Quiz segundo corte

David Santiago Noguera Perez - 827823

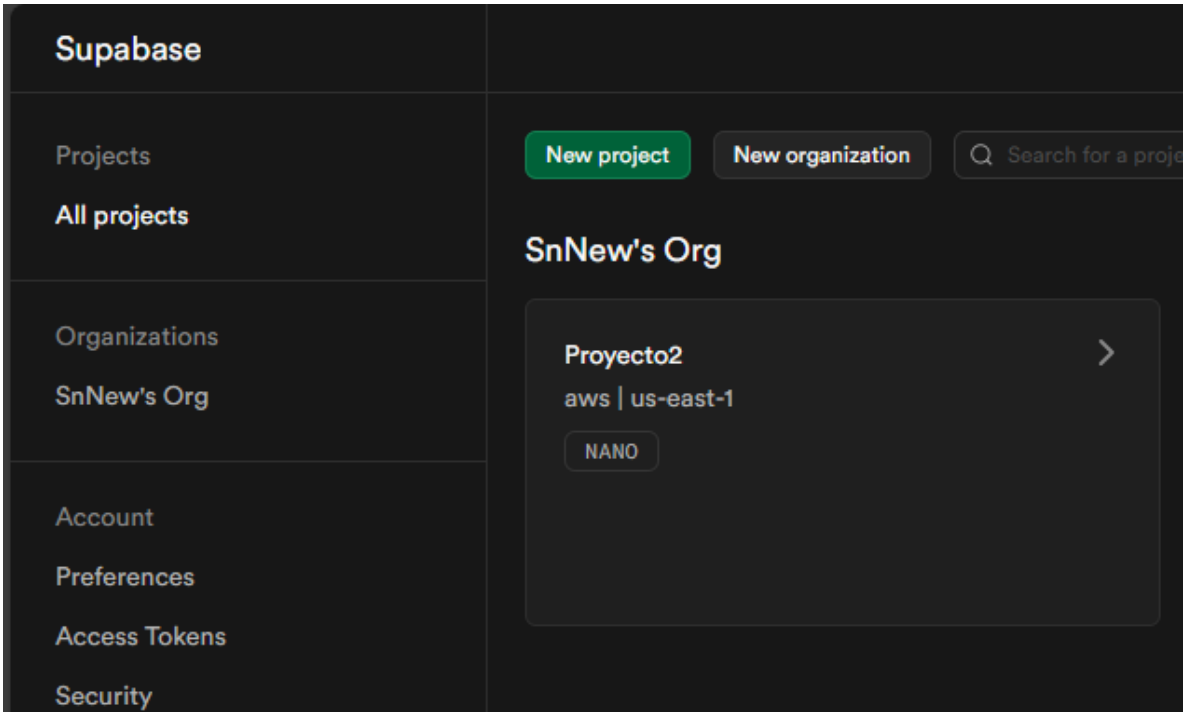
Corporación Universitaria Minuto de Dios

