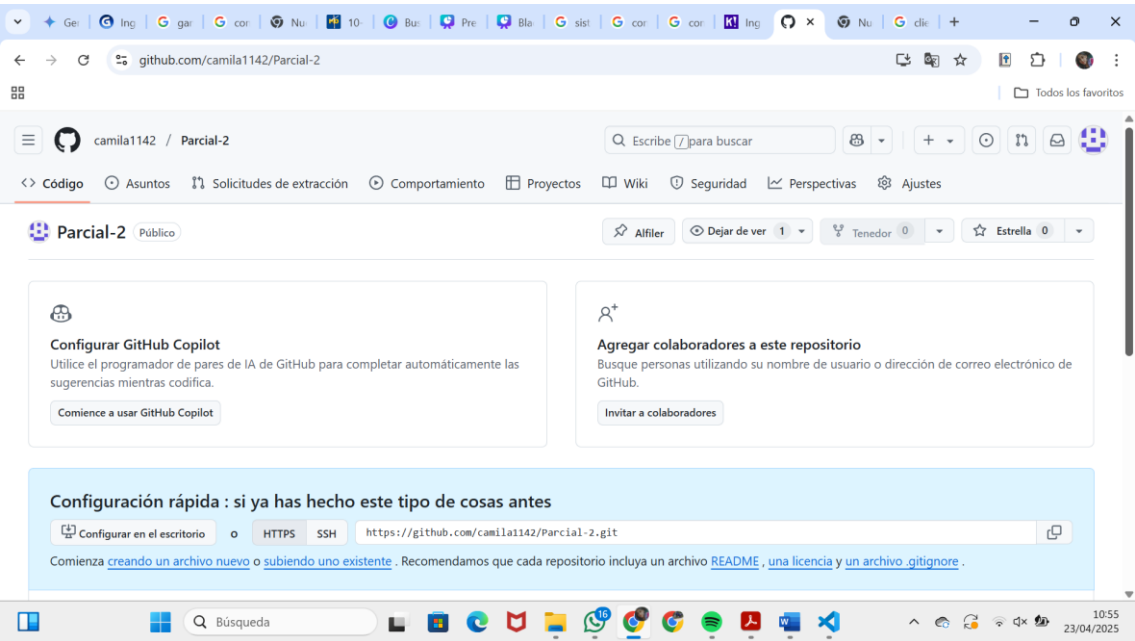


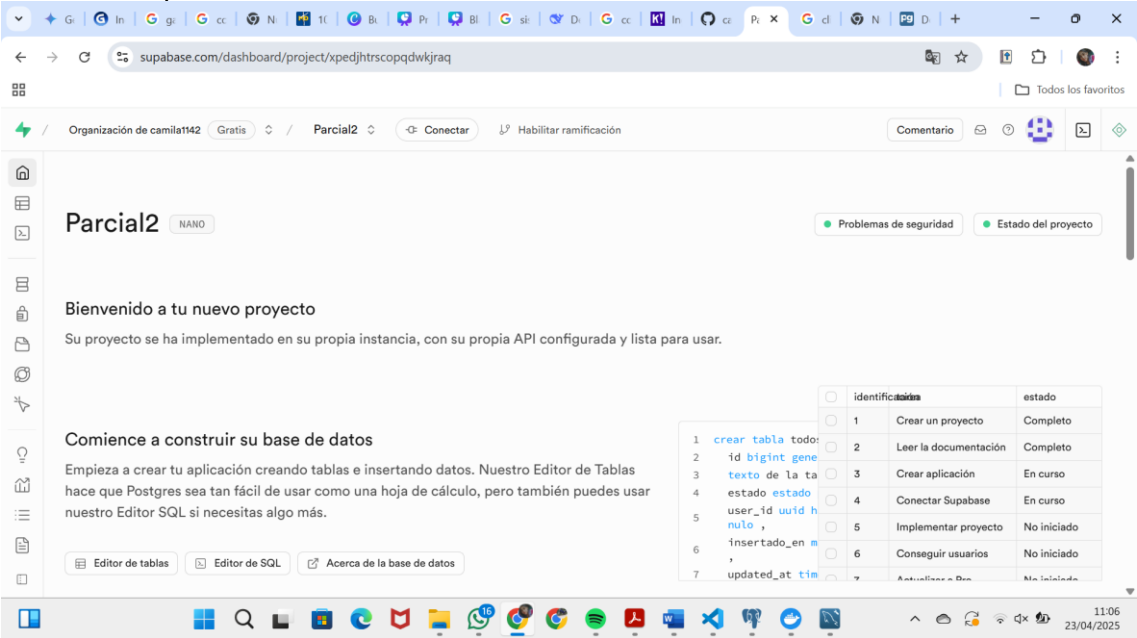
Parcial corte 2

Julieth Camila Castro Parada

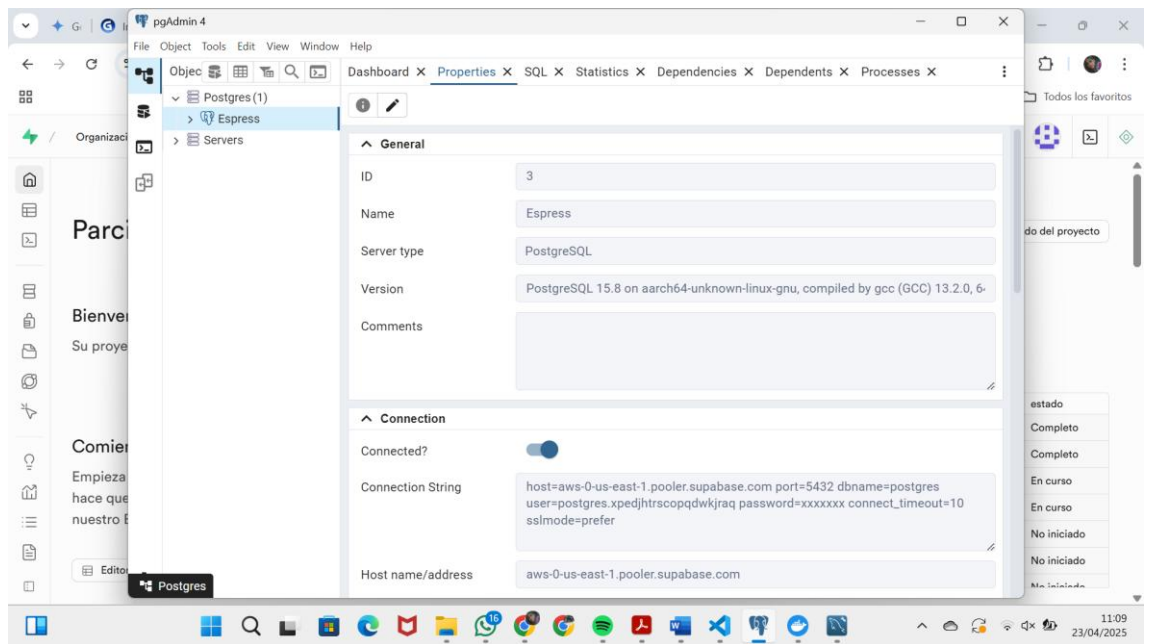
1. Crear repositorio



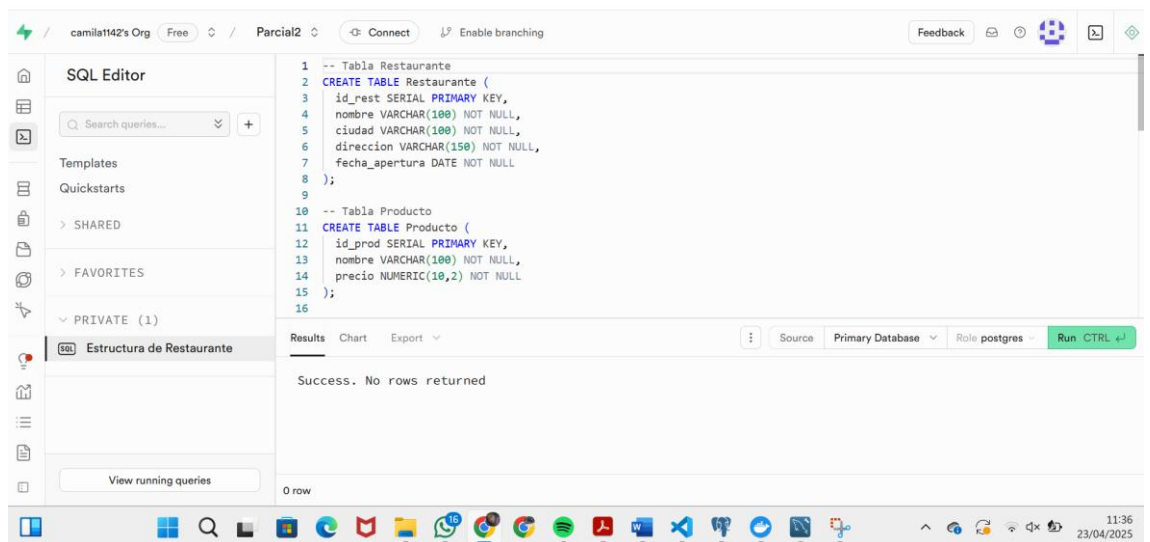
2. Crear en supabase

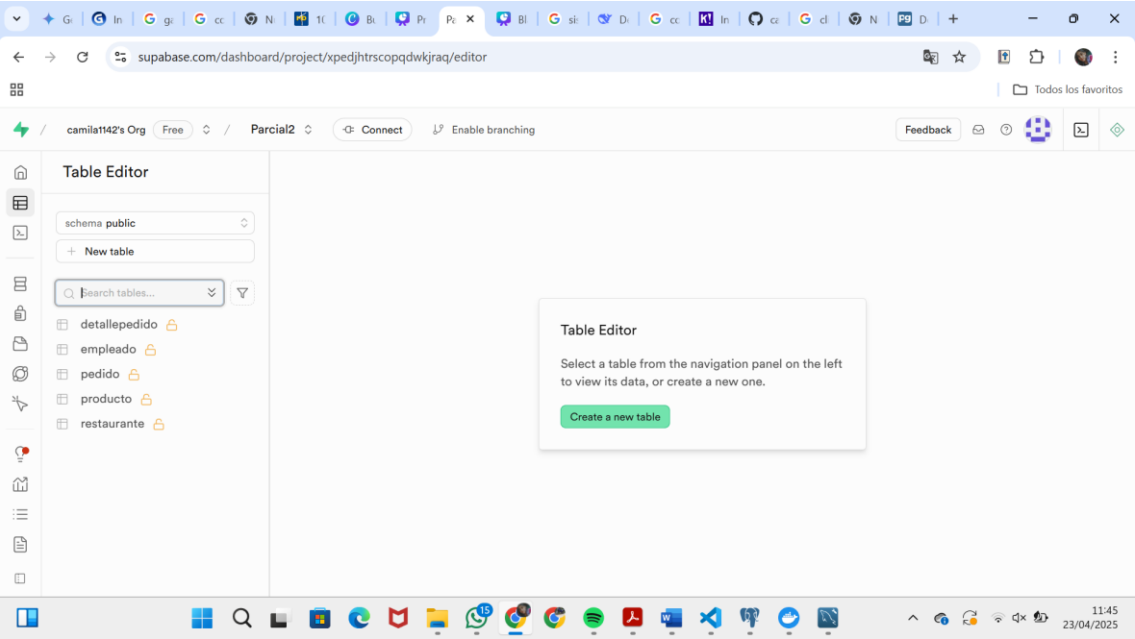


3. Hacer conexion con supabase

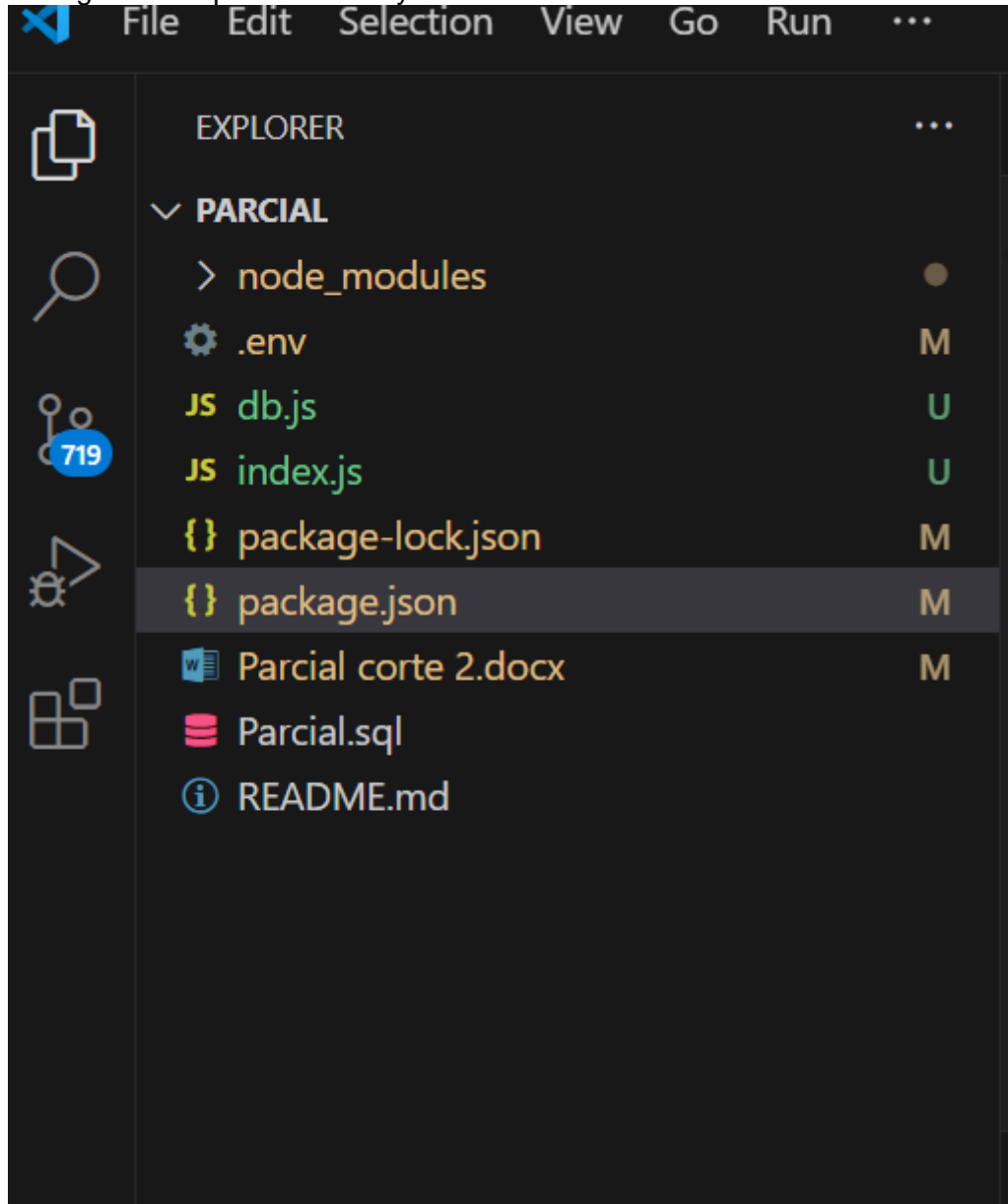


Crear tablas en supabase

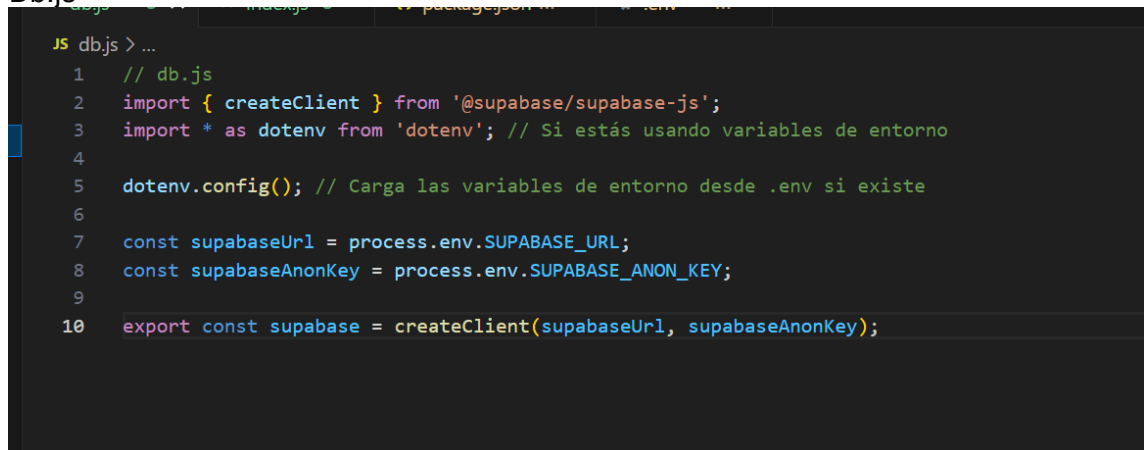




Configurar las api en el visual y el crud



Db.js



Api get restaurante

```
// Rutas para la tabla Restaurante
app.get('/api/restaurante', async (req, res) => {
  const { data, error } = await supabase
    .from('Restaurante')
    .select('*');

  if (error) {
    return res.status(500).json({ error: error.message });
  }

  res.json(data);
});
```

Post

```
app.post('/api/restaurante', async (req, res) => {
  const { nombre, ciudad, direccion, fecha_apertura } = req.body;
  const { data, error } = await supabase
    .from('Restaurante')
