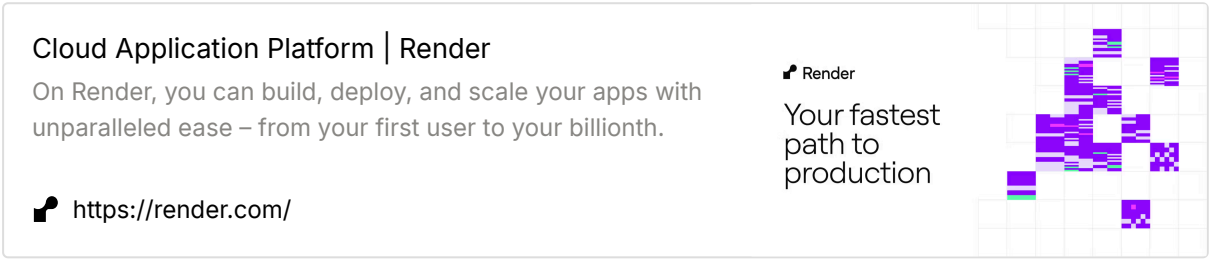
PP 1 - DER + BD + API

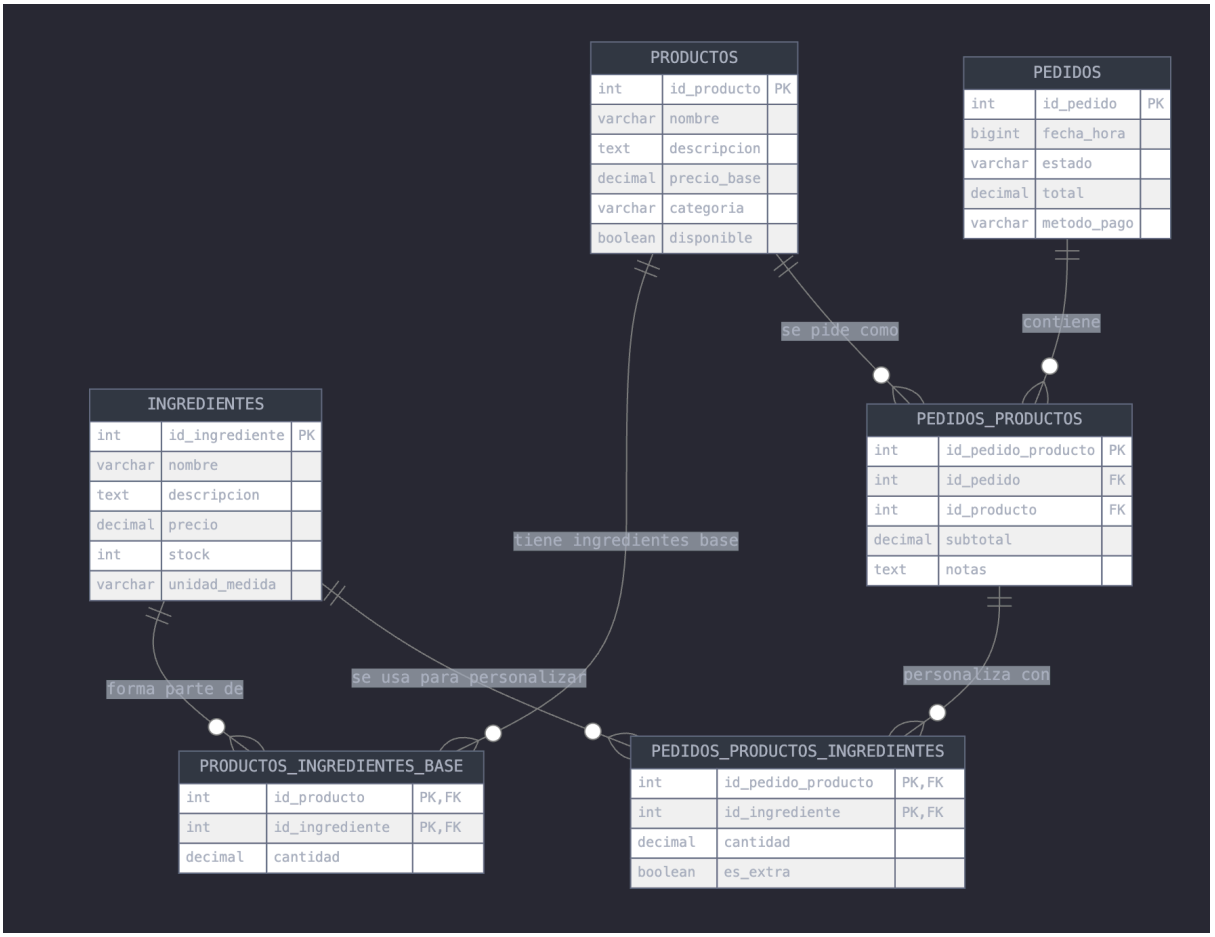
BD alojada en Render



Para ejecutar ejemplo en Node JS hace falta el .env con lo siguiente...

```
# URL de conexión a PostgreSQL en Render.com
DATABASE_URL= .....
PORT=3000
```

Estructura y Relaciones de la Base de Datos



Explicación del Modelo de Datos:

1. **Productos y sus ingredientes base**:
   - Un producto (como una hamburguesa) tiene varios ingredientes base definidos en `productos_ingredientes_base`
   - Esta tabla vincula productos con sus ingredientes predeterminados y especifica la cantidad de cada ingrediente
2. **Pedidos y sus productos**:
   - Un pedido contiene uno o más productos a través de la tabla `pedidos_productos`
   - Cada entrada en `pedidos_productos` representa un ítem específico del pedido
3. **Personalización de productos**:

- Los clientes pueden personalizar productos agregando o quitando ingredientes
- Esto se maneja en `pedidos_productos_ingredientes`, vinculada a `pedidos_productos`
- El campo `es_extra` determina si es un ingrediente adicional (true) o parte de la receta base (false)

Flujo Típico:

1. Cliente selecciona un producto (ej. hamburguesa clásica)
2. Sistema muestra ingredientes base de la hamburguesa desde `productos_ingredientes_base`
3. Cliente puede agregar ingredientes extra o eliminar algunos base
4. Se registra el pedido en `pedidos` y el producto en `pedidos_productos`
5. Las personalizaciones se guardan en `pedidos_productos_ingredientes`

Script de creación de BD

```
-- Creación de tablas para un autoservicio de hamburguesas
-- Fechas en formato UTC (segundos desde 1970-01-01 00:00:00)

-- Tabla productos
CREATE TABLE productos (
    id_producto SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    descripcion TEXT,
