

## **TALLER BDM**

MIGUEL ANGEL JIMENEZ PORRAS

ID 834889

BASE DE DATOS MASIVA

WILLIAM ALEXANDER MATALLANA PORRAS

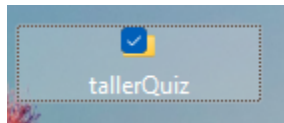
BASE DATOS MASIVAS

5/04/2025

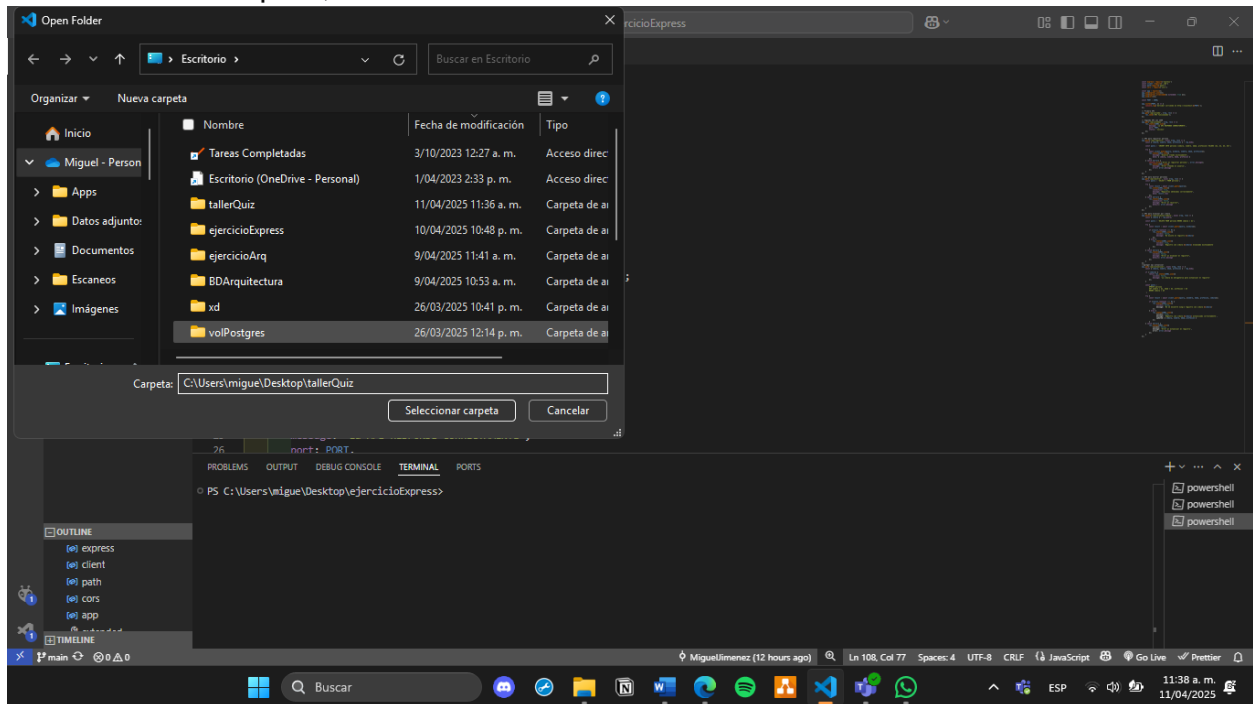
CORPORACIÓN UNIVERSITARIA MINUTO DE DIOS UNIMINUTO

