2do Avance Proyecto Final

Daniel Santiago Suancha Araujo – 20221578057

Juan Jose Acevedo Pinzon - 20221578121

Claudia Liliana Hernández García

Docente

Bases de Datos Avanzadas

UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Junio de 2025

ENUNCIADO

El restaurante "COLOMBIANO, COMA CONTENTO" requiere una solución informática para el proceso de elaboración de menú de la comida típica colombiana que ofrece. Actualmente, el restaurante maneja unos platos en su carta que abarcan las diferentes regiones del país (Andina, Llanos orientales, Caribe, Pacífica y Amazonía). Cada plato tiene un nombre, una descripción, un nivel de complejidad en su preparación, una foto y el precio de venta al público. Estos platos se encuentran clasificados en la carta de acuerdo con la región del país a la cual pertenece y que dentro del restaurante tiene un encargado o responsable.

El restaurante necesita que la aplicación permita crear los platos que serán incluidos en la carta, de tal forma que se puedan ingresar los datos básicos y además la lista de ingredientes con su cantidad requerida, las unidades de medida de estos ingredientes y una breve descripción de la receta. Además, debe ser posible generar la carta y su vigencia para el restaurante y conservar los datos de las cartas definidas con anterioridad a la actual. Debe proporcionarse la opción de observar las fotografías de los platos que se estén consultando. También debe considerarse, que cada plato puede pertenecer a una categoría de comida (sopa, carnes, pollo, postre, arroces, entre otros) y que puede ofrecerse como desayuno, almuerzo o cena.

Para decidir si se requiere generar una nueva carta, es necesario que la aplicación permita:

a.Consultar mensual y anualmente las ventas en dinero y en cantidad de los diferentes platos.

b.Consultar mensual y anualmente la cantidad de platos por cada región.

c.ConsultaClaro, puedo ayudarte a revisar un documento en términos de coherencia. Por favor, copia y pega el texto del documento aquí, y haré lo posible por analizarlo y darte feedback sobre su cohesión.

r mensual y anualmente la cantidad de platos vendidos como desayuno, como almuerzo y como cena.

1. Definición del problema
    a. Descripción de funcionalidades (aplicación)

Funcionalidad	Descripción	Rol
Registro de Platos	Permite ingresar nuevos platos al sistema, capturando información como nombre, descripción, nivel de complejidad, foto, precio, región, categoría y turno.	Administrador
Gestión de Ingredientes	Registro de ingredientes con su nombre y unidad de medida, para ser utilizados en la creación de recetas.	Administrador
Creación de Recetas	Asociación de platos con ingredientes, indicando la cantidad requerida y una descripción de la preparación.	Administrador / Chef
Generación de Cartas de Menú	Permite crear una nueva carta de platos vigente por un periodo definido, conservando versiones anteriores para consulta histórica.	Administrador
Visualización de Platos	Consulta de platos registrados, incluyendo visualización de la foto y sus detalles relacionados.	Administrador / Cliente

Consultas de Ventas por Plato	Generación de reportes mensuales y anuales con datos de ventas en dinero y cantidad por cada plato.	Administrador
Consultas por Región	Reporte de la cantidad de platos vendidos por región en rangos mensuales y anuales.	Administrador
Consultas por Turno	Reporte de ventas mensuales y anuales según el turno en que se ofreció el plato (desayuno, almuerzo o cena).	Administrador

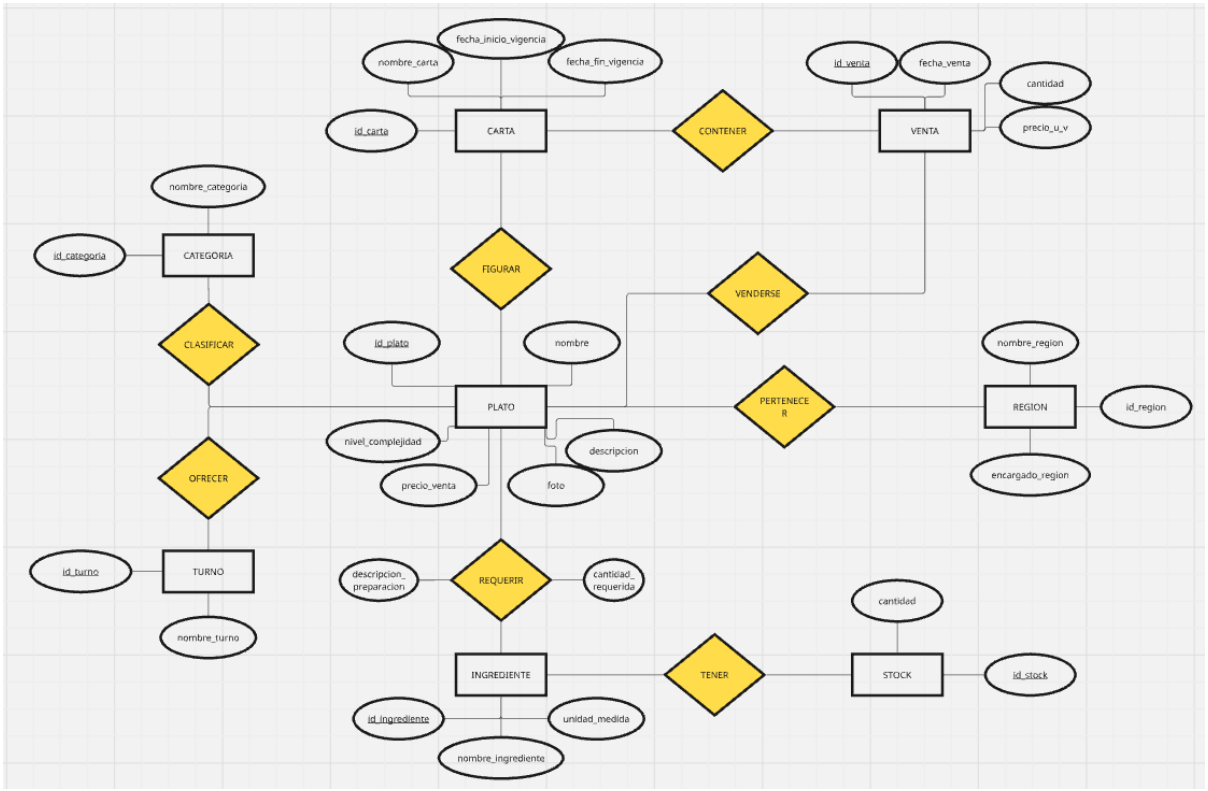
b. Descripción del entorno de ejecución (S.O., Motor de BD…..)

