



Diseño y Desarrollo de Sistemas de Información

UNIVERSIDAD
DE GRANADA

PRÁCTICA

D2 La Botella Infinita



**Doble Grado Ing.
Informática y ADE**

**José Antonio Fernández
Aranda**

Alejandro Coman Venceslá

Francisco José Ramos

Moya

Ignacio Rojas Valenzuela

Pablo Anel Rancaño

ÍNDICE

1. Subsistema de Gestión de Inventario (José Antonio Fernández Aranda)	4
2. Subsistema de Gestión de Ventas (Ignacio Rojas Valenzuela)	6
3. Subsistema de Gestión de Clientes (Alejandro Coman Venceslá)	8
4. Subsistema de Gestión de Proveedores (Pablo Anel Rancaño)	9
5. Subsistema de Informes y Análisis (Francisco José Ramos Moya)	10
1. Subsistema de Gestión de Inventario (José Antonio Fernández Aranda)	12
2. Subsistema de Gestión de Ventas (Ignacio Rojas Valenzuela)	21
3. Subsistema de Gestión de Clientes (Alejandro Coman Venceslá).	33
4. Subsistema de Gestión de Proveedores (Pablo Anel Rancaño)	42
5. Subsistema de Informes y Análisis (Francisco José Ramos Moya)	54
DFDs	64
Caja negra	64
Armazón (DFD0)	64
DFD1s	65
DFD1 Subsistema de Gestión de Inventario	65
DFD1 Subsistema de Gestión de Ventas	65
DFD1 Subsistema de Gestión de Clientes	66
DFD1 Subsistema de Gestión de Proveedores	66
DFD1 Subsistema de Gestión de Análisis e Informes	67
Esquemas externos	67
Subsistema de gestión de inventario.	67
Subsistema de gestión de ventas.	70
Subsistema de gestión de clientes.	73
Subsistema de gestión de proveedores.	77
Subsistema de gestión de análisis e informes.	80
Esquema E/R completo del sistema	82
Paso a tablas	83
Entidades	84
Relaciones	86
Dependencias Funcionales	87
Relaciones:	88
Tablas normalizadas	89
Entidades	89
Relaciones	91
Primera Forma Normal (1FN)	92
Segunda Forma Normal (2FN)	92
Tercera Forma Normal (3FN)	93
Forma Normal de Boyce-Codd (BCNF)	94

Subsistema de Inventario	95
Sentencias de creación de tablas, con claves y restricciones	95
Descripción de las transacciones	96
Código Disparadores	101
Datos sensibles y Descripción de los aspectos legales	101
Subsistema de Ventas	102
Sentencias de creación de tablas, con claves y restricciones	102
Descripción de las transacciones	103
Código Disparadores	107
Datos sensibles y Descripción de los aspectos legales	109
Subsistema de Clientes	109
Creación de tablas, con claves y restricciones	109
Descripción de las transacciones	110
Código Disparadores	115
Datos sensibles y Descripción de los aspectos legales	115
Subsistema de Proveedores	116
Sentencias de creación de tablas, con claves y restricciones	116
Descripción de las transacciones	117
Código Disparadores	129
Datos sensibles y Descripción de los aspectos legales	130
Subsistema de Informes y Análisis	131
Sentencias de creación de tablas, con claves y restricciones	131
Descripción de las transacciones	131
Código Disparadores	140
Datos sensibles y Descripción de los aspectos legales	141
Motivación de la elección de software	142

Nombre del sistema: *Licorería*

Nombres de los subsistemas y requisitos funcionales:

Subsistema de gestión de inventario:

- RF1.1: Dar de alta un producto. (Registrar)
- RF1.2: Modificar el stock de un producto.
- RF1.3: Consultar el stock de un producto.
- RF1.4: Enviar alerta por bajo stock de un producto.
- RF1.5: Dar de baja un producto. (Descatalogar)

Subsistema de gestión de ventas:

- RF2.1: Registrar una venta.
- RF2.2: Generar recibo o factura.
- RF2.3: Aplicar descuentos o promociones.
- RF2.4: Consultar historial de ventas.
- RF2.5: Gestionar una devolución.

Subsistema de clientes:

- RF3.1: Dar de alta un cliente.
- RF3.2: Consultar historial de compras de un cliente.
- RF3.3: Editar datos del cliente.
- RF3.4: Enviar notificación a un cliente.
- RF3.5: Dar de baja un cliente.

Subsistema de proveedores:

- RF4.1: Dar de alta un proveedor.
- RF4.2: Realizar un pedido a un proveedor.
- RF4.3: Consultar historial de compras a proveedores.
- RF4.4: Cancelar un pedido a un proveedor.
- RF4.5: Marcar el pedido como completado.
- RF4.6: Dar de baja a un proveedor.

Subsistema de informes y análisis:

- RF5.1: Generar informe de ventas de un intervalo.
- RF5.2: Generar análisis de productos más vendidos en un intervalo.
- RF5.3: Generar análisis de ingresos para un producto.
- RF5.4: Generar informe de pedidos de un intervalo.
- RF5.5: Generar análisis de ventas por familia de productos.

Descripción del sistema en lenguaje natural:

Este sistema de información se encarga de gestionar de manera eficiente todas las operaciones clave de una licorería, facilitando el control de inventarios, ventas, proveedores, clientes y análisis de datos. Está compuesto por cinco subsistemas principales que cubren diferentes aspectos de la operación.

1. Subsistema de Gestión de Inventario (José Antonio Fernández Aranda)

Este subsistema se encarga de gestionar de manera eficiente los productos disponibles en el inventario, asegurando que las existencias sean suficientes para satisfacer la demanda y facilitando la toma de decisiones para reponer productos y evitar faltantes. Incluye funcionalidades para dar de alta productos, modificar el stock, consultar inventario, generar alertas por bajo stock y dar de baja productos cuando ya no estén en venta.

- Alta de productos: Permite registrar un nuevo producto en el inventario, capturando datos esenciales como el nombre del producto (cadena de caracteres de 50), su código (cadena de caracteres de 13), una descripción detallada (cadena de caracteres de 100), el precio de compra (número decimal) y el precio de venta (número decimal), la cantidad inicial disponible (número entero), el stock mínimo permitido (número entero) y la categoría a la que pertenece (cadena de caracteres de 30). El sistema asegura que el código de producto sea único, evitando duplicados en el inventario. Además, valida que los precios de compra y venta, así como la cantidad inicial y el stock mínimo, no sean valores negativos. En caso de que alguno de estos requisitos no se cumpla, el sistema no permite registrar el producto y emite un error.
- Modificación de stock: Permite ajustar el stock de los productos tras ventas, compras o devoluciones. Los empleados pueden modificar el stock ingresando el código del producto (cadena de caracteres de 13) y la cantidad a modificar (número entero). El sistema asegura que el producto exista en el inventario y que el stock no se vuelva negativo tras la modificación. Si la cantidad ingresada provoca que el stock sea negativo, el sistema emite un error y no realiza la actualización.
- Consulta del stock: Los empleados pueden consultar en cualquier momento el stock de un producto utilizando su código (cadena de caracteres de 13). El sistema devuelve el stock actual (número entero) si el producto existe en el inventario. Si el código no está registrado, se emite un error informando de la situación.
- Alerta por bajo stock: El sistema genera automáticamente una alerta cuando el stock de un producto cae por debajo de su nivel mínimo establecido. Esta alerta incluye información relevante, como el nombre del producto (cadena de caracteres de 50), su código (cadena de caracteres de 13) y el stock actual (número entero). Esta notificación permite a los empleados tomar medidas preventivas para evitar la falta de productos en el inventario.

- Dar de baja productos: Permite descatalogar productos que ya no estarán disponibles para la venta, cambiando su estado a "baja" en el sistema. Para dar de baja un producto, el empleado ingresa su código (cadena de caracteres de 13). El sistema solo dará de baja el producto si este existe en el inventario y no ha sido previamente descatalogado. Si el producto ya está en estado de baja o no existe, se emite un error. Una vez dado de baja, el producto no podrá ser seleccionado en futuras operaciones de venta o modificación de stock.

2. Subsistema de Gestión de Ventas (*Ignacio Rojas Valenzuela*)

Este subsistema se encarga de gestionar eficientemente las ventas de productos, asegurando que todas las transacciones se registren y procesen adecuadamente. También incluye funcionalidades para generar recibos o facturas, aplicar descuentos, consultar el historial de ventas y gestionar devoluciones, proporcionando una experiencia integral para los usuarios y manteniendo el control sobre el inventario y las finanzas del negocio.

- Registro de ventas: Permite registrar cada venta realizada capturando el DNI del cliente (cadena de caracteres de 10), junto con una lista de productos adquiridos. Para cada producto, se almacena su código (cadena de caracteres de 13), la cantidad comprada (número entero) y el precio de venta (número decimal). Además, se registran la fecha y hora de la venta (tipo Date y Time respectivamente), el código de venta (cadena de caracteres de 20) y el precio total de la transacción (número decimal). El sistema garantiza que los productos existan en el inventario, que haya suficiente stock y que estén activos; de lo contrario, no permite registrar la venta y emite un error.
- Generación de recibo o factura: Una vez que la venta ha sido registrada correctamente, el sistema genera un recibo o factura en función del código de venta (cadena de caracteres de 20). Este documento incluye detalles como la fecha y hora de la venta, los productos adquiridos, la cantidad de cada uno y el precio total (número decimal). Solo se genera el recibo o factura si la venta ha sido registrada sin errores, asegurando la validez de la transacción.
- Aplicación de descuentos o promociones: Permite aplicar descuentos o promociones en las ventas mediante un código de promoción (cadena de caracteres de 20). Este código identifica la reducción a aplicar sobre el precio total (número decimal), que se recalcula para reflejar el descuento. El sistema valida que el código de promoción sea válido antes de aplicar el descuento.
- Consulta del historial de ventas: Los empleados pueden consultar el historial de ventas mediante un rango de fechas especificado por la fecha de inicio y fecha de fin (ambas de tipo Date) y el código del producto (cadena de caracteres de 13). La consulta devuelve una lista de ventas realizadas, que incluye detalles de cada transacción como los productos vendidos, las cantidades (número entero), el precio de venta (número decimal), y la fecha y hora de venta. El sistema asegura que las fechas sean válidas y que el historial solo se muestre si existen ventas registradas para el producto consultado.
- Gestión de devoluciones: Facilita la gestión de devoluciones permitiendo al empleado ingresar el código de venta (cadena de caracteres de 20), el código del producto devuelto (cadena de caracteres de 13), la cantidad a devolver (número entero), y el motivo de la devolución

(cadena de caracteres de 100). Tras registrar la devolución, el sistema actualiza el stock del producto (número entero), asegurándose de que la cantidad devuelta no sea mayor que la comprada y que la devolución esté dentro del plazo permitido.

3. Subsistema de Gestión de Clientes (*Alejandro Coman Venceslá*)

El Subsistema de Gestión de Clientes se encarga de gestionar de manera eficiente las relaciones con los clientes, permitiendo su registro, modificación de datos, consulta de historial de compras y gestión de bajas. A continuación, se detallan las funcionalidades clave del subsistema:

- Registro de Clientes: Permite dar de alta a un nuevo cliente en el sistema, capturando los datos personales necesarios, tales como el nombre (cadena de caracteres), apellidos (cadena de caracteres), DNI (cadena de caracteres), correo electrónico (cadena de caracteres), teléfono (cadena de caracteres), dirección (cadena de caracteres) y la preferencia de contacto (booleano). Esto asegura que los clientes queden registrados de forma adecuada para futuras interacciones. Además, se verifica que no exista un cliente con el mismo DNI en el sistema.
- Consulta del Historial de Compras de un Cliente: Los empleados pueden consultar el historial de compras de un cliente, lo que permite visualizar todas las transacciones pasadas. Esta funcionalidad incluye detalles como el código de cliente (cadena de caracteres), la lista de productos adquiridos (array de productos), la cantidad comprada (número entero), la fecha (tipo fecha) y la hora de venta (tipo hora). Esta información es útil para realizar un seguimiento detallado de las compras realizadas y para la gestión de la relación con los clientes.
- Edición de Datos del Cliente: Facilita la modificación de los datos personales de los clientes, permitiendo a los empleados actualizar el nombre, apellidos, correo electrónico, teléfono, dirección y preferencia de contacto de un cliente registrado. El sistema asegura que el cliente existe antes de realizar cualquier modificación, y en caso de errores en los formatos de los datos (correo, teléfono, dirección), se muestra un mensaje de error.
- Notificación a Clientes: El sistema permite a los empleados enviar notificaciones a los clientes mediante su preferencia de contacto (correo electrónico o SMS). Se selecciona el cliente y se envía el mensaje correspondiente. El sistema verifica que el cliente exista y se asegura de enviar la notificación por el canal preferido del cliente.
- Baja de Clientes: Esta función permite dar de baja a un cliente en el sistema, actualizando su estado a "inactivo". El empleado introduce el código de cliente (DNI), y el sistema confirma la baja, impidiendo que el cliente realice más transacciones hasta que sea reactivado. En caso de que el cliente no exista o ya esté inactivo, se devuelve un mensaje de error.

4. Subsistema de Gestión de Proveedores (Pablo Anel Rancaño)

El Subsistema de Gestión de Proveedores se encarga de gestionar eficientemente las relaciones y transacciones con los proveedores, facilitando el control de proveedores, pedidos y su historial. A continuación, se describen las principales funcionalidades de este subsistema:

- Alta de proveedor: Permite registrar nuevos proveedores en el sistema, asegurando la correcta inserción de los datos para facilitar futuras transacciones y actualizaciones. Los datos que se capturan incluyen el nombre del proveedor (cadena de caracteres), el código CIF (cadena de caracteres), teléfono (cadena de caracteres), correo (cadena de caracteres) y dirección social (cadena de caracteres). Además, el sistema verifica que el CIF sea único y que el formato de teléfono, correo y dirección sean correctos antes de permitir el alta.
- Realización de pedidos: Los empleados pueden realizar pedidos a los proveedores, registrando todos los detalles del pedido como el código del proveedor (CIF), fecha de solicitud (tipo fecha), productos pedidos (estructura de producto que incluye código de producto, cantidad, y precio de compra). El sistema calcula automáticamente el importe total del pedido y mantiene un control del estado del mismo, que puede estar como "Pendiente", "Cancelado" o "Completado".
- Consulta del historial de compras a proveedores: Los empleados pueden consultar el historial de compras realizadas a los proveedores, permitiendo ver los pedidos, fechas, productos y cantidades. Esta función se basa en la fecha de transacción (tipo fecha), el producto adquirido (cadena de caracteres), y el total de la transacción (número decimal). La consulta se puede realizar sobre un rango de fechas, proporcionando reportes útiles para la gestión de la relación con los proveedores.
- Gestión de devoluciones de pedidos: El sistema permite gestionar la cancelación de pedidos. Para ello, el empleado puede seleccionar el código de pedido (cadena de caracteres) y especificar el motivo de la cancelación (cadena de caracteres). El sistema verifica que el pedido esté en estado "Pendiente" antes de permitir la cancelación, y actualiza automáticamente el estado a "Cancelado", registrando también la fecha de la cancelación y el motivo correspondiente.
- Marcar pedidos como completados: Una vez que el pedido ha sido recibido, el empleado puede marcarlo como "Completado" en el sistema. Esta función incluye el registro del código de pedido (cadena de caracteres) y el estado final del pedido. El sistema se asegura de que solo los pedidos en estado "Pendiente" puedan marcarse como completados, evitando errores en el proceso de gestión de pedidos.
- Baja de proveedores: Permite dar de baja a un proveedor del sistema cuando ya no se requiere su relación comercial. Esta función comprueba que el proveedor se encuentre activo antes de permitir la baja, registrando el código CIF del proveedor y cambiando su estado a "Inactivo". Además, el sistema emite una confirmación de la baja para mantener un control adecuado de los proveedores activos.

5. Subsistema de Informes y Análisis (Francisco José Ramos Moya)

El subsistema de Informes y Análisis se encarga de proporcionar reportes y análisis detallados sobre las ventas, pedidos y beneficios, facilitando la toma de decisiones en base a datos precisos. A continuación, se describen las principales funcionalidades del subsistema:

- Generación de informes de ventas por intervalo: Este módulo permite generar un informe que resume las transacciones de ventas realizadas en un período específico. El sistema recibe como entrada las fechas de inicio y fin (tipo Date) y devuelve un informe detallado con la lista de productos vendidos, la cantidad (número entero), Nombre de producto (número decimal) y el código de producto (cadena de 13 caracteres) de un intervalo especificado.
- Generación de análisis de productos más vendidos: Permite generar un análisis de los productos más vendidos durante un intervalo de tiempo determinado. Se especifica un límite de productos a mostrar (número entero), y el sistema devuelve un análisis con el nombre de los productos (cadena de caracteres), su código (cadena de 13 caracteres) y la cantidad vendida (número entero), destacando los productos más populares. Además, garantiza que se muestren solo productos que hayan tenido ventas en el período seleccionado.
- Análisis de beneficios por producto: El sistema permite analizar el beneficio obtenido por un producto específico durante un intervalo de tiempo determinado, utilizando su código (cadena de caracteres) como identificador. Se considera el precio de venta y el precio de compra de los productos vendidos en un intervalo, proporcionando un análisis detallado de Nombre del producto (cadena de caracteres), su código(cadena 13 caracteres) y los ingresos totales(número entero)
- Generación de informes de pedidos: El subsistema también permite generar informes detallados de pedidos en un intervalo de fechas, registrando cada pedido y los productos adquiridos. Para cada pedido se registra el nombre(cadena caracteres) y código del producto (cadena de caracteres), la cantidad comprada (número entero) y el precio de compra (número decimal). Este informe ayuda a mantener un control sobre los gastos realizados en la compra de productos.
- Análisis de ventas por familia de productos: Permite generar un análisis de las ventas agrupadas por familia de productos. El sistema recibe la categoría del producto (cadena de caracteres) junto con un intervalo de fechas, y devuelve un análisis con la cantidad vendida (número entero), la facturación total (números decimales) relacionados con dicha familia. Esto facilita la evaluación del rendimiento de una categoría de productos específica dentro de un período. En este caso, se relaciona con el “análisis de beneficios por producto, ya que aquí se realizará la misma tarea, pero tantas veces como productos haya en una familia o categoría.

Restricciones y validaciones:

- Verificación de fechas: En todos los casos, el sistema valida que la fecha de inicio no sea posterior a la fecha de fin. Si ocurre lo contrario, se cancela el proceso y se notifica al usuario con un mensaje de error.
- Datos inexistentes: Si no se encuentran ventas o pedidos en el intervalo especificado, el sistema lo indicará explícitamente en el informe generado.

Este subsistema garantiza la creación de informes y análisis eficientes, contribuyendo a una gestión precisa de las ventas y pedidos, lo que permite a los empleados y gerentes tomar decisiones basadas en datos concretos.

Requisitos funcionales:

1. Subsistema de Gestión de Inventario (José Antonio Fernández Aranda)

RF1.1: Dar de alta un producto			
Número de referencia	RF1.1		
Descripción del requisito	Se da de alta un nuevo producto.		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Solicitar la inserción de un nuevo producto	
	Requisito de datos	<p>Número de referencia</p> <p>Composición</p>	<p>RDE1.1</p> <ul style="list-style-type: none"> • Nombre: Cadena de caracteres (50). • Código Producto: Cadena de caracteres (13). • Descripción: Cadena de caracteres (100). • Precio Compra: Número decimal. • Precio Venta: Número decimal. • Cantidad Inicial: Número entero. • Stock Mínimo: Número entero. • Categoría: Cadena de caracteres (30).
Datos internos	Requisito de datos	Número de referencia	RDW1.1
		Composición	<ul style="list-style-type: none"> • Los mismos que RDE1.1 • Estado: Booleano. (Inicializado a 1).
	Requisito de datos	<p>Número de referencia</p> <p>Descripción</p>	<p>RDR1.1</p> <ul style="list-style-type: none"> • Código de Producto. • Precio Compra • Precio Venta • Cantidad Inicial

Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Confirmación de la inserción.	
	Requisito de datos	Número de referencia	Ninguno
		Composición	Ninguno
Restricciones semánticas		Número de referencia	RS1.1.1
		Descripción	Un código de producto corresponde a un único producto.
		Requisito funcional relacionado	RF1.1
		Requisito/s de datos relacionados	RDE1.1 + RDR1.1
		Número de referencia	RS1.1.2
		Descripción	No negatividad en el precio de compra.
		Requisito funcional relacionado	RF1.1
		Requisito/s de datos relacionados	RDE1.1
		Número de referencia	RS 1.1.3
		Descripción	No negatividad en el precio de venta.
		Requisito funcional relacionado	RF1.1
		Requisito/s de datos relacionados	RDE1.1
		Número de referencia	RS 1.1.4
		Descripción	No negatividad en la cantidad inicial.
		Requisito funcional relacionado	RF1.1
		Requisito/s de datos relacionados	RDE1.1 + RDW1.1

		Número de referencia	RS 1.1.5
		Descripción	No negatividad en el stock mínimo.
		Requisito funcional relacionado	RF1.1
		Requisito/s de datos relacionados	RDE1.1
		Número de referencia	RS 1.1.6
		Descripción	Se inicializa el estado del producto a 1.
		Requisito funcional relacionado	RF1.1
		Requisito/s de datos relacionados	RDW1.1

RF1.2: Modificar el stock de un producto			
Número de referencia	RF1.2		
Descripción del requisito	Modificación del stock de un producto existente		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Solicitar la modificación del stock.	
	Requisito de datos	Número de referencia	RDE1.2
Datos internos		Composición	<ul style="list-style-type: none"> • Código Producto: Cadena de caracteres (13) • Cantidad a Modificar: Número entero (puede ser positivo o negativo).
Requisito de datos	Número de referencia	RDW1.2	
	Composición	<ul style="list-style-type: none"> • Código Producto. Cadena de caracteres (13) • Stock Nuevo: Número entero (stock actualizado del producto tras la operación). 	
Requisito de datos	Número de referencia	RDR1.2	
	Composición	<ul style="list-style-type: none"> • Código de Producto. Cadena de caracteres (13) • Stock Actual: Número entero. 	
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Imprimir nuevo stock	
	Requisito de datos	Número de referencia	RDS1.2
Restricciones semánticas		Composición	<ul style="list-style-type: none"> • Nuevo Stock: Número entero.
	Número de referencia	RS1.2.1	
	Descripción	El código de producto debe existir en el inventario.	
	Requisito funcional relacionado	RF1.1 (Dar de alta un producto).	

	Requisito/s de datos relacionados	RDE1.2 + RDW1.2 + RDR1.2
	Número de referencia	RS1.2.2
	Descripción	El stock final será igual al stock actual + (cantidad a modificar).
	Requisito funcional relacionado	RF1.2
	Requisito/s de datos relacionados	RDE1.2 + RDW1.2 + RDR1.2
	Número de referencia	RS1.2.3
	Descripción	El stock del producto no puede quedar en un valor negativo.
	Requisito funcional relacionado	RF1.2
	Requisito/s de datos relacionados	RDE1.2 + RDW1.2 + RDR1.2

RF1.3: Consultar el stock de un producto			
Número de referencia	RF1.3		
Descripción del requisito	Solicitar información sobre el stock de un producto específico		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Solicitar la consulta del stock de un producto	
	Requisito de datos	Número de referencia	RDE1.3
Datos internos		Composición	<ul style="list-style-type: none"> • Código Producto: Cadena de caracteres (13).
Requisito de datos	Número de referencia	RDR1.3	
	Composición	<ul style="list-style-type: none"> • Código Producto: Cadena de caracteres (13). • Stock Actual: Número entero. 	
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Visualizar el stock del producto	
	Requisito de datos	Número de referencia	RDS1.3
Restricciones semánticas		Composición	<ul style="list-style-type: none"> • Código Producto: Cadena de caracteres (13). • Stock Actual: Número entero.
	Número de referencia	RS1.3	
	Descripción	El código de producto debe existir en el inventario para poder consultar su stock.	
	Requisito funcional relacionado	RF1.1 (Dar de alta un producto)	
Requisito/s de datos relacionados		RDE1.3 + RDR1.3	

RF1.4: Consultar Alerta Producto			
Número de referencia	RF1.4		
Descripción del requisito	Consultar las alertas asociadas al stock de un producto.		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Comprueba, para todos los productos, que el stock de un producto esté por debajo del mínimo.	
	Requisito de datos	Número de referencia	Ninguno
		Composición	Ninguno
Datos internos	Requisito de datos	Número de referencia	RDR1.4
		Composición	<ul style="list-style-type: none"> • Stock Actual: Número entero. • Código Producto: Cadena de caracteres (13) • Código de Alerta: Cadena de caracteres (10) • Fecha de Alerta: tipo Date. • Hora de Alerta: tipo Time.
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Notificar mediante alerta: el código del producto con stock bajo y su cantidad de stock.	
	Requisito de datos	Número de referencia	RDS1.4
		Composición	<ul style="list-style-type: none"> • Mensaje de Bajo Stock. Cadena de caracteres (100). • Nombre. Cadena de caracteres (50). • Código Producto: Cadena de caracteres (13). • Stock: Número entero.
Restricciones semánticas		Número de referencia	RS1.4
		Descripción	El stock actual debe ser menor o igual al stock mínimo definido para enviar la alerta.
		Requisito funcional relacionado	RF1.4
		Requisito/s de datos relacionados	RDR1.4

RF1.5: Dar de baja un producto (Descatalogar)			
Número de referencia	RF1.5		
Descripción del requisito	Se lleva a cabo la baja de un producto del inventario		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Solicitar la baja de un producto	
	Requisito de datos	Número de referencia	RDE1.5
Datos internos		Composición	<ul style="list-style-type: none"> • Código Producto: Cadena de caracteres (13).
Requisito de datos	Número de referencia	RDW1.5	
	Composición	<ul style="list-style-type: none"> • Código Producto: Cadena de caracteres (13). • Estado. Booleano (se cambia a 0). 	
Requisito de datos	Número de referencia	RDR1.5	
	Composición	<ul style="list-style-type: none"> • Código Producto: Cadena de caracteres (13). • Estado. Booleano. 	
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Confirmación de la baja del producto	
	Requisito de datos	Número de referencia	Ninguno
Restricciones semánticas		Composición	Ninguno
	Número de referencia	RS1.5.1	
	Descripción	El producto se dará de baja si el código proporcionado existe y no está dado de baja.	
	Requisito funcional relacionado	RF1.1 (Dar de alta un producto).	

	Requisito/s de datos relacionados	RDE1.5 + RDR1.5
	Número de referencia	RS 1.5.2
	Descripción	El producto se dará de baja sólamente cuando el producto no esté dado ya de baja
	Requisito funcional relacionado	RF1.5
	Requisito/s de datos relacionados	RDE1.5 + RDR1.5

2. Subsistema de Gestión de Ventas (Ignacio Rojas Valenzuela)

RF2.1: Registrar una venta			
Número de referencia	RF2.1		
Descripción del requisito	Registrar una transacción de productos vendidos.		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Registrar una venta	
	Requisito de datos	Número de referencia	RDE2.1
Datos internos	Requisito de datos	Composición	<ul style="list-style-type: none"> • DNI Cliente: Cadena de caracteres (10). • Códigos de Productos: Lista de Cadena de caracteres (13). • Cantidades: Lista de Número enteros.
		Número de referencia	RDW2.1
	Requisito de datos	Composición	<ul style="list-style-type: none"> • Los mismos datos que RDE2.1. • Código de Venta. Cadena de caracteres (20). • Fecha de Venta: Tipo Date. • Hora de Venta: Tipo Time. • Fecha de devolución. Tipo Date. • Precio Total: Número decimal.
		Número de referencia	RDR2.1
Salida	Requisito de datos	Composición	<ul style="list-style-type: none"> • Igual que RDW2.1.
		Número de referencia	RDS2.1
	Requisito de datos	Composición	<ul style="list-style-type: none"> • Código de venta. Cadena de caracteres (20).
		Número de referencia	

Restricciones semánticas		Número de referencia	RS2.1.1
		Descripción	El producto debe existir en el inventario para registrar la venta. Si no existe, se devuelve un error.
		Requisito funcional relacionado	RF2.1
		Requisito/s de datos relacionados	RDE2.1 + RDW2.1 + RDR2.1
		Número de referencia	RS2.1.2
		Descripción	Debe haber suficiente stock para cubrir la cantidad solicitada en la venta.
		Requisito funcional relacionado	RF1.3 (Consultar stock de un producto).
		Requisito/s de datos relacionados	RDE2.1 + RDR2.1
		Número de referencia	RS2.1.3
		Descripción	Cada producto debe estar en estado activo.
		Requisito funcional relacionado	RF1.1 (Dar de alta un producto).
		Requisito/s de datos relacionados	RDE2.1 + RDW2.1 + RDR2.1
		Número de referencia	RS2.1.4
		Descripción	La fecha de devolución será la misma fecha de compra a la que se le añadirán 30 días.
		Requisito funcional relacionado	RF2.5 (Gestionar una devolución)
		Requisito/s de datos relacionados	RDW2.1 + RDR2.1
		Número de referencia	RS2.1.5
		Descripción	El importe total de la venta será igual a la suma de la cantidad de cada producto vendida multiplicada por su respectivo precio de venta.
		Requisito funcional relacionado	RF2.5 (Gestionar una devolución)
		Requisito/s de datos relacionados	RDW2.1 + RDR2.1

		Número de referencia	RS2.1.6
		Descripción	El cliente debe existir.
		Requisito funcional relacionado	RF3.1 (Dar de alta un cliente).
		Requisito/s de datos relacionados	RDW2.1 + RDR2.1

RF2.2: Generar recibo o factura			
Número de referencia	RF2.2		
Descripción del requisito	Emitir un comprobante de pago o factura.		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Solicitar recibo o factura.	
	Requisito de datos	Número de referencia	RDE2.2
Datos internos		Composición	<ul style="list-style-type: none"> • Código de Venta: Cadena de caracteres (20).
Requisito de datos	Número de referencia	RDR2.2	
	Composición	<ul style="list-style-type: none"> • Código de Venta: Cadena de caracteres (20). • Códigos de Productos: Lista de Cadena de caracteres (13). • Cantidad: Lista de Números enteros. • Precios de Venta: Lista de Números decimales. • Fecha de Venta: Tipo Date. • Hora de Venta: Tipo Time. • Precio Total: Número decimal. 	
Salida	Agente externo	Cliente	
	Acción iniciada por el agente externo	Recibir recibo o factura	
	Requisito de datos	Número de referencia	RDS2.2
Restricciones semánticas		Composición	<ul style="list-style-type: none"> • Los mismo que RDR2.2
	Número de referencia	RS2.2.1	
	Descripción	El recibo o factura debe generarse solo si la venta está registrada correctamente (el código de venta es válido). Si no se devuelve un error	
	Restricciones semánticas		Requisito funcional relacionado

	Requisito/s de datos relacionados	RDE2.2 + RDR2.2 + RDS2.2
	Número de referencia	RS2.2.2
	Descripción	El precio total se calcula como la sumatoria del precio de venta por unidad multiplicado por la cantidad de producto, de todos los productos.
	Requisito funcional relacionado	RF2.1 (Registrar una venta).
	Requisito/s de datos relacionados	RDE2.2 + RDR2.2

RF2.3: Aplicar descuentos o promociones			
Número de referencia	RF2.3		
Descripción del requisito	Implementar reducciones o promociones en los precios.		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Aplicar descuento o promoción	
	Requisito de datos	Número de referencia	RDE2.3
Datos internos		Composición	<ul style="list-style-type: none"> • Código de promoción: Cadena de caracteres (20) • Código Venta: Cadena de caracteres (20).
Requisito de datos	Número de referencia	RDW2.3	
	Composición	<ul style="list-style-type: none"> • Código Venta: Cadena de caracteres (20). • Código de promoción: Cadena de caracteres (20) • Precio Total: Número decimal. • Descuento aplicado: Número decimal 	
Requisito de datos	Número de referencia	RDR2.3	
	Composición	<ul style="list-style-type: none"> • Código Venta: Cadena de caracteres (20). • Código de promoción: Cadena de caracteres (20) • Precio Total: Número decimal. • Descuento aplicado: Número decimal 	
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Asignación del nuevo precio final tras el descuento aplicado	
	Requisito de datos	Número de referencia	RDS2.3
Restricciones		Composición	Precio final con el descuento aplicado
	Número de referencia	RS2.3.1	

semánticas		Descripción	El código de promoción debe ser válido.
		Requisito funcional relacionado	RF2.3
		Requisito/s de datos relacionados	RDE2.3 + RDR2.3
		Número de referencia	RS2.3.2
		Descripción	El código de venta debe de ser válido
		Requisito funcional relacionado	RF2.3
		Requisito/s de datos relacionados	RDE2.3 + RDR2.3
		Número de referencia	RS2.3.3
		Descripción	El precio total será igual al importe de la venta multiplicado por el descuento aplicado.
		Requisito funcional relacionado	RF2.3
		Requisito/s de datos relacionados	RDE2.3 + RDR2.3

RF2.4: Consultar historial de ventas de productos.			
Número de referencia	RF2.4		
Descripción del requisito	Revisar el registro de transacciones de ventas		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Solicitar historial de ventas	
	Requisito de datos	Número de referencia RDE2.4 Composición <ul style="list-style-type: none"> • Fecha de inicio de consulta: Date • Fecha final de consulta: Date 	
Datos internos	Requisito de datos	Número de referencia RDR2.4 Composición <ul style="list-style-type: none"> • Códigos de Ventas: Lista de Cadena de caracteres (20) • Códigos de Productos: Lista de Cadena de caracteres (13). • Cantidades: Lista de Números enteros. • Precio Total: Número decimal. • Fecha de Venta: Tipo Date. • Hora de Venta: Tipo Time. 	
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Mostrar historial de ventas	
	Requisito de datos	Número de referencia RDS2.4 Composición <ul style="list-style-type: none"> • Mensaje de confirmación de consulta. Cadena de caracteres (100). • RDR2.4 	
Restricciones semánticas		Número de referencia RS2.4.1	
		Descripción El código del producto debe de existir. En caso contrario, se cancela la consulta.	
		Requisito funcional relacionado RF1.1 (Registrar un producto).	

