Parcial 2

Bases de datos masivas

UNIMINUTO zipaquira

Juan manuel moncada felix

809397

23 de abril del del 2025

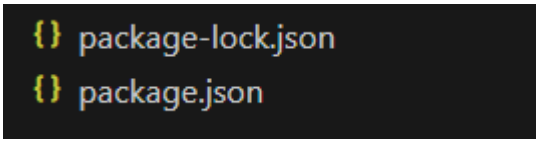
1. **Creamos un nuevo proyecto donde creamos dos documentos, db.js e index.js**

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

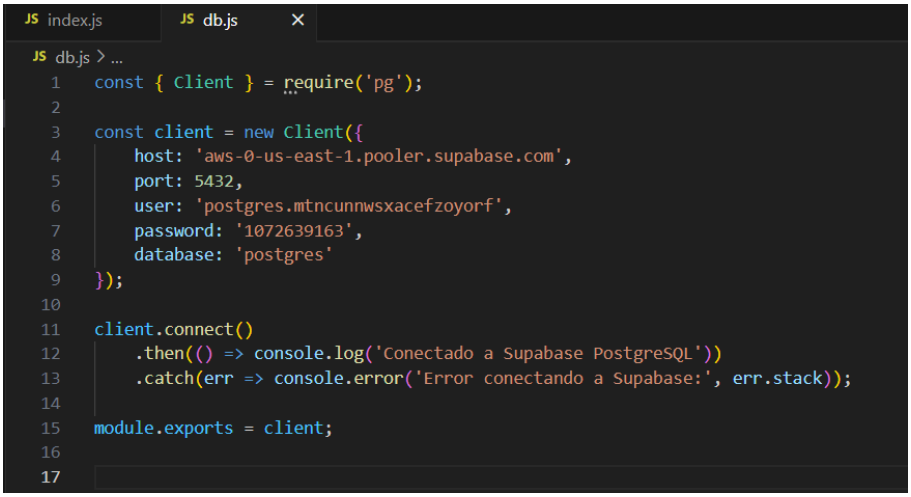
```
PS C:\Users\monca\Desktop\parcial2BD> npm install express pg cors

added 14 packages, and audited 83 packages in 4s

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

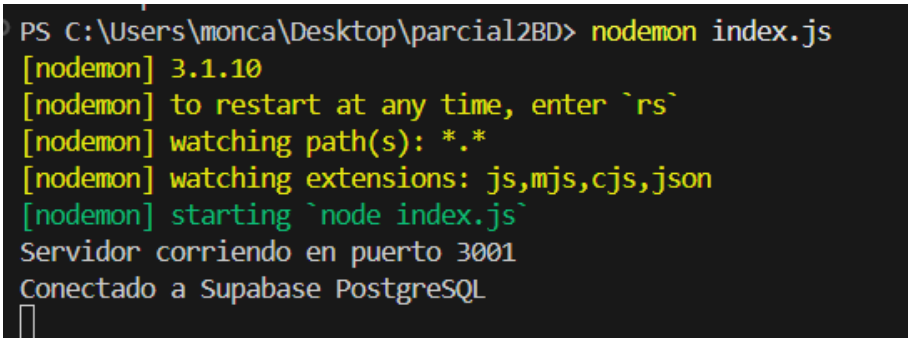
```
PS C:\Users\monca\Desktop\parcial2BD> npm init -y
Wrote to C:\Users\monca\Desktop\parcial2BD\package.json:

{
  "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
```

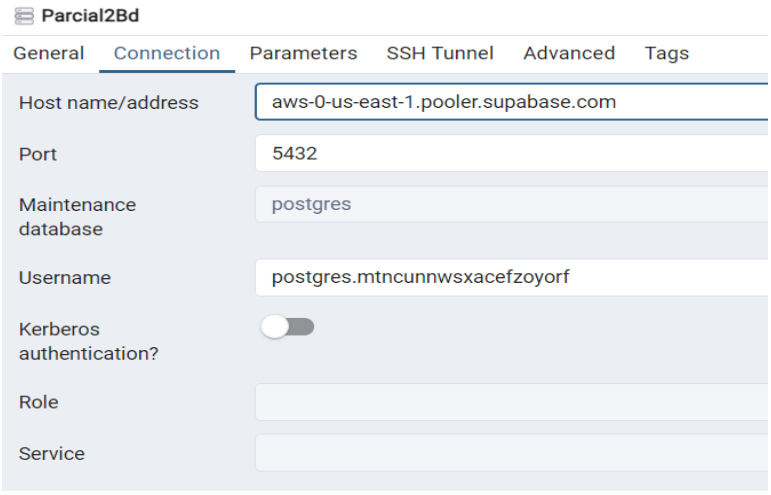


3. **En nuestro db.js, creamos la conexión con nuestro proyecto en supabase**

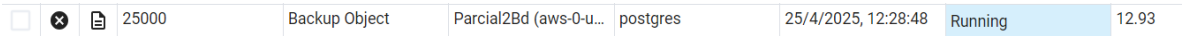




4. **Realizamos la conexión con PGadmin 4, siendo este un puente y ayudarnos a crear el backup**



- **Backup**



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

