Parcial 2

Jose Y. Ospina Pacheco

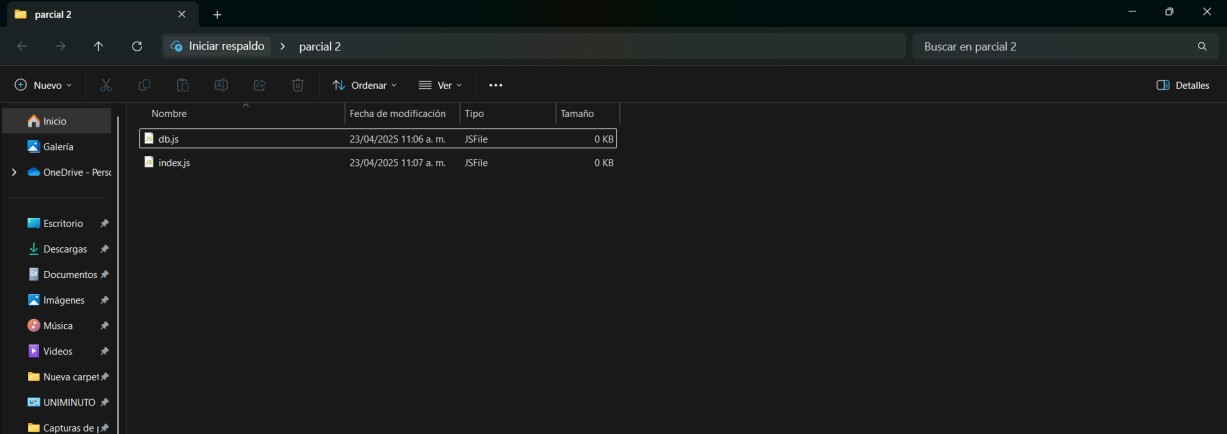
Ingeniería de sistemas, Corporación universitaria minuto Dios

NRC-10-60747, Bases de Datos Masivas

WILLIAM A. MATALLANA PORRAS

23/04/2025

1. Creamos una carpeta donde vamos a guardar nuestro proyecto he incluiremos los archivos index.js y db.js



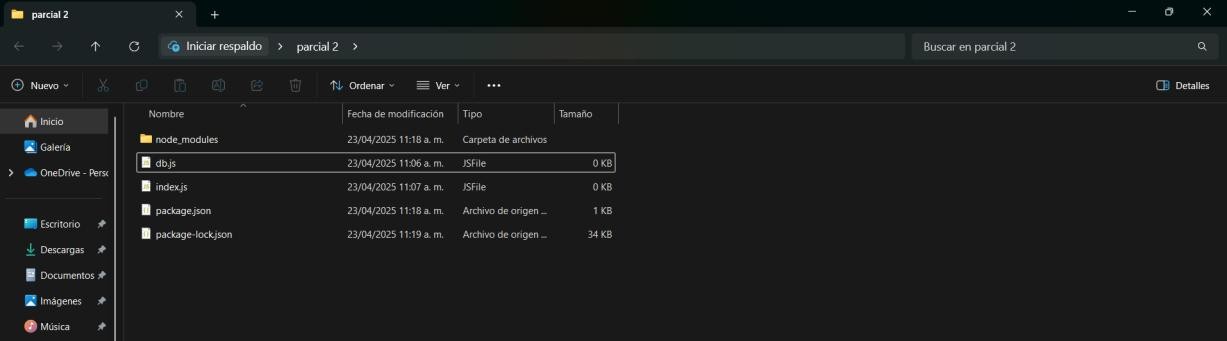
1. Abrimos nuestra carpeta en visual estudio, abrimos una terminal y empezamos a descargar las dependencias de Node.js

* + npm init -y
  + npm install express cors pg
  + npm install pg

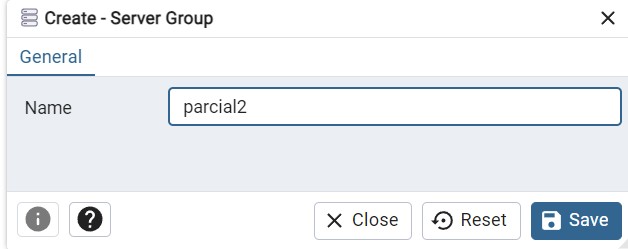
El comando npm init -y se utiliza para iniciar un proyecto en Node.js de forma rápida. Al ejecutarlo, se crea automáticamente un archivo llamado package.json, el cual contiene información básica del proyecto como el nombre, la versión y las dependencias. La opción -ypermite aceptar todas las configuraciones por defecto sin necesidad de ir respondiendo preguntas.

Por otro lado, el comando npm install express cors pg sirve para instalar tres dependencias esenciales en el desarrollo de aplicaciones web. La primera es expressun framework ligero que facilita la creación de servidores y rutas en Node.js. La segunda es cors, un middleware que permite habilitar o restringir el acceso a recursos desde diferentes dominios, lo cual es muy útil cuando se trabaja con frontend y backend por separado. La tercera es pg, un cliente que permite establecer conexiones con bases de datos PostgreSQL y ejecutar consultas desde Node.js.