    precio_base DECIMAL(10, 2) NOT NULL,
    categoria VARCHAR(50) NOT NULL,
    disponible BOOLEAN DEFAULT TRUE
);

-- Tabla ingredientes
CREATE TABLE ingredientes (
    id_ingrediente SERIAL PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    descripcion TEXT,
```

```
    precio DECIMAL(10, 2) NOT NULL,
    stock INTEGER NOT NULL,
    unidad_medida VARCHAR(20) NOT NULL
);

-- Tabla productos_ingredientes_base
CREATE TABLE productos_ingredientes_base (
    id_producto INTEGER REFERENCES productos(id_producto),
    id_ingrediente INTEGER REFERENCES ingredientes(id_ingrediente),
    cantidad DECIMAL(10, 2) NOT NULL,
    PRIMARY KEY (id_producto, id_ingrediente)
);

-- Tabla pedidos
CREATE TABLE pedidos (
    id_pedido SERIAL PRIMARY KEY,
    fecha_hora BIGINT NOT NULL, -- Formato UTC (segundos desde epoch)
    estado VARCHAR(50) NOT NULL DEFAULT 'pendiente',
    total DECIMAL(10, 2) NOT NULL,
    metodo_pago VARCHAR(50) NOT NULL
);

-- Tabla pedidos_productos (MODIFICADA - sin columna cantidad)
CREATE TABLE pedidos_productos (
    id_pedido_producto SERIAL PRIMARY KEY,
    id_pedido INTEGER REFERENCES pedidos(id_pedido),
    id_producto INTEGER REFERENCES productos(id_producto),
    -- cantidad INTEGER NOT NULL, -- REMOVIDO: cada fila representa una
unidad
    subtotal DECIMAL(10, 2) NOT NULL, -- precio final de este producto esp
ecífico
    notas TEXT -- personalizaciones específicas para este producto individu
al
);

-- Tabla pedidos_productos_ingredientes (sin cambios)
CREATE TABLE pedidos_productos_ingredientes (
    id_pedido_producto INTEGER REFERENCES pedidos_productos(id_pedid
```

```
o_producto),
    id_ingrediente INTEGER REFERENCES ingredientes(id_ingrediente),
    cantidad DECIMAL(10, 2) NOT NULL,
    es_extra BOOLEAN NOT NULL,
    PRIMARY KEY (id_pedido_producto, id_ingrediente)
);
```