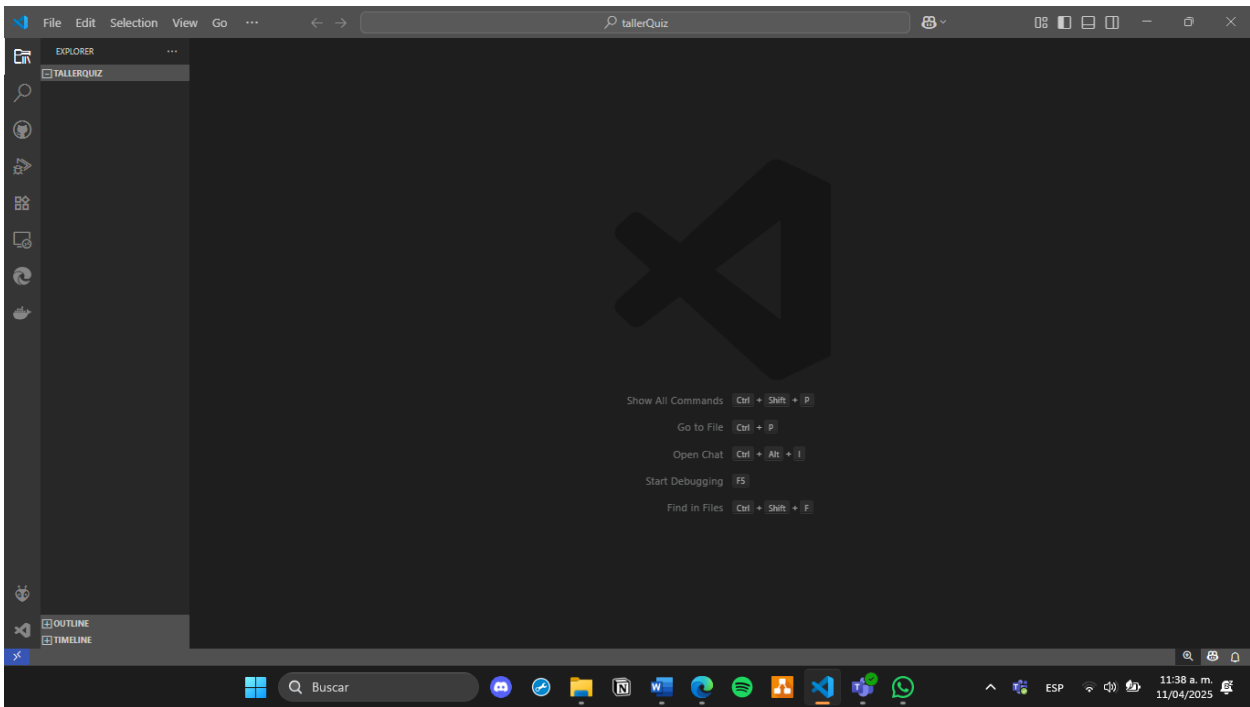
## Crear carpeta del proyecto

Para iniciar el proyecto, lo primero es iniciar la carpeta que contenga los documentos para iniciar, entonces, en mi caso inicio la carpeta, en el escritorio



