Ejercicio Supabase

Andrés Felipe Cholo González

ID: 000832608

Base De Datos Masivas

Ing. William Alexander Matallana Porras

NRC: 10-60747

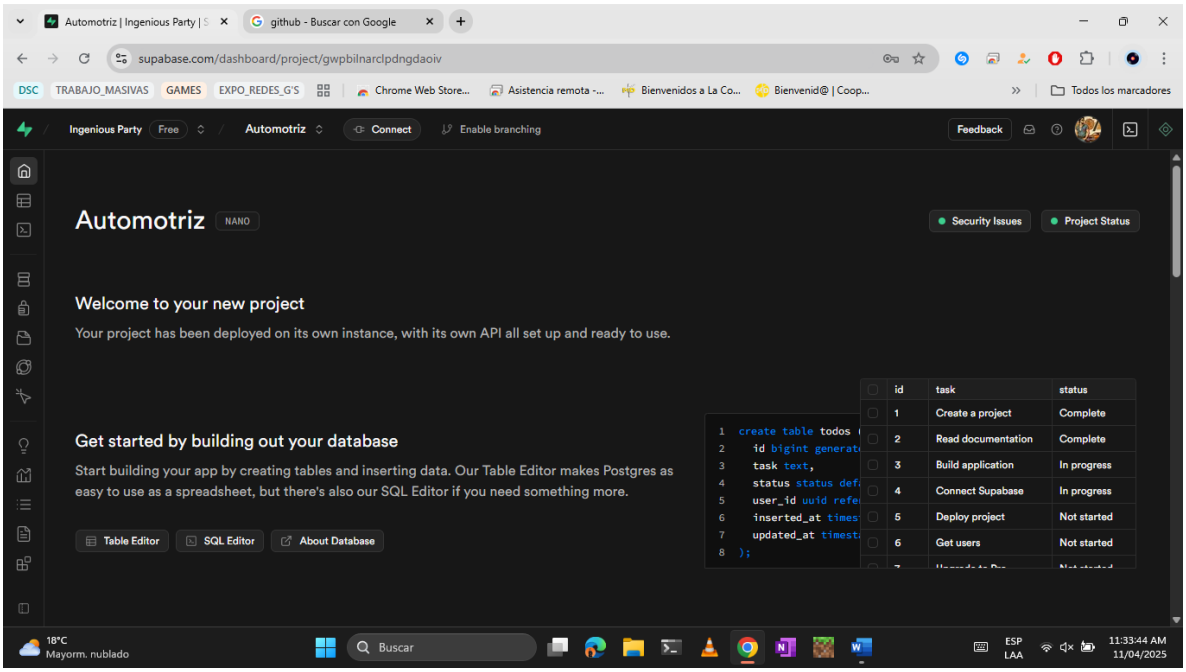
Escuela De Ingeniería, Ingeniería De Sistemas