Para nuestro entorno de ejecución, hemos optado por utilizar Java como Framework con interfaces web JSP para nuestro proyecto debido a su capacidad de conectar bases de datos a través de JDBC y la facilidad que posee para crear interfaces web dinámicas mediante JSP. Además de esto, el desarrollo anterior de otros developers permiten una recursividad y documentación que nos brinda un sólido soporte para el desarrollo. Esta elección nos permitirá crear una aplicación eficiente y posiblemente escalable que cumple con los requisitos para en debido desarrollo de nuestro proyecto. El sistema operativo elegimos es Windows 10 o Windows 11 debido a su amplia disponibilidad y compatibilidad, lo que facilitará el desarrollo y la ejecución de nuestra aplicación en la mayoría de los entornos, aparte de ser este sistema más usado por muchas empresas y usuarios. Respecto a postgreSQL 15, su elección se debe a su accesibilidad, disponibilidad y facilidad de uso, esto nos permitirá diseñar y gestionar la base de datos de manera eficaz. Esta combinación de herramientas nos brindará una infraestructura sólida y amigable para nuestro proyecto.

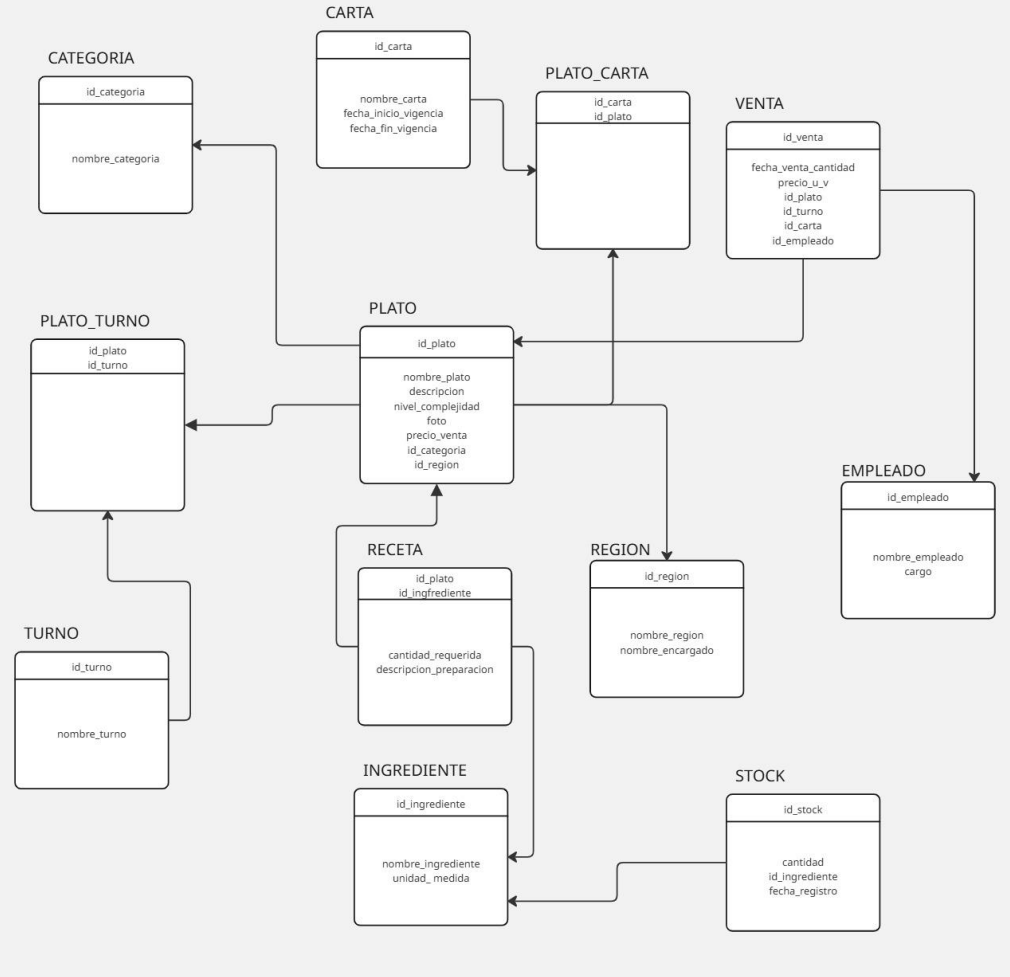
c. Definición de las reglas de negocio (delimitar problema)

