

Quiz Bases de Datos

Juan Sebastian Dueñas Robayo

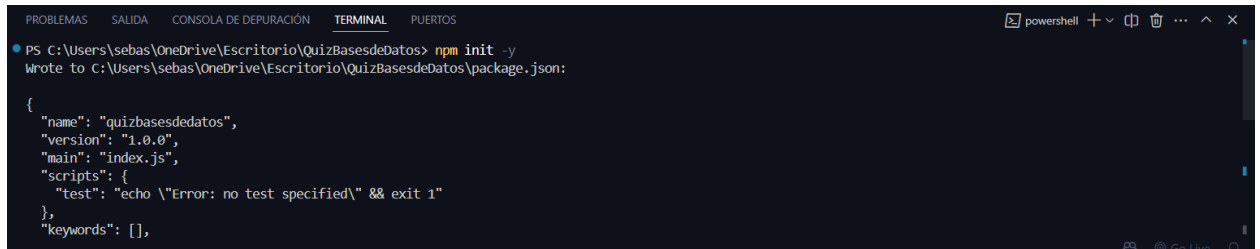
833539

Corporación Universitaria Minuto de Dios

Ing. William Alexander Matallana Porras

10-60747: Bases de datos masivas

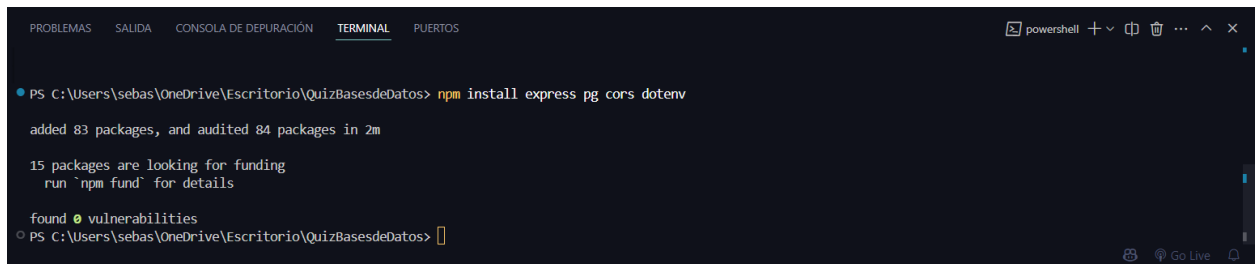
1. Creamos una nueva carpeta donde desemos, dentro de esta misma con nuestro Visual Studio Code, abriremos un terminal e inicializaremos nuestro package.json con el siguiente comando:



```
PS C:\Users\sebas\OneDrive\Escritorio\QuizBasesdeDatos> npm init -y
Wrote to C:\Users\sebas\OneDrive\Escritorio\QuizBasesdeDatos\package.json:

{
  "name": "quizbasesdedatos",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
```

2. Esta no creará nuestro. json, lo siguiente por hacer es instalar el express con postgres para hacer nuestra conexión a supabase, con el siguiente comando instalaremos los node_modules:

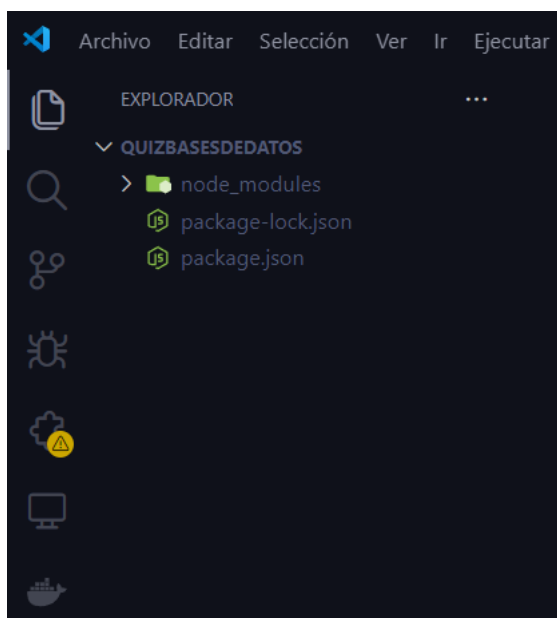


```
PS C:\Users\sebas\OneDrive\Escritorio\QuizBasesdeDatos> npm install express pg cors dotenv
added 83 packages, and audited 84 packages in 2m

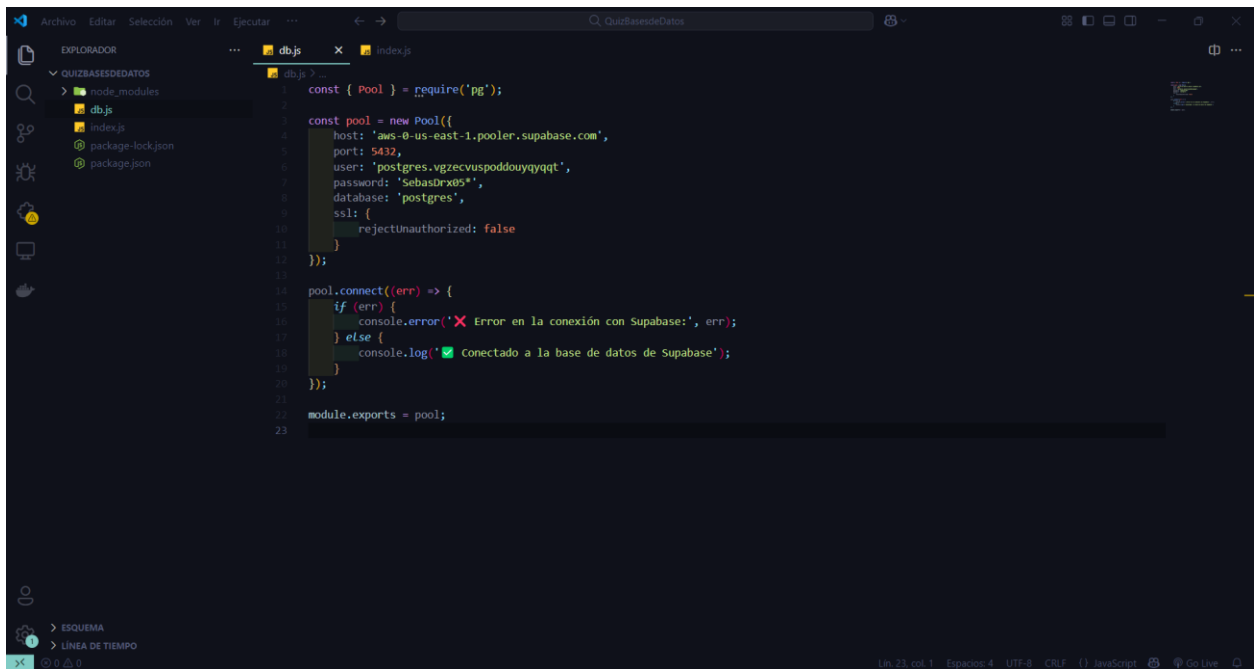
15 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\sebas\OneDrive\Escritorio\QuizBasesdeDatos>
```