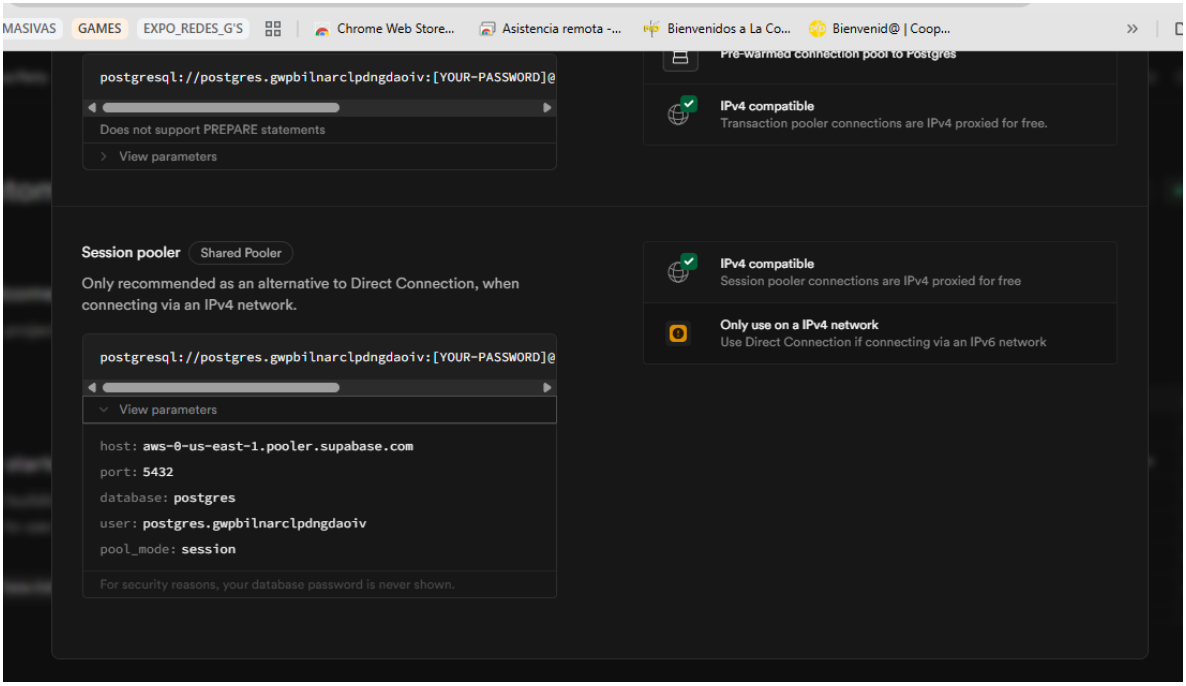
Sexto Semestre, Segundo Corte

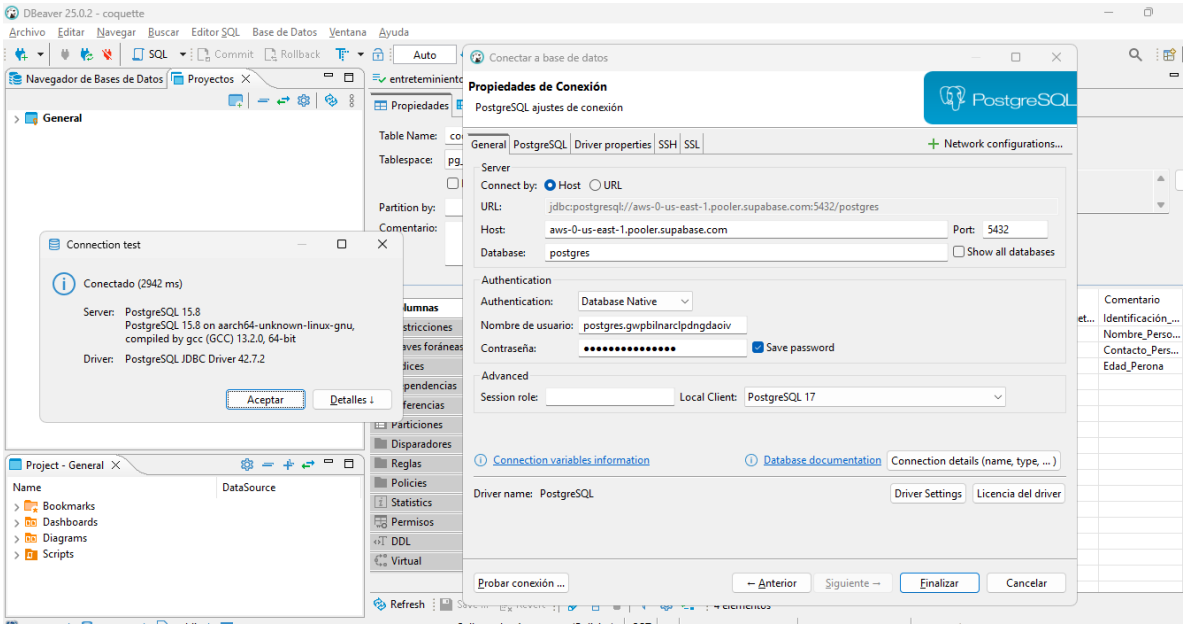
Corporación Universitaria Minuto De Dios, Sede Zipaquirá