    .insert([ { nombre, ciudad, direccion, fecha_apertura } ])
    .select()
    .single();

  if (error) {
    return res.status(500).json({ error: error.message });
  }

  res.status(201).json(data);
});
```

Delete

```
app.delete('/api/restaurante/:id', async (req, res) => {
  const { id } = req.params;
  const { error } = await supabase
    .from('Restaurante')
    .delete()
    .eq('id_rest', id);

  if (error) {
    return res.status(500).json({ error: error.message });
  }

  res.status(204).send(); // 204 No Content para indicar éxito sin cuerpo
});
```

Get empleado

```
// Rutas para la tabla Empleado
app.get('/api/empleado', async (req, res) => {
  const { data, error } = await supabase
    .from('Empleado')
    .select('*');

  if (error) {
    return res.status(500).json({ error: error.message });
  }
  res.json(data);
});

app.get('/api/empleado/:id', async (req, res) => {
  const { id } = req.params;
  const { data, error } = await supabase
    .from('Empleado')
    .select('*')
    .eq('id_empleado', id)
    .single();

  if (error) {
    return res.status(404).json({ error: 'Empleado no encontrado' });
  }
  res.json(data);
});
```

Post y delete empleado

```
app.post('/api/empleado', async (req, res) => {
  const { nombre, rol, id_rest } = req.body;
  const { data, error } = await supabase
    .from('Empleado')
    .insert([ { nombre, rol, id_rest } ])
    .select()
    .single();

  if (error) {
    return res.status(500).json({ error: error.message });
  }
  res.status(201).json(data);
});

app.delete('/api/empleado/:id', async (req, res) => {
  const { id } = req.params;
  const { error } = await supabase
    .from('Empleado')
    .delete()
    .eq('id_empleado', id);

  if (error) {
    return res.status(500).json({ error: error.message });
  }
  res.status(204).send();
});
```

Get Producto

```
// Rutas para la tabla Producto
app.get('/api/producto', async (req, res) => {
  const { data, error } = await supabase
    .from('Producto')
    .select('*');

  if (error) {
    return res.status(500).json({ error: error.message });
  }
  res.json(data);
});

app.get('/api/producto/:id', async (req, res) => {
  const { id } = req.params;
  const { data, error } = await supabase
    .from('Producto')
    .select('*')
    .eq('id_prod', id)
    .single();

  if (error) {
    return res.status(404).json({ error: 'Producto no encontrado' });
  }
  res.json(data);
});
```

Post producto

```
app.post('/api/producto', async (req, res) => {
  const { nombre, precio } = req.body;
  const { data, error } = await supabase
    .from('Producto')
    .insert([ { nombre, precio } ])
    .select()
    .single();

  if (error) {
    return res.status(500).json({ error: error.message });
  }
  res.status(201).json(data);
});
```

Delete Producto

```
app.delete('/api/producto/:id', async (req, res) => {
  const { id } = req.params;
  const { error } = await supabase
    .from('Producto')
    .delete()
    .eq('id_prod', id);

  if (error) {
    return res.status(500).json({ error: error.message });
  }
  res.status(204).send();
});
```

