Universidad de Sevilla

IISSI

Proyecto

Modas Omaita

Autores:
Daniel González Corzo
Jesús Pineda Márquez
José Luis Mármol Romero
Roberto Hueso Gómez

22 de junio de 2017

Índice

1. Introducción

Omaita Modas es una tienda situada en la localidad de Alcalá de Guadaira y más exactamente en la calle Pepe Luces nº20, la cual pertenece a una cadena de tiendas de ropa que se especializa en la venta de ropa y accesorios a un público maduro y femenino. Nuestro cliente busca ser una tienda puntera en su cadena gracias a que tiene a su disposición toda la atención del público de la localidad, ya que es la única de esta cadena en la misma. Actualmente nuestro cliente no pasa por la mejor situación en lo que a clientela se refiere, además de que carece de personal, por tanto, nuestro proyecto tiene como base ayudar a nuestro cliente en la gestión de la tienda con un sistema informático, el cual nos permita realizar pedidos a proveedores y visualizar productos en el stock de la otras tiendas de la cadena, y la creación de una página online donde sus clientes puedan visualizar y comprar un producto determinado en cualquier momento, con esto último se quiere conseguir ampliar la clientela de la tienda.

2. Glosario de Términos

Término	Descripción
Albarán de entrega	Documento obtenido en la recepción del pedido que verifica la correcta entrega del mismo.
Almacén	Almacén que tienen en común todas las tiendas de la cadena Omaita modas.
Aviso	Notificación o anuncio dado para comunicar de la falta o limitación.
Consulta	Actividad que se realiza para tratar de encontrar cualquier entidad almacenada en la base de datos.
Cadena	Conjunto de tiendas relacionadas entre sí que ofrecen una mezcla estándar de productos, las cuales disponen de almacenes en común.
Categoría	Tipo de producto que hay en la tienda.
Emplazamiento	Inmueble de la cadena, puede ser una tienda o el almacén
Factura	Documento en el que se plasman las compras realizadas además del número de referencia en la base de datos a la lista de compras.
Pedido	Petición de productos de un emplazamiento al proveedor.
Proveedor	Entidad económica a la cual varias empresas le compran el material que usarán en la misma o que luego lo venderán al por menor.
Solicitud	Petición de traspaso de un emplazamiento a otro
Stock	Conjunto de mercancías o productos que se tienen almacenados en espera de su venta o comercialización.
Stock mínimo	5 unidades de un producto almacenadas
Traspaso	Transferencia de uno o varios productos de un emplazamiento a otro

3. Modelo de negocio

3.1. Proceso de venta

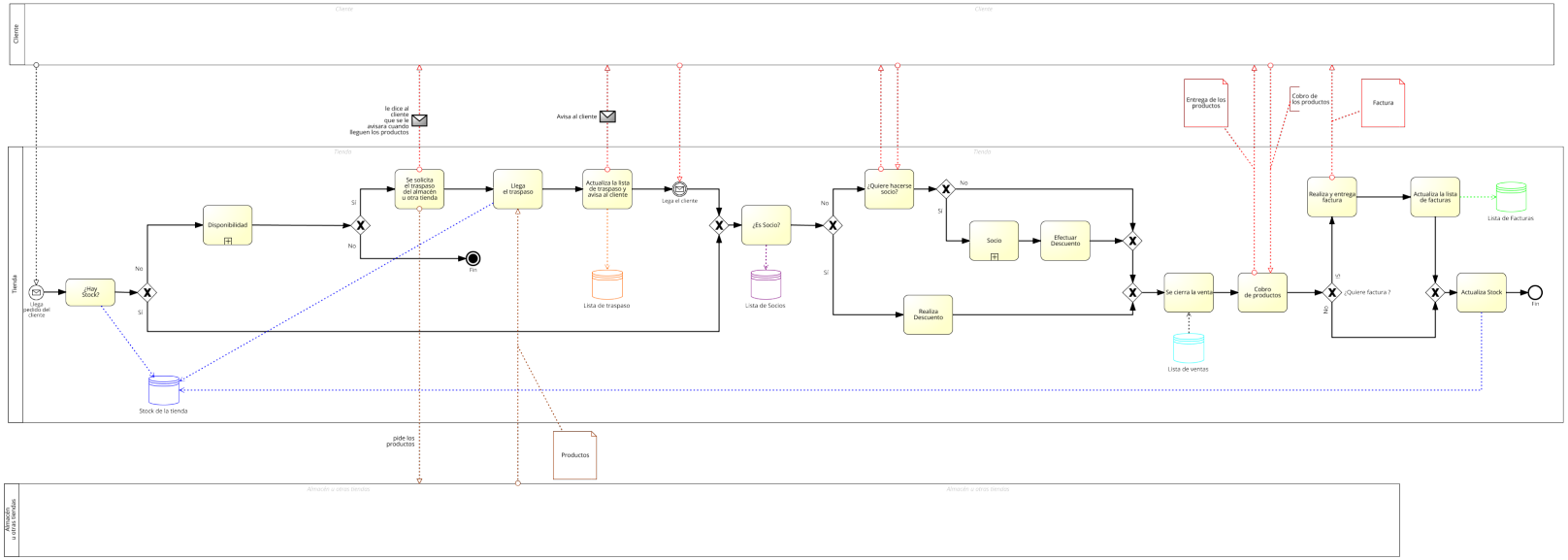
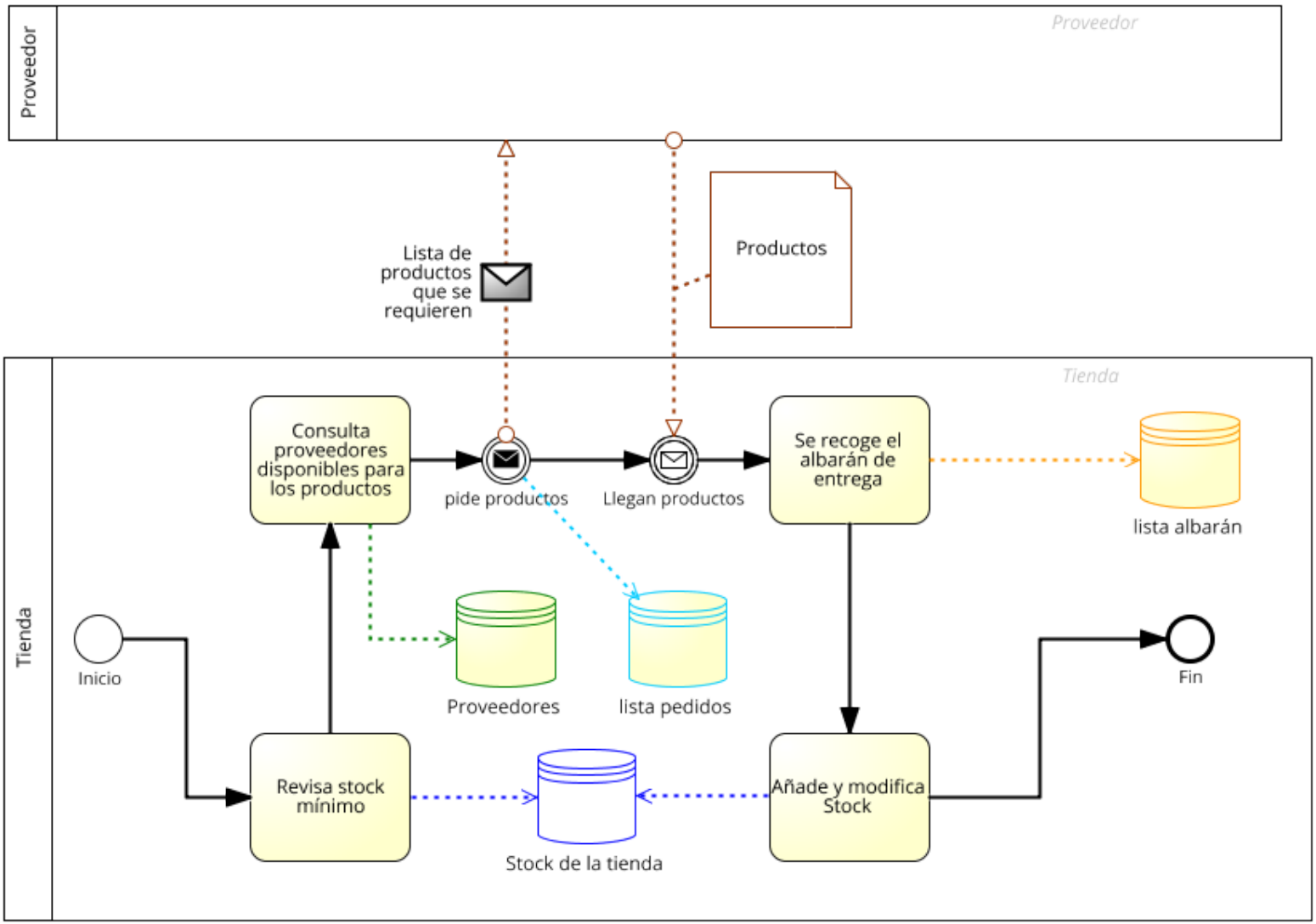
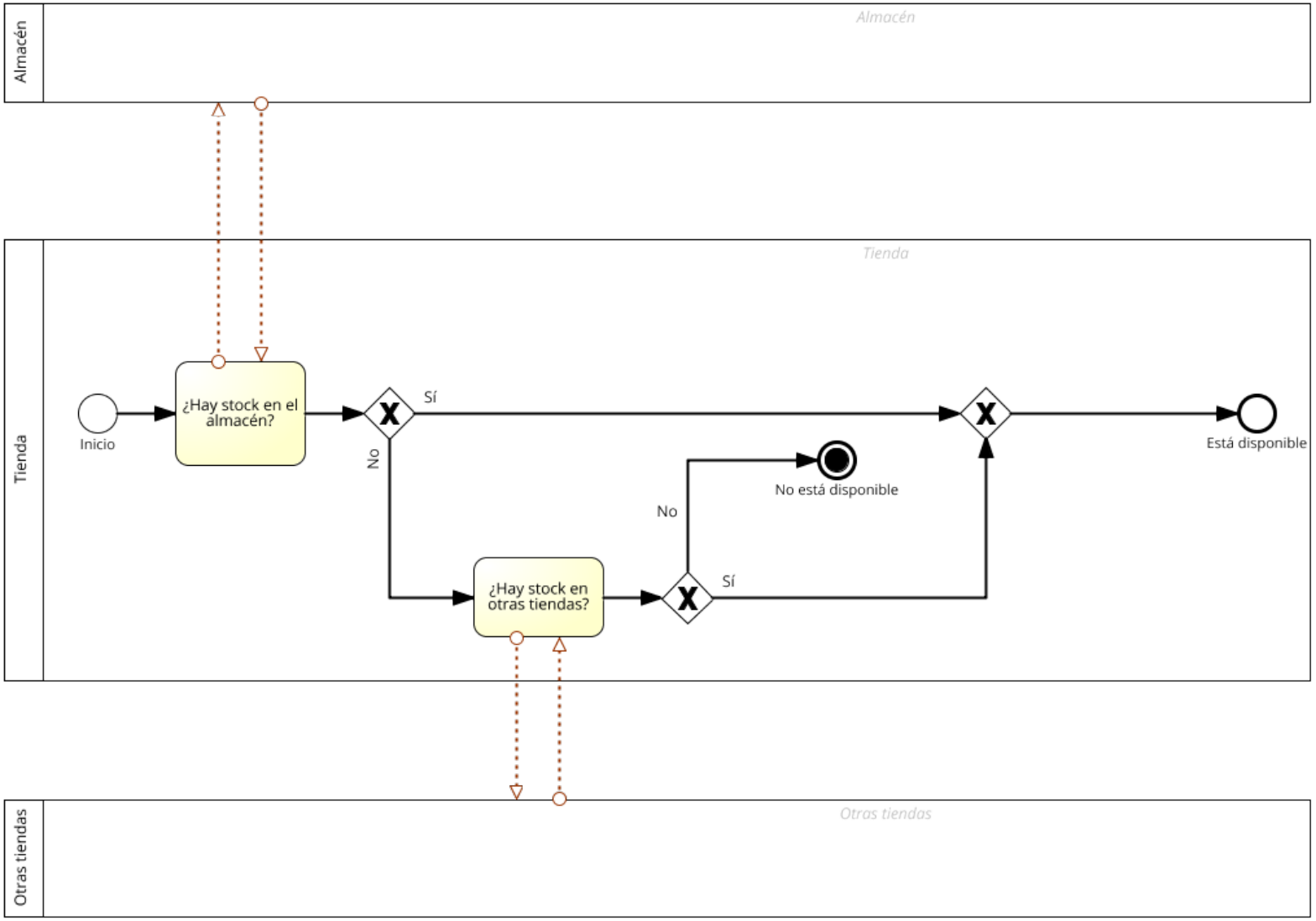


Figura 1: Proceso de venta

3.2. Proceso de compra al proveedor



3.3. Proceso de disponibilidad



3.4. Proceso de creación de socios

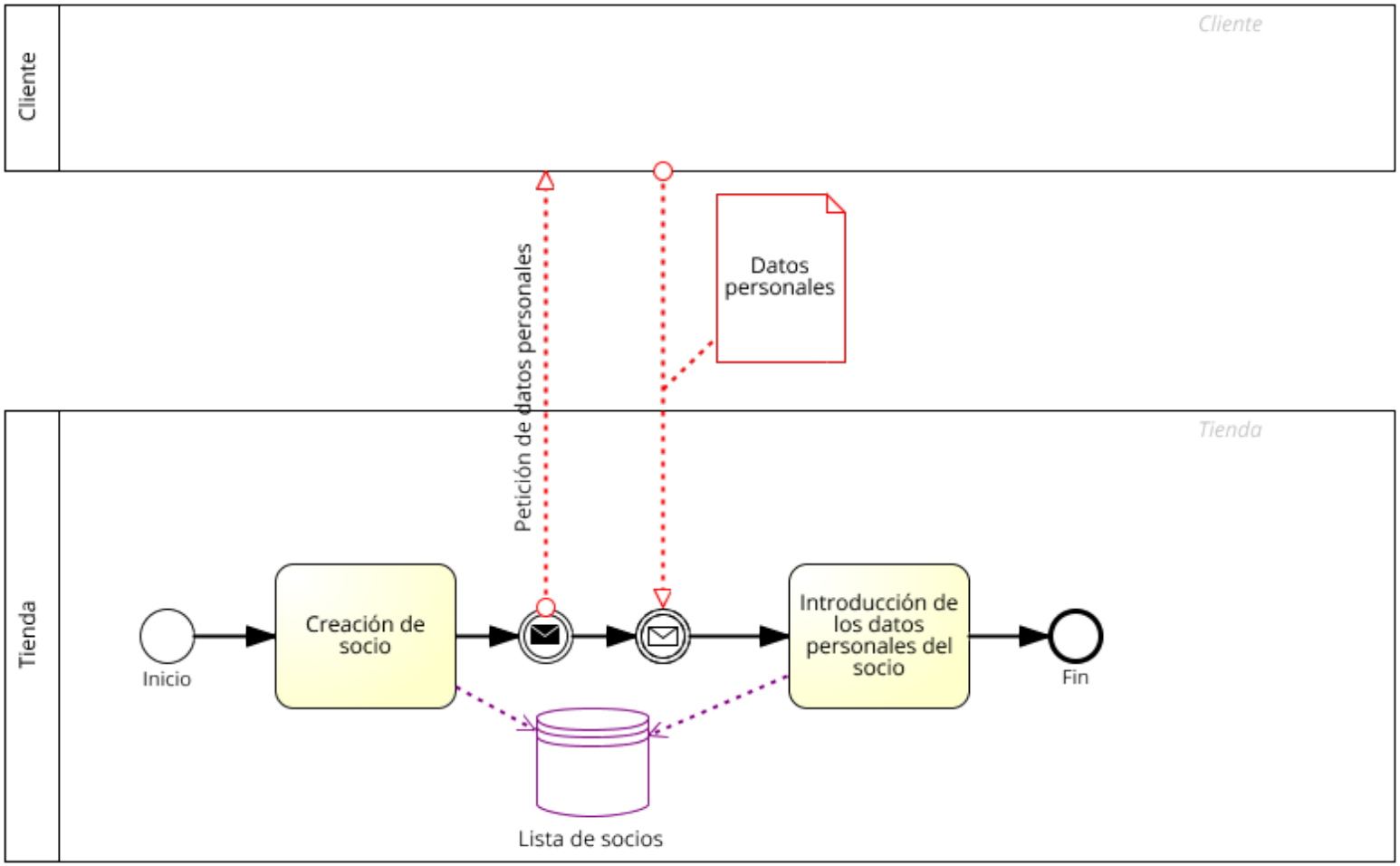


Figura 4: Proceso de creación de socios

4. Visión General del Sistema

4.1. Descripción

Para solucionar los problemas planteados de clientela y gestión, el sistema estará diseñado para permitir la gestión de las ventas, los productos, los pedidos, los traspasos y los clientes de manera más eficaz.

El objetivo principal de nuestro cliente es aumentar su clientela con la ayuda de una página web, esto lo haremos desarrollando un catálogo online de la tienda para que los clientes puedan consultar o comprar cualquier producto en cualquier momento, además de facilitar la gestión interna de la cadena.

Con todo esto principalmente lo que se quiere es aumentar los beneficios y hacer más eficiente la gestión de la cadena.