```sql
CREATE TABLE Restaurante (
    id_rest INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(100),
    ciudad VARCHAR(100),
    direccion VARCHAR(150),
    fecha_apertura DATE
);
```

```sql
CREATE TABLE Restaurante (
    id_rest SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    ciudad VARCHAR(100),
    direccion VARCHAR(150),
    fecha_apertura DATE
);
```

```sql
CREATE TABLE Empleado (
    id_empleado SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    rol VARCHAR(50),
    id_rest INT REFERENCES Restaurante(id_rest)
);
```

```sql
CREATE TABLE Producto (
    id_prod SERIAL PRIMARY KEY,
    nombre VARCHAR(100),
    precio NUMERIC(10,2)
);
```

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

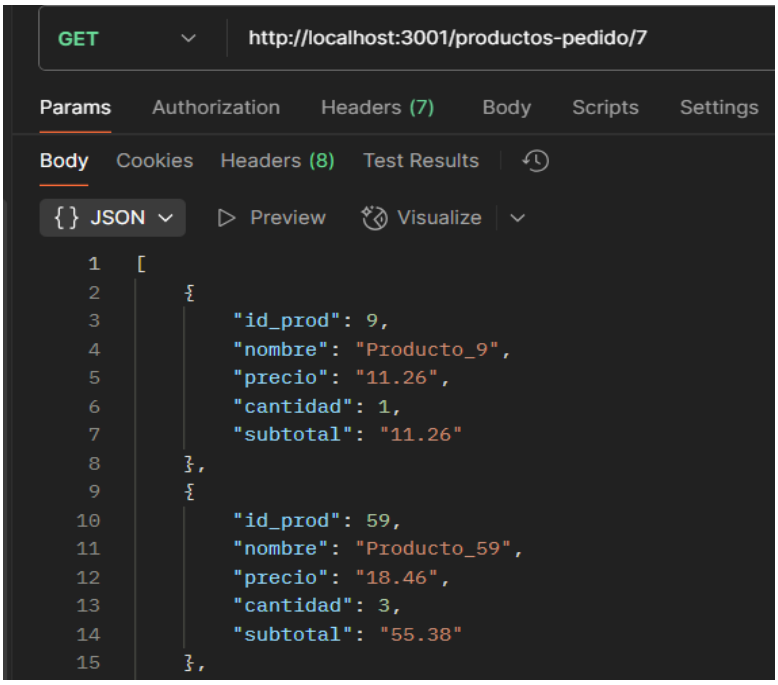
6. En el index.js configuraremos la conexión con postman y asignamos cada uno de los parámetros necesarios

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 trabes 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,

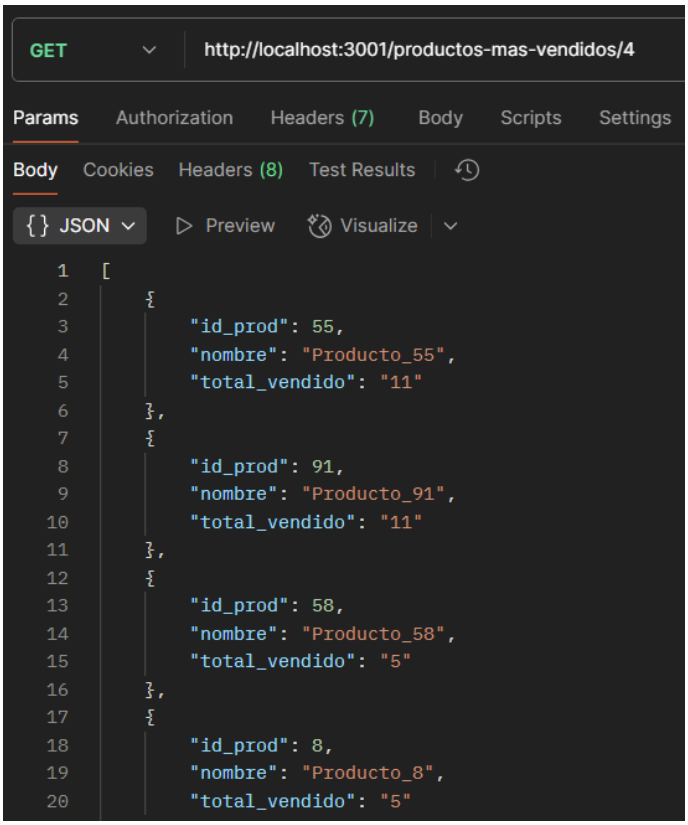
```
// 1. Obtener productos de un pedido
app.get('/productos-pedido/:id', async (req, res) => {
  const { id } = req.params;
  const result = await client.query(`
    SELECT p.id_prod, p.nombre, p.precio, dp.cantidad, dp.subtotal
    FROM DetallePedido dp
    JOIN Producto p ON dp.id_prod = p.id_prod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