- Cada plato tiene un nombre, una descripción, un nivel de complejidad en su preparación, una foto y el precio de venta al público.
- Cada plato posee un encargado o responsable.
- Cada plato pertenece a una categoría, y estos también se ofrecen en los distintos horarios de comida.
- Cada plato poseerá un nombre único dentro de su respectiva región.
- no pueden existir dos cartas activas con periodos que se solapen.
- No se podrá vender un plato si falta alguno de sus ingredientes en stock.
- No se podrá eliminar una región, categoría, turno o ingrediente que esté asociado a un plato o carta activos.
- Los cálculos de ventas deben considerar la fecha de facturación y el tipo de servicio (desayuno, almuerzo o cena) al momento de la transacción.
- La aplicación debe permitir consultar ventas y cantidades de platos por región y turno, tanto mensual como anualmente.
- La vigencia de cada carta debe estar claramente definida y no debe solaparse con otras cartas activas.
- Un plato puede pertenecer a múltiples cartas.
- Cada carta debe tener al menos un plato

2. Diagrama del modelo Entidad Relación



3. Diagrama del modelo relacional



Análisis de Normalización Paso a Paso por Tabla

A continuación, se analiza cada tabla para determinar su nivel de normalización hasta la 3FN:

- CATEGORIA: Cumple con 1FN (clave primaria y atributos atómicos). Cumple con 2FN (clave primaria simple). Cumple con 3FN (no hay dependencias transitivas). Estado: 3FN.
- CARTA: Cumple con 1FN. Cumple con 2FN (clave primaria simple). Cumple con 3FN (no hay dependencias transitivas). Estado: 3FN.
- PLATO_CARTA: Cumple con 1FN (clave primaria compuesta). Cumple con 2FN (no hay atributos no clave). Cumple con 3FN (solo claves primarias). Estado: 3FN.
- VENTA: Cumple con 1FN. Cumple con 2FN (clave primaria simple). Cumple con 3FN (los atributos no clave dependen directamente de id_venta). Estado: 3FN.
- PLATO: Cumple con 1FN. Cumple con 2FN (clave primaria simple). Cumple con 3FN (los atributos no clave dependen directamente de id_plato). Estado: 3FN.

- REGION: Cumple con 1FN. Cumple con 2FN (clave primaria simple). Cumple con 3FN (los atributos no clave dependen directamente de id_region). Estado: 3FN.

4. Diccionario de datos
   a. Descripción de tablas y columnas

0. TABLA GENERAL

Tabla	Descripción general
PLATO	Almacena información básica y detallada de los platos que ofrece el restaurante.
INGREDIENTE	Registra los ingredientes utilizados en la preparación de los platos, con sus unidades de medida y stock.
CARTA	Contiene las cartas del menú, con información de vigencia y los platos asociados.
REGION	Define las regiones del país a las que pertenecen los platos.
CATEGORIA	Clasifica los platos por tipo (sopa, carnes, pollo, postre, arroces, etc.).
TURNO	Indica los turnos de comida en los que se ofrece cada plato (desayuno, almuerzo, cena).
VENTA	Registra las ventas de los platos, incluyendo fecha, cantidad, valor, región y turno.
EMPLEADO	Contiene la informacion basica del empleado (nombre, id y cargo)

1. CATEGORIA

Descripción general:
Almacena las categorías de los platos (por ejemplo: sopa, carnes, postre), permitiendo agrupar y filtrar los platos según su tipo culinario.
Observaciones / Reglas de negocio:

- El nombre de categoría debe ser único y no nulo.
- No se puede eliminar una categoría asociada a algún plato activo.

Columna	**Tipo de dato**	**Descripción**
id_categoria	SERIAL/ PK	Identificador único de la categoría
nombre_categoria	VARCHAR	Nombre de la categoría del plato (ej. Sopa, Carnes)

2. TURNO

Descripción general:
Define los turnos de servicio (desayuno, almuerzo, cena), utilizados en ventas y en asociación de platos.
Observaciones / Reglas de negocio:

- El nombre de turno debe ser único.
- No se puede eliminar un turno asociado a ventas o a asignaciones de plato.

Columna	Tipo de dato	Descripción
id_turno	INT / PK	Identificador único del turno
nombre_turno	VARCHAR	Nombre del turno (Desayuno, Almuerzo...)

3. REGION

Descripción general:
Contiene las regiones geográficas (Andina, Caribe, Pacífica, etc.) para clasificar los platos por procedencia.
Observaciones / Reglas de negocio:

- El nombre de región debe ser único.
- No se puede eliminar una región con platos activos en cartas.

Columna	Tipo de dato	Descripción
id_region	SERIAL/ PK	Identificador único de la región
nombre_region	VARCHAR	Nombre de la región (Andina, Caribe, etc.)
nombre_encargado	VARCHAR	Nombre del encargado de la región

4. PLATO

Descripción general:
Contiene la información de cada plato: nombre, descripción, complejidad, foto, precio, región y categoría.
Observaciones / Reglas de negocio:

- Nombre único por región.
- Nivel de complejidad entre 1 y 5.
- Precio > 0.
- No eliminar si está en carta activa.

Columna	Tipo de dato	Descripción
id_plato	SERIAL/ PK	Identificador único del plato
nombre_plato	VARCHAR	Nombre asociado al plato
descripcion	TEXT	Breve descripción del plato
nivel_complejidad	INT	Nivel de complejidad (1 a 5, por ejemplo)
foto	VARCHAR / URL	Enlace o nombre del archivo de imagen
precio_venta	DECIMAL	Precio del plato al público
id_categoria	INT / FK	Categoría del plato

id_region	INT / FK	Región a la que pertenece el plato

5. PLATO_TURNO

Descripción general:
Relaciona cada plato con los turnos en los que está disponible (desayuno, almuerzo, cena).
Observaciones / Reglas de negocio:

- Combinación única de plato y turno.