5. Catálogo de requisitos

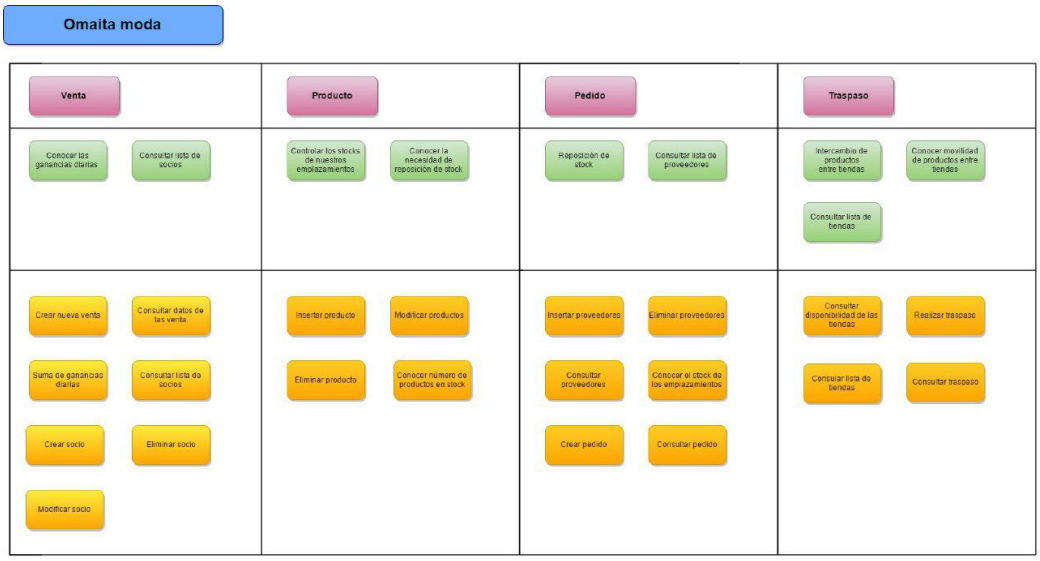


Figura 5: Mapa de requisitos

5.1. Requisitos generales

RG-01 - Gestion ventas

- **Como** Propietario de la cadena
- **Quiero** poder gestionar y consultar las ventas de las tiendas
- **Para** llevar una buena gestión de ellas

RG-02 - Gestion proveedores

- **Como** Propietario de la cadena
- **Quiero** poder gestionar y consultar los pedidos a proveedores
- **Para** llevar una buena gestión de ellos

RG-03 - Traspasos

- **Como** Propietario de la cadena
- **Quiero** que haya un sistema de traspasos
- **Para** poder aprovechar al máximo el stock de las tiendas

---

RG-04 - Catálogo

- **Como** Cliente
- **Quiero** poder ver una lista de los productos ofrecidos y disponibles
- **Para** realizar mis compras

5.2. Requisitos de información

RI-01 - Información sobre ventas

- **Como** Propietario de la cadena
- **Quiero** que el sistema almacene la información correspondiente a las ventas, guardando así los siguientes datos
  - Fecha en la que se llevó acabo la venta.
  - La relación entre producto y venta que almacenará: el producto vendido, el precio y el IVA del mismo en el momento en el que se vendió.
- **Para** conocer los datos de las ventas realizadas.

P-RI-01

1. Cuando se cree una nueva venta, si los datos introducidos son correctos, se almacenará en la lista de ventas.
2. En caso de que algún dato sea incorrecto no se almacenara.

RI-02 - Información sobre facturas

- **Como** propietario de la cadena
- **Quiero** que el sistema almacene la información correspondiente a las facturas, guardando así los siguientes datos:
  - Fecha en la que se tramita la factura.
  - Número de la factura.
  - Un campo llamado devuelto, usado para las devoluciones, si el campo tiene el valor false, entonces podemos contabilizar esa factura sin problemas, si el campo tiene el valor true significará que el dinero fue devuelto al cliente y por la tanto no podríamos contabilizarlo.
- **Para** conocer los datos de las facturas expedidas.

P-RI-02

1. Cuando se cree una nueva factura, si los datos introducidos son correctos, se almacenará en la lista de factura.
2. En caso de que algún dato sea incorrecto no se almacenara.

RI-03 - Información sobre socios

- **Como** propietario de la cadena
- **Quiero** que el sistema almacene la información de los clientes que son socios, guardando los siguientes datos en él:
  - Nombre.
  - Apellidos.
  - DNI.
  - Dirección.
  - Fecha de Nacimiento.
  - Email.
- **Para** llevar un control de los clientes que son socios y así poderles aplicar descuentos en sus compras.

P-RI-03

1. Cuando se cree una nueva factura, si los datos introducidos son correctos, se almacenará en la lista de factura.
2. En caso de que algún dato sea incorrecto (como por ejemplo un DNI duplicado) no se almacenara.

RI-04 - Información sobre productos

- **Como** propietario de la cadena
- **Quiero** que los productos que venda la cadena se guarden con las siguientes características en el sistema:

    - Nombre.
    - Una breve descripción.
    - La categoría del producto: abrigos, chaquetas, camisas, camisetas, jersey, vestidos, faldas, pantalones, calzado, accesorios y bisutería.
    - Precio del producto.
    - IVA actual, para controlar el IVA en todo momento
- **Para** conocer los datos de cada producto.

P-RI-04

1. Cuando se inserte un nuevo producto, si los datos introducidos son correctos, se almacenará en la lista de productos.
2. En caso de que algún dato sea incorrecto (como por ejemplo un precio negativo duplicado) no se almacenara.

RI-05 - Stock del producto

- **Como** propietario de la cadena
- **Quiero** saber el stock de cada producto en cada tienda de la cadena y del almacén
- **Para** controlar el stock de la cadena

P-RI-05

1. Actualizar la lista de stocks cuando se cree uno nuevo o se modifique alguno ya existente.

RI-06 - Información sobre los emplazamientos

- **Como** propietario de la cadena
- **Quiero** guardar la dirección de las tiendas y del almacén de la cadena, además de su teléfono de contacto
- **Para** conocer la localización de las mismas

P-RI-06

1. Cuando se inserte un nuevo emplazamiento, si los datos introducidos son correctos, se almacenará en la lista de emplazamientos.
2. En caso de que algún dato sea incorrecto (como por ejemplo una dirección inexistente) no se almacenara.

RI-07 - Información sobre los proveedores

- **Como** propietario de la cadena
- **Quiero** disponer de la siguiente información sobre proveedores:
  - Nombre.
  - CIF.
  - Número de teléfono.
  - Email.
- **Para** saber a qué proveedores puede realizar los pedidos

P-RI-07

1. Cuando se inserte un nuevo proveedor, si los datos introducidos son correctos, se almacenará en la lista de proveedores.
2. En caso de que algún dato sea incorrecto no se almacenara.

RI-08 - Información sobre los pedidos

- **Como** propietario de la cadena
- **Quiero** que el sistema almacene los pedidos guardando los siguientes datos:
  - Fecha en la que se realiza.
  - Una asociación que controla la cantidad de cada producto del pedido, el precio y el IVA.
- **Para** conocer los datos de los pedidos realizados.

P-RI-08

1. Cuando se inserte un nuevo pedido, si los datos introducidos son correctos, se almacenará en la lista de pedidos.
2. En caso de que algún dato sea incorrecto no se almacenara.

RI-09 - Información sobre traspasos

- **Como** propietario de la cadena
- **Quiero** que el sistema guarde los traspasos almacenando los siguientes datos:
  - Fecha de traspaso.
  - Asociación que controla la cantidad de cada producto en el traspaso.
- **Para** conocer los datos de los traspasos realizados.

P-RI-09

1. Cuando se inserte un nuevo traspaso, si los datos introducidos son correctos, se almacenará en la lista de traspasos.
2. En caso de que algún dato sea incorrecto no se almacenara.

RI-10 - Información sobre el albarán

- **Como** propietario de la cadena
- **Quiero** que el sistema almacene información correspondiente a los albaranes, guardando así los siguientes datos
  - Fecha de firma del albarán.
- **Para** tener un control de los pedidos recibidos los cuales son controlados por los albaranes.

P-RI-10

1. Cuando se inserte un nuevo albarán, si los datos introducidos son correctos, se almacenará en la lista de albaranes.
2. En caso de que algún dato sea incorrecto no se almacenara.

RI-11 - Información sobre solicitud

- **Como** propietario de la cadena
- **Quiero** que el sistema almacene información correspondiente a las solicitudes de traspaso, guardando así los siguientes datos
  - Fecha de solicitud.
  - Asociación que controla la cantidad de cada producto en la solicitud
- **Para** conocer los datos de las solicitudes realizadas.

P-RI-11

1. Cuando se inserte un nuevo pedido, si los datos introducidos son correctos, se almacenará en la lista de pedidos.
2. En caso de que algún dato sea incorrecto no se almacenara.

5.3. Requisitos funcionales

RF-01 - Crear ventas

- **Como** empleado
- **Quiero** poder crear y consultar ventas realizadas
- **Para** poder así conocer las ganancias de la tienda

P-RF-01

1. Crear una nueva venta con nuevos datos y tener la posibilidad de acceder a los datos de las ventas ya realizadas.
2. No se creará la nueva venta si algún dato introducido no es válido.

RF-02 - Actualizar stocks tras venta

- **Como** propietario de la cadena
- **Quiero** que tras una factura asociada a una venta el stock se actualice
- **Para** controlar el stock de manera correcta

P-RF-02

1. Tras realizar una factura el stock se actualiza en función de los parámetros de las ventas asociadas a dicha factura.

RF-03 - Crear socios

- **Como** empleado
- **Quiero** poder crear socios en una lista y poder consultarla
- **Para** llevar así un control sobre estos y saber si se tiene que aplicar o no el descuento.

P-RF-03

1. Crear un socio rellenando un formulario con los datos del cliente en cuestión.
2. Poder acceder a la lista de los socios ya realizados.
3. No se creará el nuevo socio si algún dato introducido no es válido.

RF-04 - Modificar socios

- **Como** empleado
- **Quiero** poder modificar la dirección y el email de un socio ya creado
- **Para** tener los datos correctos del socio.

P-RF-04

1. Cuando se cambie algún dato de un socio, este se actualizará en la lista de socios.

2. o se efectuará la actualización si alguno de los datos introducidos para el cambio no es valido.

RF-05 - Eliminar socios

- **Como** empleado
- **Quiero** poder eliminar un socio de la base de datos
- **Para** dejar de tener almacenados los datos de un cliente que desiste de su derecho de ser socio.

P-RF-05

1. Cuando se elimine un socio, la lista de socios se actualizará de forma de que el socio eliminado ya no aparezca en esta.

RF-06 - Crear, modificar y eliminar productos

- **Como** empleado
- **Quiero** poder insertar, modificar la descripción, el precio y el IVA de un producto, o eliminar un producto de mi base de datos
- **Para** poder así llevar una buena gestión de los productos.

P-RF-06

1. Insertar en la lista de productos un nuevo producto rellenando un formulario con los datos de este, y tras ello se actualizará dicha lista apareciendo en ella el nuevo producto.

2. Cuando se cambie algún dato de un producto existente, este se actualizará en la lista de productos.

3. Cuando se elimine un producto, la lista de productos se actualizará de forma de que el producto eliminado ya no aparezca en esta.

4. Si algunos de los datos introducidos a la hora de insertar o actualizar un producto es incorrecto no se realizará la operación.

RF-07 - Crear y modificar stocks

- **Como** empleado
- **Quiero** poder crear el stock de un producto y modificar su cantidad
- **Para** llevar un control sobre la disponibilidad de los productos según su emplazamiento.

P-RF-07

1. Cuando llegue un producto nuevo a la tienda se creará un stock de este con la cantidad que llegue del mismo.
2. Cuando se realice cualquier interacción con un producto de tal forma que modifique la cantidad que existe en la tienda del mismo, se actualizara el stock.
3. No se creará el nuevo stock si algún dato introducido no es válido, o si el producto al que se quiere realizar un stock ya posee uno en la tienda en cuestión.

RF-08 - Crear, modificar y eliminar proveedores

- **Como** propietario de la cadena
- **Quiero** poder insertar, modificar el nombre, el email y el teléfono, o eliminar proveedores en mi base de datos
- **Para** conocer los proveedores disponibles y tener sus datos actualizados.

P-RF-08

1. Insertar en la lista de proveedores un nuevo proveedor rellenando un formulario con los datos de este, y tras ello se actualizará dicha lista apareciendo en ella el nuevo proveedor.

2. Cuando se cambie algún dato de un proveedor existente, este se actualizará en la lista de proveedores.
3. Cuando se elimine un proveedor, la lista de proveedores se actualizará de forma de que el proveedor eliminado ya no aparezca en esta.
4. Si algunos de los datos introducidos a la hora de insertar o actualizar un producto es incorrecto no se realizará la operación.

RF-09 - Crear pedidos

- **Como** empleado
- **Quiero** poder crear pedidos
- **Para** tener constancia de todos los pedidos realizados por cada uno de mis emplazamientos

P-RF-09

1. Crear un nuevo pedido con los productos que sean necesario reponer.
2. No se podrá realizar un pedido si algún dato introducido no es válido.

RF-10 - Actualizar stock tras pedido

- **Como** propietario de la cadena
- **Quiero** que tras recibir el albarán que confirma la entrega del pedido el stock se actualice
- **Para** controlar correctamente el stock.

P-RF-10

1. Tras recibir el pedido realizado, se actualizará la cantidad del stock de los productos del pedido.

RF-11 - Crear solicitud de traspaso

- **Como** empleado

- **Quiero** poder crear una solicitud de traspaso
- **Para** así poder pedir a otra tienda productos que necesite.

P-RF-11

1. Poder realizar una solicitud de traspaso a otra tienda de un producto que sea necesario y esté disponible en la otra tienda.

RF-12 - Crear traspasos

- **Como** empleado
- **Quiero** poder crear traspasos
- **Para** responder a la necesidad de las solicitudes de traspaso.

P-RF-12

1. Responder a la otra tienda siempre que sea posible cumplir con sus peticiones de solicitud de traspaso, creando en el acto un traspaso con los datos de los productos traspasados.

RF-13 - Actualizar stock tras traspaso

- **Como** propietario de la cadena
- **Quiero** que cuando se realice un traspaso, se modifiquen de manera correcta los stocks de las tiendas implicadas
- **Para** así poder tener una buena gestión sobre nuestros stocks

P-RF-13

1. Cuando un traspaso se realiza de manera correctas ambas tiendas implicadas en el traspaso deberán actualizar la cantidad del stock de los productos implicados en el traspaso.

RF-14 - Consultar traspasos

- **Como** propietario de la cadena

- **Quiero** poder consultar todos los traspasos realizados por mis emplazamientos
- **Para** conocer la movilidad de los productos entre estos.

P-RF-14

1. Siempre se tendrá la posibilidad de acceder a los datos de los traspasos en los que esté implicado mi tienda.

RF-15 - Crear y eliminar emplazamientos

- **Como** propietario de la cadena
- **Quiero** poder añadir o eliminar emplazamientos en la base de datos
- **Para** saber que emplazamientos están en la cadena.

P-RF-15

1. Insertar en la lista de emplazamientos un nuevo emplazamiento rellenando un formulario con los datos de este, y tras ello se actualizará dicha lista apareciendo en ella el nuevo emplazamiento.
2. Cuando se elimine un emplazamiento, la lista de emplazamientos se actualizará de forma de que el emplazamiento eliminado ya nTras una devolución se marcará en la factura como que se ha realizado dicha acción modificando el campo correspondiente.
3. En el caso de que la devolución solo sea parcial y no total se creara una copia de la factura donde solo aparezcan los productos no devueltos y la antigua se marcara como devuelta.
4. En ambos casos se actualizará el stock de los productos devueltos.
5. Si se intenta realizar la devolución de una factura, la cual haya excedido el tiempo de devolución, no se realizará dicha devolución.o aparezca en esta.

RF-16 - Modificar emplazamientos

- **Como** propietario de la cadena
- **Quiero** poder modificar la dirección y el número de teléfono de un emplazamiento
- **Para** tener los datos actualizados de mis emplazamientos.

P-RF-16

1. Cuando se cambie algún dato de un emplazamiento existente, este se actualizará en la lista de emplazamientos.
2. Si al modificar algún emplazamiento se introduce algún dato no valido, no se actualizará la lista de emplazamientos.

RF-17 - Devolución

- **Como** propietario de la cadena
- **Quiero** que cuando se realiza una devolución el campo devuelto de las facturas pase a ser True, y si es necesario que el empleado cree de nuevo toda la venta y la factura pero sin los productos devueltos, esta factura deberá tener el campo devuelto como False
- **Para** tener un control sobre las devoluciones.

P-RF-17

1. Tras una devolución se marcará en la factura como que se ha realizado dicha acción modificando el campo correspondiente.
2. En el caso de que la devolución solo sea parcial y no total se creara una copia de la factura donde solo aparezcan los productos no devueltos y la antigua se marcara como devuelta.
3. En ambos casos se actualizará el stock de los productos devueltos.
4. Si se intenta realizar la devolución de una factura, la cual haya excedido el tiempo de devolución, no se realizará dicha devolución.

RF-18 - Crear albarán

- **Como** empleado
- **Quiero** crear una copia del albarán de entrega que me ha dado el proveedor al traer el pedido que habíamos hecho
- **Para** tener la información de los albaranes recogida en la base de datos.

P-RF-18

1. Cuando llegue cualquier pedido se guardará una copia de los albaranes

5.4. Requisitos no funcionales

RNF-01 - Acceso al sistema

- **Como** propietario de la cadena
- **Quiero** que todos mis empleados tengan acceso al sistema
- **Para** facilitar la gestión de las tiendas y el control del sistema.

RNF-02 - Protección de datos socios

- **Como** propietario de la cadena
- **Quiero** que los datos de los socios permanezcan privados y seguros
- **Para** cumplir la Ley de Protección de Datos

RNF-03 - Mantenimiento

- **Como** propietario de la cadena
- **Quiero** realizar el mantenimiento del sistema al menos una vez al mes
- **Para** prevenir problemas mayores en el sistema.