Ing. William Alexander Matallana Porras

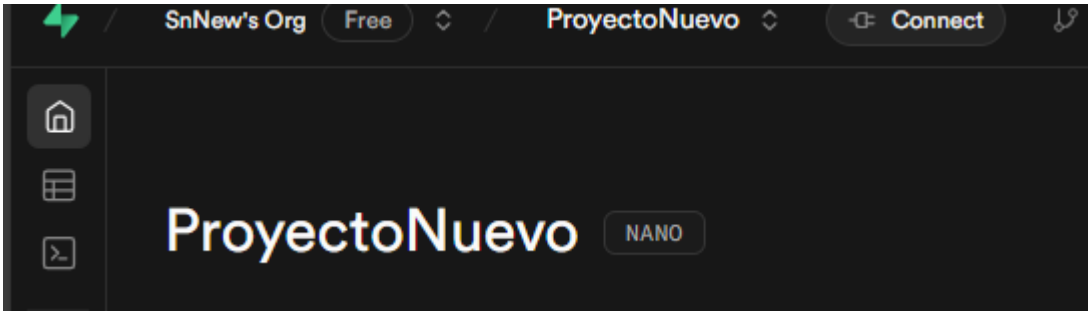
10-60747: Bases de datos

2025

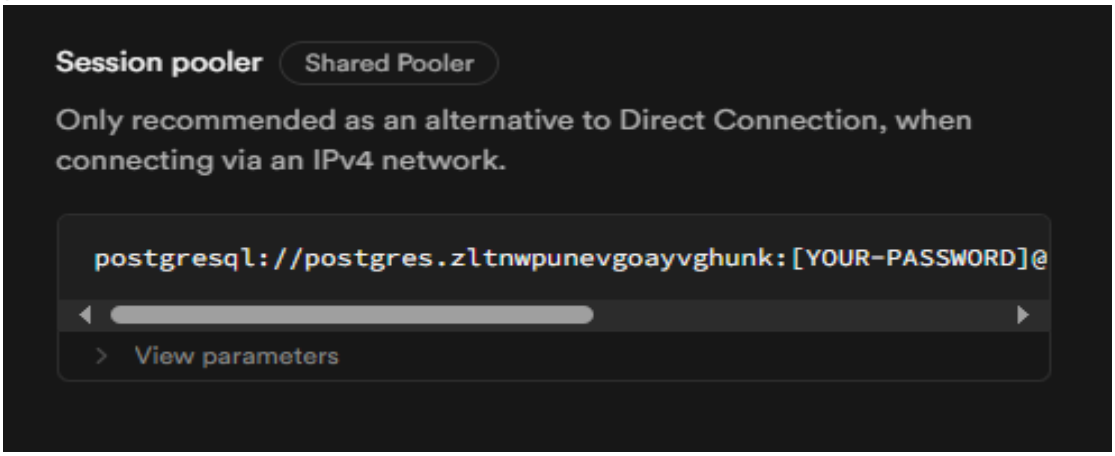
Primero nos dirigiremos a supabase en la web y crearemos un nuevo proyecto



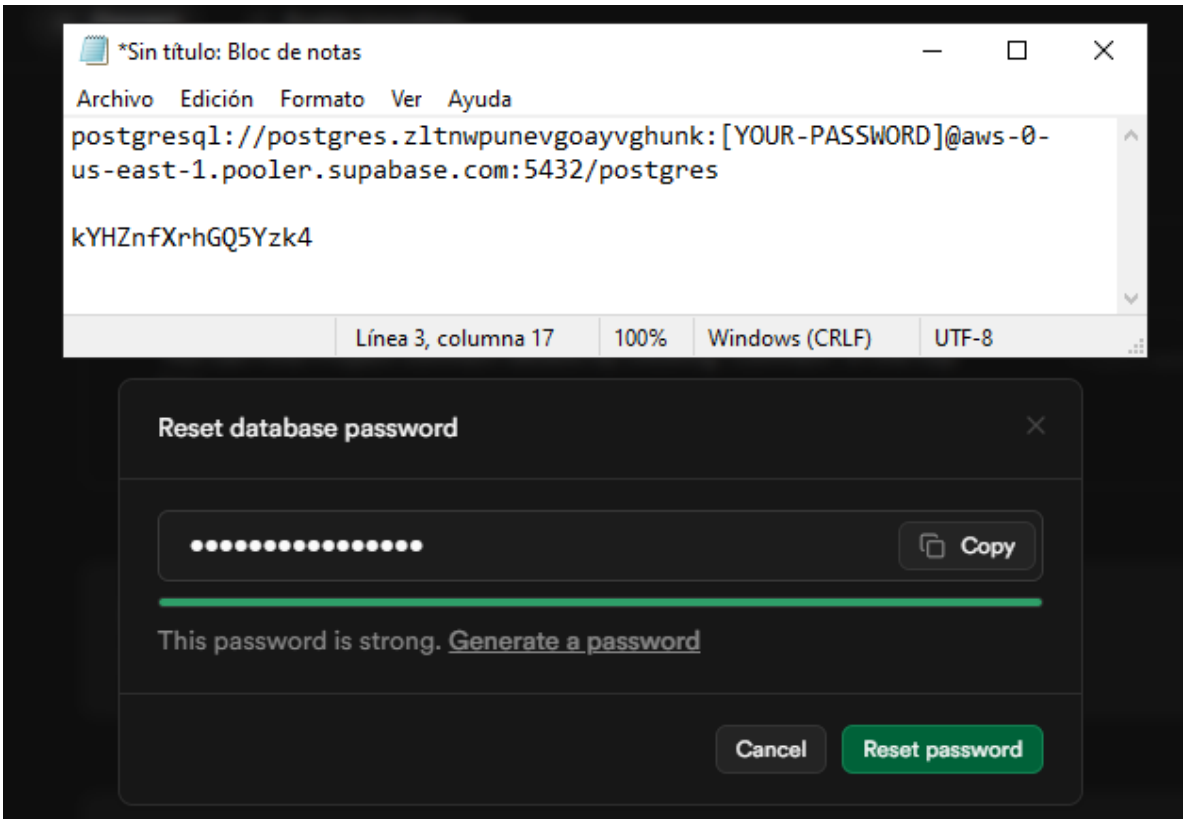
En este caso el proyecto se llamará **ProyectoNuevo**