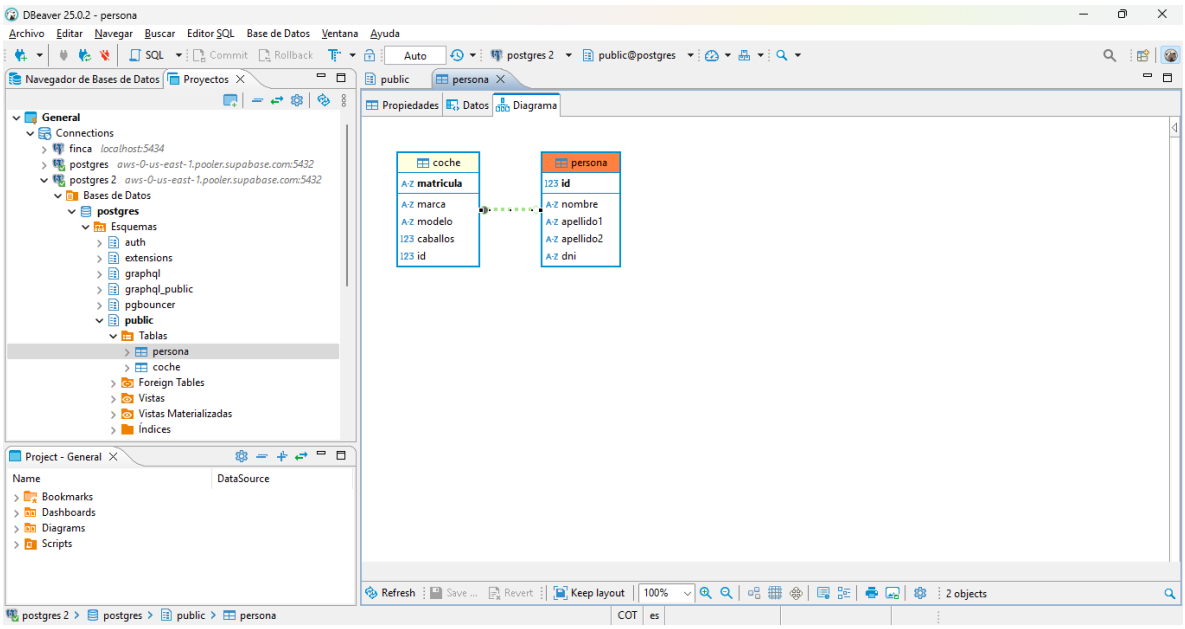
Viernes, 11 de Abril, 2025

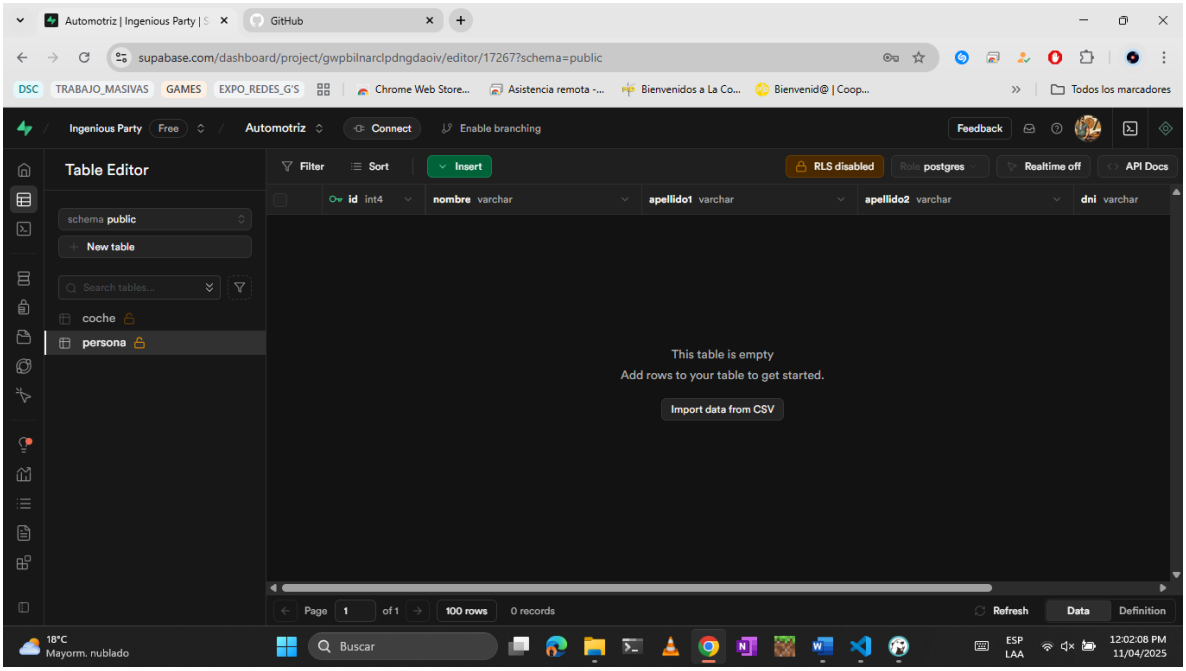
Generamos en supabase un nuevo proyecto

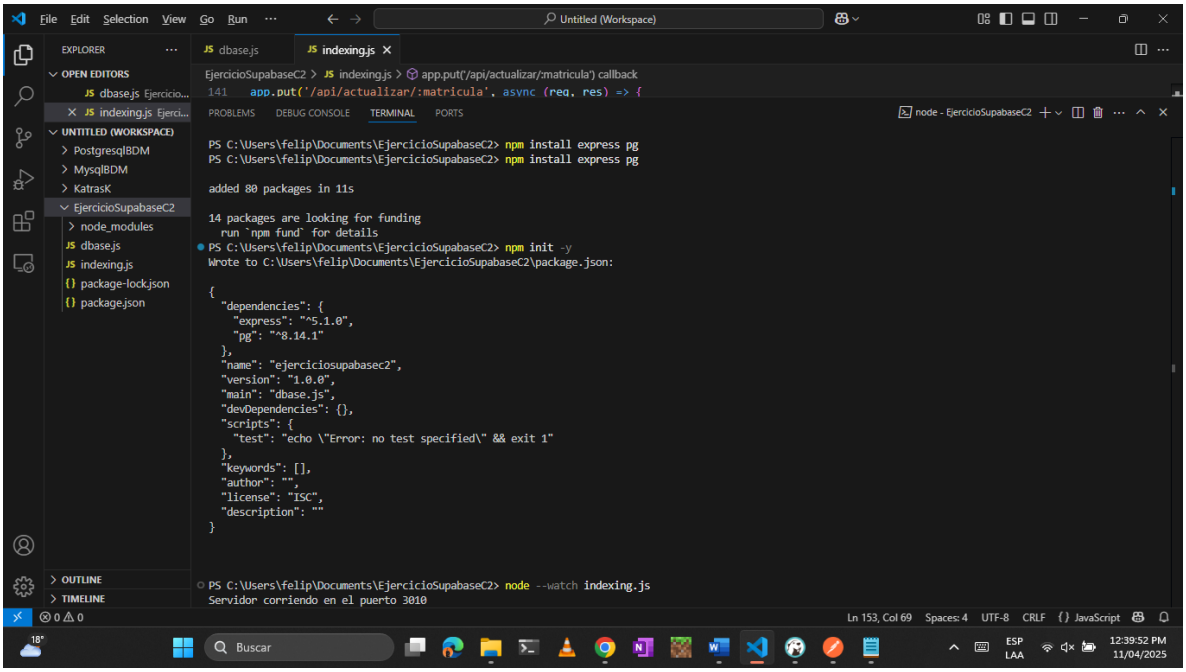


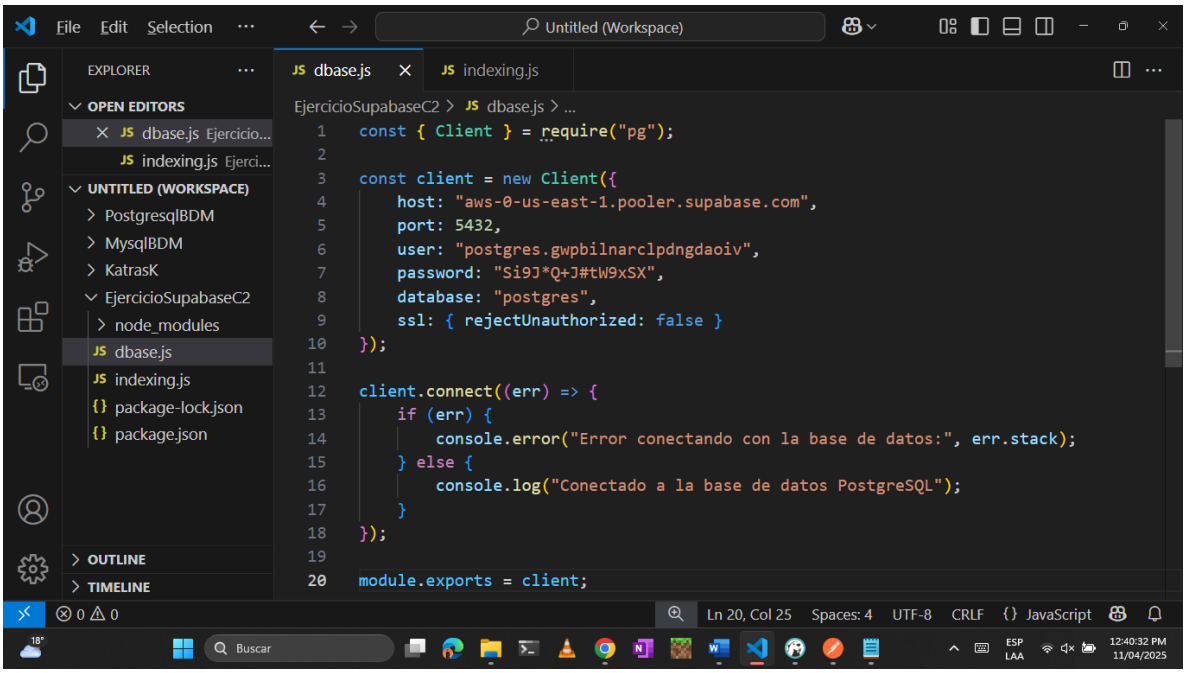



DBeaver 25.0.2 - coquette
Archivo Editar Navegar Buscar Editor SQL Base de Datos Ventana Ayuda
Navegador de Bases de Datos
Proyectos
General
Conectar a base de datos
Propiedades de Conexión
PostgreSQL ajustes de conexión
PostgreSQL
General PostgreSQL Driver properties SSH SSL
Network configurations...
Server
Connect by: Host URL
URL: jdbc:postgresql://aws-0-us-east-1.pooler.supabase.com:5432/postgres
Host: aws-0-us-east-1.pooler.supabase.com
Port: 5432
Database: postgres
Show all databases
Authentication
Authentication: Database Native
Nombre de usuario: postgres.gwpbilnarclpdngdaoiv
Contraseña:
Save password
Advanced
Session role:
Local Client: PostgreSQL 17
Connection variables information
Database documentation
Connection details (name, type, ...)
Driver name: PostgreSQL
Driver Settings
Licencia del driver
Probar conexión ...
← Anterior
Siguiente →
Finalizar
Cancelar
Connection test
Conectado (2942 ms)
Server: PostgreSQL 15.8
PostgreSQL 15.8 on aarch64-unknown-linux-gnu, compiled by gcc (GCC) 13.2.0, 64-bit
Driver: PostgreSQL JDBC Driver 42.7.2
Aceptar
Detalles ↓
Project - General
Name
DataSource
Bookmarks
Dashboards
Diagrams
Scripts
Particiones
Disparadores
Reglas
Policies
Statistics
Permisos
DDL
Virtual
Comentario
Identificación_...
Nombre_Perso...
Contacto_Pers...
Edad_Perona