Columna	**Tipo de dato**	**Descripción**
id_plato	INT / PK+FK	Plato ofrecido
id_turno	INT / PK+FK	Turno en que se ofrece ese plato

6. CARTA

Descripción general:
Define las cartas (menús) con su periodo de vigencia, sin solapamientos.
Observaciones / Reglas de negocio:

- No pueden solaparse períodos de cartas activas (trigger valida OVERLAPS).

Columna	**Tipo de dato**	**Descripción**
id_carta	SERIAL/ PK	Identificador único de la carta
nombre_carta	VARCHAR	Nombre de la carta (ej. Carta febrero 2025)
fecha_inicio_vigencia	DATE	Fecha de inicio de validez de la carta
fecha_fin_vigencia	DATE	Fecha final de vigencia de la carta

7. PLATO_CARTA

Descripción general:
Asocia platos a cartas, determinando qué platos incluye cada menú.
Observaciones / Reglas de negocio:

- Combinación única de carta y plato.

Columna	**Tipo de dato**	**Descripción**
id_plato	INT / PK+FK	Plato incluido en la carta
id_carta	INT / PK+FK	Carta en la que aparece ese plato

8. INGREDIENTE

Descripción general:
Almacena los ingredientes disponibles con su unidad de medida, para ser usados en recetas y control de stock.
Observaciones / Reglas de negocio:

- El nombre de ingrediente debe ser único.
- No se puede eliminar un ingrediente presente en recetas o en stock.

Columna	**Tipo de dato**	**Descripción**
id_ingrediente	SERIAL/ PK	Identificador único del ingrediente
nombre_ingrediente	VARCHAR	Nombre del ingrediente
unidad_medida	VARCHAR	Unidad de medida (gramos, unidades, litros...)

9. RECETA

Descripción general:
Detalla los ingredientes y cantidades requeridas para preparar cada plato, con descripción de la preparación.
Observaciones / Reglas de negocio:

- Cada combinación plato–ingrediente debe ser única.

Columna	**Tipo de dato**	**Descripción**
id_plato	INT / PK+FK	Plato que requiere el ingrediente
id_ingrediente	INT / PK+FK	Ingrediente requerido
cantidad_requerida	DECIMAL	Cantidad necesaria del ingrediente
descripcion_preparacion	TEXT	Instrucciones o pasos de preparación

10. STOCK

Descripción general:
Lleva el registro de existencias de cada ingrediente, incluyendo movimientos de entrada y salida.
Observaciones / Reglas de negocio:

- No permitir stock negativo; trigger valida antes de restar.
- Cada registro es histórico, indicar fecha de movimiento.

Columna	**Tipo de dato**	**Descripción**
id_stock	INT / PK	Identificador del registro de stock
id_ingrediente	INT / FK	Ingrediente al que corresponde el stock

cantidad	DECIMAL	Cantidad disponible en existencia
fecha_registro	TIMESTAMP DEFAULT CURRENT_TIMESTAMP	Fecha y hora del movimiento de stock.

11. VENTA

Descripción general:
Registra cada venta: fecha, cantidad, precio unitario, plato, turno, carta y empleado responsable.
Observaciones / Reglas de negocio:

- Antes de insertar, trigger valida stock suficiente y actualiza existencias.
- No se puede vender si falta ingrediente.

Columna	**Tipo de dato**	**Descripción**
id_venta	INT / PK	Identificador único de la venta
fecha_venta	DATE	Fecha de la transacción
cantidad	INT	Cantidad de platos vendidos
precio_u_v	DECIMAL	Precio unitario del plato
id_plato	INT / FK	Plato vendido
id_turno	INT / FK	Turno de la venta
id_carta	INT / FK	Carta en la que se encuentra el plato vendido
Id_empleado	INT/FK	Empleado que registro la venta

12. EMPLEADO

Descripción general:
Registra los empleados que realizan ventas y gestionan cartas, incluyendo meseros y administradores.
Observaciones / Reglas de negocio:

- Cada empleado debe tener un rol y nombre únicos.
- No se puede eliminar un empleado que figure en ventas o asignaciones de carta.

Columna	**Tipo de dato**	**Descripción**
id_empleado	SERIAL/ PK	Identificador del empleado
nombre_empleado	VARCHAR	Nombre del empleado
cargo	VARCHAR	Cargo que posee el empleado

b. Descripción de vistas

1. vista_ventas_mensuales

Descripción general:
Permite consultar las ventas de cada plato agrupadas por año, mes, región y turno, mostrando tanto la cantidad vendida como el monto total de ingresos.
Observaciones / Reglas:

- Solo incluye ventas registradas en la tabla venta.
- Agrupa por año, mes, nombre de plato, región y turno.
- Ordena cronológicamente y por región/turno.

Codigo SQL:

```
CREATE OR REPLACE VIEW vista_ventas_mensuales AS
SELECT
        EXTRACT(YEAR FROM v.fecha_venta) AS año,
        EXTRACT(MONTH FROM v.fecha_venta) AS mes,
         p.nombre_plato AS plato, r.nombre_region AS región,
        t.nombre_turno AS turno, SUM(v.cantidad) AS unidades_vendidas,
        SUM(v.cantidad * v.precio_u_v) AS ingresos_totales
FROM venta v
JOIN plato p ON v.id_plato = p.id_plato
JOIN region r ON p.id_region = r.id_region
JOIN turno t ON v.id_turno = t.id_turno
GROUP BY año, mes, plato, región, turno
ORDER BY año, mes, región, turno, plato;
```

2. vista_ventas_anuales

Descripción general:
Sintetiza las ventas por año, agrupando por plato, región y turno, para análisis de tendencias anuales.
Observaciones / Reglas:

- Agrupa solo por año, plato, región y turno.
- Facilita comparaciones interanuales de desempeño.