5.5. Reglas de negocio

RN-01 - Descuento a socios

- **Como** propietario de la cadena
- **Quiero** aplicar un 5 % de descuento en las ventas realizadas a socios
- **Para** recompensar su fidelidad

P-RN-01

1. Cuando se realice una venta, si esta está asociada a un socio , se aplicara el descuento en la factura.

RN-02 - Tamaño mínimo de pedidos

- **Como** propietario de la cadena quiero
- **Quiero** que los pedidos sean como mínimo de 20 unidades por producto
- **Para** abaratar gastos de transporte.

P-RN-02

1. Cuando se realice un pedido, si la cantidad de algún producto no supera el mínimo no se realizará dicho pedido, en caso contrario se realizara sin ningún problema.

RN-03 - Evitar traspaso si se provoca stock mínimo

- **Como** propietario de la cadena
- **Quiero** que las tiendas no puedan ofrecer el traspaso de un producto, si debido al traspaso el producto llega a su stock mínimo
- **Para** evitar que éstas se queden desabastecidas

P-RN-03

1. No se podrá responder a una solicitud de traspaso si en caso de realizar el traspaso el producto solicitado se nos quedaría en stock mínimo

---

RN-04 - Política de devoluciones

- **Como** propietario de la cadena
- **Quiero** que solo se acepten devoluciones, con un plazo máximo de 30 días después de la compra
- **Para** dificultar la devolución de productos usados

P-RN-04

1. Si se intenta realizar una devolución dentro del plazo de devolución establecido por la tienda (en nuestro caso 30 días a partir del día de la compra) se permite realizar dicha devolución ya sea total o parcial.
2. En caso contrario no se podrá realizar la devolución.

RN-05 - Aviso stock mínimo

- **Como** trabajador de una tienda
- **Quiero** obtener un aviso cuando el stock de un producto llegue al stock mínimo
- **Para** evitar el desabastecimiento de un producto

