**Parcial No. 2 – “Comidas Rápidas”**

**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, 25 de Abril, 2025**

**ÍNDICE DE CONTENIDO**

[Conexión a base de datos PostgreSQL (Supabase) 4](#_Toc196449802)

[Creamos una carpeta donde se alojará los archivos relacionados a la base de datos: 4](#_Toc196449803)

[Ya en Supabase creamos el proyecto: 4](#_Toc196449804)

[En la conexión tenemos en cuenta estos parámetros: 4](#_Toc196449805)

[Ya en el editor de código, creamos los archivos que nos ayudarán con la parte funcional backend de la base de datos, conexión y funcionalidades que programaremos para que funcionen con postman 5](#_Toc196449806)

[Hacemos la conexión con Supabase con el fichero desde Visual, teniendo en cuenta los parámetros de la conexión: 5](#_Toc196449807)

[Configuramos el cuerpo del CRUD, pero primero nos fijamos en que la conexión sea exitosa: 6](#_Toc196449808)

[Escucha del puerto: 7](#_Toc196449809)

[Y al final, he aquí el modelo relacional: 8](#_Toc196449810)

[Crear rutas y controladores con Express para cada entidad (CRUD) 9](#_Toc196449811)

[CREATE datos restaurante: >\_http://localhost:3020/api/restaurantes\_< 9](#_Toc196449812)

[READ datos restaurante: >\_http://localhost:3020/api/restaurantes\_< 10](#_Toc196449813)

[UPDATE datos restaurante: >\_http://localhost:3020/api/restaurantes/##\_< 11](#_Toc196449814)

[DELETE datos restaurante: >\_http://localhost:3020/api/restaurantes/##\_< 12](#_Toc196449815)

[CREATE datos empleado: >\_http://localhost:3020/api/empleados\_< 13](#_Toc196449816)

[READ datos empleado: >\_http://localhost:3020/api/empleados\_< 14](#_Toc196449817)

[UPDATE datos empleado: >\_http://localhost:3020/api/empleados/##\_< 15](#_Toc196449818)

[DELETE datos empleado: >\_http://localhost:3020/api/empleados/##\_< 16](#_Toc196449819)

[CREATE datos producto: >\_http://localhost:3020/api/productos\_< 17](#_Toc196449820)

[READ datos producto: >\_http://localhost:3020/api/productos\_< 18](#_Toc196449821)

[UPDATE datos producto: >\_http://localhost:3020/api/productos/##\_< 19](#_Toc196449822)

[DELETE datos producto: >\_http://localhost:3020/api/ productos/##\_< 20](#_Toc196449823)

[CREATE datos pedido: >\_http://localhost:3020/api/pedidos\_< 21](#_Toc196449824)

[READ datos pedido: >\_http://localhost:3020/api/ pedidos\_< 22](#_Toc196449825)

[UPDATE datos pedido: >\_http://localhost:3020/api/ pedidos/##\_< 23](#_Toc196449826)

[DELETE datos pedido: >\_http://localhost:3020/api/ pedidos/##\_< 24](#_Toc196449827)

[CREATE datos detalle\_pedido: >\_http://localhost:3020/api/detalles\_< 25](#_Toc196449828)

[READ datos detalle\_pedido: >\_http://localhost:3020/api/detalles\_< 26](#_Toc196449829)

[UPDATE datos detalles\_pedido: >\_http://localhost:3020/api/detalles/##\_< 27](#_Toc196449830)

[DELETE datos detalle\_pedido: >\_http://localhost:3020/api/detalles/##\_< 28](#_Toc196449831)

[Consultas Nativas: Con Ruta Específica, y Petición De Postman 29](#_Toc196449832)

[Obtener todos los productos de un pedido específico: 29](#_Toc196449833)

[Obtener los productos más vendidos (más de X unidades): 30](#_Toc196449834)

[Obtener el total de ventas por restaurante: 31](#_Toc196449835)

[Obtener los pedidos realizados en una fecha específica: 32](#_Toc196449836)

[Obtener los empleados por rol en un restaurante: 33](#_Toc196449837)

[Anexos 34](#_Toc196449838)

[100 Registros Por Tabla: 34](#_Toc196449839)

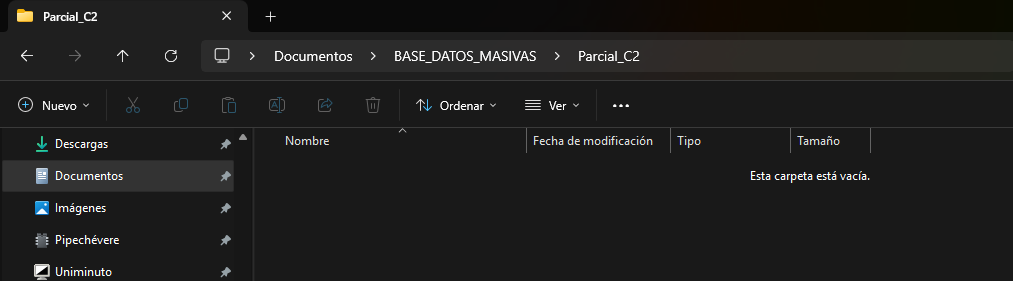
[Modelo De Base De Datos De Supabase: 34](#_Toc196449840)

[Ficheros de la base de datos: 34](#_Toc196449841)