Codigo SQL:

```
CREATE OR REPLACE VIEW vista_ventas_anuales AS
SELECT
        EXTRACT(YEAR FROM v.fecha_venta) AS año,
        p.nombre_plato AS plato, r.nombre_region AS región,
        t.nombre_turno AS turno, SUM(v.cantidad) AS unidades_vendidas,
        SUM(v.cantidad * v.precio_u_v) AS ingresos_totales
FROM venta v
JOIN plato p ON v.id_plato = p.id_plato
JOIN region r ON p.id_region = r.id_region
JOIN turno t ON v.id_turno = t.id_turno
```

```
GROUP BY año, plato, región, turno
ORDER BY año, región, turno, plato;
```

3. vista_platos_region_turno_mensual

Descripción general:
Muestra el número total de platos vendidos cada mes por región y turno, sin detalle de plato individual.
Observaciones / Reglas:

- Útil para comparar rendimiento mensual entre regiones y turnos.
- Agrupa por año, mes, región y turno.

Codigo SQL:

```
CREATE OR REPLACE VIEW vista_platos_region_turno_mensual AS
SELECT
        EXTRACT(YEAR FROM v.fecha_venta) AS año,
        EXTRACT(MONTH FROM v.fecha_venta) AS mes,
        r.nombre_region AS región, t.nombre_turno AS turno,
        SUM(v.cantidad) AS total_platos
FROM venta v
JOIN plato p ON v.id_plato = p.id_plato
JOIN region r ON p.id_region = r.id_region
JOIN turno t ON v.id_turno = t.id_turno
GROUP BY año, mes, región, turno
ORDER BY año, mes, región, turno;
```

4. vista_platos_region_turno_anual

Descripción general:
Agrega las ventas de platos por región y turno a nivel anual.
Observaciones / Reglas:

- Permite identificar regiones/turnos con mayor volumen en el año.
- Agrupa por año, región y turno.

Código SQL:

```
CREATE OR REPLACE VIEW vista_platos_region_turno_anual AS
SELECT
        EXTRACT(YEAR FROM v.fecha_venta) AS año,
        r.nombre_region AS región,
        t.nombre_turno AS turno,
        SUM(v.cantidad) AS total_platos
FROM venta v
JOIN plato p ON v.id_plato = p.id_plato
```

```
JOIN region r ON p.id_region = r.id_region
JOIN turno t ON v.id_turno = t.id_turno
GROUP BY año, región, turno
ORDER BY año, región, turno;
```

5. vista_platos_por_carta

Descripción general:
Lista los platos de cada carta, indicando fechas de vigencia para distinguir menús activos e históricos.
Observaciones / Reglas:

- Permite filtrar menús por fecha.
- Ordena por fecha de inicio, nombre de carta y plato.

Código SQL:

```
CREATE OR REPLACE VIEW vista_platos_por_carta AS
SELECT
    c.nombre_carta,
    c.fecha_inicio_vigencia,
    c.fecha_fin_vigencia,
    p.nombre_plato,
    p.descripcion,
    p.precio_venta
FROM carta c
JOIN plato_carta pc ON c.id_carta = pc.id_carta
JOIN plato p ON pc.id_plato = p.id_plato
ORDER BY c.fecha_inicio_vigencia, c.nombre_carta, p.nombre_plato;
```

c. Descripción de procedimientos y triggers

Procedimiento Almacenado Gestión Automática de Stock tras Venta

Descripción General

Antes de insertar un nuevo registro en la tabla venta, verifica que exista stock suficiente de cada ingrediente necesario para la preparación del plato vendido y, de ser así, genera movimientos de salida en la tabla stock para reflejar la reducción de inventario.
Flujo de operación:

1. Recorre la receta del plato (receta) para obtener cada ingrediente y la cantidad requerida.
2. Para cada ingrediente:
    a. Consulta el último registro de stock disponible.
    b. Si no existe o la cantidad disponible es menor a la requerida (multiplicada por la cantidad de platos vendidos), lanza una excepción y aborta la inserción.

3. Una vez verificado el stock, inserta un nuevo registro en stock con cantidad negativa (igual a la cantidad consumida) y marca la fecha/hora actual.
4. Devuelve la nueva fila de venta para completar la inserción.

Observaciones / Reglas de negocio:

- Impide ventas cuando algún ingrediente carece de stock suficiente.
- Cada ajuste de inventario se conserva como un registro histórico en la tabla stock.
- Asegura la integridad referencial y la consistencia de inventario.

Codigo SQL:

```
CREATE OR REPLACE FUNCTION fn_actualizar_stock_venta()
RETURNS TRIGGER AS $$
DECLARE
        rec_receta RECORD;
        stock_actual NUMERIC;
BEGIN
        FOR rec_receta IN
                SELECT id_ingrediente, cantidad_requerida
                FROM receta
                WHERE id_plato = NEW.id_plato
        LOOP
                SELECT cantidad
                        INTO stock_actual
                FROM stock
                WHERE id_ingrediente = rec_receta.id_ingrediente
                ORDER BY fecha_registro DESC
                LIMIT 1;
                IF stock_actual IS NULL OR stock_actual <
                (rec_receta.cantidad_requerida * NEW.cantidad) THEN
          RAISE EXCEPTION 'Stock insuficiente para el ingrediente %',
        rec_receta.id_ingrediente;
                END IF;
        END LOOP;
        FOR rec_receta IN
                SELECT id_ingrediente, cantidad_requerida
                FROM receta
                WHERE id_plato = NEW.id_plato
        LOOP
                INSERT INTO stock (id_ingrediente, cantidad, fecha_registro)
                VALUES (
                rec_receta.id_ingrediente,
                -(rec_receta.cantidad_requerida * NEW.cantidad),
                CURRENT_TIMESTAMP
```

```
            );
        END LOOP;
        RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Trigger:

Se asocia a la tabla venta para invocar fn_actualizar_stock_venta() antes de cada intento de inserción de una venta.
Descripción del proceso:

- Al recibir un INSERT sobre venta, el trigger dispara fn_actualizar_stock_venta().
- Si la función valida correctamente el stock y realiza los movimientos de inventario, la venta se inserta finalmente.
- Si se produce una excepción (falta de stock), el trigger detiene la operación y notifica el error.

Esto permite tener una automatizaion en el inventario en tiempo real, evita la inconsistencia de datos al garantizar que no exista ventas registradas sin sus correspondientes descontajes del stock y facilita trazar los movimientos de los ingredientes

Codigo SQL:

```
CREATE TRIGGER tg_venta_actualiza_stock
BEFORE INSERT ON venta
FOR EACH ROW
EXECUTE FUNCTION fn_actualizar_stock_venta();
```

5. Código fuente

a. Estructura de la base de datos

La base de datos del restaurante “COLOMBIANO, COMA CONTENTO” se sustenta en un esquema de tablas interrelacionadas que cubren entidades clave como Categoría, Turno, Región, Empleado, Ingrediente, Stock, Plato, Carta, Receta y Venta. Cada tabla está diseñada para capturar aspectos específicos del negocio: desde la clasificación y disponibilidad de los platos hasta el control dinámico de inventario y el registro detallado de transacciones de venta. Estas entidades se vinculan mediante claves foráneas que garantizan la integridad referencial y permiten seguir el ciclo completo de preparación y comercialización de cada plato.

Además de este modelo relacional, la base de datos incluye vistas que facilitan el análisis de ventas mensuales y anuales, el seguimiento de los platos por carta y turno, el monitoreo de niveles de stock y el rendimiento del personal. Complementariamente, se integran funciones

almacenadas y triggers para automatizar procesos críticos, como la validación y ajuste de inventario al registrar una venta o la prevención de solapamiento de cartas activas.

b. Inserción de datos

El proceso de inserción de datos se realizó bajo la modalidad de carga masiva, planificado y ejecutado con el objetivo de integrar información detallada y precisa en el sistema del restaurante “COLOMBIANO, COMA CONTENTO”.
Primero que todo, se aseguró que los archivos estuvieran en un formato CSV adecuado y libre de inconsistencias, utilizando delimitador punto y coma (‘;’) para facilitar su lectura.

Para gestionar la carga masiva se empleó la utilidad nativa de PostgreSQL \copy, que permite importar eficientemente grandes volúmenes de datos. Antes de cada carga, se validaron los tipos de datos y la correspondencia de columnas entre los CSV y las tablas de la base de datos, evitando así errores o duplicidades.

Durante la ejecución, se monitoreó constantemente la salida de errores. Una vez finalizada la inserción, se llevaron a cabo consultas de verificación (por ejemplo, conteos de registros por tabla) para garantizar la integridad y exactitud de la información.

Los archivos CSV para la carga masiva se encuentran alojados en:

C:\Documentos\DATOS_RESTAURANTE

Las rutas específicas y comandos \copy para cada tabla son los siguientes:

```
\copy categoria FROM
'C:\Users\zudex\OneDrive\Documentos\DATOS_RESTAURANTE\categoria.csv' WITH
(FORMAT csv, DELIMITER ';', HEADER true);
\copy turno FROM
'C:\Users\zudex\OneDrive\Documentos\DATOS_RESTAURANTE\turno.csv' WITH
(FORMAT csv, DELIMITER ';', HEADER true);
 \copy region FROM
'C:\Users\zudex\OneDrive\Documentos\DATOS_RESTAURANTE\region.csv' WITH
(FORMAT csv, DELIMITER ';', HEADER true);
\copy empleado FROM
'C:\Users\zudex\OneDrive\Documentos\DATOS_RESTAURANTE\empleado.csv' WITH
(FORMAT csv, DELIMITER ';', HEADER true);
\copy ingrediente FROM
'C:\Users\zudex\OneDrive\Documentos\DATOS_RESTAURANTE\ingrediente.csv' WITH
(FORMAT csv, DELIMITER ';', HEADER true);
\copy stock FROM
'C:\Users\zudex\OneDrive\Documentos\DATOS_RESTAURANTE\stock.csv' WITH
(FORMAT csv, DELIMITER ';', HEADER true);
```

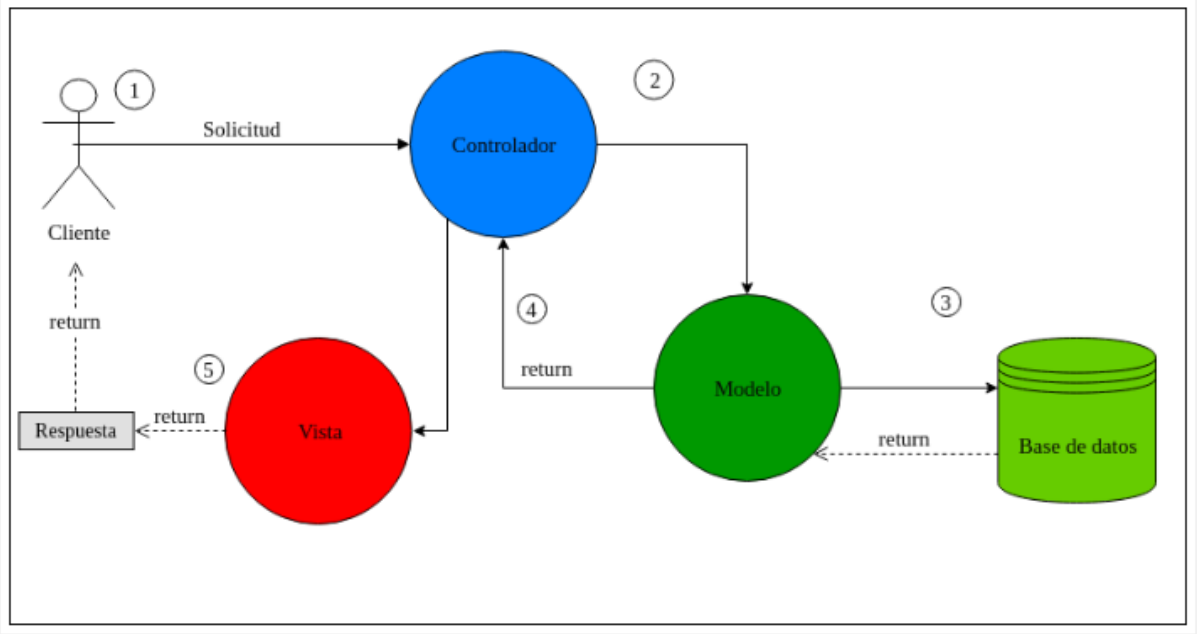
```
\copy plato FROM
'C:\Users\zudex\OneDrive\Documentos\DATOS_RESTAURANTE\plato.csv' WITH
(FORMAT csv, DELIMITER ';', HEADER true);
\copy plato_turno FROM
'C:\Users\zudex\OneDrive\Documentos\DATOS_RESTAURANTE\plato_turno.csv' WITH
(FORMAT csv, DELIMITER ';', HEADER true);
\copy carta FROM
'C:\Users\zudex\OneDrive\Documentos\DATOS_RESTAURANTE\carta.csv' WITH
(FORMAT csv, DELIMITER ';', HEADER true);
\copy plato_carta FROM
'C:\Users\zudex\OneDrive\Documentos\DATOS_RESTAURANTE\plato_carta.csv' WITH
(FORMAT csv, DELIMITER ';', HEADER true);
\copy receta FROM
'C:\Users\zudex\OneDrive\Documentos\DATOS_RESTAURANTE\receta.csv' WITH
(FORMAT csv, DELIMITER ';', HEADER true);

SELECT setval( pg_get_serial_sequence('stock','id_stock'), (SELECT
COALESCE(MAX(id_stock),0) FROM stock) );
(por el trigger fn_actualizar_stock_venta() inserta movimientos en la tabla stock sin
especificar el campo id_stock, confiando en el SERIAL, Esto fuerza a la secuencia interna de
stock.id_stock a partir de MAX(id_stock)+1. A partir de ahí, cada nuevo INSERT (ya sea
manual, mediante COPY o trigger) obtendrá un id_stock único y creciente, evitando la
violación de la clave primaria )