Get pedido

```
// Rutas para la tabla Pedido
app.get('/api/pedido', async (req, res) => {
  const { data, error } = await supabase
    .from('Pedido')
    .select('*');

  if (error) {
    return res.status(500).json({ error: error.message });
  }
  res.json(data);
});

app.get('/api/pedido/:id', async (req, res) => {
  const { id } = req.params;
  const { data, error } = await supabase
    .from('Pedido')
    .select('*')
    .eq('id_pedido', id)
    .single();

  if (error) {
    return res.status(404).json({ error: 'Pedido no encontrado' });
  }
  res.json(data);
});

app.post('/api/pedido', async (req, res) => {
  const { fecha, id_rest, total } = req.body;
  const { data, error } = await supabase
```

Post pedido

```
app.post('/api/pedido', async (req, res) => {
  const { fecha, id_rest, total } = req.body;
  const { data, error } = await supabase
    .from('Pedido')
    .insert([ { fecha, id_rest, total } ])
    .select()
    .single();

  if (error) {
    return res.status(500).json({ error: error.message });
  }
  res.status(201).json(data);
});
```

Delete pedido

```
app.delete('/api/pedido/:id', async (req, res) => {
  const { id } = req.params;
  const { error } = await supabase
    .from('Pedido')
    .delete()
    .eq('id_pedido', id);

  if (error) {
    return res.status(500).json({ error: error.message });
  }
  res.status(204).send();
});
```

Get de detalle de pedido


```

    app.get('/api/detallepedido', async (req, res) => {
      const { data, error } = await supabase
        .from('DetallePedido')
        .select('*');

      if (error) {
        return res.status(500).json({ error: error.message });
      }
      res.json(data);
    });

    app.get('/api/detallepedido/:id', async (req, res) => {
      const { id } = req.params;
      const { data, error } = await supabase
        .from('DetallePedido')
        .select('*')
        .eq('id_detalle', id)
        .single();

      if (error) {
        return res.status(404).json({ error: 'Detalle de Pedido no encontrado' });
      }
      res.json(data);
    });
  });

```

Post detalle de pedido

```

app.post('/api/detallepedido', async (req, res) => {
  const { id_pedido, id_prod, cantidad, subtotal } = req.body;
  const { data, error } = await supabase
    .from('DetallePedido')
    .insert([ { id_pedido, id_prod, cantidad, subtotal } ])
    .select()
    .single();

  if (error) {
    return res.status(500).json({ error: error.message });
  }
  res.status(201).json(data);
});