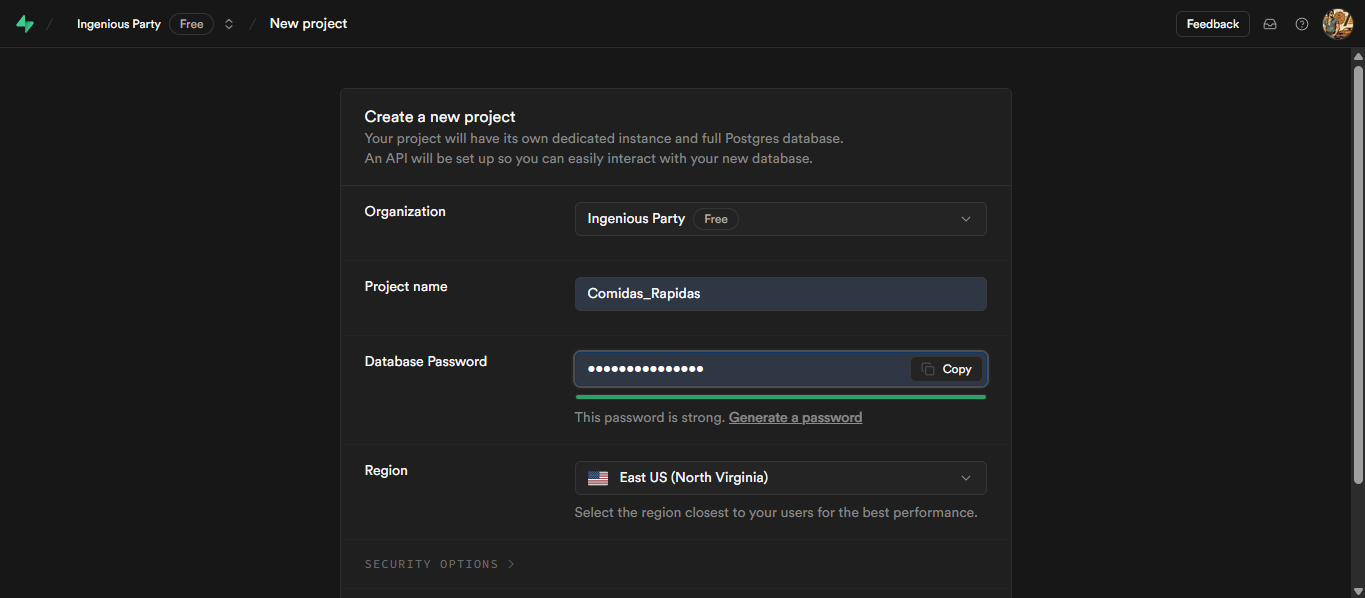
[Colección de Postman: 34](#_Toc196449842)

# Conexión a base de datos PostgreSQL (Supabase)

## Creamos una carpeta donde se alojará los archivos relacionados a la base de datos:



## Ya en Supabase creamos el proyecto:



## En la conexión tenemos en cuenta estos parámetros:

Texto

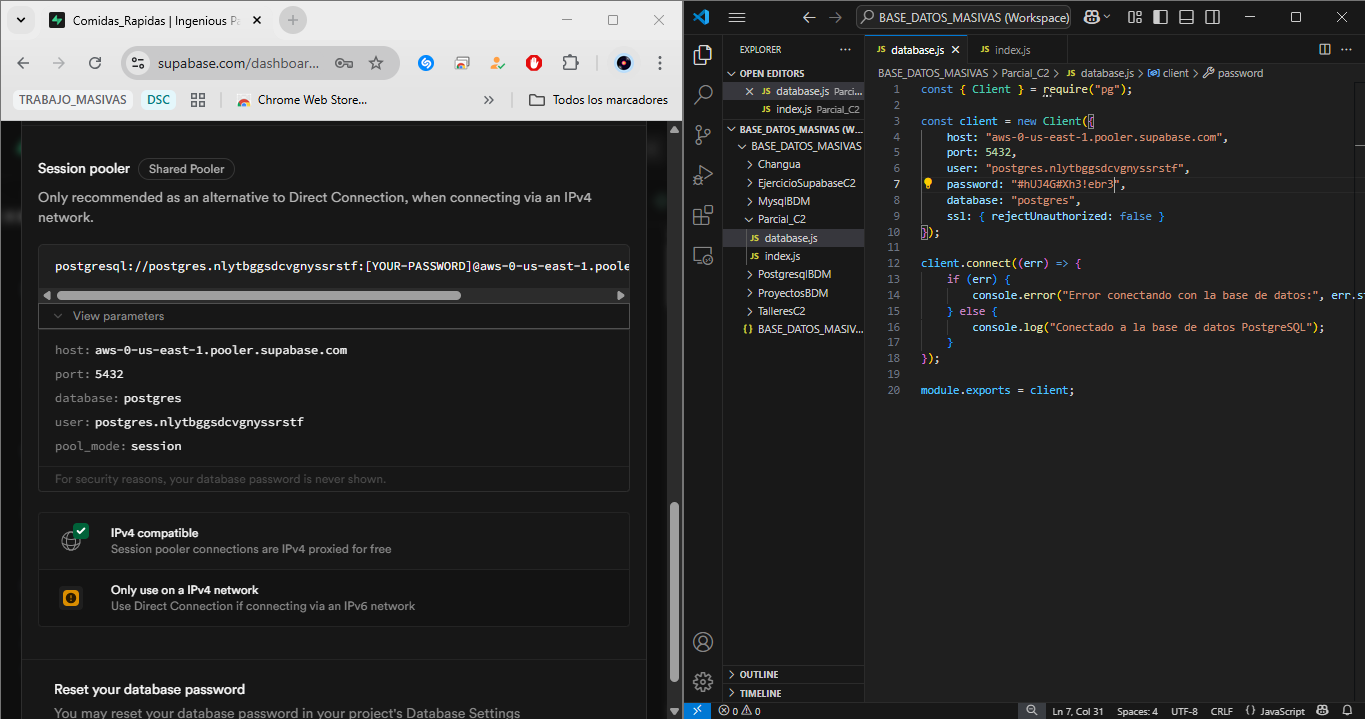
El contenido generado por IA puede ser incorrecto.