P-RN-05

1. Cuando un producto este bajo el stock mínimo saltara un mensaje de aviso para tener en cuenta su futura reposición

6. Modelo Conceptual

6.1. Diagramas de clases UML

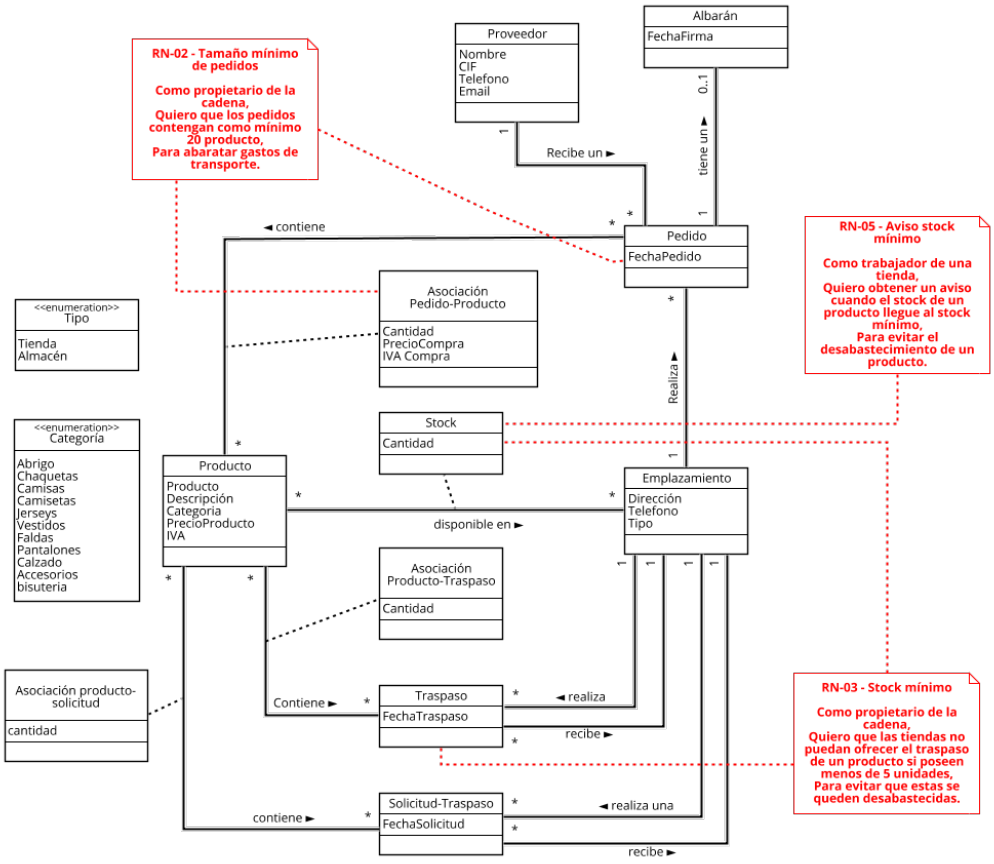


Figura 6: UML Traspaso / Pedido

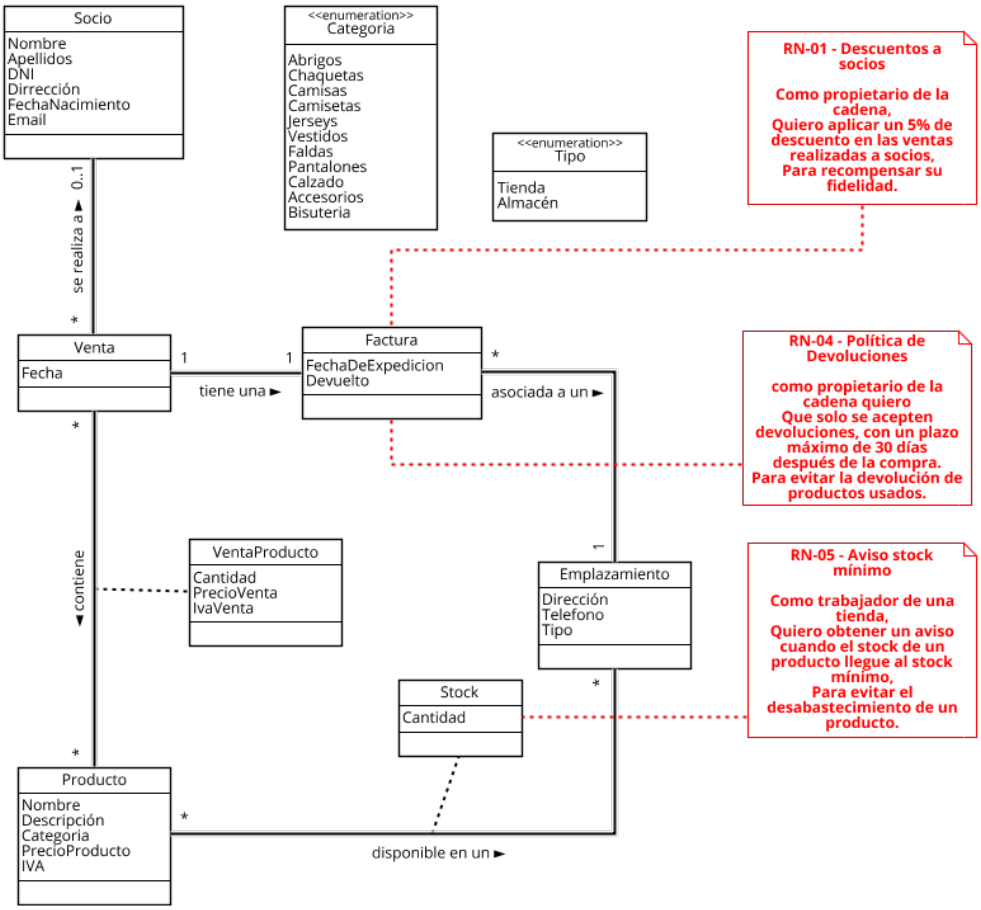


Figura 7: UML Venta

6.2. Escenarios de prueba

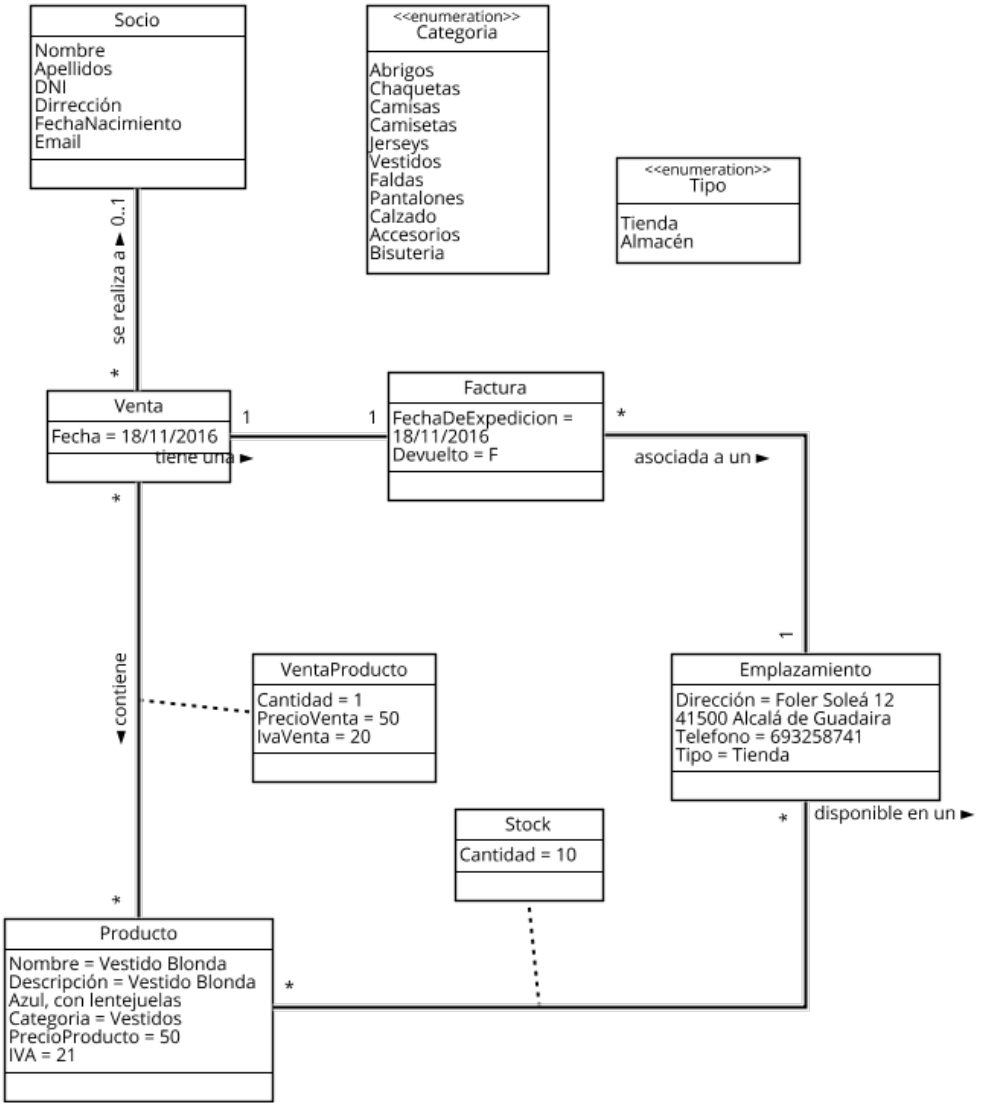


Figura 8: UML Venta NO Socio

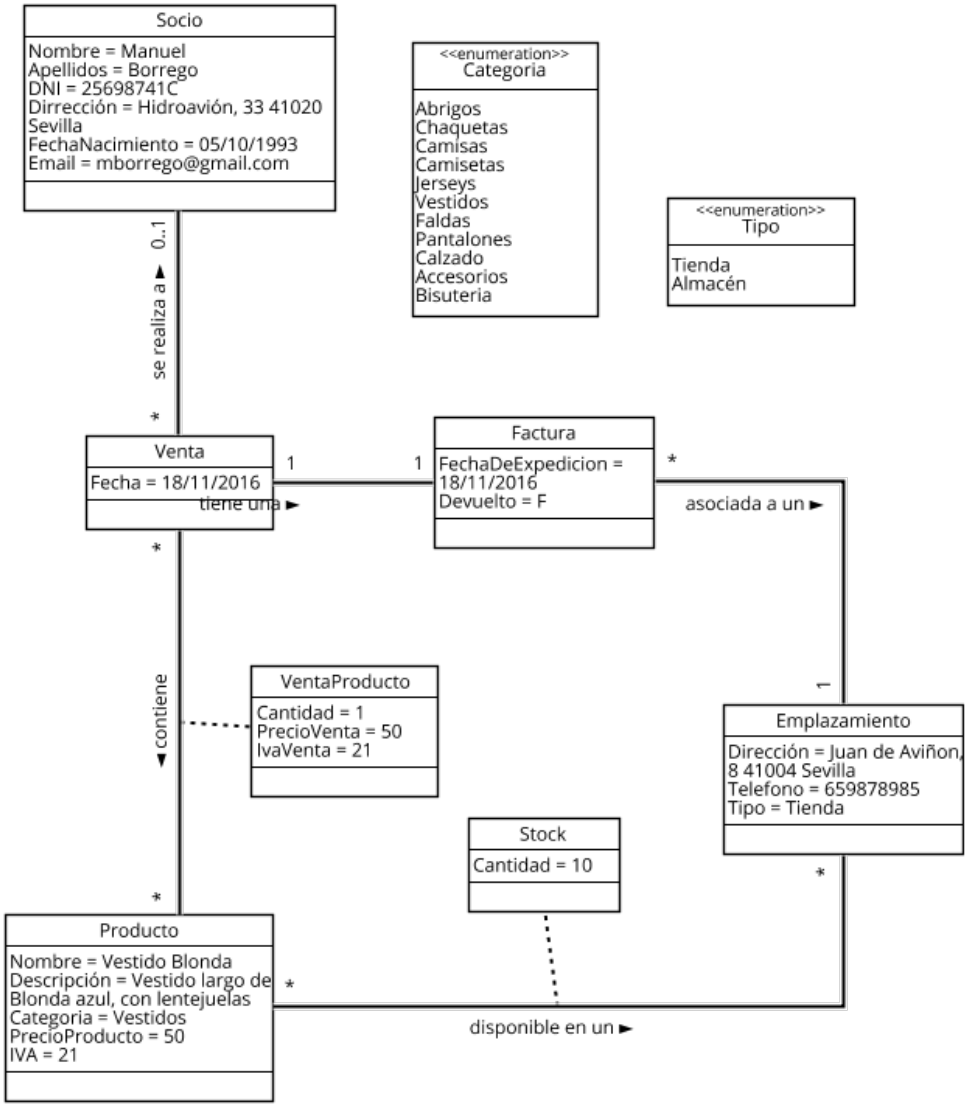


Figura 9: UML Venta Socio

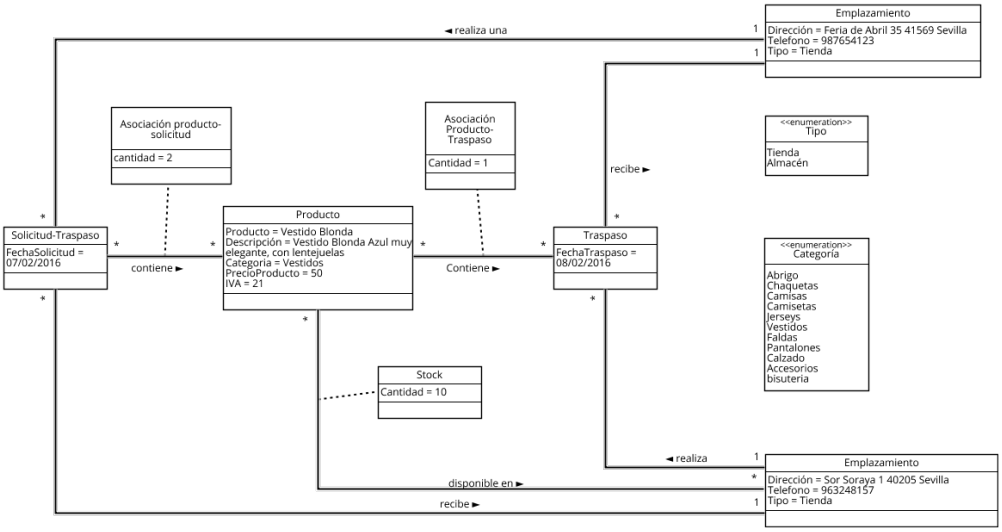


Figura 10: UML Traspaso Tienda - Tienda

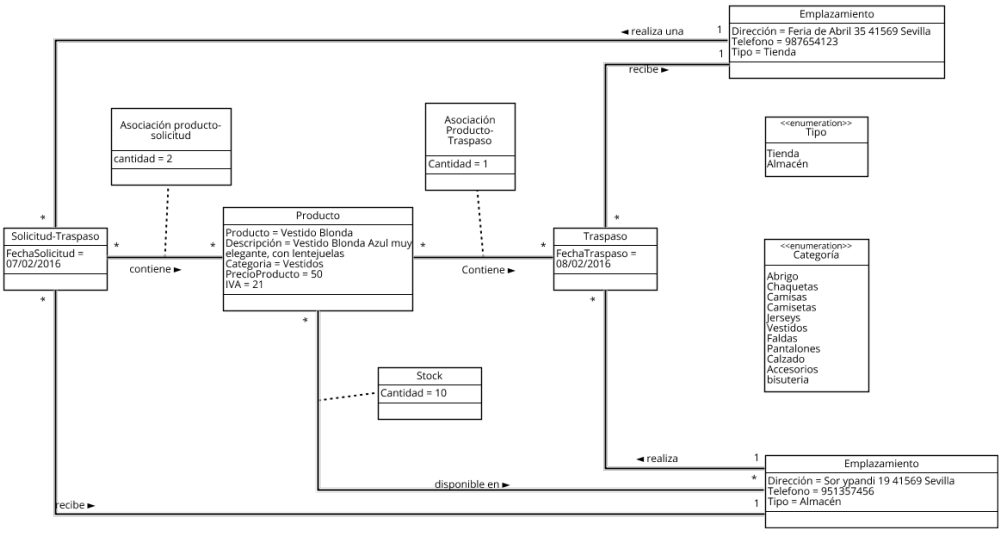


Figura 11: UML Traspaso Tienda - Almacén

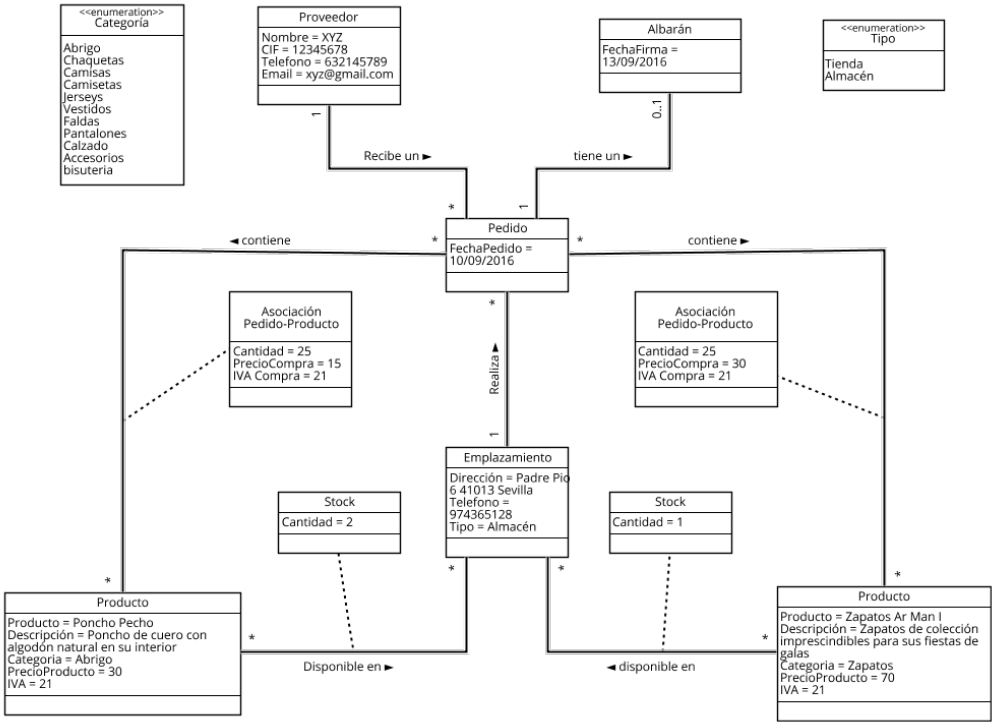


Figura 12: UML Pedido Tienda - Proveedor

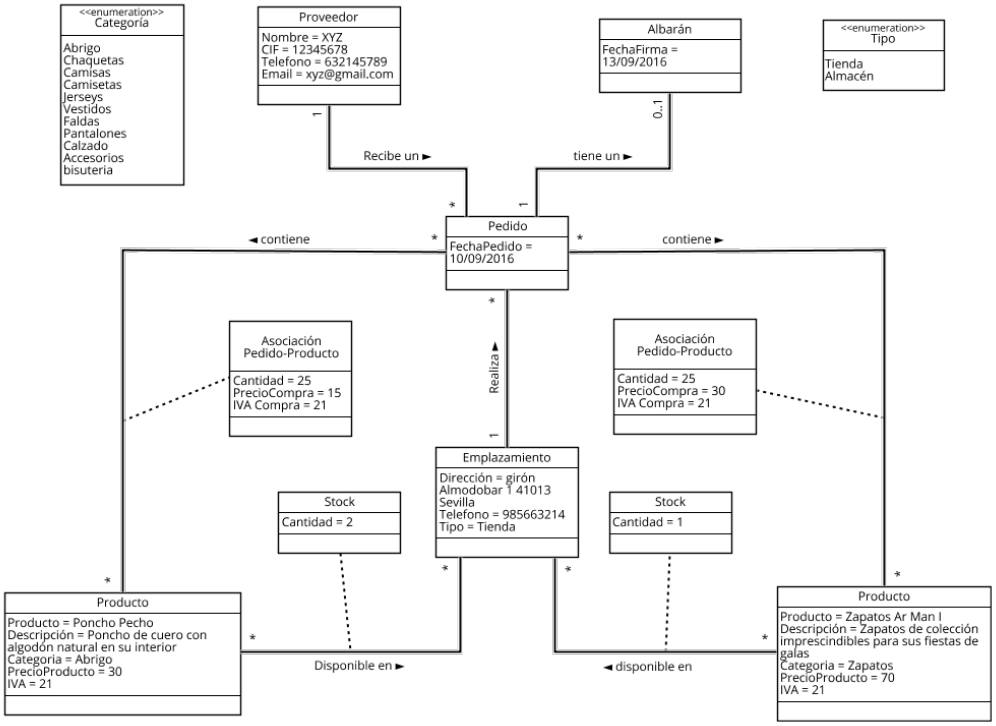


Figura 13: UML Pedido Almacén - Proveedor

7. Matrices de trazabilidad

7.1. Pruebas de aceptación frente a requisitos

Cuadro 1: **Requisitos funcionales**

P-RF-01	X																
P-RF-02		X															
P-RF-03			X														
P-RF-04				X													
P-RF-05					X												
P-RF-06						X											
P-RF-07							X										
P-RF-08								X									
P-RF-09									X								
P-RF-10										X							
P-RF-11											X						
P-RF-12												X					
P-RF-13													X				
P-RF-14														X			
P-RF-15															X		
P-RF-16																X	
P-RF-17																	X
P-RF-18																	
	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**

Cuadro 2: **Requisitos de información**

P-RI-01	X										
P-RI-02		X									
P-RI-03			X								
P-RI-04				X							
P-RI-05					X						
P-RI-06						X					
P-RI-07							X				
P-RI-08								X			
P-RI-09									X		
P-RI-10										X	
P-RI-11											X
	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**

Cuadro 3: **Reglas de negocio**

P-RN-01	X				
P-RN-02		X			
P-RN-03			X		
P-RN-04				X	
P-RN-05					X
	01	**02**	**03**	**04**	**05**

7.2. Pruebas de aceptación frente a escenarios de prueba

Cuadro 4: **Escenarios**

Escenario 01	X	X	X	X	X	X					
Escenario 02	X	X	X	X	X	X					
Escenario 03				X	X	X			X		
Escenario 04				X	X	X			X		
Escenario 05				X	X	X	X	X		X	X
Escenario 06				X	X	X	X	X		X	X
	P-RI-01	**P-RI-02**	**P-RI-03**	**P-RI-04**	**P-RI-05**	**P-RI-06**	**P-RI-07**	**P-RI-08**	**P-RI-09**	**P-RI-10**	**P-RI-11**

Cuadro 5: **Escenarios**

Escenario 01	X	X				X									X	X	X	
Escenario 02	X	X	X	X	X	X									X	X	X	
Escenario 03						X					X	X	X	X	X	X		
Escenario 04						X	X				X	X	X	X	X	X		
Escenario 05						X	X	X	X	X					X	X		X
Escenario 06						X	X	X	X	X					X	X		X
	P-RF-01	**P-RF-02**	**P-RF-03**	**P-RF-04**	**P-RF-05**	**P-RF-06**	**P-RF-07**	**P-RF-08**	**P-RF-09**	**P-RF-10**	**P-RF-11**	**P-RF-12**	**P-RF-13**	**P-RF-14**	**P-RF-15**	**P-RF-16**	**P-RF-17**	**P-RF-18**

Cuadro 6: **Escenarios**

Escenario 01	X			X	X
Escenario 02	X			X	X
Escenario 03			X		X
Escenario 04			X		X
Escenario 05		X			X
Escenario 06		X			X
	P-RN-01	**P-RN-02**	**P-RN-03**	**P-RN-04**	**P-RN-05**

7.3. Tipos de UML frente a Requisitos

Cuadro 7: **Requisitos de informacion**

Venta	x	x	x	x	x	x					
Traspaso - Pedido				x	x	x	x	x	x	x	x
	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**

Cuadro 8: **Reglas de negocio**

Venta	x			x	x
Traspaso - Pedido		x	x		x
	01	**02**	**03**	**04**	**05**

Cuadro 9: **Requisitos Funcionales**

Venta	X	X	X	X	X	X	X								X	X	X	
Traspaso - Venta						X	X	X	X	X	X	X	X	X	X	X		X
	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**

8. Modelos relacionales

Para poder pasar de UML(MC) a 3FN(MR), tenemos que ser capaces de normalizar tanto en 2FN como en 1FN, esto lo hemos comprobado de la siguiente manera: En cada tupla de la 3FN se le asigna a cada atributo un solo valor del dominio sobre el que está definido, es decir, no existen atributos con varios valores. Esto prueba que está en 1FN. Tras asegurar la 1FN, pasamos a la comprobación de la 2FN, para validar esta normalización tenemos que tener en cuenta que los atributos que no sean Primary Key o Foreign Key (atributos no primos) deben de ser dependientes de estas claves, esto lo cumplimos gracias a las foreign keys y primary keys creadas para las asociaciones, esta relación está explicada más adelante. Finalmente llegamos a la comprobación de la 3FN, para esta comprobación tenemos que tener en cuenta que la relación tiene que estar en 2FN y que la relación de tablas simplemente se haga mediante las primary y las foreign key, todo lo que no sea claves candidatas no puede estar en varias entidades asociadas a la vez.

8.1. 3FN Venta

Socio Los atributos son directos en la 3FN, añadiendo que DNI será una primray key. Tiene una relación 0..1 - n con Venta, venta mantiene todos sus atributos de manera directa en la 3FN, añadiendo un ID_VENTA como primary key y debido a la relación con socio una foreign key llamada DNI.

Venta Tiene una relación n - m con Producto, se crea una tabla intermedia debido a esta relación que es la tabla VentaProducto, con los siguientes atributos: cantidad, precioVenta e ivaVenta. Además tiene ID_VENTA y ID_PRODUCTO como primary keys y foreigns keys, estas claves son provocadas por la relación n - m entre Venta y Producto.

Producto La tabla producto mantiene todos sus atributos de manera directa en la 3FN y añade ID_PRODUCTO que será su primary key.

Venta Tiene una relación 1 - 1 con Factura, factura mantiene todos sus atributos de manera directa en la 3FN y añade un ID_FACTURA como primary key, un ID_VENTA como foreign key debido a la relación con Venta y un ID_EMPLAZAMIENTO debido a la relación con Emplazamiento.

Factura tiene una relación n - 1 con Emplazamiento, emplazamiento mantiene todos sus atributos de manera directa en la 3FN y añade un ID_EMPLAZAMIENTO como primary key.

Producto Tiene una relación n - m con Emplazamiento, esto da lugar a la creación de una tabla intermedia, para poder tratar la relación n - m, con los siguientes atributos: cantidad, ID_PRODUCTO(primary key y foreign key) y ID_EMPLAZAMIENTO(primary key y foreign key). Estas claves son provocadas por la relacion n - m entre Producto y Emplazamiento.

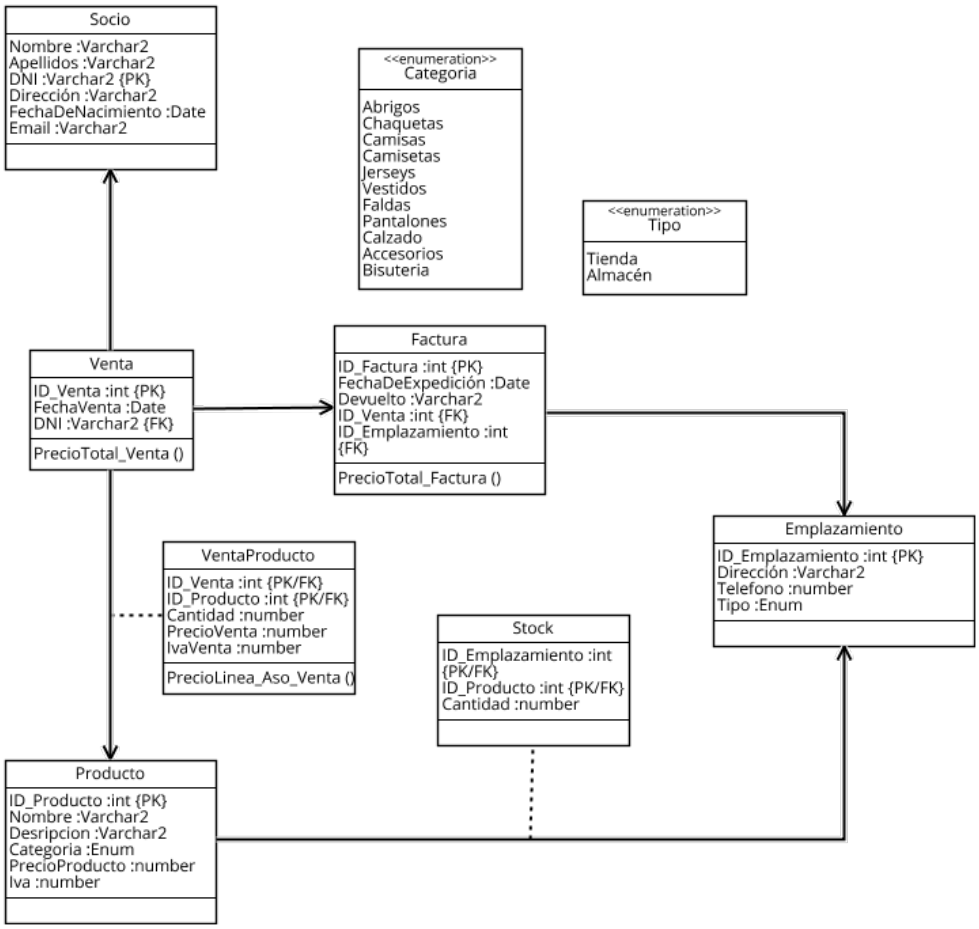


Figura 14: 3FN Venta

8.2. 3FN Traspaso - Pedido

Proveedor Tiene una relación 1 - n con Pedido, proveedor mantiene todos sus atributos de manera directa en la 3FN y añade un ID_PROVEEDOR como primary key.

Emplazamiento Tiene una relación 1 - n con Pedido, pedido mantiene todos sus atributos de manera directa en la 3FN, añade un ID_PEDIDO como primary

key y un ID_PROVEEDOR como foreign key debido a la relación con Proveedor.

Pedido Tiene una relación n - m con Producto, se crea una tabla intermedia debido a esta relación que es la tabla Asociación Pedido-Producto, con los siguientes atributos: cantidad, precioCompra, ivaCompra. Además tiene ID_PEDIDO y ID_PRODUCTO como primary keys y foreigns keys, estas claves son provocadas por la relación n - m entre Pedido y Producto.

Albarán Tiene una relación 1 - 0..1 con Pedido, albarán mantiene todos sus atributos de manera directa en la 3FN y añade un ID_ALBARAN como primary key.

Emplazamiento Tiene una relación 1 - m bidireccional con Traspaso, traspaso mantiene todos sus atributos de manera directa en la 3FN y añade un ID_TRASPASO.

Emplazamiento Tiene una relación 1 - m bidireccional con Solicitud-Trapaso, solicitud-traspaso mantiene todos sus atributos de manera directa en la 3FN y añade un ID_Solicitud.

Traspaso Tiene una relación n - m con Producto, esto da lugar a la creación de una tabla intermedia, para poder tratas la relación n-m, con los siguientes atributos: cantidad, ID_Producto(primary key y foreign key) y ID_Traspaso(primary key y foreign key). Estas claves son provocadas por la relación n - m entre Traspaso y Producto.

Solicitud-Trapaso Tiene una relación n - m con Producto, esto da lugar a la creación de una tabla intermedia, para poder tratas la relación n-m, con los siguientes atributos: cantidad, ID_Producto(primary key y foreign key) y ID_SOLICITUD(primary key y foreign key). Estas claves son provocadas por la relación n - m entre Solicitud-Traspaso y Producto.