```

Delete detalle de pedido

```

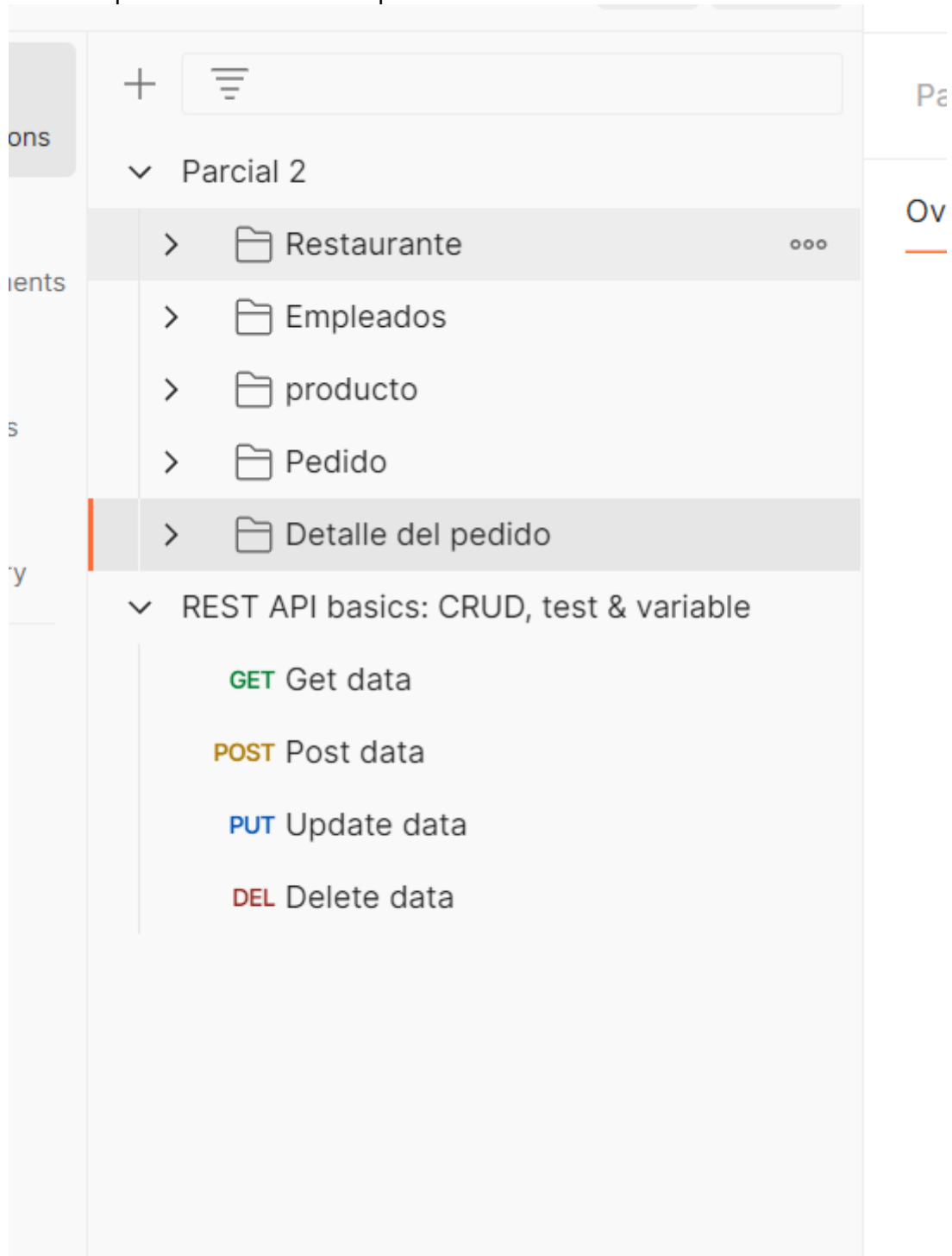
app.delete('/api/detallepedido/:id', async (req, res) => {
  const { id } = req.params;
  const { error } = await supabase
    .from('DetallePedido')
    .delete()
    .eq('id_detalle', id);

  if (error) {
    return res.status(500).json({ error: error.message });
  }
  res.status(204).send();
});

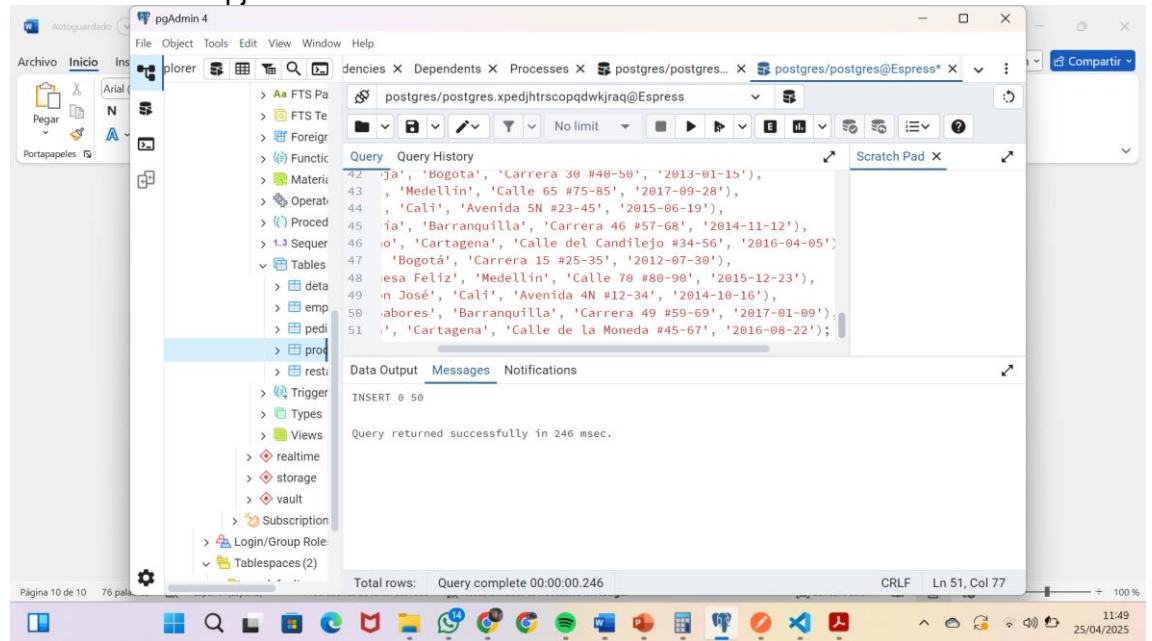
app.listen(port, () => {
  console.log(`Servidor escuchando en el puerto ${port}`);
});

