**Apis postgres**

**Dylan Felipe Avellaneda Garcia**

**Id:862216**

**Ing. William Alexander Matallana Porras**

**Corporación Universitaria Minuto de Dios**

**ingeniería de Sistemas**

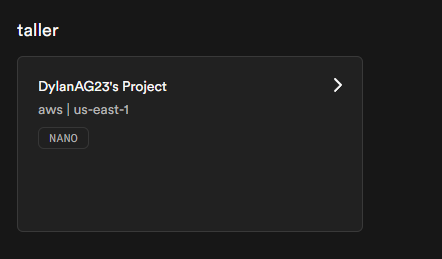
**Bases de Datos Masivas**

**NRC:60747**

**Zipaquirá, 11 de abril de 2025**

**Comandos Docker**

1. **Creamos una base de datos en supabase**

****

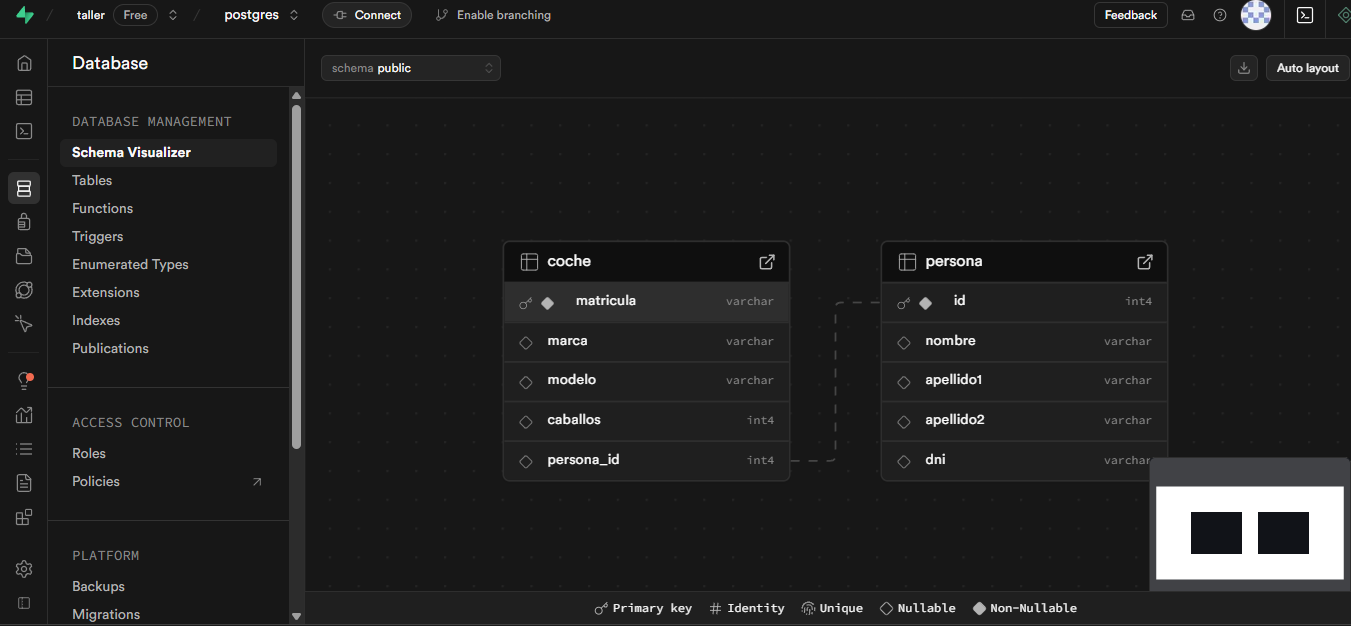
1. **Creamos las tablas de nuestra base de datos según el uml dado**

**Interfaz de usuario gráfica, Texto, Sitio web

El contenido generado por IA puede ser incorrecto.**

**Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.**

****

1. **Ya creadas las tablas insertamos los registros de cada tabla para terminar nuestra base de datos.**

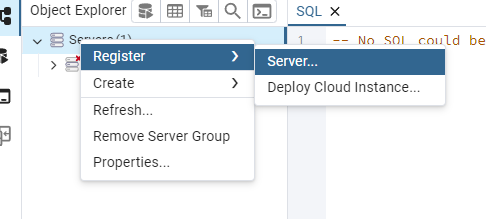
**Captura de pantalla de computadora

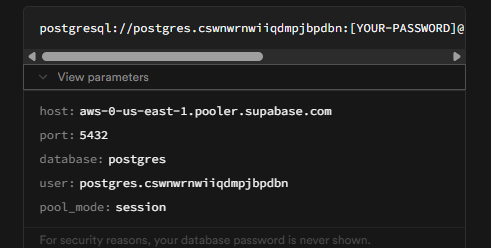
El contenido generado por IA puede ser incorrecto.**

**Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.**

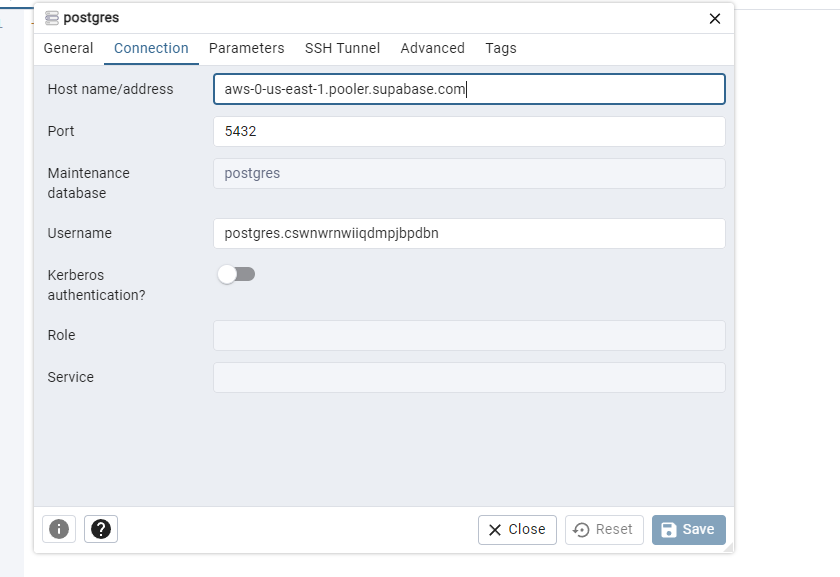
1. **Procedemos a hacer la conexión entre supabase y pgadmin, luego verificamos que este correctamente y que se visualicen las tablas que creamos desde supabase.**

****

****

**Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.**



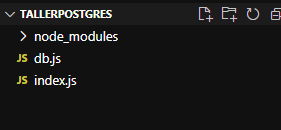
Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Aplicación

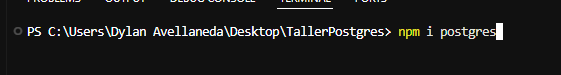
El contenido generado por IA puede ser incorrecto.

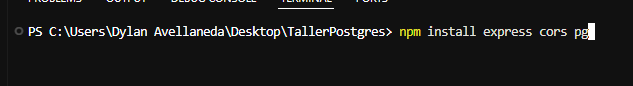
1. **Ahora creamos una nueva carpeta para hacer el archivo de conexión a la base de datos y el respectivo index y enviando la carpeta al vscode.**

****

1. **Hacemos la instalación de las librerías que vamos a usar.**

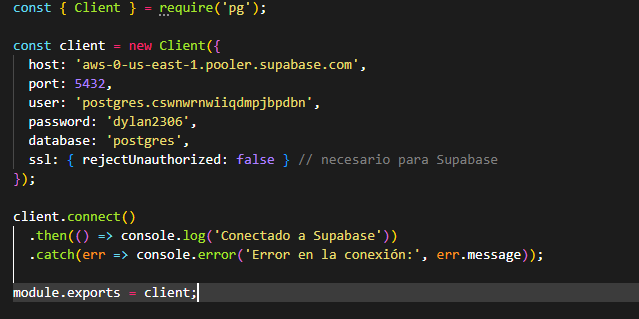
****

****

****

1. **Empezamos con la conexión a la base de datos con los mismos parámetros de supabase.**

* En este apartado creamos la variable cliente que va a usar la librerio de pg para poderse conectar con nuestro puente a la base de datos en supabase que exactamente es pgadmin4.
* Para la conexión colocamos los mismos parámetros de conexión de pusimos para la conexión de supabase con pgadmin incluyendo el puerto.