\copy venta FROM
'C:\Users\zudex\OneDrive\Documentos\DATOS_RESTAURANTE\venta.csv' WITH
(FORMAT csv, DELIMITER ';', HEADER true);
```

c. Conexión de la aplicación a la BD

En la implementación de la base de datos del restaurante “COLOMBIANO, COMA CONTENTO” mediante una aplicación web, se adoptó el Modelo Vista Controlador (MVC). MVC facilita la separación clara y estructurada de responsabilidades:
El Modelo encapsula la lógica de acceso y manipulación de datos en la base de datos PostgreSQL, representando las entidades del negocio (Plato, Venta, Ingrediente, Carta, etc.) mediante clases Java que utilizan JDBC para ejecutar consultas y comandos SQL.



Referencia encontrada en: *Crear entregables*. (2019, 1 junio). Línea 1 del Metro de Lima. https://softwaremetro1.wordpress.com/crear-entregables/

La Vista está formada por páginas JSP que presentan la interfaz de usuario, mostrando formularios para registrar platos, generar cartas, consultar ventas o visualizar reportes. Estas vistas reciben los datos del controlador y los renderizan en HTML, CSS y JavaScript.
El Controlador consiste en servlets que reciben las peticiones HTTP, validan parámetros, invocan a las clases del modelo (o a los DAOs) y despachan la respuesta adecuada a la vista correspondiente. De esta manera, el controlador actúa como intermediario, dando el flujo de información entre usuario y base de datos.
Adicionalmente, se emplea el Patrón DAO (Data Access Object) para desacoplar la lógica de acceso a datos de la lógica de negocio. Cada entidad dispone de su propio DAO (por ejemplo, PlatoDAO, VentaDAO, IngredienteDAO, CartaDAO), responsables de:

- Abrir y cerrar conexiones JDBC de manera centralizada.
- Ejecutar sentencias SQL preparadas para operaciones CRUD.
- Mapear resultados de ResultSet a objetos de dominio.

Este patrón abstrae los detalles de la base de datos, facilitando la reutilización y permitiendo reemplazar o extender la capa de persistencia sin afectar la lógica de negocio ni la presentación.

Usar estos dos modelos (MVC+DAO) beneficia enormemente la modularidad y separacion de capas, facilitando el mantenimiento y la escalabilidad, poder reutilizar el codigo al centralizar el acceso a datos en los DAOs, da seguridad y control de transacciones al gestionar conexiones y excepciones en una capa dedicada

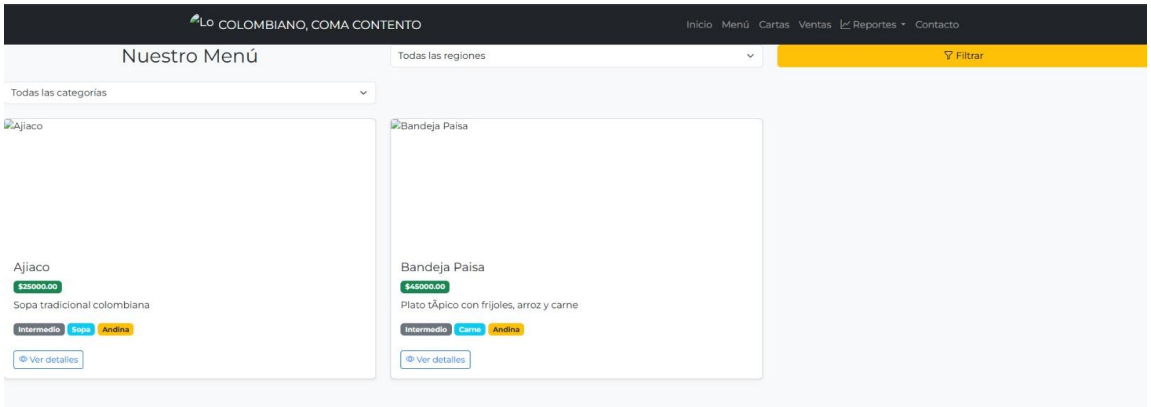
Un escema simplificado del flujo de interaccion es:

1. El usuario realiza una acción (por ejemplo, “Registrar nueva venta”) en la Vista (JSP/formulario).
2. La petición llega al Controlador (VentaController), que extrae y valida los parámetros.
3. El controlador invoca al Modelo a través de VentaDAO.registrarVenta(venta).
4. VentaDAO abre una conexión JDBC, ejecuta el INSERT en la tabla venta y cierra la conexión.
5. Tras el éxito de la operación, el controlador redirige a la vista de confirmación o informe de ventas.

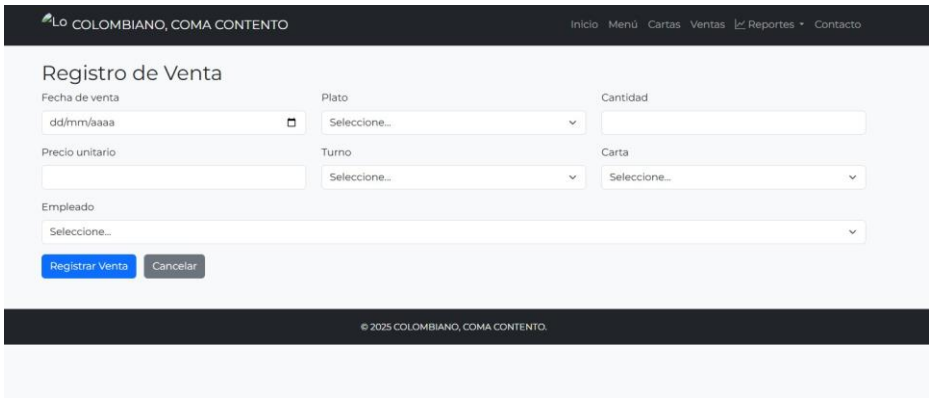
Este diseño garantiza que el usuario nunca interactúe directamente con la base de datos, preservando la integridad y la consistencia de los datos, al mismo tiempo que mantiene una arquitectura limpia y extensible.

6. Descripción de la aplicación (Manual de usuario)

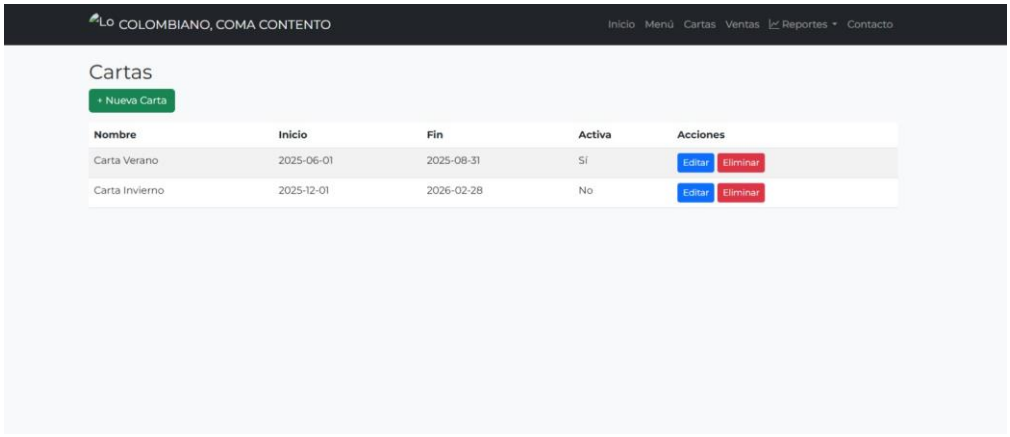
Menú Principal: Accede a Platos, Cartas, Recetas, Stock, Ventas, Reportes y Empleados desde la interfaz principal.



Registro de Ventas: Ve a Ventas → selecciona fecha, plato, cantidad, turno, carta y empleado → guarda (el sistema valida el stock automáticamente).



Gestión de Cartas: En Cartas → crea o edita carta con nombre, fechas de vigencia y estado → asigna platos → guarda (no permite solapamientos activos)



Consultas y Reportes: Revisa vistas como ventas mensuales/anuales, platos por región y turno, y menú por carta; disponibles para análisis o exportaciones.