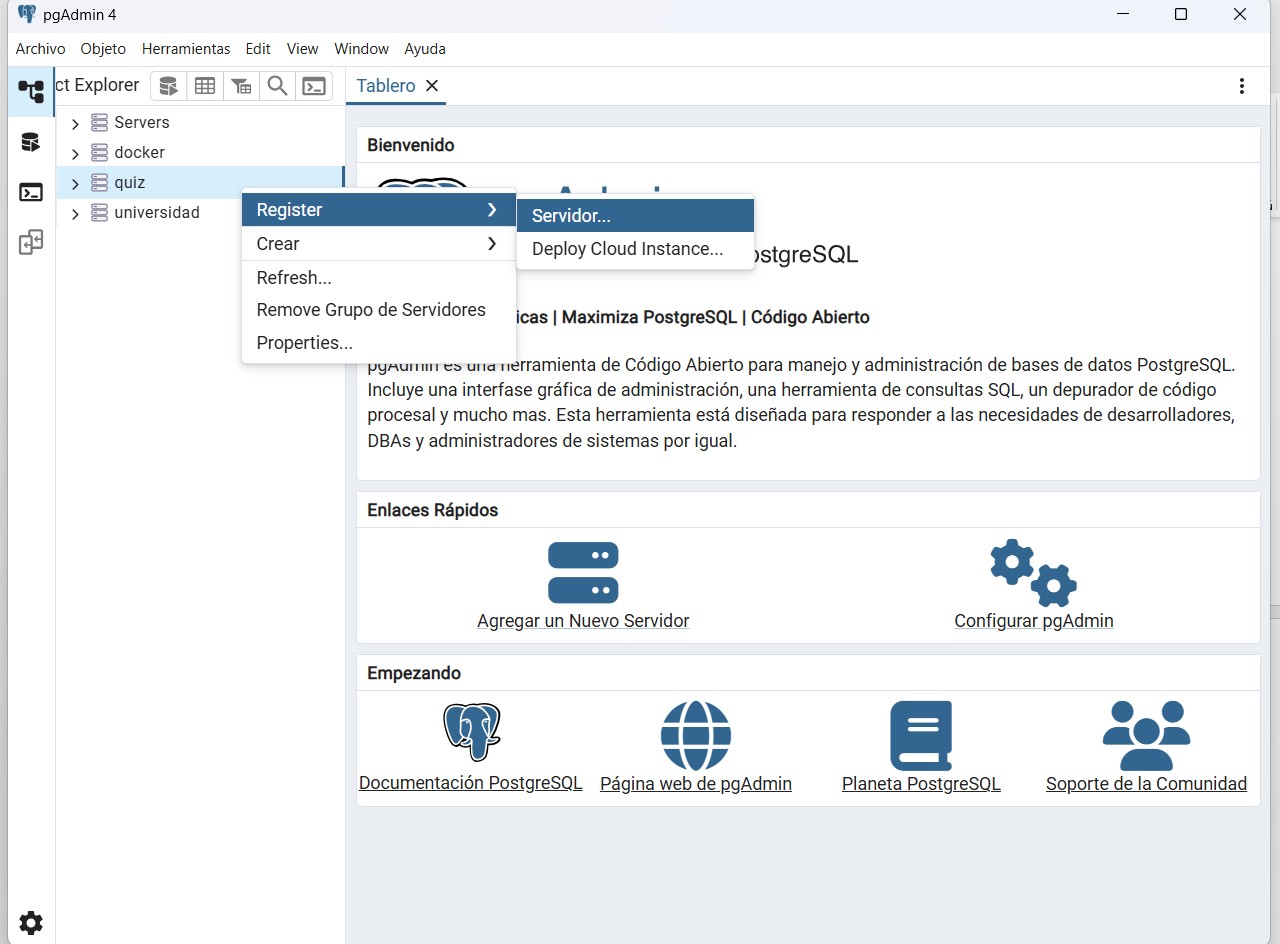
Finalmente, el comando npm install pg permite instalar únicamente la dependencia pg. Este comando es útil si ya tienes configurado tu proyecto con otras librerías y ahora necesitas agregar soporte para conectarte a una base de datos PostgreSQL.