## Ya en el editor de código, creamos los archivos que nos ayudarán con la parte funcional backend de la base de datos, conexión y funcionalidades que programaremos para que funcionen con postman

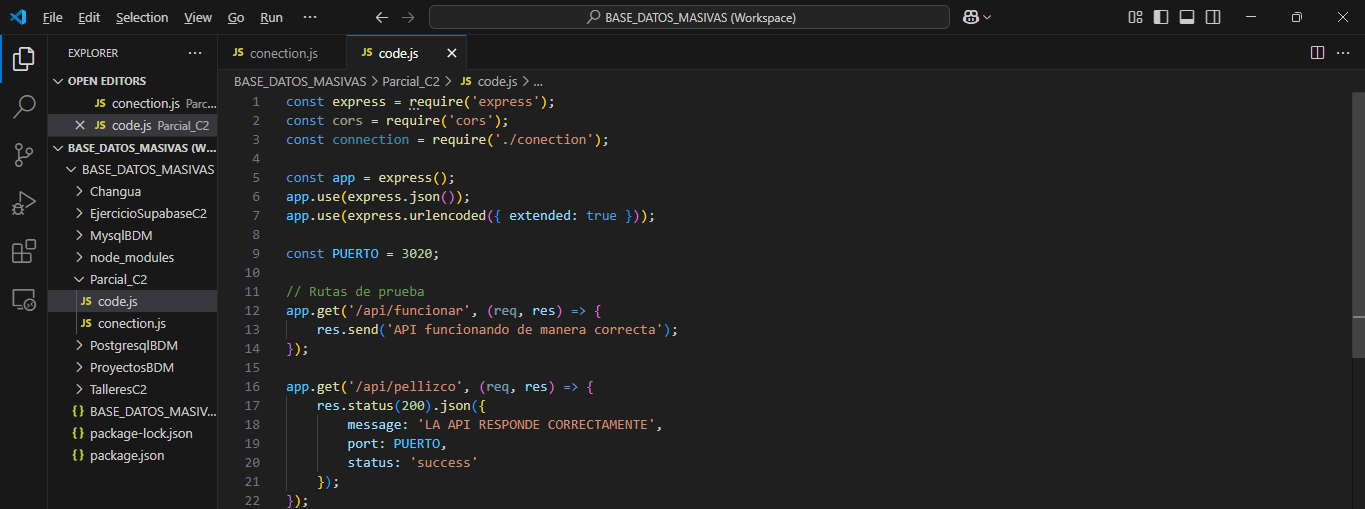
Captura de pantalla de un celular

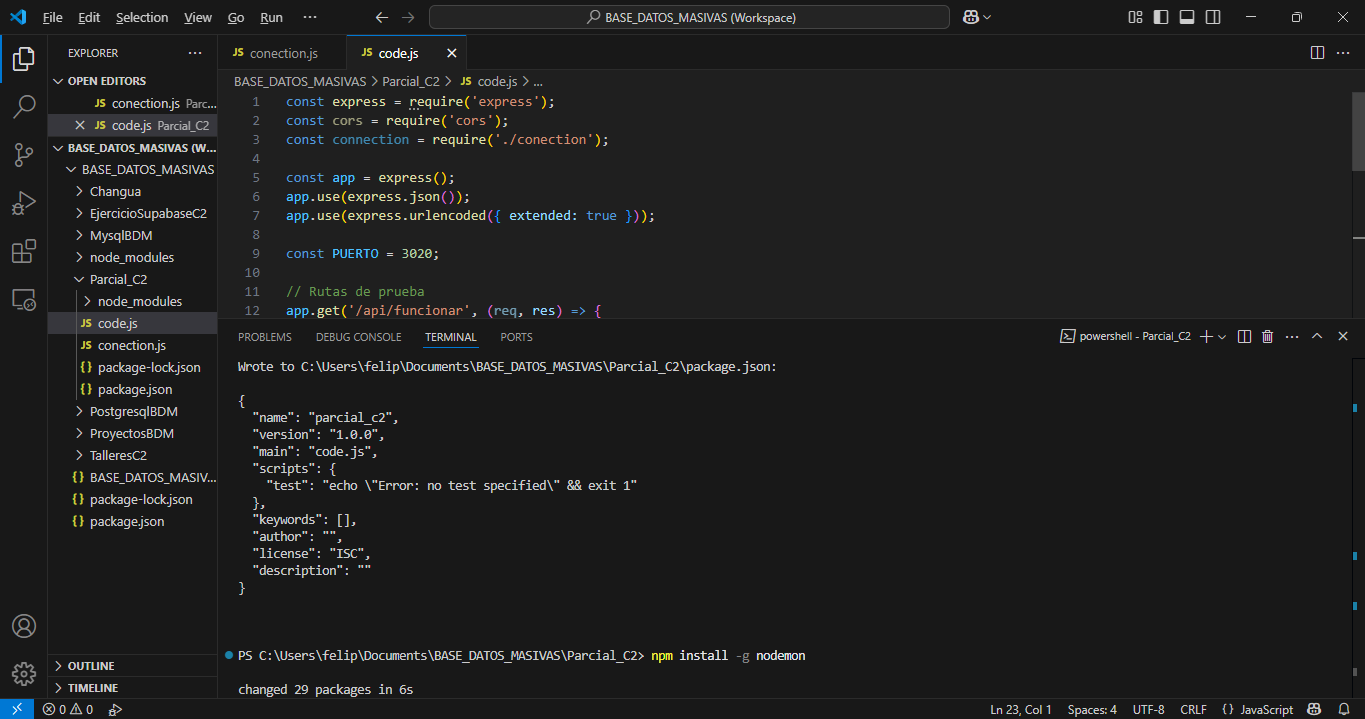
El contenido generado por IA puede ser incorrecto.