	Requisito/s de datos relacionados	RDE2.4 + RDR2.4
	Número de referencia	RS2.4.2
	Descripción	El rango de fechas debe ser válido (fecha de inicio anterior a la fecha de fin).
	Requisito funcional relacionado	RF2.4
	Requisito/s de datos relacionados	RDE2.4
	Número de referencia	RS2.4.3
	Descripción	Las ventas deben estar en el rango temporal de las fechas.
	Requisito funcional relacionado	RF2.4
	Requisito/s de datos relacionados	RDE2.4 + RDR2.4

RF2.5: Gestionar una devolución			
Número de referencia	RF2.5		
Descripción del requisito	Procesar la devolución de una venta.		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Gestionar la devolución de un producto	
	Requisito de datos	Número de referencia	RDE2.5
Datos internos	Requisito de datos	Composición	<ul style="list-style-type: none"> • Código de venta. Cadena de caracteres (20). • Códigos de productos. Lista de Cadena de caracteres (13). • Cantidades a devolver: Lista de Números enteros. • Motivo de la devolución: Cadena de caracteres (100).
		Número de referencia	RDW2.5
	Requisito de datos	Composición	<ul style="list-style-type: none"> • Código de venta. Cadena de caracteres (20). • Códigos de productos. Cadena de caracteres (13). • Cantidades a devolver. Lista de números enteros. • Stock Nuevo. Número entero.
		Número de referencia	RDR2.5
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Confirmación de devolución	
	Requisito de datos	Número de referencia	Ninguno

		Composición	Ninguno
Restricciones semánticas		Número de referencia	RS2.5.1
		Descripción	El código de venta debe ser válido (debe existir).
		Requisito funcional relacionado	RF2.1 (Registrar una venta).
		Requisito/s de datos relacionados	RDE2.5 + RDR2.5
		Número de referencia	RS2.5.2
		Descripción	La cantidad devuelta no puede ser mayor que la cantidad comprada.
		Requisito funcional relacionado	RF2.5
		Requisito/s de datos relacionados	RDE2.5 + RDW2.5 + RDR2.5
		Número de referencia	RS2.5.3
		Descripción	El plazo para gestionar una devolución debe estar dentro del período permitido.
		Requisito funcional relacionado	RF2.5
		Requisito/s de datos relacionados	RDE2.5 + RDR2.5
		Número de referencia	RS2.5.4
		Descripción	El stock de los productos involucrados en la venta deben actualizarse tras la devolución. Será el stock actual sumado a la cantidad a devolver.
		Requisito funcional relacionado	RF1.2 (Modificar el stock de un producto).
		Requisito/s de datos relacionados	RDE2.5 + RDW2.5 + RDR2.5
		Número de referencia	2.5.5
		Descripción	La cantidad a devolver debe ser positiva.
		Requisito funcional relacionado	R.F2.5

		Requisito/s de datos relacionados	RDE2.5 + RDR2.5
--	--	--	-----------------

3. Subsistema de Gestión de Clientes (Alejandro Coman Venceslá).

RF3.1: Dar de alta un cliente			
Número de referencia	RF3.1		
Descripción del requisito	Dar de alta un nuevo cliente en el sistema.		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Introducir los datos personales del cliente.	
	Requisito de datos	Número de referencia Composición	RDE3.1 <ul style="list-style-type: none"> • Nombre. Cadena de caracteres (100). • Código Cliente (DNI). Cadena de caracteres (10). • Correo electrónico. Cadena de caracteres (50). • Teléfono. Cadena de caracteres (20) • Dirección. Cadena de caracteres (100). • Preferencia de Contacto. Booleano.
Datos internos	Requisito de datos	Número de referencia	RDW3.1
		Composición	<ul style="list-style-type: none"> • Igual que RDE3.1 • Estado. Booleano (inicializado a 1).
	Requisito de datos	Número de referencia	RDR3.1
		Composición	<ul style="list-style-type: none"> • Igual que RDE3.1.
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Confirmación de cliente registrado.	
	Requisito de datos	Número de referencia Composición	Ninguno Ninguno

Restricciones semánticas		Número de referencia	RS3.1.1
		Descripción	El código de cliente debe ser único.
		Requisito funcional relacionado	RF3.1.
		Requisito/s de datos relacionados	RDE3.1 + RDR3.1
		Número de referencia	RDS3.1.2
		Descripción	El correo debe tener un formato válido.
		Requisito funcional relacionado	RF3.1
		Requisito/s de datos relacionados	RDE3.1
		Número de referencia	RDS3.1.3
		Descripción	El teléfono debe tener un formato válido.
		Requisito funcional relacionado	RF3.1
		Requisito/s de datos relacionados	RDE3.1
		Número de referencia	RDS3.1.4
		Descripción	La dirección debe tener un formato válido.
		Requisito funcional relacionado	RF3.1
		Requisito/s de datos relacionados	RDE3.1
		Número de referencia	RDS3.1.5
		Descripción	Se inicializará el estado del cliente a 1 (significando que está activo)
		Requisito funcional relacionado	RF3.1
		Requisito/s de datos relacionados	RDW3.1

RF3.2: Consultar historial de compras de un cliente			
Número de referencia	RF3.2		
Descripción del requisito	Permite consultar el historial de compras de un cliente.		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Introducir o seleccionar el identificador del cliente.	
	Requisito de datos	Número de referencia	RDE3.2
Datos internos		Composición	<ul style="list-style-type: none"> • Código Cliente. Cadena de caracteres (10).
Requisito de datos	Número de referencia	RDR3.2	
	Composición	<ul style="list-style-type: none"> • Códigos de Venta. Lista de cadena de caracteres (20). • Códigos de Producto: Lista de cadena de caracteres (13). • Códigos de DetalleVenta: Lista de cadena de caracteres (13). • Cantidad: Lista de Números enteros. • Fecha de Venta: Tipo Date. • Hora de Venta: Tipo Time. 	
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Visualizar historial de compras del cliente.	
	Requisito de datos	Número de referencia	RDS3.2
Restricciones semánticas		Composición	<ul style="list-style-type: none"> • Código de cliente. Cadena de caracteres (10). • RDR3.2
	Número de referencia	RS3.2: El cliente seleccionado debe existir.	
	Descripción	El código de cliente debe de ser válido	
	Requisito funcional relacionado	RF3.1 (Dar de alta un cliente).	
	Requisito/s de datos relacionados	RDE3.2 + RDR3.2	

RF3.3: Editar datos del cliente			
Número de referencia	RF3.3		
Descripción del requisito	Modifica los datos personales de un cliente.		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Seleccionar cliente y modificar sus datos.	
	Requisito de datos	Número de referencia	RDE3.3
Datos internos		Composición	<ul style="list-style-type: none"> • Igual que RDE3.1.
Requisito de datos	Número de referencia	RDW3.3	
	Composición	<ul style="list-style-type: none"> • Igual que RDE3.1. 	
Requisito de datos	Número de referencia	RDR3.3	
	Composición	<ul style="list-style-type: none"> • Igual que RDE3.1. 	
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Confirmación del descuento aplicado.	
	Requisito de datos	Número de referencia	Ninguno
Restricciones semánticas		Composición	Ninguno
	Número de referencia	RS3.3.1	
	Descripción	El código de cliente debe ser único.	
	Requisito funcional relacionado	RF3.1 (Dar de alta un cliente).	

	Requisito/s de datos relacionados	RDE3.3 + RDR3.3
	Número de referencia	RDS3.3.2
	Descripción	El correo debe tener un formato válido.
	Requisito funcional relacionado	RF3.3
	Requisito/s de datos relacionados	RDE3.3
	Número de referencia	RDS3.3.3
	Descripción	El teléfono debe tener un formato válido.
	Requisito funcional relacionado	RF3.3
	Requisito/s de datos relacionados	RDE3.3
	Número de referencia	RDS3.3.4
	Descripción	La dirección debe tener un formato válido.
	Requisito funcional relacionado	RF3.3
	Requisito/s de datos relacionados	RDE3.3

RF3.4: Enviar una notificación a un cliente			
Número de referencia	RF3.4		
Descripción del requisito	Enviar notificaciones (email o SMS) al cliente, según la preferencia de contacto (0 para correo y 1 para SMS).		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Seleccionar cliente y el mensaje a enviar	
	Requisito de datos	Número de referencia	RDE3.4
Datos internos		Composición	<ul style="list-style-type: none"> Código Cliente (DNI). Cadena de caracteres (10). Mensaje a enviar. Cadena de caracteres (300).
Requisito de datos	Número de referencia	RDW3.4	
	Composición	<ul style="list-style-type: none"> Código Cliente (DNI). Cadena de caracteres (10). Código de notificación. Cadena de caracteres (10). Mensaje a enviar. Cadena de caracteres (300). Fecha del mensaje. Date Hora de mensaje. Hora 	
Requisito de datos	Número de referencia	RDR3.4	
	Composición	<ul style="list-style-type: none"> Código Cliente (DNI). Cadena de caracteres (10). Correo. Cadena de caracteres (50). Teléfono. Cadena de caracteres (20). Preferencia de contacto. Booleano. 	
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Confirmación de envío de notificación.	
	Requisito de datos	Número de referencia	Ninguno
		Composición	Ninguno
		Número de referencia	RS3.4.1. El cliente seleccionado debe existir.

Restricciones semánticas	Descripción	El código de cliente debe existir.
	Requisito funcional relacionado	RF3.1 (Dar de alta un cliente)
	Requisito/s de datos relacionados	RDE3.4 + RDR3.4
	Número de referencia	RS3.4.2. El Código de notificación tiene el formato “N<número>”
	Descripción	Comienza por N y los 9 dígitos restantes son una secuencia que empieza con 1, rellenando con 0.
	Requisito funcional relacionado	RF3.4
	Requisito/s de datos relacionados	RDW3.4

RF3.5: Dar de baja un cliente			
Número de referencia	RF3.5		
Descripción del requisito	Dar de baja un cliente en el sistema.		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Introduce el código del cliente que quiere dar de baja.	
	Requisito de datos	Número de referencia	RDE3.5
Datos internos		Composición	<ul style="list-style-type: none"> • Código Cliente (DNI). Cadena de caracteres (10).
Requisito de datos	Número de referencia	RDW3.5	
	Composición	<ul style="list-style-type: none"> • Código Cliente (DNI). Cadena de caracteres (10) • Estado. Booleano. 	
Requisito de datos	Número de referencia	RDR3.5	
	Composición	<ul style="list-style-type: none"> • Código Cliente (DNI). Cadena de caracteres (10). • Estado. Booleano. 	
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Confirmación de la baja del cliente.	
	Requisito de datos	Número de referencia	Ninguno
Restricciones semánticas		Composición	Ninguno
	Número de referencia	RS3.5.1.	
	Descripción	El cliente debe existir.	
	Requisito funcional relacionado	RF3.1 (Dar de alta el cliente)	

	Requisito/s de datos relacionados	RDE3.5 + RDR3.5
	Número de referencia	RS3.5.1: El cliente seleccionado debe existir.
	Descripción	El cliente debe de estar dado de alta para poder darlo de baja.
	Requisito funcional relacionado	RF3.1 (Dar de alta a un cliente).
	Requisito/s de datos relacionados	RDE3.5 + RDR3.5
	Número de referencia	RDS3.5.2
	Descripción	Se actualizará el valor del estado del cliente a 0.
	Requisito funcional relacionado	RF3.5
	Requisito/s de datos relacionados	RDW3.5

4. Subsistema de Gestión de Proveedores (Pablo Anel Rancaño)

RF4.1: Dar de alta un proveedor			
Número de referencia	RF4.1		
Descripción del requisito	El empleado registra un nuevo proveedor en el sistema.		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Solicitar la inserción de un nuevo proveedor.	
	Requisito de datos	Número de referencia	RDE4.1
Datos internos	Requisito de datos	Composición	<ul style="list-style-type: none"> • Nombre proveedor: Cadena de caracteres (40). • Código de proveedor (CIF): Cadena de caracteres (10). • Teléfono: Cadena de caracteres (15). • Correo: Cadena de caracteres (50). • Dirección social: Cadena de caracteres (10).
		Número de referencia	RDW4.1
	Requisito de datos	Composición	<ul style="list-style-type: none"> • Los mismos que RDE4.1 • Estado: Booleano.
		Número de referencia	RDR4.1
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Confirmación del resultado (inserción del proveedor).	
	Requisito de datos	Número de referencia	Ninguno
		Composición	Ninguno

Restricciones semánticas		Número de referencia	RDS4.1.1
		Descripción	El CIF de un proveedor debe ser único, en caso contrario no se dará de alta, y se devolverá un mensaje de error.
		Requisito funcional relacionado	RF4.1
		Requisito/s de datos relacionados	RDE4.1 + RDR4.1
		Número de referencia	RDS4.1.2
		Descripción	El correo debe tener un formato válido.
		Requisito funcional relacionado	RF4.1
		Requisito/s de datos relacionados	RDE4.1
		Número de referencia	RDS4.1.3
		Descripción	El teléfono debe tener un formato válido
		Requisito funcional relacionado	RF4.1
		Requisito/s de datos relacionados	RDE4.1
		Número de referencia	RDS4.1.4
		Descripción	La dirección debe tener un formato válido.
		Requisito funcional relacionado	RF4.1
		Requisito/s de datos relacionados	RDE4.1

RF4.2: Realizar un pedido a un proveedor			
Número de referencia	RF4.2		
Descripción del requisito	El empleado realiza un pedido a los proveedores.		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Solicitar la creación de un nuevo pedido a un proveedor y actualizar su lista de pedidos.	
	Requisito de datos	Número de referencia Composición	RDE4.2 <ul style="list-style-type: none"> • Código de proveedor (CIF): cadena de caracteres (20). • Fecha de solicitud de pedido: Date • Código de Producto: Cadena de caracteres (13). • Cantidad: Número entero.
Datos internos	Requisito de datos	Número de referencia	RDW4.2
		Composición	<ul style="list-style-type: none"> • RDE4.2 • Código de pedido: cadena de caracteres (20). • Importe del pedido: número decimal. • Estado del pedido: cadena de caracteres (15) (Posibles valores: "Pendiente", "Cancelado", "Completado").
	Requisito de datos	Número de referencia Composición	RDR4.2 <ul style="list-style-type: none"> • RDW4.2
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Confirmación de que el pedido fue realizado correctamente.	
	Requisito de datos	Número de referencia Composición	Ninguno Ninguno

Restricciones semánticas		Número de referencia	RS4.2.1
		Descripción	El estado del pedido tendrá solamente los valores “Pendiente”, “Cancelado” o “Completado”.
		Requisito funcional relacionado	RF4.2
		Requisito/s de datos relacionados	RDW4.2
		Número de referencia	RS4.2.2
		Descripción	Un código de pedido debe ser único.
		Requisito funcional relacionado	RF4.2
		Requisito/s de datos relacionados	RDE4.2 + RDR4.2
		Número de referencia	RS4.2.3
		Descripción	El CIF del proveedor debe existir en el sistema (y estar dado de alta).
		Requisito funcional relacionado	RF4.1 (Dar de alta un proveedor)
		Requisito/s de datos relacionados	RDE4.2 + RDR4.2
		Número de referencia	RS4.2.4
		Descripción	El estado del pedido se inicializará a “Pendiente”.
		Requisito funcional relacionado	RF4.2
		Requisito/s de datos relacionados	RDW4.2
		Número de referencia	RS4.2.5
		Descripción	El importe total del pedido es igual a la cantidad pedida del producto por su precio de compra
		Requisito funcional relacionado	RF4.2
		Requisito/s de datos relacionados	RDE4.2 + RDR4.2

RF4.3:			
Número de referencia	RF4.3		
Descripción del requisito	El empleado consulta el historial de compras realizadas a los proveedores; pedidos, fechas, productos y cantidades.		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Solicitar la consulta del historial de pedidos realizados a un proveedor.	
	Requisito de datos	Número de referencia	RDE4.3
Datos internos	Requisito de datos	Composición	<ul style="list-style-type: none"> • Código de proveedor (CIF): Cadena de caracteres (10) • Fechas de inicio de consulta. Date. • Fechas de cierre de consulta. Date.
		Número de referencia	RDR4.3
Salida	Requisito de datos	Composición	<ul style="list-style-type: none"> • Código de proveedor (CIF): Cadena de caracteres (10) • Códigos de Pedido. Lista de cadenas de caracteres (20). • Cantidad: Números enteros. • Fecha de solicitud: Tipo Date. • Importe del pedido: Número decimal.
		Agente externo	Empleado
		Acción iniciada por el agente externo	Devuelve la lista de pedidos
Restricciones semánticas		Número de referencia	RDS4.3
		Descripción	El código de proveedor debe existir en el sistema
		Requisito funcional relacionado	RF4.1

	Requisito/s de datos relacionados	RDE4.3 + RDR4.3
	Número de referencia	RS4.3.2
	Descripción	La fecha de fin debe de ser posterior a la fecha de inicio.
	Requisito funcional relacionado	RF4.3
	Requisito/s de datos relacionados	RDE4.3
	Número de referencia	RS4.3.3
	Descripción	Los pedidos deben estar en el rango temporal de las fechas.
	Requisito funcional relacionado	RF4.3
	Requisito/s de datos relacionados	RDE4.3 + RDR4.3
	Número de referencia	RS4.3.4
	Descripción	El importe total del pedido es igual a la cantidad pedida del producto por su precio de compra
	Requisito funcional relacionado	RF4.3
	Requisito/s de datos relacionados	RDE4.3 + RDR4.3

RF4.4: Cancelar un pedido a un proveedor			
Número de referencia	RF4.4		
Descripción del requisito	El empleado puede cancelar un pedido que se haya realizado previamente a un proveedor.		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Solicitar la cancelación de un pedido existente.	
	Requisito de datos	Número de referencia	RDE4.4
		Composición	<ul style="list-style-type: none"> Código del pedido: Cadena de caracteres (10). Motivo de la cancelación: Cadena de caracteres (100).
Datos internos	Requisito de datos	Número de referencia	RDW4.4
		Composición	<ul style="list-style-type: none"> Código del pedido: cadena de caracteres (10) Estado del pedido: Cadena de Caracteres (15) Fecha de cancelación del pedido: Date. Motivo de la cancelación. Cadena de caracteres (100).
	Requisito de datos	Número de referencia	RDR4.4
		Composición	<ul style="list-style-type: none"> Código del pedido: Cadena de caracteres (10). Estado del pedido: Cadena de Caracteres (15).
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Mensaje de confirmación de cancelación del pedido.	
	Requisito de datos	Número de referencia	Ninguno
		Composición	Ninguno
Restricciones		Número de referencia	RS4.4.1

semánticas		Descripción	El código del pedido debe estar registrado en el sistema. Debe haberse realizado previamente. En caso contrario cancelar el proceso.
		Requisito funcional relacionado	RF4.2 (Realizar un pedido a un proveedor).
		Requisito/s de datos relacionados	RDE4.4 + RDR4.4
		Número de referencia	RS4.4.2
		Descripción	El pedido no debe de haberse completado. En caso contrario cancelar el proceso.
		Requisito funcional relacionado	RF4.5 (Marcar pedido como completado)
		Requisito/s de datos relacionados	RDE4.4 + RDR4.4

RF4.5: Marcar pedido como completado			
Número de referencia	RF4.5		
Descripción del requisito	Se ha recibido el pedido de un producto y se marca como completado		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Solicitar el cambio de estado asociado a un pedido a “Completado”.	
	Requisito de datos	Número de referencia	RDE4.5
Datos internos		Composición	<ul style="list-style-type: none"> • Código de pedido. Cadena de caracteres (10).
Requisito de datos	Número de referencia	RDW4.5	
	Composición	<ul style="list-style-type: none"> • Código de pedido. Cadena de caracteres (10). • Estado del pedido. Cadena de caracteres (15) 	
Requisito de datos	Número de referencia	RDR4.5	
	Composición	<ul style="list-style-type: none"> • Código de pedido. Cadena de caracteres (10). • Estado del pedido. Cadena de caracteres (15) 	
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Mensaje de confirmación de pedido completado.	
	Requisito de datos	Número de referencia	Ninguno
Restricciones semánticas		Composición	Ninguno
	Número de referencia	RS4.5.1	
	Descripción	El código de pedido debe de ser válido. En caso contrario, cancelar el proceso.	
	Requisito funcional relacionado	RF4.2 (Realizar un pedido a un proveedor).	

	Requisito/s de datos relacionados	RDE4.5 + RDR4.5
	Número de referencia	RS4.5.1
	Descripción	El estado del pedido seleccionado debe de estar en “Pendiente”. En caso contrario, cancelar el proceso.
	Requisito funcional relacionado	RF4.2 (Realizar un pedido a un proveedor).
	Requisito/s de datos relacionados	RDR4.5

RF4.6: Dar de baja un proveedor			
Número de referencia	RF4.6		
Descripción del requisito	El empleado dará de baja a un proveedor del sistema.		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Solicitar la baja de un proveedor.	
	Requisito de datos	Número de referencia	RDE4.6
Datos internos		Composición	<ul style="list-style-type: none"> • Código de proveedor(CIF): Cadena de caracteres (10).
Requisito de datos	Número de referencia	RDW4.6	
	Composición	<ul style="list-style-type: none"> • Código de proveedor(CIF): Cadena de caracteres (10). • Estado. Booleano. 	
Requisito de datos	Número de referencia	RDR4.6	
	Composición	<ul style="list-style-type: none"> • Código de proveedor(CIF): Cadena de caracteres (10). • Estado. Booleano. 	
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Confirmación de la baja del proveedor.	
	Requisito de datos	Número de referencia	Ninguno
Restricciones semánticas		Composición	Ninguno
	Número de referencia	RS4.6.1	
	Descripción	El código de proveedor debe de existir en el sistema. En caso contrario, cancelar el proceso.	

		Requisito funcional relacionado	RF4.1 (Dar de alta a un proveedor).
		Requisito/s de datos relacionados	RDE4.6 + RDR4.6
		Número de referencia	RS4.6.2
		Descripción	El estado del proveedor debe estar "Activo" para poder darse de baja. En caso contrario, cancelar el proceso.
		Requisito funcional relacionado	RF4.1 (Dar de alta a un proveedor).
		Requisito/s de datos relacionados	RDR4.6

5. Subsistema de Informes y Análisis (Francisco José Ramos Moya)

RF5.1: Generar informe de ventas de un intervalo.			
Número de referencia	RF5.1		
Descripción del requisito	Permite generar un informe de ventas que resume las transacciones realizadas en un intervalo de fechas especificado.		
Entrada	Agente externo	Empleado.	
	Acción iniciada por el agente externo	Solicitar la generación de un informe de ventas para un intervalo de fechas.	
	Requisito de datos	Número de referencia	RDE5.1
Datos internos		Composición	<ul style="list-style-type: none"> • Fecha de inicio: Date. • Fecha de fin: Date.
Requisito de datos	Número de referencia	RDR5.1	
	Composición	<ul style="list-style-type: none"> • Códigos de Ventas: Lista de cadenas de caracteres (20). • Nombres de producto: Lista de cadena de caracteres (50). • Códigos de Producto: Lista de cadena de caracteres (13). • Cantidades: Lista de Números entero. • Fecha de Venta: Tipo Date. 	
Salida	Agente externo	Empleado.	
	Acción iniciada por el agente externo	Recepción del informe generado y mensaje de confirmación.	
	Requisito de datos	Número de referencia	RDS5.1
Restricciones semánticas		Composición	<ul style="list-style-type: none"> • Nombres de producto: Lista de cadena de caracteres (50). • Códigos de Producto: Lista de Cadena de caracteres (13). • Cantidades: Lista de Números enteros.
	Número de referencia	RS5.1.1	
	Descripción	El sistema debe verificar que la fecha de inicio no sea posterior a la fecha de fin. En caso contrario,	

			cancelar el proceso.
		Requisito funcional relacionado	RF5.1
		Requisito/s de datos relacionados	RDE5.1 + RDR5.1
		Número de referencia	RS5.1.2
		Descripción	El análisis debe limitarse a productos que tengan ventas en el intervalo especificado.
		Requisito funcional relacionado	RF5.1
		Requisito/s de datos relacionados	RDE5.1 + RDR5.1

RF5.2: Generar análisis de productos más vendidos.			
Número de referencia	RF5.2		
Descripción del requisito	Este requisito permite generar un informe del análisis de productos más vendidos en un intervalo		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Solicitar la generación de un análisis de productos más vendidos en un intervalo.	
	Requisito de datos	Número de referencia	RDE5.2
Datos internos	Requisito de datos	Composición	<ul style="list-style-type: none"> • Fecha de inicio: Date • Fecha de fin: Date • Límite de productos: número entero (indica cuántos productos se mostrarán)
		Número de referencia	RDR5.2
	Requisito de datos	Composición	<ul style="list-style-type: none"> • Códigos de Ventas: Lista de cadenas de caracteres (20). • Nombres de producto: Lista de cadena de caracteres (50). • Códigos de Producto: Lista de cadena de caracteres (13). • Cantidades: Lista de Números entero. • Fecha de Venta: Tipo Date.
Salida	Agente externo	Empleado.	
	Acción iniciada por el agente externo	Lista de productos más vendidos.	
	Requisito de datos	Número de referencia	RDS5.2
Restricciones semánticas		Composición	<ul style="list-style-type: none"> • Nombres de producto: Lista de cadena de caracteres (50). • Códigos de Producto: Lista de cadena de caracteres (13). • Cantidades: Lista de Números entero.
		Número de referencia	RS5.2.1
		Descripción	El análisis debe limitarse a productos que tengan ventas en el intervalo especificado.
		Requisito funcional relacionado	RF5.2

		Requisito/s de datos relacionados	RDE5.2 + RDR5.2
		Número de referencia	RS5.2.2
		Descripción	La fecha de fin debe ser posterior a la fecha de inicio. En caso contrario, cancelar el proceso.
		Requisito funcional relacionado	RF5.2
		Requisito/s de datos relacionados	RDE5.2
		Número de referencia	RS5.2.3
		Descripción	El límite de productos debe de ser positivo. En caso contrario, cancelar el proceso.
		Requisito funcional relacionado	RF5.2
		Requisito/s de datos relacionados	RDE5.2 + RDR5.2

RF5.3: Generar análisis de ingresos para un producto.			
Número de referencia	RF5.3		
Descripción del requisito	Generar análisis de ingresos para un producto utilizando su código.		
Entrada	Agente externo	Empleado	
	Acción iniciada por el agente externo	Solicitar la generación de un análisis de ingresos por producto mediante el código del producto.	
	Requisito de datos	Número de referencia RDE5.3 Composición <ul style="list-style-type: none"> • Código del producto: Cadena de caracteres (13). • Fecha de inicio: Date • Fecha de fin: Date 	
Datos internos	Requisito de datos	Número de referencia RDR5.3 Composición <ul style="list-style-type: none"> • Códigos de Ventas: Lista de cadenas de caracteres (20). • Nombres de producto: Lista de cadena de caracteres (50). • Códigos de Producto: Lista de cadena de caracteres (13). • Cantidades: Lista de Números entero. • Fecha de Venta: Tipo Date. 	
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Mensaje de confirmación del análisis y reporte de ingresos del producto.	
	Requisito de datos	Número de referencia RS5.3 Composición <ul style="list-style-type: none"> • Nombre de Producto. Cadena de caracteres (50). • Código Producto: Cadena de caracteres (13). • Ingresos. Número decimal. 	
Restricciones semánticas		Número de referencia RSR5.3.1	
		Descripción La fecha de fin debe de ser posterior a la fecha de inicio.	
		Requisito funcional relacionado RF5.3	

	Requisito/s de datos relacionados	RDE5.3
	Número de referencia	RSR 5.3.2
	Descripción	Las ventas seleccionadas para el análisis deben de estar comprendidas en el rango de fechas.
	Requisito funcional relacionado	RF5.3
	Requisito/s de datos relacionados	RDE5.3 + RDR5.3

RF5.4: Generar informe de pedidos de un intervalo.			
Número de referencia	RF5.4		
Descripción del requisito	Permite generar un informe de pedidos que resume las transacciones realizadas en un intervalo de fechas especificado.		
Entrada	Agente externo	Empleado.	
	Acción iniciada por el agente externo	Solicitar la generación de un informe de pedidos para un intervalo de fechas.	
	Requisito de datos	Número de referencia	RDE5.4
		Composición	<ul style="list-style-type: none"> • Fecha de inicio: Date. • Fecha de fin: Date.
Datos internos	Requisito de datos	Número de referencia	RDR5.4
		Composición	<ul style="list-style-type: none"> • Códigos de Pedidos: Lista de cadenas de caracteres (20). • Nombres de producto: Lista de cadena de caracteres (50). • Códigos de Producto: Lista de cadena de caracteres (13). • Cantidades: Lista de Números entero. • Fecha de Solicitud: Tipo Date.
Salida	Agente externo	Empleado	
	Acción iniciada por el agente externo	Recepción del informe generado y mensaje de confirmación.	
	Requisito de datos	Número de referencia	RDS5.4
		Composición	<ul style="list-style-type: none"> • Nombres de producto: Lista de cadena de caracteres (50). • Códigos de Producto: Lista de cadena de caracteres (13). • Cantidades: Lista de Números entero.
Restricciones semánticas		Número de referencia	RS5.4.1
		Descripción	El sistema debe verificar que la fecha de inicio no sea posterior a la fecha de fin.
		Requisito funcional	RF5.4

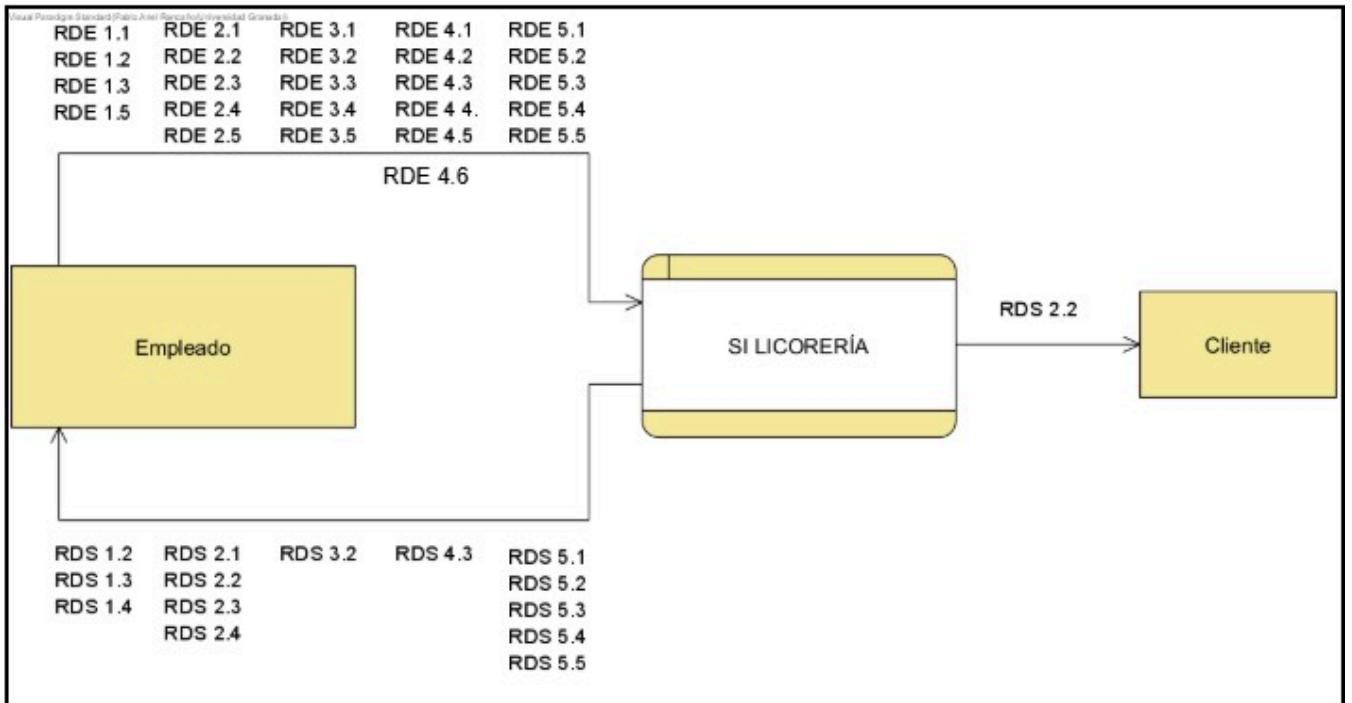
		relacionado	
		Requisito/s de datos relacionados	RDE5.4
		Número de referencia	RSR 5.4.2
		Descripción	Los pedidos seleccionados para el análisis deben de estar comprendidos en el rango de fechas.
		Requisito funcional relacionado	RF5.4
		Requisito/s de datos relacionados	RDE5.4 + RDR5.4