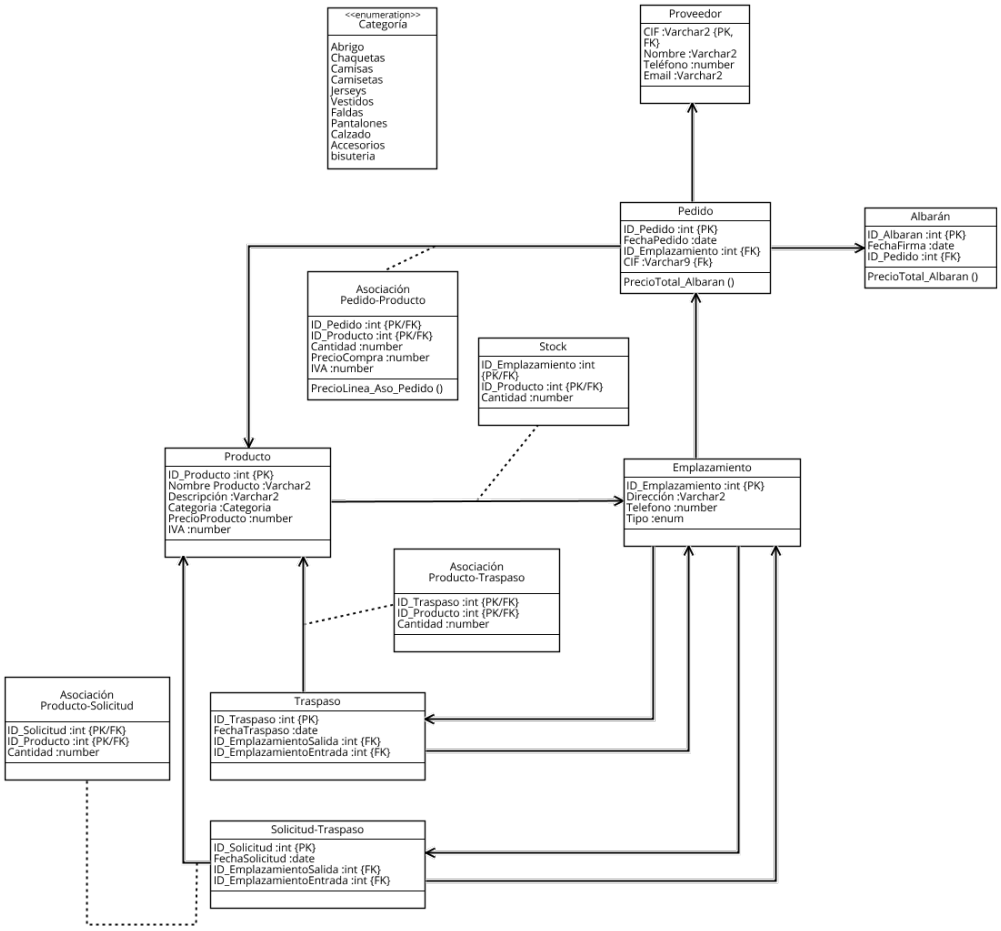


Figura 15: 3FN Traspaso - Pedido

9. Código SQL de la base de datos

9.1. Tablas

```
DROP TABLE ALBARAN;
DROP TABLE ASOCIACION_PRODUCTO_TRASPASO;
DROP TABLE ASOCIACION_PRODUCTO_SOLICITUD;
DROP TABLE ASOCIACION_PEDIDO_PRODUCTO;
DROP TABLE ASOCIACION_VENTA_PRODUCTO;
DROP TABLE FACTURA;
DROP TABLE VENTA;
DROP TABLE PEDIDO;
```

```
DROP TABLE PROVEEDOR;
DROP TABLE STOCK;
DROP TABLE PRODUCTO;
DROP TABLE TRASPASO;
DROP TABLE SOLICITUD_TRASPASO;
DROP TABLE SOCIO;
DROP TABLE EMPLAZAMIENTO;


CREATE TABLE EMPLAZAMIENTO(
  ID_Emplazamiento int PRIMARY KEY,
  Direccion VARCHAR2(100) NOT NULL,
  Telefono NUMBER(9),
  Tipo VARCHAR2(10) CHECK( Tipo IN('TIENDA','ALMACEN'))
);

CREATE TABLE SOCIO (
  Nombre VARCHAR2(25) NOT NULL,
  Apellidos VARCHAR2(50) NOT NULL,
  Direccion VARCHAR2(200) NOT NULL,
  FechaDeNacimiento DATE NOT NULL,
  Email VARCHAR2(50) NOT NULL,
  DNI VARCHAR2(9) PRIMARY KEY
);

CREATE TABLE VENTA (
  ID_VENTA int PRIMARY KEY,
  FechaVenta DATE NOT NULL,
  DNI VARCHAR2(9),
  FOREIGN KEY(DNI) REFERENCES SOCIO
);

CREATE TABLE FACTURA (
  ID_FACTURA int PRIMARY KEY,
  FechaDeExpedicion DATE DEFAULT SYSDATE,
  Devuelto VARCHAR2(1) CHECK( Devuelto IN('F','T')),
  ID_Venta int,
  ID_Emplazamiento int,
  FOREIGN KEY (ID_Venta) REFERENCES VENTA,
  FOREIGN KEY (ID_Emplazamiento) REFERENCES Emplazamiento
);

CREATE TABLE PROVEEDOR (
    CIF VARCHAR2(9) PRIMARY KEY,
    Nombre VARCHAR2(75) NOT NULL,
    Telefono NUMBER(9) NOT NULL,
    Email VARCHAR2(50) NOT NULL
);

CREATE TABLE PEDIDO(
ID_Pedido int PRIMARY KEY,
FechaPedido DATE NOT NULL,
ID_Emplazamiento int,
CIF VARCHAR2(9),
FOREIGN KEY (CIF) REFERENCES PROVEEDOR,
FOREIGN KEY (ID_Emplazamiento) REFERENCES EMPLAZAMIENTO
);

CREATE TABLE ALBARAN (
ID_Albaran int NOT NULL,
FechaFirma DATE NOT NULL,
ID_Pedido INT PRIMARY KEY,
FOREIGN KEY (ID_Pedido) REFERENCES PEDIDO);


CREATE TABLE PRODUCTO(
    ID_Producto int PRIMARY KEY,
    Nombre VARCHAR2(50) NOT NULL,
    Descripcion VARCHAR2(300) NOT NULL,
    Categoria VARCHAR2(20) CHECK ( Categoria IN ('Abrigos','Chaquetas','Camisas'
                            ,'Camisetas','Jerseys','Vestidos','Faldas'
                            ,'Pantalones','Calzado','Accesorios','Bisuteria')),
    PrecioProducto NUMBER NOT NULL check(PrecioProducto>=0),
    IVA NUMBER NOT NULL check(IVA>=0 AND IVA<=1)
);

CREATE TABLE STOCK(
```

```
    ID_Emplazamiento int,
    ID_Producto int,
    PRIMARY KEY (ID_Emplazamiento, ID_Producto),
    Cantidad NUMBER(6) NOT NULL check(Cantidad>=0),
    FOREIGN KEY (ID_Emplazamiento) REFERENCES EMPLAZAMIENTO,
    FOREIGN KEY (ID_Producto) REFERENCES PRODUCTO
);

CREATE TABLE ASOCIACION_PEDIDO_PRODUCTO(
    ID_PEDIDO int,
    ID_PRODUCTO int,
    PRIMARY KEY (ID_PEDIDO, ID_PRODUCTO),
    Cantidad NUMBER(10) NOT NULL check(Cantidad>=20),
    PrecioCompra NUMBER NOT NULL check(PrecioCompra>=0),
    IVA NUMBER NOT NULL check(IVA>=0 AND IVA<=1),
    FOREIGN KEY (ID_PEDIDO) REFERENCES PEDIDO,
    FOREIGN KEY (ID_PRODUCTO) REFERENCES PRODUCTO
);

CREATE TABLE TRASPASO(
    ID_Traspaso int PRIMARY KEY,
    FechaTraspaso DATE NOT NULL,
    ID_EmplazamientoSalida int,
    ID_EmplazamientoEntrada int,
    FOREIGN KEY(ID_EmplazamientoSalida) REFERENCES EMPLAZAMIENTO,
    FOREIGN KEY(ID_EmplazamientoEntrada) REFERENCES EMPLAZAMIENTO
);

CREATE TABLE ASOCIACION_PRODUCTO_TRASPASO(
    ID_Traspaso int,
    ID_Producto int,
    PRIMARY KEY (ID_Producto, ID_Traspaso),
    Cantidad NUMBER(10) NOT NULL check(Cantidad>=0),
    FOREIGN KEY(ID_Traspaso) REFERENCES TRASPASO,
    FOREIGN KEY(ID_Producto) REFERENCES producto
);



CREATE TABLE SOLICITUD_TRASPASO(
    ID_Solicitud int PRIMARY KEY,
    FechaSolicitud DATE NOT NULL,
    ID_EmplazamientoSalida int,
    ID_EmplazamientoEntrada int,
    FOREIGN KEY(ID_EmplazamientoSalida) REFERENCES EMPLAZAMIENTO,
    FOREIGN KEY(ID_EmplazamientoEntrada) REFERENCES EMPLAZAMIENTO
);

CREATE TABLE ASOCIACION_PRODUCTO_SOLICITUD(
    ID_Solicitud int,
    ID_Producto int,
    PRIMARY KEY (ID_Producto, ID_Solicitud),
    Cantidad NUMBER(10) NOT NULL check(Cantidad>=0),
    FOREIGN KEY(ID_Solicitud) REFERENCES SOLICITUD_TRASPASO,
    FOREIGN KEY(ID_Producto) REFERENCES producto
);
CREATE TABLE ASOCIACION_VENTA_PRODUCTO(
    ID_Venta int,
    ID_Producto int,
    PRIMARY KEY (ID_Venta, ID_Producto),
    FOREIGN KEY (ID_Venta) REFERENCES VENTA,
    FOREIGN KEY (ID_Producto) REFERENCES PRODUCTO,
    Cantidad NUMBER(6) check(Cantidad>=0),
    PrecioVenta NUMBER check(PrecioVenta>=0),
    IvaVenta NUMBER check(IvaVenta>=0 AND IvaVenta<=1)
);

/* SECUENCIAS */

DROP SEQUENCE S_ID_Producto;
DROP SEQUENCE S_ID_Traspaso;
DROP SEQUENCE S_ID_Solicitud;
DROP SEQUENCE S_ID_Pedido;
DROP SEQUENCE S_ID_Albaran;
DROP SEQUENCE S_ID_Factura;
DROP SEQUENCE S_ID_Emplazamiento;
```

```
DROP SEQUENCE S_ID_Venta;



CREATE SEQUENCE S_ID_Emplazamiento START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Venta START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Pedido START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Albaran START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Producto START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Traspaso START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Solicitud START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Factura START WITH 1 INCREMENT BY 1 MAXVALUE 9999;

CREATE OR REPLACE TRIGGER crea_ID_Pedido
BEFORE INSERT ON Pedido
FOR EACH ROW
BEGIN
SELECT S_ID_Pedido.NEXTVAL INTO:NEW.ID_Pedido FROM DUAL;
END crea_ID_Pedido;

/

CREATE OR REPLACE TRIGGER crea_ID_Albaran
BEFORE INSERT ON ALBARAN
FOR EACH ROW
BEGIN
SELECT S_ID_Albaran.NEXTVAL INTO:NEW.ID_Albaran FROM DUAL;
END crea_ID_Albaran;

/

CREATE OR REPLACE TRIGGER Crea_ID_Producto
BEFORE INSERT ON PRODUCTO
FOR EACH ROW
BEGIN
SELECT S_ID_Producto.NEXTVAL INTO:NEW.ID_Producto FROM DUAL;
END Crea_Nuevo_Producto;

/

CREATE OR REPLACE TRIGGER Crea_ID_Traspaso
BEFORE INSERT ON TRASPASO
FOR EACH ROW
BEGIN
SELECT S_ID_Traspaso.NEXTVAL INTO:NEW.ID_Traspaso FROM DUAL;
END Crea_Nuevo_Traspaso;

/

CREATE OR REPLACE TRIGGER Crea_ID_Solicitud
BEFORE INSERT ON SOLICITUD_TRASPASO
FOR EACH ROW
BEGIN
SELECT S_ID_Solicitud.NEXTVAL INTO:NEW.ID_Solicitud FROM DUAL;
END Crea_Nuevo_Solicitud;

/

CREATE OR REPLACE TRIGGER crea_ID_Factura
BEFORE INSERT ON FACTURA
    FOR EACH ROW
        BEGIN
        SELECT S_ID_Factura.NEXTVAL INTO:NEW.ID_Factura FROM DUAL;
END crea_ID_Factura;

/

CREATE OR REPLACE TRIGGER crea_ID_Emplazamiento
  BEFORE INSERT ON EMPLAZAMIENTO
  FOR EACH ROW
  BEGIN
    SELECT S_ID_Emplazamiento.NEXTVAL
    INTO :NEW.ID_Emplazamiento FROM DUAL;
END crea_ID_Emplazamiento;
/

```

```
CREATE OR REPLACE TRIGGER crea_ID_Venta
  BEFORE INSERT ON Venta
  FOR EACH ROW
  BEGIN
    SELECT S_ID_Venta.NEXTVAL
    INTO :NEW.ID_Venta FROM DUAL;
END crea_ID_Venta;
/
```