## Hacemos la conexión con Supabase con el fichero desde Visual, teniendo en cuenta los parámetros de la conexión:



## Configuramos el cuerpo del CRUD, pero primero nos fijamos en que la conexión sea exitosa:





Texto

El contenido generado por IA puede ser incorrecto.

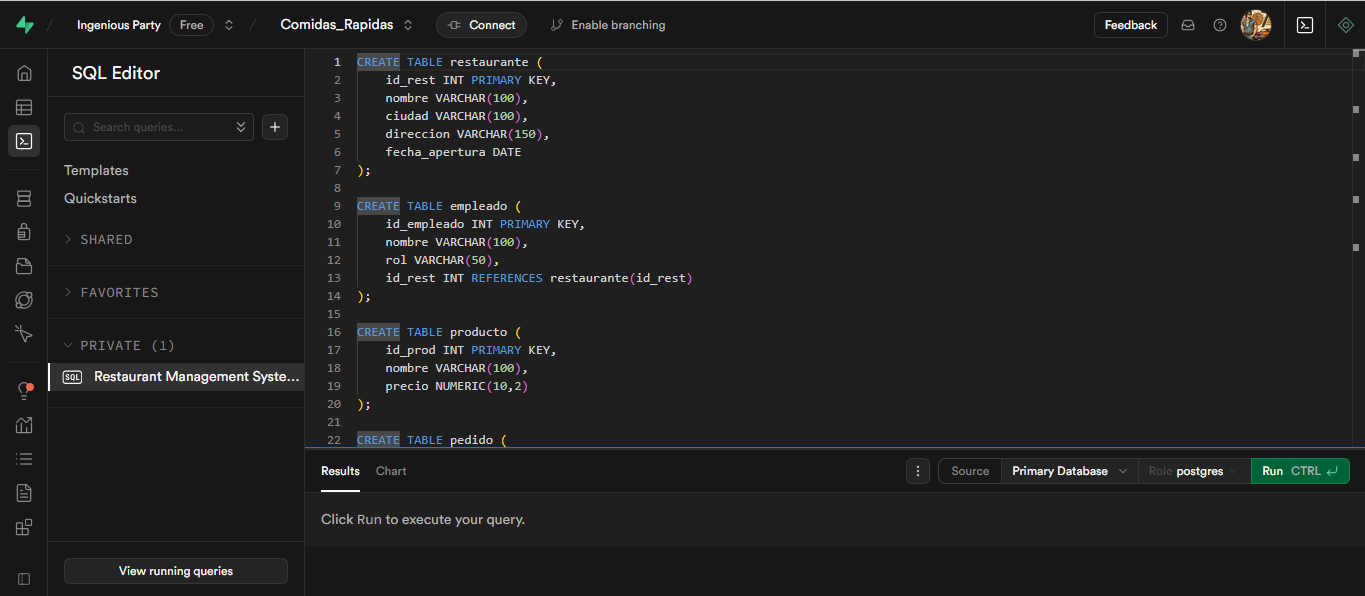
## Escucha del puerto:

Aquí importamos las dependencias express y enlazamos con el fichero de la base de datos “Conection.js”; además le pedimos que nos responda por un puerto, que en este caso es el 3020. Ya de primera instancia encontramos un “API” que se encarga de escuchar y señalarnos por medio del terminal de visual studio code, si efectivamente está conectada a SupaBase.

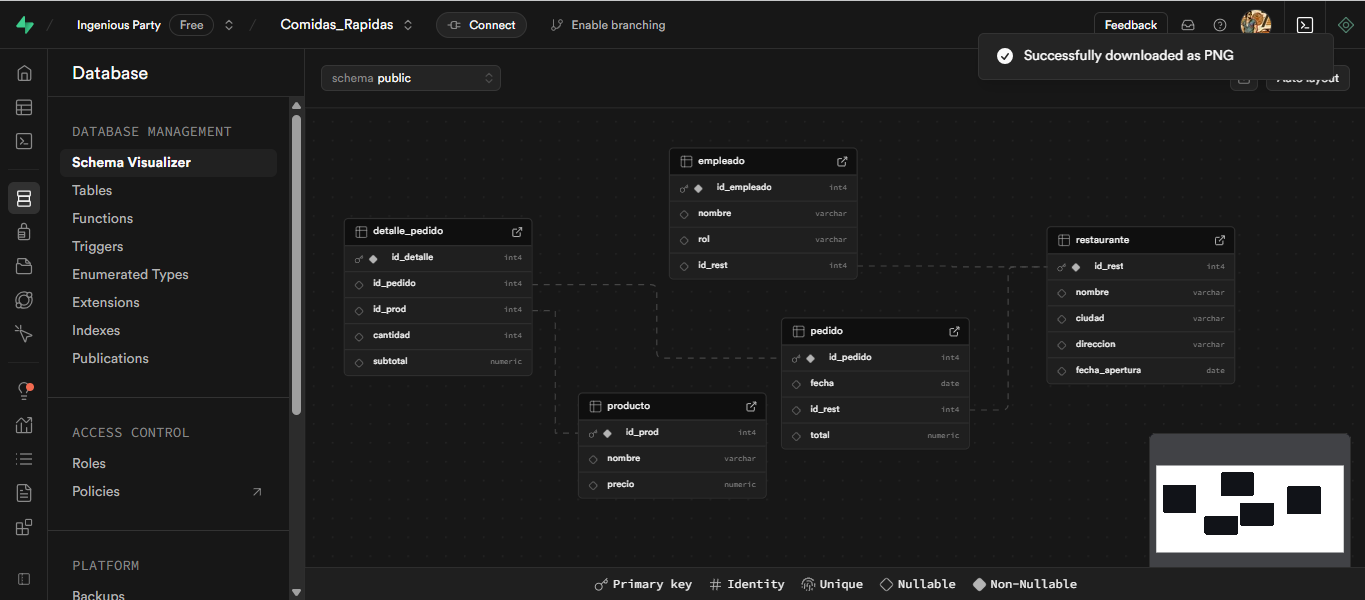
Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Creamos las tablas desde Supabase:



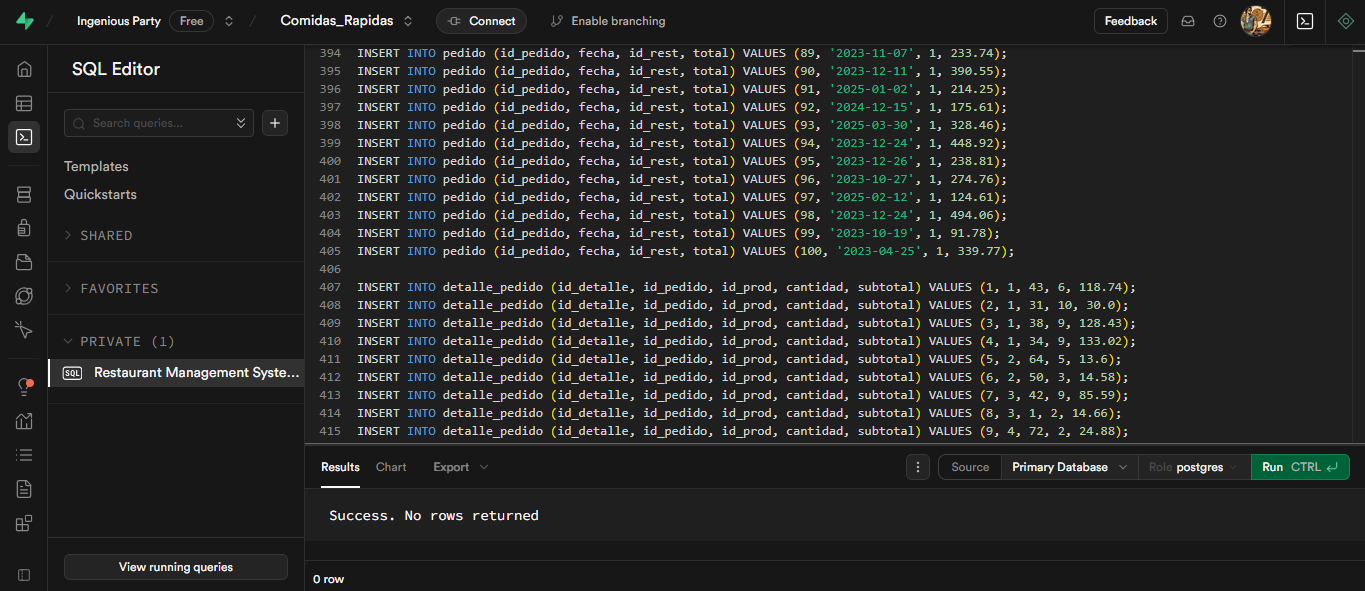
## Y al final, he aquí el modelo relacional:



Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Ingresamos 100 registros a cada tabla, por medio del SQL Editor:



# Crear rutas y controladores con Express para cada entidad (CRUD)

## CREATE datos restaurante: >\_http://localhost:3020/api/restaurantes\_<

Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de hacer un registro de ingreso de un **restaurante**, donde todos sus parámetros son ingresados; y como todo funciona por medio de una consulta SQL, lo que hacemos es guardarla en una variable dependiente a la estructura de la tabla; para luego hacer un intento de captura exitosa, y si no de lo contrario arrojarnos que no se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, y como queremos ingresar hacemos uso de la estructura para indicar los valores de cada columna de la tabla:

Texto

El contenido generado por IA puede ser incorrecto.

## READ datos restaurante: >\_http://localhost:3020/api/restaurantes\_<

Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de botarnos todos los registros existentes de todos los **restaurantes**, donde simplemente le pasamos una sentencia SQL dentro del intento de captura exitosa, y si no de lo contrario nos lanza que no se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, y si es correcta la sintaxis, nos arroja todos los **restaurantes** registrados hasta el momento:

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

## UPDATE datos restaurante: >\_http://localhost:3020/api/restaurantes/##\_<

Texto

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de hacer una sobreescritura de un ingreso de un **restaurante** ya creado, donde todos sus parámetros son modificados; y como todo funciona por medio de una consulta SQL, lo que hacemos es guardarla en una variable dependiente a la estructura de la tabla; para luego hacer un intento de captura exitosa, y si no de lo contrario arrojarnos que no se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, junto al id a modificar, y como queremos modificar hacemos uso de la estructura para indicar los valores de cada columna de la tabla:

Texto

El contenido generado por IA puede ser incorrecto.

## DELETE datos restaurante: >\_http://localhost:3020/api/restaurantes/##\_<

Texto

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de hacer una eliminación de un **restaurante** ya creado, donde solo por medio de la ruta ingresamos el id de este mismo; y como todo funciona por medio de una consulta SQL, lo que hacemos es guardarla en el intento de captura exitosa, y si no de lo contrario nos arrojarnos que se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, junto al id del **restaurante** a eliminar.