RF5.5: Generar análisis de ventas por familia de productos.			
Número de referencia	RF5.5		
Descripción del requisito	Generar un informe del análisis de ventas por familia de producto.		
Entrada	Agente externo	Empleado.	
	Acción iniciada por el agente externo	Solicitar el análisis de ventas para la familia de productos especificada durante un período determinado.	
	Requisito de datos	Número de referencia RDE5.5 Composición <ul style="list-style-type: none"> • Fecha de inicio: Date • Fecha de fin: Date • Categoría. Cadena de caracteres (30). 	
Datos internos	Requisito de datos	Número de referencia RDR5.5	
		Composición <ul style="list-style-type: none"> • RDE5.5 • Códigos de ventas. Lista de cadena de caracteres • Cantidad. Lista de números enteros. • Categoría del producto. Cadena de caracteres (30). 	
Salida	Agente externo	Empleado.	
	Acción iniciada por el agente externo	Confirmación del análisis y reporte de ventas por productos.	
	Requisito de datos	Número de referencia RS5.5 Composición <ul style="list-style-type: none"> • Categoría. Cadena de caracteres (50). • Cantidad. Número Entero. 	
Restricciones semánticas		Número de referencia RSR5.5.2	
		Descripción	La fecha de fin debe de ser posterior a la fecha de inicio.
		Requisito funcional relacionado	RF5.3

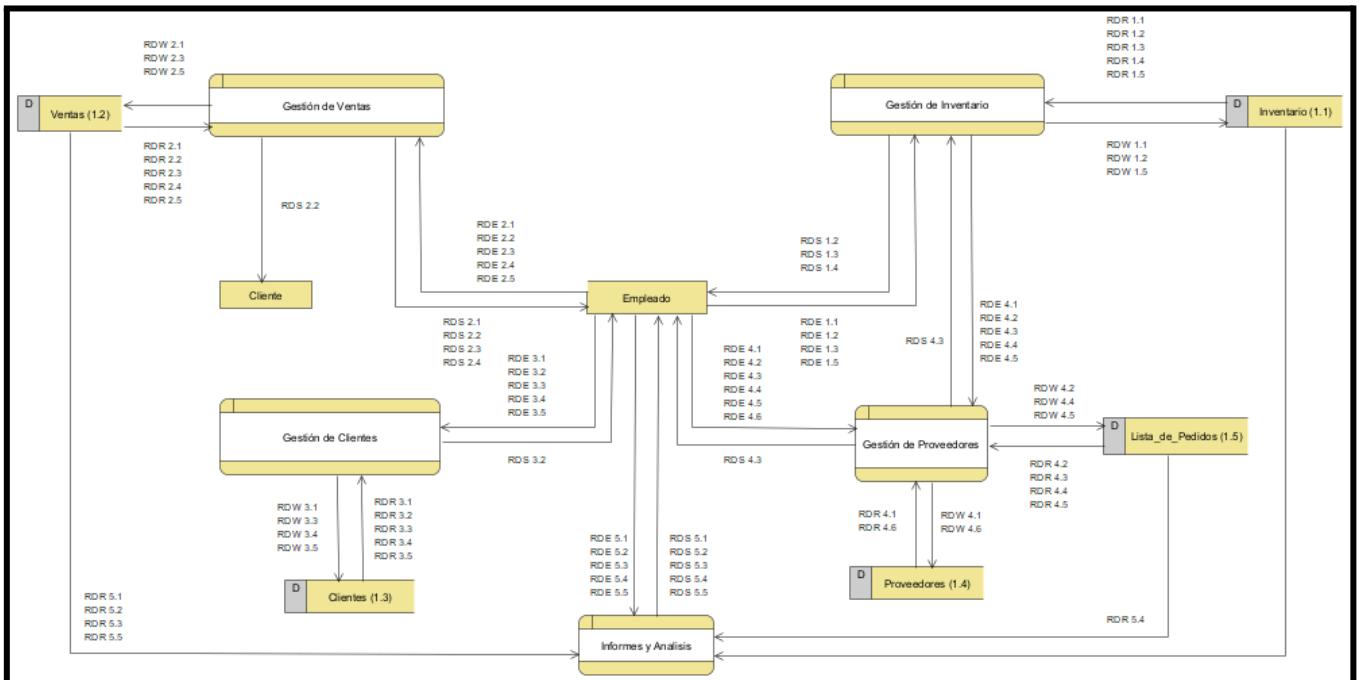
		Requisito/s de datos relacionados	RDE5.3 .
		Número de referencia	RSR5.5.3
		Descripción	Las ventas y pedidos seleccionados para el análisis deben de estar comprendidas en el rango de fechas.
		Requisito funcional relacionado	RF5.3
		Requisito/s de datos relacionados	RDE5.3 + RDR5.3.

DFDs

Caja negra

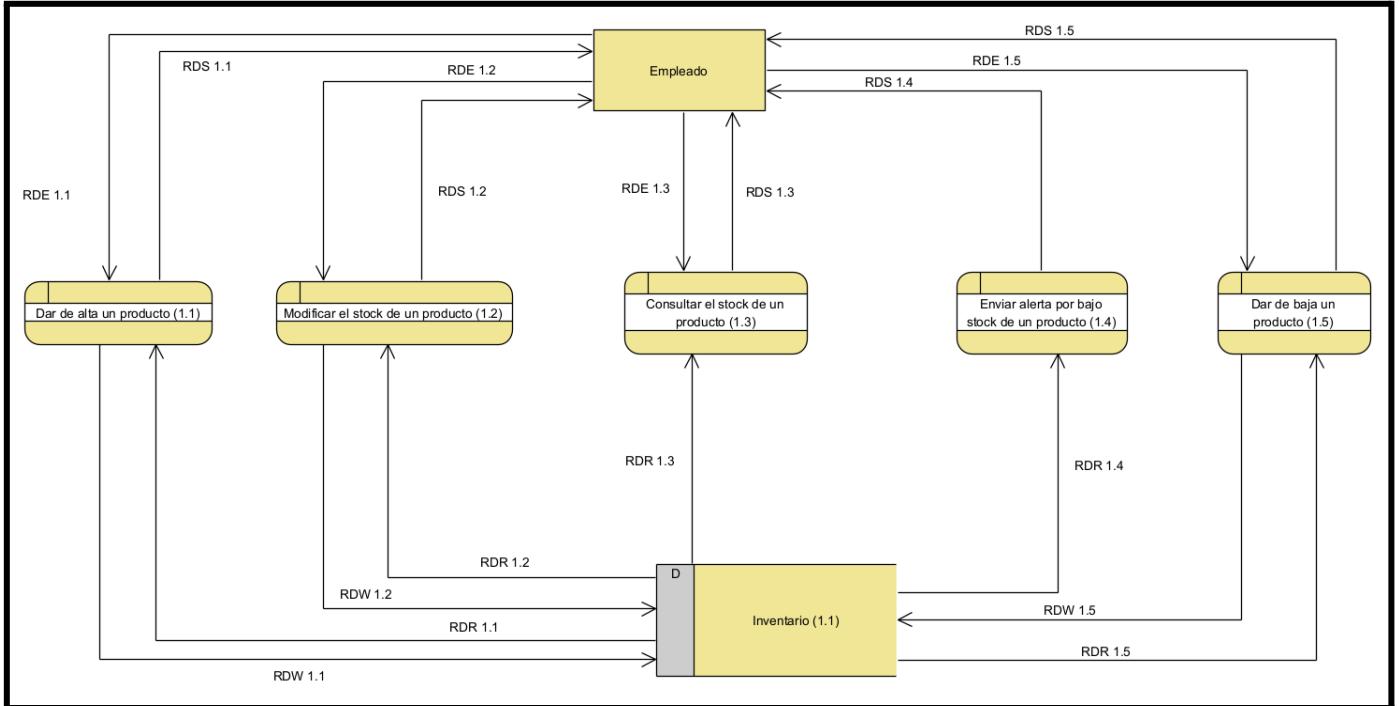


Armazón (DFD0)

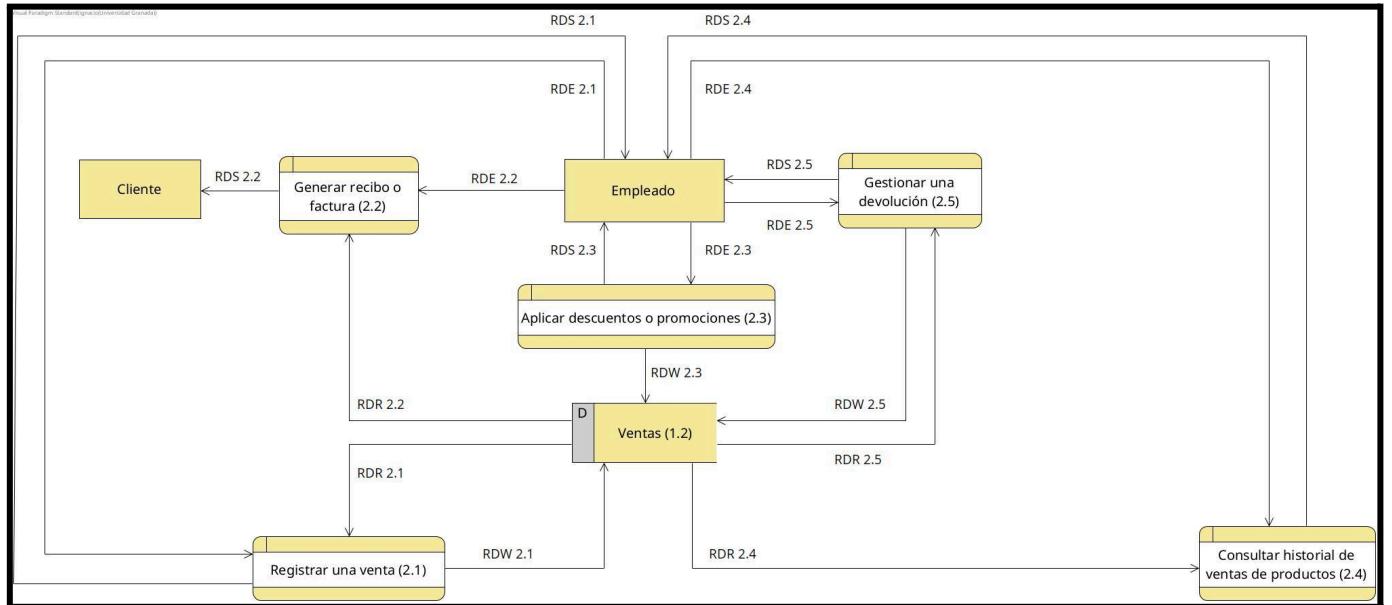


DFD1s

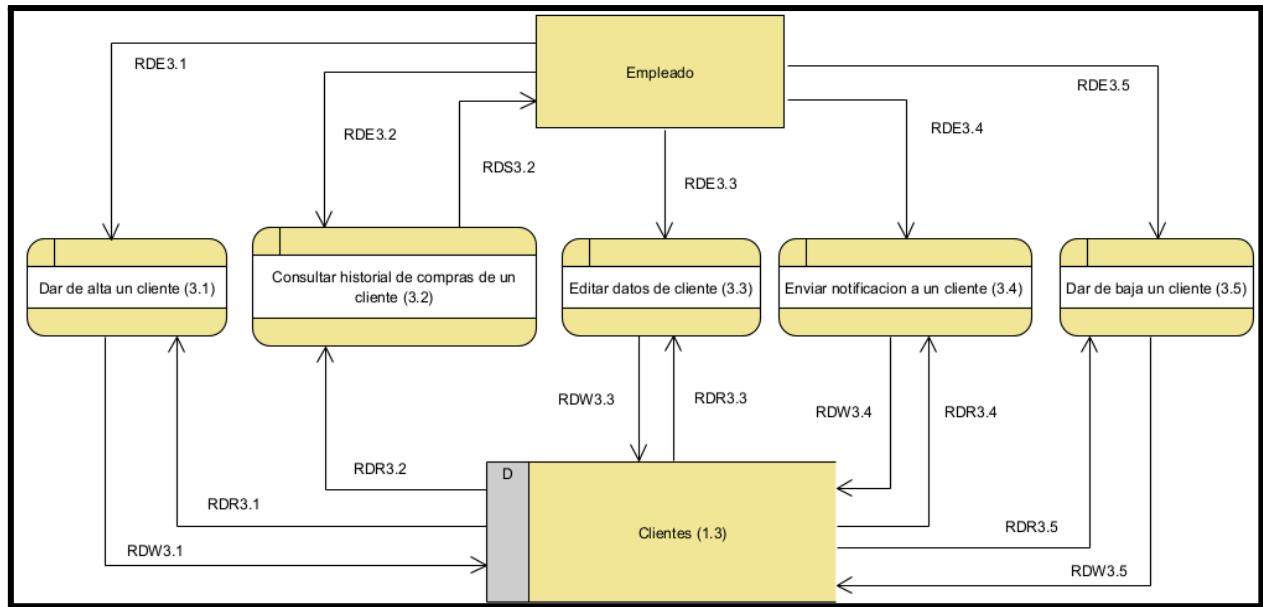
DFD1 Subsistema de Gestión de Inventario



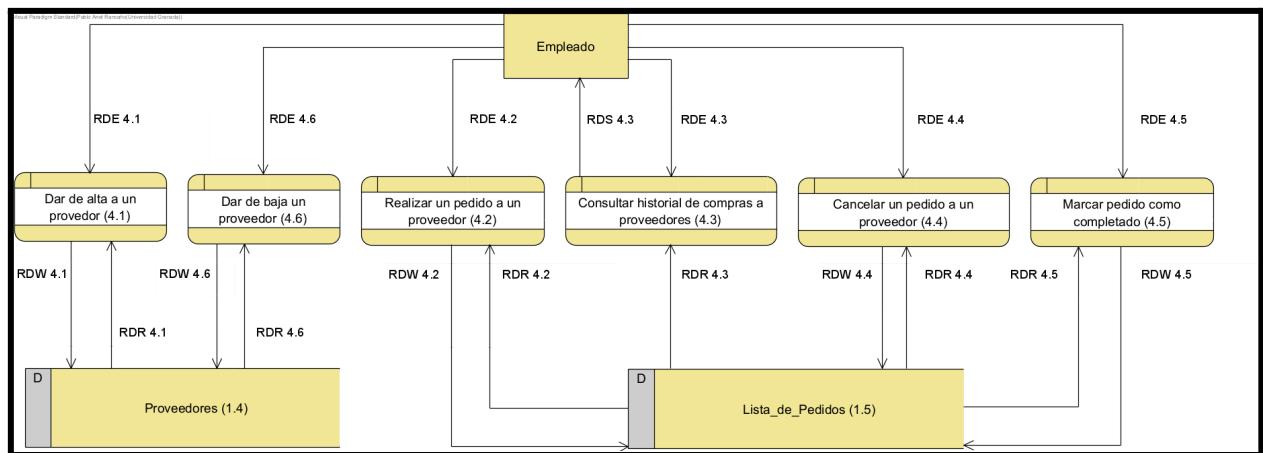
DFD1 Subsistema de Gestión de Ventas



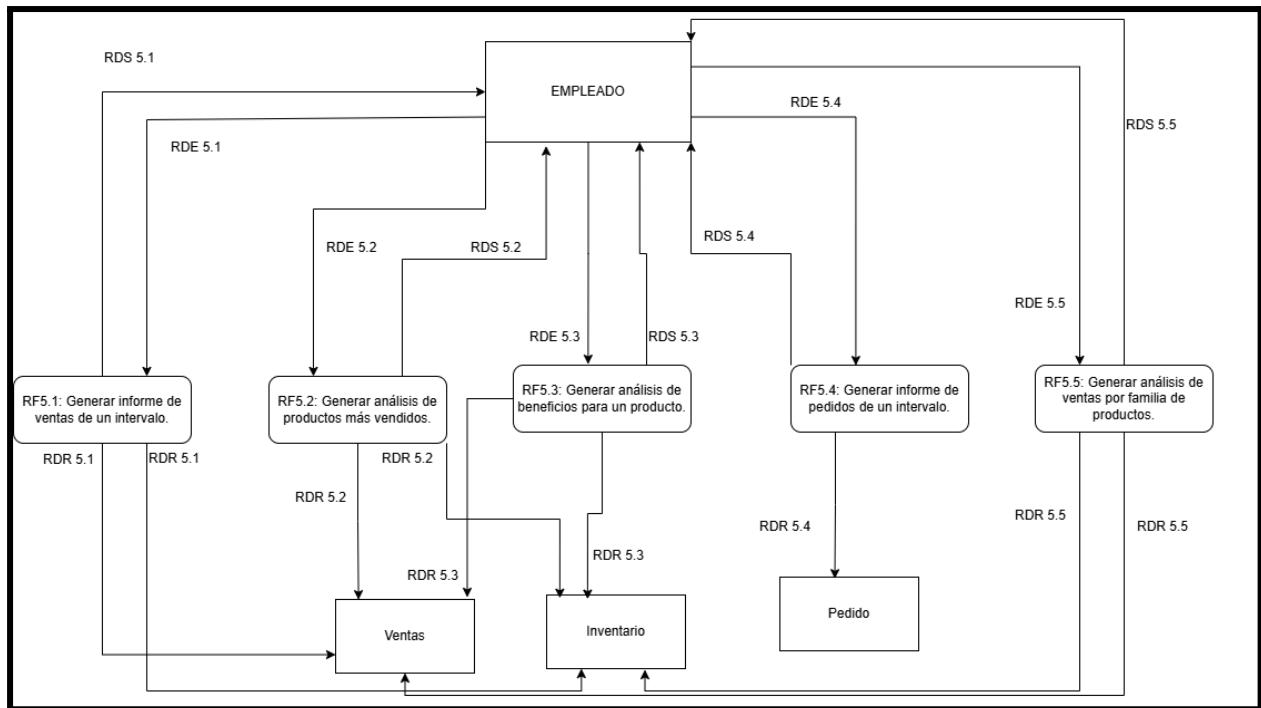
DFD1 Subsistema de Gestión de Clientes



DFD1 Subsistema de Gestión de Proveedores



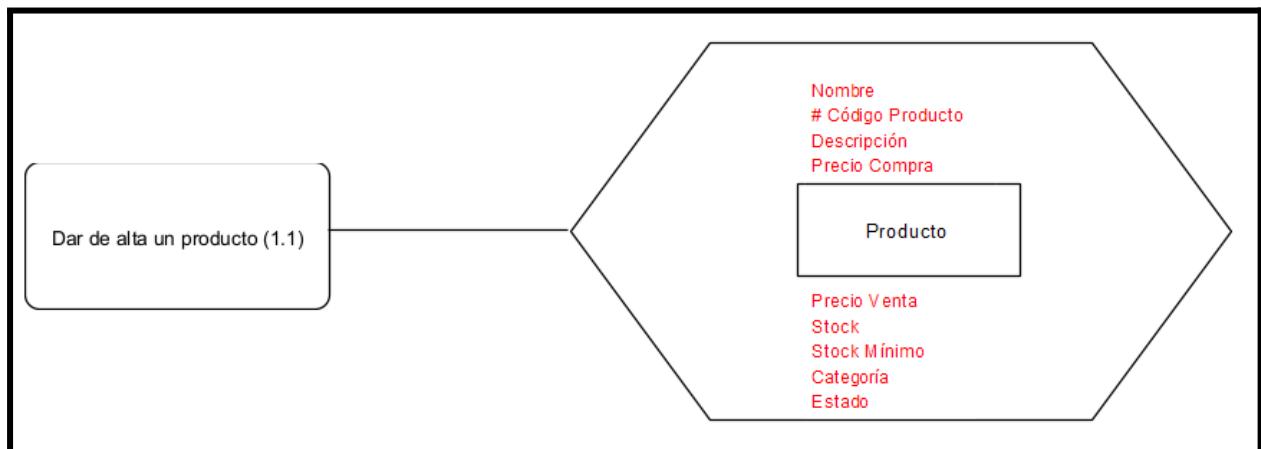
DFD1 Subsistema de Gestión de Análisis e Informes

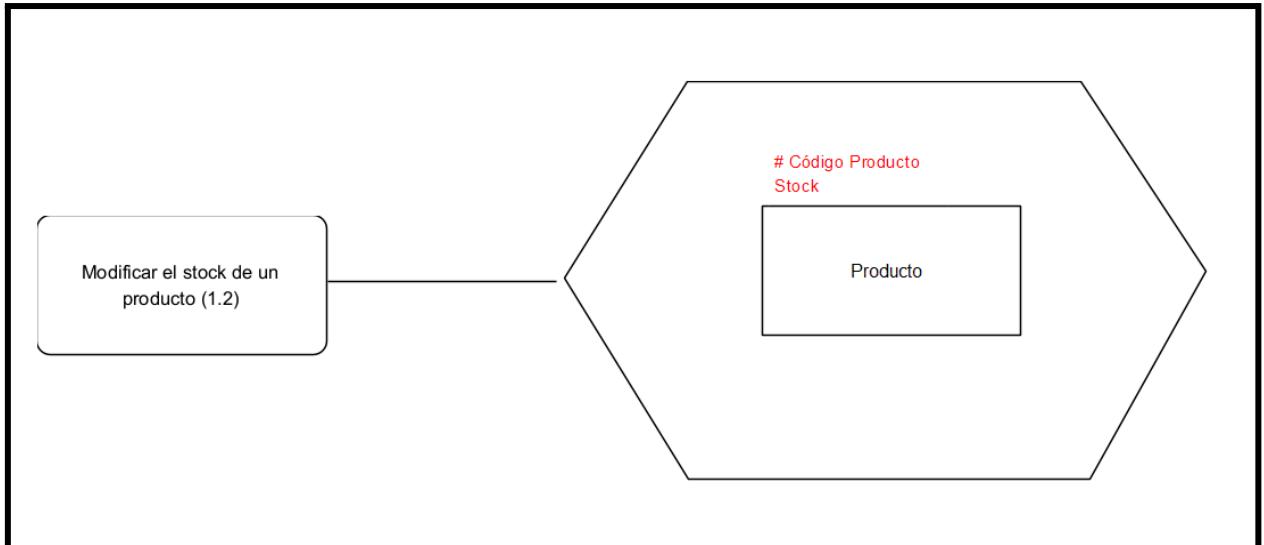


Esquemas externos

Subsistema de gestión de inventario.

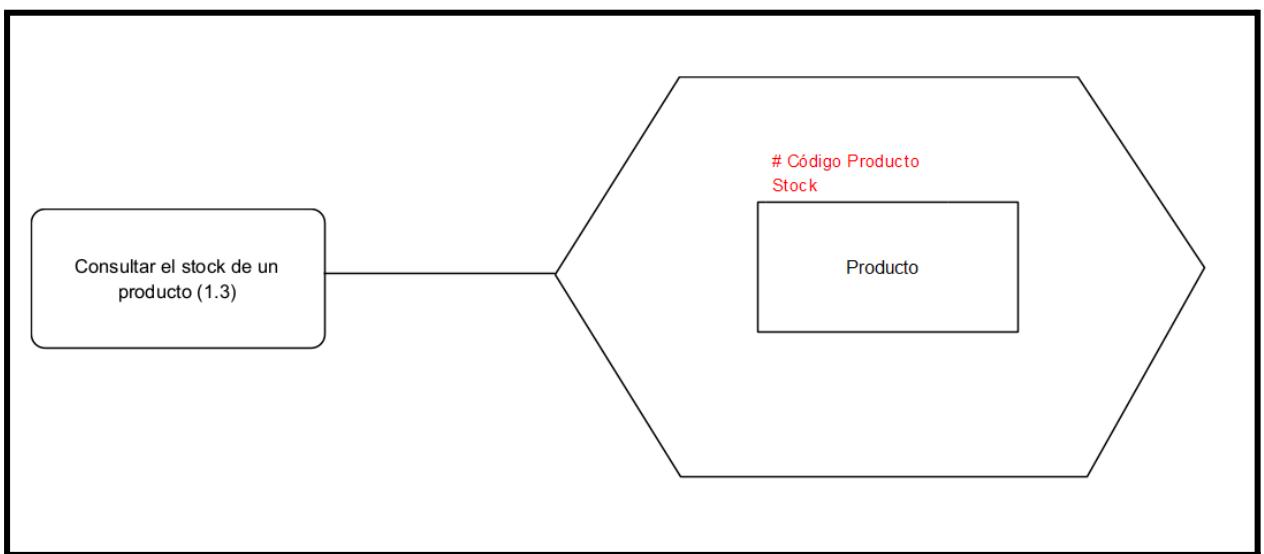
DFD1 (RF 1.1): El empleado registra un nuevo producto en el sistema.



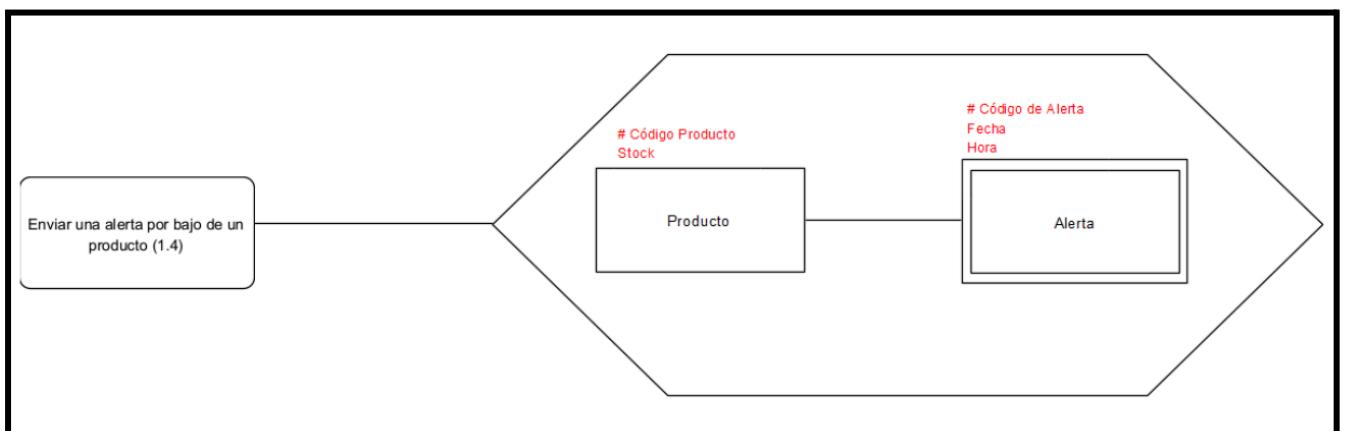


DFD1 (RF 1.2): El empleado modifica el stock de un producto ya creado

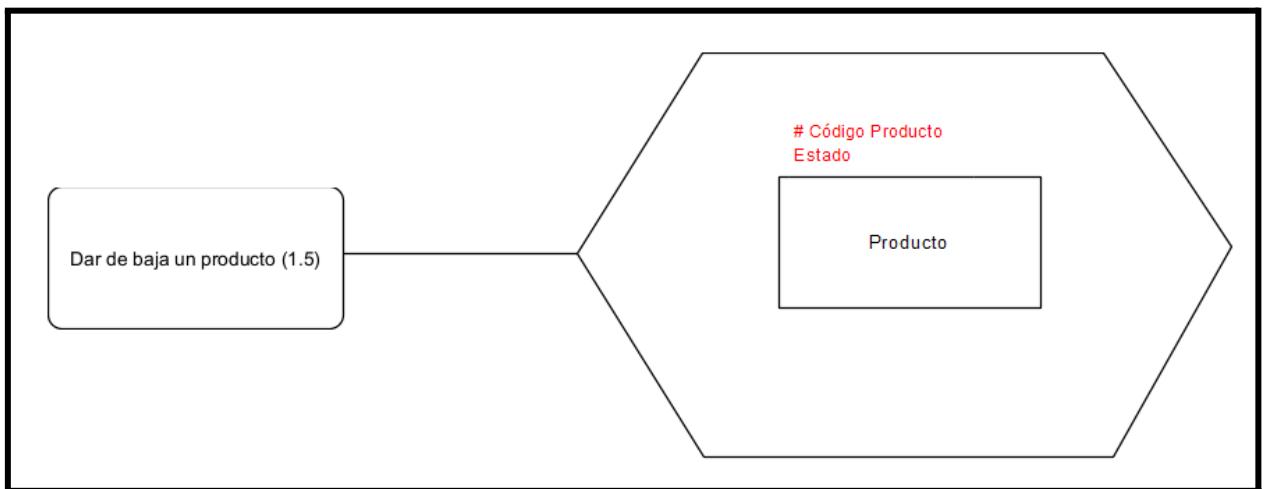
DFD1 (RF 1.3): El empleado puede consultar el stock de un producto



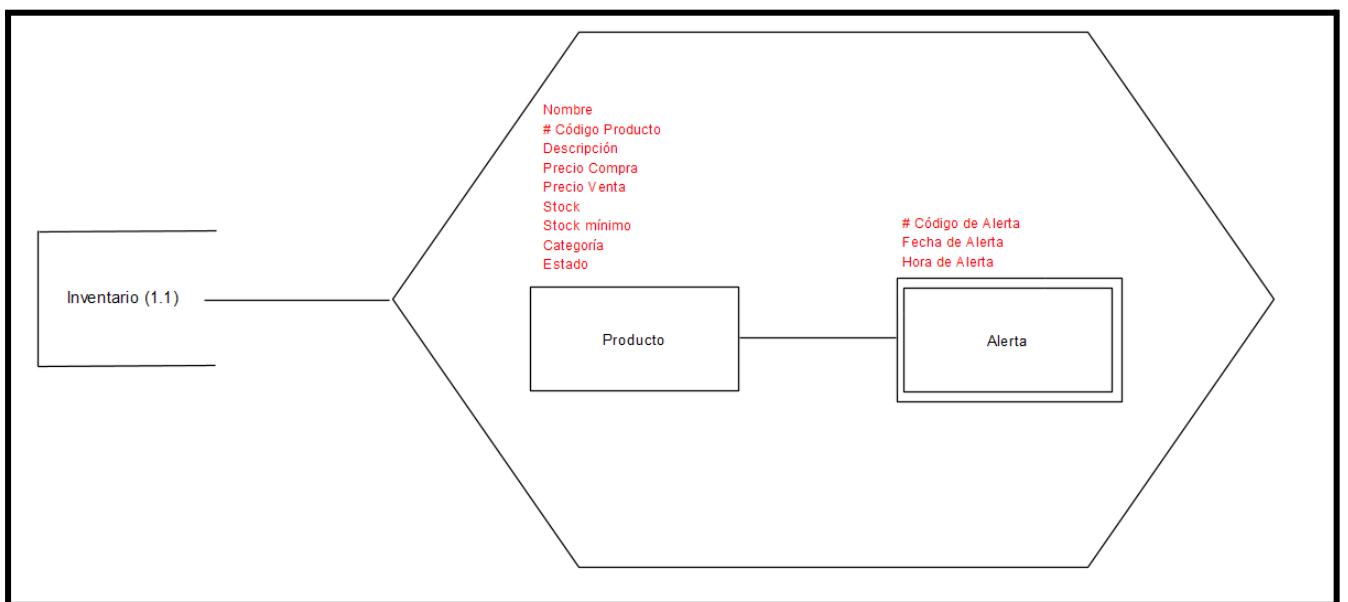
DFD1 (RF 1.4): El empleado recibe una alerta por bajo stock de un producto.