```

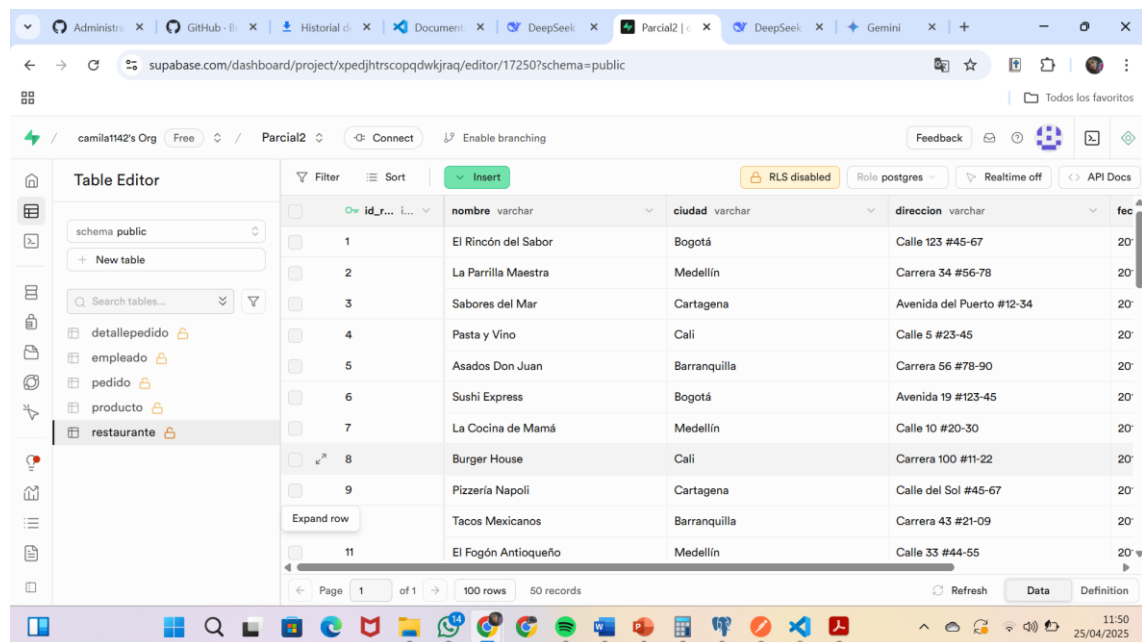
Crear carpetas de cada tabla en potsman



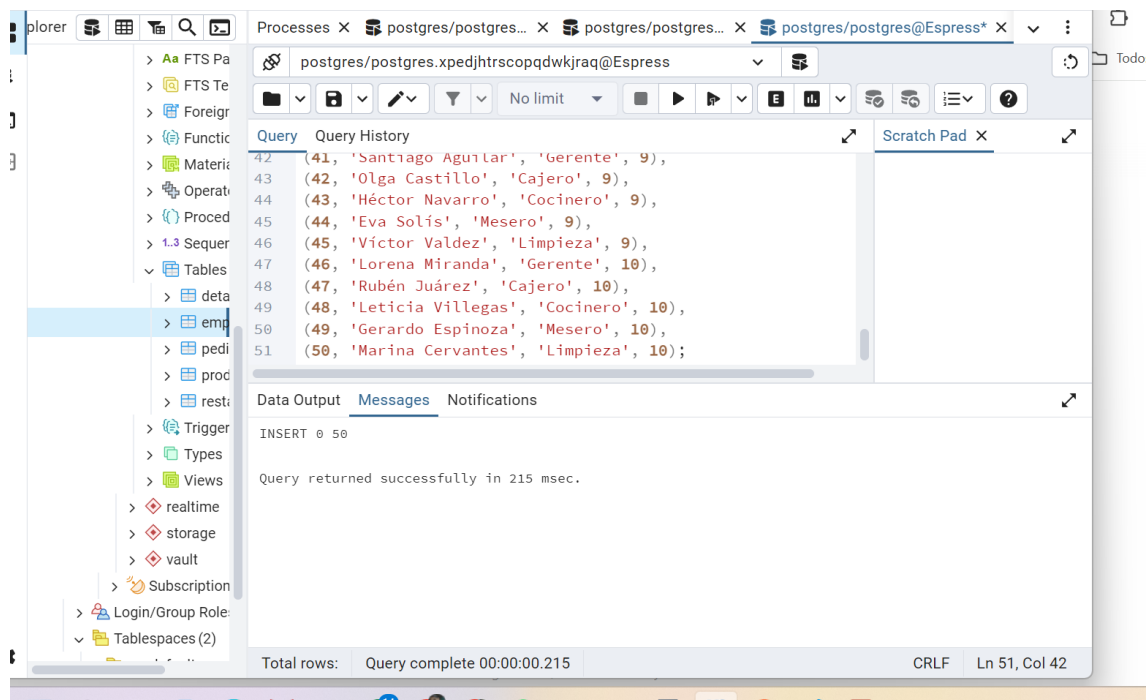
Inertar datos en pj admin



Verifica que aperezcan en supabase



Inserta datos en la tabla empleados



Verifica en subabase los datos insertados

camila1142's Org Free / Parcial2 Connect Enable branching Feedback

Table Editor schema public New table Search tables...

detallepedido empleado pedido producto restaurante

Filter Sort Insert RLS disabled Role postgres Realtime off API Docs

	id_em...	nombre	rol	id_r...
<input type="checkbox"/>	1	Juan Pérez	Gerente	1
<input type="checkbox"/>	2	María Rodríguez	Cajero	1
<input type="checkbox"/>	3	Carlos Gómez	Cocinero	1
<input type="checkbox"/>	4	Ana López	Mesero	1
<input type="checkbox"/>	5	Pedro Martínez	Limpieza	1
<input type="checkbox"/>	6	Laura García	Gerente	2
<input type="checkbox"/>	7	Andrés Sánchez	Cajero	2
<input type="checkbox"/>	8	Sofía Ramírez	Cocinero	2
<input type="checkbox"/>	9	Jorge Díaz	Mesero	2
<input type="checkbox"/>	10	Diana Torres	Limpieza	2
<input type="checkbox"/>	11	Ricardo Vargas	Gerente	3

Page 1 of 1 100 rows 50 records Refresh Data Definition

Tabla producto

Query:

```

INSERT INTO producto (id_producto, nombre, precio)
VALUES
(40, 'Sashimi', 20000.00),
(41, 'Tempura', 22000.00),
(42, 'Agua Mineral', 3000.00),
(43, 'Gaseosa', 4000.00),
(44, 'Jugo Natural', 5000.00),
(45, 'Limonada', 4500.00),
(46, 'Cerveza Nacional', 6000.00),
(47, 'Cerveza Importada', 8000.00),
(48, 'Vino Tinto', 12000.00),
(49, 'Vino Blanco', 12000.00),
(50, 'Postre del Día', 7000.00);
    
```