Texto

El contenido generado por IA puede ser incorrecto.

## CREATE datos empleado: >\_http://localhost:3020/api/empleados\_<

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de hacer un registro de ingreso de un **empleado**, donde todos sus parámetros son ingresados; y como todo funciona por medio de una consulta SQL, lo que hacemos es guardarla en una variable dependiente a la estructura de la tabla; para luego hacer un intento de captura exitosa, y si no de lo contrario arrojarnos que no se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, y como queremos ingresar hacemos uso de la estructura para indicar los valores de cada columna de la tabla:

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

## READ datos empleado: >\_http://localhost:3020/api/empleados\_<

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de botarnos todos los registros existentes de todos los **empleados**, donde simplemente le pasamos una sentencia SQL dentro del intento de captura exitosa, y si no de lo contrario nos lanza que no se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, y si es correcta la sintaxis, nos arroja todos los **empleados** registrados hasta el momento:

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

## UPDATE datos empleado: >\_http://localhost:3020/api/empleados/##\_<

Texto

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de hacer una sobreescritura de un ingreso de un **empleado** ya creado, donde todos sus parámetros son modificados; y como todo funciona por medio de una consulta SQL, lo que hacemos es guardarla en una variable dependiente a la estructura de la tabla; para luego hacer un intento de captura exitosa, y si no de lo contrario arrojarnos que no se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, junto al id a modificar, y como queremos modificar hacemos uso de la estructura para indicar los valores de cada columna de la tabla:

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

## DELETE datos empleado: >\_http://localhost:3020/api/empleados/##\_<

Texto

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de hacer una eliminación de un **empleado** ya creado, donde solo por medio de la ruta ingresamos el id de este mismo; y como todo funciona por medio de una consulta SQL, lo que hacemos es guardarla en el intento de captura exitosa, y si no de lo contrario nos arrojarnos que se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, junto al id del **empleado** a eliminar:

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

## CREATE datos producto: >\_http://localhost:3020/api/productos\_<

Imagen que contiene Texto

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de hacer un registro de ingreso de un **producto**, donde todos sus parámetros son ingresados; y como todo funciona por medio de una consulta SQL, lo que hacemos es guardarla en una variable dependiente a la estructura de la tabla; para luego hacer un intento de captura exitosa, y si no de lo contrario arrojarnos que no se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, y como queremos ingresar hacemos uso de la estructura para indicar los valores de cada columna de la tabla:

Texto

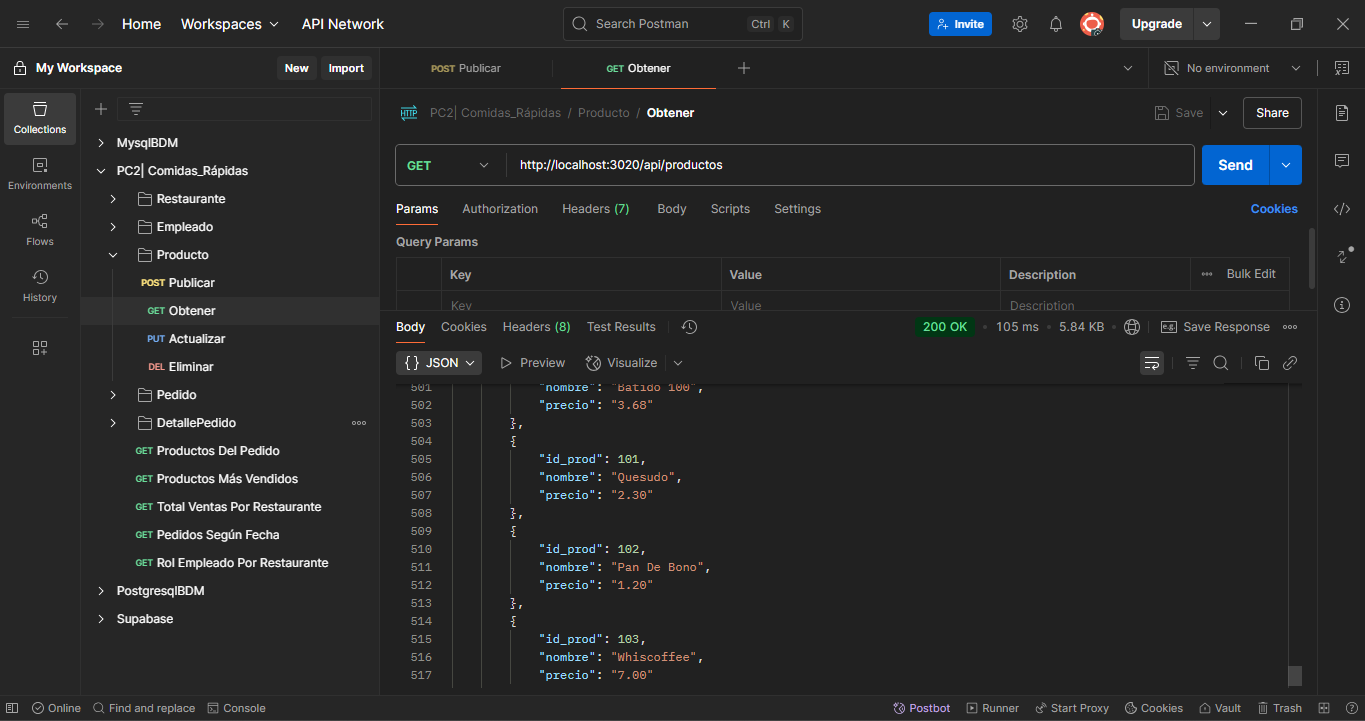
El contenido generado por IA puede ser incorrecto.