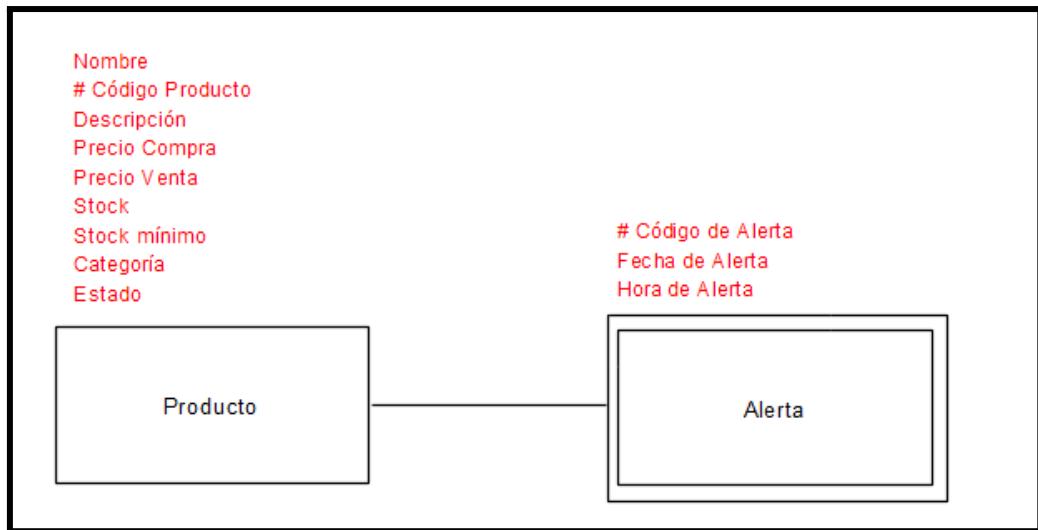
DFD1 (RF 1.5): El empleado puede dar de baja un producto, es decir descatalogarlo.



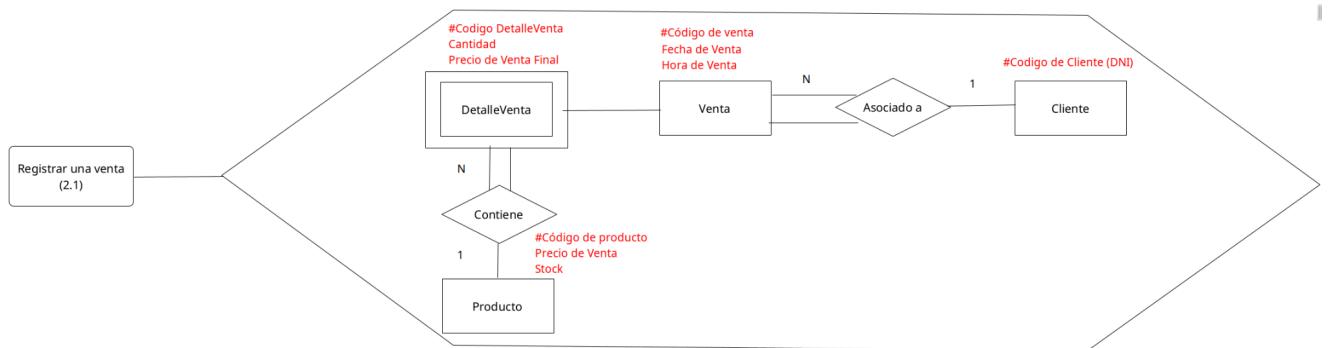
Almacenes de datos (Subsistema de Gestión de Inventario)



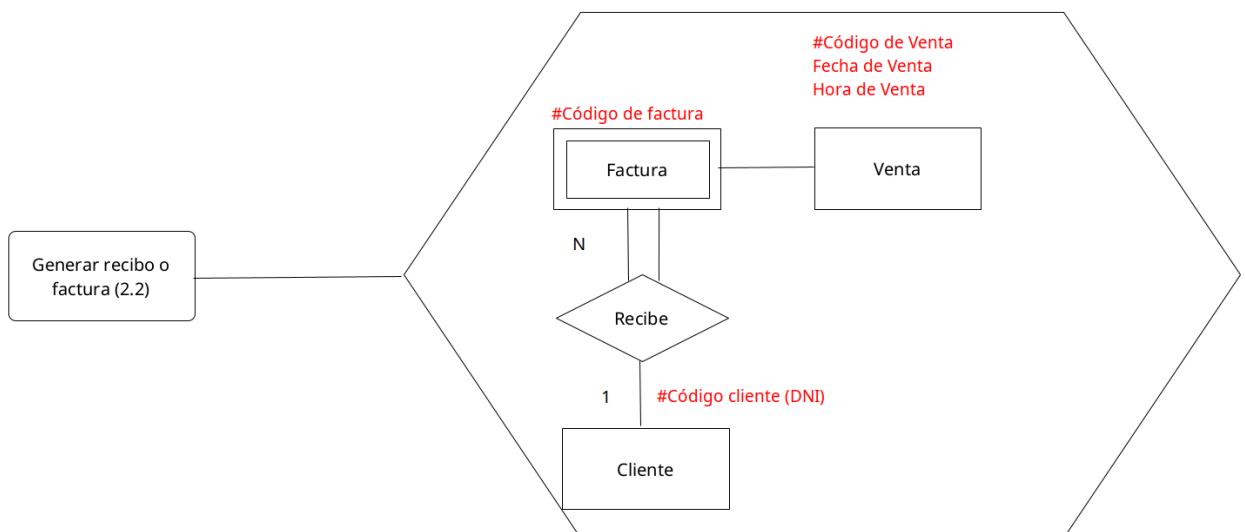
Esquema Conceptual Subsistema Gestión de Inventario



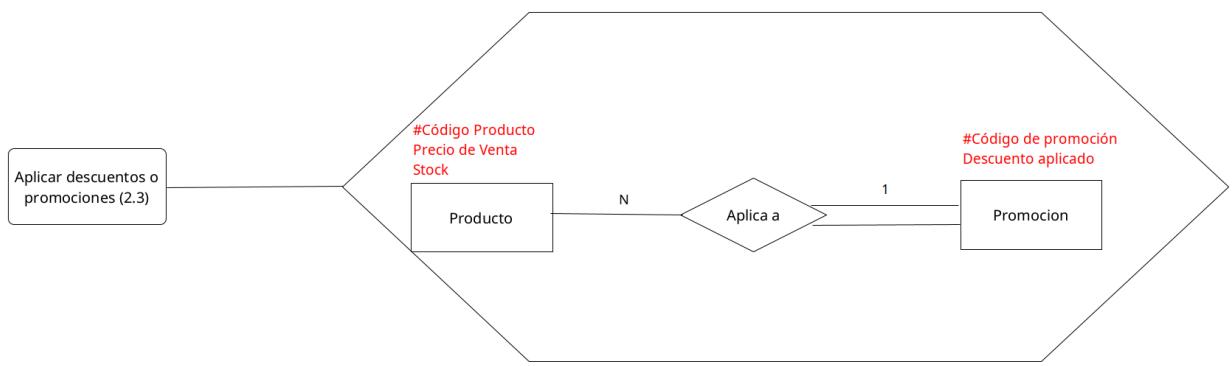
Subsistema de gestión de ventas.



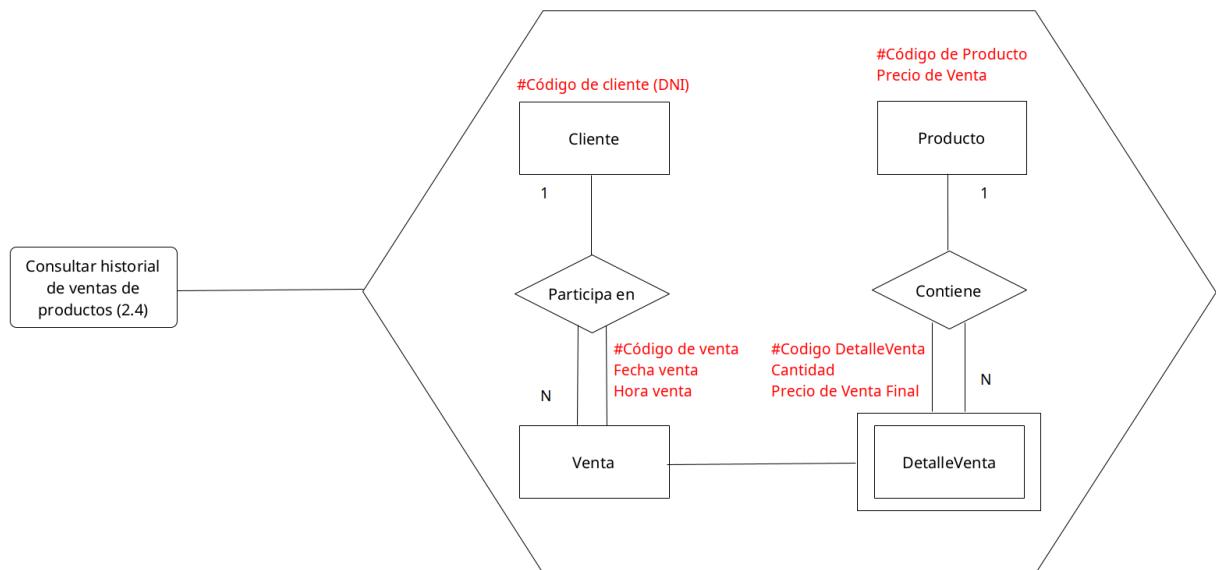
DFD1 (RF 2.2): Generar un recibo o factura de una venta



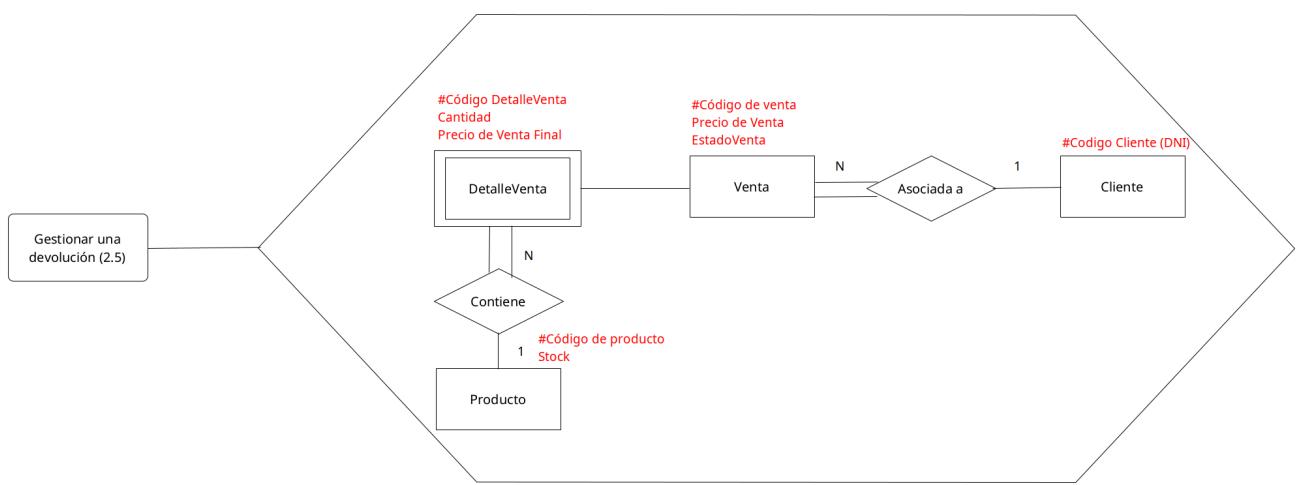
DFD1 (RF 2.3): Aplicar promociones a productos y rebajar su precio



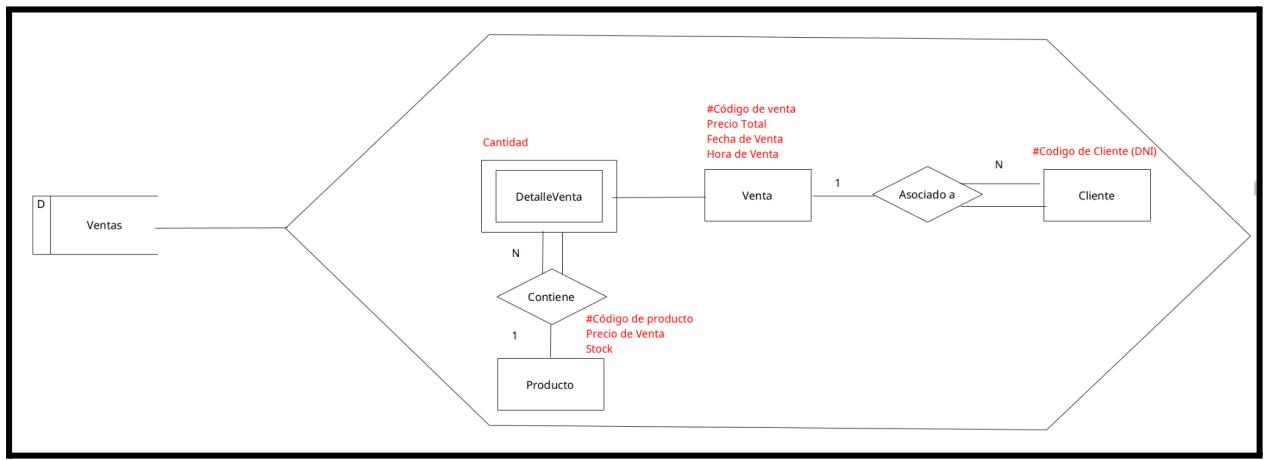
DFD1 (RF 2.4): Realizar una consulta del registro de transacciones de ventas de productos



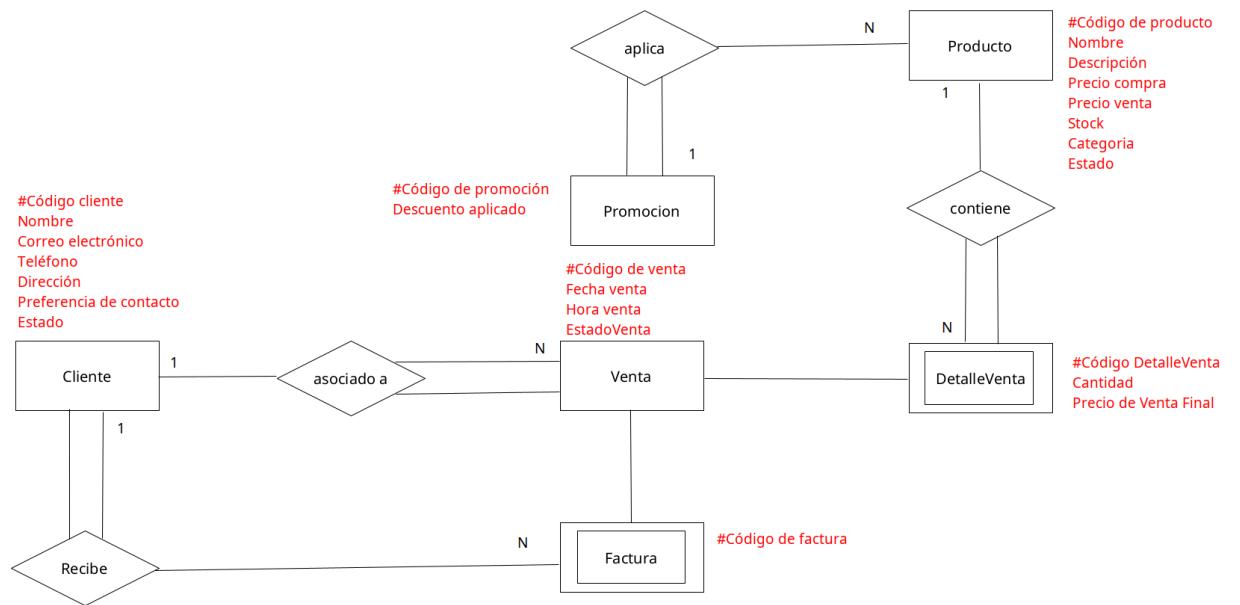
DFD1 (RF 2.5): El cliente realiza una devolución



Almacenes de datos (Subsistema de Gestión de Ventas)

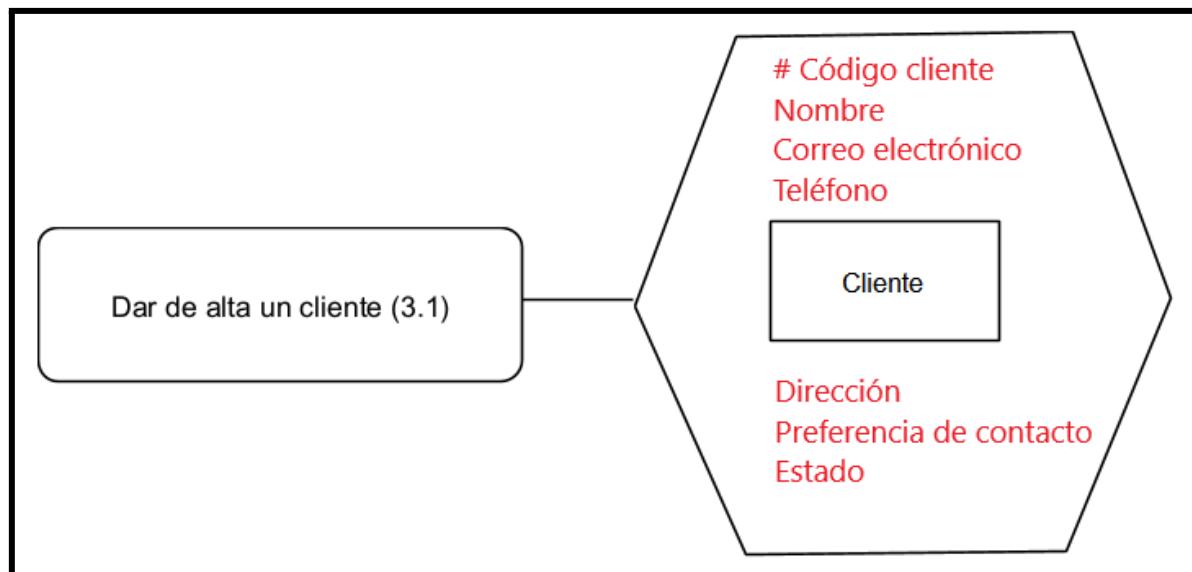


Esquema Conceptual Subsistema Gestión de Ventas

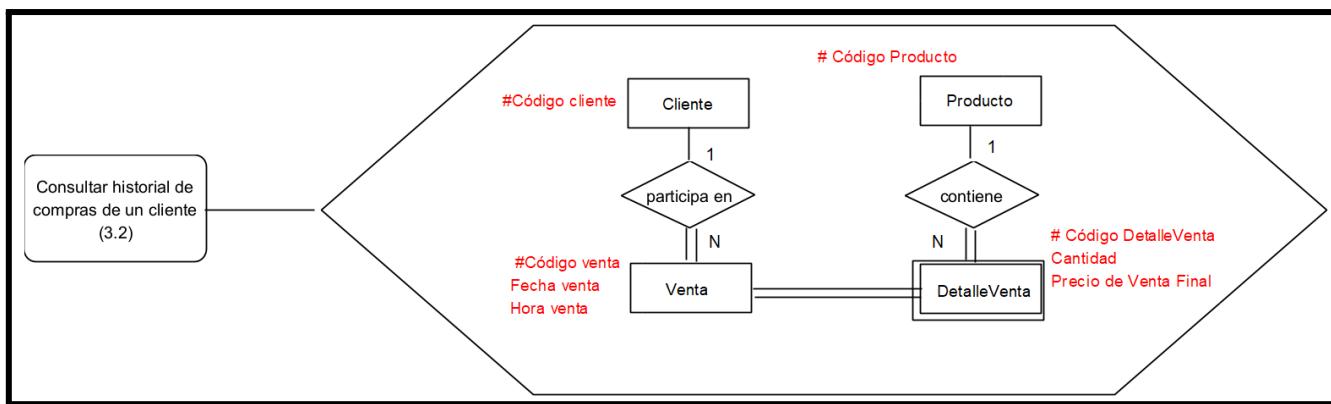


Subsistema de gestión de clientes.

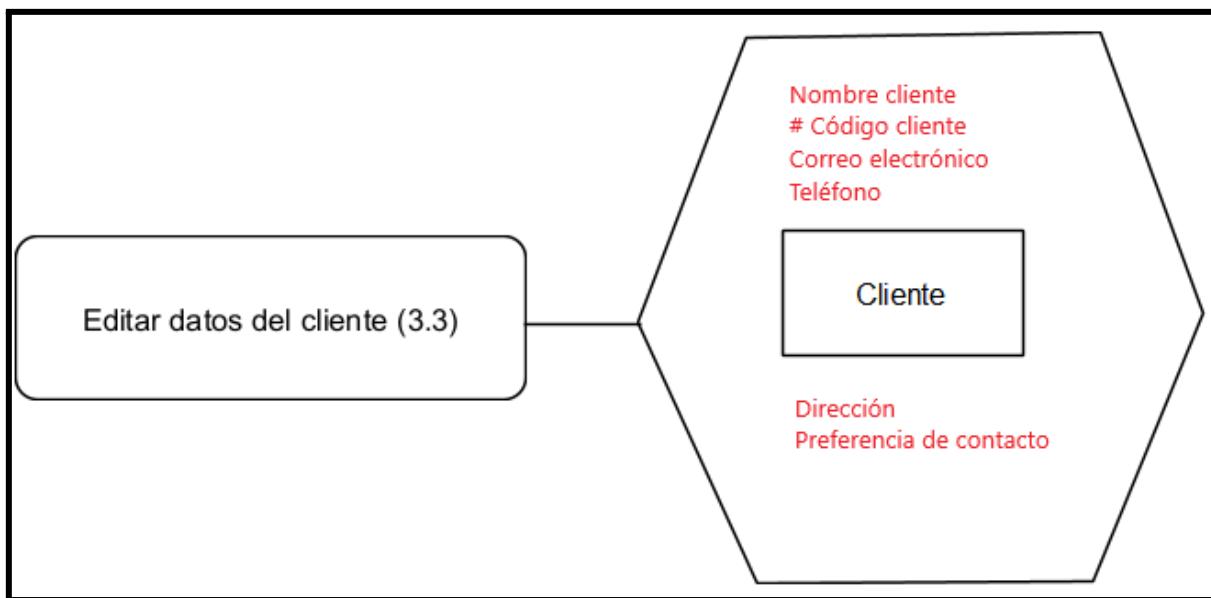
DFD1 (RF 3.1): El empleado registra un nuevo cliente en el sistema.



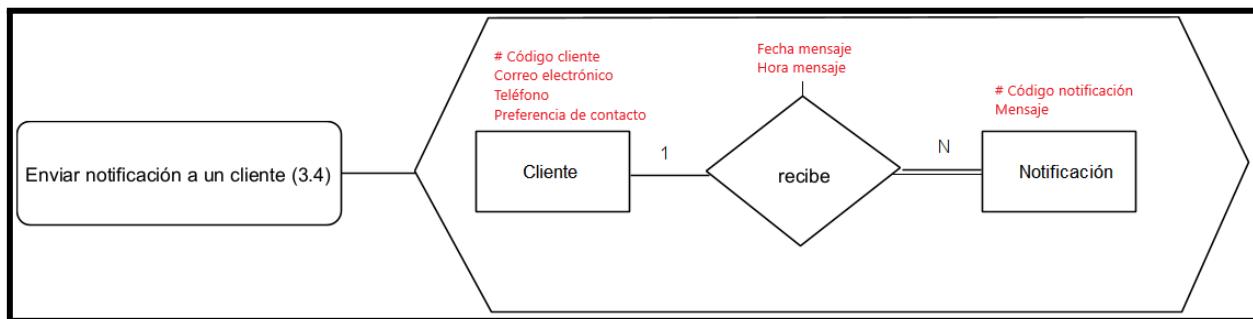
DFD1 (RF 3.2): El empleado consulta las compras de un cliente



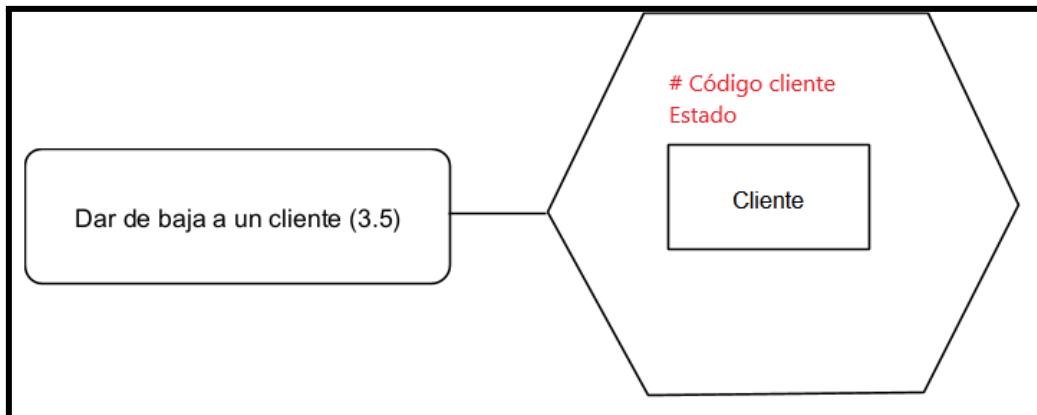
DFD1 (RF 3.3): El empleado edita datos del cliente



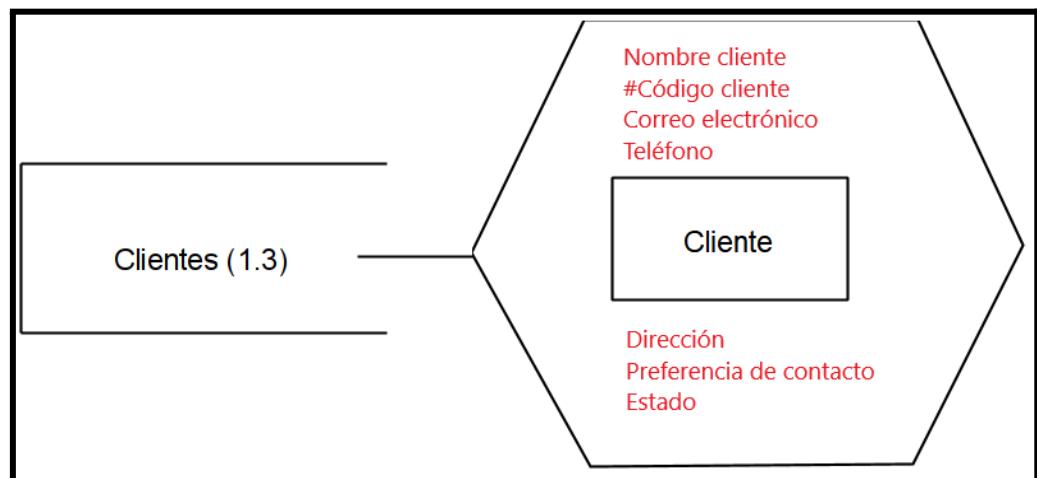
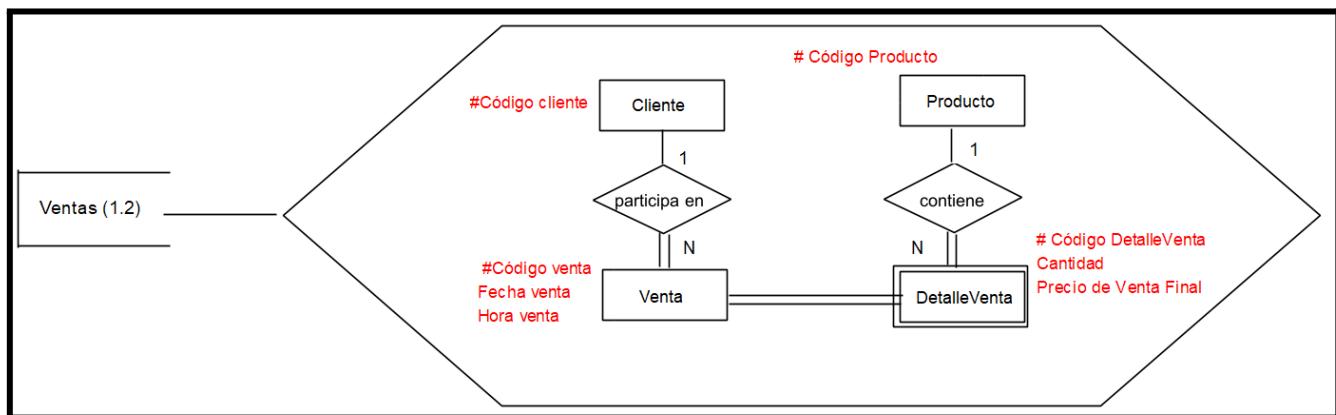
DFD1 (RF 3.4): El empleado envía una notificación a cliente



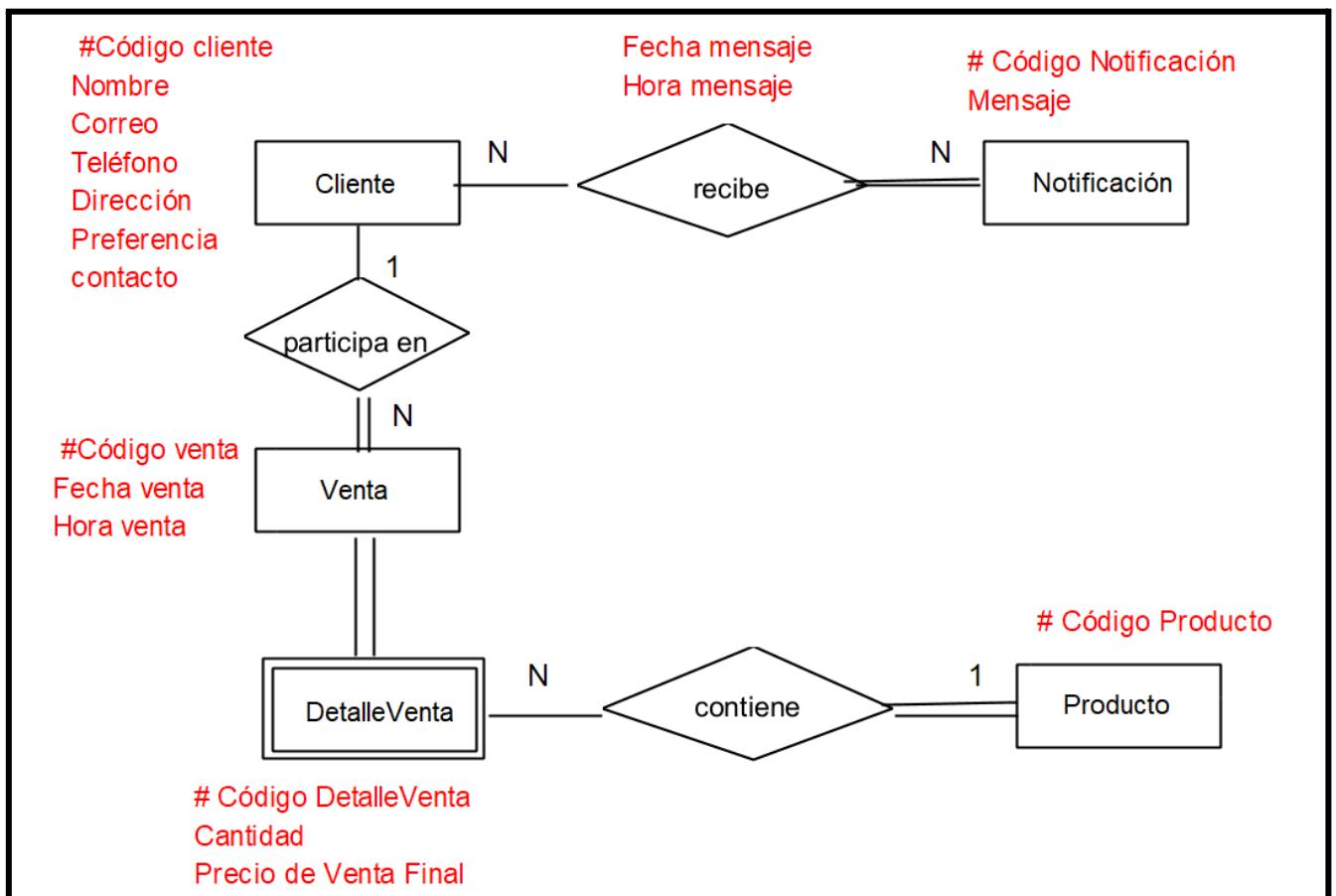
DFD1 (RF 3.5): El empleado da de baja un cliente



Almacenes de datos (Subsistema de Gestión de Clientes)

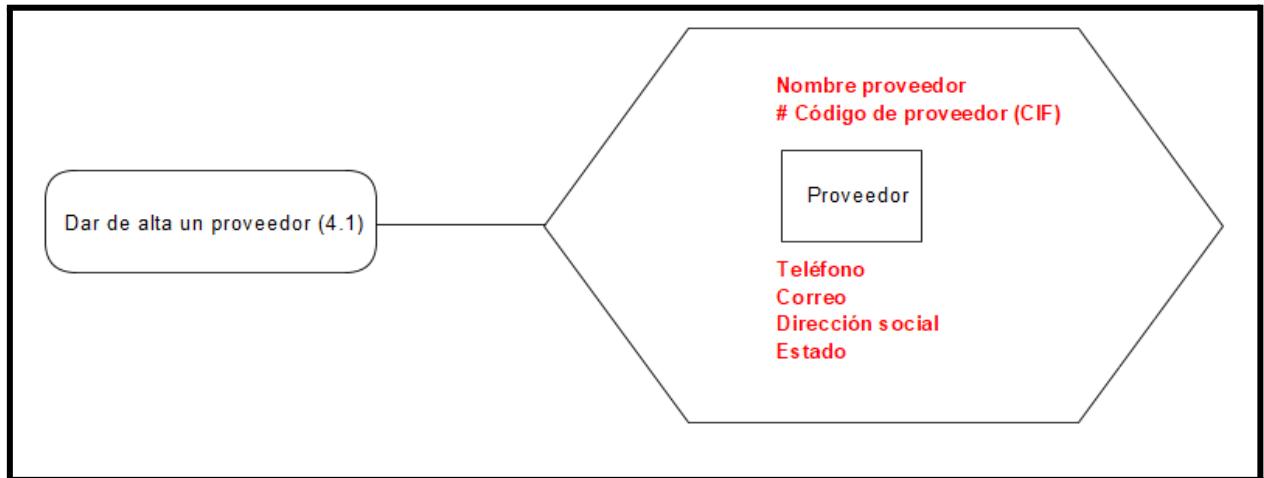


Esquema Conceptual Subsistema Gestión de clientes.

