PRÁCTICA 27: BACKEND CON SUPABASE Y RENDER + INTERFAZ WEB PARA INSERTAR Y LISTAR USUARIOS

OBJETIVO

Desarrollar una **aplicación completa**:

- **Backend (Node.js + PostgreSQL)** en Render
- **Base de datos Supabase**
- **Frontend HTML/CSS/JS** para consultar e insertar datos

REQUISITOS

- Cuenta en Render
- Cuenta en Supabase
- Cuenta en GitHub
- Conocimientos básicos de Node.js, Fetch y HTML

1. CONFIGURAR SUPABASE

1. CREAR LA BASE DE DATOS EN SUPABASE

1. Ir a https://app.supabase.com
2. Clic en **"New project"**
3. Rellena:
    - Nombre: basedatos-alumno
    - Region: EU (recomendado)
    - Password segura
4. Espera unos segundos a que se cree

2. CREAR UNA TABLA DESDE EL PANEL

Ve a **Table Editor** y crea, por ejemplo, una tabla usuarios:

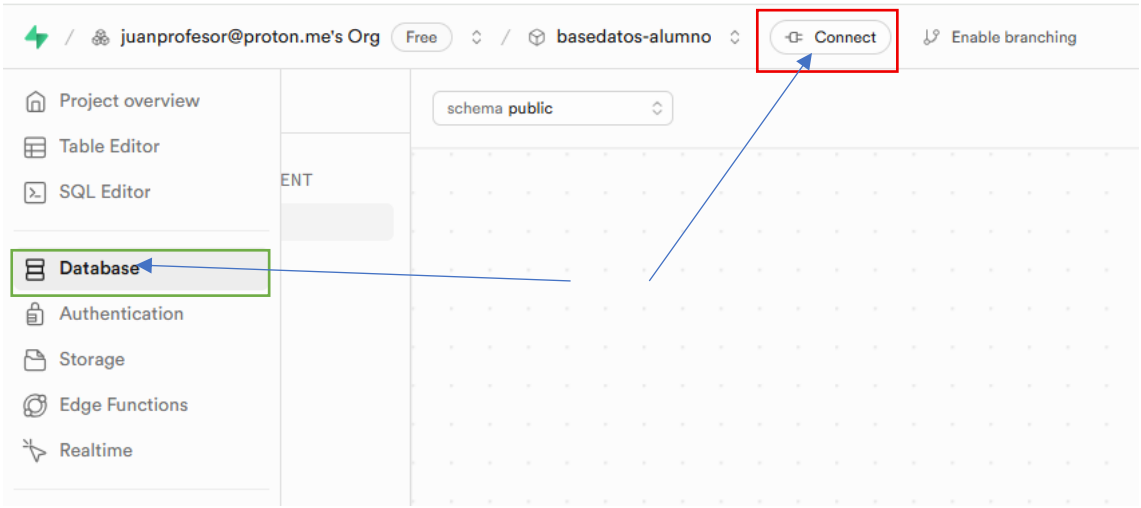
Campo	Tipo	Clave primaria
id	UUID	✅ Sí
nombre	text	
email	text	

Activa el modo de inserción automática de id con gen_random_uuid()

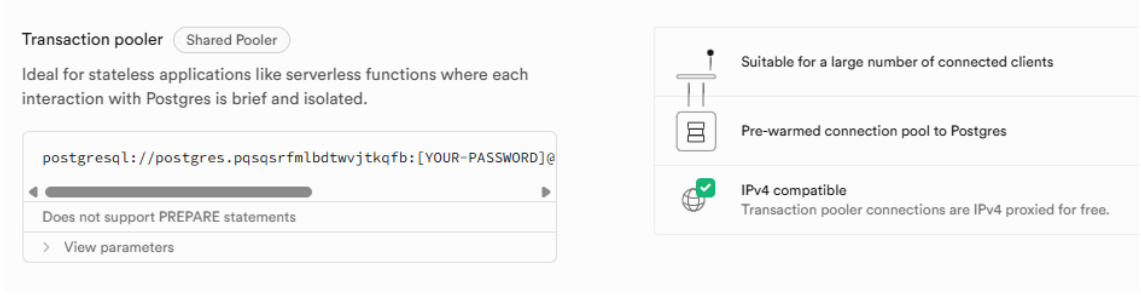
3. COPIAR LAS CREDENCIALES DE CONEXIÓN

En el panel de Supabase, ve a:

Project Settings > Database



Usaremos una alternativa llamada **Transaction Pooler**, que es **perfecta para Render y entornos serverless**.