## READ datos producto: >\_http://localhost:3020/api/productos\_<

Texto

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de botarnos todos los registros existentes de todos los **productos**, donde simplemente le pasamos una sentencia SQL dentro del intento de captura exitosa, y si no de lo contrario nos lanza que no se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, y si es correcta la sintaxis, nos arroja todos los **productos** registrados hasta el momento:



## UPDATE datos producto: >\_http://localhost:3020/api/productos/##\_<

Texto

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de hacer una sobreescritura de un ingreso de un **producto** ya creado, donde todos sus parámetros son modificados; y como todo funciona por medio de una consulta SQL, lo que hacemos es guardarla en una variable dependiente a la estructura de la tabla; para luego hacer un intento de captura exitosa, y si no de lo contrario arrojarnos que no se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, junto al id a modificar, y como queremos modificar hacemos uso de la estructura para indicar los valores de cada columna de la tabla:

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

## DELETE datos producto: >\_http://localhost:3020/api/ productos/##\_<

Texto

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de hacer una eliminación de un **producto** ya creado, donde solo por medio de la ruta ingresamos el id de este mismo; y como todo funciona por medio de una consulta SQL, lo que hacemos es guardarla en el intento de captura exitosa, y si no de lo contrario nos arrojarnos que se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, junto al id del **producto** a eliminar:

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

## CREATE datos pedido: >\_http://localhost:3020/api/pedidos\_<

Imagen que contiene Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de hacer un registro de ingreso de un **pedido**, donde todos sus parámetros son ingresados; y como todo funciona por medio de una consulta SQL, lo que hacemos es guardarla en una variable dependiente a la estructura de la tabla; para luego hacer un intento de captura exitosa, y si no de lo contrario arrojarnos que no se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, y como queremos ingresar hacemos uso de la estructura para indicar los valores de cada columna de la tabla:

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

## READ datos pedido: >\_http://localhost:3020/api/ pedidos\_<

Texto

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de botarnos todos los registros existentes de todos los **pedidos**, donde simplemente le pasamos una sentencia SQL dentro del intento de captura exitosa, y si no de lo contrario nos lanza que no se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, y si es correcta la sintaxis, nos arroja todos los **pedidos** registrados hasta el momento:

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

## UPDATE datos pedido: >\_http://localhost:3020/api/ pedidos/##\_<

Texto

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de hacer una sobreescritura de un ingreso de un **pedido** ya creado, donde todos sus parámetros son modificados; y como todo funciona por medio de una consulta SQL, lo que hacemos es guardarla en una variable dependiente a la estructura de la tabla; para luego hacer un intento de captura exitosa, y si no de lo contrario arrojarnos que no se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, junto al id a modificar, y como queremos modificar hacemos uso de la estructura para indicar los valores de cada columna de la tabla:

Texto

El contenido generado por IA puede ser incorrecto.

## DELETE datos pedido: >\_http://localhost:3020/api/ pedidos/##\_<

Texto

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de hacer una eliminación de un **pedido** ya creado, donde solo por medio de la ruta ingresamos el id de este mismo; y como todo funciona por medio de una consulta SQL, lo que hacemos es guardarla en el intento de captura exitosa, y si no de lo contrario nos arrojarnos que se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, junto al id del **pedido** a eliminar:

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

## CREATE datos detalle\_pedido: >\_http://localhost:3020/api/detalles\_<

Patrón de fondo

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de hacer un registro de ingreso de un **detalle de pedido**, donde todos sus parámetros son ingresados; y como todo funciona por medio de una consulta SQL, lo que hacemos es guardarla en una variable dependiente a la estructura de la tabla; para luego hacer un intento de captura exitosa, y si no de lo contrario arrojarnos que no se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, y como queremos ingresar hacemos uso de la estructura para indicar los valores de cada columna de la tabla:

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

## READ datos detalle\_pedido: >\_http://localhost:3020/api/detalles\_<

Texto

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de botarnos todos los registros existentes de todos los **detalles de pedido**, donde simplemente le pasamos una sentencia SQL dentro del intento de captura exitosa, y si no de lo contrario nos lanza que no se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, y si es correcta la sintaxis, nos arroja todos los **detalles de pedido** registrados hasta el momento:

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

## UPDATE datos detalles\_pedido: >\_http://localhost:3020/api/detalles/##\_<

Texto

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de hacer una sobreescritura de un ingreso de un **detalle de pedido** ya creado, donde todos sus parámetros son modificados; y como todo funciona por medio de una consulta SQL, lo que hacemos es guardarla en una variable dependiente a la estructura de la tabla; para luego hacer un intento de captura exitosa, y si no de lo contrario arrojarnos que no se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, junto al id a modificar, y como queremos modificar hacemos uso de la estructura para indicar los valores de cada columna de la tabla:

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

## DELETE datos detalle\_pedido: >\_http://localhost:3020/api/detalles/##\_<

Texto

El contenido generado por IA puede ser incorrecto.