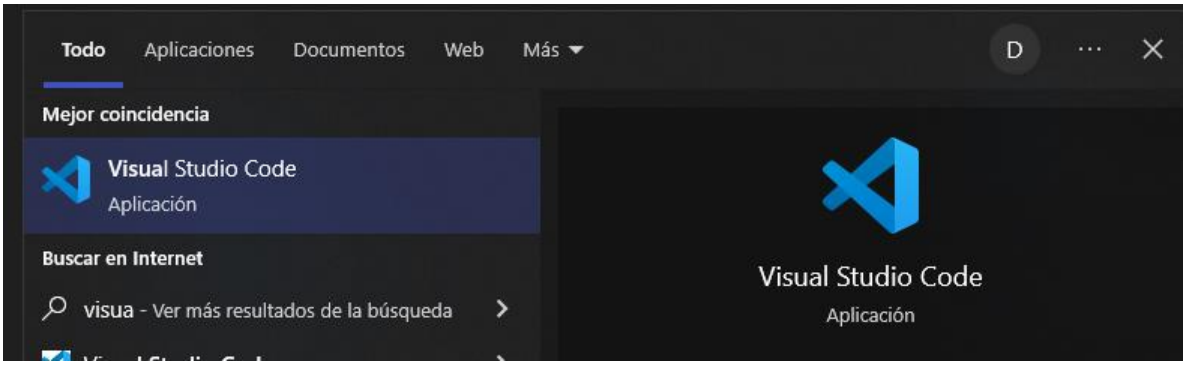
Nos iremos a la parte de connect y copiaremos las credenciales de conexión del sesión pooler