Data Output:

```

INSERT 0 50
Query returned successfully in 249 msec.
    
```

✓ Query returned successfully in 249 msec. ✕

Total rows: Query complete 00:00:00.249 CRLF Ln 51, Col 33

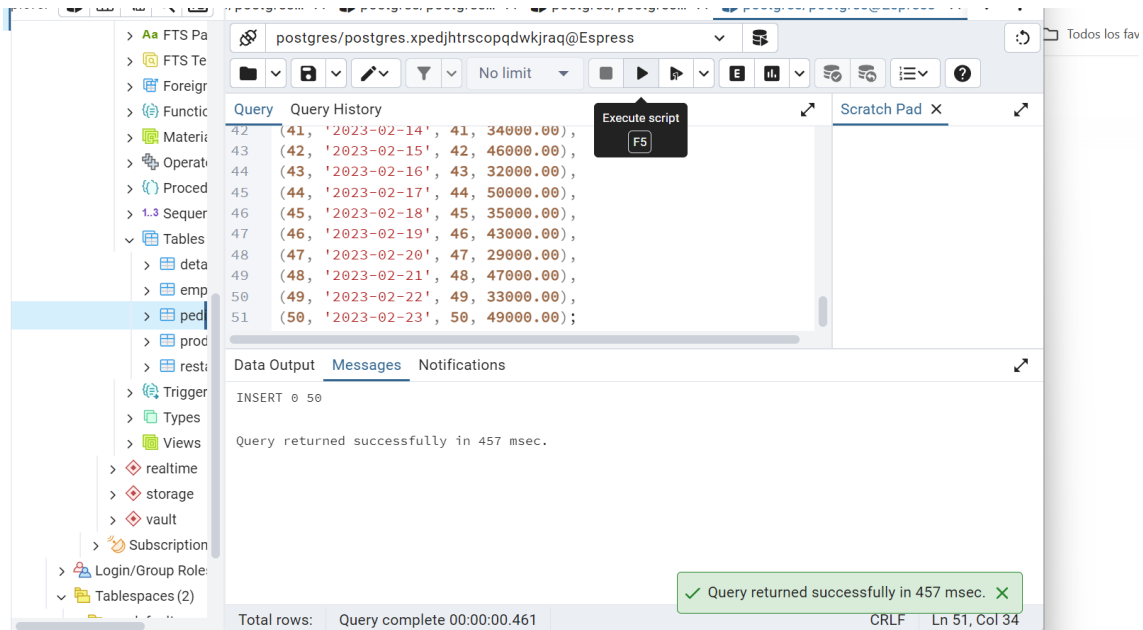
Verifica los datos en supabase

Table Editor: producto

id_producto	nombre	precio
1	Hamburguesa Clásica	12000.00
2	Hamburguesa con Queso	15000.00
3	Hamburguesa Doble	18000.00
4	Papas Fritas	5000.00
Expand row	Papas Fritas con Queso	8000.00
6	Ensalada César	10000.00
7	Ensalada Griega	12000.00
8	Sándwich de Pollo	11000.00
9	Sándwich de Carne	13000.00
10	Sándwich Vegetariano	9000.00
11	Pizza Margarita	20000.00

Page 1 of 1 100 rows 50 records Refresh Data Definition

Inserta en la tabla pedidos



Verifica en supabase

The screenshot shows the Supabase Table Editor interface. The table 'pedido' is selected, and its schema is displayed. The table has columns: id_pedido, fecha, id_producto, and total. The data is shown in a table with 11 rows. The interface includes a sidebar with a list of tables, a search bar, and a table editor with columns and rows. The status bar at the bottom shows "Data" and "Refresh".

id_pedido	fecha	id_producto	total
1	2023-01-05	1	35000.00
2	2023-01-06	2	42000.00
3	2023-01-07	3	38000.00
4	2023-01-08	4	29000.00
5	2023-01-09	5	51000.00
6	2023-01-10	6	27000.00
7	2023-01-11	7	33000.00
8	2023-01-12	8	45000.00
9	2023-01-13	9	36000.00
10	2023-01-14	10	41000.00
11	2023-01-15	11	28000.00

Inserta en la tabla detalle del pedido

postgres/postgres.xpedjhtrscopqdwkjqraq@Espress

Query Query History

```
42 (41, 16, 42, 2, 6000.00),
43 (42, 16, 50, 1, 7000.00),
44 (43, 17, 25, 1, 28000.00),
45 (44, 17, 46, 1, 6000.00),
46 (45, 17, 47, 1, 8000.00),
47 (46, 18, 17, 1, 20000.00),
48 (47, 18, 42, 2, 6000.00),
49 (48, 19, 28, 1, 35000.00),
50 (49, 19, 48, 1, 12000.00),
51 (50, 20, 39, 1, 25000.00);
```

Data Output Messages Notifications

INSERT 0 50

Query returned successfully in 231 msec.

Verifica en supabase

Table Editor

schema public

+ New table

Search tables...

detallepedido

empleado

pedido

producto

restaurante

Filter Sort Insert

RLS disabled Role postgres Realtime off

	id_d...	id_p...	id...	cantidad	subto...	nume...	+
	1	1	1	2	24000.00		
	2	1	4	1	5000.00		
	3	1	42	2	6000.00		
	4	2	11	1	20000.00		
	5	2	42	1	3000.00		
	6	2	43	2	8000.00		
	7	2	50	1	7000.00		
	8	3	25	1	28000.00		
	9	3	46	1	6000.00		
	10	3	47	1	8000.00		
	11	4	15	1	18000.00		

Page 1 of 1 100 rows 50 records Refresh Data Defi

Esquema en supabase