Script Data Ejemplo

```
-- Datos de ejemplo actualizados para el nuevo esquema

-- Inserción de productos (sin cambios)
INSERT INTO productos (nombre, descripcion, precio_base, categoria, disp
onible) VALUES
('McRaulo Cheese', 'Hamburguesa clásica con queso cheddar', 8.99, 'ham
burguesa', TRUE),
('McRaulo Veggie', 'Hamburguesa vegetariana con proteína de soja', 9.99,
'hamburguesa', TRUE),
('McRaulo Pollo', 'Hamburguesa de pollo crujiente', 9.49, 'hamburguesa', T
RUE),
('Papas fritas', 'Papas fritas crujientes', 3.99, 'acompañamiento', TRUE),
('Papas con cheddar', 'Papas fritas con queso cheddar derretido', 5.99, 'ac
ompañamiento', TRUE),
('Coca-Cola', 'Refresco de cola 500ml', 2.50, 'bebida', TRUE),
('Sprite', 'Refresco de lima-limón 500ml', 2.50, 'bebida', TRUE),
('Fanta', 'Refresco de naranja 500ml', 2.50, 'bebida', TRUE),
('Limonada', 'Limonada natural 500ml', 3.50, 'bebida', TRUE),
('Agua mineral', 'Agua mineral 500ml', 1.50, 'bebida', TRUE);

-- Inserción de ingredientes (sin cambios)
INSERT INTO ingredientes (nombre, descripcion, precio, stock, unidad_me
dida) VALUES
('Pan de hamburguesa', 'Pan brioche para hamburguesa', 0.50, 500, 'unida
d'),
('Carne de res', 'Medallón de carne de res 150g', 2.50, 300, 'unidad'),
('Pollo crispy', 'Medallón de pollo rebozado 120g', 2.00, 200, 'unidad'),
('Medallón veggie', 'Medallón de proteína vegetal 120g', 2.50, 150, 'unida
d'),
```

```
('Queso cheddar', 'Feta de queso cheddar', 0.75, 400, 'unidad'),
('Lechuga', 'Lechuga fresca', 0.30, 5000, 'g'),
('Tomate', 'Rodaja de tomate', 0.40, 10000, 'g'),
('Cebolla', 'Cebolla en rodajas', 0.30, 8000, 'g'),
('Pepinillos', 'Pepinillos en rodajas', 0.50, 3000, 'g'),
('Bacon', 'Tiras de bacon crujiente', 1.50, 5000, 'g'),
('Salsa especial', 'Salsa especial de la casa', 0.50, 10000, 'ml'),
('Ketchup', 'Salsa de tomate', 0.20, 15000, 'ml'),
('Mostaza', 'Mostaza amarilla', 0.20, 12000, 'ml'),
('Mayonesa', 'Mayonesa casera', 0.30, 10000, 'ml'),
('Guacamole', 'Guacamole fresco', 1.20, 5000, 'g');

-- Asociación de ingredientes base a productos (sin cambios)
-- McRaulo Cheese
INSERT INTO productos_ingredientes_base (id_producto, id_ingrediente, ca
ntidad) VALUES