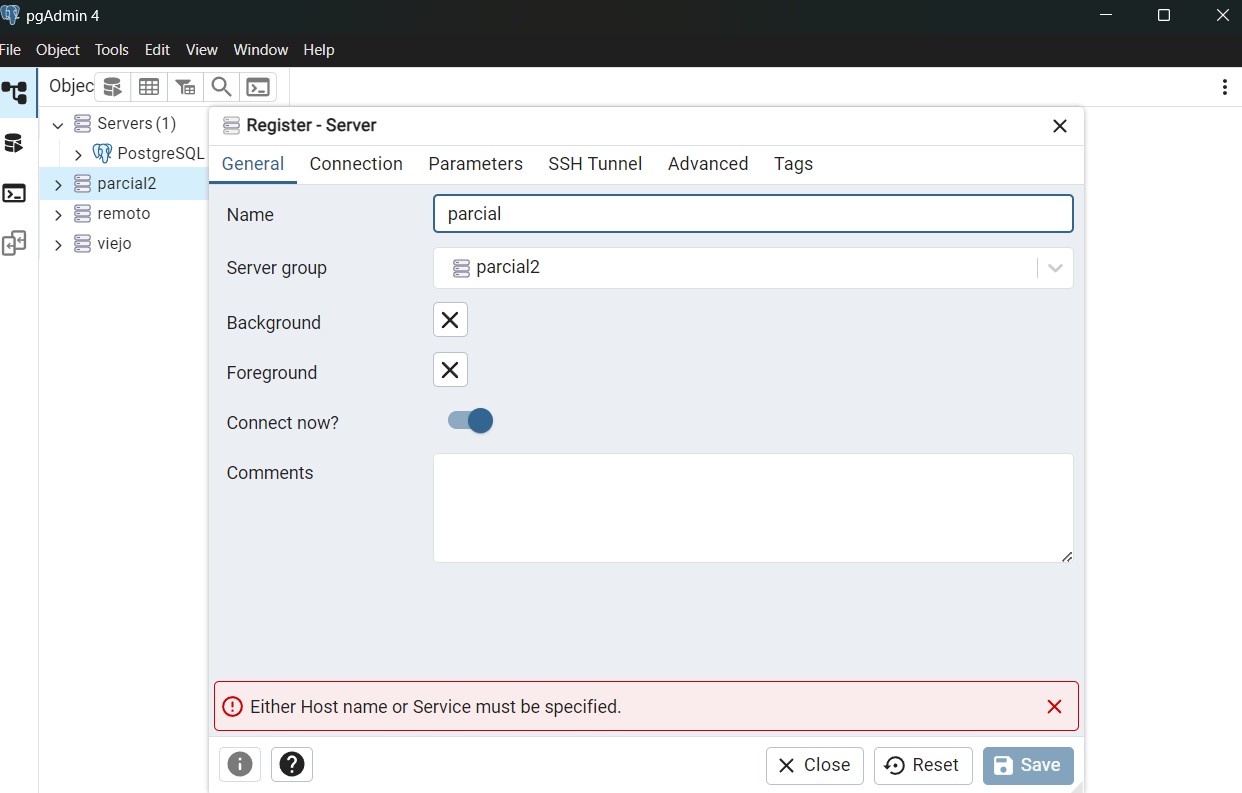
1. procedemos a realizar la conexión de pgadmin con supabase, nos dirigimos al cliente y con clic derecho sobre servers seleccionamos crear grupo de servidores y asignamos un nombre.



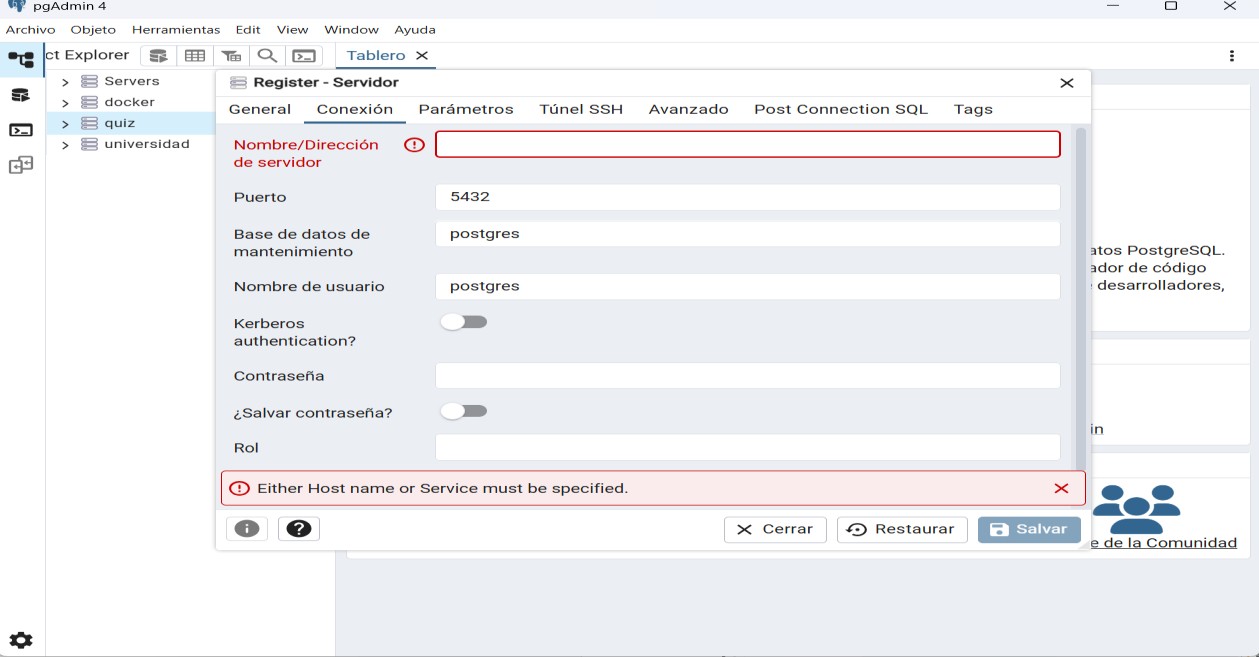
Realizamos el mismo procedimiento al nuevo servidor que creamos que se llama quiz y clicamos en la opción crear servidor y le asignamos un nombre.