Una creada la carpeta, se lleva al entorno de Visual Studio Code



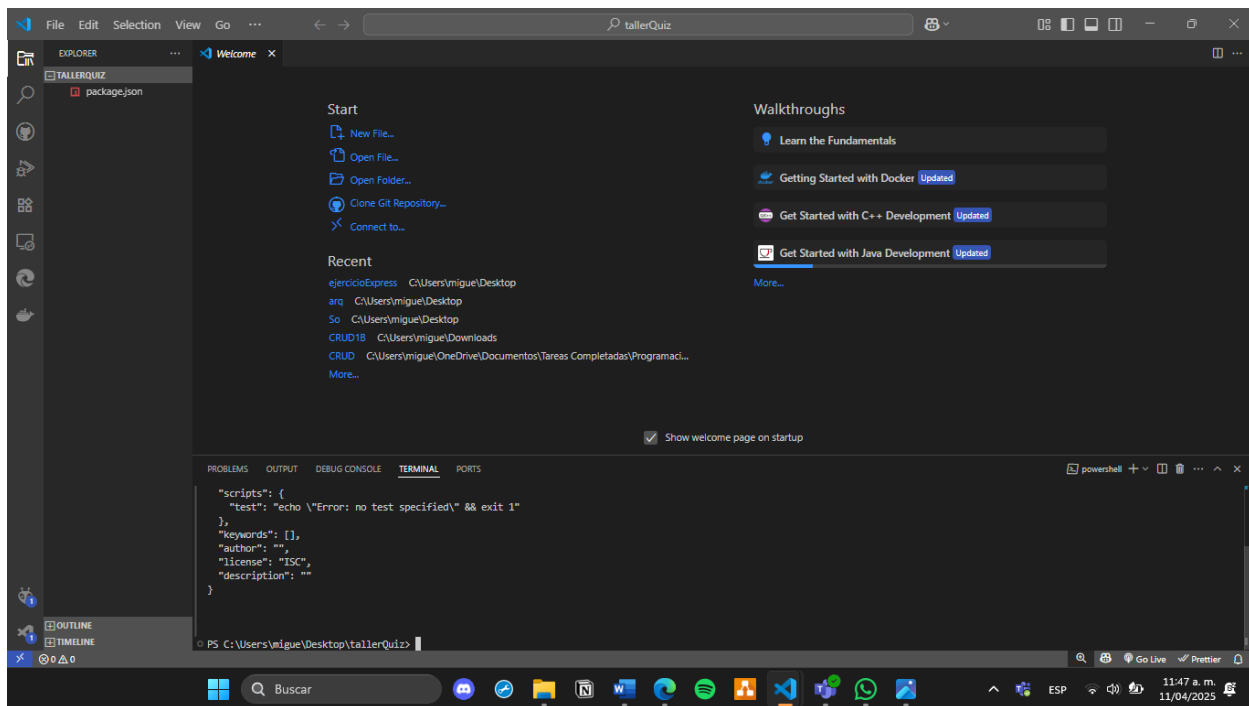


Una vez en el entorno de Visual Studio Code, podremos crear los documentos que se necesiten

### **Enlace de la carpeta con node.js**

Para enlazar la carpeta creada, en el entorno de visual studio code, en una nueva terminal, se ejecuta el comando

```
Npm init -y
```



Con el documento llamado *package.json* nos aseguramos de que quedó bien. Ahora se usarán las dependencias, con el comando

```
Npm i express pg dotenv cors
```

Y se descargarán

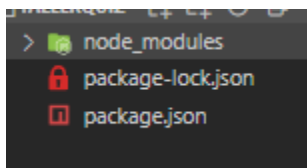
```
PS C:\Users\miguel\Desktop\tallerQuiz> Npm i express pg dotenv cors

added 83 packages, and audited 84 packages in 12s

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

found 0 vulnerabilities
PS C:\Users\miguel\Desktop\tallerQuiz>
```

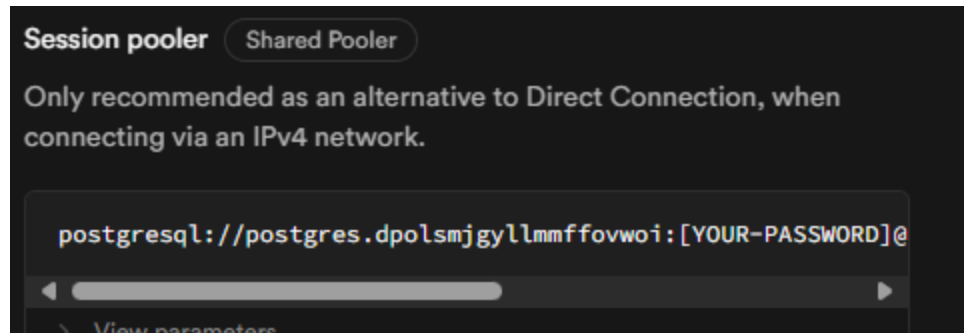
Y podremos ver que quedó creada la carpeta *node\_modules*



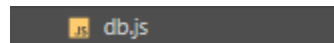
## Creación de conexión de base de datos

Para la conexión, usaré un archivo llamado *db.js*

Para ello es necesario una base de datos en Supabase, para que solo sea necesario copiar y pegar las credenciales, para ello, en supabase, creamos y obtenemos el **session pooler**, y lo copiamos en un bloc de notas, por ejemplo, nos servirá para más adelante



Ahora, en el entorno de Visual Studio Code, se crea el documento **db.js**, en mi caso, que será el puente de conexión con la base de datos



En este documento, creamos la conexión con supabase, con el **session pooler**, anteriormente mencionado,