Copia el valor de **Connection string (URI)** en formato:

```
postgresql://usuario:clave@host:puerto/nombrebd
postgresql://postgres.pqsqsrfmlbdtwvjtkqfb:H0lacarac0la@aws-0-eu-west-3.pooler.supabase.com:6543/postgres
```

Guárdalo como variable SUPABASE_DB_URL

2. BACKEND CON EXPRESS + PG

📁 Estructura del proyecto

```
/backend
 ├── server.js
 ├── package.json
 ├── .env          (local)
 └── public/
    ├── index.html
```

```
├── style.css
└── script.js
```

SERVER.JS

```
require('dotenv').config();
const express = require('express');
const { Pool } = require('pg');
const path = require('path');

const app = express();
app.use(express.json());
app.use(express.static('public'));

const pool = new Pool({
 connectionString: process.env.SUPABASE_DB_URL,
 ssl: { rejectUnauthorized: false }
});

app.get('/api/usuarios', async (req, res) => {
 const result = await pool.query('SELECT * FROM usuarios ORDER BY nombre');
 res.json(result.rows);
});

app.post('/api/usuarios', async (req, res) => {
 const { nombre, email } = req.body;
 await pool.query('INSERT INTO usuarios (id, nombre, email) VALUES
(gen_random_uuid(), $1, $2)', [nombre, email]);
 res.json({ mensaje: 'Usuario agregado' });
});

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => console.log(`Servidor corriendo en puerto ${PORT}`));
```

3. INTERFAZ: PUBLIC/INDEX.HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="UTF-8">
 <title>Usuarios con Supabase</title>
 <link rel="stylesheet" href="style.css">
</head>
<body>
 <h1>Gestión de Usuarios</h1>

 <form id="formulario">
  <input type="text" id="nombre" placeholder="Nombre" required>
  <input type="email" id="email" placeholder="Email" required>
  <button type="submit">Agregar usuario</button>
 </form>
```

```
  <ul id="listaUsuarios"></ul>

  <script src="script.js"></script>
</body>
</html>
```

PUBLIC/STYLE.CSS

```
body {
  font-family: Arial;
  padding: 2em;
  max-width: 600px;
  margin: auto;
}
input, button {
  padding: 0.5em;
  margin: 0.5em 0;
  width: 100%;
}
ul {
  list-style: none;
  padding: 0;
}
li {
  background: #f0f0f0;
  margin-bottom: 0.5em;
  padding: 0.5em;
}
```

PUBLIC/SCRIPT.JS

```
async function cargarUsuarios() {
  const res = await fetch('/api/usuarios');
  const usuarios = await res.json();
  const lista = document.getElementById('listaUsuarios');
  lista.innerHTML = '';
  usuarios.forEach(u => {
    lista.innerHTML += `<li><strong>${u.nombre}</strong> (${u.email})</li>`;
  });
}

document.getElementById('formulario').addEventListener('submit', async e => {
  e.preventDefault();
  const nombre = document.getElementById('nombre').value;
  const email = document.getElementById('email').value;
  await fetch('/api/usuarios', {
    method: 'POST',
    headers: { 'Content-Type': 'application/json' },
    body: JSON.stringify({ nombre, email })
  });
```

```
  cargarUsuarios();
  e.target.reset();
});

cargarUsuarios();
```

PACKAGE.JSON

```
{
  "name": "render-supabase-app",
  "version": "1.0.0",
  "description": "App fullstack con Node.js, Express y Supabase desplegada en Render",
  "main": "server.js",
  "scripts": {
    "start": "node server.js"
  },
  "keywords": [],
  "author": "Alumno MF0952",
  "license": "MIT",
  "dependencies": {
    "dotenv": "^16.3.1",
    "express": "^4.18.2",
    "pg": "^8.11.1"
  }
}
```

4. DESPLEGAR EN RENDER

1. Subir el proyecto a GitHub
2. En Render: **New > Web Service**
3. Configuración:

Campo	Valor
Start command	**node server.js**
Build command	**npm install**
Root directory	**/backend (si el código está en subcarpeta)**
Environment	**Node**

4. Deploy

RESULTADO ESPERADO

- Web pública en https://tuapp.onrender.com
- Lista visible de usuarios (desde Supabase)

- Formulario que inserta nuevos usuarios en la base de datos

ACTIVIDAD DEL ALUMNO

1. Crear Supabase + tabla
2. Configurar y subir backend con frontend incluido
3. Publicar en Render
4. Entregar la URL + captura de su interfaz funcionando