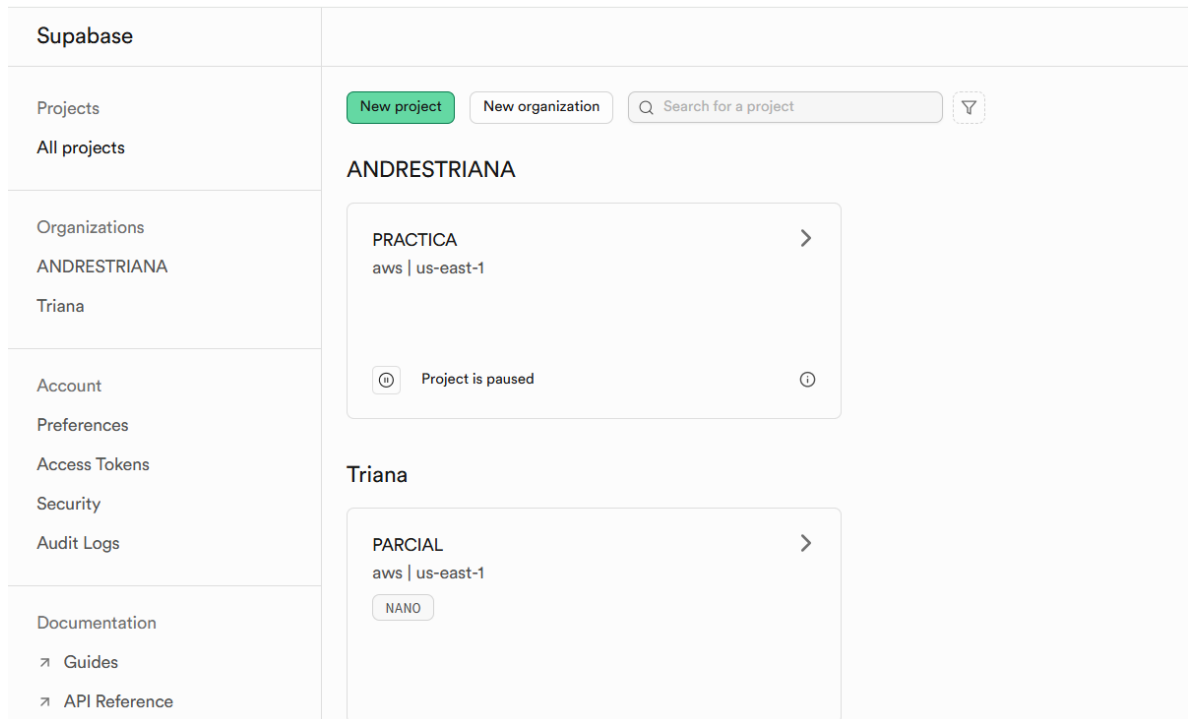


PARCIAL 2 CORTE  
ANDRES FELIPE TRIANA TORRES  
BASES DE DATOS MASIVAS

1. Ingresamos a la plataforma de Supabase y procedemos a crear un nuevo proyecto desde el panel principal.



- 2.

Una vez creado el proyecto en Supabase, hacemos clic en la opción **“Connect”** para obtener los datos de conexión a la base de datos. Supabase proporciona una URL tipo **Session Pooler**, que se utiliza para conectarse desde clientes externos como **pgAdmin**.