DBeaver 25.0.2 - persona
Archivo Editar Navegar Buscar Editor SQL Base de Datos Ventana Ayuda
Navegador de Bases de Datos
Proyectos
General
Connections
finca localhost:5434
postgres aws-0-us-east-1.pooler.supabase.com:5432
postgres 2 aws-0-us-east-1.pooler.supabase.com:5432
Bases de Datos
postgres
Esquemas
auth
extensions
graphql
graphql_public
pgbouncer
public
Tablas
persona
coche
Foreign Tables
Vistas
Vistas Materializadas
Índices
public
persona
Propiedades
Datos
Diagrama
coche
matricula
marca
modelo
caballos
id
persona
id
nombre
apellido1
apellido2
dni
Refresh
Save ...
Revert
Keep layout
100%
2 objects
Project - General
Name
DataSource
Bookmarks
Dashboards
Diagrams
Scripts
postgres 2 > postgres > public > persona
COT
es


Automotriz | Ingenious Party |
GitHub
supabase.com/dashboard/project/gwpbilnarclpdngdaoiv/editor/17267?schema=public
Ingenious Party
Free
Automotriz
Connect
Enable branching
Feedback
Table Editor
Filter
Sort
Insert
RLS disabled
Realtime off
API Docs
schema public
New table
Search tables...
coche
persona
id int4
nombre varchar
apellido1 varchar
apellido2 varchar
dni varchar
This table is empty
Add rows to your table to get started.
Import data from CSV
Page 1 of 1
100 rows
0 records
Refresh
Data
Definition
18°C
Mayorm. nublado
Buscar
ESP
LAA
12:02:08 PM
11/04/2025


File Edit Selection View Go Run
Untitled (Workspace)
EXPLORER
OPEN EDITORS
dbase.js Ejercicio...
indexing.js Ejerci...
UNTITLED (WORKSPACE)
PostgresqlBDM
MysqlBDM
KatrasK
EjercicioSupabaseC2
node_modules
dbase.js
indexing.js
package-lock.json
package.json
OUTLINE
TIMELINE
dbase.js
indexing.js
EjercicioSupabaseC2 > indexing.js > app.put('/api/actualizar/:matricula') callback
141 app.put('/api/actualizar/:matricula', async (req, res) => {
PROBLEMS DEBUG CONSOLE TERMINAL PORTS
node - EjercicioSupabaseC2
PS C:\Users\felip\Documents\EjercicioSupabaseC2> npm install express pg
PS C:\Users\felip\Documents\EjercicioSupabaseC2> npm install express pg
added 80 packages in 11s
14 packages are looking for funding
run `npm fund` for details
PS C:\Users\felip\Documents\EjercicioSupabaseC2> npm init -y
Wrote to C:\Users\felip\Documents\EjercicioSupabaseC2\package.json:
{
"dependencies": {
"express": "^5.1.0",
"pg": "^8.14.1"
},
"name": "ejerciciosupabasec2",
"version": "1.0.0",
"main": "dbase.js",
"devDependencies": {},
"scripts": {
"test": "echo \"Error: no test specified\" && exit 1"
},
"keywords": [],
"author": "",
"license": "ISC",
"description": ""
}
PS C:\Users\felip\Documents\EjercicioSupabaseC2> node --watch indexing.js
Servidor corriendo en el puerto 3010
Ln 153, Col 69 Spaces: 4 UTF-8 CRLF {} JavaScript
Buscar
ESP
LAA
12:39:52 PM
11/04/2025