(1, 1, 1), -- Pan
(1, 2, 1), -- Carne
(1, 5, 2), -- Queso cheddar (2 fetas)
(1, 6, 20), -- Lechuga (20g)
(1, 7, 30), -- Tomate (30g)
(1, 8, 15), -- Cebolla (15g)
(1, 11, 15), -- Salsa especial (15ml)
(1, 12, 10); -- Ketchup (10ml)

-- McRaulo Veggie
INSERT INTO productos_ingredientes_base (id_producto, id_ingrediente, ca
ntidad) VALUES
(2, 1, 1), -- Pan
(2, 4, 1), -- Medallón veggie
(2, 5, 1), -- Queso cheddar
(2, 6, 25), -- Lechuga (25g)
(2, 7, 30), -- Tomate (30g)
(2, 8, 15), -- Cebolla (15g)
(2, 9, 15), -- Pepinillos (15g)
(2, 11, 15), -- Salsa especial (15ml)
(2, 14, 10); -- Mayonesa (10ml)
```

```
-- McRaulo Pollo
INSERT INTO productos_ingredientes_base (id_producto, id_ingrediente, ca
ntidad) VALUES
(3, 1, 1), -- Pan
(3, 3, 1), -- Pollo crispy
(3, 5, 1), -- Queso cheddar
(3, 6, 20), -- Lechuga (20g)
(3, 7, 30), -- Tomate (30g)
(3, 14, 15); -- Mayonesa (15ml)

-- Ejemplo de pedidos
INSERT INTO pedidos (fecha_hora, estado, total, metodo_pago) VALUES
(1716056400, 'pendiente', 24.97, 'mercadopago'), -- 18 mayo 2025 15:00:0
0 UTC
(1716142800, 'entregado', 32.48, 'efectivo'); -- 19 mayo 2025 15:00:00 UTC

-- Productos del primer pedido (MODIFICADO - cada producto es una fila i
ndividual)
-- Pedido 1: 1 McRaulo Cheese + 1 Papas fritas + 2 Coca-Cola + 1 Agua min
eral
INSERT INTO pedidos_productos (id_pedido, id_producto, subtotal, notas)
VALUES
-- McRaulo Cheese personalizada (sin cebolla, con bacon extra y guacamo
le extra)
(1, 1, 11.69, 'Sin cebolla, con bacon extra y guacamole extra'),
-- Papas fritas normales
(1, 4, 3.99, NULL),
-- Primera Coca-Cola
(1, 6, 2.50, NULL),
-- Segunda Coca-Cola
(1, 6, 2.50, NULL),
-- Agua mineral
(1, 10, 1.50, NULL);

-- Ingredientes personalizados del primer pedido
-- Para la hamburguesa McRaulo Cheese (id_pedido_producto = 1)
INSERT INTO pedidos_productos_ingredientes (id_pedido_producto, id_ingr
ediente, cantidad, es_extra) VALUES
```

```
(1, 8, 0, FALSE), -- Sin cebolla (cantidad 0, no es extra)
(1, 10, 2, TRUE), -- Extra bacon (2 unidades, es extra)
(1, 15, 1, TRUE); -- Extra guacamole (1 unidad, es extra)

-- Productos del segundo pedido (MODIFICADO - cada producto es una fil
a individual)