3. Con todo instalado se nos tendría que ver de esta manera:

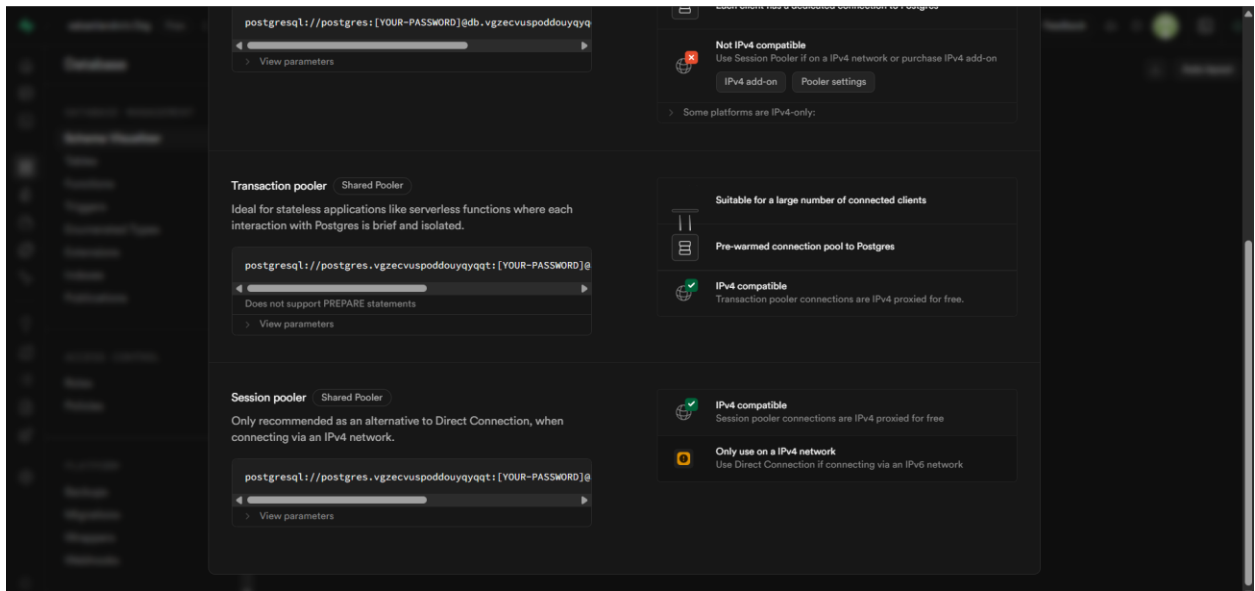


4. Aquí ya empezaremos a trabajar con nuestro db.js que contendrá todos nuestros datos de conexión y con el index que contendrá nuestras apis
5. Empezamos con nuestro db.js, como se mencionaba anteriormente este contiene toda la información necesaria para la conexión local con supabase, con todo esto ajustado se vería de la siguiente manera:



```
1  const { Pool } = require('pg');
2
3  const pool = new Pool({
4    host: 'aws-0-us-east-1.pooler.supabase.com',
5    port: 5432,
6    user: 'postgres.vgzcwspoddouyqyqt',
7    password: 'SebasDrx05',
8    database: 'postgres',
9    ssl: {
10      rejectUnauthorized: false
11    }
12  });
13
14  pool.connect((err) => {
15    if (err) {
16      console.error('❌ Error en la conexión con Supabase:', err);
17    } else {
18      console.log('✅ Conectado a la base de datos de Supabase');
19    }
20  });
21
22  module.exports = pool;
23
```

Siendo el host, el user, el puerto y la base de datos lo que nos proporciona supabase en el apartado de conexión, ya la contraseña va por cuenta de nosotros que se adjuntó al momento de crear el proyecto, o si queremos igualmente supabase nos proporciona una aleatoria.