sql/tablas.sql

¡

9.2. Funciones y Procedures

```
CREATE OR REPLACE PROCEDURE Socio_Nuevo (
  p_Nombre IN SOCIO.Nombre%TYPE,
  p_Apellidos IN SOCIO.Apellidos%TYPE,
  p_Direccion IN SOCIO.Direccion%TYPE,
  p_FechaDeNacimiento IN SOCIO.FechaDeNacimiento%TYPE,
  p_Email IN SOCIO.Email%TYPE,
  p_DNI IN SOCIO.DNI%TYPE)
 IS BEGIN
  INSERT INTO SOCIO
  VALUES (p_Nombre, p_Apellidos, p_Direccion, p_FechaDeNacimiento,
          p_Email,p_DNI);
END Socio_Nuevo;

/

CREATE OR REPLACE PROCEDURE Producto_Nuevo(
  P_ID_Producto IN PRODUCTO.ID_Producto%TYPE,
  P_Nombre IN PRODUCTO.Nombre%TYPE,
  P_Descripcion IN PRODUCTO.Descripcion%TYPE,
  P_Categoria IN PRODUCTO.Categoria%TYPE,
  P_PrecioProducto IN PRODUCTO.PrecioProducto%TYPE,
  P_IVA IN PRODUCTO.IVA%TYPE)
  IS BEGIN
  INSERT INTO PRODUCTO
    VALUES(P_ID_Producto,P_Nombre,P_Descripcion,P_Categoria,
           P_PrecioProducto, P_IVA);
END Producto_Nuevo;

/

CREATE OR REPLACE PROCEDURE Stock_Nuevo (
    P_ID_Emplazamiento IN STOCK.ID_Emplazamiento%TYPE,
    P_ID_Producto IN STOCK.ID_Producto%TYPE,
    P_Cantidad IN STOCK.Cantidad%TYPE)
    IS BEGIN
    INSERT INTO STOCK
    VALUES (P_ID_Emplazamiento, P_ID_Producto, P_Cantidad);
END Stock_Nuevo;

/

CREATE OR REPLACE PROCEDURE PRODUCTO_TRASPASO_Nuevo (
    p_ID_Traspaso IN ASOCIACION_PRODUCTO_TRASPASO.ID_Traspaso%TYPE,
    p_ID_Producto IN ASOCIACION_PRODUCTO_TRASPASO.ID_Producto%TYPE,
    p_Cantidad IN ASOCIACION_PRODUCTO_TRASPASO.Cantidad%TYPE)
    IS BEGIN
    INSERT INTO ASOCIACION_PRODUCTO_TRASPASO
    VALUES(p_ID_Traspaso,p_ID_Producto,p_Cantidad);
END PRODUCTO_TRASPASO_Nuevo;

/

CREATE OR REPLACE PROCEDURE PRODUCTO_SOLICITUD_Nuevo (
    p_ID_Solicitud IN ASOCIACION_PRODUCTO_SOLICITUD.ID_Solicitud%TYPE,
    p_ID_Producto IN ASOCIACION_PRODUCTO_SOLICITUD.ID_Producto%TYPE,
    p_Cantidad IN ASOCIACION_PRODUCTO_SOLICITUD.Cantidad%TYPE)
```

```
    IS BEGIN
    INSERT INTO ASOCIACION_PRODUCTO_SOLICITUD
    VALUES(p_ID_Solicitud,p_ID_Producto,p_Cantidad);
END PRODUCTO_SOLICITUD_Nuevo;

/

CREATE OR REPLACE PROCEDURE TRASPASO_Nuevo (
    p_ID_Traspaso IN TRASPASO.ID_Traspaso%TYPE,
    p_FechaTraspaso IN TRASPASO.FechaTraspaso%TYPE,
    p_ID_EmplazamientoSalida IN TRASPASO.ID_EmplazamientoSalida%TYPE,
    p_ID_EmplazamientoEntrada IN TRASPASO.ID_EmplazamientoEntrada%TYPE)
    IS BEGIN
    INSERT INTO TRASPASO
    VALUES(p_ID_Traspaso, p_FechaTraspaso, p_ID_EmplazamientoSalida,
           p_ID_EmplazamientoEntrada);
END TRASPASO_Nuevo;

/

CREATE OR REPLACE PROCEDURE SOLICITUD_Nuevo (
    p_ID_Solicitud IN SOLICITUD_TRASPASO.ID_Solicitud%TYPE,
    p_FechaSolicitud IN SOLICITUD_TRASPASO.FechaSolicitud%TYPE,
    p_ID_EmplazamientoSalida IN SOLICITUD_TRASPASO.ID_EmplazamientoSalida%TYPE,
    p_ID_EmplazamientoEntrada IN SOLICITUD_TRASPASO.ID_EmplazamientoEntrada%TYPE)
    IS BEGIN
    INSERT INTO SOLICITUD_TRASPASO
    VALUES(p_ID_Solicitud, p_FechaSolicitud, p_ID_EmplazamientoSalida,
           p_ID_EmplazamientoEntrada);
END SOLICITUD_Nuevo;

/



CREATE OR REPLACE PROCEDURE Emplazamiento_Nuevo(
  P_ID_Emplazamiento IN EMPLAZAMIENTO.ID_Emplazamiento%TYPE,
  P_Direccion IN EMPLAZAMIENTO.Direccion%TYPE,
  P_Telefono IN EMPLAZAMIENTO.Telefono%TYPE,
  P_Tipo IN EMPLAZAMIENTO.Tipo%TYPE)
  IS BEGIN
  INSERT INTO EMPLAZAMIENTO
    VALUES(P_ID_Emplazamiento,P_Direccion,P_Telefono,P_Tipo);
END Emplazamiento_Nuevo;

/

CREATE OR REPLACE PROCEDURE Factura_Nueva(
    p_ID IN FACTURA.ID_FACTURA%TYPE,
    p_FechaDeExpedicion IN FACTURA.FechaDeExpedicion%TYPE,
    p_Devuelto IN FACTURA.Devuelto%TYPE,
    p_ID_Venta IN FACTURA.ID_Venta%TYPE,
    p_ID_Emplazamiento IN FACTURA.ID_Emplazamiento%TYPE
)
IS BEGIN
    INSERT INTO FACTURA VALUES(
        p_ID, p_FechaDeExpedicion, p_Devuelto, p_ID_Venta,
        p_ID_Emplazamiento);
END Factura_Nueva;

/

CREATE OR REPLACE PROCEDURE PROVEEDOR_Nuevo (
  p_CIF IN PROVEEDOR.CIF%TYPE,
  p_Nombre IN PROVEEDOR.Nombre%TYPE,
  p_Telefono IN PROVEEDOR.Telefono%TYPE,
  p_Email IN PROVEEDOR.Email%TYPE
  )IS BEGIN
  INSERT INTO PROVEEDOR
  VALUES (p_CIF, p_Nombre, p_Telefono, p_Email);
END PROVEEDOR_Nuevo;

/

CREATE OR REPLACE PROCEDURE ALBARAN_Nuevo (
  p_ID_Albaran IN ALBARAN.ID_Albaran%TYPE,
  p_FechaFirma IN ALBARAN.FechaFirma%TYPE,
```

```
  p_ID_Pedido IN ALBARAN.ID_PEDIDO%TYPE
  )IS BEGIN
  INSERT INTO ALBARAN
  VALUES ( p_ID_Albaran, p_FechaFirma, p_ID_Pedido);
END ALBARAN_Nuevo;

/

CREATE OR REPLACE PROCEDURE PEDIDO_Nuevo (
  p_ID_Pedido IN PEDIDO.ID_Pedido%TYPE,
  p_FechaPedido IN PEDIDO.FechaPedido%TYPE,
  p_ID_Emplazamiento IN PEDIDO.ID_Emplazamiento%TYPE,
  p_CIF IN PEDIDO.CIF%TYPE
  )IS BEGIN
  INSERT INTO PEDIDO
  VALUES ( p_ID_Pedido ,p_FechaPedido ,
           p_ID_Emplazamiento, p_CIF);
END PEDIDO_Nuevo;

/

CREATE OR REPLACE PROCEDURE PEDIDO_PRODUCTO_Nuevo (
    p_ID_Producto IN ASOCIACION_PEDIDO_PRODUCTO.ID_Producto%TYPE,
    p_ID_Pedido IN ASOCIACION_PEDIDO_PRODUCTO.ID_Pedido%TYPE,
    p_Cantidad IN ASOCIACION_PEDIDO_PRODUCTO.Cantidad%TYPE,
    p_PrecioCompra IN ASOCIACION_PEDIDO_PRODUCTO.PrecioCompra%TYPE,
    p_IVA IN ASOCIACION_PEDIDO_PRODUCTO.IVA%TYPE)IS
    BEGIN

    INSERT INTO ASOCIACION_PEDIDO_PRODUCTO
    VALUES (p_ID_Producto, p_ID_Pedido , p_Cantidad , p_PrecioCompra,
            p_IVA);
END PEDIDO_PRODUCTO_Nuevo;

/


CREATE OR REPLACE PROCEDURE Venta_Nueva(
  P_ID_Venta IN VENTA.ID_Venta%TYPE,
  P_FechaVenta IN VENTA.FechaVenta%TYPE,
  P_DNI IN VENTA.DNI%TYPE)
  IS BEGIN
  INSERT INTO VENTA
    VALUES(P_ID_Venta, P_FechaVenta, P_DNI);
END Venta_Nueva;

/

CREATE OR REPLACE PROCEDURE VENTA_PRODUCTO_Nueva(
    P_ID_Venta IN ASOCIACION_VENTA_PRODUCTO.ID_Venta%TYPE,
    P_ID_Producto IN ASOCIACION_VENTA_PRODUCTO.ID_Producto%TYPE,
    P_Cantidad IN ASOCIACION_VENTA_PRODUCTO.Cantidad%TYPE,
    P_PrecioVenta IN ASOCIACION_VENTA_PRODUCTO.PrecioVenta%TYPE,
    P_IvaVenta IN ASOCIACION_VENTA_PRODUCTO.IvaVenta%TYPE
    )
    IS
    BEGIN
    INSERT INTO ASOCIACION_VENTA_PRODUCTO
        VALUES(P_ID_Venta, P_ID_Producto,P_Cantidad,P_PrecioVenta,P_IvaVenta);
END VENTA_PRODUCTO_Nueva;

/

CREATE OR REPLACE PROCEDURE MODIFICA_PROVEEDOR_NOMBRE
(p_CIF IN PROVEEDOR.CIF%TYPE,
p_Nombre IN PROVEEDOR.Nombre%TYPE) IS
BEGIN
UPDATE PROVEEDOR SET Nombre = p_Nombre WHERE p_CIF = CIF;
END MODIFICA_PROVEEDOR_NOMBRE;

/

CREATE OR REPLACE PROCEDURE MODIFICA_PROVEEDOR_Telefono
(p_CIF IN PROVEEDOR.CIF%TYPE,
p_Telefono IN PROVEEDOR.Telefono%TYPE) IS
BEGIN
UPDATE PROVEEDOR SET Telefono = p_Telefono WHERE p_CIF = CIF;
```

```
END MODIFICA_PROVEEDOR_Telefono;

/

CREATE OR REPLACE PROCEDURE MODIFICA_PROVEEDOR_EMAIL
(p_CIF IN PROVEEDOR.CIF%TYPE,
p_Email IN PROVEEDOR.Email%TYPE) IS
BEGIN
UPDATE PROVEEDOR SET Email = p_Email WHERE p_CIF = CIF;
END MODIFICA_PROVEEDOR_EMAIL;

/

CREATE OR REPLACE PROCEDURE MODIFICA_PRODUCTO_DESCRIPCION (
    p_ID_Producto IN PRODUCTO.ID_Producto%TYPE,
    p_Nombre IN PRODUCTO.Nombre%TYPE,
    p_Descripcion IN PRODUCTO.Descripcion%TYPE,
    p_Categoria IN PRODUCTO.Categoria%TYPE,
    p_PrecioProducto IN PRODUCTO.PrecioProducto%TYPE,
    p_IVA IN PRODUCTO.IVA%TYPE)
    IS BEGIN
    UPDATE PRODUCTO SET Descripcion = p_Descripcion WHERE p_ID_Producto = ID_Producto;
END MODIFICA_PRODUCTO_DESCRIPCION;

/

CREATE OR REPLACE PROCEDURE MODIFICA_PRODUCTO_PRECIO (
    p_ID_Producto IN PRODUCTO.ID_Producto%TYPE,
    p_Nombre IN PRODUCTO.Nombre%TYPE,
    p_Descripcion IN PRODUCTO.Descripcion%TYPE,
    p_Categoria IN PRODUCTO.Categoria%TYPE,
    p_PrecioProducto IN PRODUCTO.PrecioProducto%TYPE,
    p_IVA IN PRODUCTO.IVA%TYPE)
    IS BEGIN
    UPDATE PRODUCTO SET PrecioProducto = p_PrecioProducto WHERE p_ID_Producto = ID_Producto;
END MODIFICA_PRODUCTO_PRECIO;

/

CREATE OR REPLACE PROCEDURE MODIFICA_PRODUCTO_IVA (
    p_ID_Producto IN PRODUCTO.ID_Producto%TYPE,
    p_Nombre IN PRODUCTO.Nombre%TYPE,
    p_Descripcion IN PRODUCTO.Descripcion%TYPE,
    p_Categoria IN PRODUCTO.Categoria%TYPE,
    p_PrecioProducto IN PRODUCTO.PrecioProducto%TYPE,
    p_IVA IN PRODUCTO.IVA%TYPE)
    IS BEGIN
    UPDATE PRODUCTO SET IVA = p_IVA
    WHERE p_ID_Producto = ID_Producto;
END MODIFICA_PRODUCTO_IVA;

/

CREATE OR REPLACE PROCEDURE MODIFICA_STOCK_CANTIDAD
  (p_ID_Emplazamiento IN STOCK.ID_Emplazamiento%TYPE,
  p_ID_Producto IN STOCK.ID_Producto%TYPE,
  p_Cantidad IN STOCK.Cantidad%TYPE)
  IS BEGIN
  UPDATE STOCK SET Cantidad = p_Cantidad
  WHERE p_ID_Emplazamiento = ID_Emplazamiento AND p_ID_PRODUCTO = ID_PRODUCTO;
END MODIFICA_STOCK_CANTIDAD;

/

create or replace PROCEDURE MODIFICA_SOCIO_DIRECCION(
    m_DNI IN SOCIO.DNI%TYPE,
    m_DIRECCION IN SOCIO.DIRECCION%TYPE)
    IS BEGIN
    UPDATE SOCIO SET DIRECCION = m_DIRECCION where m_DNI = DNI;
    COMMIT WORK;
END MODIFICA_SOCIO_DIRECCION;

/

create or replace PROCEDURE MODIFICA_SOCIO_EMAIL(m_DNI IN SOCIO.DNI%TYPE,m_email IN SOCIO.EMAIL%TYPE)
  IS BEGIN
  UPDATE SOCIO SET EMAIL = m_email where m_DNI = DNI;
```

```
  COMMIT WORK;
END MODIFICA_SOCIO_EMAIL;

/

CREATE OR REPLACE PROCEDURE MODIFICA_EMPLAZAMIENTO_DIR(
    m_ID IN EMPLAZAMIENTO.ID_Emplazamiento%TYPE, m_Direccion IN EMPLAZAMIENTO.Direccion%TYPE)
IS BEGIN
    UPDATE EMPLAZAMIENTO SET Direccion = m_Direccion where m_ID = ID_Emplazamiento;
END MODIFICA_EMPLAZAMIENTO_DIR;

/

CREATE OR REPLACE PROCEDURE MODIFICA_EMPLAZAMIENTO_TEL(
    m_ID IN EMPLAZAMIENTO.ID_Emplazamiento%TYPE, m_Telefono IN EMPLAZAMIENTO.Telefono%TYPE)
    IS BEGIN
    UPDATE EMPLAZAMIENTO SET Telefono = m_Telefono where m_ID = ID_Emplazamiento;

END MODIFICA_EMPLAZAMIENTO_TEL;

/

CREATE OR REPLACE PROCEDURE MODIFICA_FACTURA_DEVUELTO
(m_ID_FACTURA IN FACTURA.ID_FACTURA%TYPE,m_Devuelto IN FACTURA.Devuelto%TYPE)
  IS BEGIN
  UPDATE FACTURA SET Devuelto = m_Devuelto where m_ID_FACTURA = ID_FACTURA;
END MODIFICA_FACTURA_DEVUELTO;

/

CREATE OR REPLACE PROCEDURE ELIMINA_PROVEEDOR(p_CIF IN PROVEEDOR.CIF%TYPE)
  IS BEGIN
  DELETE FROM PROVEEDOR WHERE p_CIF = CIF;
END ELIMINA_PROVEEDOR;

/

CREATE OR REPLACE PROCEDURE ELIMINA_PRODUCTO(p_ID_Producto IN PRODUCTO.ID_Producto%TYPE)
  IS BEGIN
  DELETE FROM PRODUCTO WHERE ID_Producto =  p_ID_Producto;
END ELIMINA_PRODUCTO;

/

CREATE OR REPLACE PROCEDURE ELIMINA_EMPLAZAMIENTO(p_ID_Emplazamiento IN EMPLAZAMIENTO.
     ID_Emplazamiento%TYPE)
  IS BEGIN
  DELETE FROM EMPLAZAMIENTO WHERE ID_Emplazamiento =  p_ID_Emplazamiento;
END ELIMINA_EMPLAZAMIENTO;

/

CREATE OR REPLACE PROCEDURE ELIMINA_SOCIO(p_DNI IN SOCIO.DNI%TYPE)
  IS BEGIN
  DELETE FROM SOCIO WHERE DNI =  p_DNI;
END ELIMINA_SOCIO;

/

CREATE OR REPLACE PROCEDURE ELIMINA_A_VENTA(e_ID_Producto IN Producto.ID_PRODUCTO%TYPE,e_ID_VENTA IN
     VENTA.ID_VENTA%TYPE)
  IS BEGIN
  DELETE FROM ASOCIACION_VENTA_PRODUCTO WHERE ID_PRODUCTO = e_ID_Producto AND ID_VENTA = e_ID_VENTA;
END ELIMINA_A_VENTA;

/


/*FUNCIONES*/
/* Esta abajo con menos parametros
CREATE OR REPLACE FUNCTION precio_A_Venta_producto
(f_ID_Venta IN ASOCIACION_VENTA_PRODUCTO.ID_Venta%TYPE,
f_Cantidad IN ASOCIACION_VENTA_PRODUCTO.Cantidad%TYPE,
f_PrecioVenta IN ASOCIACION_VENTA_PRODUCTO.PrecioVenta%TYPE)
RETURN NUMBER is f_PrecioLinea ASOCIACION_VENTA_PRODUCTO.PRECIOLINEA%TYPE;
BEGIN
f_PrecioLinea := f_PrecioVenta * f_Cantidad;
```

```
RETURN(f_PrecioLinea);
END precio_A_Venta_producto;

/
*/
/*
CREATE OR REPLACE FUNCTION precio_Venta
(f_ID_Venta IN VENTA.ID_Venta%TYPE)
RETURN NUMBER is f_PrecioTotal VENTA.PRECIOTOTAL%TYPE;
BEGIN
select SUM(precioLinea) into f_PrecioTotal from ASOCIACION_VENTA_PRODUCTO
where ID_Venta = f_ID_Venta;
RETURN(f_PrecioTotal);
END precio_Venta;

/
*/
/* Esta abajo con menos parametros
CREATE OR REPLACE FUNCTION precio_A_Pedido_producto
(f_ID_Pedido IN ASOCIACION_PEDIDO_PRODUCTO.ID_Pedido%TYPE,
f_Cantidad IN ASOCIACION_PEDIDO_PRODUCTO.Cantidad%TYPE,
f_PrecioCompra IN ASOCIACION_PEDIDO_PRODUCTO.PrecioCompra%TYPE)
RETURN NUMBER is f_PrecioLinea ASOCIACION_PEDIDO_PRODUCTO.PrecioLinea%TYPE;
BEGIN
f_PrecioLinea := f_PrecioCompra * f_Cantidad;
RETURN(f_PrecioLinea);
END precio_A_Pedido_producto;

/
*/
/*
CREATE OR REPLACE FUNCTION precio_Pedido
(f_ID_Pedido IN PEDIDO.ID_Pedido%TYPE)
RETURN NUMBER is f_PrecioTotal PEDIDO.PRECIOTOTAL%TYPE;
BEGIN
select SUM(PrecioLinea) into f_PrecioTotal from ASOCIACION_PEDIDO_PRODUCTO
where ID_Pedido = f_ID_Pedido;
RETURN(f_PrecioTotal);
END precio_Pedido;

/
*/

CREATE OR REPLACE FUNCTION precioLinea_Aso_Pedido
(f_ID_PRODUCTO IN ASOCIACION_PEDIDO_PRODUCTO.ID_PRODUCTO%TYPE,f_ID_PEDIDO IN
     ASOCIACION_PEDIDO_PRODUCTO.ID_PEDIDO%TYPE)
RETURN NUMBER is f_PrecioLinea ASOCIACION_PEDIDO_PRODUCTO.PRECIOCOMPRA%TYPE;
f_Cantidad ASOCIACION_PEDIDO_PRODUCTO.CANTIDAD%TYPE;
f_PrecioCompra ASOCIACION_PEDIDO_PRODUCTO.PRECIOCOMPRA%TYPE;
BEGIN
select Cantidad into f_Cantidad from ASOCIACION_PEDIDO_PRODUCTO where ID_Pedido = f_ID_PEDIDO AND
     ID_PRODUCTO = f_ID_PRODUCTO;
select PrecioCompra into f_PrecioCompra from ASOCIACION_PEDIDO_PRODUCTO where ID_Pedido = f_ID_PEDIDO
      AND ID_PRODUCTO = f_ID_PRODUCTO;
f_PrecioLinea := f_Cantidad * f_PrecioCompra;
RETURN(f_PrecioLinea);
END precioLinea_Aso_Pedido;

/

CREATE OR REPLACE FUNCTION precioLinea_Aso_Venta
(f_ID_PRODUCTO IN ASOCIACION_VENTA_PRODUCTO.ID_PRODUCTO%TYPE,f_ID_VENTA IN ASOCIACION_VENTA_PRODUCTO.
     ID_VENTA%TYPE)
RETURN NUMBER is f_PrecioLinea ASOCIACION_VENTA_PRODUCTO.PRECIOVENTA%TYPE;
f_Cantidad ASOCIACION_VENTA_PRODUCTO.CANTIDAD%TYPE;
f_PrecioVenta ASOCIACION_VENTA_PRODUCTO.PRECIOVENTA%TYPE;
BEGIN
select Cantidad into f_Cantidad from ASOCIACION_VENTA_PRODUCTO where ID_VENTA = f_ID_VENTA AND
     ID_PRODUCTO = f_ID_PRODUCTO;
select PrecioVenta into f_PrecioVenta from ASOCIACION_VENTA_PRODUCTO where ID_VENTA = f_ID_VENTA AND
     ID_PRODUCTO = f_ID_PRODUCTO;
f_PrecioLinea := f_Cantidad * f_PrecioVenta;
RETURN(f_PrecioLinea);
END precioLinea_Aso_Venta;

/