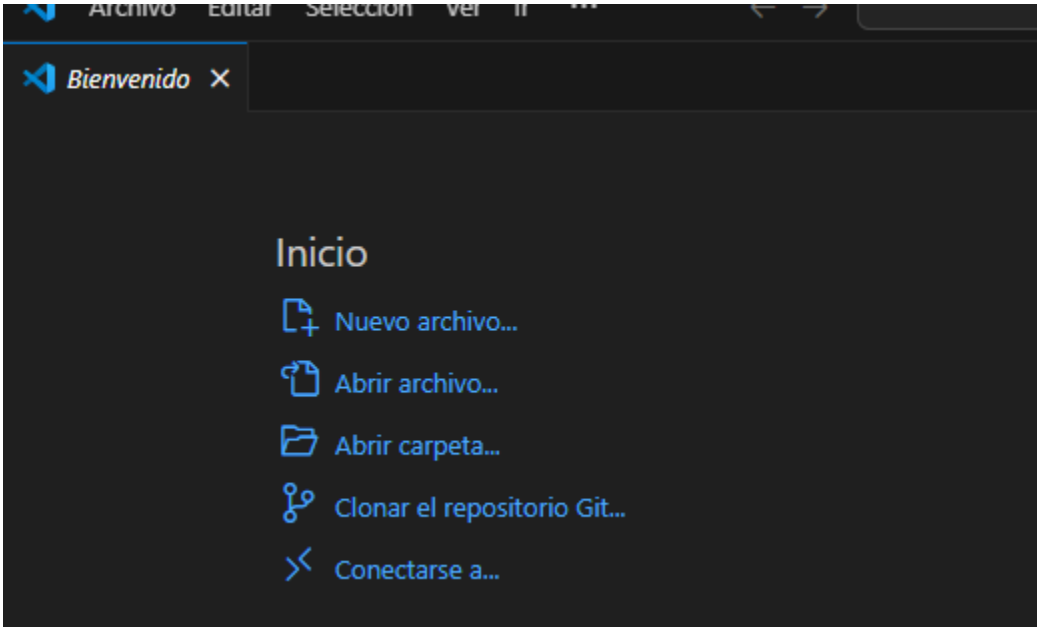
Esto lo pegaremos junto a la contraseña del proyecto en un bloc de notas para usarlo mas tarde

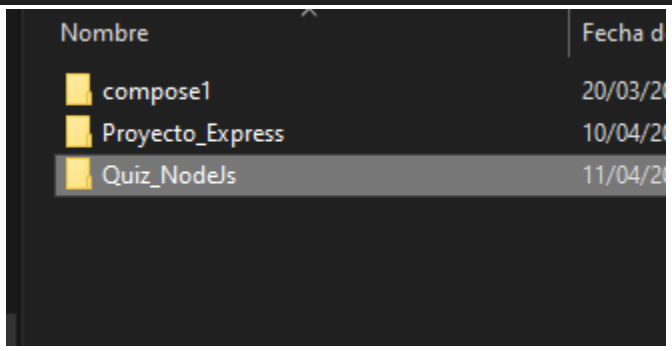


Ahora abriremos el vsCode en nuestro computador para crear nuestro proyecto

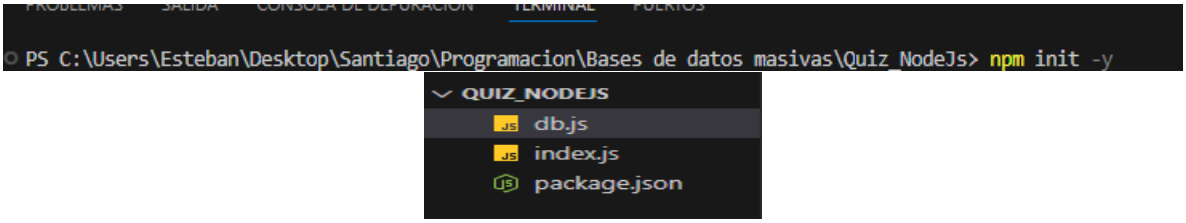


Aquí abriremos la carpeta raíz del proyecto

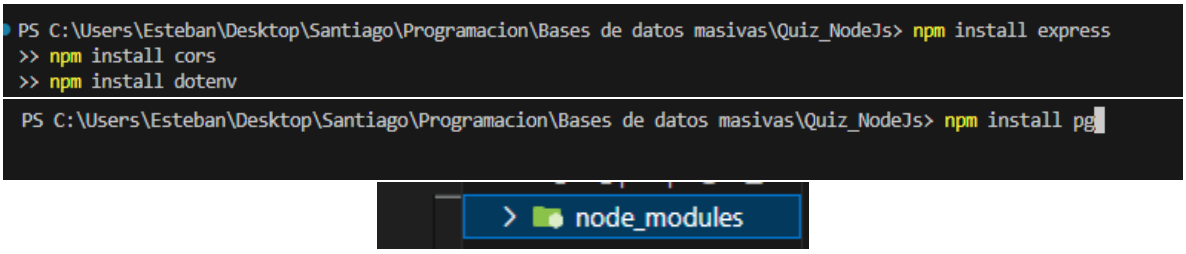




Ahora inicializaremos nuestro jsos y crearemos los archivos de index y db para posteriormente escribir nuestro código