Creamos un API capaz de hacer una eliminación de un **detalle de pedido** ya creado, donde solo por medio de la ruta ingresamos el id de este mismo; y como todo funciona por medio de una consulta SQL, lo que hacemos es guardarla en el intento de captura exitosa, y si no de lo contrario nos arrojarnos que se pudo lograr el registro (claro teniendo en cuenta los tipos de errores, para ver en que lo estamos configurando mal). Ya en Postman utilizamos la dirección creada al principio del app.get, junto al id del **detalle de pedido** a eliminar:

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

# Consultas Nativas: Con Ruta Específica, y Petición De Postman

*Cabe recalcar que como son consultas SQL que obtenemos de las tablas (jalamos información) son automáticamente todas peticiones de tipo* ***[GET]***

## 

## Obtener todos los productos de un pedido específico:

SELECT p.id\_prod, p.nombre, p.precio, dp.cantidad, dp.subtotal FROM detalle\_pedido dp JOIN producto p ON dp.id\_prod = p.id\_prod WHERE dp.id\_pedido = $1

Se selecciona los valores de un detalle de pedido, donde este mismo sea de un pedido, al que usuario quiera obtener; se indica con el respectivo id del pedido a averiguar

Texto

El contenido generado por IA puede ser incorrecto.

Podemos obtener los resultados por la consulta nativa solicitada, por medio de un API que de una ruta para poder diferenciarla de todas las demás; donde esta requiere de la estructura previa de la tabla (columnas), es por ello que requiere parámetros y es guardada en una variable, que junto a los parámetros son ingresados, en un try catch, donde evalúa si efectivamente está bien definidos los parámetros. Y esto lo podemos evidenciar en Postman:

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

## Obtener los productos más vendidos (más de X unidades):

SELECT p.id\_prod, p.nombre, SUM(dp.cantidad) AS total\_vendido FROM detalle\_pedido dp JOIN producto p ON dp.id\_prod = p.id\_prod GROUP BY p.id\_prod, p.nombre HAVING SUM(dp.cantidad) > $1 ORDER BY total\_vendido DESC

Se listan los productos que más se han vendido por medio de su id y su nombre, y se van sumando todas sus cantidades en los pedidos registrados. Aun así, solo se muestran los que hayan vendido igual o más unidades al número que el usuario haya indicado por medio del parámetro de variable ($1). El resultado se ordena del más vendido al de menor cantidad. Se van filtrando por un mínimo de ventas.

Imagen que contiene circuito

El contenido generado por IA puede ser incorrecto.

Podemos obtener los resultados por la consulta nativa solicitada, por medio de un API que de una ruta para poder diferenciarla de todas las demás; donde esta requiere de la estructura previa de la tabla (columnas), es por ello que requiere parámetros y es guardada en una variable, que junto a los parámetros son ingresados, en un try catch, donde evalúa si efectivamente está bien definidos los parámetros. Y esto lo podemos evidenciar en Postman:

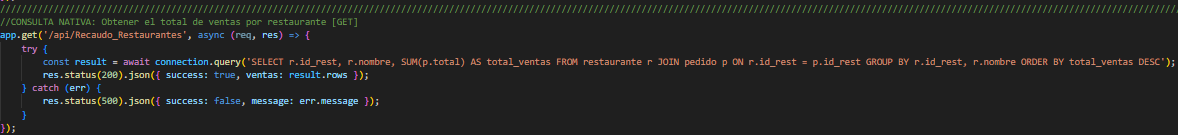
Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

## Obtener el total de ventas por restaurante:

SELECT r.id\_rest, r.nombre, SUM(p.total) AS total\_ventas FROM restaurante r JOIN pedido p ON r.id\_rest = p.id\_rest GROUP BY r.id\_rest, r.nombre ORDER BY total\_ventas DESC

Se selecciona el id del restaurante a evaluar y su respectivo nombre, y nos arroja el total de ventas de éste mismo, por medio de la suma de todos sus de pedidos hechos en cada restaurante, para al final mostrar por orden descendente.



Podemos obtener los resultados por la consulta nativa solicitada, por medio de un API que de una ruta para poder diferenciarla de todas las demás; donde ésta simplemente se evalúa con un try catch, donde evalúa si efectivamente está bien definidos los parámetros. Y esto lo podemos evidenciar en Postman:

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

## Obtener los pedidos realizados en una fecha específica:

SELECT \* FROM pedido WHERE fecha = $1 ORDER BY id\_pedido

Selecciona todos los registros de la tabla pedido, devolviendo todos los pedidos hechos en una fecha específica, que el usuario indique en la ruta de la API ($1). Y se muestra según el orden de ingreso de esa fecha indicada.

Texto

El contenido generado por IA puede ser incorrecto.

Podemos obtener los resultados por la consulta nativa solicitada, por medio de un API que de una ruta para poder diferenciarla de todas las demás; donde esta requiere de la estructura previa de la tabla (columnas), es por ello que requiere parámetros y es guardada en una variable, que junto a los parámetros son ingresados, en un try catch, donde evalúa si efectivamente está bien definidos los parámetros. Y esto lo podemos evidenciar en Postman:

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

## Obtener los empleados por rol en un restaurante:

SELECT \* FROM empleado WHERE id\_rest = $1 AND rol = $2

Selecciona todos los registros de la tabla empleados, y devuelve solo aquellos que el usuario le indique el id del restaurante ($1), y el rol que se quiera investigar ($2), al igual que el anterior si muestra más de una coincidencia muestra por orden de registro en la tabla.

Texto

El contenido generado por IA puede ser incorrecto.

Podemos obtener los resultados por la consulta nativa solicitada, por medio de un API que de una ruta para poder diferenciarla de todas las demás; donde esta requiere de la estructura previa de la tabla (columnas), es por ello que requiere parámetros y es guardada en una variable, que junto a los parámetros son ingresados, en un try catch, donde evalúa si efectivamente está bien definidos los parámetros. Y esto lo podemos evidenciar en Postman:

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

# Anexos

## 100 Registros Por Tabla:



## Modelo De Base De Datos De Supabase:

Interfaz de usuario gráfica

El contenido generado por IA puede ser incorrecto.

## Ficheros de la base de datos:



## Colección de Postman:

