

Parcial II

Cristian Camilo Tique Tapiero

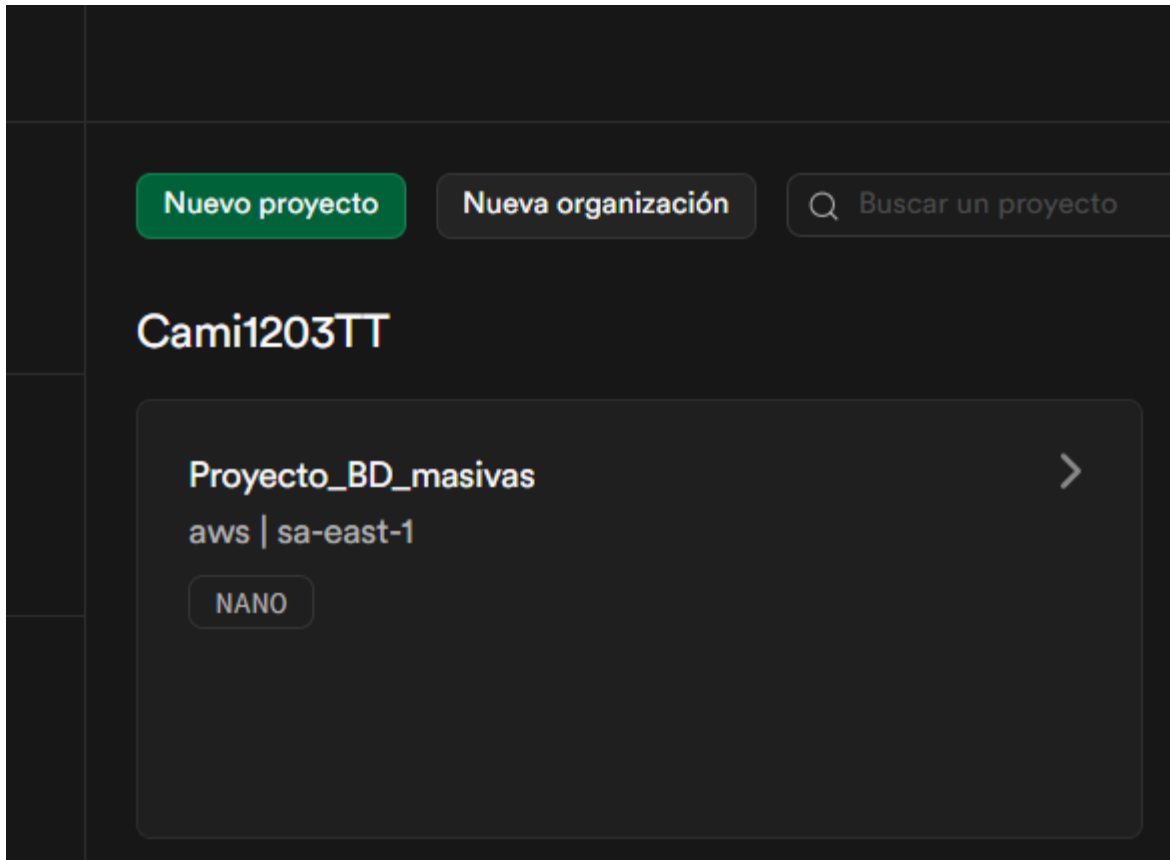
Ingeniería de sistemas, Corporación Universitaria Minuto de Dios

60747: Bases de Datos Masivas

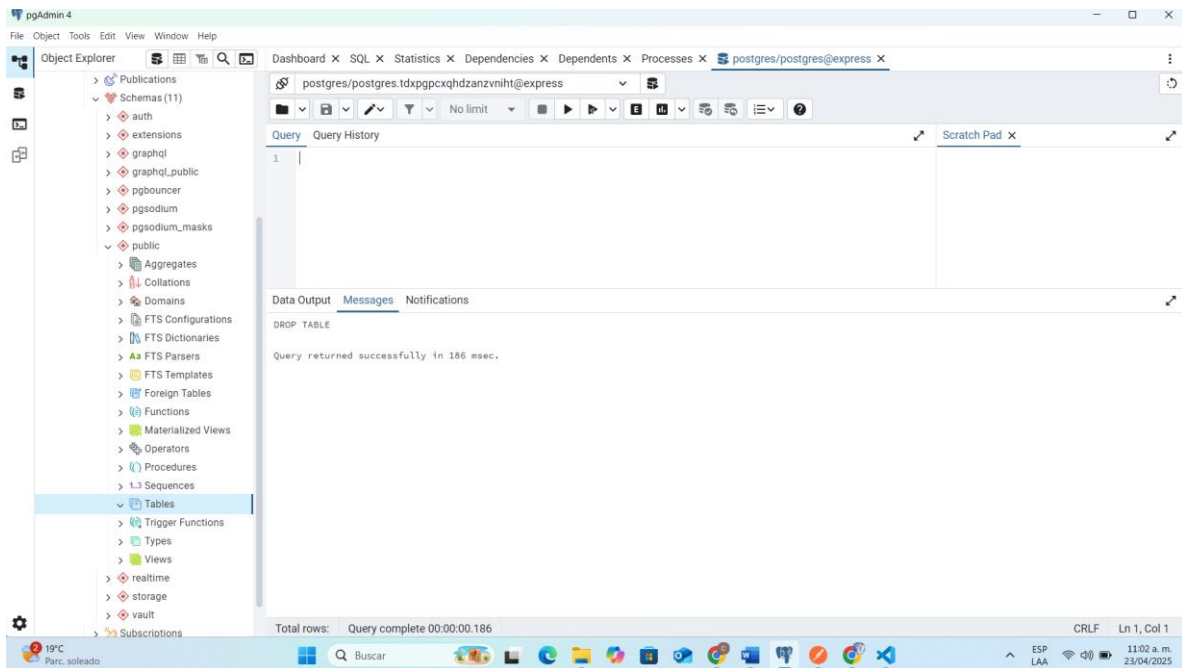
Prof. William Alexander Matallana Parra

23 de abril de 2025

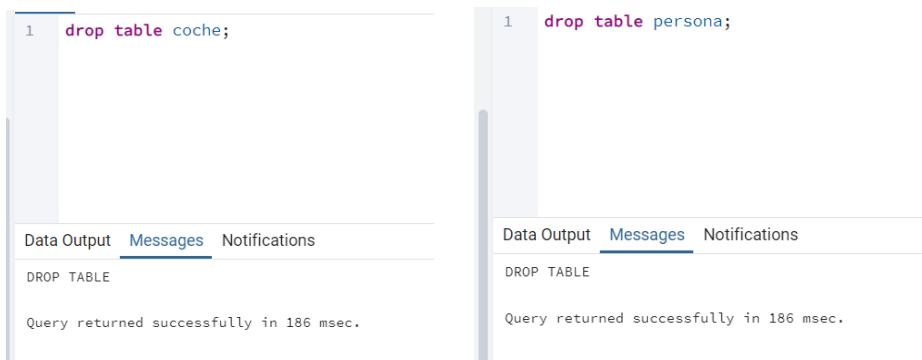
Hago la conexión de pgadmin 4 con supabase. En mi caso, utilizare el mismo proyecto que se trabajo en la clase anterior, debido a que supabase solo permite dos proyectos en el modo gratuito



- Conexión con pgadmin lista:



Como en la anterior clase se había creado una tabla persona, proceso a eliminarla



El siguiente paso, será crear las tablas desde la consola de pgadmin4 para no hacerlo desde Supabase:

Para abrir la consola, se expande las siguientes opciones: schemas y public, en la opción public se le da clic derecho y se toma la opción que dice Query tool

- Create >
- Delete
- Delete (Cascade)
- Refresh...
- Restore...
- Backup...
- CREATE Script
- ERD For Schema
- Maintenance...
- Grant Wizard...
- Search Objects...
- PSQL Tool
- Query Tool
- Properties...

Tabla Restaurante

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

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

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 505 msec.

✓ Query returned successfully in 505 msec. ✕

Reviso que la tabla aparezca en las tablas:



Tabla Empleado

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

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

Verifico que la tabla esta:

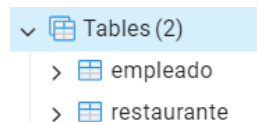


Tabla Producto

```
CREATE TABLE Producto (
```

```
id_prod INT PRIMARY KEY,  
nombre VARCHAR(100),  
precio NUMERIC(10,2)  
);
```

```
1 CREATE TABLE Producto (  
2     id_prod INT PRIMARY KEY,  
3     nombre VARCHAR(100),  
4     precio NUMERIC(10,2)  
5 );  
6
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 236 msec.

Tables (3)

- empleado
- producto
- restaurante

Tabla Pedido

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

```


1  CREATE TABLE Pedido (
2      id_pedido INT PRIMARY KEY,
3      fecha DATE,
4      id_rest INT,
5      total NUMERIC(10,2),
6      FOREIGN KEY (id_rest) REFERENCES Restaurante(id_rest)
7  );
8

```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 200 msec.

✓  Tables (4)





- >  empleado
- >  pedido
- >  producto
- >  restaurante

Tabla DetallePedido

```

CREATE TABLE DetallePedido (
    id_detalle INT PRIMARY KEY,
    id_pedido INT,
    id_prod INT,
    cantidad INT,
    subtotal NUMERIC(10,2),
    FOREIGN KEY (id_pedido) REFERENCES Pedido(id_pedido),
    FOREIGN KEY (id_prod) REFERENCES Producto(id_prod)
);


```






```
1  CREATE TABLE DetallePedido (  
2      id_detalle INT PRIMARY KEY,  
3      id_pedido INT,  
4      id_prod INT,  
5      cantidad INT,  
6      subtotal NUMERIC(10,2),  
7      FOREIGN KEY (id_pedido) REFERENCES Pedido(id_pedido),  
8      FOREIGN KEY (id_prod) REFERENCES Producto(id_prod)  
9  );  
10
```

Data Output Messages Notifications

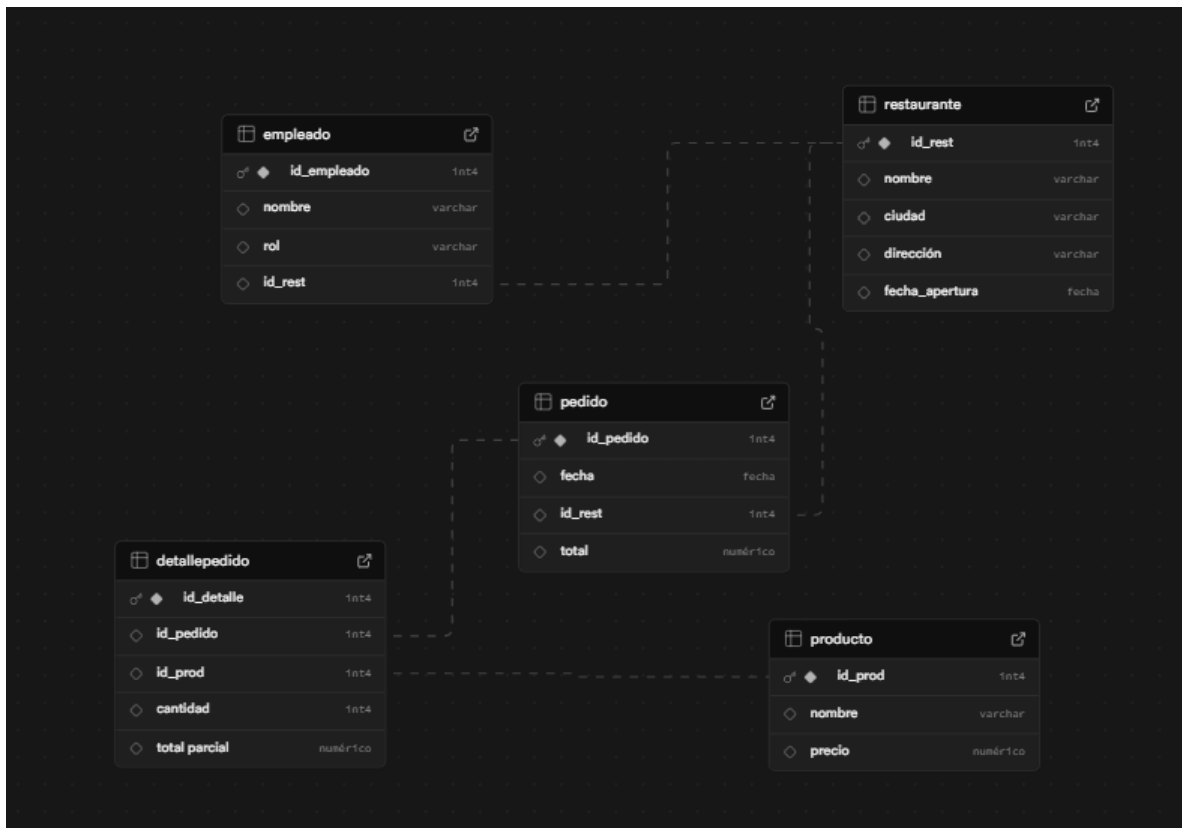
CREATE TABLE

Query returned successfully in 296 msec.

▼  Tables (5)

- >  detallepedido
- >  empleado
- >  pedido
- >  producto
- >  restaurante

Si vamos a Supabase, ingresamos al proyecto, y de las opciones de lado izquierdo, seleccionamos “bases de datos” y después la opción “Visualizador de esquemas”. Allí podremos ver las tablas



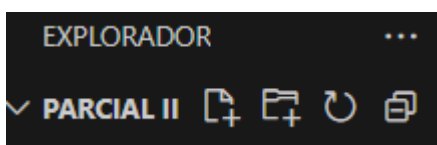
La imagen anterior es el modelo que me da Supabase

La siguiente parte es crear un nuevo proyecto en visual studio code, hacer la conexión, crear las APIS, y realizar el CRUD, siendo esto último ejecutado desde POSTMAN

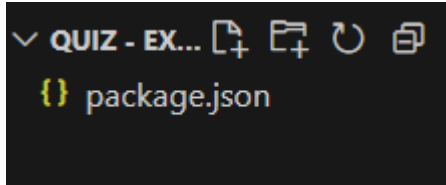
1. Creo una carpeta en el escritorio llamada “parcial II”



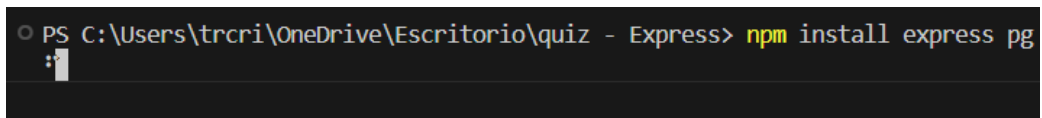
2. Cargo dicha carpeta visualstudio code



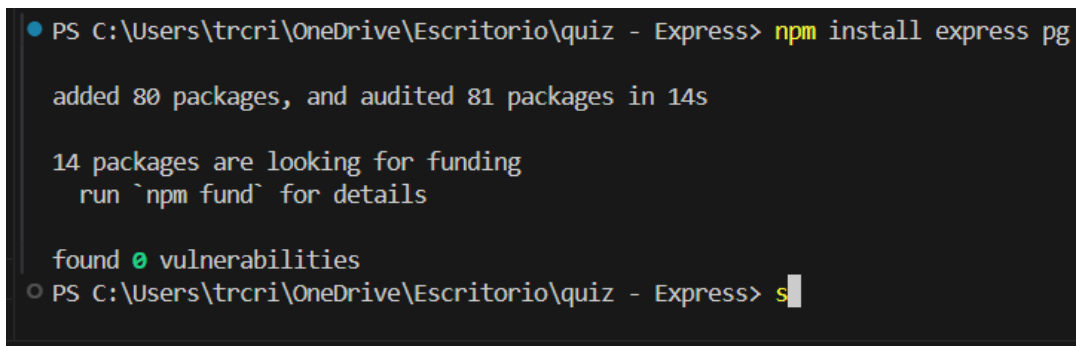
3. Necesito crear el package.json y para esto habro una terminal nueva y ejecuto el siguiente comando: `npm init -y`
- Si todo ha sido exitoso, debería de aparecer un archivo llamado package.json dentro de la carpeta



4. Ahora se tiene que descargar las dependencias de postgres, para que sea posible trabajar sobre el proyecto. Para ello ejecutamos el siguiente comando “`npm install express pg cors`”. Al ejecutar deberá de aparecer que está cargando

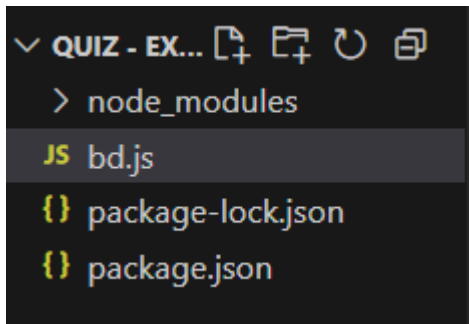


Y después de cargar, si todo ha salido bien, deberá de aparecer el siguiente mensaje:

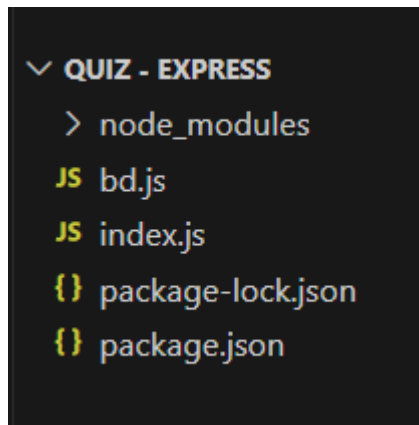


Esto confirma que fue exitoso el proceso

5. Ahora, dentro de la carpeta vamos a agregar dos archivos nuevos con extensión .js
- El primero, se llamará bd.js (donde estará la conexión con supabase)



- El segundo, llevara como nombre index.js (donde se creará las APIS del CRUD)



6. Ahora, tenemos que agregarle líneas de código para que se pueda realizar la conexión y la construcción de las APIS

6.1. Archivo bd.js

Para este archivo se agrega el siguiente código, donde, en la primera parte, se hace la conexión con Supabase (estos datos se obtienen desde el proyecto de SUPABASE, como, en mi caso, estoy trabajando sobre el mismo proyecto del ejercicio anterior, la conexión será la misma, para este nuevo proyecto)

```
JS bd.js X JS index.js
JS bd.js > ...
3 // Datos de conexión con Supabase
4 const client = new Client({
5   host: 'aws-0-sa-east-1.pooler.supabase.com',
6   port: 5432,
7   user: 'postgres.tdxpgpcxqhdzanzvniht',
8   password: 'k7uARMxtf0qyJ9m3',
9   database: 'postgres',
10  ssl: {
11    rejectUnauthorized: false
12  }
13 });
14 // mensajes los cuales me arrojaran en caso de que haya un error, o, por lo contrario, se hizo la conexión exitosamente
15 client.connect((error) => {
16   if (error) {
17     console.log('Error conectando con la base de datos:', error);
18     return;
19   } else {
20     console.log('Conectado con la base de datos de Supabase');
21   }
22 });
```

Este archivo se encarga de establecer la conexión entre mi proyecto en Node.js y la base de datos PostgreSQL que tengo alojada en Supabase. Para lograrlo, utilizo el paquete pg para conectarme a bases de datos PostgreSQL desde Node.js.

```
const client = new Client({
  host: 'aws-0-sa-east-1.pooler.supabase.com',
  port: 5432,
  user: 'postgres.tdxpgpcxqhdzanzvniht',
  password: 'k7uARMxtf0qyJ9m3',
  database: 'postgres',
  ssl: {
    rejectUnauthorized: false
  }
});
```

De la imagen anterior: lo que estoy haciendo es crear un nuevo cliente “client” configurándolo con los datos de conexión que me da Supabase:

- ✓ host: es la dirección del servidor donde está la base de datos.
- ✓ port: es el puerto por donde se conecta (el 5432 es el estándar de PostgreSQL).
- ✓ user: es el usuario de la base de datos.
- ✓ password: es la clave de ese usuario.
- ✓ database: en este caso, la base de datos por defecto de Supabase se llama postgres.
- ✓ ssl: se usa para hacer la conexión segura. Puse rejectUnauthorized: false para evitar problemas de certificados.

```
client.connect((error) => {
  if (error) {
    console.log('Error conectando con la base de datos:', error);
    return;
  } else {
    console.log('Conectado con la base de datos de Supabase');
  }
});
(alias) const export=: Client
import export=
module.exports = client;
```

De la imagen anterior: intento conectar el cliente con la base de datos. Si ocurre algún error, se muestra en la consola con un mensaje de, si la conexión es exitosa, muestra un mensaje confirmando que se estableció correctamente.

Por otro lado, exporto el objeto client para poder usar esta conexión en otros archivos del proyecto, como en los controladores o rutas.

```
EXPLORADOR  ...  # bd.js  # index.js  X  ...  [icon] ...

QUIZ - EXPRESS
  > documenta.pdf
  > node_modules
  # bd.js
  # index.js
  {} package-lock.json
  {} package.json

# index.js
1  # index.js > app.post('/api/persona') callback
2  const express = require('express');
3  const cors = require('cors');
4  const client = require('./db');
5  const app = express();
6  const PORT = 3000;
7
8  app.use(cors());
9  app.use(express.json());
10 app.use(express.urlencoded({ extended: true }));
11
12 // RUTA BÁSICA
13 app.get('/api/prueba', (req, res) => {
14   res.send('API funcionando correctamente');
15 });
16
17 // de aquí en adelante son las APIs para el CRUO DE PERSONA
18
19 // crear persona
20 app.post('/api/persona', async (req, res) => {
21   const { nombre, apellido1, apellido2, dni } = req.body;
22   try {
23     await client.query(
24       'INSERT INTO persona (nombre, apellido1, apellido2, dni) VALUES ($1, $2, $3, $4)',
25       [nombre, apellido1, apellido2, dni]
26     );
27     res.status(201).json({ success: true, message: 'Persona registrada' });
28   } catch (error) {
29     res.status(500).json({ success: false, error: error.message });
30   }
31 });
32
33 // Obtener personas
```