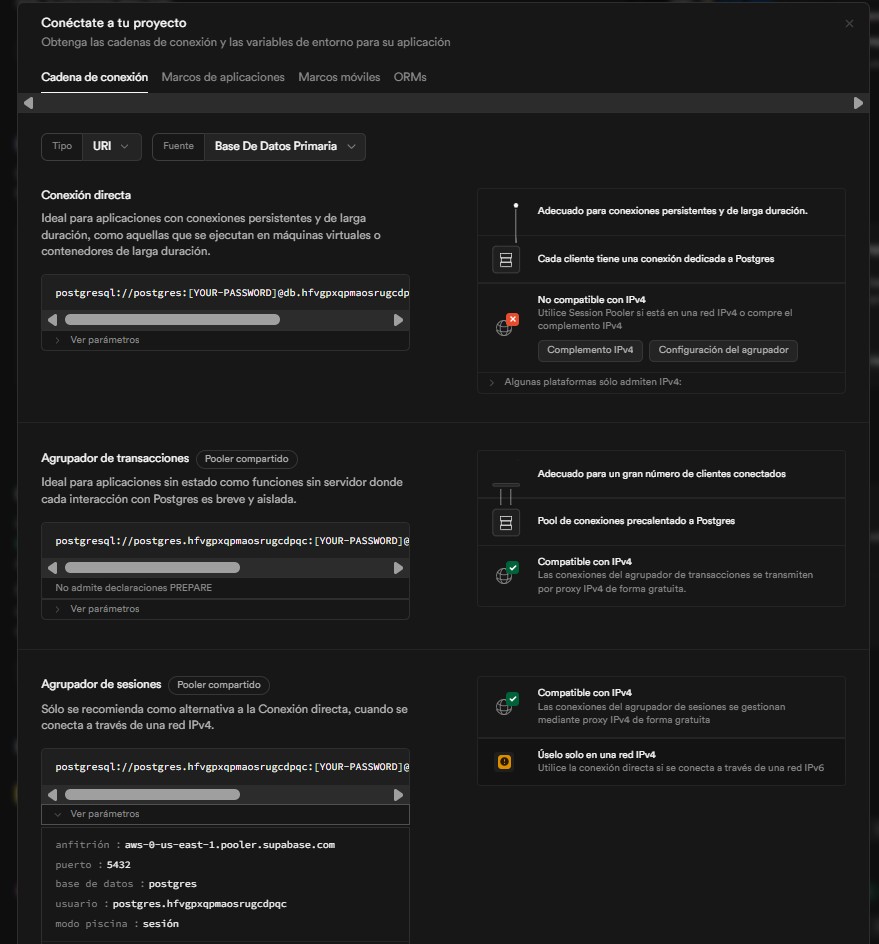
Le asignamos un nombre y luego procedemos a realizar la conexión con supabase



En esta nueva ventana pondremos las credenciales que nos brinda supabase

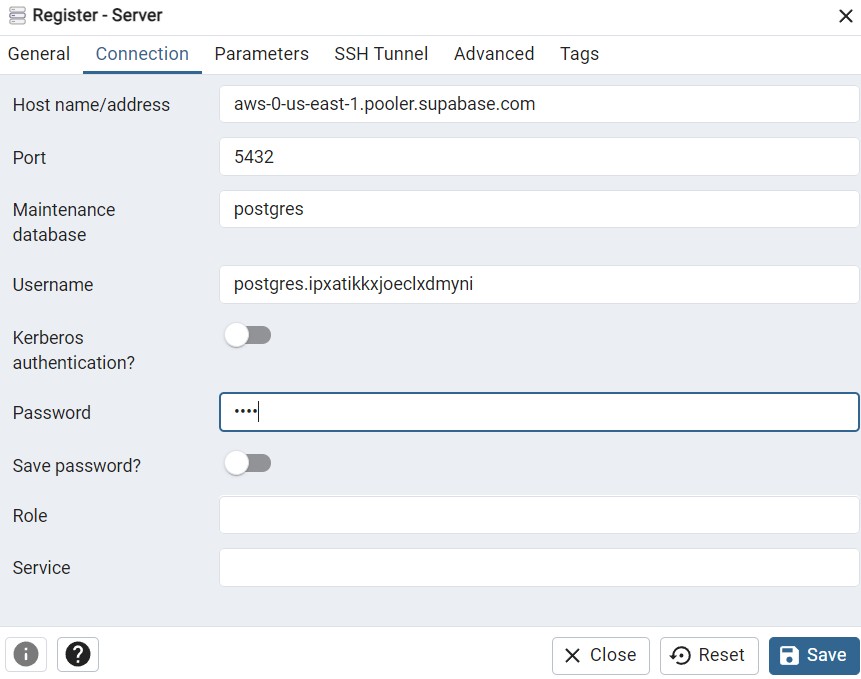


Nos dirigimos a supabase, creamos la conexión luego de ingresar en la parte superior nos encontramos con la opción conectar y veremos las credenciales necesarias para realizar dicha conexión.