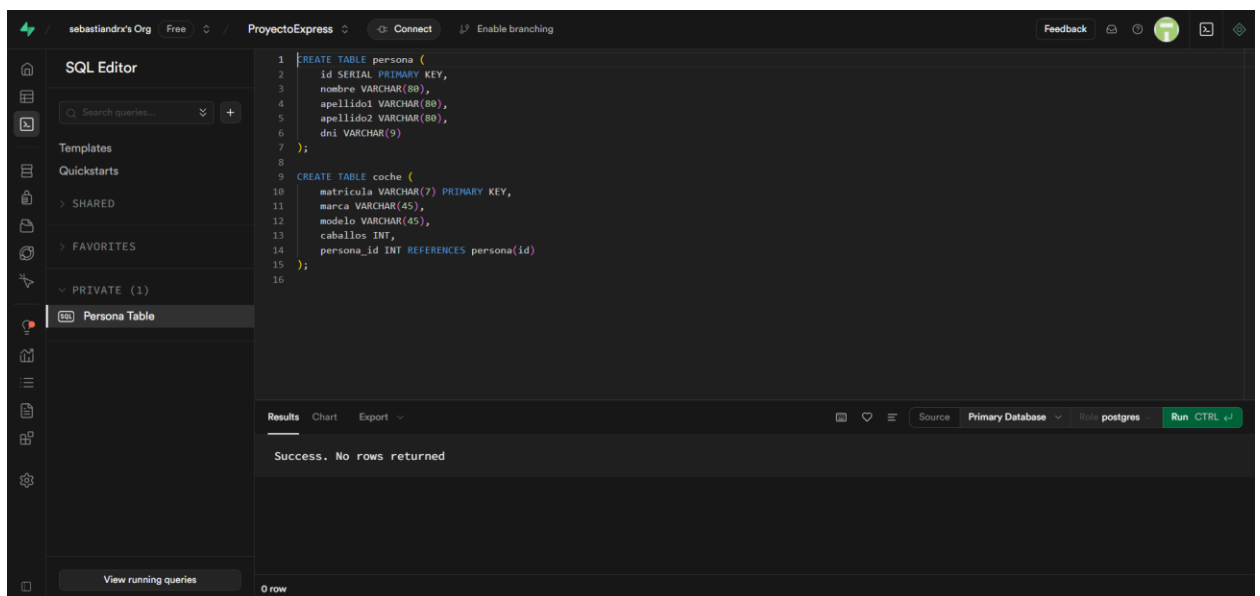
Por defecto para el tema de conectarnos a supabase usaremos siempre el **SESSION**

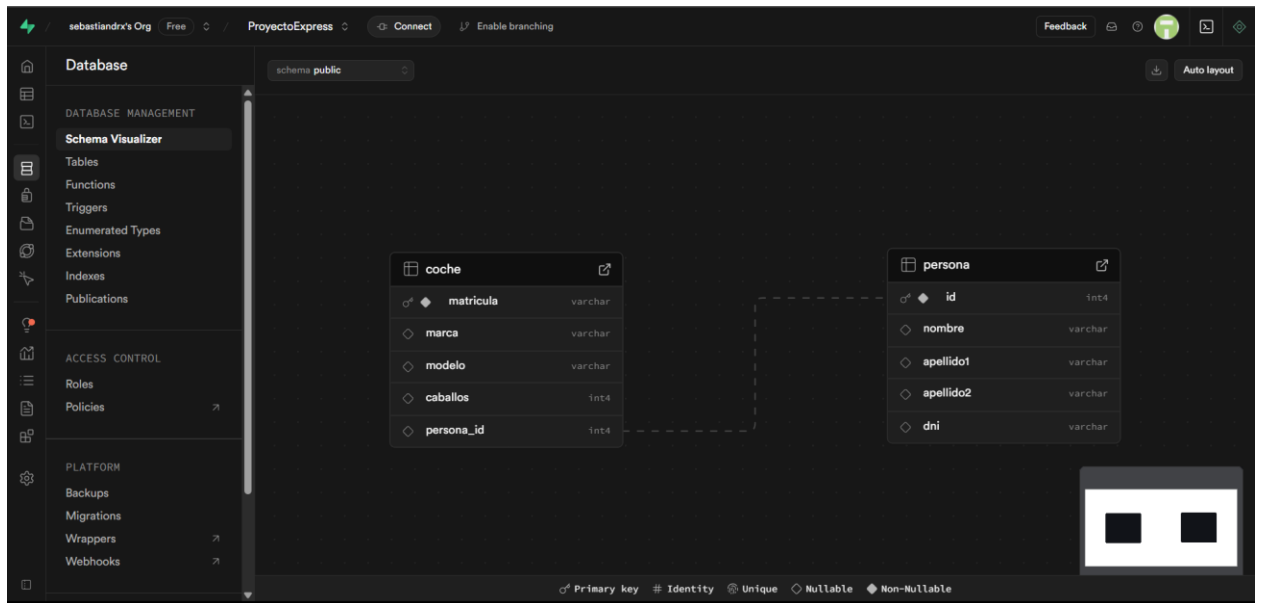
POOLER: En este caso nos brindó el siguiente enlace

