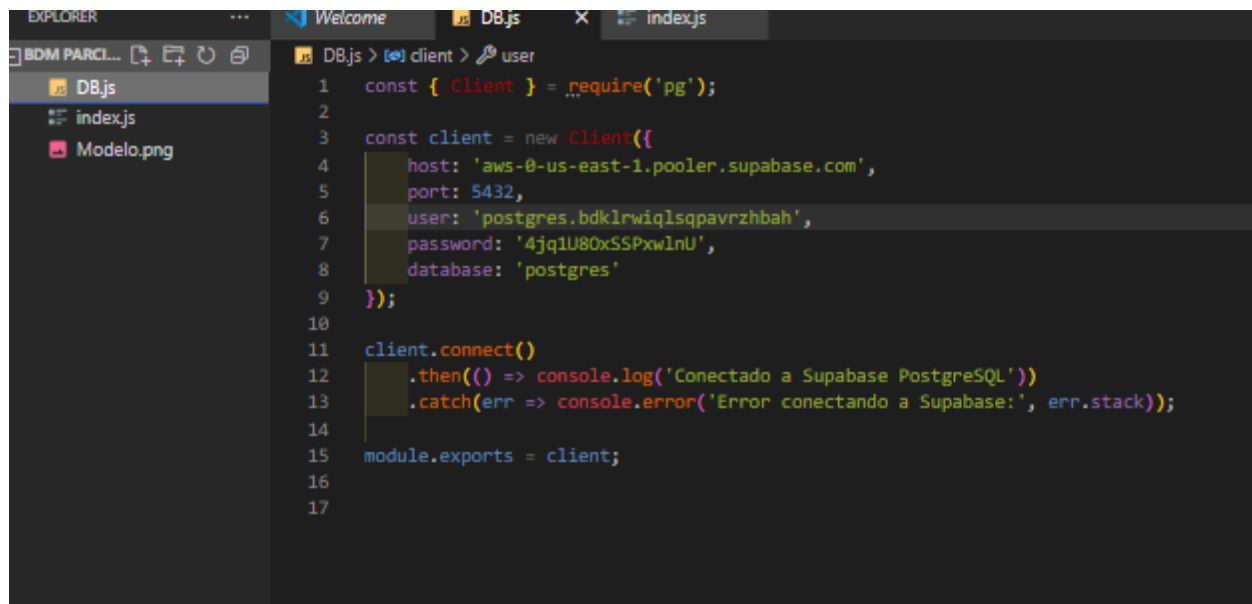
DB.js para la conexión



The screenshot shows the VS Code editor with the Explorer sidebar on the left displaying a project structure with files DB.js, index.js, and Modelo.png. The main editor window shows the content of DB.js, which is a JavaScript file for connecting to a Supabase PostgreSQL database. The code includes a Client class definition, connection parameters (host, port, user, password, database), and a client.connect() call with .then() and .catch() handlers for logging success or errors. The module is exported as client.

```
1 const { Client } = require('pg');
2
3 const client = new Client({
4   host: 'aws-0-us-east-1.pooler.supabase.com',
5   port: 5432,
6   user: 'postgres.bdklrwiqlsqpavrzhbah',
7   password: '4jq1U80xSSPxwlnU',
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;
16
17
```

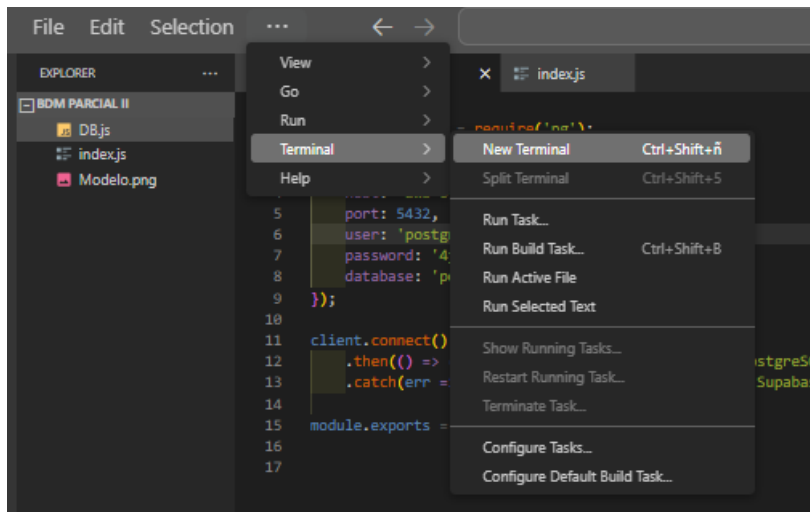
Y index.js para que contenga todas las apis



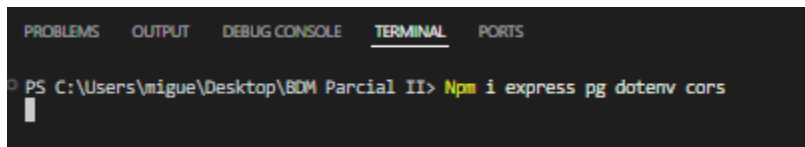
The screenshot shows the VS Code editor with the Explorer sidebar on the left displaying a project structure with files DB.js, index.js, and Modelo.png. The main editor window shows the content of index.js, which is a JavaScript file that imports the client from DB.js and exports it. The code is very similar to DB.js, including the Client class definition, connection parameters, and a client.connect() call with .then() and .catch() handlers. The module is exported as client.

```
1 const { Client } = require('pg');
2
3 const client = new Client({
4   host: 'aws-0-us-east-1.pooler.supabase.com',
5   port: 5432,
6   user: 'postgres.bdklrwiqlsqpavrzhbah',
7   password: '4jq1U80xSSPxwlnU',
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;
16
17
```

Y ahora con new terminal, iniciar todo el proceso para node.js y express



Y se importan las bibliotecas



Y se confirman estas bibliotecas, con los documentos creados