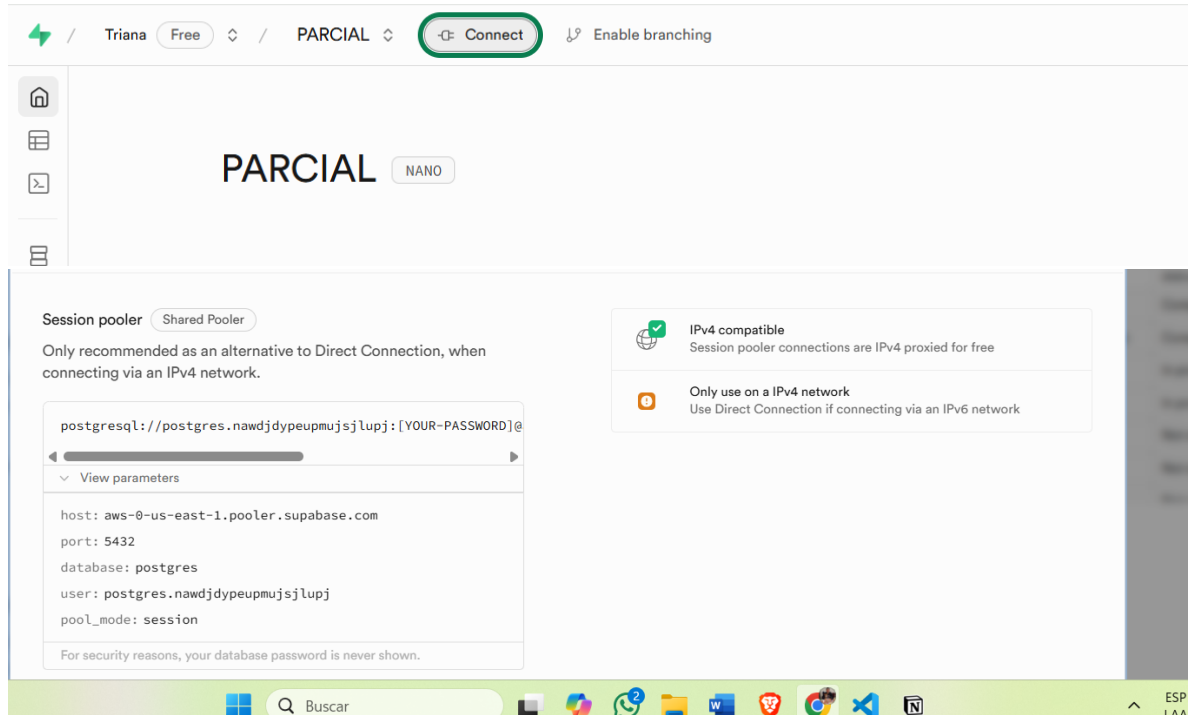
El formato de conexión es el siguiente:

bash

CopiarEditar

```
postgresql://postgres.bqcdnezerhzkdvmzagbe:[YOUR-PASSWORD]@aws-0-us-east-1.pooler.supabase.com:5432/postgres
```

si deseas ver más detalles o copiar los parámetros de conexión por separado, puedes hacer clic en **“View Parameters”**, donde también se muestran enlaces útiles para conectarse.