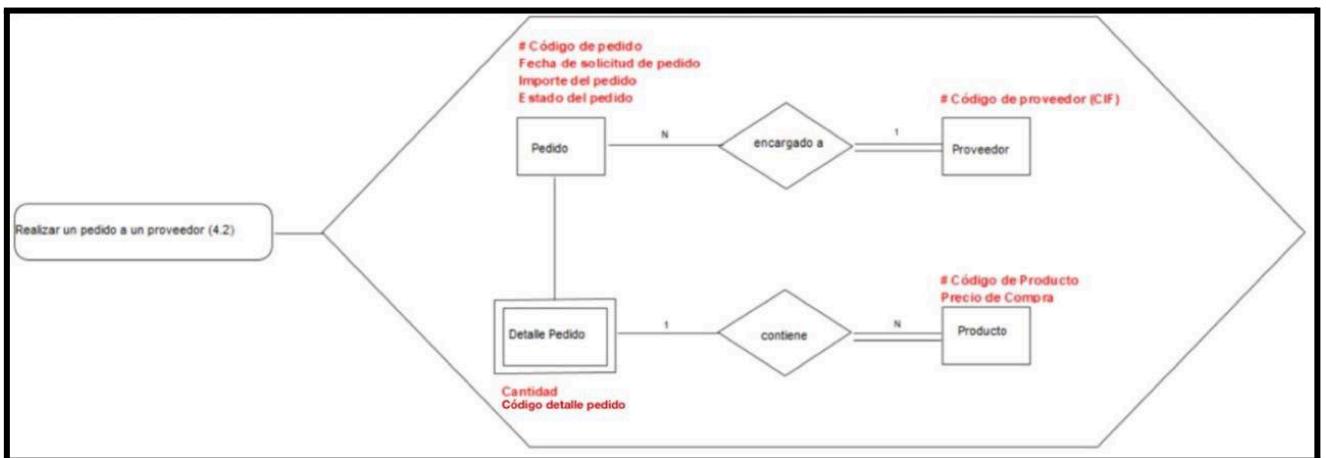


Subsistema de gestión de proveedores.

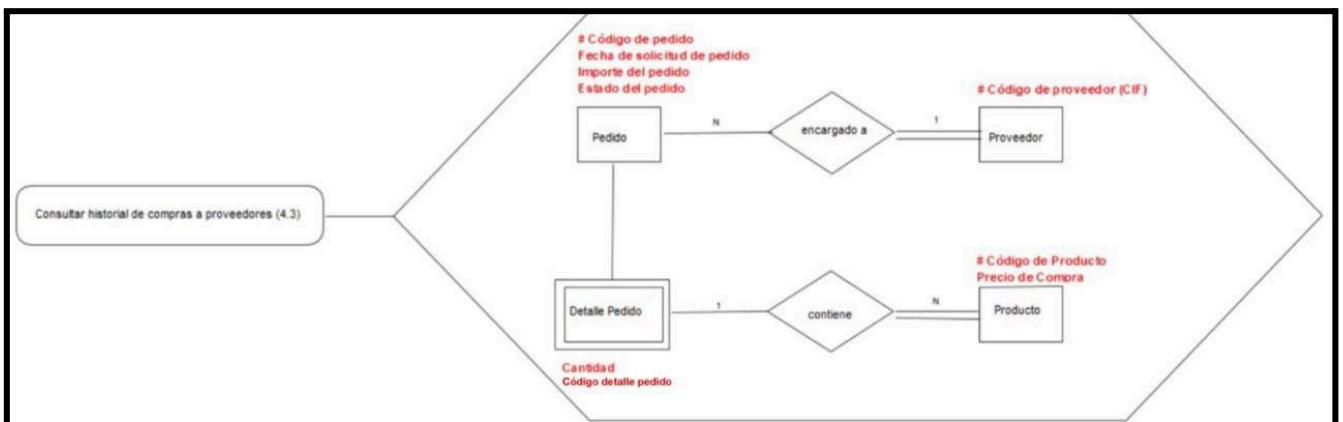
DFD1 (RF 4.1): El empleado registra un nuevo proveedor en el sistema.



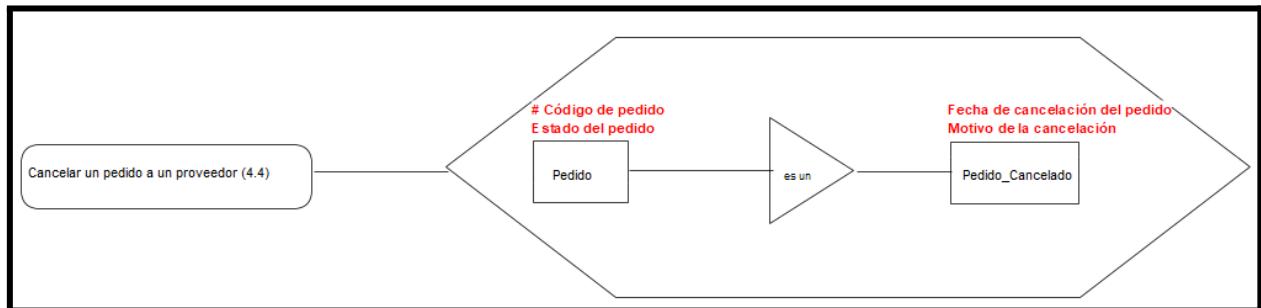
DFD1 (RF 4.2): El empleado realiza un pedido a los proveedores.



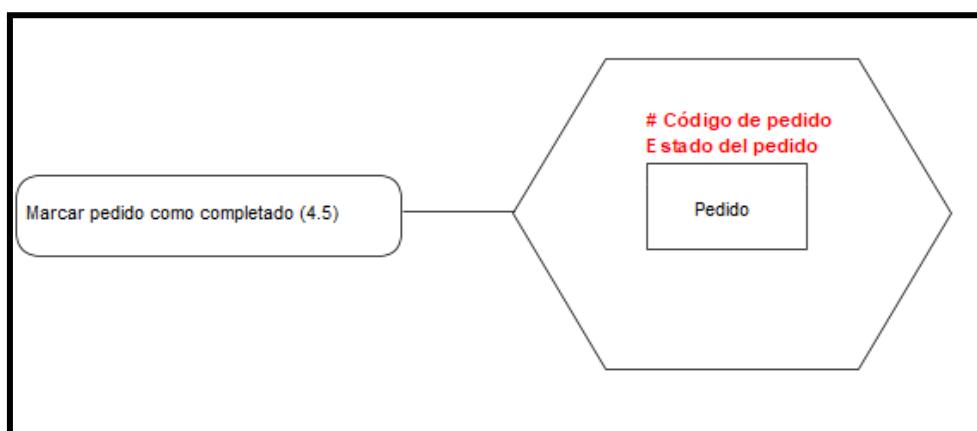
DFD1 (RF 4.3): El empleado consulta el historial de compras realizadas a los proveedores; pedidos, fechas, productos y cantidades.



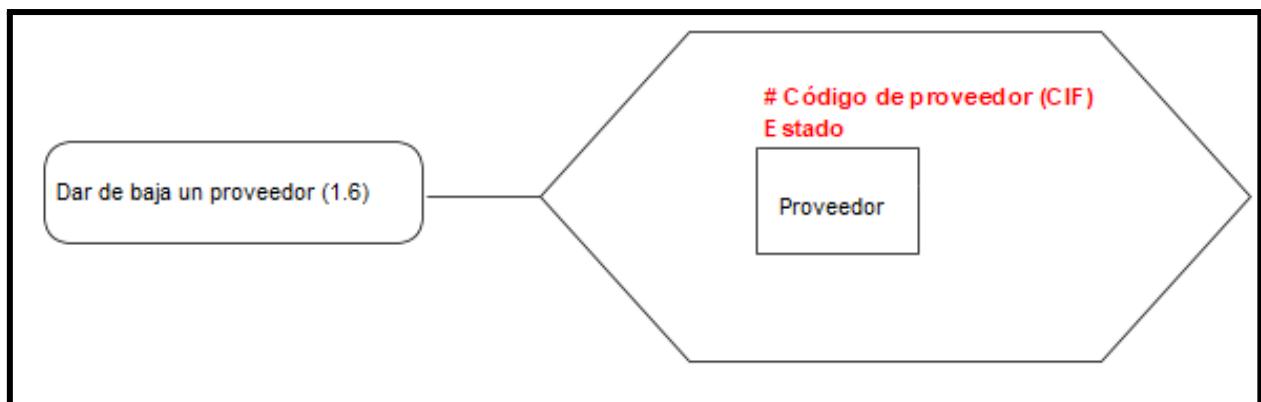
DFD1 (RF 4.4): El empleado puede cancelar un pedido que se haya realizado previamente a un proveedor.



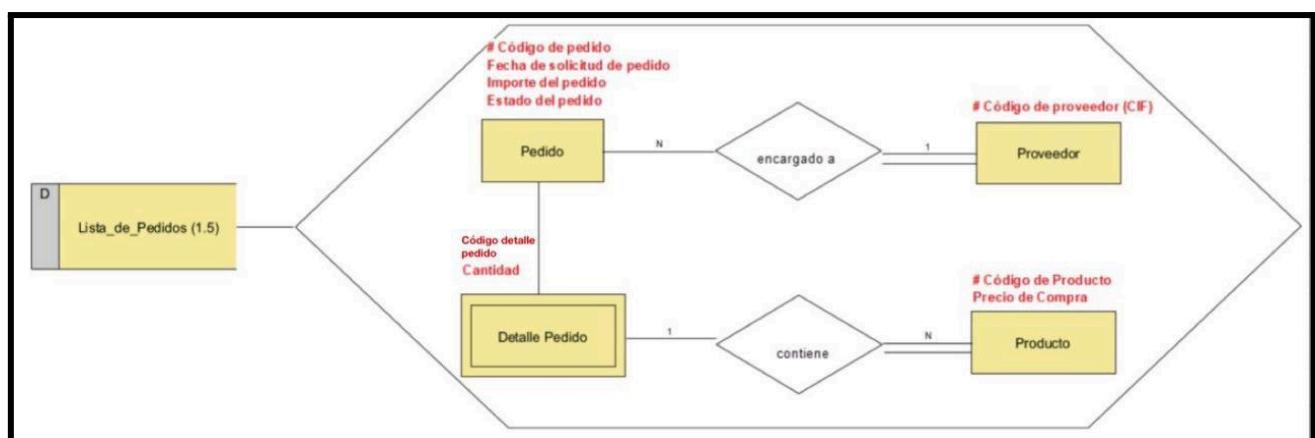
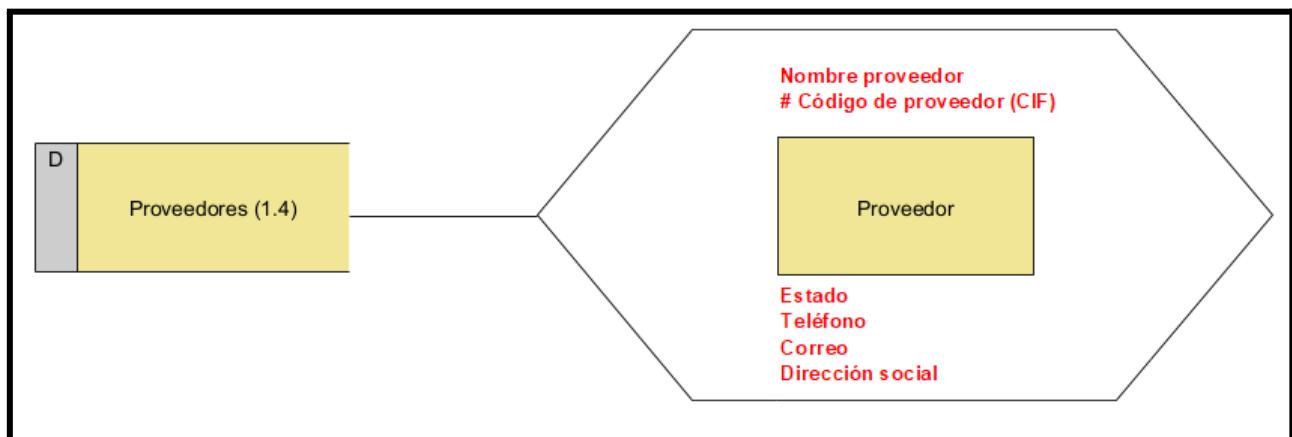
DFD1 (RF 4.5): Se ha recibido el pedido de un producto y se marca como completado.



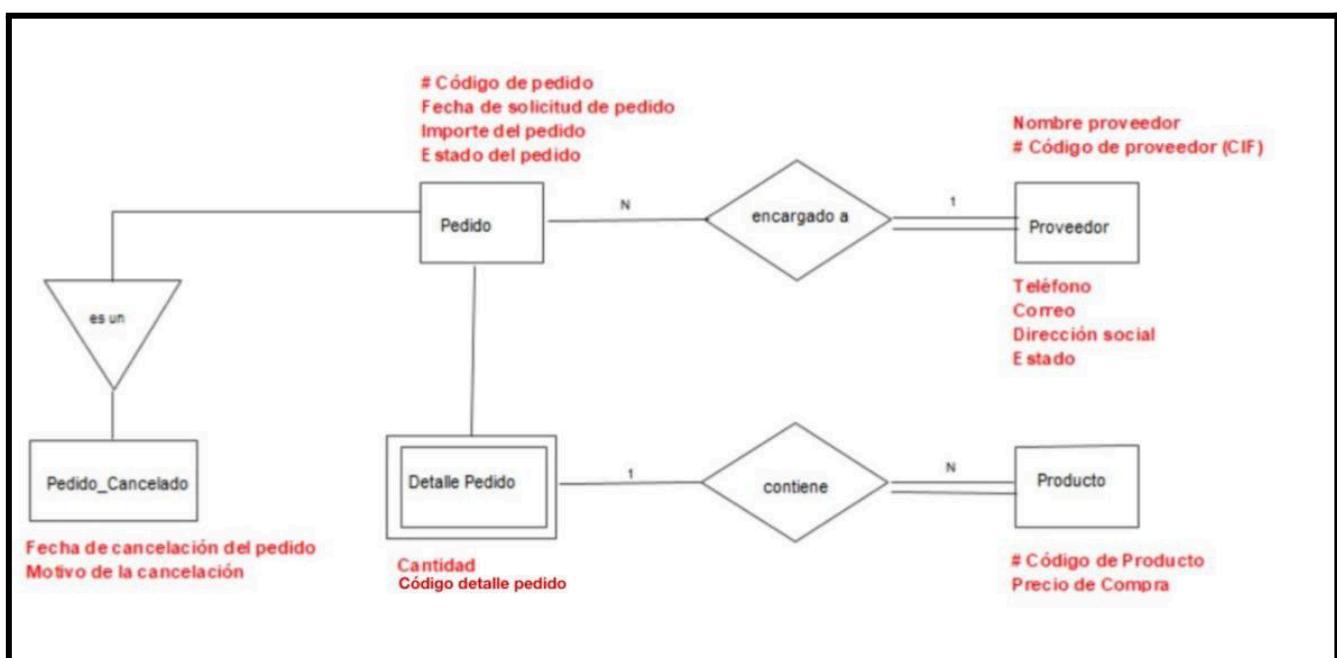
DFD1 (RF 4.6): El empleado dará de baja a un proveedor del sistema.



Almacenes de datos (Subsistema de Gestión de Proveedores)

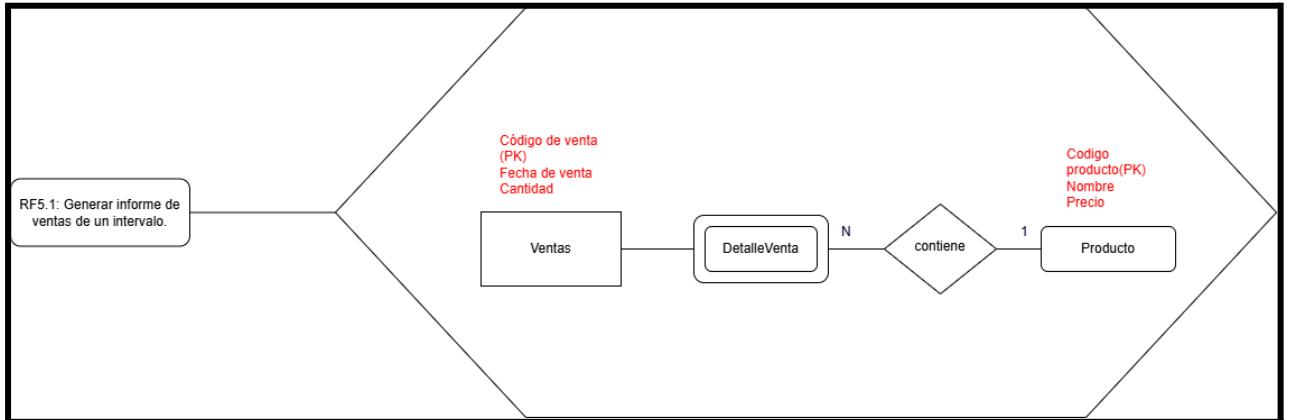


Esquema Conceptual Subsistema Gestión de proveedores

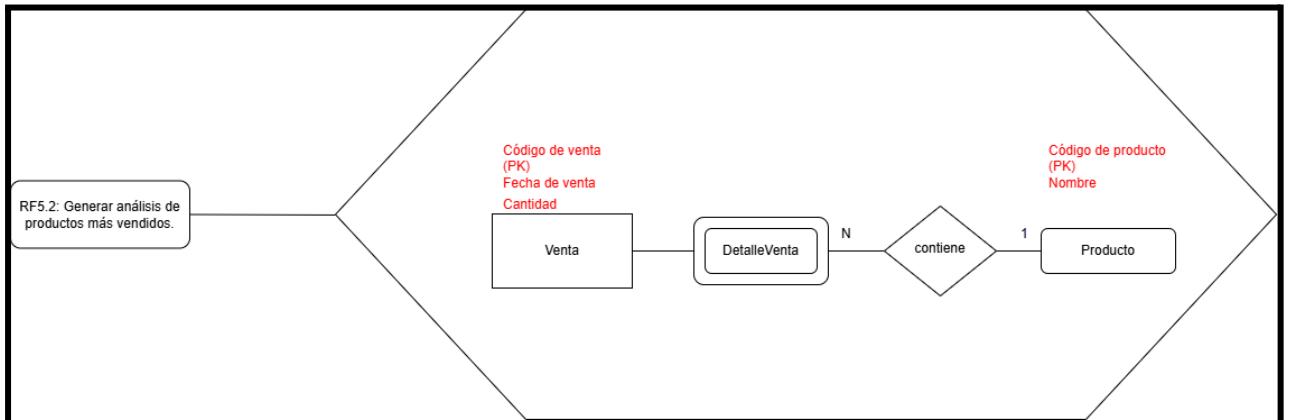


Subsistema de gestión de análisis e informes.

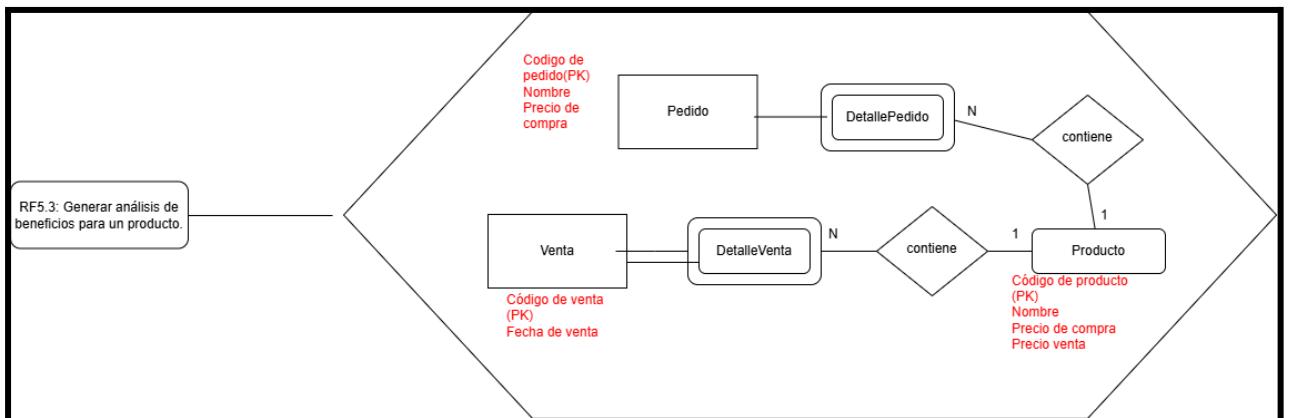
DFD1 (RF 5.1): Genera un informe que resume las ventas realizadas en un período de tiempo específico.



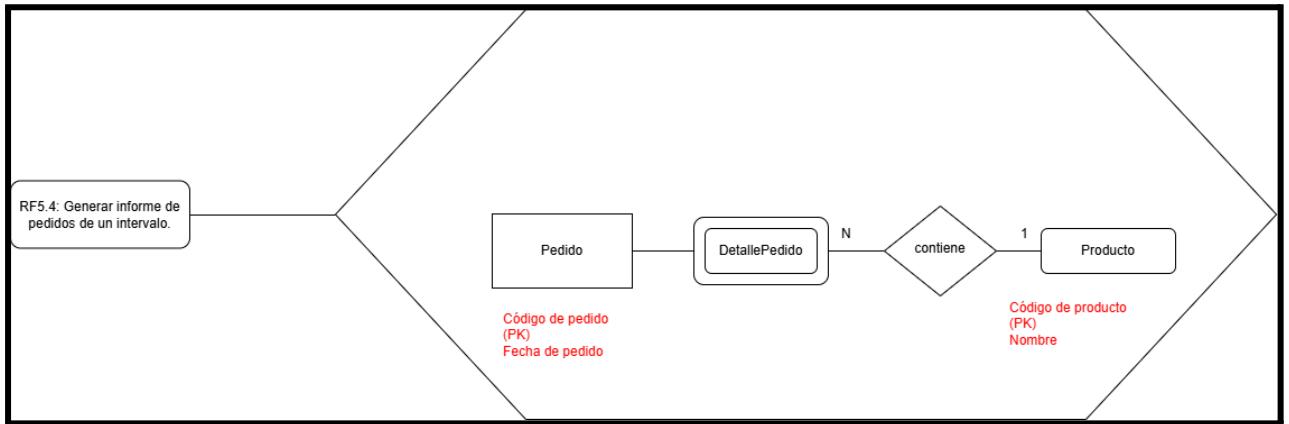
DFD1 (RF 5.2): Analiza y devuelve los productos más vendidos en un intervalo de tiempo, destacando los más populares.



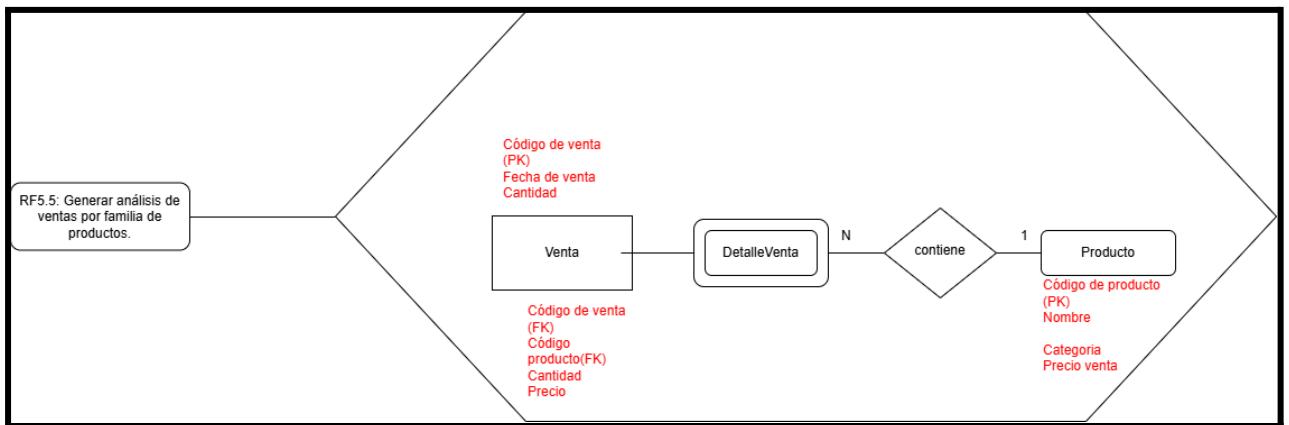
DFD1 (RF 5.3): Calcula los ingresos, gastos y beneficios por un producto específico durante un intervalo de tiempo.



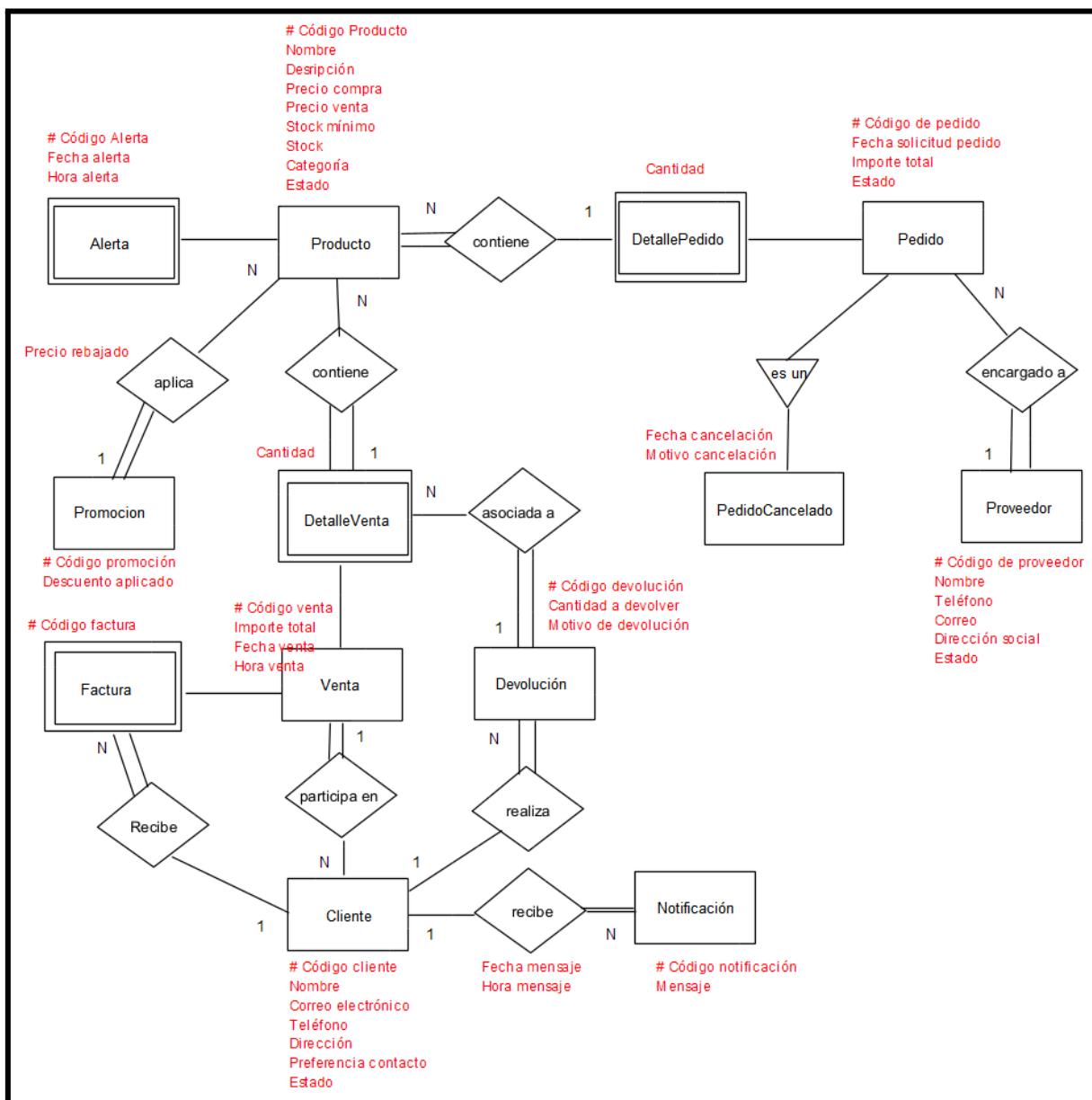
DFD1 (RF 5.4): Genera un informe detallado de los pedidos realizados en un intervalo de tiempo, incluyendo los productos comprados.



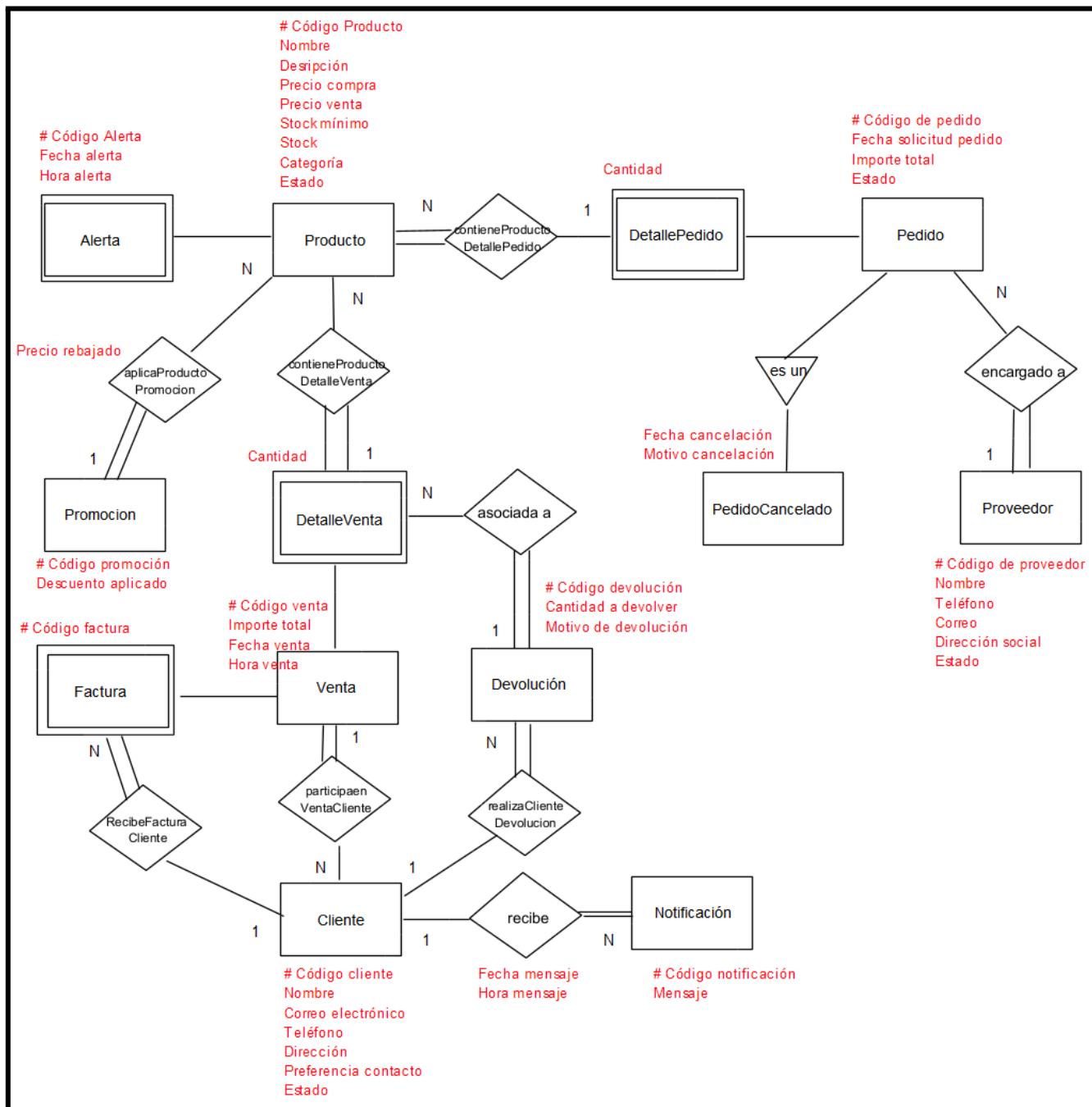
DFD1 (RF 5.5): Analiza las ventas agrupadas por familia de productos, evaluando el rendimiento de cada categoría.