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

const client = new Client({
  host: 'aws-0-us-east-1.pooler.supabase.com',
  port: 5432,
  user: 'postgres.dpolismjgyllmmffovwoi',
  password: 'DmDL7JLNsbqyPbs4',
  database: 'postgres'
});

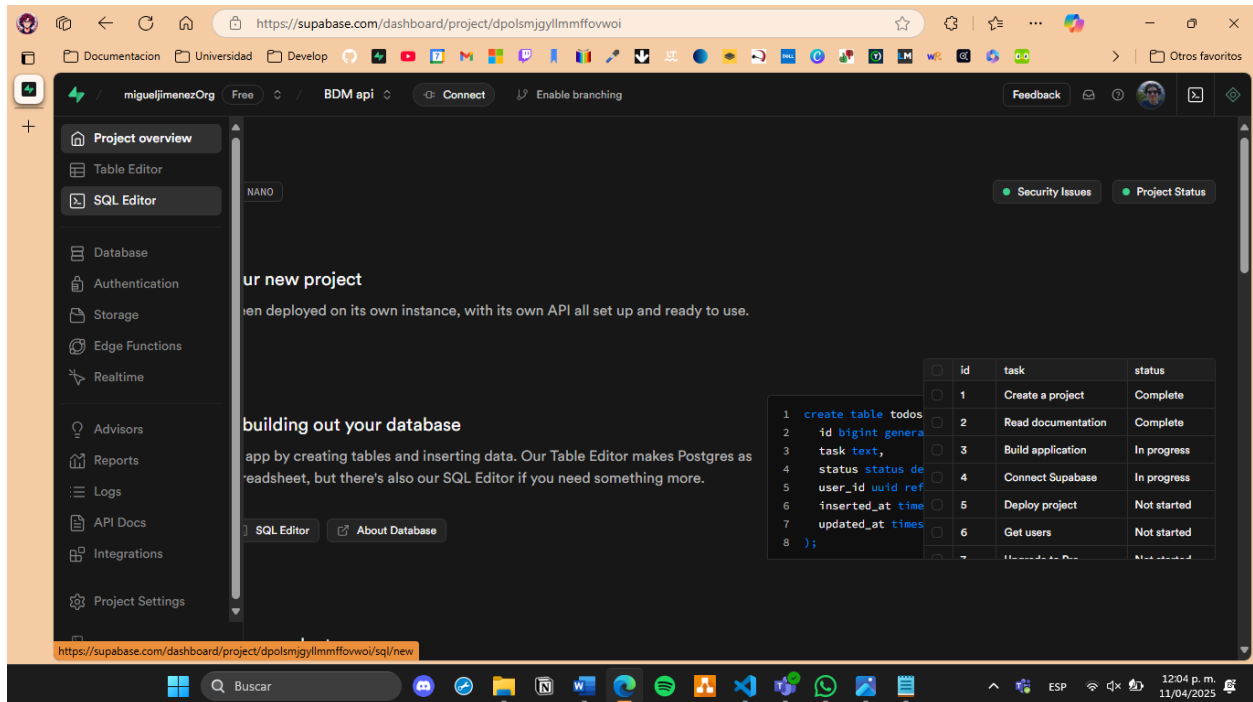
client.connect()
  .then(() => console.log('Conectado a Supabase PostgreSQL'))
  .catch(err => console.error('Error conectando a Supabase:', err.stack));

module.exports = client;
```

## Creación de las tablas requeridas

La creación de las tablas, las voy a hacer a través de supabase, para ello:

1. En el entorno del proyecto, vamos a **SQL Editor**, que nos permitirá generar las tablas,



2. En la interfaz se usará PostgreSQL, para crear la tabla Persona, usaremos:

```
1 CREATE TABLE persona (  
2   id SERIAL PRIMARY KEY,  
3   nombre VARCHAR(80),  
4   apellido1 VARCHAR(80),  
5   apellido2 VARCHAR(80),  
6   DNI VARCHAR(9)  
7 );
```

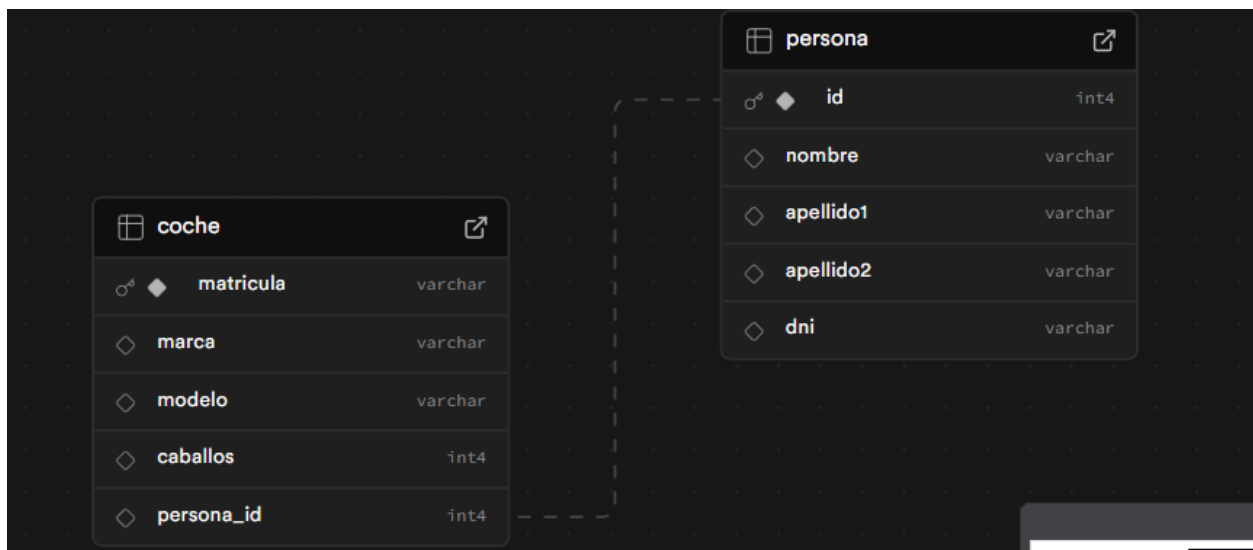
3. Ahora para crear la tabla Coche, se hará similar a el anterior caso, en la interfaz, para crear la tabla, se usará:

```

1 CREATE TABLE Coche (
2   Matricula varchar(7) PRIMARY KEY,
3   Marca VARCHAR(45),
4   Modelo VARCHAR(80),
5   Caballos integer,
6   Persona_id integer, foreign key(Persona_id) references persona(id)
7 );
8

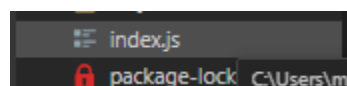
```

- Para confirmar que las tablas han quedado bien relacionadas, en Supabase, nos dirigimos a **DATABASE**, en el menú desplegable en la izquierda, y allí seleccionamos **Schema Visualizer**, y en el diagrama, podemos ver



## Creación de las Apis

En Visual Studio Code, Crearemos el archivo index.js, que será el documento que va a contener las apis.



Y en este documento, la parte mas esencial es la parte de la conexión a la base de datos, esta conexión incluye :

```

const express= require('express')
const client =require('./db')
const path=require('path')
const cors = require('cors')

const app = express();
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use(cors());

const PORT = 3000;

app.listen(PORT, () => {
  console.log(`Servidor corriendo en http://localhost:${PORT}`);
});

```

En estas líneas, se especifica la ruta de conexión(la llamé db.js), la constante **app**, que será nuestra aplicación de apis, el uso de *express*, con JSON, y El Puerto de conexión, en mi caso 3000, y finalmente un listen, que será el que nos confirme la conexión del puerto, con la conexión establecida, creamos una api de prueba, que nos confirme que todo quedó bien, para ello, ejecutamos el servicio con