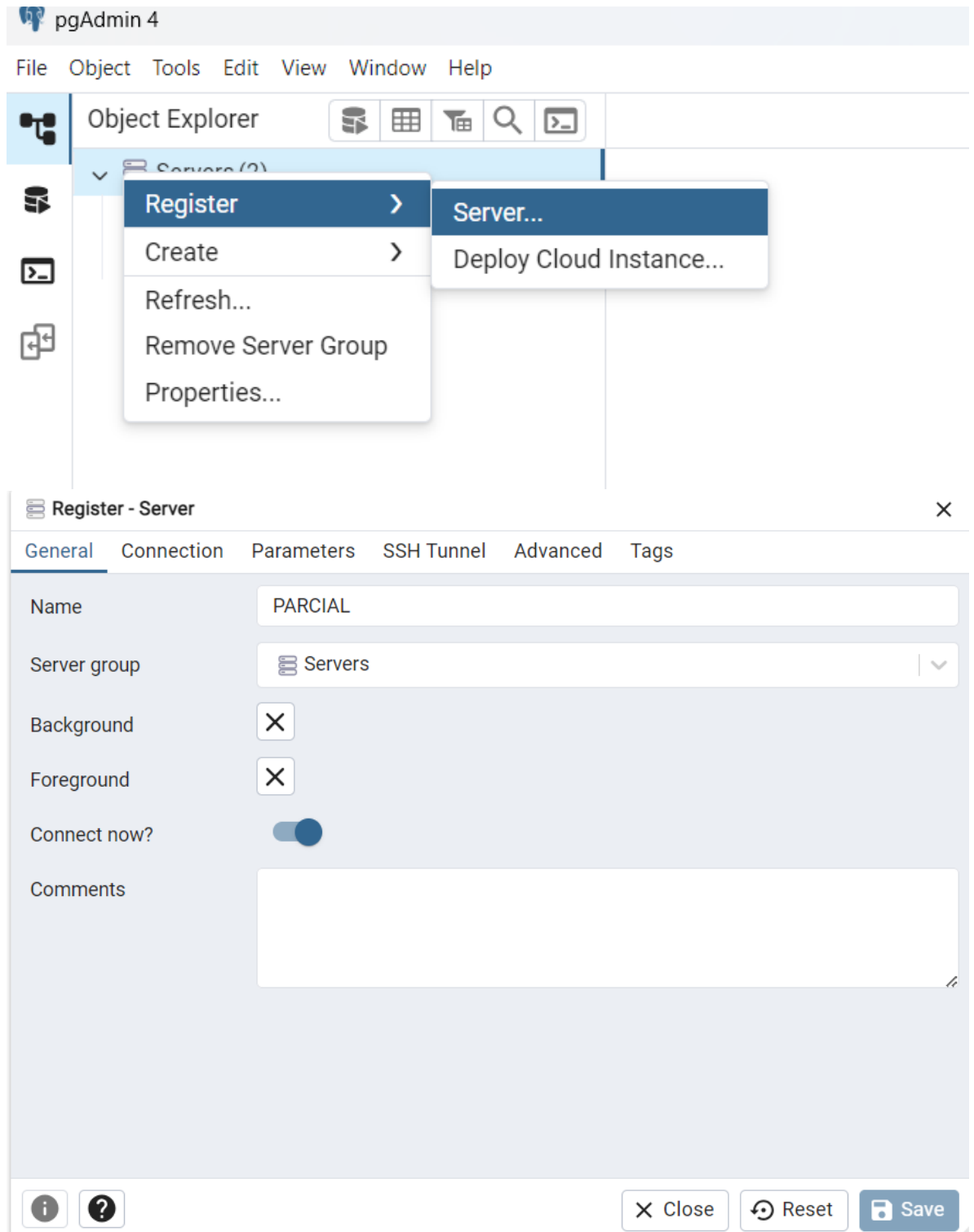


3.

Después de obtener los datos de conexión en Supabase, abrimos **pgAdmin** para registrar un nuevo servidor. Para ello, hacemos clic derecho sobre la opción **"Servers"**, seleccionamos **"Register"** y luego **"Server..."**. Esto abrirá una ventana emergente donde debemos configurar los parámetros de conexión.

En la pestaña **General**, asignamos un nombre descriptivo al servidor, como por ejemplo "SupabaseDB". Luego, en la pestaña **Connection**, llenamos los campos con la información obtenida del *Session Pooler* de Supabase: en **Host name/address** colocamos el host, en **Port** dejamos el valor por defecto 5432, en **Username** ingresamos el usuario proporcionado y en **Password** escribimos la contraseña que configuramos al crear el proyecto. Finalmente, en **Maintenance database** colocamos postgres.

Una vez completados todos los campos, hacemos clic en **Save**. Si los datos son correctos, se establecerá la conexión y podremos acceder a nuestra base de datos de Supabase directamente desde pgAdmin.