****

1. **Creamos el index y las respectivas apis para cada ruta tanto la tabla de personas como la de coches.**

const express = require('express');

const cors = require('cors');

const client = require('./db');

const app = express();

const PORT = 3000;

app.use(cors());

app.use(express.json());

app.use(express.urlencoded({ extended: true }));

// Ruta de prueba

app.get('/api/prueba', (req, res) => {

res.status(200).json({

message: 'API FUNCIONANDO CORRECTAMENTE',

port: PORT,

status: 'success'

});

});

// ====================

// RUTAS PARA PERSONAS

// ====================

// Obtener todas las personas

app.get('/api/personas', async (req, res) => {

try {

const result = await client.query('SELECT \* FROM Persona');

res.status(200).json(result.rows);

} catch (error) {

res.status(500).json({ message: 'Error al obtener personas', error: error.message });

}

});

// Crear una persona

app.post('/api/personas', async (req, res) => {

const { id, Nombre, Apellido1, Apellido2, DNI } = req.body;

const query = `INSERT INTO Persona (id, Nombre, Apellido1, Apellido2, DNI) VALUES ($1, $2, $3, $4, $5)`;

try {

await client.query(query, [id, Nombre, Apellido1, Apellido2, DNI]);

res.status(201).json({ id, Nombre, Apellido1, Apellido2, DNI });

} catch (error) {

res.status(500).json({ message: 'Error al crear persona', error: error.message });

}

});

// Actualizar una persona

app.put('/api/personas/:id', async (req, res) => {

const { id } = req.params;

const { Nombre, Apellido1, Apellido2, DNI } = req.body;

const query = `

UPDATE Persona

SET Nombre = $1, Apellido1 = $2, Apellido2 = $3, DNI = $4

WHERE id = $5

`;

try {

const result = await client.query(query, [Nombre, Apellido1, Apellido2, DNI, id]);

if (result.rowCount === 0) {

return res.status(404).json({ message: 'Persona no encontrada' });

}

res.status(200).json({ message: 'Persona actualizada correctamente' });

} catch (error) {

res.status(500).json({ message: 'Error al actualizar persona', error: error.message });

}

});

// Eliminar una persona

app.delete('/api/personas/:id', async (req, res) => {

const { id } = req.params;

try {

const result = await client.query('DELETE FROM Persona WHERE id = $1', [id]);

if (result.rowCount === 0) {

return res.status(404).json({ message: 'Persona no encontrada' });

}

res.status(200).json({ message: 'Persona eliminada correctamente' });

} catch (error) {

res.status(500).json({ message: 'Error al eliminar persona', error: error.message });

}

});

// ====================

// RUTAS PARA COCHES

// ====================

// Obtener todos los coches

app.get('/api/coches', async (req, res) => {

try {

const result = await client.query('SELECT \* FROM Coche');

res.status(200).json(result.rows);

} catch (error) {

res.status(500).json({ message: 'Error al obtener coches', error: error.message });

}

});

// Crear un coche

app.post('/api/coches', async (req, res) => {

const { Matricula, Marca, Modelo, Caballos, Persona\_id } = req.body;

const query = `

INSERT INTO Coche (Matricula, Marca, Modelo, Caballos, Persona\_id)

VALUES ($1, $2, $3, $4, $5)

`;

try {

await client.query(query, [Matricula, Marca, Modelo, Caballos, Persona\_id]);

res.status(201).json({ Matricula, Marca, Modelo, Caballos, Persona\_id });

} catch (error) {

res.status(500).json({ message: 'Error al crear coche', error: error.message });

}

});

// Actualizar un coche

app.put('/api/coches/:matricula', async (req, res) => {

const { matricula } = req.params;

const { Marca, Modelo, Caballos, Persona\_id } = req.body;

const query = `

UPDATE Coche

SET Marca = $1, Modelo = $2, Caballos = $3, Persona\_id = $4

WHERE Matricula = $5

`;

try {

const result = await client.query(query, [Marca, Modelo, Caballos, Persona\_id, matricula]);

if (result.rowCount === 0) {

return res.status(404).json({ message: 'Coche no encontrado' });

}

res.status(200).json({ message: 'Coche actualizado correctamente' });

} catch (error) {

res.status(500).json({ message: 'Error al actualizar coche', error: error.message });

}

});

// Eliminar un coche

app.delete('/api/coches/:matricula', async (req, res) => {

const { matricula } = req.params;

try {

const result = await client.query('DELETE FROM Coche WHERE Matricula = $1', [matricula]);

if (result.rowCount === 0) {

return res.status(404).json({ message: 'Coche no encontrado' });

}

res.status(200).json({ message: 'Coche eliminado correctamente' });

} catch (error) {

res.status(500).json({ message: 'Error al eliminar coche', error: error.message });

}

});

app.listen(PORT, () => {

console.log(`Servidor corriendo en puerto ${PORT}`);

});

1. **Vemos que la conexión se estableció de manera correcta.**

**Captura de pantalla de un celular

El contenido generado por IA puede ser incorrecto.**