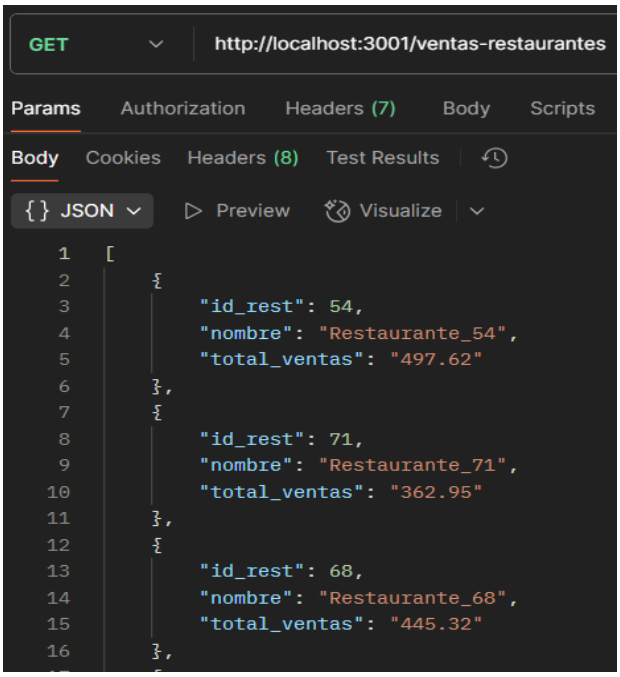
```
// 2. Productos más vendidos (más de X unidades)
app.get('/productos-mas-vendidos/:min', async (req, res) => {
  const { min } = req.params;
  const result = await client.query(`
    SELECT p.id_prod, p.nombre, SUM(dp.cantidad) AS total_vendido
    FROM DetallePedido dp
    JOIN Producto p ON dp.id_prod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

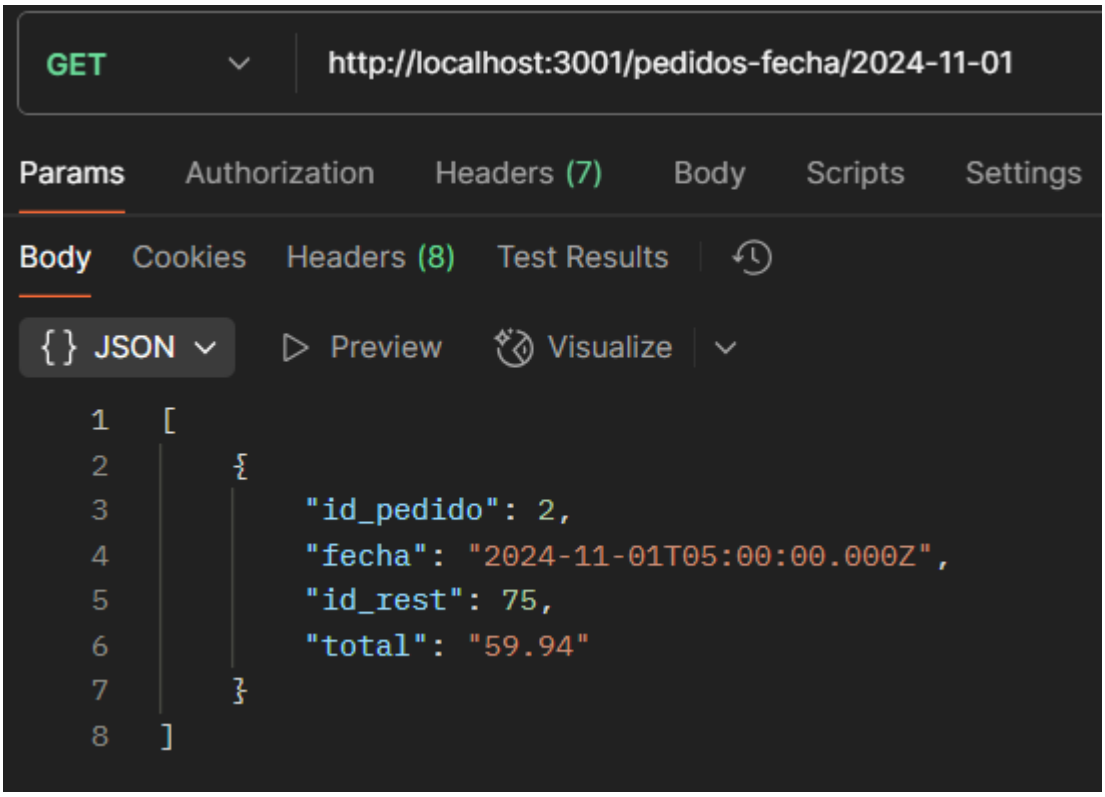
```
// 3. Total de ventas por restaurante
app.get('/ventas-restaurantes', async (req, res) => {
  const result = await client.query(`
    SELECT r.id_rest, r.nombre, SUM(pe.total) AS total_ventas
    FROM Pedido pe
    JOIN Restaurante r ON pe.id_rest = r.id_rest
    GROUP BY r.id_rest, r.nombre
  `);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