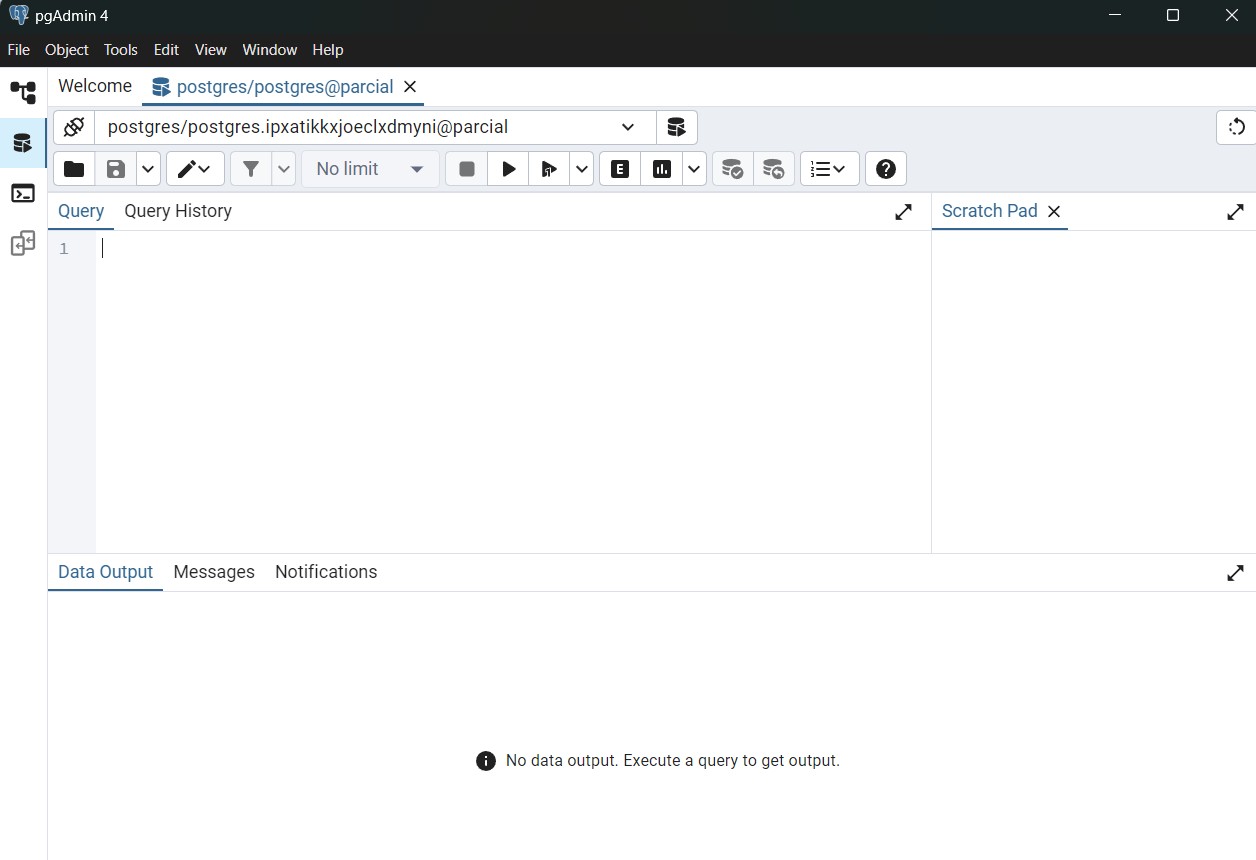


En el ultimo recuadro encontramos las credenciales que debemos poner en pgadmin

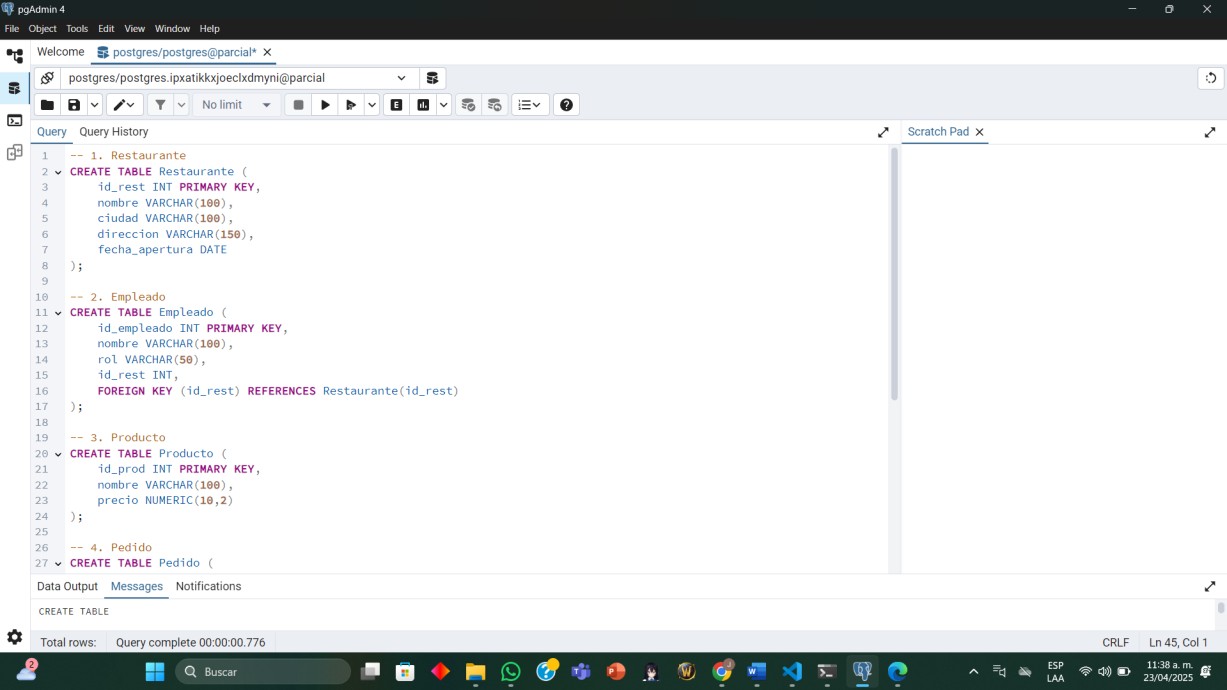
que es el cliente.