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

El host siempre será la parte señalada en color rojo y nuestro usuario la parte señalada en color azul

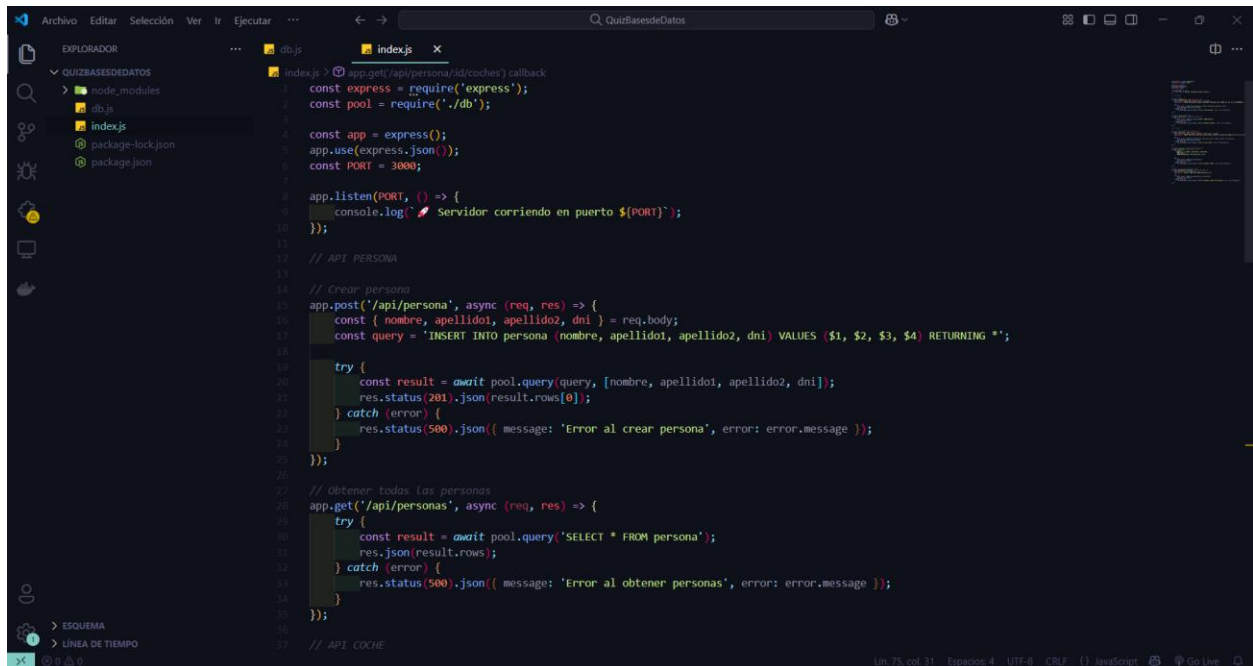
6. Antes de manipular nuestro index.js, en supabase específicamente en el apartado de **SQL EDITOR**, crearemos las dos tablas y les insertaremos 100 registros a cada una:



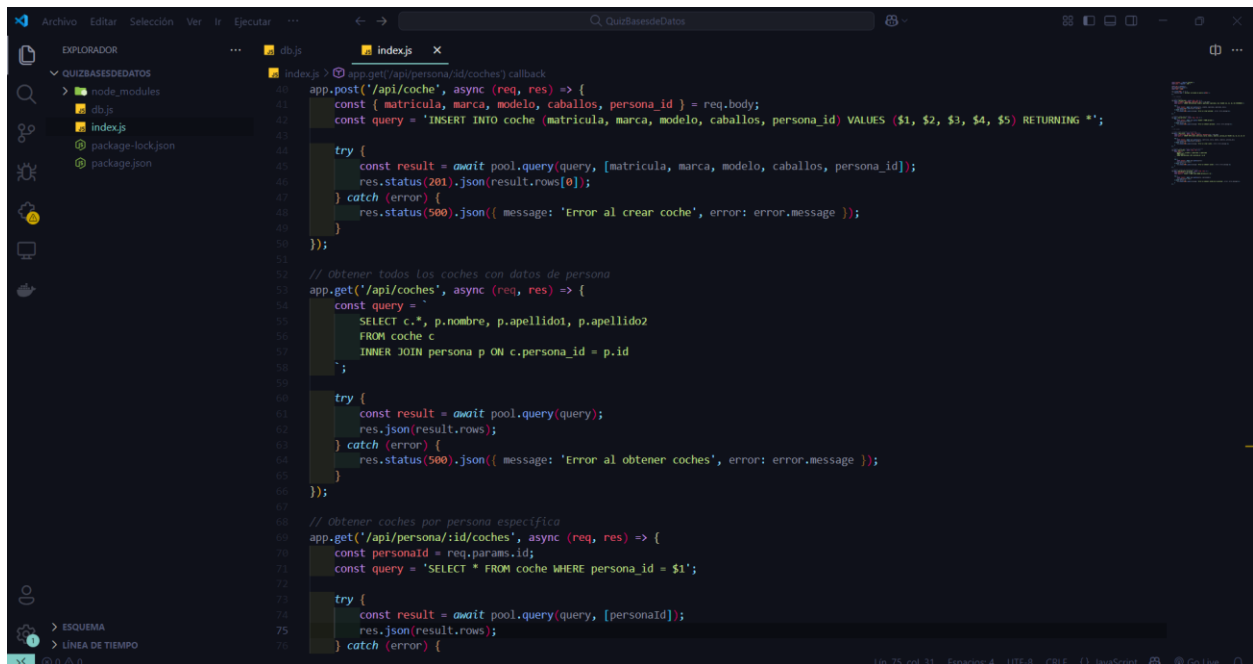


7. Luego de esto en el mismo apartado de **SQL EDITOR**, insertaremos los registros

8. Luego de esto creamos el index.js con todas nuestras API'S



```
1  index.js > app.get('/api/persona/:id/coches') callback
2  const express = require('express');
3  const pool = require('./db');
4
5  const app = express();
6  app.use(express.json());
7  const PORT = 3000;
8
9  app.listen(PORT, () => {
10     console.log(`Servidor corriendo en puerto ${PORT}`);
11 });
12
13 // API PERSONA
14
15 // Crear persona
16 app.post('/api/persona', async (req, res) => {
17     const { nombre, apellido1, apellido2, dni } = req.body;
18     const query = 'INSERT INTO persona (nombre, apellido1, apellido2, dni) VALUES ($1, $2, $3, $4) RETURNING *';
19
20     try {
21         const result = await pool.query(query, [nombre, apellido1, apellido2, dni]);
22         res.status(201).json(result.rows[0]);
23     } catch (error) {
24         res.status(500).json({ message: 'Error al crear persona', error: error.message });
25     }
26 });
27
28 // Obtener todas las personas
29 app.get('/api/personas', async (req, res) => {
30     try {
31         const result = await pool.query('SELECT * FROM persona');
32         res.json(result.rows);
33     } catch (error) {
34         res.status(500).json({ message: 'Error al obtener personas', error: error.message });
35     }
36 });
37
38 // API COCHE
```