En la consola también escribiremos los comandos para instalar librerías que usaremos para el proyecto, estas se instalaran automáticamente en la carpeta de módulos



El archivo **db.js** se encarga de **configurar la conexión a la base de datos** de Supabase usando la librería pg (que sirve para conectarse a PostgreSQL desde Node.js).
Es básicamente el punto donde le decimos al proyecto cómo y dónde conectarse a la base de datos.

```
index.js   db.js
db.js > pool > ssl
1   const { Pool } = require('pg');
2
3   const pool = new Pool({
4     user: 'postgres.zltnwpunevgoayvghunk',
5     host: 'aws-0-us-east-1.pooler.supabase.com',
6     database: 'postgres',
7     password: '9ZIHfz64mmI3msWf',
8     port: 5432,
9     ssl: {
10      rejectUnauthorized: false
11    }
12  });
13
14  (async () => {
15    try {
16      const client = await pool.connect();
17      console.log('Conectado exitosamente a Supabase');
18      client.release();
19    } catch (err) {
20      console.error('Error al conectar con Supabase:', err.stack);
21    }
22  })();
23
24  module.exports = pool;
25
```

En esta parte probaremos que si funciona con un index que solo lanza los servicios

```
index.js   db.js
index.js > ...
1   require('./db');
2
3   const express = require('express');
4   const cors = require('cors');
5   const app = express();
6
7   app.use(cors());
8   app.use(express.json());
9
10  app.listen(3000, () => {
11    console.log('Servidor funcionando');
12  });
```

```
PS C:\Users\Esteban\Desktop\Santiago\Programacion\Bases de datos masivas\Quiz_NodeJs> node index.js
Servidor funcionando
Conectado exitosamente a Supabase
```

El archivo **index.js** es el **corazón del servidor**. Es donde configuramos **Express**, las **rutas (endpoints)** para manejar personas y coches, y donde finalmente **levantamos el servidor** para que escuche en el puerto 3000.

express: librería para crear el servidor web.

cors: permite que otros sitios (como el front-end) puedan hacer peticiones a nuestro servidor.

pool: lo importamos del archivo db.js para poder hacer consultas a Supabase.app es nuestra aplicación Express.

app.use(cors()): permite peticiones desde cualquier origen (útil para desarrollo).

app.use(express.json()): permite recibir datos en formato JSON (por ejemplo, con POST y PUT).

```
const express = require('express');
const cors = require('cors');
const pool = require('./db');

const app = express();
app.use(cors());
app.use(express.json());
```

Ahora crearemos el respectivo CRUD para persona, las demás apis se ven en el proyecto

```
//Persona

// Obtener todas las personas
app.get('/personas', async (req, res) => {
  try {
    const result = await pool.query('SELECT * FROM persona');
    res.json(result.rows);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// Obtener una persona por ID
app.get('/personas/:id', async (req, res) => {
  try {
    const { id } = req.params;
    const result = await pool.query('SELECT * FROM persona WHERE id = $1', [id]);
    res.json(result.rows[0]);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});
```