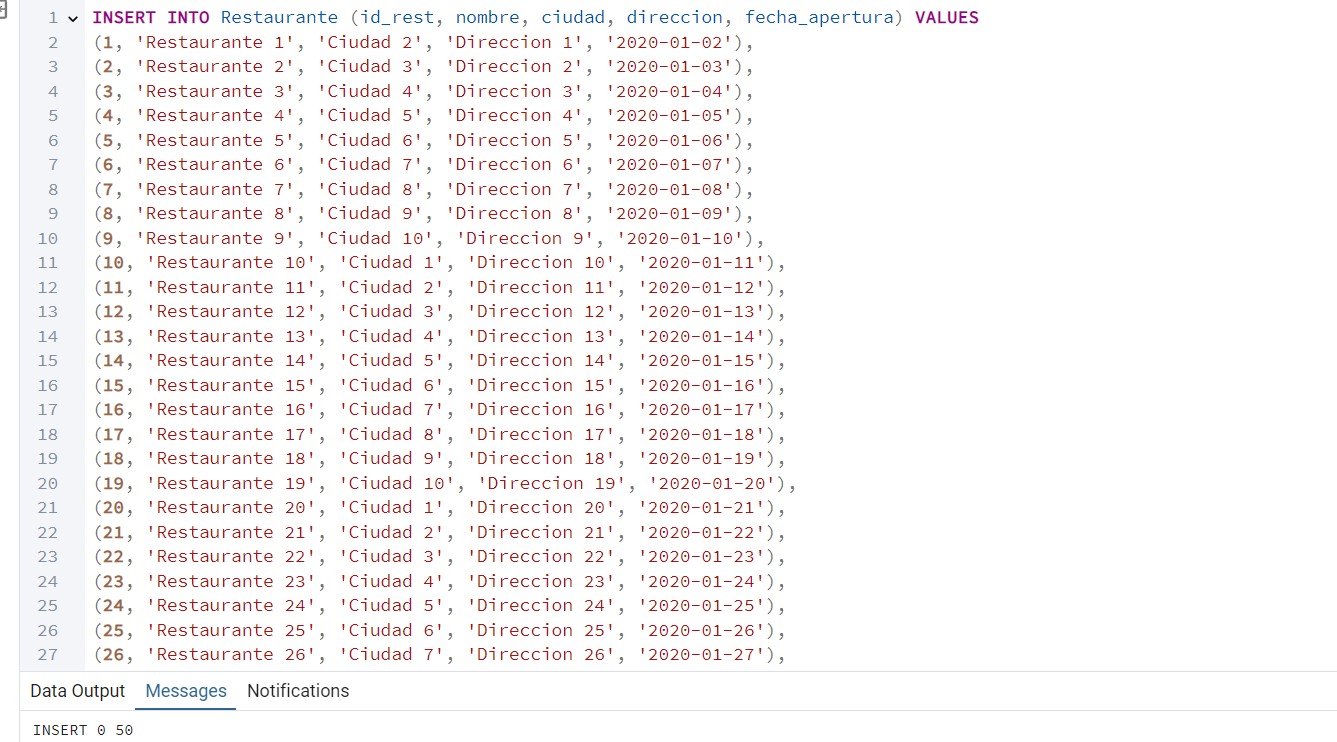
Desplegamos el menú practica y clicamos en bases de datos, clicamos en postgres y presionamos clic derecho y bajamos a la opción que dice herramientas de consulta para realizar la creación de las respectivas tablas y registros.