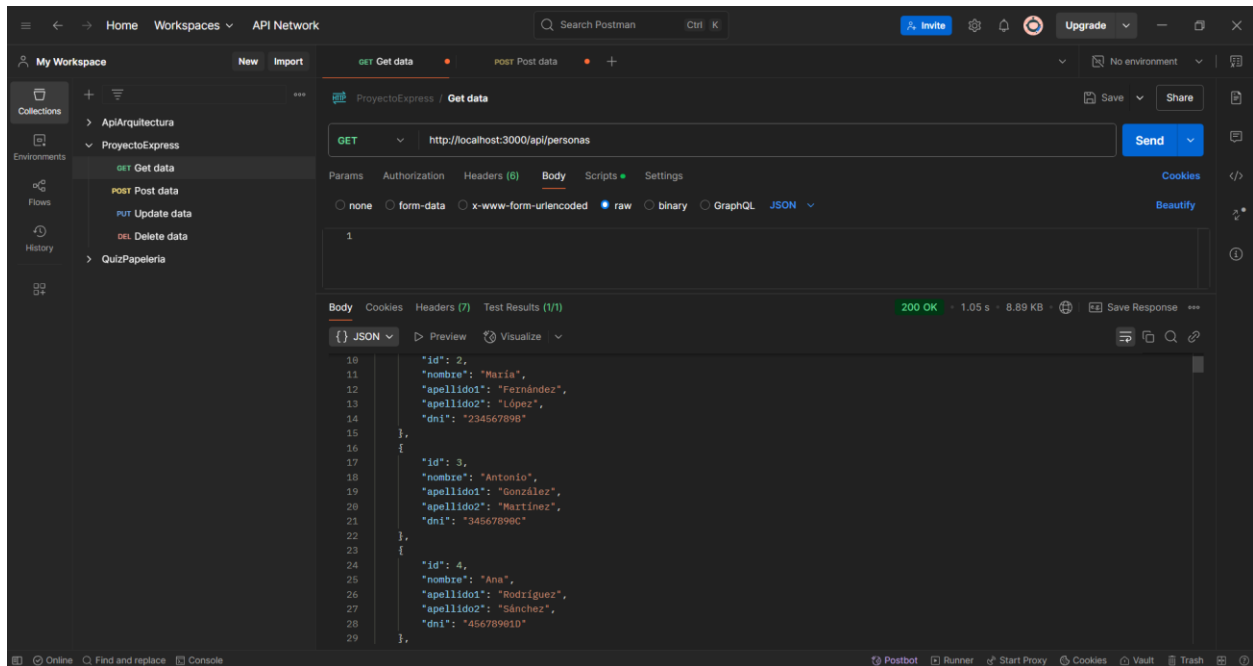


```
40  index.js > app.get('/api/persona/:id/coches') callback
41  app.post('/api/coche', async (req, res) => {
42     const { matricula, marca, modelo, caballos, persona_id } = req.body;
43     const query = 'INSERT INTO coche (matricula, marca, modelo, caballos, persona_id) VALUES ($1, $2, $3, $4, $5) RETURNING *';
44
45     try {
46         const result = await pool.query(query, [matricula, marca, modelo, caballos, persona_id]);
47         res.status(201).json(result.rows[0]);
48     } catch (error) {
49         res.status(500).json({ message: 'Error al crear coche', error: error.message });
50     }
51 });
52
53 // Obtener todos los coches con datos de persona
54 app.get('/api/coches', async (req, res) => {
55     const query = `
56     SELECT c.*, p.nombre, p.apellido1, p.apellido2
57     FROM coche c
58     INNER JOIN persona p ON c.persona_id = p.id
59     `;
60
61     try {
62         const result = await pool.query(query);
63         res.json(result.rows);
64     } catch (error) {
65         res.status(500).json({ message: 'Error al obtener coches', error: error.message });
66     }
67 });
68
69 // Obtener coches por persona específica
70 app.get('/api/persona/:id/coches', async (req, res) => {
71     const personId = req.params.id;
72     const query = 'SELECT * FROM coche WHERE persona_id = $1';
73
74     try {
75         const result = await pool.query(query, [personId]);
76         res.json(result.rows);
77     } catch (error) {
78         res.status(500).json({ message: 'Error al obtener coches', error: error.message });
79     }
80 });
```

9. Ya con todo esto listo probamos en el postman:

- Si queremos consultar todas las personas en el GET: colocamos la siguiente URL:

<http://localhost:3000/api/personas>



- Si queremos filtrar la búsqueda de coches por personas en el método GET con la siguiente URL lo podremos hacer: <http://localhost:3000/api/personas>