Esquema E/R completo del sistema



Paso a tablas



Entidades

Alerta

- **codigo_alerta - codigo_producto [PK] [FK] → Producto**
- fecha_alerta
- hora_alerta

Producto

- **codigo_producto [PK]**
- nombre
- descripcion
- precio_compra
- precio_venta
- stock_minimo
- stock
- categoria
- estado

DetallePedido

- **codigo_pedido [PK] [FK] → Pedido**
- cantidad
- codigo_producto [FK] → Producto

Pedido

- **codigo_pedido [PK]**
- fecha_solicitud
- importe_total
- estado

PedidoCancelado

- **codigo_pedido [PK] [FK] → Pedido**
- fecha_cancelacion
- motivo_cancelacion

Proveedor

- **codigo_proveedor [PK]**
- nombre
- telefono
- correo
- direccion_social
- estado

Promocion

- **codigo_promocion [PK]**
- descuento_aplicado

DetalleVenta

- **codigo_venta [PK] [FK] → Venta**
- cantidad
- codigo_producto [FK] → Producto
- codigo_devolucion [FK] → Promocion

Venta

- **codigo_venta [PK]**
- importe_total
- fecha_venta
- hora_venta

Devolucion

- **codigo_devolucion [PK]**
- cantidad_devolver
- motivo_devolucion
- codigo_venta [FK] → Venta

Cliente

- **codigo_cliente [PK]**
- nombre
- correo_electronico
- telefono
- direccion
- preferencia_contacto
- estado

Notificacion

- **codigo_notificacion [PK]**
- mensaje
- fecha_hora
- codigo_cliente [FK] → Cliente

Factura

- **codigo_factura - codigo_venta [PK] [FK] → Venta**

Relaciones

ContieneProductoDetallePedido

- **codigo_producto [PK] [FK]** → **Producto**
- **codigo_pedido [FK]** → **Pedido**

Encargado a

- **codigo_pedido [PK] [FK]** → **Pedido**
- **codigo_proveedor [FK]** → **Proveedor**

AplicaProductoPromocion

- **codigo_producto [PK] [FK]** → **Producto**
- **codigo_promocion [FK]** → **Promocion**

ContieneProductoDetalleVenta

- **codigo_producto [PK] [FK]** → **Producto**
- **codigo_venta [FK]** → **DetalleVenta**

Asociada a

- **codigo_venta [PK] [FK]** → **DetalleVenta**
- **codigo_devolucion [FK]** → **Devolucion**

RecibeFacturaCliente

- **codigo_factura [PK] [FK]** → **Factura**
- **codigo_cliente [FK]** → **Cliente**

ParticipaenVentaCliente

- **codigo_cliente [PK] [FK]** → **Cliente**
- **codigo_venta [FK]** → **Venta**

RealizaClienteDevolucion

- **codigo_devolucion [PK] [FK]** → **Devolucion**
- **codigo_cliente [FK]** → **Cliente**

RecibeClienteNotificacion

- **codigo_notificacion [PK] [FK]** → **Notificacion**
- **codigo_cliente [FK]** → **Cliente**

Dependencias Funcionales

Entidades:

- **Alerta:**
 - **codigo_alerta** → fecha_alerta, hora_alerta, codigo_producto
- **Producto:**
 - **codigo_producto** → nombre, descripcion, precio_compra, precio_venta, stock_minimo, stock, categoria, estado
- **DetallePedido:**
 - **{codigo_pedido, codigo_producto}** → cantidad
- **Pedido:**
 - **codigo_pedido** → fecha_solicitud, importe_total, estado
- **PedidoCancelado:**
 - **codigo_pedido** → fecha_cancelacion, motivo_cancelacion
- **Proveedor:**
 - **codigo_proveedor** → nombre, telefono, correo, direccion_social, estado
- **Promocion:**
 - **codigo_promocion** → descuento_aplicado
- **DetalleVenta:**
 - **{codigo_venta, codigo_producto}** → cantidad
- **Venta:**
 - **codigo_venta** → importe_total, fecha_venta, hora_venta
- **Devolucion:**
 - **codigo_devolucion** → cantidad_devolver, motivo_devolucion, codigo_venta
- **Cliente:**
 - **codigo_cliente** → nombre, correo_electronico, telefono, direccion, preferencia_contacto, estado
- **Notificacion:**
 - **codigo_notificacion** → mensaje, fecha_hora, codigo_cliente

- **Factura:**

- **codigo_factura** → **codigo_venta**

Relaciones:

- **ContieneProdDetalleVenta:**

- **{codigo_producto, codigo_pedido}** → \emptyset

- **Encargado a:**

- **codigo_pedido** → **codigo_proveedor**

- **AplicaProductoPromocion:**

- **{codigo_producto, codigo_promocion}** → \emptyset

- **ContieneProdudctoDetalleVenta:**

- **{codigo_producto, codigo_detalle_venta}** → \emptyset

- **Asociada a:**

- **codigo_detalle_venta** → **codigo_devolucion**

- **RecibeFacturaCliente:**

- **codigo_factura** → **codigo_cliente**

- **ParticipaenVentaCliente:**

- **{codigo_cliente, codigo_venta}** → \emptyset

- **RealizaClienteDevolucion:**

- **codigo_devolucion** → **codigo_cliente**

- **RecibeClienteNotificacion:** **codigo_notificacion** → **codigo_cliente**

Tablas normalizadas

Entidades

Alerta

- **codigo_alerta - codigo_producto [PK] [FK] → Producto**
- fecha_alerta
- hora_alerta

Producto

- **codigo_producto [PK]**
- nombre
- descripcion
- precio_compra
- precio_venta
- stock_minimo
- stock
- categoria
- estado

DetallePedido

- **codigo_pedido [PK] [FK] → Pedido**
- cantidad
- codigo_producto [FK] → Producto

Pedido

- **codigo_pedido [PK]**
- fecha_solicitud
- importe_total
- estado

PedidoCancelado

- **codigo_pedido [PK] [FK] → Pedido**
- fecha_cancelacion
- motivo_cancelacion

Proveedor

- **codigo_proveedor [PK]**
- nombre
- telefono
- correo
- direccion_social
- estado

Promocion

- **codigo_promocion [PK]**
- descuento_aplicado

DetalleVenta

- **codigo_venta [PK] [FK] → Venta**
- cantidad
- codigo_producto [FK] → Producto
- codigo_devolucion [FK] → Promocion

Venta

- **codigo_venta [PK]**
- importe_total
- fecha_venta
- hora_venta

Devolucion

- **codigo_devolucion [PK]**
- cantidad_devolver
- motivo_devolucion
- codigo_venta [FK] → Venta

Cliente

- **codigo_cliente [PK]**
- nombre
- correo_electronico
- telefono
- direccion
- preferencia_contacto
- estado

Notificacion

- **codigo_notificacion [PK]**
- mensaje
- fecha_hora
- codigo_cliente [FK] → Cliente

Factura

- **codigo_factura - codigo_venta [PK] [FK] → Venta**

Relaciones

ContieneProductoDetallePedido

- **codigo_producto [PK] [FK]** → Producto
- **codigo_pedido [FK]** → Pedido

Encargado a

- **codigo_pedido [PK] [FK]** → Pedido
- **codigo_proveedor [FK]** → Proveedor

AplicaProductoPromocion

- **codigo_producto [PK] [FK]** → Producto
- **codigo_promocion [FK]** → Promocion

ContieneProductoDetalleVenta

- **codigo_producto [PK] [FK]** → Producto
- **codigo_venta [FK]** → DetalleVenta

Asociada a

- **codigo_detalle_venta [PK] [FK]** → DetalleVenta
- **codigo_devolucion [FK]** → Devolucion

RecibeFacturaCliente

- **codigo_factura [PK] [FK]** → Factura
- **codigo_cliente [FK]** → Cliente

ParticipaenVentaCliente

- **codigo_cliente [PK] [FK]** → Cliente
- **codigo_venta [FK]** → Venta

RealizaClienteDevolucion

- **codigo_devolucion [PK] [FK]** → Devolucion
- **codigo_cliente [FK]** → Cliente

RecibeClienteNotificacion

- **codigo_notificacion [PK] [FK]** → Notificacion
- **codigo_cliente [FK]** → Cliente

Primera Forma Normal (1FN)

Para que nuestras tablas se encuentren en la **1FN**, deben cumplir con las siguientes condiciones:

1. Los datos deben estar organizados en un formato tabular con filas y columnas.
2. Cada columna debe contener valores atómicos, es decir, no divisibles.
3. Cada fila debe ser única, identificada por una clave primaria.

Dado que todas nuestras tablas tienen claves primarias bien definidas y cada atributo contiene valores atómicos, podemos concluir que todas las tablas proporcionadas cumplen con la **primera forma normal (1FN)**.

Segunda Forma Normal (2FN)

Para que las tablas estén en **segunda forma normal**, deben:

1. Estar en **1FN**.
2. Todos los atributos no primos deben depender completamente de la clave primaria.

A continuación, analizamos cada tabla:

- **Alerta:** La clave primaria es codigo_alerta - codigo_producto. Los atributos fecha_alerta y hora_alerta dependen completamente de la clave primaria. No hay dependencias parciales. **Está en 2FN.**
- **Producto:** La clave primaria es codigo_producto. Todos los atributos (nombre, descripcion, precio_compra, precio_venta, etc.) dependen completamente de codigo_producto. **Está en 2FN.**
- **DetallePedido:** La clave primaria es codigo_pedido. Los atributos cantidad y codigo_producto dependen completamente de la clave primaria. **Está en 2FN.**
- **Pedido:** La clave primaria es codigo_pedido. Todos los atributos (fecha_solicitud, importe_total, estado) dependen completamente de la clave primaria. **Está en 2FN.**
- **PedidoCancelado:** La clave primaria es codigo_pedido. Los atributos fecha_cancelacion y motivo_cancelacion dependen completamente de la clave primaria. **Está en 2FN.**
- **Proveedor:** La clave primaria es codigo_proveedor. Todos los atributos (nombre, telefono, correo, direccion_social, estado) dependen completamente de la clave primaria. **Está en 2FN.**
- **Promocion:** La clave primaria es codigo_promocion. El atributo descuento_aplicado depende completamente de la clave primaria. **Está en 2FN.**
- **DetalleVenta:** La clave primaria es codigo_venta. Los atributos cantidad, codigo_producto, y codigo_devolucion dependen completamente de la clave primaria. **Está en 2FN.**
- **Venta:** La clave primaria es codigo_venta. Todos los atributos (importe_total, fecha_venta, hora_venta) dependen completamente de la clave primaria. **Está en 2FN.**

- **Devolucion:** La clave primaria es codigo_devolucion. Los atributos cantidad_devolver, motivo_devolucion y codigo_venta dependen completamente de la clave primaria. **Está en 2FN.**
- **Cliente:** La clave primaria es codigo_cliente. Todos los atributos (nombre, correo_electronico, telefono, direccion, preferencia_contacto, estado) dependen completamente de la clave primaria. **Está en 2FN.**
- **Notificacion:** La clave primaria es codigo_notificacion. Los atributos mensaje, fecha_hora, y codigo_cliente dependen completamente de la clave primaria. **Está en 2FN.**
- **Factura:** La clave primaria es codigo_factura - codigo_venta. Los atributos dependen completamente de la clave primaria. **Está en 2FN.**
- **ContieneProductoDetallePedido:** La clave primaria es codigo_producto - codigo_pedido. Los atributos dependen completamente de la clave primaria. **Está en 2FN.**
- **Encargado a:** La clave primaria es codigo_pedido - codigo_proveedor. Los atributos dependen completamente de la clave primaria. **Está en 2FN.**
- **AplicaProductoPromocion:** La clave primaria es codigo_producto - codigo_promocion. Los atributos dependen completamente de la clave primaria. **Está en 2FN.**
- **ContieneProductoDetalleVenta:** La clave primaria es codigo_producto - codigo_detalle_venta. Los atributos dependen completamente de la clave primaria. **Está en 2FN.**
- **Asociada a:** La clave primaria es codigo_detalle_venta - codigo_devolucion. Los atributos dependen completamente de la clave primaria. **Está en 2FN.**
- **RecibeFacturaCliente:** La clave primaria es codigo_factura - codigo_cliente. Los atributos dependen completamente de la clave primaria. **Está en 2FN.**
- **ParticipaenVentaCliente:** La clave primaria es codigo_cliente - codigo_venta. Los atributos dependen completamente de la clave primaria. **Está en 2FN.**
- **RealizaClienteDevolucion:** La clave primaria es codigo_devolucion - codigo_cliente. Los atributos dependen completamente de la clave primaria. **Está en 2FN.**
- **RecibeClienteNotificacion:** La clave primaria es codigo_notificacion - codigo_cliente. Los atributos dependen completamente de la clave primaria. **Está en 2FN.**

Por lo tanto, todas las tablas están en **2FN**.

Tercera Forma Normal (3FN)

Para que las tablas estén en **tercera forma normal**, deben:

1. Estar en **2FN**.
2. No debe haber dependencias transitivas, es decir, ningún atributo no primo debe depender de otro atributo no primo.

Analizando cada tabla:

- **Alerta:** Todos los atributos (fecha_alerta, hora_alerta) dependen directamente de la clave primaria. No hay dependencias transitivas. **Está en 3FN.**
- **Producto:** No hay dependencias transitivas entre los atributos. **Está en 3FN.**
- **DetallePedido:** No hay dependencias transitivas. **Está en 3FN.**
- **Pedido:** No hay dependencias transitivas. **Está en 3FN.**
- **PedidoCancelado:** No hay dependencias transitivas. **Está en 3FN.**
- **Proveedor:** No hay dependencias transitivas. **Está en 3FN.**
- **Promocion:** No hay dependencias transitivas. **Está en 3FN.**
- **DetalleVenta:** No hay dependencias transitivas. **Está en 3FN.**
- **Venta:** No hay dependencias transitivas. **Está en 3FN.**
- **Devolucion:** No hay dependencias transitivas. **Está en 3FN.**
- **Cliente:** No hay dependencias transitivas. **Está en 3FN.**
- **Notificacion:** No hay dependencias transitivas. **Está en 3FN.**
- **Factura:** No hay dependencias transitivas. **Está en 3FN.**
- Las relaciones (**ContieneProductoDetallePedido**, **Encargado a**, etc.) no tienen dependencias transitivas. Todas están en **3FN**.

Por lo tanto, todas las tablas están en **3FN**.

Forma Normal de Boyce-Codd (BCNF)

Para que las tablas estén en **forma normal de Boyce-Codd (BCNF)**, todas las dependencias funcionales no triviales deben cumplir:

1. La parte izquierda de la dependencia (determinante) debe ser una superclave.

En todas las tablas y relaciones, las dependencias funcionales no triviales tienen como determinante una superclave. No se encuentran excepciones.

Por lo tanto, todas las tablas están en **forma normal de Boyce-Codd (BCNF)**.

Subsistema de Inventario

Sentencias de creación de tablas, con claves y restricciones

Tabla Producto

```
CREATE TABLE Producto (
    CódigoProducto VARCHAR(13) PRIMARY KEY,
    Nombre VARCHAR(100) NOT NULL,
    Descripción VARCHAR(500) NOT NULL,
    PrecioCompra DECIMAL(10, 3) NOT NULL CHECK (PrecioCompra ≥ 0),
    PrecioVenta DECIMAL(10, 2) NOT NULL CHECK (PrecioVenta ≥ 0),
    Stock INT NOT NULL CHECK (Stock ≥ 0),
    StockMínimo INT NOT NULL CHECK (StockMínimo ≥ 0),
    Categoría VARCHAR(50),
    Estado NUMBER(1) DEFAULT 1 CHECK (Estado IN(0,1))
);
```

Tabla Alerta

```
CREATE TABLE Alerta (
    CódigoAlerta INT PRIMARY KEY,
    CódigoProducto VARCHAR(13) NOT NULL,
    FechaHoraAlerta TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (CódigoProducto) REFERENCES Producto(CódigoProducto) ON DELETE CASCADE
);
```

Descripción de las transacciones

RF 1.1. Dar de alta un producto

SENTENCIA SQL

```
INSERT INTO Producto (CodigoProducto, Nombre, Descripcion, PrecioCompra, PrecioVenta,
Stock, StockMinimo, Categoria, Estado)
VALUES (<codigo>, <nombre>, <descripcion>, <precio_compra>, <precio_venta>, <stock>,
<stock_minimo>, <categoria>, <estado>);
```

Pido por teclado que se ingresen los campos de “codigoProducto” (El código de barras del producto que se lea con el lector de código de barras), “nombre”, “descripcion”, “precio_compra”, “precio_venta”, “stock”, “stock_minimo”, “categoria”, y “estado”. Una vez el usuario ha facilitado esta información nos conectamos a la base de datos y realizamos dicha sentencia INSERT a la tabla “Producto” para añadir este producto.

Código en Python

```
def dar_alta_producto():
    aceptado = ""

    codigo_producto = input("Ingrese el código del producto (máximo 13 caracteres): ").strip()
    nombre = input("Ingrese el nombre del producto: ").strip()
    descripcion = input("Ingrese una descripción para el producto: ").strip()

    precio_compra = -1
    while precio_compra < 0:
        precio_compra = float(input("Ingrese el precio del producto: "))
        if precio_compra < 0:
            print("El precio no puede ser negativo.")

    precio_venta = -1
    while precio_venta < 0:
        precio_venta = float(input("Ingrese el precio de venta del producto: "))
        if precio_venta < 0:
            print("El precio de venta no puede ser negativo.")

    stock = -1
    while stock < 0:
        stock = int(input("Ingrese el stock inicial del producto: "))
        if stock < 0:
            print("El stock no puede ser negativo.")

    stock_minimo = -1
    while stock_minimo < 0:
        stock_minimo = int(input("Ingrese el stock mínimo permitido para el producto: "))
        if stock_minimo < 0:
            print("El stock mínimo no puede ser negativo.")

    categoria = input("Ingrese la categoría del producto (opcional): ").strip()

    try:
        conn, cursor = conectar_bd()
        cursor.execute("""
            INSERT INTO Producto (CodigoProducto, Nombre, Descripcion, PrecioCompra, PrecioVenta, Stock, StockMinimo, Categoria)
            VALUES (:codigo_producto, :nombre, :descripcion, :precio_compra, :precio_venta, :stock, :stock_minimo, :categoria)
        """, [codigo_producto, nombre, descripcion, precio_compra, precio_venta, stock, stock_minimo, categoria])
        conn.commit()
        print(f"Producto '{nombre}' añadido exitosamente.")
    except oracledb.DatabaseError as e:
        conn.rollback()
        print(f"Error al añadir producto: {e}")
    finally:
        cursor.close()
        conn.close()
```

RF 1.2. Modificar el stock de un producto

SENTENCIA SQL

```
UPDATE Producto
SET Stock = Stock + <incremento>
WHERE CódigoProducto = <codigo_producto>;
```

Dado el código del producto de un artículo ya existente, tomamos el valor numérico que queremos modificar y modificamos el valor de dicho nuevo incremento en el valor final del stock del producto inicialmente referenciado.

Código en Python

```
def modificar_stock_producto():
    código_producto = input("Ingrese el código del producto que desea modificar: ").strip()

    nuevo_stock = -1
    while nuevo_stock < 0:
        nuevo_stock = int(input("Ingrese el nuevo valor de stock para este producto: "))
        if nuevo_stock < 0:
            print("El stock no puede ser negativo.")

    try:
        conn, cursor = conectar_bd()

        cursor.execute("SELECT COUNT(*) FROM Producto WHERE CódigoProducto = :código_producto", {"código_producto": código_producto})
        producto_existe = cursor.fetchone()[0]

        if producto_existe == 0:
            print(f"No se encontró ningún producto con el código: {código_producto}.")
            return

        cursor.execute("""
            UPDATE Producto
            SET Stock = :nuevo_stock
            WHERE CódigoProducto = :código_producto
        """, {
            "nuevo_stock": nuevo_stock,
            "código_producto": código_producto
        })
        conn.commit()
        print(f"El stock del producto con código {código_producto} ha sido actualizado exitosamente a {nuevo_stock}.")
    except oracledb.DatabaseError as e:
        conn.rollback()
        print(f"Error al modificar el stock del producto: {e}")
    finally:
        cursor.close()
        conn.close()
```

RF 1.3. Consultar el stock de un producto

SENTENCIA SQL

```
SELECT CódigoProducto, Nombre, Stock
FROM Producto
WHERE CódigoProducto = <codigo_producto>;
```

Esta consulta SQL obtiene el código, nombre y stock de un producto específico filtrando por su CódigoProducto. Es útil para verificar la existencia y disponibilidad en inventario. Si el código no existe, no devuelve resultados.

Código en Python

```
def consultar_stock_producto():
    código_producto = input("Ingrese el código del producto que desea consultar: ").strip()

    try:
        conn, cursor = conectar_bd()

        cursor.execute("""
            SELECT Nombre, Stock
            FROM Producto
            WHERE CódigoProducto = :código_producto
        """, {"código_producto": código_producto})

        producto = cursor.fetchone()

        if producto:
            nombre, stock = producto
            print(f"El producto '{nombre}' tiene un stock actual de {stock} unidades.")
        else:
            print(f"No se encontró ningún producto con el código: {código_producto}.")
    except oracledb.DatabaseError as e:
        print(f"Error al consultar el stock del producto: {e}")
    finally:
        cursor.close()
        conn.close()
```

RF 1.4. Consultar Alerta Producto

SENTENCIA SQL

```
-- Para las alertas asociadas al producto
SELECT CódigoAlerta, FechaHoraAlerta
FROM Alerta
WHERE CódigoProducto = <codigo_producto>
ORDER BY FechaHoraAlerta DESC;
```

Recupera las alertas asociadas a un producto específico, identificadas por su CódigoProducto, desde la tabla Alerta. Devuelve dos columnas: el código de la alerta (CódigoAlerta) y la fecha y hora en que se generó (FechaHoraAlerta). Los resultados se ordenan en orden descendente por fecha y hora, mostrando primero las alertas más recientes. Es útil para realizar un seguimiento cronológico de las alertas de un producto.

Código en Python

```
def consultar_alerta_producto():

    código_producto = input("Ingrese el código del producto para consultar las alertas: ").strip()

    try:
        conn, cursor = conectar_bd()

        cursor.execute("SELECT Nombre FROM Producto WHERE CódigoProducto = :código_producto", {"código_producto": código_producto})
        producto = cursor.fetchone()

        if not producto:
            print(f"No se encontró ningún producto con el código: {código_producto}.")
            return

        cursor.execute("""
            SELECT CódigoAlerta, FechaHoraAlerta
            FROM Alerta
            WHERE CódigoProducto = :código_producto
            ORDER BY FechaHoraAlerta DESC
        """, {"código_producto": código_producto})

        alertas = cursor.fetchall()

        if alertas:
            print(f"Alertas para el producto '{producto[0]}' (Código: {código_producto}):")
            for alerta in alertas:
                print(f" - Alerta ID: {alerta[0]}, Fecha y hora: {alerta[1]}")
        else:
            print(f"No hay alertas registradas para el producto '{producto[0]}' (Código: {código_producto}).")

    except oracledb.DatabaseError as e:
        print(f"Error al consultar las alertas del producto: {e}")
    finally:
        cursor.close()
        conn.close()
```

RF 1.5. Dar de baja un producto (Descatalogar)

SENTENCIA SQL

```
UPDATE Producto
SET Estado = 0
WHERE CódigoProducto = <codigo_producto>;
```

Actualiza el estado de un producto específico en la tabla Producto, identificándose por su CódigoProducto. Cambia el valor del campo Estado a 0, indicando que el producto ha sido descatalogado o desactivado. Es útil para gestionar la disponibilidad de productos en el inventario sin eliminarlos de la base de datos.

Código en Python

```
def dar_de_baja_producto():
    código_producto = input("Ingrese el código del producto que desea dar de baja: ").strip()
    try:
        conn, cursor = conectar_bd()

        cursor.execute("SELECT Nombre, Estado FROM Producto WHERE CódigoProducto = :código_producto", {"código_producto": código_producto})
        producto = cursor.fetchone()

        if not producto:
            print(f"No se encontró ningún producto con el código: {código_producto}.")
            return

        nombre_producto, estado_actual = producto

        if estado_actual == 0:
            print(f"El producto '{nombre_producto}' (Código: {código_producto}) ya está descatalogado.")
            return

        cursor.execute("""
            UPDATE Producto
            SET Estado = 0
            WHERE CódigoProducto = :código_producto
        """, {"código_producto": código_producto})
        conn.commit()

        print(f"El producto '{nombre_producto}' (Código: {código_producto}) ha sido dado de baja exitosamente.")
    except oracledb.DatabaseError as e:
        conn.rollback()
        print(f"Error al dar de baja el producto: {e}")
    finally:
        cursor.close()
        conn.close()
```

Código Disparadores

Disparador para generar Alerta Automática por Stock Bajo

```
CREATE OR REPLACE TRIGGER generar_alerta_bajo_stock
AFTER UPDATE ON Producto
FOR EACH ROW
DECLARE
    v_codigo_alerta VARCHAR2(10);
BEGIN
    IF :NEW.Stock < :NEW.StockMinimo THEN
        SELECT CONCAT('N', LPAD(SEQ_ALERTA.NEXTVAL, 9, '0'))
        INTO v_codigo_alerta
        FROM DUAL;

        INSERT INTO Alerta (CodigoAlerta, CodigoProducto)
        VALUES (v_codigo_alerta, :NEW.CodigoProducto);
    END IF;
END;
```

Este disparador se activa automáticamente después de una actualización en la tabla Producto. Su propósito es generar una alerta en la tabla Alerta si el valor del Stock del producto actualizado cae por debajo de su StockMinimo. Esto asegura un seguimiento proactivo del inventario para evitar desabastecimientos.

Disparador para validar Precio de Venta contra Precio de Compra

```
CREATE OR REPLACE TRIGGER validar_precio_venta
BEFORE INSERT OR UPDATE OF PrecioVenta
ON Producto
FOR EACH ROW
BEGIN
    IF :NEW.PrecioVenta < :NEW.PrecioCompra THEN
        RAISE_APPLICATION_ERROR(-20001, 'El precio de venta no puede ser inferior al
precio de compra.');
    END IF;
END;
```

Este disparador se ejecuta antes de realizar una inserción o actualización en el campo PrecioVenta de la tabla Producto. Su función es garantizar que el precio de venta nunca sea inferior al precio de compra, preservando la rentabilidad del producto. Si esta condición no se cumple, se genera un error con un mensaje explicativo, impidiendo la operación.

Datos sensibles y Descripción de los aspectos legales

El subsistema de inventario desarrollado no maneja datos sensibles, ya que toda la información procesada está relacionada exclusivamente con la gestión de productos, tales como códigos de identificación, nombres, precios, cantidades en stock, categorías, y alertas de inventario. Estos datos son de naturaleza comercial y administrativa, y no incluyen información personal, confidencial, o sensible, como se define en normativas como el Reglamento General de Protección de Datos.

Subsistema de Ventas

Sentencias de creación de tablas, con claves y restricciones

Tabla Venta

```
CREATE TABLE Venta (
    CodigoVenta VARCHAR(13) PRIMARY KEY,
    FechaHoraVenta TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    EstadoVenta VARCHAR(10) CHECK (EstadoVenta IN ('Finalizado', 'Devuelto')),
    CodigoCliente VARCHAR(15) NOT NULL,
    FOREIGN KEY (CodigoCliente) REFERENCES Cliente(CodigoCliente) ON DELETE CASCADE
);
```

Tabla DetalleVenta

```
CREATE TABLE DetalleVenta (
    CodigoDetalleVenta VARCHAR(13) PRIMARY KEY,
    CodigoVenta VARCHAR(13) NOT NULL,
    CodigoProducto VARCHAR(13) NOT NULL,
    Cantidad INT NOT NULL CHECK (Cantidad > 0),
    PrecioVentaFinal DECIMAL(10, 2) NOT NULL CHECK (PrecioVentaFinal ≥ 0),
    FOREIGN KEY (CodigoVenta) REFERENCES Venta(CodigoVenta) ON DELETE CASCADE,
    FOREIGN KEY (CodigoProducto) REFERENCES Producto(CodigoProducto) ON DELETE CASCADE
);
```

Descripción de las transacciones

RF 2.1 Registrar una venta

Sentencia SQL:

```
INSERT INTO Venta (CodigoVenta, FechaHoraVenta, EstadoVenta, CódigoCliente)
VALUES (:codigo_venta, CURRENT_TIMESTAMP, 'Finalizado', :codigo_cliente);
```

Pido por teclado el código del cliente, genero automáticamente un código de venta único y registro la venta con estado "Finalizado". A continuación, solicito los productos asociados a la venta y, por cada uno, registro sus detalles en la tabla DetalleVenta con su cantidad y precio final.

```
def registrar_venta():
    """
    Solicita los datos para registrar una nueva venta en la base de datos.
    """
    try:
        conn, cursor = conectar_bd()
        if not conn or not cursor:
            return

        # Generar código de venta
        cursor.execute("SELECT SEQ_VENTA.NEXTVAL FROM DUAL")
        venta_seq = cursor.fetchone()[0]
        codigo_venta = f"V{venta_seq:012}"
        fecha_hora = datetime.now().strftime('%Y-%m-%d %H:%M:%S')

        # Solicitar el cliente
        codigo_cliente = input("Ingrese el código del cliente: ")

        # Insertar la venta
        cursor.execute("""
            INSERT INTO Venta (CodigoVenta, FechaHoraVenta, EstadoVenta, CódigoCliente)
            VALUES (:1, TO_TIMESTAMP(:2, 'YYYY-MM-DD HH24:MI:SS'), 'Finalizado', :3)
        """, [codigo_venta, fecha_hora, codigo_cliente])

        # Registrar detalles de la venta
        while True:
            codigo_producto = input("Código del producto (o 'fin' para terminar): ")
            if codigo_producto.lower() == 'fin':
                break

            cantidad = int(input("Cantidad a vender: "))

            cursor.execute("""
                SELECT PrecioVenta FROM Producto WHERE CódigoProducto = :codigo_producto
            """, [codigo_producto])
            precio_venta = cursor.fetchone()[0]

            # Generar un código único para el detalle
            cursor.execute("SELECT SEQ_DETALLEVENTA.NEXTVAL FROM DUAL")
            detalle_seq = cursor.fetchone()[0]
            codigo_detalle = f"D{detalle_seq:012}" # Prefijo "D" con un número de 12 dígitos (total 13)
            cursor.execute("""
                INSERT INTO DetalleVenta (CódigoDetalleVenta, CódigoVenta, CódigoProducto, PrecioVentaFinal, Cantidad)
                VALUES (:codigo_detalle, :codigo_venta, :codigo_producto, :precio_venta, :cantidad)
            """, [codigo_detalle, codigo_venta, codigo_producto, precio_venta, cantidad])

        conn.commit()
        print(f"Venta {codigo_venta} registrada correctamente.")
    except Exception as e:
        if conn:
            conn.rollback()
        print("Error registrando la venta:", e)
    finally:
        if cursor:
            cursor.close()
        if conn:
            conn.close()
```

RF2.2: Generar recibo o factura

Sentencia SQL:

```
INSERT INTO Factura (CodigoFactura, CodigoVenta, FechaHoraVenta)
VALUES (:codigo_factura, :codigo_venta, :fecha_hora);
```

Solicito al usuario el código de la venta y verifico su existencia. Si es válida, genero un código único para la factura y registro los datos en la tabla Factura. Esta acción genera una relación directa entre la venta y su factura.

```
def generar_factura():
    """
    Genera una factura para una venta específica.
    """
    try:
        conn, cursor = conectar_bd()
        if not conn or not cursor:
            print("No se pudo establecer conexión con la base de datos.")
            return

        # Mostrar las ventas disponibles
        print("\nVentas disponibles:")
        cursor.execute("SELECT CodigoVenta, FechaHoraVenta, EstadoVenta FROM Venta")
        ventas = cursor.fetchall()
        if ventas:
            for v in ventas:
                print(f"Código: {v[0]}, Fecha: {v[1]}, Estado: {v[2]}")
        else:
            print("No hay ventas registradas.")
            return

        # Solicitar el código de la venta
        codigo_venta = input("Ingrese el código de la venta para generar la factura: ")

        # Verificar que la venta exista
        cursor.execute("SELECT * FROM Venta WHERE CodigoVenta = :codigo_venta", [codigo_venta])
        venta = cursor.fetchone()
        if not venta:
            print(f"No se encontró la venta con código {codigo_venta}.")
            return

        # Generar el código de la factura
        codigo_factura = f"F_{codigo_venta}"

        # Insertar en la tabla Factura
        cursor.execute(
            """
            INSERT INTO Factura (CodigoFactura, CodigoVenta, FechaHoraVenta)
            VALUES (:codigo_factura, :codigo_venta, :fecha_hora)
            """,
            {
                'codigo_factura': codigo_factura,
                'codigo_venta': codigo_venta,
                'fecha_hora': venta[1] # Fecha y hora de la venta
            }
        )

        conn.commit()
        print(f"Factura generada exitosamente con el código {codigo_factura}.")

    except Exception as e:
        if conn:
            conn.rollback()
        print("Error al generar la factura:", e)

    finally:
        if cursor:
            cursor.close()
        if conn:
            conn.close()
```

RF2.3: Aplicar descuentos o promociones

Sentencia SQL:

```
INSERT INTO Promocion (CodigoPromocion, DescuentoAplicado)
VALUES (:codigo_promocion, :descuento);
```

```
INSERT INTO ProductoPromocion (CodigoProducto, CodigoPromocion)
VALUES (:codigo_producto, :codigo_promocion);
```

```
UPDATE Producto
SET PrecioVenta = PrecioVenta * (1 - :descuento)
WHERE CodigoProducto = :codigo_producto;
```

Solicito un código de producto y un porcentaje de descuento, verificando que esté dentro del rango permitido (0-1). Posteriormente, registro la promoción en la tabla correspondiente y actualizo el precio de venta del producto

```
def aplicar_promocion():
    """
    Aplica una promoción a un producto y actualiza su precio.
    """
    try:
        conn, cursor = conectar_bd()
        if not conn or not cursor:
            print("No se pudo establecer conexión con la base de datos.")
            return

        # Solicitar datos de la promoción
        codigo_producto = input("Ingrese el código del producto: ")
        cursor.execute("SELECT * FROM Producto WHERE CodigoProducto = :codigo_producto", [codigo_producto])
        producto = cursor.fetchone()
        if not producto:
            print(f"No se encontró el producto con código {codigo_producto}.")
            return

        codigo_promocion = input("Ingrese el código de la promoción: ")
        descuento = float(input("Ingrese el porcentaje de descuento (entre 0 y 1): "))

        if not (0 <= descuento <= 1):
            print("El descuento debe estar entre 0 y 1.")
            return

        # Insertar en la tabla Promoción
        cursor.execute(
            """
            INSERT INTO Promocion (CodigoPromocion, DescuentoAplicado)
            VALUES (:codigo_promocion, :descuento)
            """,
            {
                'codigo_promocion': codigo_promocion,
                'descuento': descuento
            }
        )
```

```
# Insertar en ProductoPromocion
cursor.execute(
    """
    INSERT INTO ProductoPromocion (CodigoProducto, CodigoPromocion)
    VALUES (:codigo_producto, :codigo_promocion)
    """
)

# Actualizar precio del producto
nuevo_precio = producto[4] * (1 - descuento) # producto[4] es el precio actual
cursor.execute(
    """
    UPDATE Producto
    SET PrecioVenta = :nuevo_precio
    WHERE CodigoProducto = :codigo_producto
    """
)

conn.commit()
print(f"Promoción aplicada. Nuevo precio de {codigo_producto}: {nuevo_precio:.2f}")

except Exception as e:
    if conn:
        conn.rollback()
        print("Error al aplicar la promoción:", e)

finally:
    if cursor:
        cursor.close()
    if conn:
        conn.close()
```

RF2.4: Consultar historial de ventas de productos.