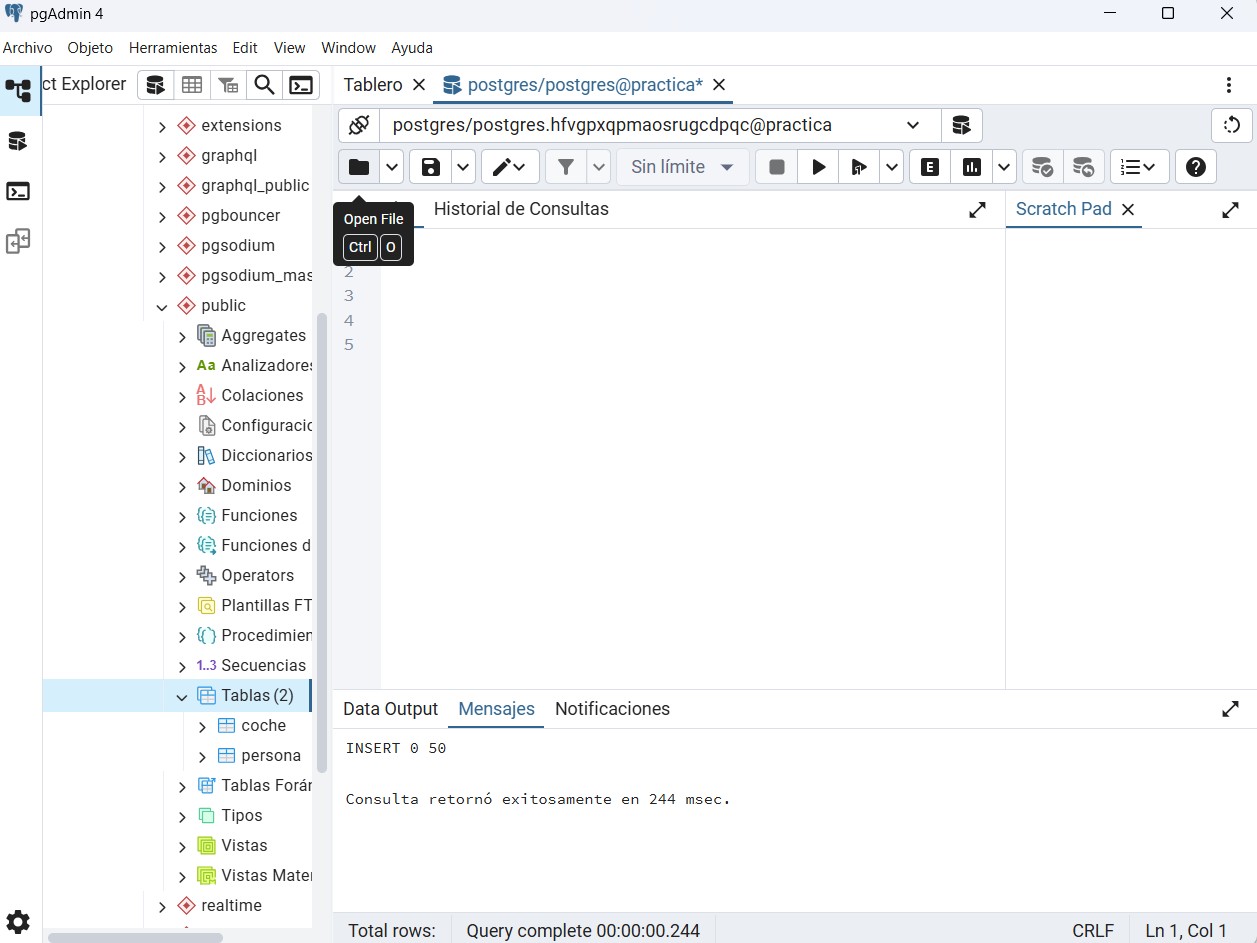
Realizaremos la creación de las tablas correspondientes