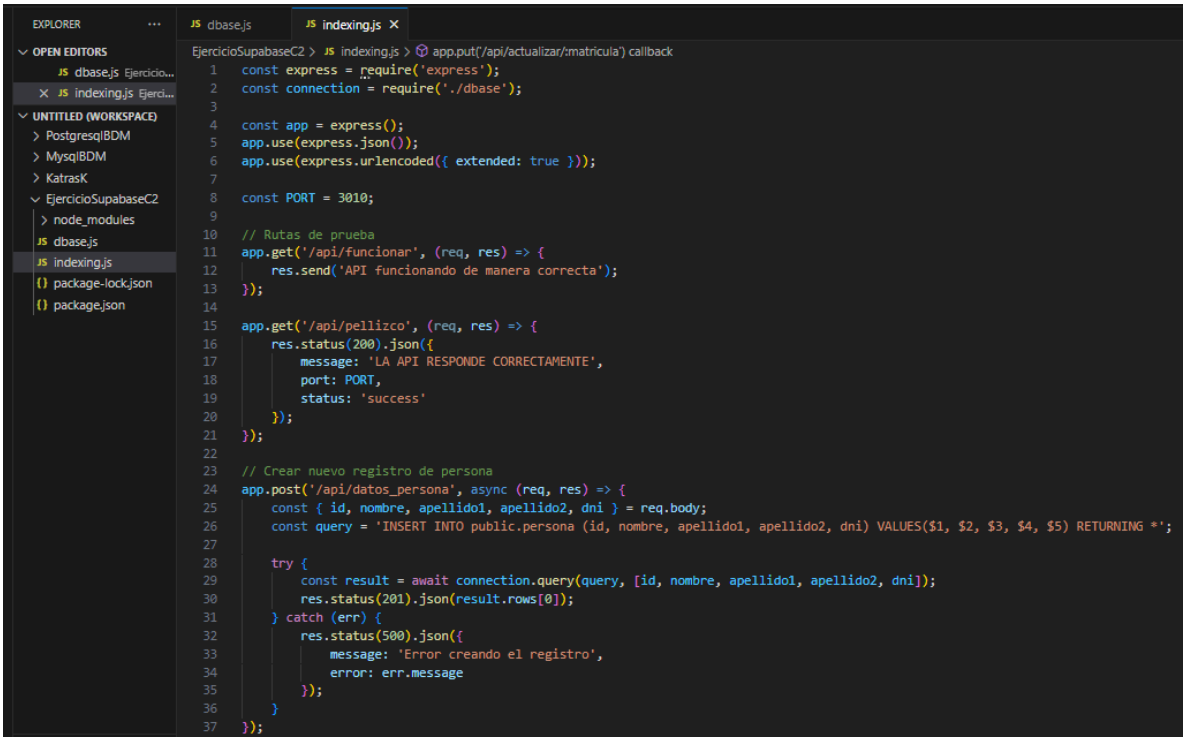


```
const { Client } = require("pg");

const client = new Client({
    host: "aws-0-us-east-1.pooler.supabase.com",
    port: 5432,
    user: "postgres.gwpbilnarclpdngdaoiv",
    password: "Si9J*Q+J#tW9xSX",
    database: "postgres",
    ssl: { rejectUnauthorized: false }
});

client.connect((err) => {
    if (err) {
        console.error("Error conectando con la base de datos:", err.stack);
    } else {
        console.log("Conectado a la base de datos PostgreSQL");
    }
});

module.exports = client;
```



```
const express = require('express');
const connection = require('./dbase');

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

const PORT = 3010;

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

app.get('/api/pellizco', (req, res) => {
    res.status(200).json({
        message: 'LA API RESPONDE CORRECTAMENTE',
        port: PORT,
        status: 'success'
    });
});

// Crear nuevo registro de persona
app.post('/api/datos_persona', async (req, res) => {
    const { id, nombre, apellido1, apellido2, dni } = req.body;
    const query = 'INSERT INTO public.persona (id, nombre, apellido1, apellido2, dni) VALUES($1, $2, $3, $4, $5) RETURNING *';

    try {
        const result = await connection.query(query, [id, nombre, apellido1, apellido2, dni]);
        res.status(201).json(result.rows[0]);
    } catch (err) {
        res.status(500).json({
            message: 'Error creando el registro',
            error: err.message
        });
    }
});
```

dbase.js | indexing.js

EjercicioSupabaseC2 > indexing.js > app.put('/api/actualizar/:matricula') callback

```
24  app.post('/api/datos_persona', async (req, res) => {
37  });
38
39  // Obtener todos los registros de perona
40  app.get('/api/obtener_Personas', async (req, res) => {
41      const query = 'SELECT * FROM public.persona';
42      try {
43          const result = await connection.query(query);
44          res.status(200).json({
45              success: true,
46              data: result.rows
47          });
48      } catch (err) {
49          res.status(500).json({
50              success: false,
51              message: 'Error al recuperar los datos',
52              details: err.message
53          });
54      }
55  });
56
57  // Eliminar por ID una persona
58  app.delete('/api/quitar_persona/:id', async (req, res) => {
59      const { id } = req.params;
60      const query = 'DELETE FROM public.persona WHERE id = $1 RETURNING *';
61
62      try {
63          const result = await connection.query(query, [id]);
64          if (result.rowCount === 0) {
65              return res.status(404).json({ success: false, message: `No existe el registro con ID ${id}` });
66          }
67          res.status(200).json({ success: true, deleted: result.rows[0] });
68      } catch (err) {
69          res.status(500).json({ success: false, error: err.message });
70      }
71  });
72
73  // Actualizar por ID una persona
74  app.put('/api/actualizar/:id', async (req, res) => {
```

dbase.js | indexing.js

EjercicioSupabaseC2 > indexing.js > app.put('/api/actualizar/:matricula') callback

```
app.delete('/api/quitar_persona/:id', async (req, res) => {
});

// Actualizar por ID una persona
app.put('/api/actualizar/:id', async (req, res) => {
    const { id } = req.params;
    const { nombre, apellido1, apellido2, dni } = req.body;
    const query = 'UPDATE public.persona SET nombre = $1, apellido1 = $2, apellido2 = $3, dni = $4 WHERE id = $5 RETURNING *';

    try {
        const result = await connection.query(query, [nombre, apellido1, apellido2, dni, id]);
        if (result.rowCount === 0) {
            return res.status(404).json({ success: false, message: `No se encontró ningún registro con el ID ${id}` });
        }
        res.status(200).json({ success: true, updated: result.rows[0] });
    } catch (err) {
        res.status(500).json({ success: false, error: err.message });
    }
});

// Crear nuevo registro del coche
app.post('/api/datos_coche', async (req, res) => {
    const { matricula, marca, modelo, caballos, id } = req.body;
    const query = 'INSERT INTO public.coche ( matricula, marca, modelo, caballos, id) VALUES($1, $2, $3, $4, $5) RETURNING *';

    try {
        const result = await connection.query(query, [ matricula, marca, modelo, caballos, id]);
        res.status(201).json(result.rows[0]);
    } catch (err) {
        res.status(500).json({
            message: 'Error creando el registro',
            error: err.message
        });
    }
});
```