-- Pedido 2: 1 McRaulo Veggie + 1 McRaulo Pollo + 1 Papas con cheddar + 1
Limonada
INSERT INTO pedidos_productos (id_pedido, id_producto, subtotal, notas)
VALUES
-- McRaulo Veggie con queso extra
(2, 2, 10.74, 'Con queso extra'),
-- McRaulo Pollo sin tomate, bien cocido
(2, 3, 9.49, 'Sin tomate, bien cocido'),
-- Papas con cheddar normales
(2, 5, 5.99, NULL),
-- Limonada con poco hielo
(2, 9, 3.50, 'Con poco hielo');

-- Ingredientes personalizados del segundo pedido
-- Para la hamburguesa McRaulo Veggie (id_pedido_producto = 5)
INSERT INTO pedidos_productos_ingredientes (id_pedido_producto, id_ingr
ediente, cantidad, es_extra) VALUES
(5, 5, 1, TRUE); -- Extra queso cheddar

-- Para la hamburguesa McRaulo Pollo (id_pedido_producto = 6)
INSERT INTO pedidos_productos_ingredientes (id_pedido_producto, id_ingr
ediente, cantidad, es_extra) VALUES
(6, 7, 0, FALSE); -- Sin tomate

-- EJEMPLO ADICIONAL: Pedido con múltiples hamburguesas iguales pero
personalizadas diferente
INSERT INTO pedidos (fecha_hora, estado, total, metodo_pago) VALUES
(1716229200, 'en_preparacion', 45.96, 'tarjeta'); -- 20 mayo 2025 15:00:00
UTC

-- Pedido 3: 3 McRaulo Cheese (cada una personalizada diferente)
INSERT INTO pedidos_productos (id_pedido, id_producto, subtotal, notas)
```

```
VALUES
-- Primera McRaulo Cheese - normal
(3, 1, 8.99, 'Normal, sin modificaciones'),
-- Segunda McRaulo Cheese - sin lechuga, con bacon extra
(3, 1, 10.49, 'Sin lechuga, con bacon extra'),
-- Tercera McRaulo Cheese - sin cebolla, sin tomate, doble queso, con gua
camole
(3, 1, 11.49, 'Sin cebolla, sin tomate, doble queso, con guacamole');

-- Personalizaciones para cada hamburguesa individual
-- Segunda hamburguesa (id_pedido_producto = 8)
INSERT INTO pedidos_productos_ingredientes (id_pedido_producto, id_ingr
ediente, cantidad, es_extra) VALUES
(8, 6, 0, FALSE), -- Sin lechuga
(8, 10, 2, TRUE); -- Con bacon extra

-- Tercera hamburguesa (id_pedido_producto = 9)
INSERT INTO pedidos_productos_ingredientes (id_pedido_producto, id_ingr
ediente, cantidad, es_extra) VALUES
(9, 8, 0, FALSE), -- Sin cebolla
(9, 7, 0, FALSE), -- Sin tomate
(9, 5, 1, TRUE),  -- Queso extra (doble queso)
(9, 15, 1, TRUE); -- Con guacamole
```