X

## Connection

## SSH Tunnel

## Tags

aws-0-us-east-1.pooler.supabase.com

5432

postgres

postgres.nawdjdypeupmujsjlupi

☐

• • • •

☐ Reset


 Save

pgAdmin 4


File Object Tools Edit View Window Help

## Object Explorer



✓  Servers (3)



>  PARCIAL



>  PRACTICA

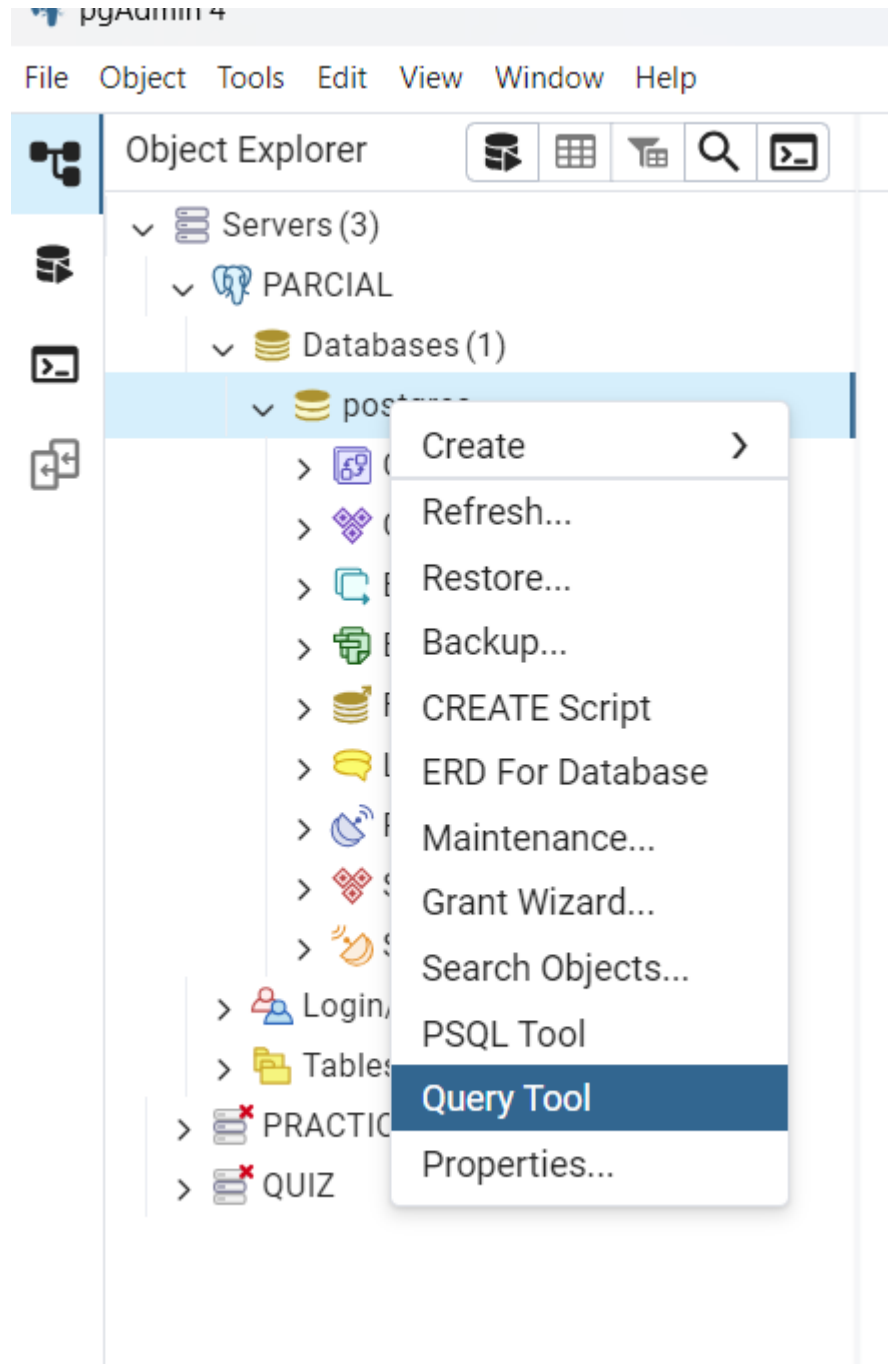


>  QUIZ

4.

