

Parcial 2

Bases de datos masivas

UNIMINUTO zipaquira

Juan manuel moncada felix

809397

23 de abril del del 2025

1. Creamos un nuevo proyecto donde creamos dos documentos, db.js e index.js

```
JS db.js
JS index.js
```

2. Instalamos los paquetes para poder trabajar de la manera mas optima nuestras apis y conexiones

```
PS C:\Users\monca\Desktop\parcial2BD> npm install express pg cors
added 14 packages, and audited 83 packages in 4s

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

```
PS C:\Users\monca\Desktop\parcial2BD> npm init -y
Wrote to C:\Users\monca\Desktop\parcial2BD\package.json:

{
  "dependencies": {
    "cors": "^2.8.5",
    "express": "^5.1.0",
    "pg": "^8.15.1"
  },
```

```
PS C:\Users\monca\Desktop\parcial2BD> npm install -g nodemon
added 29 packages in 4s

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

```
{ } package-lock.json
{ } package.json
```

3. En nuestro db.js, creamos la conexión con nuestro proyecto en supabase

```
JS index.js JS db.js X
JS db.js > ...
1  const { Client } = require('pg');
2
3  const client = new Client({
4    host: 'aws-0-us-east-1.pooler.supabase.com',
5    port: 5432,
6    user: 'postgres.mtncunnwsxacefzoyorf',
7    password: '1072639163',
8    database: 'postgres'
9  });
10
11 client.connect()
12   .then(() => console.log('Conectado a Supabase PostgreSQL'))
13   .catch(err => console.error('Error conectando a Supabase:', err.stack));
14
15 module.exports = client;
16
17
```

```
PS C:\Users\monca\Desktop\parcial2BD> nodemon index.js
[nodemon] 3.1.10
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,cjs,json
[nodemon] starting `node index.js`
Servidor corriendo en puerto 3001
Conectado a Supabase PostgreSQL
```

4. Realizamos la conexión con PGAdmin 4, siendo este un puente y ayudarnos a crear el backup

Parcial2Bd

General	Connection	Parameters	SSH Tunnel	Advanced	Tags
Host name/address	aws-0-us-east-1.pooler.supabase.com				
Port	5432				
Maintenance database	postgres				
Username	postgres.mtncunnwsxacefzoyorf				
Kerberos authentication?	<input type="checkbox"/>				
Role					
Service					

- Backup

<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	25000	Backup Object	Parcial2Bd (aws-0-u...	postgres	25/4/2025, 12:28:48	Running	12.93
--------------------------	-------------------------------------	--------------------------	-------	---------------	------------------------	----------	---------------------	---------	-------

5. Creamos nuestras tablas en mi caso fueron creadas desde supabase, e ingresamos datos

```
1 CREATE TABLE Restaurante (  
2     id_rest INT PRIMARY KEY AUTO_INCREMENT,  
3     nombre VARCHAR(100),  
4     ciudad VARCHAR(100),  
5     direccion VARCHAR(150),  
6     fecha_apertura DATE  
7 );
```

```
1 CREATE TABLE Restaurante (  
2     id_rest SERIAL PRIMARY KEY,  
3     nombre VARCHAR(100),  
4     ciudad VARCHAR(100),  
5     direccion VARCHAR(150),  
6     fecha_apertura DATE  
7 );  
8
```

```
1 CREATE TABLE Empleado (  
2     id_empleado SERIAL PRIMARY KEY,  
3     nombre VARCHAR(100),  
4     rol VARCHAR(50),  
5     id_rest INT REFERENCES Restaurante(id_rest)  
6 );  
7
```

```
1 CREATE TABLE Producto (  
2     id_prod SERIAL PRIMARY KEY,  
3     nombre VARCHAR(100),  
4     precio NUMERIC(10,2)  
5 );
```

```
CREATE TABLE Pedido (  
    id_pedido SERIAL PRIMARY KEY,  
    fecha DATE,  
    id_rest INT REFERENCES Restaurante(id_rest),  
    total NUMERIC(10,2)  
);
```

6. En el index.js configuraremos la conexión con postman y asignamos cada uno de los parámetros necesarios

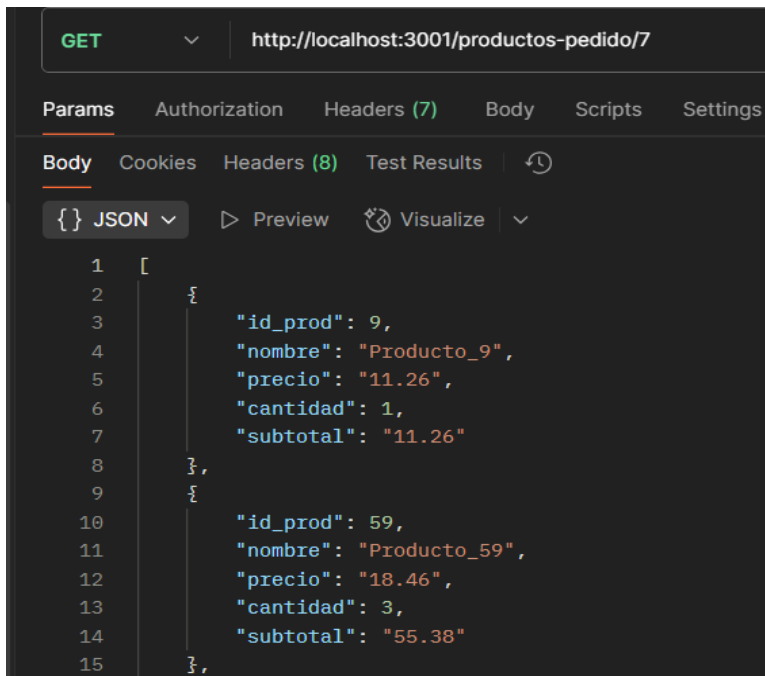
```
JS index.js > ...
1  const express = require('express');
2  const client = require('./db');
3  const cors = require('cors');
4
5  const app = express();
6  const PORT = 3001;
7
8  app.use(cors());
9  app.use(express.json());
10 app.use(express.urlencoded({ extended: true }));
11
12 app.listen(PORT, () => {
13   console.log(`Servidor corriendo en puerto ${PORT}`);
14 });
15
```

7. Creamos cada una de las apis, además de las consultas nativas que debemos realizar

- Obtener todos los productos de un pedido específico

Esta consulta, realiza a traves de la url en posman, un llamado a el **nombre**, **precio** y **cantidad** de **todos los productos** que están en un **pedido**, en la cual se debe agregar el id del pedido que quiere consultar,

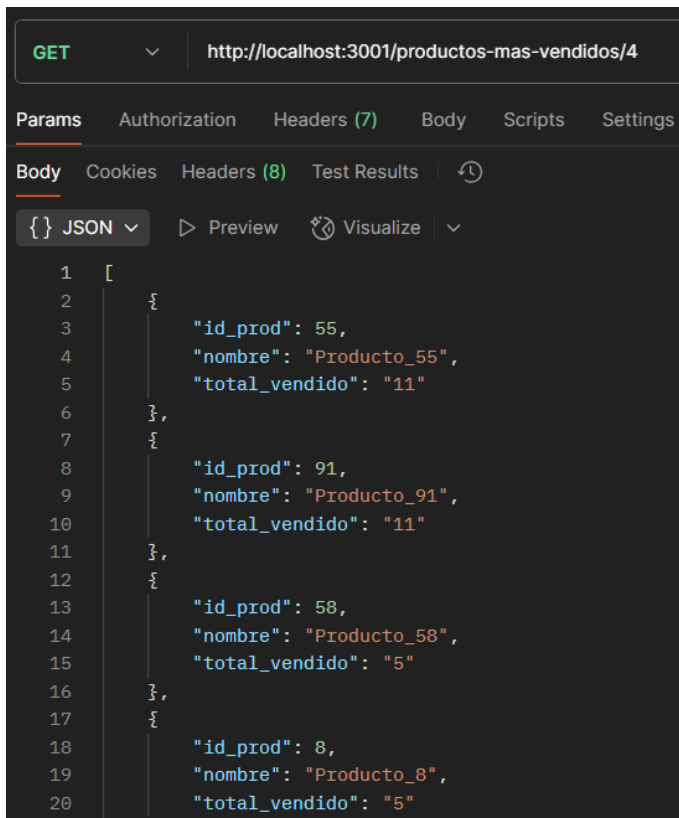
```
// 1. Obtener productos de un pedido
app.get('/productos-pedido/:id', async (req, res) => {
  const { id } = req.params;
  const result = await client.query(`
    SELECT p.id_prod, p.nombre, p.precio, dp.cantidad, dp.subtotal
    FROM DetallePedido dp
    JOIN Producto p ON dp.id_prod = p.id_prod
    WHERE dp.id_pedido = $1
  `, [id]);
  res.json(result.rows);
});
```