```
node --watch index.js
```

En la terminal, y en la consola se verá

```

PS C:\Users\miguel\Desktop\tallerQuiz> node --watch index.js
Servidor corriendo en http://localhost:3000
Conectado a Supabase PostgreSQL

```

Confirmando, que la conexión quedó bien.

Para comenzar con las apis, primero es mejor, hacer una api de prueba, para ello, se usa el comando

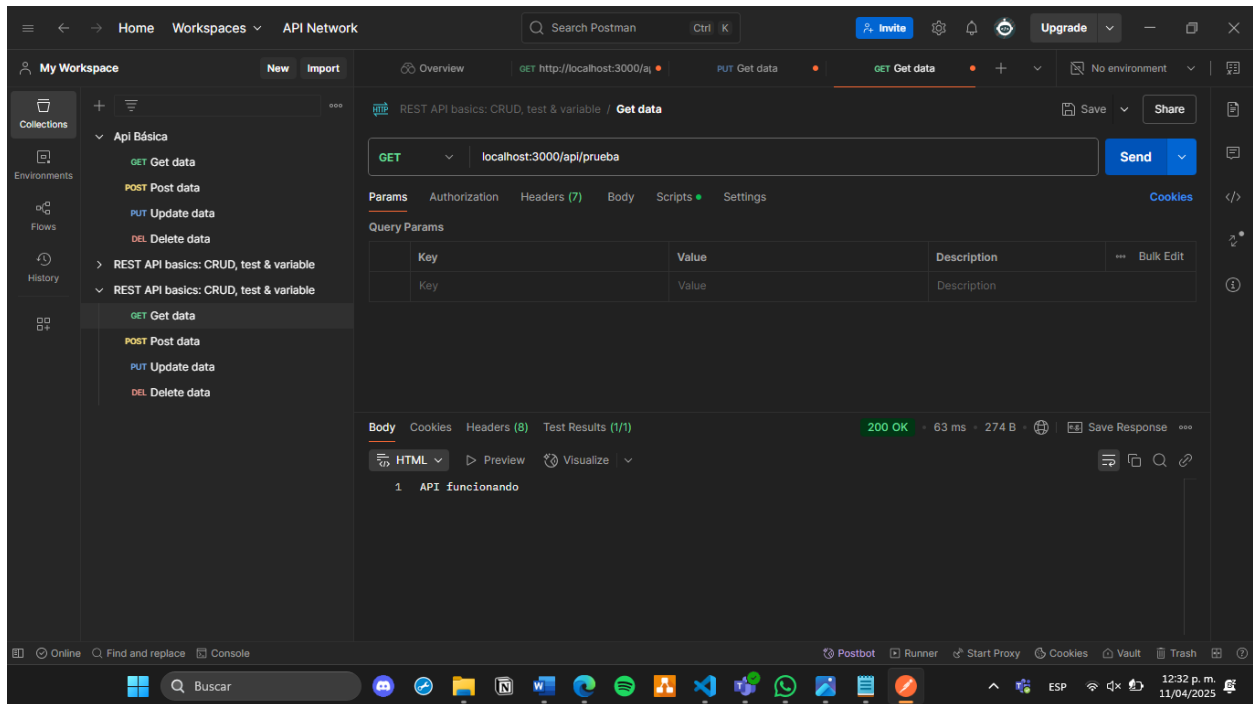
```