API REST

Resumen de todos los endpoints disponibles:

📋 Pedidos (Endpoints principales modificados)

1. `GET /api/pedidos` - Obtener todos los pedidos
2. `GET /api/pedidos/:id` - Obtener pedido específico con detalles
3. `POST /api/pedidos` - **Crear nuevo pedido (MODIFICADO)**
4. `PATCH /api/pedidos/:id/estado` - Actualizar estado
5. `DELETE /api/pedidos/:id` - Eliminar pedido

6. `GET /api/pedidos/estado/:estado` - Filtrar por estado
7. `GET /api/pedidos/:idPedido/productos/:idPedidoProducto` - Detalle de producto específico
8. `GET /api/pedidos/estadisticas/resumen` - **Estadísticas (MODIFICADO)**
9. `POST /api/pedidos/:id/productos` - **Agregar productos (MODIFICADO)**
10. `GET /api/pedidos/filtro/fecha` - Filtrar por fechas
11. `GET /api/pedidos/:id/resumen` - **Resumen agrupado (NUEVO)**
12. `DELETE /api/pedidos/:id/productos/:idProducto` - **Eliminar producto específico (NUEVO)**

🍔 Productos

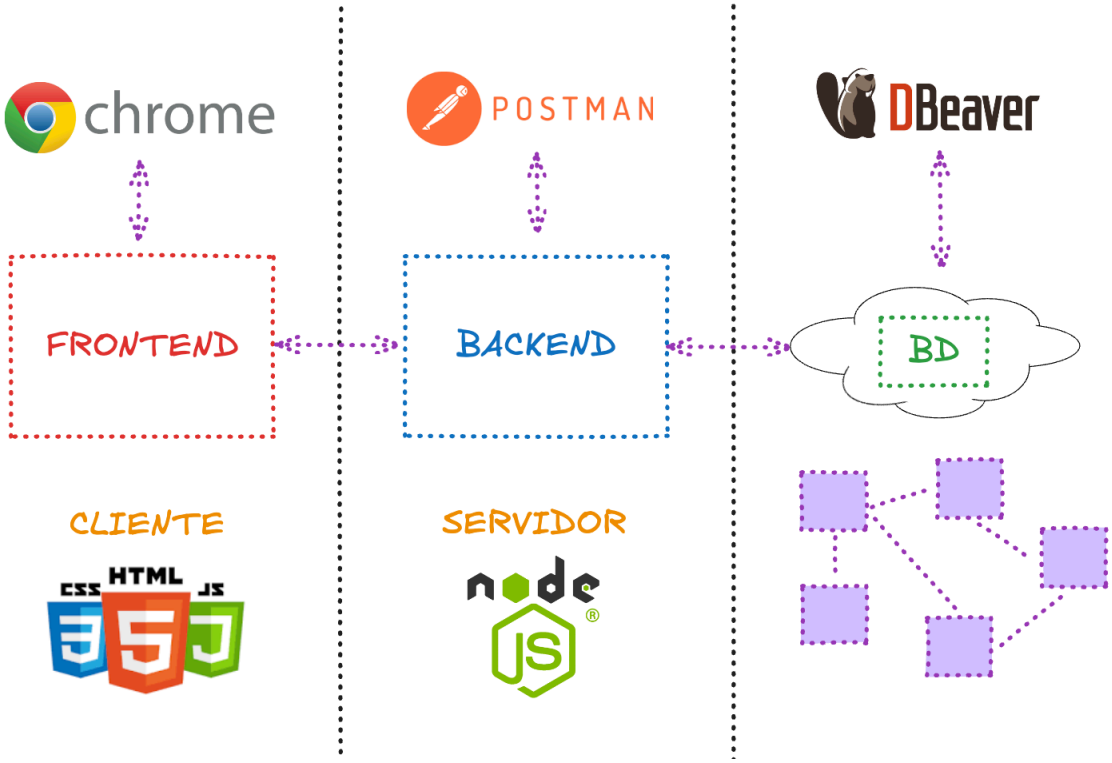
1. `GET /api/productos` - Todos los productos
2. `GET /api/productos/:id` - Producto específico con ingredientes base
3. `GET /api/productos/categoria/:categoria` - Productos por categoría
4. `POST /api/productos/:id/calcular-precio` - **Calcular precio personalizado (NUEVO)**

🥬 Ingredientes

1. `GET /api/ingredientes` - Todos los ingredientes
2. `GET /api/ingredientes/:id` - Ingrediente específico

📊 Adicionales

1. `GET /api/categorias` - **Obtener categorías (NUEVO)**
2. `GET /api/pedidos/metodo-pago/:metodo` - **Pedidos por método de pago (NUEVO)**



Repo

https://github.com/raulborda/pp1-burger

Formas normales según Claude AI

Normalización del DER - Sistema de Pedidos

Tablas Normalizadas (3FN)

1. ESTADO_PEDIDO

```
ESTADO_PEDIDO (
    idEstadoPedido PK,
    nombre
)
```

2. TIPO_CONTRIBUYENTE

```
TIPO_CONTRIBUYENTE (
    idTipoContribuyente PK,
    nombre
)
```

3. CLIENTE

```
CLIENTE (
    idCliente PK,
    nombre,
    usuario,
    contraseña,
    email,
    telefono,
    idTipoContribuyente FK
)
```