Para crear las tablas en la base de datos desde **pgAdmin**, primero accedemos a la base de datos correspondiente (en este caso, llamada parcial). Hacemos clic sobre su nombre para seleccionarla. Luego, en el panel superior o haciendo clic derecho sobre la base de datos, seleccionamos la opción **"Query Tool"** (herramienta de consultas).

Esta acción abrirá una ventana emergente en la que podremos escribir comandos SQL. Desde allí es posible crear tablas, insertar datos o realizar cualquier tipo de consulta.



Query Query History

```
1  ✓ CREATE TABLE restaurante (  
2      id_rest INT PRIMARY KEY,  
3      nombre VARCHAR(100) NOT NULL,  
4      ciudad VARCHAR(100) NOT NULL,  
5      direccion VARCHAR(150) NOT NULL,  
6      fecha_apertura DATE NOT NULL  
7  );
```

```
CREATE TABLE empleado (  
    id_empleado INT PRIMARY KEY,  
    nombre VARCHAR(100) NOT NULL,  
    rol VARCHAR(50) NOT NULL,  
    id_rest INT NOT NULL,  
    FOREIGN KEY (id_rest) REFERENCES restaurante(id_rest)  
);
```

```
✓ CREATE TABLE producto (  
    id_prod INT PRIMARY KEY,  
    nombre VARCHAR(100) NOT NULL,  
    precio NUMERIC(10,2) NOT NULL  
);
```

```

v CREATE TABLE pedido (
    id_pedido INT PRIMARY KEY,
    fecha DATE NOT NULL,
    id_rest INT NOT NULL,
    total NUMERIC(10,2) NOT NULL,
    FOREIGN KEY (id_rest) REFERENCES restaurante(id_rest)
);

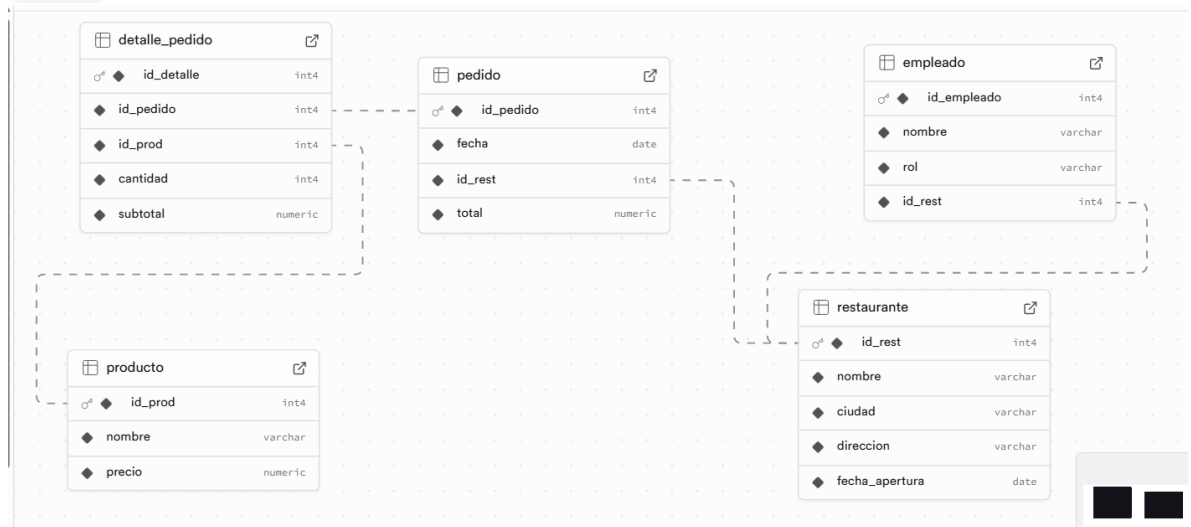
```