Sentencia SQL:

```
SELECT V.CodigoVenta, V.FechaHoraVenta, V.EstadoVenta,
       C.Nombre AS Cliente, DV.CodigoProducto, DV.Cantidad, DV.PrecioVentaFinal
  FROM Venta V
 JOIN Cliente C ON V.CodigoCliente = C.CodigoCliente
 JOIN DetalleVenta DV ON V.CodigoVenta = DV.CodigoVenta;
```

Muestro un listado completo de las transacciones realizadas, incluyendo los datos de las ventas, los detalles de los productos asociados y el cliente correspondiente.

```
def consultar_transacciones():
    """
    Muestra el registro de todas las transacciones realizadas.
    """
    try:
        conn, cursor = conectar_bd()
        if not conn or not cursor:
            print("No se pudo establecer conexión con la base de datos.")
            return

        # Consultar las transacciones
        cursor.execute(
            """
            SELECT V.CodigoVenta, V.FechaHoraVenta, V.EstadoVenta,
                   C.Nombre AS Cliente, DV.CodigoProducto, DV.Cantidad, DV.PrecioVentaFinal
            FROM Venta V
            JOIN Cliente C ON V.CodigoCliente = C.CodigoCliente
            JOIN DetalleVenta DV ON V.CodigoVenta = DV.CodigoVenta
            """
        )
        transacciones = cursor.fetchall()
        if not transacciones:
            print("No se encontraron transacciones registradas.")
            return

        # Mostrar las transacciones
        print("\nRegistro de Transacciones de Ventas:")
        print("-----")
        for t in transacciones:
            print(f"Código Venta: {t[0]}, Fecha y Hora: {t[1]}, Estado: {t[2]}")
            print(f"Cliente: {t[3]}, Producto: {t[4]}, Cantidad: {t[5]}, Precio Final: {t[6]:.2f}")
            print("-----")

    except Exception as e:
        print("Error al consultar transacciones:", e)

    finally:
        if cursor:
            cursor.close()
        if conn:
            conn.close()
```

RF2.5: Gestionar una devolución

Sentencia SQL:

```
UPDATE Venta
SET EstadoVenta = 'Devuelto'
WHERE CodigoVenta = :codigo_venta;

UPDATE Producto
SET Stock = Stock + :cantidad
WHERE CodigoProducto = :codigo_producto;
```

Permito gestionar devoluciones solo de ventas con estado "Finalizado". Restauro el stock de los productos involucrados y actualizo el estado de la venta a "Devuelto". La lógica asegura que no puedan devolverse ventas previamente marcadas como "Devuelto".

```
def gestionar_devolucion():
    """
    Permite gestionar una devolución, marcando la venta como 'Devuelto' y restaurando el stock.
    """
    try:
        conn, cursor = conectar_bd()
        if not conn or not cursor:
            print("No se pudo establecer conexión con la base de datos.")
            return

        # Solicitar código de la venta
        codigo_venta = input("Ingrese el código de la venta para gestionar la devolución: ")

        # Verificar que la venta exista
        cursor.execute("SELECT * FROM Venta WHERE CodigoVenta = :codigo_venta", [codigo_venta])
        venta = cursor.fetchone()
        if not venta:
            print(f"No se encontró la venta con código {codigo_venta}.")
            return

        # Actualizar el estado de la venta
        cursor.execute(
            """
            UPDATE Venta
            SET EstadoVenta = 'Devuelto'
            WHERE CodigoVenta = :codigo_venta
            """,
            {'codigo_venta': codigo_venta}
        )
```

```
# Restaurar stock de los productos
cursor.execute(
    """
    SELECT CodigoProducto, Cantidad
    FROM DetalleVenta
    WHERE CodigoVenta = :codigo_venta
    """,
    {'codigo_venta': codigo_venta}
)
detalles = cursor.fetchall()

for d in detalles:
    cursor.execute(
        """
        UPDATE Producto
        SET Stock = Stock + :cantidad
        WHERE CodigoProducto = :codigo_producto
        """,
        {
            'cantidad': d[1],
            'codigo_producto': d[0]
        }
    )

conn.commit()
print(f"Devolución procesada correctamente para la venta {codigo_venta}.")

except Exception as e:
    if conn:
        conn.rollback()
        print("Error al gestionar devolución:", e)

finally:
    if cursor:
        cursor.close()
    if conn:
        conn.close()
```

Código Disparadores

Actualizar el estado de una venta al realizar una devolución.

```

CREATE OR REPLACE TRIGGER ActualizarEstadoVenta
AFTER INSERT ON DetalleVenta
FOR EACH ROW
DECLARE
    TotalDevuelto INT;
BEGIN
    SELECT COUNT(*)
    INTO TotalDevuelto
    FROM DetalleVenta
    WHERE CodigoVenta = :NEW.CodigoVenta;
    IF TotalDevuelto > 0 THEN
        UPDATE Venta
        SET EstadoVenta = 'Devuelto'
        WHERE CodigoVenta = :NEW.CodigoVenta;
    END IF;
END;

```

El propósito de este disparador es garantizar que el estado de una venta solo pueda cambiar a "Devuelto" si cumple con las siguientes condiciones:

1. La venta debe haber estado previamente en el estado "Finalizado". Esto asegura que solo puedan devolverse ventas que ya han sido completadas y no ventas ya marcadas como "Devuelto".
2. La venta debe tener detalles asociados registrados en la tabla DetalleVenta, lo que significa que debe haber productos vinculados a esa venta para considerar la devolución válida.

El disparador evita cambios no válidos en el estado de la venta mediante el uso de una validación previa. Si alguna de estas condiciones no se cumple, el sistema cancela la operación y devuelve un mensaje de error, asegurando así la integridad de los datos y evitando inconsistencias en las devoluciones.

(Este disparador es fundamental para mantener la lógica de negocio relacionada con las devoluciones, asegurándose de que solo se devuelvan ventas válidas y previamente finalizadas).

Disparador que controla el stock de los productos cuando se realiza una inserción en DetalleVenta.

```
CREATE OR REPLACE TRIGGER ControlarStockMinimo
```

```
BEFORE INSERT ON DetalleVenta
FOR EACH ROW
DECLARE
    StockActual INT;
BEGIN
    SELECT Stock
    INTO StockActual
    FROM Producto
    WHERE CódigoProducto = :NEW.CódigoProducto;

    IF StockActual - :NEW.Cantidad < 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Error: Stock insuficiente para realizar la
venta.');
    ELSE
        UPDATE Producto
        SET Stock = Stock - :NEW.Cantidad
        WHERE CódigoProducto = :NEW.CódigoProducto;
    END IF;
END;
```

(Realizado por Alejandro Coman).

El propósito de este disparador es que al ir registrando ventas, no vendamos más productos de los que tenemos. Simplemente cuando insertamos una tupla en “DetalleVenta”, comprobamos si el stock actual del producto menos la cantidad que queremos insertar es positiva. Si es positiva es que aún hay stock disponible y entonces podemos disminuir el stock actual. En caso contrario, significa que no disponemos de stock suficiente por lo que cancelamos la operación.

Datos sensibles y Descripción de los aspectos legales

El subsistema de ventas no introduce nuevos datos sensibles, pero gestiona datos personales ya existentes en la base de datos. Se garantiza el cumplimiento del RGPD y la LOPDGDD, informando a los usuarios sobre sus derechos y asegurando la protección de la información manejada.

Subsistema de Clientes

Creación de tablas, con claves y restricciones

Tabla cliente

```
CREATE TABLE Cliente (
   CodigoCliente VARCHAR(15) PRIMARY KEY,
   Nombre VARCHAR(80) NOT NULL,
   Correo VARCHAR(50) NOT NULL,
   Telefono VARCHAR(20) NOT NULL,
   Direccion VARCHAR(100) NOT NULL,
   PreferenciaContacto NUMBER(1) DEFAULT 0 CHECK (PreferenciaContacto IN (0, 1)),
   Estado NUMBER(1) DEFAULT 1 CHECK (Estado IN (0, 1))
);
```

Tabla Notificación

```
CREATE TABLE Notificacion (
   CodigoNotificacion VARCHAR(10) PRIMARY KEY,
   Mensaje VARCHAR(300) NOT NULL
);
```

Tabla ClienteNotificacion

```
CREATE TABLE ClienteNotificacion (
   CodigoCliente VARCHAR(15) NOT NULL,
   CodigoNotificacion VARCHAR(10) NOT NULL,
   FechaHoraMensaje TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   PRIMARY KEY (CodigoCliente, CodigoNotificacion),
   FOREIGN KEY (CodigoCliente) REFERENCES Cliente(CodigoCliente) ON DELETE CASCADE,
   FOREIGN KEY (CodigoNotificacion) REFERENCES Notificacion(CodigoNotificacion) ON
DELETE CASCADE
);
```

Descripción de las transacciones

RF 3.1. Dar de alta un cliente

SENTENCIAS SQL

```
INSERT INTO Cliente (CodigoCliente, Nombre, Correo, Telefono, Direccion,
PreferenciaContacto) VALUES (<codigo>, <nombre>, <correo>, <telefono>, <direccion>,
<preferencia_contacto>)
```

Pido por teclado que se ingresen los campos de “código” (El DNI en este caso), “nombre”, “correo”, “teléfono”, “dirección”, y “preferencia_contacto”.

Una vez el usuario ha facilitado esta información nos conectamos a la base de datos y realizamos dicha sentencia INSERT a la tabla “Cliente” para añadir este cliente.

CÓDIGO EN PYTHON

```
def alta_cliente():
    aceptado = ""
    while aceptado != "S" and aceptado != "N":
        aceptado = input("A continuación se le solicitarán datos personales, algunos de los cuales son considerados sensibles. \
                         ¿Está seguro de que desea continuar? (S/N): ")
        if aceptado != "S" and aceptado != "N":
            print("Por favor, introduzca una opción válida.")

    if aceptado == "N":
        print("Operación cancelada.")
        return

    codigo = input("Ingrese el código del cliente: ")
    nombre = input("Ingrese el nombre del cliente: ")
    correo = input("Ingrese el correo electrónico del cliente: ")
    telefono = input("Ingrese el teléfono del cliente: ")
    direccion = input("Ingrese la dirección del cliente: ")
    preferencia_contacto = int(input("Ingrese la preferencia de contacto (0: Correo, 1: Teléfono):"))

    try:
        conn, cursor = conectar_bd()
        cursor.execute("""
            INSERT INTO Cliente (CodigoCliente, Nombre, Correo, Telefono, Direccion, PreferenciaContacto)
            VALUES (:codigo, :nombre, :correo, :telefono, :direccion, :preferencia_contacto)
        """, [codigo, nombre, correo, telefono, direccion, preferencia_contacto])
        conn.commit()
        print(f"Cliente {nombre} añadido exitosamente.")
    except oracledb.DatabaseError as e:
        conn.rollback()
        print(f"Error al añadir cliente: {e}")
    finally:
        cursor.close()
        conn.close()
```

RF 3.2 Consultar historial de ventas

SENTENCIA SQL

```
SELECT v.CodigoVenta, v.FechaHoraVenta FROM Venta v
WHERE v.CodigoCliente = <codigo_cliente>
```

```
SELECT dv.CodigoDetalleVenta, dv.CodigoProducto, dv.Cantidad, dv.PrecioVentaFinal,
p.Nombre FROM DetalleVenta dv JOIN Producto p ON dv.CodigoProducto = p.CodigoProducto
WHERE dv.CodigoVenta = <codigo_venta>
```

Pido por teclado el código de cliente para el cual queremos consultar su historial de ventas. Realizamos entonces la primera consulta donde conseguimos los códigos de venta de las ventas en las que este cliente participa. Para cada uno de los códigos de venta luego hacemos otra consulta en la tabla DetalleVenta para quedarnos con aquellos detalles que pertenezcan a dicha venta. Además hacemos una unión con la tabla de producto ya que para cada detalle de la venta imprimimos, aparte de la cantidad de dicho detalle, el nombre y el precio de dicho producto, que están contenidos en la tabla Producto.

CÓDIGO EN PYTHON

```
def consultar_historial_ventas():
    codigo_cliente = input("Ingrese el código del cliente: ")

    try:
        conn, cursor = conectar_bd()
        cursor.execute("""
            SELECT v.CodigoVenta, v.FechaHoraVenta
            FROM Venta v
            WHERE v.CodigoCliente = :codigo_cliente
        """, [codigo_cliente])
        ventas = cursor.fetchall()

        if not ventas:
            print("No hay ventas asociadas a este cliente.")
            return

        print(f"Historial de ventas para el cliente {codigo_cliente}:")
        for venta in ventas:
            print(f"\nVenta {venta[0]} realizada en la fecha y hora {venta[1]}")

            cursor.execute("""
                SELECT dv.CodigoDetalleVenta, dv.CodigoProducto, dv.Cantidad, dv.PrecioVentaFinal, p.Nombre
                FROM DetalleVenta dv
                JOIN Producto p ON dv.CodigoProducto = p.CodigoProducto
                WHERE dv.CodigoVenta = :codigo_venta
            """, [venta[0]])
            detalles = cursor.fetchall()

            for detalle in detalles:
                print(f"\tDetalle {detalle[0]}: \n\t{detalle[4]} ({detalle[1]}) \n\t - Importe: {detalle[3]}€ \n\t - Cantidad: {detalle[2]}")

    except oracledb.DatabaseError as e:
        print(f"Error al consultar historial: {e}")
    finally:
        cursor.close()
        conn.close()
```

RF 3.3. Editar datos del cliente.

SENTENCIA SQL

```
UPDATE Cliente SET Nombre = <nombre>, Correo = <correo>, Telefono = <telefono>,
Direccion = <direccion>, PreferenciaContacto = <preferencia_contacto> WHERE
CodigoCliente = <codigo>
```

Dado un código de cliente específico, podremos modificar su información, siguiendo esta transacción una estructura similar a la del RF 3.1 solo que mediante un UPDATE en lugar de un INSERT.

CÓDIGO EN PYTHON

```
def editar_cliente():
    aceptado = ""
    while aceptado != "S" and aceptado != "N":
        aceptado = input("A continuación se le solicitarán datos personales, algunos de los cuales son considerados sensibles. \
                         ¿Está seguro de que desea continuar? (S/N): ")
        if aceptado != "S" and aceptado != "N":
            print("Por favor, introduzca una opción válida.")

    if aceptado == "N":
        print("Operación cancelada.")
        return

    codigo = input("Ingrese el código del cliente a editar: ")
    nombre = input("Ingrese el nuevo nombre del cliente: ")
    correo = input("Ingrese el nuevo correo electrónico del cliente: ")
    telefono = input("Ingrese el nuevo teléfono del cliente: ")
    direccion = input("Ingrese la nueva dirección del cliente: ")
    preferencia_contacto = int(input("Ingrese la nueva preferencia de contacto (0: Correo, 1: Teléfono): "))

    try:
        conn, cursor = conectar_bd() # Asume que conectar_bd() devuelve conexión y cursor
        # Actualizar los datos del cliente
        cursor.execute("""
            UPDATE Cliente
            SET Nombre = :nombre, Correo = :correo, Telefono = :telefono,
                Direccion = :direccion, PreferenciaContacto = :preferencia_contacto
            WHERE CodigoCliente = :codigo
        """, [nombre, correo, telefono, direccion, preferencia_contacto, codigo])
        conn.commit()
        print(f"Cliente con código {codigo} actualizado exitosamente.")
    except oracledb.DatabaseError as e:
        conn.rollback()
        print(f"Error al editar cliente: {e}")
    finally:
        cursor.close()
        conn.close()
```

RF 3.4. Enviar notificación a cliente

SENTENCIA SQL

```

DECLARE v_codigo_notificacion VARCHAR2(10);
BEGIN
INSERT INTO Notificacion (Mensaje) VALUES (<mensaje>)
RETURNING CódigoNotificación INTO v_codigo_notificacion;
INSERT INTO ClienteNotificación (CódigoCliente, CódigoNotificación) VALUES
(<codigo_cliente>, v_codigo_notificacion);
END;

```

Pedimos un código de cliente y un mensaje para enviarlo a dicho cliente. Lo que haremos será primero insertar una tupla en la tabla Notificación para crear dicha notificación (el código de notificación es generado automáticamente siguiendo un formato específico “N<número>”, donde número es una secuencia que empieza en el 1 e incrementa de uno en uno por cada notificación que se crea y rellenando con ceros hasta que haya 9 dígitos (manejamos esto con un disparador para asegurar este formato). Una vez se ha insertado la notificación, nos quedamos con el código mediante RETURNING y lo usamos para insertar una tupla en ClienteNotificación.

CÓDIGO EN PYTHON

```

def enviar_notificacion():
    código_cliente = input("Ingrese el código del cliente: ")
    mensaje = input("Ingrese el mensaje para enviar al cliente: ")

    try:
        conn, cursor = conectar_bd()

        cursor.execute("""
            DECLARE
                v_codigo_notificacion VARCHAR2(10);
            BEGIN
                -- Insertar en Notificacion y obtener el código generado
                INSERT INTO Notificacion (Mensaje)
                VALUES (:mensaje)
                RETURNING CódigoNotificación INTO v_codigo_notificacion;

                -- Insertar en ClienteNotificación utilizando el código recuperado
                INSERT INTO ClienteNotificación (CódigoCliente, CódigoNotificación)
                VALUES (:código_cliente, v_codigo_notificacion);
            END;
        """, [mensaje, código_cliente])
        conn.commit()
        print(f"Notificación enviada al cliente {código_cliente}.")
    except oracledb.DatabaseError as e:
        conn.rollback()
        print(f"Error al enviar notificación: {e}")
    finally:
        cursor.close()
        conn.close()

```

RF 3.5. Dar de baja cliente

SENTENCIA SQL

```
UPDATE Cliente SET Estado = 0 WHERE CódigoCliente = <codigo_cliente>
```

Actualizamos el estado del cliente y lo ponemos a 0 para indicar que lo damos de baja (borrado lógico).

CÓDIGO EN PYTHON

```
def baja_cliente():
    codigo_cliente = input("Ingrese el código del cliente a dar de baja: ")
    try:
        conn, cursor = conectar_bd()
        cursor.execute("""
            UPDATE Cliente
            SET Estado = 0
            WHERE CódigoCliente = :codigo_cliente
        """, [codigo_cliente])
        conn.commit()
        print(f"Cliente {codigo_cliente} dado de baja exitosamente.")
    except oracledb.DatabaseError as e:
        conn.rollback()
        print(f"Error al dar de baja cliente: {e}")
    finally:
        cursor.close()
        conn.close()
```

Código Disparadores

Disparador para asegurar el formato del código de notificación.

```
CREATE OR REPLACE TRIGGER NumerarNotificacion
BEFORE INSERT ON Notificacion
FOR EACH ROW
BEGIN
    :NEW.CódigoNotificación := CONCAT('N', LPAD(SEQ_NOTIFICACION.NEXTVAL, 9, '0'));
END;
```

Cuando insertamos una tupla en la tabla de Notificación modificamos el valor del atributo “CódigoNotificación” para que siga el formato “N<Número>”, siendo número una secuencia que empieza por 1 y se incrementa de 1 en 1, rellenando con ceros hasta que tenga 9 dígitos.

Datos sensibles y Descripción de los aspectos legales

Este subsistema es de los que quizás más datos sensibles tiene pues estamos almacenando información como el DNI del cliente, cuál es su número de teléfono, dónde vive, etc... Más concretamente los datos sensibles del subsistema de Clientes son:

De la tabla Cliente:

- Código Cliente (DNI)
- Nombre
- Correo

- Teléfono
- Dirección

Es por ello que, en los requisitos funcionales 3.1 y 3.3 que es donde creamos y modificamos los datos del cliente respectivamente, mostramos en la interfaz un mensaje de advertencia al usuario, preguntando si desea continuar e introducir los datos. En caso de que el usuario indique que sí desea continuar, pediremos los datos. En caso contrario, cancelaremos la transacción.

Subsistema de Proveedores

Sentencias de creación de tablas, con claves y restricciones

Tabla Proveedor

```
CREATE TABLE Proveedor (
    CodigoProveedor VARCHAR(10) PRIMARY KEY,
    Nombre VARCHAR(40) NOT NULL,
    Telefono VARCHAR(15) NOT NULL,
    Correo VARCHAR(50) NOT NULL,
    DireccionSocial VARCHAR(100) NOT NULL,
    Estado NUMBER(1) DEFAULT 1 CHECK (Estado IN (0, 1))
```

);

Tabla Pedido

```
CREATE TABLE Pedido (
   CodigoPedido VARCHAR(20) PRIMARY KEY,
   FechaSolicitud TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
   EstadoPedido VARCHAR(15) DEFAULT 'Pendiente' CHECK (EstadoPedido IN ('Pendiente',
'Cancelado', 'Completado')),
   CodigoProveedor VARCHAR(10) NOT NULL,
    FOREIGN KEY (CodigoProveedor) REFERENCES Proveedor(CodigoProveedor) ON DELETE
CASCADE
);
```

Tabla PedidosCancelados

```
CREATE TABLE PedidosCancelados (
   CodigoCancelacion VARCHAR(20) PRIMARY KEY,
   CodigoPedido VARCHAR(20) NOT NULL,
   FechaCancelacion DATE NOT NULL,
   MotivoCancelacion VARCHAR(100),
    FOREIGN KEY (CodigoPedido) REFERENCES Pedido(CodigoPedido) ON DELETE CASCADE
);
```

Tabla DetallePedido

```
CREATE TABLE DetallePedido (
   CodigoDetallePedido VARCHAR(20) PRIMARY KEY,
   CodigoPedido VARCHAR(20),
   CodigoProducto VARCHAR(13) NOT NULL,
   Cantidad NUMBER(10) CHECK (Cantidad > 0),
    FOREIGN KEY (CodigoPedido) REFERENCES Pedido(CodigoPedido) ON DELETE CASCADE,
    FOREIGN KEY (CodigoProducto) REFERENCES Producto(CodigoProducto) ON DELETE CASCADE
);
```

Descripción de las transacciones

RF4.1: Dar de alta un proveedor.

SENTENCIA SQL

```
INSERT INTO Proveedor (CodigoProveedor, Nombre, Telefono, Correo, DireccionSocial,
Estado)
VALUES (:codigo, :nombre, :telefono, :correo, :direccion, 1)
```

Esta sentencia realiza la inserción de un nuevo registro en la tabla Proveedor con los valores proporcionados por el usuario.

CÓDIGO EN PYTHON

```

def alta_proveedor():
    # Solicitar datos del proveedor
    codigo = input("Ingrese el código del proveedor (10 caracteres): ")
    nombre = input("Ingrese el nombre del proveedor (40 caracteres): ")
    telefono = input("Ingrese el teléfono del proveedor (9-15 dígitos): ")
    correo = input("Ingrese el correo electrónico del proveedor: ")
    direccion = input("Ingrese la dirección social del proveedor (100 caracteres): ")

    try:
        # Conexión a la base de datos
        conn, cursor = conectar_bd()

        # Ejecución de la inserción
        cursor.execute("""
            INSERT INTO Proveedor (CodigoProveedor, Nombre, Telefono, Correo, DireccionSocial, Estado)
            VALUES (:codigo, :nombre, :telefono, :correo, :direccion, 1)
        """, {
            "codigo": codigo,
            "nombre": nombre,
            "telefono": telefono,
            "correo": correo,
            "direccion": direccion
        })

        # Confirmar los cambios
        conn.commit()
        print(f"Proveedor '{nombre}' añadido exitosamente.")

    except oracledb.DatabaseError as e:
        # Manejo de errores
        print(f"Error al añadir proveedor: {e}")

    finally:
        # Cerrar la conexión
        cursor.close()
        conn.close()

# Llamar a la función para probarla
alta_proveedor()

```

La función solicita al usuario los datos necesarios para registrar un nuevo proveedor (código, nombre, teléfono, correo y dirección), estableciendo conexión con la base de datos, y ejecutando la sentencia SQL para insertar estos datos en la tabla Proveedor.

RF4.2: Realizar un pedido a un proveedor.

SENTENCIA SQL 1

```

INSERT INTO Pedido (CodigoPedido, FechaSolicitud, EstadoPedido, CodigoProveedor)
VALUES (:codigo_pedido, TO_DATE(:fecha_solicitud, 'YYYY-MM-DD'), 'Pendiente',
:codigo_proveedor)

```

Esta sentencia inserta un nuevo registro en la tabla Pedido

SENTENCIA SQL 2

```

SELECT * FROM Producto WHERE CodigoProducto = :1

```

Esta consulta busca el producto en la tabla Producto utilizando el CódigoProducto proporcionado por el usuario, para verificar si el producto existe en la base de datos.

SENTENCIA SQL 3

```
INSERT INTO DetallePedido (CodigoDetallePedido, CódigoPedido, CódigoProducto, Cantidad)
VALUES (:codigo_detalle, :codigo_pedido, :codigo_producto, :cantidad)
```

Esta sentencia inserta un nuevo detalle del pedido en la tabla DetallePedido.

CÓDIGO EN PYTHON

La función permite registrar un pedido en la base de datos, con detalles de los productos solicitados. Se solicita al usuario los datos del pedido, como el Código de Pedido, Fecha de Solicitud, y el Código de Proveedor, (se conecta a la base de datos), e inserta el pedido en la tabla Pedido. Posteriormente entra en un bucle donde permite ingresar los detalles de los productos asociados al pedido.

```

from datetime import datetime

def añadir_pedido():
    # Pedir al usuario los datos del pedido
    print("Ingrese los datos del pedido:")
    codigo_pedido = input("Código del pedido (20 caracteres): ")
    fecha_solicitud = input("Fecha de solicitud del pedido (YYYY-MM-DD): ")
    codigo_proveedor = input("Código del proveedor (10 caracteres): ")

    try:
        # Conexión a la base de datos
        conn, cursor = conectar_bd()
        print("Conexión a la base de datos establecida.")

        # Insertar el pedido en la base de datos
        cursor.execute("""
            INSERT INTO Pedido (CodigoPedido, FechaSolicitud, EstadoPedido, CodigoProveedor)
            VALUES (:codigo_pedido, TO_DATE(:fecha_solicitud, 'YYYY-MM-DD'), 'Pendiente', :codigo_proveedor)
        """, {
            "codigo_pedido": codigo_pedido,
            "fecha_solicitud": fecha_solicitud,
            "codigo_proveedor": codigo_proveedor
        })
        print("Pedido insertado correctamente.")

        # Registrar detalles de la venta
        while True:
            codigo_producto = input("Código del producto (o 'fin' para terminar): ")
            if codigo_producto.lower() == 'fin':
                break

            cursor.execute("SELECT * FROM Producto WHERE CodigoProducto = :1", [codigo_producto])
            producto = cursor.fetchone()
            if not producto:
                print(f"Producto con código {codigo_producto} no existe.")
                continue

            cantidad = int(input(f"Ingrese la cantidad para {codigo_producto}:"))

            codigo_detalle = input("Código del detalle de pedido")

            cursor.execute("""
                INSERT INTO DetallePedido (CodigoDetallePedido, CodigoPedido, CodigoProducto, Cantidad)
                VALUES (:1, :2, :3, :4)
            """, [codigo_detalle, codigo_pedido, codigo_producto, cantidad])
    
```

RF4.3: Consultar historial de compras a proveedores.

SENTENCIA SQL 1

```

SELECT P.CodigoPedido, P.FechaSolicitud
FROM Pedido P
WHERE P.CodigoProveedor = :codigo_proveedor
AND P.FechaSolicitud BETWEEN TO_DATE(:fecha_inicio, 'YYYY-MM-DD') AND
TO_DATE(:fecha_fin, 'YYYY-MM-DD')

```

La sentencia busca los pedidos de un proveedor dentro de un rango de fechas determinado, siendo los valores proporcionados por el usuario, y devolviendo la consulta el Código del Pedido y Fecha de Solicitud de los pedidos realizados por ese proveedor.

SENTENCIA SQL 2

```
SELECT dp.CodigoProducto, dp.Cantidad, p.Nombre, p.PrecioCompra  
FROM DetallePedido dp  
JOIN Producto p ON dp.CodigoProducto = p.CodigoProducto  
WHERE dp.CodigoPedido = :codigo_pedido
```

La sentencia obtiene los detalles de los productos para cada pedido, es decir, que para un Código de Pedido determinado, se busca en la tabla DetallePedido los productos asociados, su Cantidad, el Nombre del producto y el Precio de compra. Los datos se obtienen al hacer la unión JOIN.