- Obtener los productos más vendidos (más de X unidades)

La consulta hace el llamado al nombre, precio y cantidad de todos los productos que están en un pedido los cuales son mostrados si su venta fue mayor a x cantidad ingresada por nosotros

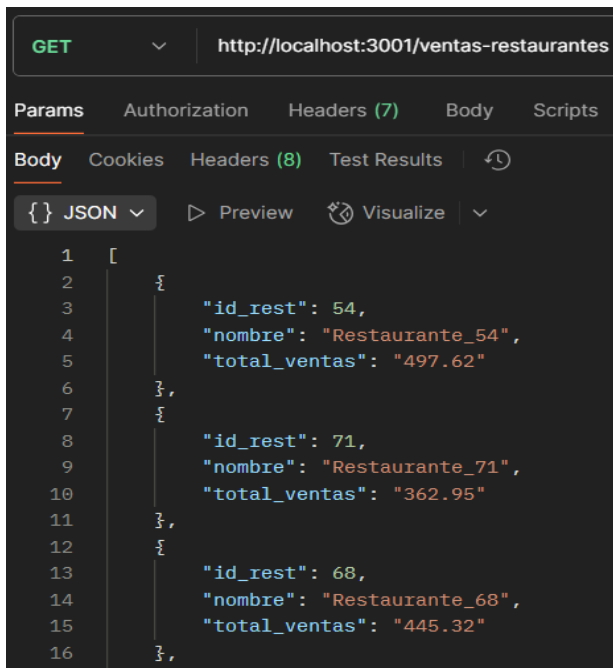
```
// 2. Productos más vendidos (más de X unidades)
app.get('/productos-mas-vendidos/:min', async (req, res) => {
  const { min } = req.params;
  const result = await client.query(`
    SELECT p.id_prod, p.nombre, SUM(dp.cantidad) AS total_vendido
    FROM DetallePedido dp
    JOIN Producto p ON dp.id_prod = p.id_prod
    GROUP BY p.id_prod, p.nombre
    HAVING SUM(dp.cantidad) > $1
  `, [min]);
  res.json(result.rows);
});
```



- Obtener el total de ventas por restaurante

Esta consulta nos Calcula la suma del precio de todos los productos vendidos por una cantidad especificada en un día específico, Básicamente te da cuánto dinero hiciste en ventas ese día.