Cambio: Movimos idTipoContribuyente aquí desde PEDIDO

4. TIPO_PAGO

```
TIPO_PAGO (
    idTipoPago PK,
    nombre
)
```

5. CUPON

```
CUPON (
    idCupon PK,
    fechaEmision,
    fechaVencimiento
)
```

6. CUPON_PREDEF

```
CUPON_PREDEF (
    idCuponPredef PK,
    porcentajeDescuento,
    costoPuntos
```

7. PEDIDO

```
PEDIDO (
    idPedido PK,
    fechaHora,
    idCliente FK,
    idEstadoPedido FK,
    idCuponDescuento FK
)
```

Cambios:

- Eliminamos `precio` (se calcula)
- Eliminamos `idTipoContribuyente` (va en CLIENTE)

8. TIPO_PRODUCTO

```
TIPO_PRODUCTO (
    idTipoProducto PK,
    nombre
)
```

9. PRODUCTO

```
PRODUCTO (
    idProducto PK,
    nombre,
    descripcion,
    precio,
    idTipoProducto FK
)
```

10. INGREDIENTES

```
INGREDIENTES (
    idIngrediente PK,
    nombre,
    precio
)
```

11. PRODUCTO_PREDEF

```
PRODUCTO_PREDEF (
    idProducto FK,
    idIngrediente FK,
    cantidad,
    PRIMARY KEY (idProducto, idIngrediente)
)
```

12. DETALLE_PEDIDO

```
DETALLE_PEDIDO (
    idDetallePedido PK,
    idPedido FK,
    idProducto FK,
    cantidad,
    precioUnitario
)
```

Cambios: Agregamos `cantidad` y `precioUnitario`

13. EXTRAS

```
EXTRAS (
    idExtra PK,
    idDetallePedido FK,
    idIngrediente FK,
    cantidad
)
```

Cambio: Normalizamos la relación multivaluada

14. PAGO

```
PAGO (
    idPago PK,
    idPedido FK,
    idTipoPago FK,
    monto,
    descripcion
)
```

Dependencias Funcionales Principales

CLIENTE

- idCliente → nombre, usuario, contraseña, email, telefono, idTipoContribuyente

PEDIDO

- idPedido → fechaHora, idCliente, idEstadoPedido, idCuponDescuento

DETALLE_PEDIDO

- idDetallePedido → idPedido, idProducto, cantidad, precioUnitario
- (idPedido, idProducto) → cantidad, precioUnitario

PRODUCTO

- idProducto → nombre, descripcion, precio, idTipoProducto

Beneficios de la Normalización

✅ 1FN Alcanzada

- Eliminamos grupos repetitivos en EXTRAS
- Cada campo contiene valores atómicos

✅ 2FN Alcanzada

- Eliminamos dependencias parciales
- `precio` del pedido se calcula, no se almacena
- `idTipoContribuyente` movido a CLIENTE

✅ 3FN Alcanzada

- Eliminamos dependencias transitivas
- `Puntos` del cliente se puede calcular desde historial
- Datos derivados como `precio` total no se almacenan

Cálculos Derivados

```
-- Precio total del pedido
SELECT SUM(cantidad * precioUnitario) as precioTotal
FROM DETALLE_PEDIDO
WHERE idPedido = ?

-- Puntos del cliente
SELECT SUM(precioTotal * 0.1) as puntos
FROM PEDIDO P
JOIN (SELECT idPedido, SUM(cantidad * precioUnitario) as precioTotal
      FROM DETALLE_PEDIDO GROUP BY idPedido) D ON P.idPedido = D.idP
edido
WHERE P.idCliente = ?
```

Mejoras Adicionales

1. **Auditoría**: Agregar campos de timestamp y usuario para trazabilidad
2. **Soft Delete**: Campos `activo` o `fechaEliminacion` en lugar de DELETE físico
3. **Índices**: Crear índices en FK frecuentemente consultadas
4. **Constraints**: Agregar CHECK constraints para validaciones de negocio