## Query Query History

```

1 v CREATE TABLE detalle_pedido (
2     id_detalle INT PRIMARY KEY,
3     id_pedido INT NOT NULL,
4     id_prod INT NOT NULL,
5     cantidad INT NOT NULL,
6     subtotal NUMERIC(10,2) NOT NULL,
7     FOREIGN KEY (id_pedido) REFERENCES pedido(id_pedido),
8     FOREIGN KEY (id_prod) REFERENCES producto(id_prod)
9 );

```



5. Creamos una nueva carpeta y la abrimos en Visual Studio Code para comenzar el ejercicio.

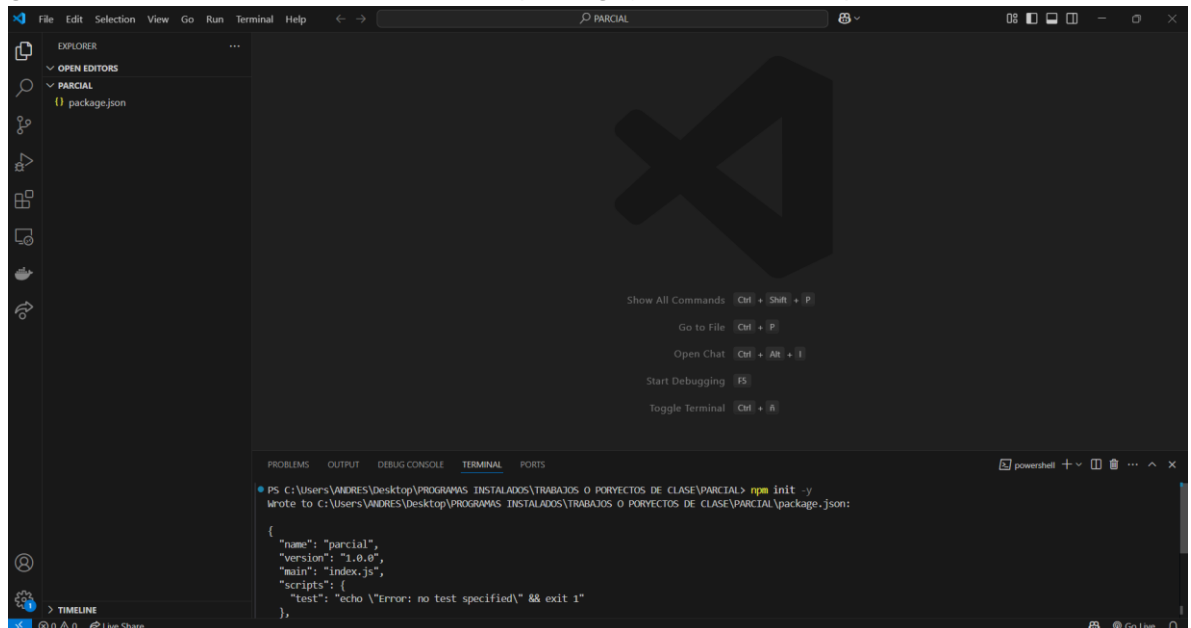
Nombre	Fecha de modificación	Tipo	Tamaño
BASESEDDATOS	9/04/2025 3:04 p. m.	Carpeta de archivos	
PARCIAL	23/04/2025 10:47 a. m.	Carpeta de archivos	
PRACTICA	14/04/2025 9:26 p. m.	Carpeta de archivos	
QUIZ_BASES_DE_DATOS	11/04/2025 12:48 p. m.	Carpeta de archivos	

6.