CÓDIGO EN PYTHON

La función permite consultar el historial de compras de un proveedor en un rango de fechas específico. Primero, solicita ingresar el Código del proveedor, la fecha de inicio y la fecha de fin de la consulta, ejecuta una consulta SQL explicada anteriormente, para obtener todos los pedidos realizados por ese proveedor dentro del rango de fechas proporcionado, y para cada pedido encontrado, se ejecuta la segunda consulta, que obtiene los detalles de ese pedido.

```

def consultar_historial_compras():
    codigo_proveedor = input("Ingrese el código del proveedor (CIF): ")
    fecha_inicio = input("Ingrese la fecha de inicio de la consulta (YYYY-MM-DD): ")
    fecha_fin = input("Ingrese la fecha de fin de la consulta (YYYY-MM-DD): ")

    try:
        # Conexión a la base de datos
        conn, cursor = conectar_bd() # Asume que conectar_bd() devuelve conexión y cursor

        # Obtener los pedidos realizados por el proveedor dentro del rango de fechas
        cursor.execute("""
            SELECT p.CodigoPedido, p.FechaSolicitud
            FROM Pedido p
            WHERE p.CodigoProveedor = :codigo_proveedor
            AND p.FechaSolicitud BETWEEN TO_DATE(:fecha_inicio, 'YYYY-MM-DD') AND TO_DATE(:fecha_fin, 'YYYY-MM-DD')
        """, [codigo_proveedor, fecha_inicio, fecha_fin])

        pedidos = cursor.fetchall()

        if not pedidos:
            print("No se encontraron pedidos para este proveedor en el rango de fechas proporcionado.")
            return

        # Mostrar los pedidos y sus detalles
        print(f"Historial de compras para el proveedor {codigo_proveedor}:")
        for pedido in pedidos:
            print(f"- Pedido {pedido[0]} realizado el {pedido[1]}")

            # Obtener los detalles de cada pedido
            cursor.execute("""
                SELECT dp.CodigoProducto, dp.Cantidad, p.Nombre, p.PrecioCompra
                FROM DetallePedido dp
                JOIN Producto p ON dp.CodigoProducto = p.CodigoProducto
                WHERE dp.CodigoPedido = :codigo_pedido
            """, [pedido[0]])
            detalles = cursor.fetchall()

            for detalle in detalles:
                print(f" - Detalle: Producto {detalle[0]} ({detalle[2]}), Cantidad {detalle[1]}, Precio de compra {detalle[3]} EUR")

        except oracledb.DatabaseError as e:
            print(f"Error al consultar el historial de compras: {e}")
    finally:
        # Cerrar la conexión
        cursor.close()
        conn.close()

    # Llamar a la función para probarla
    consultar_historial_compras()

```

RF4.4: Cancelar un pedido a un proveedor.

SENTENCIA SQL 1

```

SELECT EstadoPedido
FROM Pedido
WHERE CodigoPedido = :codigo_pedido

```

Esta consulta busca el EstadoPedido del pedido identificado por CódigoPedido en la tabla Pedido.

SENTENCIA SQL 2

```
UPDATE Pedido  
SET EstadoPedido = 'Cancelado'  
WHERE CódigoPedido = :codigo_pedido
```

Esta sentencia actualiza el estado de un pedido a "Cancelado" en la tabla **Pedido**, solo si el pedido es encontrado y está en estado "Pendiente".

SENTENCIA SQL 3

```
INSERT INTO PedidosCancelados (CódigoCancelacion, CódigoPedido, FechaCancelacion,  
MotivoCancelacion)  
VALUES (:codigo_cancelacion, :codigo_pedido, TO_DATE(:fecha_cancelacion, 'YYYY-MM-DD'),  
:motivo_cancelacion)
```

La sentencia inserta un registro en la tabla **PedidosCancelados**, que contiene los detalles del pedido cancelado.

CÓDIGO EN PYTHON

```

def cancelar_pedido():
    codigo_pedido = input("Ingrese el código del pedido a cancelar: ")
    motivo_cancelacion = input("Ingrese el motivo de la cancelación: ")

    # Obtener la conexión y el cursor de la base de datos
    conn, cursor = conectar_bd()

    if conn is None:
        print("No se pudo establecer la conexión a la base de datos.")
        return

    try:
        # Verificar si el pedido existe y está en un estado pendiente
        cursor.execute("""
            SELECT EstadoPedido
            FROM Pedido
            WHERE CodigoPedido = :codigo_pedido
        """, [codigo_pedido])

        estado_pedido = cursor.fetchone()

        if not estado_pedido:
            print(f"No se encontró el pedido con el código {codigo_pedido}.")
            return

        # Verificar que el estado del pedido sea "Pendiente"
        if estado_pedido[0] != 'Pendiente':
            print("El pedido no se puede cancelar porque ya ha sido completado o cancelado previamente.")
            return

        # Actualizar el estado del pedido a "Cancelado" en la tabla Pedido
        cursor.execute("""
            UPDATE Pedido
            SET EstadoPedido = 'Cancelado'
            WHERE CodigoPedido = :codigo_pedido
        """, [codigo_pedido])

        # Insertar el pedido cancelado en la tabla PedidosCancelados
        fecha_cancelacion = datetime.now().strftime('%Y-%m-%d')

        # Generar el código de cancelación, asegurándonos de que no exceda los 20 caracteres
        codigo_cancelacion = f"CAN{codigo_pedido[-17:]}" # Tomar los últimos 17 caracteres del código del pedido

        cursor.execute("""
            INSERT INTO PedidosCancelados (CodigoCancelacion, CodigoPedido, FechaCancelacion, MotivoCancelacion)
            VALUES (:codigo_cancelacion, :codigo_pedido, TO_DATE(:fecha_cancelacion, 'YYYY-MM-DD'), :motivo_cancelacion)
        """, {
            "codigo_cancelacion": codigo_cancelacion,
            "codigo_pedido": codigo_pedido,
            "fecha_cancelacion": fecha_cancelacion,
            "motivo_cancelacion": motivo_cancelacion
        })
    
```



```

        # Confirmar los cambios
        conn.commit()
        print(f"El pedido {codigo_pedido} ha sido cancelado exitosamente y registrado en la tabla de pedidos cancelados.")

    except oracledb.DatabaseError as e:
        print(f"Error al cancelar el pedido: {e}")

    finally:
        # Cerrar la conexión y el cursor si fueron inicializados
        if cursor:
            cursor.close()
        if conn:
            conn.close()

    # Llamar a la función para probarla
    cancelar_pedido()

```

RF4.5: Marcar el pedido como completado.

SENTENCIA SQL 1

```
SELECT EstadoPedido  
FROM Pedido  
WHERE CodigoPedido = :codigo_pedido
```

Esta consulta busca el EstadoPedido de un pedido con el código proporcionado por el usuario (CodigoPedido) en la tabla Pedido. Se utiliza para verificar si el pedido existe y obtener su estado.

SENTENCIA SQL 2

```
UPDATE Pedido  
SET EstadoPedido = 'Completado'  
WHERE CodigoPedido = :codigo_pedido
```

Esta sentencia actualiza el EstadoPedido de un pedido a "Completado" en la tabla Pedido, solo si el pedido existe y su estado era "Pendiente". Esto indica que el pedido ha sido procesado y completado.

CÓDIGO EN PYTHON

```

def completar_pedido():
    codigo_pedido = input("Ingrese el código del pedido a completar: ")

    try:
        conn, cursor = conectar_bd() # Asume que conectar_bd() devuelve la conexión y el cursor
        if conn is None:
            print("No se pudo establecer la conexión a la base de datos.")
            return

        # Verificar si el pedido existe y está en un estado pendiente
        cursor.execute("""
            SELECT EstadoPedido
            FROM Pedido
            WHERE CódigoPedido = :codigo_pedido
        """, [codigo_pedido])

        estado_pedido = cursor.fetchone()

        if not estado_pedido:
            print(f"No se encontró el pedido con el código {codigo_pedido}.")
            return

        # Verificar que el estado del pedido sea "Pendiente"
        if estado_pedido[0] != 'Pendiente':
            print("El pedido no se puede completar porque no está en estado pendiente.")
            return

        # Actualizar el estado del pedido a "Completado"
        cursor.execute("""
            UPDATE Pedido
            SET EstadoPedido = 'Completado'
            WHERE CódigoPedido = :codigo_pedido
        """, [codigo_pedido])

        # Confirmar los cambios
        conn.commit()
        print(f"El pedido {codigo_pedido} ha sido completado exitosamente.")

    except oracledb.DatabaseError as e:
        print(f"Error al completar el pedido: {e}")

    finally:
        if cursor:
            cursor.close()
        if conn:
            conn.close()

# Llamar a la función para probarla
completar_pedido()

```

RF 4.6: Dar de baja a un proveedor.

SENTENCIA SQL 1

```
SELECT COUNT(*)  
FROM Pedido  
WHERE CódigoProveedor = :código_proveedor AND EstadoPedido = 'Pendiente'
```

Esta consulta cuenta el número de pedidos que el proveedor, identificado por CódigoProveedor, tiene en estado "Pendiente". Si el proveedor tiene pedidos pendientes, no se podrá dar de baja.

SENTENCIA SQL 2

```
UPDATE Proveedor  
SET Estado = '0'  
WHERE CódigoProveedor = :código_proveedor
```

Esta sentencia actualiza el estado del proveedor a inactivo (0) en la tabla Proveedor, lo que indica que el proveedor ha sido dado de baja y ya no está activo.

CÓDIGO EN PYTHON

```
def dar_de_baja_proveedor():
    codigo_proveedor = input("Ingrese el código del proveedor a dar de baja: ")

    try:
        conn, cursor = conectar_bd()

        # Verificar si el proveedor tiene pedidos pendientes
        cursor.execute("""
            SELECT COUNT(*)
            FROM Pedido
            WHERE CodigoProveedor = :codigo_proveedor AND EstadoPedido = 'Pendiente'
        """, [codigo_proveedor])

        pedidos_pendientes = cursor.fetchone()[0]

        if pedidos_pendientes > 0:
            print("El proveedor tiene pedidos pendientes y no puede ser dado de baja.")
            return

        # Cambiar el estado del proveedor a inactivo (0)
        cursor.execute("""
            UPDATE Proveedor
            SET Estado = '0'
            WHERE CodigoProveedor = :codigo_proveedor
        """, [codigo_proveedor])

        conn.commit()
        print(f"Proveedor {codigo_proveedor} dado de baja exitosamente.")

    except oracledb.DatabaseError as e:
        print(f"Error al dar de baja al proveedor: {e}")

    finally:
        cursor.close()
        conn.close()

# Llamar a la función para probarla
dar_de_baja_proveedor()
```

Código Disparadores

```

CREATE OR REPLACE TRIGGER VerificarProveedorActivo
BEFORE INSERT ON Pedido
FOR EACH ROW
DECLARE
    EstadoProveedor INT;
BEGIN
    -- Obtener el estado del proveedor
    SELECT Estado
    INTO EstadoProveedor
    FROM Proveedor
    WHERE CódigoProveedor = :NEW.CódigoProveedor;

    -- Si el proveedor no está activo, se lanza un error
    IF EstadoProveedor ≠ 1 THEN
        RAISE_APPLICATION_ERROR(-20001, 'El proveedor no está activo. No se
puede realizar el pedido.');
    END IF;
END;

```

El trigger **VerificarProveedorActivo**, se ejecuta antes de insertar un pedido. Verifica si el proveedor del pedido está activo (estado = 1). Si el proveedor no está activo, lanza un error y evita que el pedido se registre. Si el proveedor está activo, el pedido se inserta normalmente. Esto asegura que solo los proveedores activos puedan recibir pedidos.

El disparador debe situarse en el **proceso de inserción de pedidos**. En lo que a los requisitos funcionales, este disparador debería formar parte de:

Creación de un pedido: se ejecutará cuando un usuario intente insertar un nuevo registro en la tabla Pedido. Antes de que el pedido se inserte, el sistema verificará que el proveedor asociado esté activo.

Gestión de pedidos: a un nivel más interno, el disparador se activará como parte del proceso de validación del pedido, garantizando que no se inserten pedidos para proveedores inactivos.

Disparador para actualizar stock tras pedido completado (Francisco José Ramos Moya)

```

CREATE OR REPLACE TRIGGER actualizar_stock_completado
AFTER UPDATE OF EstadoPedido ON Pedido
FOR EACH ROW
BEGIN
    IF :NEW.EstablecerEstado = 'Completado' THEN
        FOR detalle IN (
            SELECT CódigoProducto, Cantidad
            FROM DetallePedido
            WHERE CódigoPedido = :NEW.CódigoPedido
        )
        BEGIN
            UPDATE Producto
            SET Stock = Stock - :detalle.Cantidad
            WHERE CódigoProducto = :detalle.CódigoProducto;
        END;
    END IF;
END;

```

```
) LOOP
    UPDATE Producto
    SET Stock = Stock + detalle.Cantidad
    WHERE CodigoProducto = detalle.CodigoProducto;
END LOOP;
END IF;
END;
```

Cuando actualizamos el atributo EstadoPedido en la tabla Pedido, el disparador se activa para procesar cualquier cambio. Sin embargo, cuando la actualización implica un cambio de estado de 'Pendiente' a 'Completado', el disparador recorre los detalles asociados al pedido en la tabla DetallePedido y actualiza automáticamente el stock de los productos en la tabla Producto. En este proceso, el disparador incrementa el stock de cada producto involucrado sumando la cantidad especificada en el detalle del pedido, garantizando que el inventario refleje correctamente los productos recibidos.

Datos sensibles y Descripción de los aspectos legales

En el **subsistema de gestión de proveedores**, los datos sensibles supondrán aquellos, que pueden llegar a comprometer la seguridad, privacidad o integridad de los proveedores, así como de las operaciones o transacciones. Por tanto, si fuese necesario enumerarlos específicamente, nos referiríamos a:

De la tabla Proveedor:

- Código Proveedor (si estuviese integrado por el DNI)
- Teléfono
- Correo
- Dirección Social

Estos datos incluyen sobre todo información personal y de contacto, que debe protegerse adecuadamente para evitar filtraciones o mal uso.

Subsistema de Informes y Análisis

Sentencias de creación de tablas, con claves y restricciones

El diseño del Subsistema de Informes y Análisis, basado únicamente en lectura de datos existentes (RDE), garantizando así:

1. Eficiencia: Sin tablas adicionales que incrementen la complejidad del sistema.
2. Evitación de redundancia: Se utiliza información ya existente en la base de datos.
3. Simplicidad: Las consultas SQL proporcionan la información requerida para generar los informes sin modificar ni almacenar datos.

Por lo tanto no creamos ninguna tabla en la base de datos.

Descripción de las transacciones

RF 5.1 Generar informe de ventas de un intervalo.

SENTENCIA SQL

SELECT

```
p.Nombre AS NombreProducto,  
p.CodigoProducto AS CodigoProducto,  
SUM(dv.Cantidad) AS CantidadTotal  
  
FROM Venta v  
  
JOIN DetalleVenta dv ON v.CodigoVenta = dv.CodigoVenta  
  
JOIN Producto p ON dv.CodigoProducto = p.CodigoProducto  
  
WHERE v.FechaHoraVenta BETWEEN TO_TIMESTAMP(:1, 'YYYY-MM-DD HH24:MI:SS')  
AND TO_TIMESTAMP(:2, 'YYYY-MM-DD HH24:MI:SS')  
  
GROUP BY p.Nombre, p.CodigoProducto  
  
ORDER BY CantidadTotal DESC
```

Esta sentencia genera un informe que muestra los productos vendidos en un intervalo de fechas específico, agrupándolos por nombre y código. Calcula la cantidad total vendida de cada producto, ordenando los resultados de mayor a menor cantidad vendida.

CÓDIGO EN PYTHON

```

1 def generar_informe_ventas_por_intervalo():
2
3     # Solicitar fechas de inicio y fin al empleado
4     fecha_inicio = input("Ingrese la fecha de inicio (YYYY-MM-DD): ")
5     fecha_fin = input("Ingrese la fecha de fin (YYYY-MM-DD): ")
6
7     # Validación: Verificar que la fecha de inicio no sea posterior a la fecha de fin
8     if fecha_inicio > fecha_fin:
9         print("Error: La fecha de inicio no puede ser posterior a la fecha de fin.")
10        return
11
12     conn, cursor = conectar_bd()
13     try:
14         # Consulta SQL: Ventas realizadas en el intervalo especificado
15         cursor.execute("""
16             SELECT
17                 p.Nombre AS NombreProducto,
18                 p.CodigoProducto AS CodigoProducto,
19                 SUM(dv.Cantidad) AS CantidadTotal
20             FROM Venta v
21             JOIN DetalleVenta dv ON v.CodigoVenta = dv.CodigoVenta
22             JOIN Producto p ON dv.CodigoProducto = p.CodigoProducto
23             WHERE v.FechaHoraVenta BETWEEN TO_TIMESTAMP(:1, 'YYYY-MM-DD HH24:MI:SS')
24             AND TO_TIMESTAMP(:2, 'YYYY-MM-DD HH24:MI:SS')
25             GROUP BY p.Nombre, p.CodigoProducto
26             ORDER BY CantidadTotal DESC
27         """, [f"{fecha_inicio} 00:00:00", f"{fecha_fin} 23:59:59"])
28
29         # Obtener los resultados
30         resultados = cursor.fetchall()
31
32         # Verificar si hay resultados
33         if not resultados:
34             print(f"No se encontraron ventas entre {fecha_inicio} y {fecha_fin}.")
35             return
36
37         # Separar los datos en listas
38         nombres_producto = [fila[0] for fila in resultados]
39         codigos_producto = [fila[1] for fila in resultados]
40         cantidades = [fila[2] for fila in resultados]
41
42         # Mostrar las listas
43         print("\nInforme de Ventas:")
44         print(f"Nombres de Producto: {nombres_producto}")
45         print(f"Códigos de Producto: {codigos_producto}")
46         print(f"Cantidad Total Vendida: {cantidades}")
47
48         print("\nInforme generado exitosamente.")
49
50     except Exception as e:
51         print(f"Error al generar el informe: {e}")

```

RF 5.2 Generar análisis de productos más vendidos.

SENTENCIA SQL

```
SELECT  
    p.Nombre AS NombreProducto,  
    p.CodigoProducto AS CodigoProducto,  
    SUM(dv.Cantidad) AS CantidadVendida  
  
    FROM Venta v  
  
    JOIN DetalleVenta dv ON v.CodigoVenta = dv.CodigoVenta  
  
    JOIN Producto p ON dv.CodigoProducto = p.CodigoProducto  
  
    WHERE v.FechaHoraVenta BETWEEN TO_TIMESTAMP(:1, 'YYYY-MM-DD HH24:MI:SS')  
        AND TO_TIMESTAMP(:2, 'YYYY-MM-DD HH24:MI:SS')  
  
    GROUP BY p.Nombre, p.CodigoProducto  
  
    HAVING SUM(dv.Cantidad) > 0 -- Garantiza solo productos con ventas  
  
    ORDER BY CantidadVendida DESC  
  
    FETCH FIRST :3 ROWS ONLY
```

Esta sentencia genera un análisis de los productos más vendidos en un intervalo de fechas específico. Agrupa los resultados por nombre y código del producto, calcula la cantidad total vendida de cada uno y filtra solo aquellos productos con ventas mayores a 0. Finalmente, ordena los productos de mayor a menor cantidad vendida.

CÓDIGO EN PYTHON

```

1  def generar_analisis_productos_mas_vendidos():
2      # Solicitar fechas de inicio y fin
3      fecha_inicio = input("Ingrese la fecha de inicio (YYYY-MM-DD): ")
4      fecha_fin = input("Ingrese la fecha de fin (YYYY-MM-DD): ")
5
6      # Validar las fechas
7      if fecha_inicio > fecha_fin:
8          print("Error: La fecha de inicio no puede ser posterior a la fecha de fin.")
9          return
10
11     # Solicitar límite de productos a mostrar
12     limite = int(input("Ingrese el límite de productos a mostrar: "))
13     if limite <= 0:
14         print("Error: El límite debe ser un número entero positivo.")
15         return
16
17     try:
18
19         conn, cursor = conectar_bd()
20         # Consulta SQL: Obtener los productos más vendidos
21         cursor.execute("""
22             SELECT
23                 p.Nombre AS NombreProducto,
24                 p.CodigoProducto AS CodigoProducto,
25                 SUM(dv.Cantidad) AS CantidadVendida
26             FROM Venta v
27             JOIN DetalleVenta dv ON v.CodigoVenta = dv.CodigoVenta
28             JOIN Producto p ON dv.CodigoProducto = p.CodigoProducto
29             WHERE v.FechaHoraVenta BETWEEN TO_TIMESTAMP(:1, 'YYYY-MM-DD HH24:MI:SS')
30             AND TO_TIMESTAMP(:2, 'YYYY-MM-DD HH24:MI:SS')
31             GROUP BY p.Nombre, p.CodigoProducto
32             HAVING SUM(dv.Cantidad) > 0 -- Garantiza solo productos con ventas
33             ORDER BY CantidadVendida DESC
34             FETCH FIRST :3 ROWS ONLY
35             """, [f"{fecha_inicio} 00:00:00", f"{fecha_fin} 23:59:59", limite])
36
37         # Obtener los resultados
38         resultados = cursor.fetchall()
39
40         # Verificar si hay resultados
41         if not resultados:
42             print(f"No se encontraron productos vendidos entre {fecha_inicio} y {fecha_fin}.")
43             return
44
45         # Separar los datos en listas
46         nombres_producto = [fila[0] for fila in resultados]
47         codigos_producto = [fila[1] for fila in resultados]
48         cantidades = [fila[2] for fila in resultados]
49
50         # Mostrar las listas
51         print("\nAnálisis de Productos Más Vendidos:")
52         print(f"Nombres de Producto: {nombres_producto}")
53         print(f"Códigos de Producto: {codigos_producto}")
54         print(f"Cantidad Vendida: {cantidades}")
55
56         print("\nAnálisis generado exitosamente.")
57
58     except Exception as e:
59
60         print(f"Error al generar el análisis: {e}")

```

RF 5.3 Generar análisis de ingresos para un producto.

SENTENCIA SQL

```
SELECT  
      p.Nombre AS NombreProducto,  
      p.CodigoProducto AS CodigoProducto,  
      SUM(dv.Cantidad * (dv.PrecioVentaFinal - p.PrecioCompra)) AS  
      Ingresos  
  FROM Venta v  
  JOIN DetalleVenta dv ON v.CodigoVenta = dv.CodigoVenta  
  JOIN Producto p ON dv.CodigoProducto = p.CodigoProducto  
 WHERE dv.CodigoProducto = :1  
   AND v.FechaHoraVenta BETWEEN TO_TIMESTAMP(:2, 'YYYY-MM-DD HH24:MI:SS')  
   AND TO_TIMESTAMP(:3, 'YYYY-MM-DD HH24:MI:SS')  
 GROUP BY p.Nombre, p.CodigoProducto
```

Esta sentencia calcula los ingresos generados por un producto específico en un intervalo de fechas dado. Agrupa los resultados por nombre y código del producto, y calcula los ingresos como el total de la cantidad vendida multiplicada por la diferencia entre el precio de venta final y el precio de compra. Esto permite analizar el beneficio generado por un producto durante el período seleccionado.

CÓDIGO EN PYTHON

```

1 def generar_analisis_ingresos_producto():
2
3
4     # Solicitar entradas al usuario
5     codigo_producto = input("Ingrese el código del producto: ")
6     fecha_inicio = input("Ingrese la fecha de inicio (YYYY-MM-DD): ")
7     fecha_fin = input("Ingrese la fecha de fin (YYYY-MM-DD): ")
8
9     # Validar las fechas
10    if fecha_inicio > fecha_fin:
11        print("Error: La fecha de inicio no puede ser posterior a la fecha de fin.")
12        return
13    try:
14
15        conn,cursor = conectar_bd()
16
17        # Consulta SQL: Obtener ingresos del producto en el intervalo
18
19        cursor.execute("""
20            SELECT
21                p.Nombre AS NombreProducto,
22                p.CodigoProducto AS CodigoProducto,
23                SUM(dv.Cantidad * (dv.PrecioVentaFinal - p.PrecioCompra)) AS Ingresos
24            FROM Venta v
25            JOIN DetalleVenta dv ON v.CodigoVenta = dv.CodigoVenta
26            JOIN Producto p ON dv.CodigoProducto = p.CodigoProducto
27            WHERE dv.CodigoProducto = :1
28            AND v.FechaHoraVenta BETWEEN TO_TIMESTAMP(:2, 'YYYY-MM-DD HH24:MI:SS')
29            AND TO_TIMESTAMP(:3, 'YYYY-MM-DD HH24:MI:SS')
30            GROUP BY p.Nombre, p.CodigoProducto
31        """, [codigo_producto, f"{fecha_inicio} 00:00:00", f"{fecha_fin} 23:59:59"])
32
33        # Obtener los resultados
34        resultado = cursor.fetchone()
35
36        # Verificar si hay datos
37        if not resultado:
38            print(f"No se encontraron ingresos para el producto {codigo_producto} entre {fecha_inicio} y {fecha_fin}.")
39            return
40
41        # Mostrar los datos
42        print("\nAnálisis de Ingresos para el Producto:")
43        print(f"Nombre del Producto: {resultado[0]}")
44        print(f"Código del Producto: {resultado[1]}")
45        print(f"Ingresos Totales: {resultado[2]:.2f}")
46
47        print("\nAnálisis generado exitosamente.")
48
49    except Exception as e:
50        print(f"Error al generar el análisis: {e}")

```

RF 5.4 Generar informe de pedidos de un intervalo.

SENTENCIA SQL

SELECT

```
p.Nombre AS NombreProducto,  
dp.CodigoProducto AS CodigoProducto,  
SUM(dp.Cantidad) AS CantidadTotal,  
p.PrecioCompra AS PrecioCompra  
  
FROM Pedido ped  
  
JOIN DetallePedido dp ON ped.CodigoPedido = dp.CodigoPedido  
  
JOIN Producto p ON dp.CodigoProducto = p.CodigoProducto  
  
WHERE ped.FechaSolicitud BETWEEN TO_DATE(:1, 'YYYY-MM-DD') AND TO_DATE(:2,  
'YYYY-MM-DD')  
  
GROUP BY p.Nombre, dp.CodigoProducto, p.PrecioCompra  
  
ORDER BY CantidadTotal DESC
```

Esta sentencia genera un informe de los productos incluidos en pedidos realizados dentro de un intervalo de fechas. Agrupa los resultados por nombre, código del producto y precio de compra, calcula la cantidad total solicitada de cada producto y ordena los resultados de mayor a menor cantidad solicitada. Esto permite evaluar los productos más adquiridos durante el período analizado.

CÓDIGO EN PYTHON

```

1  def generar_informe_pedidos_por_intervalo():
2
3
4      # Solicitar fechas de inicio y fin al usuario
5      fecha_inicio = input("Ingrese la fecha de inicio (YYYY-MM-DD): ")
6      fecha_fin = input("Ingrese la fecha de fin (YYYY-MM-DD): ")
7
8      # Validar las fechas
9      if fecha_inicio > fecha_fin:
10         print("Error: La fecha de inicio no puede ser posterior a la fecha de fin.")
11         return
12
13     try:
14         conn,cursor = conectar_bd()
15         # Consulta SQL para obtener los pedidos en el intervalo
16         cursor.execute("""
17             SELECT
18                 p.Nombre AS NombreProducto,
19                 dp.CodigoProducto AS CodigoProducto,
20                 SUM(dp.Cantidad) AS CantidadTotal,
21                 p.PrecioCompra AS PrecioCompra
22             FROM Pedido ped
23             JOIN DetallePedido dp ON ped.CodigoPedido = dp.CodigoPedido
24             JOIN Producto p ON dp.CodigoProducto = p.CodigoProducto
25             WHERE ped.FechaSolicitud BETWEEN TO_DATE(:1, 'YYYY-MM-DD') AND TO_DATE(:2, 'YYYY-MM-DD')
26             GROUP BY p.Nombre, dp.CodigoProducto, p.PrecioCompra
27             ORDER BY CantidadTotal DESC
28             """, [fecha_inicio, fecha_fin])
29
30         # Obtener los resultados
31         resultados = cursor.fetchall()
32
33         # Verificar si hay resultados
34         if not resultados:
35             print(f"No se encontraron pedidos entre {fecha_inicio} y {fecha_fin}.")
36             return
37
38         # Separar los datos en listas
39         nombres_producto = [fila[0] for fila in resultados]
40         codigos_producto = [fila[1] for fila in resultados]
41         cantidades = [fila[2] for fila in resultados]
42         precios_compra = [fila[3] for fila in resultados]
43
44         # Mostrar el informe
45         print("\nInforme de Pedidos:")
46         print(f"Nombres de Producto: {nombres_producto}")
47         print(f"Códigos de Producto: {codigos_producto}")
48         print(f"Cantidad Total Comprada: {cantidades}")
49         print(f"Precios de Compra: {precios_compra}")
50
51         print("\nInforme generado exitosamente.")
52
53     except Exception as e:
54         print(f"Error al generar el informe: {e}")

```

RF 5.5 Generar análisis de ventas por familia de productos.

SENTENCIA SQL

```
SELECT  
    p.Categoría AS Categoría,  
    SUM(dv.Cantidad) AS CantidadTotal,  
    SUM(dv.Cantidad * dv.PrecioVentaFinal) AS Facturación  
FROM Venta v  
JOIN DetalleVenta dv ON v.CódigoVenta = dv.CódigoVenta  
JOIN Producto p ON dv.CódigoProducto = p.CódigoProducto  
WHERE p.Categoría = :1  
AND v.FechaHoraVenta BETWEEN TO_TIMESTAMP(:2, 'YYYY-MM-DD HH24:MI:SS')  
AND TO_TIMESTAMP(:3, 'YYYY-MM-DD HH24:MI:SS')  
GROUP BY p.Categoría
```

Esta sentencia genera un análisis de las ventas agrupadas por categoría de producto en un intervalo de fechas específico. Calcula la cantidad total vendida y la facturación generada (multiplicando la cantidad vendida por el precio de venta final) para la categoría especificada, agrupando los resultados por el nombre de la categoría. Esto permite evaluar el rendimiento de una familia de productos durante el período seleccionado.

CÓDIGO EN PYTHON

```

1  def generar_analisis_ventas_por_familia():
2
3
4      # Solicitar entradas al usuario
5      categoria = input("Ingrese la categoría del producto: ")
6      fecha_inicio = input("Ingrese la fecha de inicio (YYYY-MM-DD): ")
7      fecha_fin = input("Ingrese la fecha de fin (YYYY-MM-DD): ")
8
9      # Validar las fechas
10     if fecha_inicio > fecha_fin:
11         print("Error: La fecha de inicio no puede ser posterior a la fecha de fin.")
12         return
13
14     try:
15         conn,cursor = conectar_bd()
16         # Consulta SQL: Obtener las ventas agrupadas por familia de productos
17         cursor.execute("""
18             SELECT
19                 p.Categoría AS Categoría,
20                 SUM(dv.Cantidad) AS CantidadTotal,
21                 SUM(dv.Cantidad * dv.PrecioVentaFinal) AS Facturación
22             FROM Venta v
23             JOIN DetalleVenta dv ON v.CódigoVenta = dv.CódigoVenta
24             JOIN Producto p ON dv.CódigoProducto = p.CódigoProducto
25             WHERE p.Categoría = :1
26             AND v.FechaHoraVenta BETWEEN TO_TIMESTAMP(:2, 'YYYY-MM-DD HH24:MI:SS')
27             AND TO_TIMESTAMP(:3, 'YYYY-MM-DD HH24:MI:SS')
28             GROUP BY p.Categoría
29             "", [categoria, f"{fecha_inicio} 00:00:00", f"{fecha_fin} 23:59:59"])
30
31         # Obtener los resultados
32         resultado = cursor.fetchone()
33
34         # Verificar si hay datos
35         if not resultado:
36             print(f"No se encontraron ventas para la categoría '{categoria}' entre {fecha_inicio} y {fecha_fin}.")
37             return
38
39         # Mostrar los datos
40         print("\nAnálisis de Ventas por Familia de Productos:")
41         print(f"Categoría: {resultado[0]}")
42         print(f"Cantidad Total Vendida: {resultado[1]}")
43         print(f"Facturación Total: {resultado[2]:.2f}")
44
45         print("\nAnálisis generado exitosamente.")
46
47     except Exception as e:
48         print(f"Error al generar el análisis: {e}")

```

Código Disparadores

El Subsistema de Informes y Análisis se basa únicamente en lectura de datos existentes (RDE) y no realiza modificaciones en la base de datos (INSERT, DELETE, UPDATE), no necesitas implementar disparadores.

Datos sensibles y Descripción de los aspectos legales

El Subsistema de Informes y Análisis accede únicamente a datos existentes en la base de datos para generar informes y no guarda información adicional.

Accede a :

- Información de Productos
 - Nombres y códigos de producto.
 - Categorías de producto.
 - Precios de compra y venta.
 - Cantidad vendidas o compradas
- Información de Pedidos y Proveedores.
 - Código de Proveedores.
 - Fechas e importes de pedidos.
 - Precios de compra y venta.
 - Productos incluidos en pedidos.

Motivación de la elección de software

Hemos decidido implementar nuestro Sistema de Información con el mismo Software utilizado en el seminario 1. Hemos elegido utilizar un entorno de **Anaconda** para programar las interfaces y acceso a la base de datos con **Python**. La base de datos utilizada ha sido **SQL Developer**, con las cuentas facilitadas por la Universidad de Granada. La conexión con Python a la base de datos ha sido gracias al uso de la librería “**oracledb**”. Como IDE, hemos utilizado **Visual Studio Code**, que también cuenta con una extensión “**Oracle SQL Developer Extension for VSCode**” muy útil para realizar conexiones de manera sencilla a SQL Developer.

Esta elección de software está motivada por la sencillez y familiaridad de la misma. Es un método que no solamente hemos usado recientemente en el seminario 1 de esta asignatura, sino también en la asignatura “Fundamentos de Bases de Datos”, donde pudimos ganar experiencia en SQL y SQL Developer.