Y el CRUD de coche

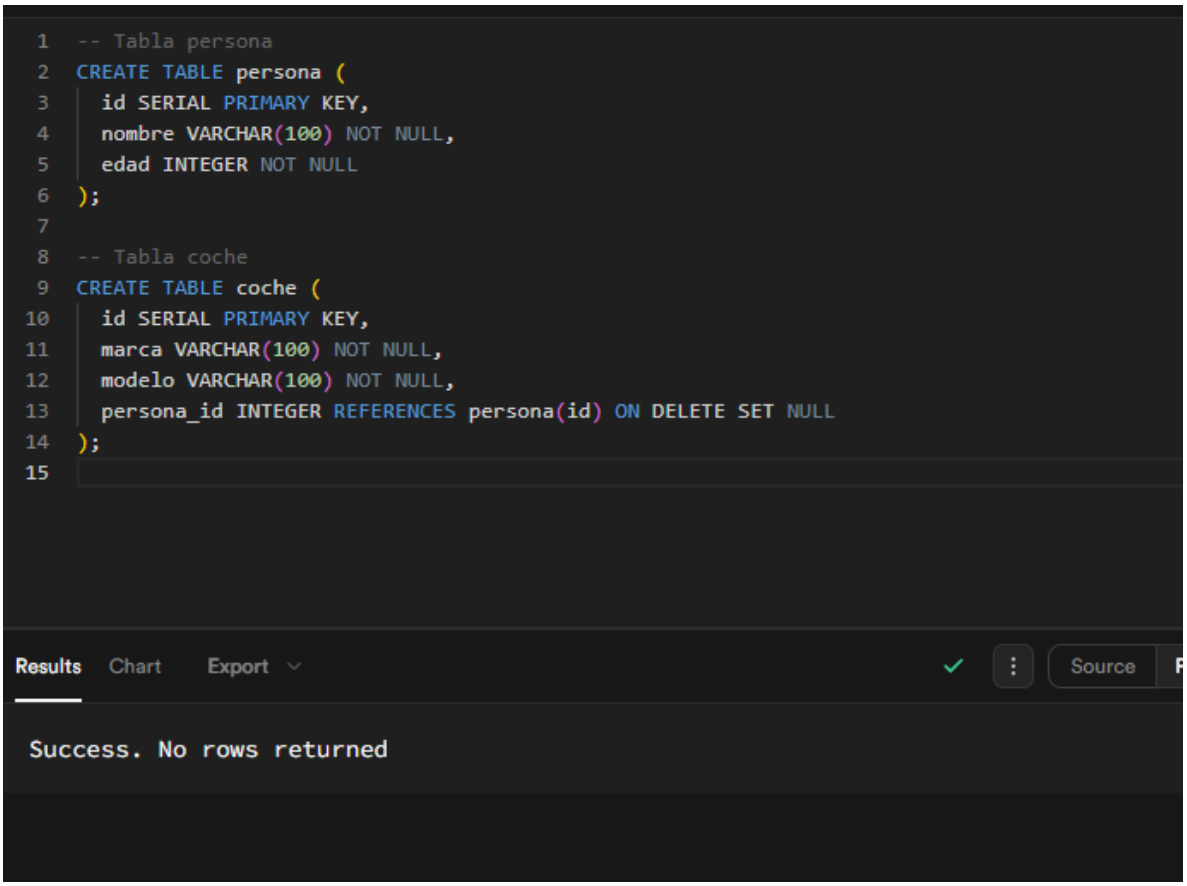
```
//Coche

// Obtener todos los coches
app.get('/coches', async (req, res) => {
  try {
    const result = await pool.query('SELECT * FROM coche');
    res.json(result.rows);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// Obtener un coche por ID
app.get('/coches/:id', async (req, res) => {
  try {
    const { id } = req.params;
    const result = await pool.query('SELECT * FROM coche WHERE id = $1', [id]);
    res.json(result.rows[0]);
  } catch (err) {
    res.status(500).json({ error: err.message });
  }
});

// Crear un nuevo coche
app.post('/coches', async (req, res) => {
  try {
    const { marca, modelo, persona_id } = req.body;
    const result = await pool.query(
      'INSERT INTO coche (marca, modelo, persona_id) VALUES ($1, $2, $3) RETURNING *',
```

Ahora en el SQL editor de supabase crearemos las tablas y haremos los registros


-- Tabla persona
CREATE TABLE persona (
id SERIAL PRIMARY KEY,
nombre VARCHAR(100) NOT NULL,
edad INTEGER NOT NULL
);
-- Tabla coche
CREATE TABLE coche (
id SERIAL PRIMARY KEY,
marca VARCHAR(100) NOT NULL,
modelo VARCHAR(100) NOT NULL,
persona_id INTEGER REFERENCES persona(id) ON DELETE SET NULL
);
Results
Chart
Export
Source
Success. No rows returned