Abrimos la terminal en Visual Studio Code y escribimos el siguiente comando:

`npm init -y`

Este comando se utiliza para **inicializar rápidamente un proyecto en Node.js**, generando automáticamente un archivo `package.json` con valores predeterminados.



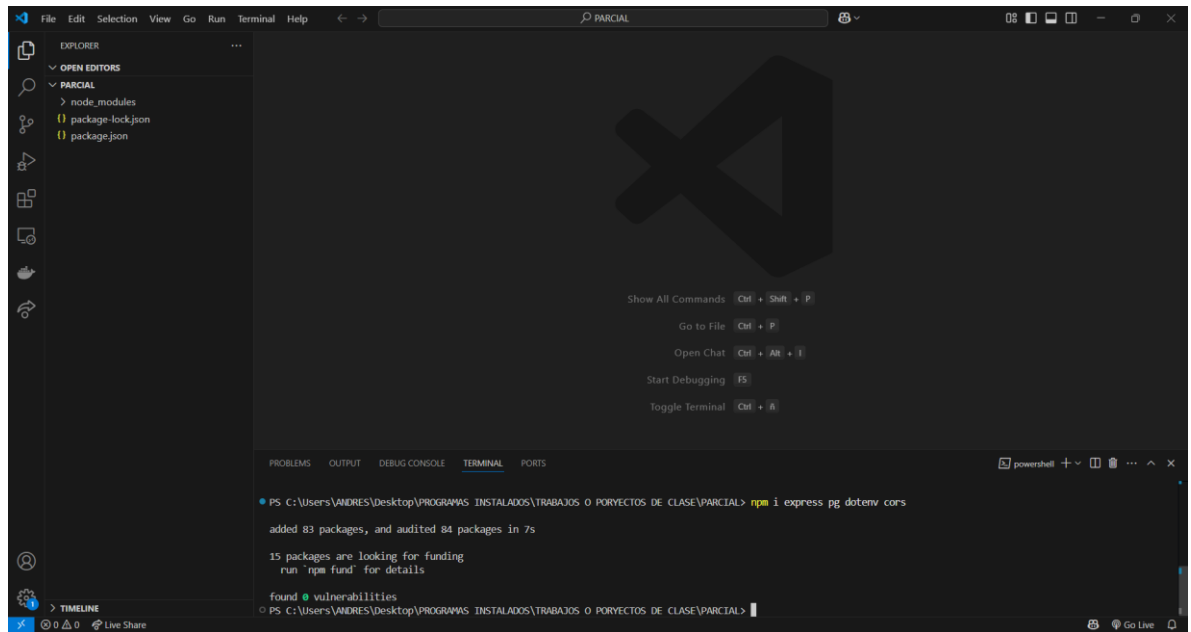
7.

Luego, escribimos el siguiente comando:

`npm i express pg dotenv cors`

Este comando se utiliza para instalar **cuatro paquetes fundamentales** al momento de desarrollar un servidor con Node.js que se conectará a una base de datos PostgreSQL.



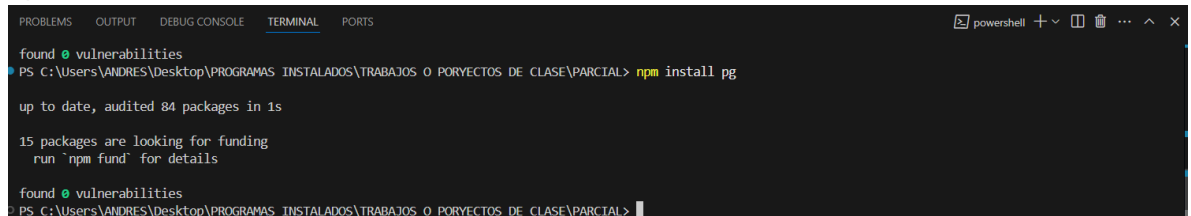


8.