```

```
CREATE OR REPLACE FUNCTION precioTotal_Venta
(f_ID_VENTA IN ASOCIACION_VENTA_PRODUCTO.ID_VENTA%TYPE)
RETURN NUMBER is f_precioTotal ASOCIACION_VENTA_PRODUCTO.PRECIOVENTA%TYPE;
precio_AUX ASOCIACION_VENTA_PRODUCTO.PRECIOVENTA%TYPE;

CURSOR all_prods
 IS
 SELECT id_venta,id_producto,cantidad
 FROM ASOCIACION_VENTA_PRODUCTO
 ORDER BY ID_producto;

 m_id_venta ASOCIACION_VENTA_PRODUCTO.id_venta%TYPE;
 m_id_producto Producto.id_producto%type;
 m_cantidad ASOCIACION_VENTA_PRODUCTO.cantidad%TYPE;
BEGIN
  f_precioTotal := 0;
  OPEN all_prods;
  LOOP
    Fetch all_prods INTO m_id_venta,m_id_producto,m_cantidad;
    EXIT WHEN all_prods%NOTFOUND;
    if m_id_venta = f_id_venta
    THEN
    select precioVenta into precio_AUX from asociacion_venta_producto where id_venta = m_id_venta and
      id_producto=m_id_producto;

    f_precioTotal := (precio_AUX*m_cantidad) + f_precioTotal;
    END IF;
  END LOOP;
  CLOSE all_prods;
RETURN(f_PrecioTotal);
END precioTotal_Venta;

/

CREATE OR REPLACE FUNCTION precioTotal_Factura
(f_ID_VENTA IN ASOCIACION_VENTA_PRODUCTO.ID_VENTA%TYPE)
RETURN NUMBER is f_precioTotal number(8,2);
precio_AUX ASOCIACION_VENTA_PRODUCTO.PRECIOVENTA%TYPE;
socio varchar2(9);
BEGIN
 select precioTotal_Venta(f_ID_VENTA) into precio_AUX from dual;
 select dni into socio from venta where id_Venta = f_id_Venta;
 f_precioTotal := precio_AUX;
 if socio is not null
 then
 f_precioTotal := f_precioTotal * 0.95;
 END IF;
RETURN(f_PrecioTotal);
END precioTotal_Factura;

/

CREATE OR REPLACE FUNCTION precioTotal_Albaran
(f_ID_PEDIDO IN ASOCIACION_PEDIDO_PRODUCTO.ID_PEDIDO%TYPE)
RETURN NUMBER is f_precioTotal number(8,2);
precio_AUX ASOCIACION_PEDIDO_PRODUCTO.PRECIOCOMPRA%TYPE;
BEGIN
 select precioTotal_Pedido(f_ID_PEDIDO) into precio_AUX from dual;
 f_precioTotal := precio_AUX;
RETURN(f_PrecioTotal);
END precioTotal_Albaran;

/

CREATE OR REPLACE FUNCTION precioTotal_PEDIDO
(f_ID_PEDIDO IN ASOCIACION_PEDIDO_PRODUCTO.ID_PEDIDO%TYPE)
RETURN NUMBER is f_precioTotal ASOCIACION_PEDIDO_PRODUCTO.PRECIOCOMPRA%TYPE;
precio_AUX ASOCIACION_PEDIDO_PRODUCTO.PRECIOCOMPRA%TYPE;

CURSOR all_prods
 IS
 SELECT id_pedido,id_producto,cantidad
 FROM ASOCIACION_PEDIDO_PRODUCTO
 ORDER BY ID_producto;

 m_id_pedido ASOCIACION_PEDIDO_PRODUCTO.id_pedido%TYPE;
 m_id_producto Producto.id_producto%type;
```

```
 m_cantidad ASOCIACION_PEDIDO_PRODUCTO.cantidad%TYPE;
BEGIN
  f_precioTotal := 0;
  OPEN all_prods;
  LOOP
    Fetch all_prods INTO m_id_pedido,m_id_producto,m_cantidad;
    EXIT WHEN all_prods%NOTFOUND;
    if m_id_pedido = f_id_pedido
    THEN
    select precioCompra into precio_AUX from ASOCIACION_PEDIDO_PRODUCTO where id_pedido = m_id_pedido
      and id_producto=m_id_producto;

    f_precioTotal := (precio_AUX*m_cantidad) + f_precioTotal;
    END IF;
  END LOOP;
  CLOSE all_prods;
RETURN(f_PrecioTotal);
END precioTotal_Pedido;

/
```

sql/funciones_y_procedures.sql

9.3. Triggers

```
/* TRIGGERS */
/*
CREATE OR REPLACE TRIGGER descuento_socio
  BEFORE INSERT ON factura
  FOR EACH ROW
  declare v_preciototal VENTA.PRECIOTOTAL%TYPE;
  v_DNI VENTA.dni%TYPE;
BEGIN
  select preciototal into v_preciototal from venta where id_Venta = :NEW.id_venta;
  select dni into v_dni from venta where id_venta = :NEW.id_Venta;
  IF v_DNI IS NOT NULL then
  UPDATE venta set preciototal = v_preciototal * 0.95 where id_Venta = :NEW.id_venta;
  :New.preciototal := v_preciototal * 0.95;
  else
  update venta set preciototal = v_preciototal where id_venta = :NEW.id_venta;
  END IF;
END;

/
*/
/*
CREATE OR REPLACE TRIGGER stock_minimo
  AFTER INSERT OR UPDATE ON stock
  FOR EACH ROW
BEGIN
  IF :NEW.cantidad <0
  THEN
  raise_application_error(-20601, :NEW.cantidad || 'No se puede realizar esta operacion');
  END IF;
END;

/
*/
CREATE OR REPLACE TRIGGER Inicializa_nueva_Venta
BEFORE INSERT ON VENTA
FOR EACH ROW
BEGIN
:NEW.FechaVenta := SYSDATE;
END Comprueba_Venta;

/

CREATE OR REPLACE TRIGGER Inicializa_nuevo_Pedido
BEFORE INSERT ON PEDIDO
FOR EACH ROW
BEGIN
```

```
:NEW.FechaPedido := SYSDATE;
END Comprueba_Pedido;

/

CREATE OR REPLACE TRIGGER Inicializa_nueva_Factura
BEFORE INSERT ON Factura
FOR EACH ROW
BEGIN
:NEW.FechaDeExpedicion := SYSDATE;
END Inicializa_nueva_Factura;


/

CREATE OR REPLACE TRIGGER modifica_stock_venta
BEFORE INSERT ON FACTURA
FOR EACH ROW
DECLARE
e_ID_Emplazamiento Emplazamiento.ID_Emplazamiento%TYPE;
v_ID_Venta Venta.ID_VENTA%TYPE;

 CURSOR all_prods
 IS
 SELECT id_venta,id_producto,cantidad
 FROM ASOCIACION_VENTA_PRODUCTO
 ORDER BY ID_producto;

 m_id_venta ASOCIACION_VENTA_PRODUCTO.id_venta%TYPE;
 m_id_producto Producto.id_producto%type;
 m_cantidad ASOCIACION_VENTA_PRODUCTO.cantidad%TYPE;

BEGIN
select ID_Emplazamiento into e_ID_Emplazamiento from Emplazamiento where ID_Emplazamiento = :NEW.
     ID_Emplazamiento;
select ID_Venta into v_ID_Venta from Venta where ID_Venta = :New.ID_Venta;

OPEN all_prods;
LOOP
    Fetch all_prods INTO m_id_venta,m_id_producto,m_cantidad;
    EXIT WHEN all_prods%NOTFOUND;
    if m_id_venta = v_id_Venta
    THEN
    update stock set cantidad = cantidad-m_cantidad where id_emplazamiento = e_ID_Emplazamiento AND
     id_producto =m_ID_Producto;
    END IF;
  END LOOP;
  CLOSE all_prods;
END modifica_stock_venta;

/
/*
CREATE OR REPLACE TRIGGER inicializa_preciolinea_alv
  BEFORE INSERT OR UPDATE ON ASOCIACION_VENTA_PRODUCTO
  FOR EACH ROW
  BEGIN
  :NEW.preciolinea := :New.cantidad * :New.precioVenta;
END inicializa_preciolinea_alv;

/
*/
/*
CREATE OR REPLACE TRIGGER inicializa_preciototal_venta
  BEFORE INSERT OR UPDATE ON ASOCIACION_VENTA_PRODUCTO
  for each row
  begin
  UPDATE venta set preciototal = PRECIOTOTAL + (:New.cantidad * :New.precioVenta) where id_venta = :
     New.id_venta;
END inicializa_preciototal_venta;

/

*/
CREATE OR REPLACE TRIGGER modifica_stock_pedido
BEFORE INSERT OR UPDATE ON ALBARAN
FOR EACH ROW
DECLARE
```

```
e_ID_Emplazamiento Emplazamiento.ID_Emplazamiento%TYPE;
p_ID_PEDIDO PEDIDO.ID_PEDIDO%TYPE;

 CURSOR all_prods
 IS
 SELECT id_pedido,id_producto,cantidad
 FROM ASOCIACION_PEDIDO_PRODUCTO
 ORDER BY ID_producto;

 m_id_pedido ASOCIACION_PEDIDO_PRODUCTO.id_pedido%TYPE;
 m_id_producto Producto.id_producto%type;
 m_cantidad ASOCIACION_PEDIDO_PRODUCTO.cantidad%TYPE;

BEGIN

select ID_Emplazamiento into e_ID_Emplazamiento from pedido where ID_pedido = :NEW.ID_Pedido;
select ID_PEDIDO into p_id_pedido from pedido where ID_pedido = :NEW.ID_Pedido;
OPEN all_prods;
LOOP
    Fetch all_prods INTO m_id_pedido,m_id_producto,m_cantidad;
    EXIT WHEN all_prods%NOTFOUND;
    if m_id_pedido = p_id_pedido
    THEN
    update stock set cantidad = cantidad+m_cantidad where id_emplazamiento = e_ID_Emplazamiento AND
     id_producto =m_ID_Producto;
    END IF;
  END LOOP;
  CLOSE all_prods;
END modifica_stock_pedido;

/

CREATE OR REPLACE TRIGGER modifica_stock_traspaso
BEFORE INSERT OR UPDATE ON ASOCIACION_PRODUCTO_TRASPASO
FOR EACH ROW
DECLARE
e_ID_Emplazamiento_salida Emplazamiento.ID_Emplazamiento%TYPE;
e_ID_Emplazamiento_entrada Emplazamiento.ID_Emplazamiento%TYPE;
t_ID_traspaso traspaso.ID_traspaso%TYPE;

BEGIN
select ID_Emplazamientosalida into e_ID_Emplazamiento_salida from traspaso where ID_traspaso = :NEW.
     ID_traspaso;
select ID_Emplazamientoentrada into e_ID_Emplazamiento_entrada from traspaso where ID_traspaso = :NEW
     .ID_traspaso;
select ID_traspaso into t_id_traspaso from traspaso where ID_traspaso = :NEW.ID_traspaso;
update stock set cantidad = (cantidad- :NEW.cantidad) where id_emplazamiento =
     e_ID_Emplazamiento_salida AND id_producto = :NEW.id_producto;
update stock set cantidad = (cantidad+ :NEW.cantidad) where id_emplazamiento =
     e_ID_Emplazamiento_entrada AND id_producto = :NEW.id_producto;

END modifica_stock_traspaso;
/
CREATE OR REPLACE TRIGGER Inicializa_Nuevo_Pedido
BEFORE INSERT ON Pedido
FOR EACH ROW
BEGIN
:NEW.FechaPedido := SYSDATE;
END Inicializa_Nuevo_Pedido;

/


CREATE OR REPLACE TRIGGER Inicializa_Nueva_ST
BEFORE INSERT ON Solicitud_Traspaso
FOR EACH ROW
BEGIN
:NEW.FechaSolicitud := SYSDATE;
END Inicializa_Nueva_ST;

/

CREATE OR REPLACE TRIGGER Inicializa_Nuevo_Traspaso
BEFORE INSERT ON Traspaso
FOR EACH ROW
BEGIN
:NEW.FechaTraspaso := SYSDATE;
```

```
END Inicializa_Nuevo_Traspaso;

/

create or replace TRIGGER solicitud_stock_minimo
BEFORE INSERT ON asociacion_producto_solicitud
FOR EACH ROW
DECLARE
e_id_emplazamientoentrada SOLICITUD_TRASPASO.ID_EMPLAZAMIENTOENTRADA%TYPE;
e_cantidad Stock.cantidad%TYPE;

BEGIN
  SELECT id_emplazamientoentrada into e_id_emplazamientoentrada from solicitud_traspaso where
     id_solicitud = :NEW.id_solicitud;
  select cantidad into e_cantidad from stock where id_emplazamiento = e_id_emplazamientoentrada and
     id_producto = :NEW.id_producto;
  if (e_cantidad - :NEW.cantidad) <=5
  THEN raise_application_error(-20601, :NEW.cantidad || 'No se permite la solicitud, el otro
     emplazamiento alcanzara su stock minimo');
  END IF;
END;

/
/*
create or replace TRIGGER inicializa_preciolinea_alp
  BEFORE INSERT OR UPDATE ON ASOCIACION_PEDIDO_PRODUCTO
  FOR EACH ROW
  BEGIN
  :NEW.preciolinea := :New.cantidad * :New.preciocompra;
END inicializa_preciolinea_alp;
/
*/
/*
create or replace TRIGGER inicializa_preciototal_albaran
  BEFORE INSERT OR UPDATE ON Albaran
  for each row
  DECLARE
  p_preciototal pedido.preciototal%TYPE;
  begin
  select preciototal into p_preciototal from pedido where id_pedido = :New.id_pedido;
  :New.preciototal := p_preciototal;
END inicializa_preciototal_albaran;

/
*/
/*
create or replace TRIGGER inicializa_preciototal_pedido
  BEFORE INSERT OR UPDATE ON ASOCIACION_pedido_producto
  for each row
  begin
  UPDATE pedido set preciototal = PRECIOTOTAL + (:New.cantidad * :New.preciocompra) where id_pedido =
      :New.id_pedido;
END inicializa_preciototal_pedido;

/
*/

create or replace trigger inicializa_precioventa_apv
  before insert on asociacion_venta_producto
  for each row
  declare
  p_precio producto.precioproducto%TYPE;
  begin
  select precioproducto into p_precio from producto where id_producto = :New.id_producto;
  :NEW.precioventa := p_precio;
END inicializa_precioventa_apv;

/

create or replace trigger inicializa_IVA_apv
  before insert on asociacion_venta_producto
  for each row
  declare
  p_IVA PRODUCTO.IVA%TYPE;
  begin
  select Iva into p_IVA from producto where id_producto = :New.id_producto;
  :NEW.IVAVENTA := p_IVA;
```

```
266 END inicializa_IVA_apv;

268 /

270 create or replace trigger inicializa_IVA_APP
  before insert on asociacion_pedido_producto
272   for each row
  declare
274   p_IVA PRODUCTO.IVA%TYPE;
  begin
276   select IVA into p_IVA from producto where id_producto = :New.id_producto;
  :NEW.IVA := p_IVA;
278 END inicializa_IVA_APP;

280 /

282 create or replace trigger devolucion
before update on factura
284 for each row
begin
286   if(sysdate - :old.fechadeexpedicion)>30 then
  raise_application_error(-20601, :NEW.fechadeexpedicion || 'No se permite la devolucion, han pasado
      mas de 30 dias');
288   END IF;
END devolucion;
290 /
/*
292 create or replace trigger mensaje
before update of cantidad on stock
294 for each row
DECLARE
296    lines dbms_output.chararr;
   num_lines number;
298 BEGIN

300     if (:old.cantidad - :new.cantidad)<5 then
   -- enable the buffer with default size 20000
302    dbms_output.enable;

304    dbms_output.put_line('Hello Reader!');
   dbms_output.put_line('Hope you have enjoyed the tutorials!');
306    dbms_output.put_line('Have a great time exploring pl/sql!');

308    num_lines := 3;

310    dbms_output.get_lines(lines, num_lines);

312    FOR i IN 1..num_lines LOOP
      dbms_output.put_line(lines(i));
314    END LOOP;
   end IF;
316 END mensaje;
/
318 */
```