dbase.js | indexing.js

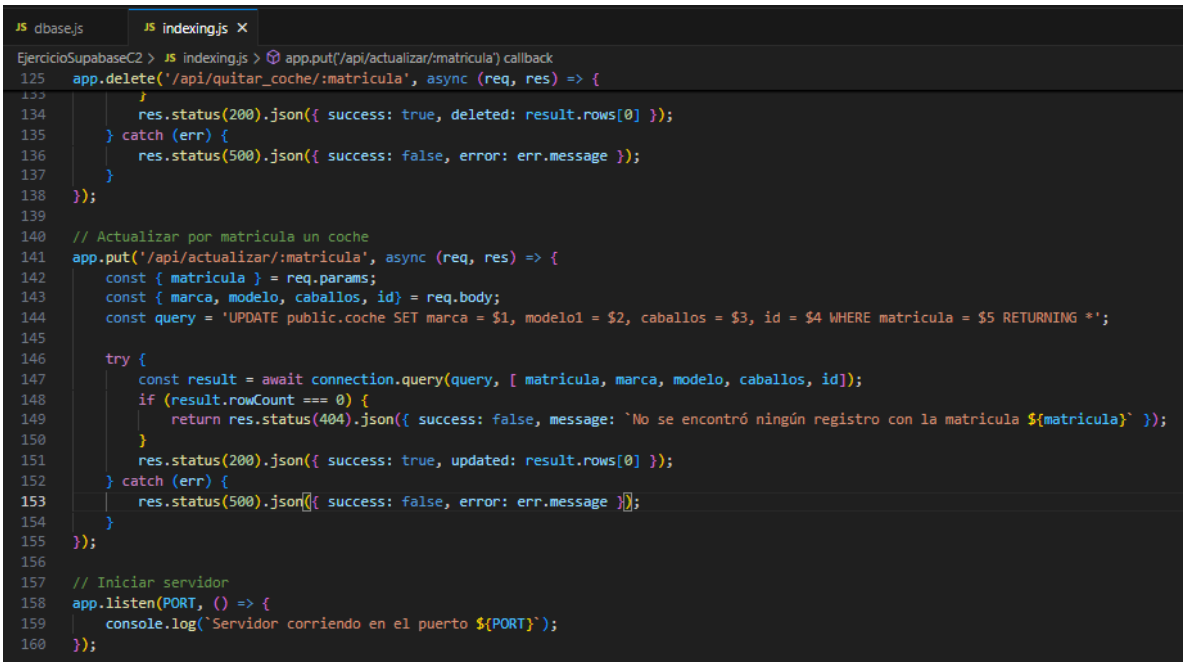
EjercicioSupabaseC2 > indexing.js > app.put('/api/actualizar/:matricula') callback

```
app.post('/api/datos_coche', async (req, res) => {
});

// Obtener todos los registros de coche
app.get('/api/obtener_coches', async (req, res) => {
    const query = 'SELECT * FROM public.coches';
    try {
        const result = await connection.query(query);
        res.status(200).json({
            success: true,
            data: result.rows
        });
    } catch (err) {
        res.status(500).json({
            success: false,
            message: 'Error al recuperar los datos',
            details: err.message
        });
    }
});

// Eliminar por matrícula un coche asociado a una persona
app.delete('/api/quitar_coche/:matricula', async (req, res) => {
    const { matricula } = req.params;
    const query = 'DELETE FROM public.coche WHERE matricula = $1 RETURNING *';

    try {
        const result = await connection.query(query, [matricula]);
        if (result.rowCount === 0) {
            return res.status(404).json({ success: false, message: `No existe el registro con matricula ${matricula}` });
        }
        res.status(200).json({ success: true, deleted: result.rows[0] });
    } catch (err) {
        res.status(500).json({ success: false, error: err.message });
    }
});
```