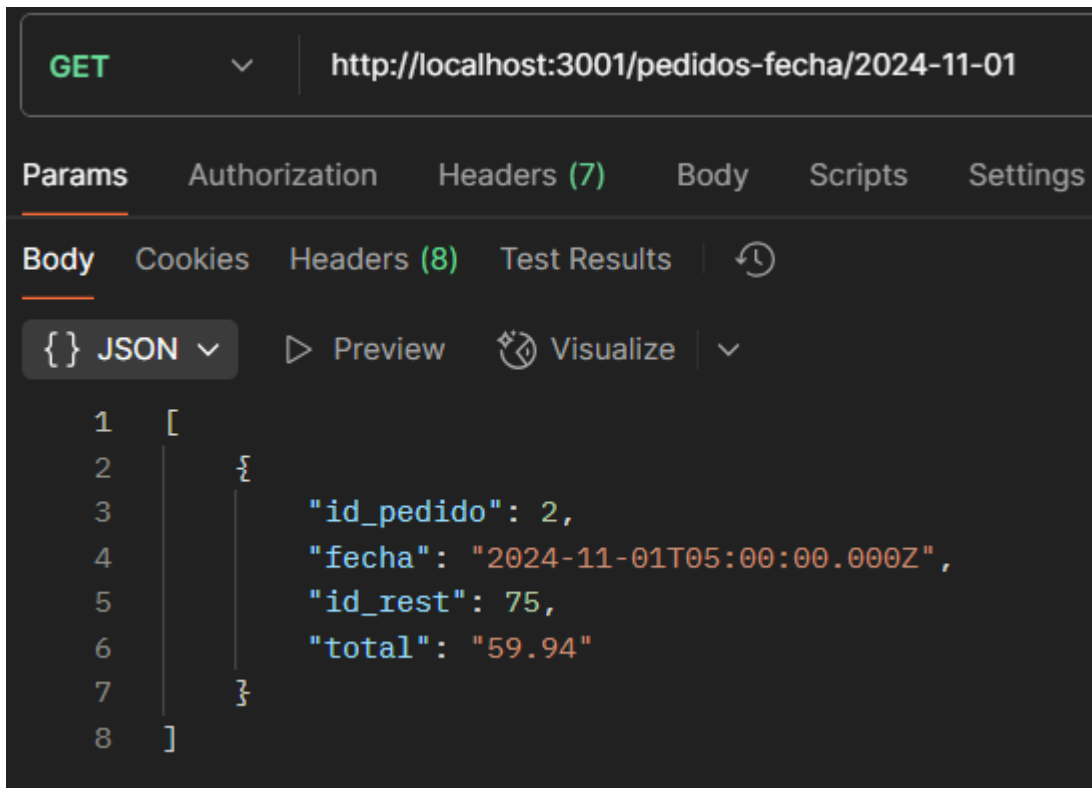
```
// 3. Total de ventas por restaurante
app.get('/ventas-restaurantes', async (req, res) => {
  const result = await client.query(`
    SELECT r.id_rest, r.nombre, SUM(pe.total) AS total_ventas
    FROM Pedido pe
    JOIN Restaurante r ON pe.id_rest = r.id_rest
    GROUP BY r.id_rest, r.nombre
  `);
  res.json(result.rows);
});
```



- Obtener los pedidos realizados en una fecha específica

La consulta busca todos los pedidos que tengan exactamente la misma fecha que se ingresa

```
// 4. Pedidos en una fecha específica
app.get('/pedidos-fecha/:fecha', async (req, res) => {
  const { fecha } = req.params;
  const result = await client.query(`
    SELECT * FROM Pedido
    WHERE fecha = $1
  `, [fecha]);
  res.json(result.rows);
});
```

- Obtener los empleados por rol en un restaurante

En esta consulta, se debe ingresar el id del restaurante y el nombre del rol del empleado, siendo que si algún empleado coincide con estas dos características es mostrado

```
// 5. Empleados por rol en un restaurante
app.get('/empleados-rol/:id_rest/:rol', async (req, res) => {
  const { id_rest, rol } = req.params;
  const result = await client.query(`
    SELECT * FROM Empleado
    WHERE id_rest = $1 AND rol = $2
  `, [id_rest, rol]);
  res.json(result.rows);
});
```

GET

▼

http://localhost:3001/empleados-rol/97/Mesero

Params

Authorization

Headers (7)

Body

Scripts

Settings

Body

Cookies

Headers (8)

Test Results

↺

{ } JSON ▼

▶ Preview

🔗 Visualize

▼

1

[

2

|

3

|

4

|

5

|

6

|

7

|

8

]

{

"id_empleado": 6,

"nombre": "Empleado_6",

"rol": "Mesero",

"id_rest": 97

}