app.get('/api/prueba', (req, res) => {
  res.send('API funcionando');
});

```



Y en postman, en la dirección localhost:3000/api/prueba, nos deberá decir que está funcionando



## Creación de registros en las tablas

Para que funcione bien el CRUD, en las tablas, es necesario tener registros, entonces con la ayuda de ChatGPT, voy a hacer una petición para que cree 100 registros en las tablas.

Primero para la tabla persona

Ayúdame a crear registros en una tabla 'Persona' de Postgresql, con los campos id integer PK, nombre varchar, apellido1 varchar, apellido2 varchar, dni varchar. Gracias

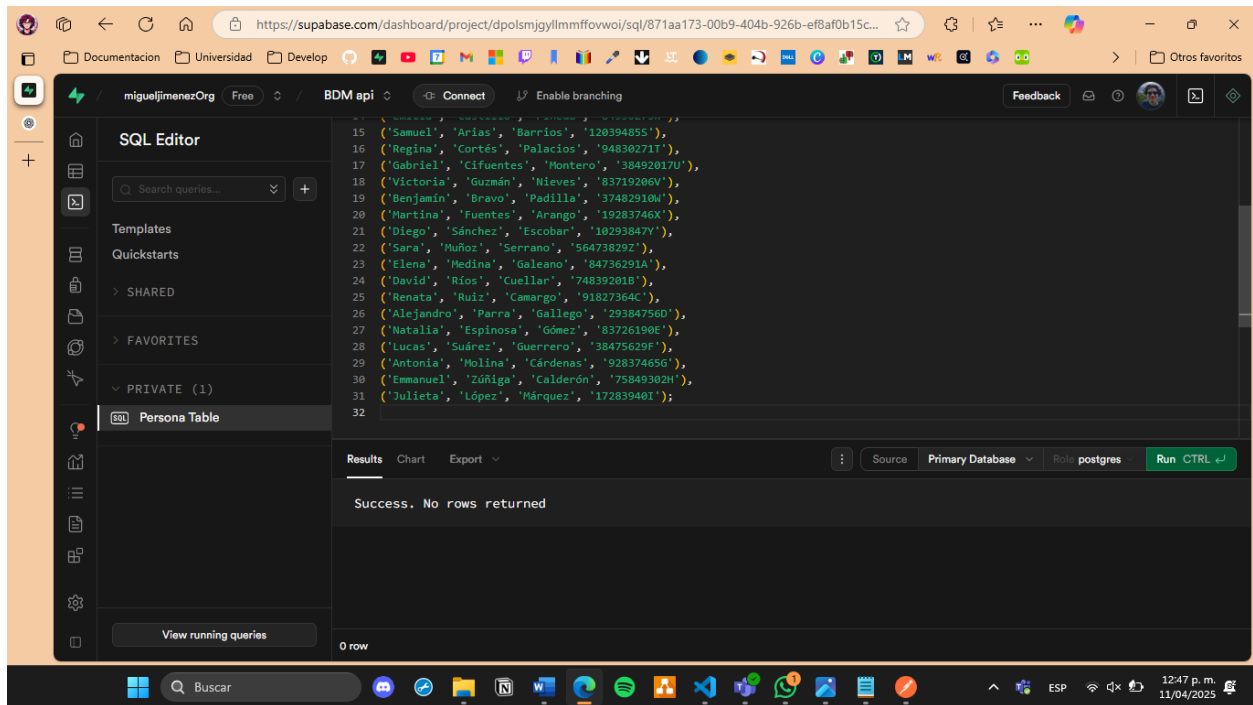
¡Claro! Aquí tienes un ejemplo de cómo crear la tabla **Persona** en PostgreSQL y cómo insertar algunos registros en ella:

### 1. Crear la tabla **Persona**

```
sql

CREATE TABLE Persona (
  id SERIAL PRIMARY KEY,
  nombre VARCHAR(50),
  apellido1 VARCHAR(50),
  apellido2 VARCHAR(50),
  dni VARCHAR(20)
);
```

Y en supabase, se crearan los registros



The screenshot shows the Supabase SQL Editor interface. The left sidebar displays the project structure with 'Persona Table' selected. The main editor area contains a list of 17 records inserted into the 'Persona' table. The records are as follows:

id	nombre	apellido1	apellido2	dni
15	Samuel	Arias	Barrios	120394855
16	Regina	Cortés	Palacios	94830271T
17	Gabriel	Cifuentes	Montero	38492017U
18	Victoria	Guzmán	Nieves	83719206V
19	Benjamin	Bravo	Padilla	37482910W
20	Martina	Fuentes	Arango	19283746X
21	Diego	Sánchez	Escobar	10293847Y
22	Sara	Muñoz	Serrano	56473829Z
23	Elena	Medina	Galeano	84736291A
24	David	Ríos	Cuellar	74839201B
25	Renata	Ruiz	Camargo	91827364C
26	Alejandro	Parra	Gallego	29384756D
27	Natalia	Espinosa	Gómez	83726190E
28	Lucas	Suárez	Guerrero	38475629F
29	Antonia	Molina	Cárdenas	92837465G
30	Emmanuel	Zúñiga	Calderón	75849302H
31	Julietta	López	Márquez	17283940I

The bottom of the editor shows the 'Results' tab with the message 'Success. No rows returned' and a 'Run' button.