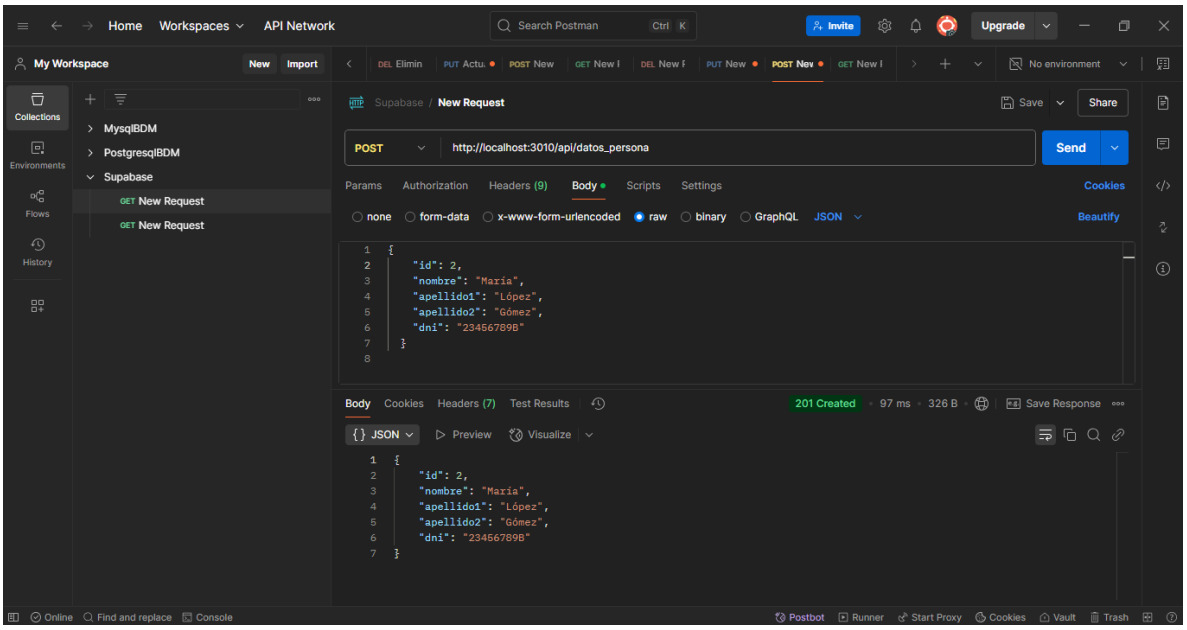


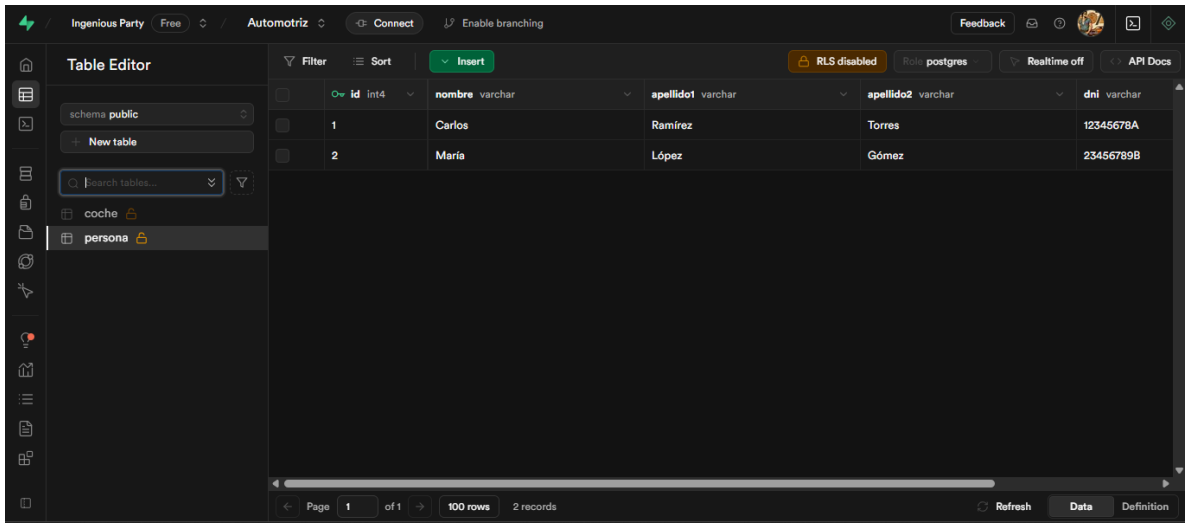
```
JS dbase.js   JS indexing.js ×
EjercicioSupabaseC2 > JS indexing.js > app.put('/api/actualizar/:matricula') callback
app.delete('/api/quitar_coche/:matricula', async (req, res) => {
        }
        res.status(200).json({ success: true, deleted: result.rows[0] });
    } catch (err) {
        res.status(500).json({ success: false, error: err.message });
    }
});

// Actualizar por matricula un coche
app.put('/api/actualizar/:matricula', async (req, res) => {
    const { matricula } = req.params;
    const { marca, modelo, caballos, id} = req.body;
    const query = 'UPDATE public.coche SET marca = $1, modelo1 = $2, caballos = $3, id = $4 WHERE matricula = $5 RETURNING *';

    try {
        const result = await connection.query(query, [ matricula, marca, modelo, caballos, id]);
        if (result.rowCount === 0) {
            return res.status(404).json({ success: false, message: `No se encontró ningún registro con la matricula ${matricula}` });
        }
        res.status(200).json({ success: true, updated: result.rows[0] });
    } catch (err) {
        res.status(500).json({ success: false, error: err.message });
    }
});

// Iniciar servidor
app.listen(PORT, () => {
    console.log(`Servidor corriendo en el puerto ${PORT}`);
});
```




Ingenious Party
Free
Automotriz
Connect
Enable branching
Feedback
Table Editor
schema public
New table
Search tables...
coche
persona
Filter
Sort
Insert
RLS disabled
Role postgres
Realtime off
API Docs
id int4
nombre varchar
apellido1 varchar
apellido2 varchar
dni varchar
1
Carlos
Ramírez
Torres
12345678A
2
María
López
Gómez
23456789B
Page 1 of 1
100 rows
2 records
Refresh
Data
Definition