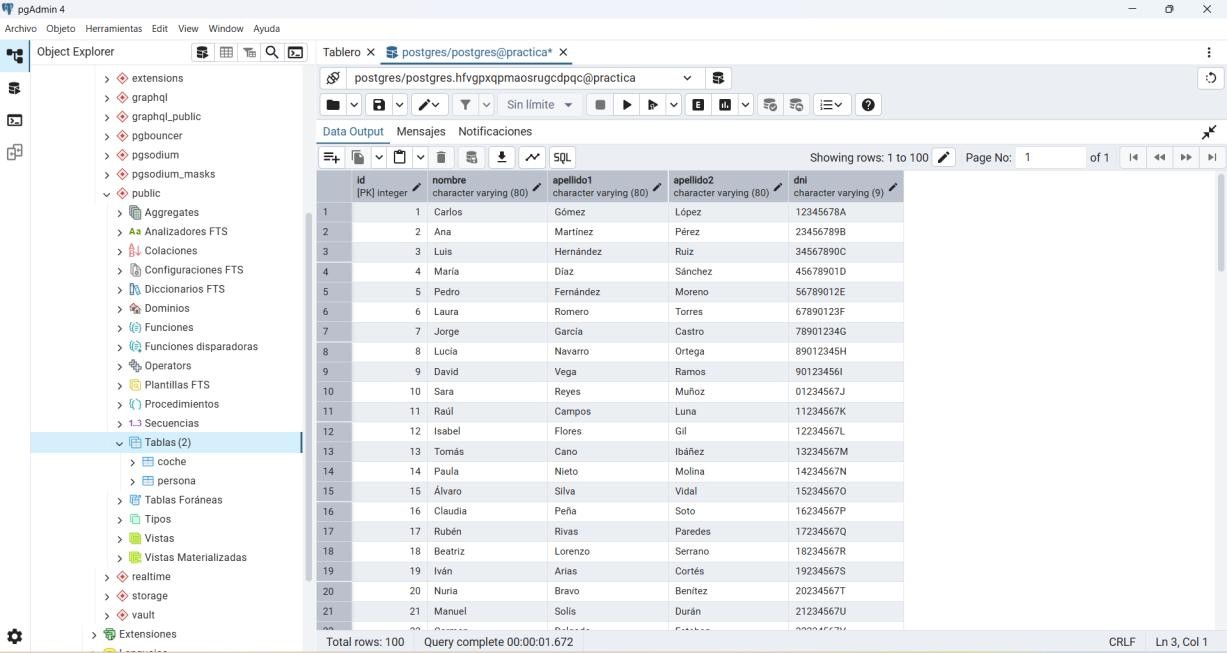


Vamos a realizar los 50 registros de cada una de las tablas



Con las tablas y los registros ya creados, nos dirigimos a la opcion esquemas, public y luego a tablas y podemos verificar que se crearon las tablas, en la misma opcion en donde se ingresaron los datos podemos realizar un SELECT \* FROME y el nombre de las tablas para verificar si se crearon los datos de manera correcta.



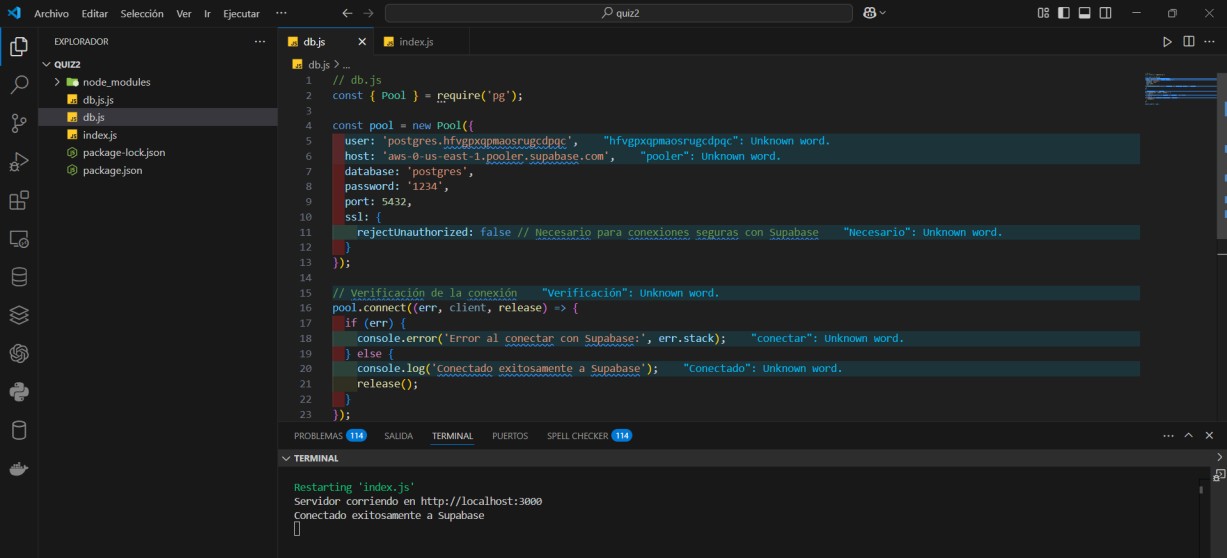


1. creación de las apis

* + El archivo db.jsse encarga de establecer la conexión entre nuestra aplicación Node.js y la base de datos PostgreSQL alojada en Supabase. Utiliza el paquete pg(PostgreSQL) y crea un "pool de conexiones" con los datos necesarios como el usuario, host, contraseña, puerto y nombre de la base de datos. Además, incluye una verificación inicial para confirmar que la conexión fue exitosa o mostrar un error si algo falla.

* + Por otro lado, el archivo index.jses el núcleo de la API creada con Express. Defina rutas para realizar operaciones CRUD (crear, leer, actualizar y eliminar) sobre tres tablas: estudiantes, cursos e inscripciones. También incluye una ruta para ver los detalles combinados de las inscripciones usando un JOIN. Usa middleware como corsy express.json() para manejar solicitudes HTTP de forma correcta, y finalmente, arranca el servidor en el puerto 3000 mostrando un mensaje cuando todo está listo.

1. Con el comando node --watch index.jspodemos ejecutar nuestra aplicación y hacer que se reinicie automáticamente cada vez que detecte cambios en el archivo index.js (o archivos relacionados). Esto es útil durante el desarrollo, ya que no tenemos que detener y volver a iniciar el servidor manualmente cada vez que modificamos el código.

 esquema supabase

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

para realizar las respectivas consultas en postman, creamos las respectivas colecciones y por medio de la url podremos ver las consultas respectivas.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.

para realizar la comprobación en postman utilizaremos las siguientes url correspondientes.

Nos dirigimos a postman para verificar que todo esté funcionando correctamente