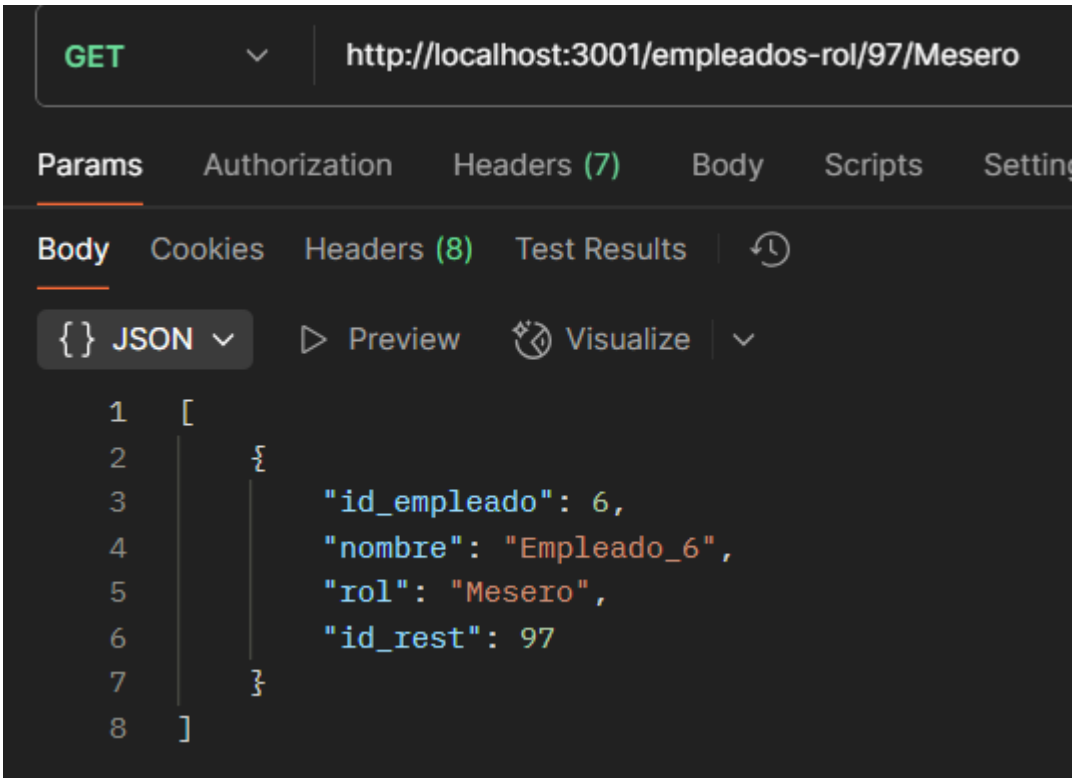
```
// 4. Pedidos en una fecha específica
app.get('/pedidos-fecha/:fecha', async (req, res) => {
  const { fecha } = req.params;
  const result = await client.query(`
    SELECT * FROM Pedido
    WHERE fecha = $1
  `, [fecha]);
  res.json(result.rows);
});
```



- 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 cons 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
Body
Cookies
Headers (8)
Test Results
JSON
Preview
Visualize
[
{
"id_empleado": 6,
"nombre": "Empleado_6",
"rol": "Mesero",
"id_rest": 97
}
]