**Creamos una base de datos en pgadmin**

**Luego de crear todo el código ejecutamos antes de hacer las pruebas con postman, el puerto es:**

[**http://localhost:5000/jsonrpc**](http://localhost:5000/jsonrpc)

* **✔️ Para insertar datos → POST**
* **✔️ Para eliminar o actualizar → POST**
* **✔️ Para consultar o listar → POST**
* **❌ Nunca GET, porque JSON-RPC usa el cuerpo JSON para definir la acción.**

**Comunicación mediante JSON-RPC**

**JSON-RPC** es una versión del modelo RPC que usa el formato **JSON** para enviar y recibir los datos.  
Cada mensaje tiene esta estructura básica:

{

"jsonrpc": "2.0",

"method": "nombre\_del\_metodo",

"params": [parametros],

"id": 1

}

* "method" → es el nombre del procedimiento remoto que se quiere ejecutar.
* "params" → son los parámetros que se le envían a esa función.
* "id" → identifica la solicitud (sirve para saber a qué petición pertenece la respuesta).
* "jsonrpc": "2.0" → indica la versión del protocolo.

**🔹 3. Funcionamiento general del proyecto**

**🔸 a) El cliente (archivo cliente\_jsonrpc.py)**

* Envía una solicitud POST al servidor, con un cuerpo JSON que incluye el método a ejecutar y los parámetros.
* Por ejemplo:
* {
* "jsonrpc": "2.0",
* "method": "insertar\_usuario",
* "params": ["Luis", 25],
* "id": 1
* }
* Ese mensaje viaja al servidor Flask en http://localhost:5000/jsonrpc.

**🔸 b) El servidor (archivo servidor\_jsonrpc.py)**

* Está desarrollado con **Flask** y expone una sola ruta /jsonrpc que recibe todas las peticiones POST.
* Cuando llega una solicitud, el servidor:
  1. Lee el JSON recibido.
  2. Extrae el nombre del método ("insertar\_usuario", "listar\_usuarios", etc.).
  3. Según el método, ejecuta una función Python que realiza la acción correspondiente:
     + insertar\_usuario: guarda un usuario en la base de datos.
     + listar\_usuarios: devuelve todos los registros.
     + eliminar\_usuario: borra un registro específico.
  4. Devuelve una respuesta JSON con el resultado o un mensaje de error, por ejemplo:
  5. {
  6. "jsonrpc": "2.0",
  7. "result": "Usuario insertado correctamente",
  8. "id": 1
  9. }

**🔸 c) Conexión con la base de datos**

* El servidor usa la librería **psycopg2** para conectarse a **PostgreSQL**.
* Cada método que necesita interactuar con la base de datos abre una conexión, ejecuta una consulta SQL (INSERT, SELECT o DELETE), y luego la cierra.
* De esta forma, el método remoto (insertar\_usuario, por ejemplo) no solo calcula algo, sino que **manipula datos reales en una base de datos.**