Luego, ejecutamos el siguiente comando:

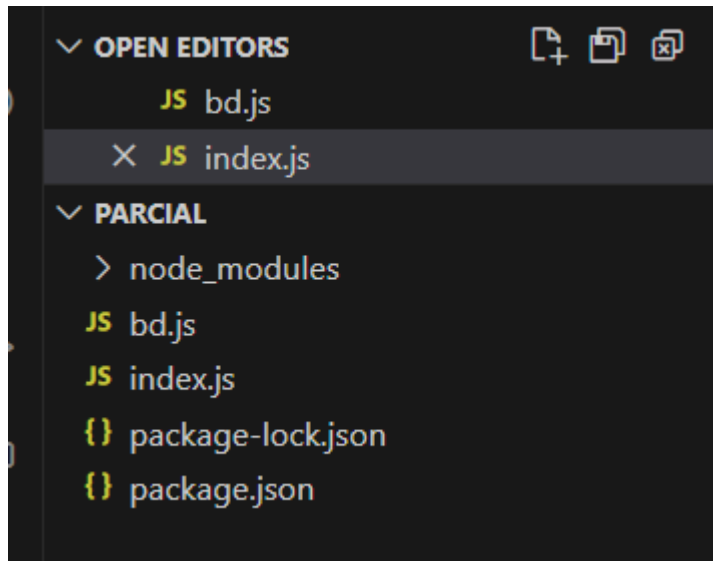
```
npm install pg
```

Este comando nos permite instalar **el cliente oficial de PostgreSQL para Node.js**, conocido como pg. Con esta herramienta podremos establecer conexiones con una base de datos PostgreSQL y realizar consultas directamente desde nuestra aplicación.



9.

Luego, abrimos dos páginas más en las que escribiremos el código correspondiente.



10.

El siguiente código permite configurar y verificar la conexión a una base de datos PostgreSQL alojada en Supabase.

```
const { Pool } = require('pg');
```

```
const pool = new Pool({  
  host: 'aws-0-us-east-1.pooler.supabase.com',  
  port: 5432,  
  user: 'postgres.nawdjdypeupmujsjlupj',  
  password: '1234',  
  database: 'postgres',  
  ssl: { rejectUnauthorized: false }  
});
```

```
pool.connect((err, client, release) => {  
  if (err) {  
    console.error('Error conectando con la base de datos', err.stack);  
  } else {  
    console.log('Conectado a la base de datos');  
    release();  
  }  
});
```

```
module.exports = pool;
```

```
1 const { Pool } = require('pg');
2
3
4 const pool = new Pool({
5   host: 'aws-0-us-east-1.pooler.supabase.com',
6   port: 5432,
7   user: 'postgres.nawjdypeupmujsjlup',
8   password: '1234',
9   database: 'postgres',
10  ssl: { rejectUnauthorized: false }
11 });
12
13
14 pool.connect((err, client, release) => {
15   if (err) {
16     console.error('Error conectando con la base de datos', err.stack);
17   } else {
18     console.log('Conectado a la base de datos');
19     release();
20   }
21 });
22
23 module.exports = pool;
```

found 0 vulnerabilities  
PS C:\Users\ANDRES\Desktop\PROGRAMAS INSTALADOS\TRABAJOS O PROYECTOS DE CLASE\PARCIAL> npm install pg

up to date, audited 84 packages in 1s

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

found 0 vulnerabilities  
PS C:\Users\ANDRES\Desktop\PROGRAMAS INSTALADOS\TRABAJOS O PROYECTOS DE CLASE\PARCIAL>

13. Como siguiente paso, abrimos Postman y hacemos clic en el botón "+" para crear una nueva pestaña de solicitud. Luego, seleccionamos la opción **"REST API (Basic)"**. Al hacerlo, se mostrarán cuatro métodos principales: **GET**, **POST**, **DELETE** y **PUT**, que utilizaremos para interactuar con nuestra API.