sql/triggers.sql

9.4. Pruebas

```
/* FUNCION AUXILIAR */
2 CREATE OR REPLACE FUNCTION EQUALS(salida BOOLEAN, salidaEsperada BOOLEAN)
RETURN VARCHAR2 AS
4 BEGIN
    IF (salida = salidaEsperada) THEN
6       RETURN 'EXITO';
    ELSE
8       RETURN 'FALLO';
    END IF;
10 END EQUALS;

12 /
```

```
14 CREATE OR REPLACE PROCEDURE PRINTR(nombre_prueba VARCHAR2, salida BOOLEAN,
                                      salidaEsperada BOOLEAN)
16 IS BEGIN
       DBMS_OUTPUT.put_line(nombre_prueba || ':' || EQUALS(salida, salidaEsperada));
18 END PRINTR;

20 /

22 /* DEFINICION PRUEBAS */
   CREATE OR REPLACE PACKAGE PRUEBAS_SOCIO AS
24     PROCEDURE inicializar;
       PROCEDURE insertar
26         (nombre_prueba VARCHAR2, i_nombre VARCHAR2, i_apellidos VARCHAR2,
           i_DNI VARCHAR2, i_direccion VARCHAR2, i_nacimiento DATE, i_email VARCHAR2
28         ,salidaEsperada BOOLEAN);
       PROCEDURE actualizar
30         (nombre_prueba VARCHAR2, a_DNI VARCHAR2, a_direccion VARCHAR2,
           a_email VARCHAR2, salidaEsperada BOOLEAN);
32     PROCEDURE eliminar
           (nombre_prueba VARCHAR2, e_DNI VARCHAR2, salidaEsperada BOOLEAN);
34 END PRUEBAS_SOCIO;

36 /

38 CREATE OR REPLACE PACKAGE PRUEBAS_PROVEEDOR AS
       PROCEDURE inicializar;
40     PROCEDURE insertar
           (nombre_prueba VARCHAR2, i_CIF VARCHAR2, i_nombre VARCHAR2,
42         i_telefono INT, i_email VARCHAR2, salidaEsperada BOOLEAN);
       PROCEDURE actualizar
44         (nombre_prueba VARCHAR2, a_cif VARCHAR2, a_nombre VARCHAR2,
           a_telefono INT, a_email VARCHAR2, salidaEsperada BOOLEAN);
46     PROCEDURE eliminar
           (nombre_prueba VARCHAR2, e_CIF VARCHAR2, salidaEsperada BOOLEAN);
48 END PRUEBAS_PROVEEDOR;

50 /

52 CREATE OR REPLACE PACKAGE PRUEBAS_EMPLAZAMIENTO AS
       PROCEDURE inicializar;
54     PROCEDURE insertar
           (nombre_prueba VARCHAR2, i_direccion VARCHAR2, i_telefono INT,
56         i_tipo VARCHAR2, salidaEsperada BOOLEAN);
       PROCEDURE actualizar
58         (nombre_prueba VARCHAR2, a_id_emplazamiento INT, a_direccion VARCHAR2,
           a_telefono INT, salidaEsperada BOOLEAN);
60     PROCEDURE eliminar
           (nombre_prueba VARCHAR2, e_id_emplazamiento INT, salidaEsperada BOOLEAN);
62 END PRUEBAS_EMPLAZAMIENTO;

64 /

66 CREATE OR REPLACE PACKAGE PRUEBAS_ALBARAN AS
       PROCEDURE inicializar;
68     PROCEDURE insertar
           (nombre_prueba VARCHAR2, i_fecha DATE, i_id_pedido INT, salidaEsperada BOOLEAN);
70     PROCEDURE eliminar
           (nombre_prueba VARCHAR2, e_id_albaran INT, salidaEsperada BOOLEAN);
72 END PRUEBAS_ALBARAN;

74 /

76 CREATE OR REPLACE PACKAGE PRUEBAS_PEDIDO AS
       PROCEDURE inicializar;
78     PROCEDURE insertar
           (nombre_prueba VARCHAR2, i_fecha DATE, i_id_emplazamiento INT, i_cif VARCHAR2, salidaEsperada
         BOOLEAN);
80     PROCEDURE eliminar
           (nombre_prueba VARCHAR2, e_id_pedido INT, salidaEsperada BOOLEAN);
82 END PRUEBAS_PEDIDO;

84 /

86 CREATE OR REPLACE PACKAGE PRUEBAS_PRODUCTO AS
       PROCEDURE inicializar;
88     PROCEDURE insertar
```

```
        (nombre_prueba VARCHAR2, i_nombre VARCHAR2, i_descripcion VARCHAR2,
        i_categoria VARCHAR2, i_precioProducto INT, i_iva INT, salidaEsperada BOOLEAN);
    PROCEDURE actualizar
        (nombre_prueba VARCHAR2, a_id_producto INT , a_descripcion VARCHAR2,
        a_precioProducto INT, a_iva INT, salidaEsperada BOOLEAN);
    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_producto INT, salidaEsperada BOOLEAN);
END PRUEBAS_PRODUCTO;

/

CREATE OR REPLACE PACKAGE PRUEBAS_VENTA AS
    PROCEDURE inicializar;
    PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_fecha DATE, i_dni VARCHAR2,
        salidaEsperada BOOLEAN);
    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_venta INT, salidaEsperada BOOLEAN);
    /*procedure descuento
        (nombre_prueba VARCHAR2, v_id_venta int,v_preciototal number,v_dni_socio varchar2,
     salidaEsperada BOOLEAN);*/
END PRUEBAS_VENTA;

/

CREATE OR REPLACE PACKAGE PRUEBAS_SOLICITUD_TRASPASO AS
    PROCEDURE inicializar;
    PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_fechaSolicitud DATE, i_id_emplazamientoSalida INT,
        i_id_emplazamientoEntrada INT, salidaEsperada BOOLEAN);
    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_solicitudTraspaso INT, salidaEsperada BOOLEAN);
END PRUEBAS_SOLICITUD_TRASPASO;

/

CREATE OR REPLACE PACKAGE PRUEBAS_FACTURA AS
    PROCEDURE inicializar;
    PROCEDURE insertar
      (nombre_prueba VARCHAR2, i_fechaDeExpedicion DATE, i_devuelto VARCHAR2,
      i_id_venta INT, i_id_emplazamiento INT, salidaEsperada BOOLEAN);
    PROCEDURE actualizar
        (nombre_prueba VARCHAR2, a_id_factura INT,a_devuelto VARCHAR2,
        salidaEsperada BOOLEAN);
    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_factura INT, salidaEsperada BOOLEAN);
END PRUEBAS_FACTURA;

/

CREATE OR REPLACE PACKAGE PRUEBAS_TRASPASO AS
    PROCEDURE inicializar;
    PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_fechaTraspaso DATE, i_id_emplazamientoSalida INT,
        i_id_emplazamientoEntrada INT, salidaEsperada BOOLEAN);
    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_Traspaso INT, salidaEsperada BOOLEAN);
END PRUEBAS_TRASPASO;

/

CREATE OR REPLACE PACKAGE PRUEBAS_A_VENTA_PRODUCTO AS
    PROCEDURE inicializar;
    PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_id_venta INT, i_id_producto INT,
            i_cantidad NUMBER, i_precioVenta NUMBER, i_ivaVenta NUMBER,
             salidaEsperada BOOLEAN);
    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_venta INT, e_id_producto INT, salidaEsperada BOOLEAN);
END PRUEBAS_A_VENTA_PRODUCTO;

/

CREATE OR REPLACE PACKAGE PRUEBA_STOCK AS
    PROCEDURE inicializar;
    PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_id_emplazamiento INT, i_id_producto INT,
```

```
        i_cantidad INT, salidaEsperada BOOLEAN);
    PROCEDURE actualizar
        (nombre_prueba VARCHAR2, a_id_emplazamiento INT, a_id_producto INT,
        a_cantidad INT, salidaEsperada BOOLEAN);
    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,
        salidaEsperada BOOLEAN);
    PROCEDURE stock_correcto
        (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,stock_previo int, cantidad
      INT, salidaEsperada BOOLEAN);
    PROCEDURE stock_correcto_p
        (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,stock_previo int, cantidad
      INT, salidaEsperada BOOLEAN);
    PROCEDURE stock_correcto_t
        (nombre_prueba VARCHAR2, e_id_emplazamiento_envia int, e_id_emplazamiento_recibe int,
     stock_previo_envia int, stock_previo_recibe int,
        cantidad int, e_id_producto int, salidaEsperada BOOLEAN);
END PRUEBA_STOCK;

/

CREATE OR REPLACE PACKAGE PRUEBAS_A_PRODUCTO_SOLICITUD AS
    PROCEDURE inicializar;
    PROCEDURE insertar
        (nombre_prueba VARCHAR2,i_id_producto INT, i_id_Solicitud INT,
            i_cantidad NUMBER, salidaEsperada BOOLEAN);
    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_Solicitud INT, e_id_producto INT, salidaEsperada BOOLEAN);
END PRUEBAS_A_PRODUCTO_SOLICITUD;

/

CREATE OR REPLACE PACKAGE PRUEBAS_A_PRODUCTO_TRASPASO AS
    PROCEDURE inicializar;
    PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_id_producto INT, i_id_Traspaso INT,
            i_cantidad NUMBER, salidaEsperada BOOLEAN);
    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_Traspaso INT, e_id_producto INT, salidaEsperada BOOLEAN);
END PRUEBAS_A_PRODUCTO_TRASPASO;

/

CREATE OR REPLACE PACKAGE PRUEBAS_A_PEDIDO_PRODUCTO AS
    PROCEDURE inicializar;
    PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_id_pedido INT, i_id_producto INT,
        i_cantidad NUMBER, i_precioCompra number, i_iva INT,
        salidaEsperada BOOLEAN);
    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_pedido INT, e_id_producto INT,
        salidaEsperada BOOLEAN);
END PRUEBAS_A_PEDIDO_PRODUCTO;

/

CREATE OR REPLACE PACKAGE PRUEBAS_FUNCIONES AS
    PROCEDURE precioLinea_A_Pedido
        (nombre_prueba VARCHAR2, ID_PRODUCTO NUMBER, ID_PEDIDO NUMBER, esperado number,
     salidaEsperada BOOLEAN);
    PROCEDURE precioLinea_A_Venta
        (nombre_prueba VARCHAR2, ID_PRODUCTO NUMBER, ID_VENTA NUMBER, esperado number, salidaEsperada
      BOOLEAN);
    PROCEDURE precio_Venta
        (nombre_prueba VARCHAR2, ID_Venta NUMBER, esperado number, salidaEsperada BOOLEAN);
    PROCEDURE precio_Factura
        (nombre_prueba VARCHAR2, ID_Venta NUMBER, esperado number, salidaEsperada BOOLEAN);
    PROCEDURE precio_Pedido
        (nombre_prueba VARCHAR2, ID_PEDIDO NUMBER, esperado number, salidaEsperada BOOLEAN);
    PROCEDURE precio_Albaran
        (nombre_prueba VARCHAR2, ID_PEDIDO NUMBER, esperado number, salidaEsperada BOOLEAN);
END PRUEBAS_FUNCIONES;
/

/* CUERPOS DE PRUEBAS */

CREATE OR REPLACE PACKAGE BODY PRUEBAS_SOCIO AS
```

```
238     PROCEDURE inicializar AS
        BEGIN
240         DELETE FROM SOCIO;
        END inicializar;
242
        PROCEDURE insertar
244         (nombre_prueba VARCHAR2, i_nombre VARCHAR2, i_apellidos VARCHAR2,
            i_DNI VARCHAR2, i_direccion VARCHAR2, i_nacimiento DATE, i_email VARCHAR2
246         ,salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
248     actual socio%ROWTYPE;
        BEGIN
250         INSERT INTO socio VALUES (i_nombre, i_apellidos, i_direccion, i_nacimiento,
            i_email, i_DNI);
252
            SELECT * INTO actual FROM SOCIO WHERE DNI = i_DNI;
254
            IF (actual.nombre<>i_nombre) OR (actual.apellidos<>i_apellidos) OR (actual.DNI<>i_DNI) OR (
         actual.direccion<>i_direccion) OR (actual.fechadenacimiento<>i_nacimiento) OR (actual.email<>
         i_email) THEN
256             salida := false;
            END IF;
258
            PRINTR(nombre_prueba, salida, salidaEsperada);
260
            EXCEPTION
262         WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
264             ROLLBACK;
        END insertar;
266
        PROCEDURE actualizar
268         (nombre_prueba VARCHAR2, a_DNI VARCHAR2, a_direccion VARCHAR2,
            a_email VARCHAR2, salidaEsperada BOOLEAN) AS
270         salida BOOLEAN := true;
            actual socio%ROWTYPE;
272         BEGIN
                UPDATE SOCIO SET DIRECCION = a_direccion, EMAIL = a_email where DNI = a_DNI;
274
                SELECT * INTO actual FROM SOCIO WHERE DNI = a_DNI;
276
                IF (actual.direccion<>a_direccion) OR (actual.email<>a_email) THEN
278                 salida := false;
                END IF;
280
                PRINTR(nombre_prueba, salida, salidaEsperada);
282
                EXCEPTION
284             WHEN OTHERS THEN
                    PRINTR(nombre_prueba, false, salidaEsperada);
286                 ROLLBACK;
            END actualizar;
288
        PROCEDURE eliminar
290         (nombre_prueba VARCHAR2, e_DNI VARCHAR2, salidaEsperada BOOLEAN) AS
            salida BOOLEAN := true;
292         cantidad INT;
            BEGIN
294             DELETE FROM SOCIO WHERE DNI = e_DNI;

296             SELECT COUNT(*) INTO cantidad FROM SOCIO WHERE DNI = e_DNI;

298             IF (cantidad<>0) THEN
                    salida := false;
300             END IF;

302             PRINTR(nombre_prueba, salida, salidaEsperada);

304             EXCEPTION
                WHEN OTHERS THEN
306                 PRINTR(nombre_prueba, false, salidaEsperada);
                    ROLLBACK;
308     END eliminar;
    END PRUEBAS_SOCIO;
310
    /
```

```
312
CREATE OR REPLACE PACKAGE BODY PRUEBAS_PROVEEDOR AS
314
    PROCEDURE inicializar AS
316     BEGIN
        DELETE FROM PROVEEDOR;
318     END inicializar;

320     PROCEDURE insertar
          (nombre_prueba VARCHAR2, i_CIF VARCHAR2, i_nombre VARCHAR2,
322       i_telefono INT, i_email VARCHAR2, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
324     actual proveedor%ROWTYPE;
        BEGIN
326         INSERT INTO PROVEEDOR VALUES (i_CIF, i_nombre, i_telefono, i_email);

328         SELECT * INTO actual FROM PROVEEDOR WHERE CIF = i_CIF;

330         IF (actual.nombre<>i_nombre) OR (actual.telefono<>i_telefono) OR (actual.CIF<>i_CIF) OR (
         actual.email<>i_email) THEN
                salida := false;
332         END IF;

334         PRINTR(nombre_prueba, salida, salidaEsperada);

336         EXCEPTION
            WHEN OTHERS THEN
338             PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
340     END insertar;

342     PROCEDURE actualizar
            (nombre_prueba VARCHAR2, a_cif VARCHAR2, a_nombre VARCHAR2,
344         a_telefono INT, a_email VARCHAR2, salidaEsperada BOOLEAN) AS
            salida BOOLEAN := true;
346         actual proveedor%ROWTYPE;
            BEGIN
348             UPDATE PROVEEDOR SET NOMBRE = a_nombre, TELEFONO = a_telefono, EMAIL = a_email where CIF
         = a_cif;

350             SELECT * INTO actual FROM PROVEEDOR WHERE CIF = a_cif;

352             IF (actual.nombre<>a_nombre) OR
                    (actual.telefono<>a_telefono) OR
354                 (actual.email<>a_email) THEN
                    salida := false;
356             END IF;

358             PRINTR(nombre_prueba, salida, salidaEsperada);

360             EXCEPTION
                WHEN OTHERS THEN
362                 PRINTR(nombre_prueba, false, salidaEsperada);
                    ROLLBACK;
364         END actualizar;

366     PROCEDURE eliminar
            (nombre_prueba VARCHAR2, e_CIF VARCHAR2, salidaEsperada BOOLEAN) AS
368         salida BOOLEAN := true;
            cantidad INT;
370         BEGIN
                DELETE FROM PROVEEDOR WHERE CIF = e_CIF;
372
                SELECT COUNT(*) INTO cantidad FROM PROVEEDOR WHERE CIF = e_cif;
374
                IF (cantidad<>0) THEN
376                 salida := false;
                END IF;
378
                PRINTR(nombre_prueba, salida, salidaEsperada);
380
                EXCEPTION
382             WHEN OTHERS THEN
                    PRINTR(nombre_prueba, false, salidaEsperada);
384                 ROLLBACK;
    END eliminar;
386 END PRUEBAS_PROVEEDOR;
```

```
/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_EMPLAZAMIENTO AS

    PROCEDURE inicializar AS

    BEGIN
        DELETE FROM EMPLAZAMIENTO;
    END inicializar;

    PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_direccion VARCHAR2, i_telefono INT,
        i_tipo VARCHAR2, salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
    actual emplazamiento%ROWTYPE;
    w_ID_Emplazamiento INT;
    BEGIN
        INSERT INTO emplazamiento VALUES(null,i_direccion, i_telefono, i_tipo);

        w_ID_Emplazamiento := S_ID_Emplazamiento.currval;
        SELECT * INTO actual FROM EMPLAZAMIENTO WHERE ID_EMPLAZAMIENTO = w_ID_Emplazamiento;

        IF (actual.direccion<>i_direccion) OR (actual.telefono<>i_telefono) OR (actual.tipo<>i_tipo)
     THEN
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
    END insertar;

    PROCEDURE actualizar
        (nombre_prueba VARCHAR2, a_id_emplazamiento INT, a_direccion VARCHAR2,
        a_telefono INT, salidaEsperada BOOLEAN)AS
        salida BOOLEAN := true;
        actual emplazamiento%ROWTYPE;
        BEGIN
            UPDATE emplazamiento SET DIRECCION = a_direccion, TELEFONO = a_telefono where
     ID_EMPLAZAMIENTO = a_id_emplazamiento;

            SELECT * INTO actual FROM EMPLAZAMIENTO WHERE ID_EMPLAZAMIENTO = a_id_emplazamiento;

            IF (actual.direccion<>a_direccion) OR
                (actual.telefono<>a_telefono)  THEN
                salida := false;
            END IF;

            PRINTR(nombre_prueba, salida, salidaEsperada);

            EXCEPTION
            WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
        END actualizar;

    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_emplazamiento INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
        cantidad INT;
        BEGIN
            DELETE FROM EMPLAZAMIENTO  WHERE ID_EMPLAZAMIENTO = e_id_emplazamiento;

            SELECT COUNT(*) INTO cantidad FROM EMPLAZAMIENTO WHERE ID_EMPLAZAMIENTO =
     e_id_emplazamiento;

            IF (cantidad<>0) THEN
                salida := false;
            END IF;

            PRINTR(nombre_prueba, salida, salidaEsperada);

            EXCEPTION
```

```
            WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
    END eliminar;
END PRUEBAS_EMPLAZAMIENTO;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_PRODUCTO AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM PRODUCTO;
    END inicializar;

    PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_nombre VARCHAR2, i_descripcion VARCHAR2,
        i_categoria VARCHAR2, i_precioProducto INT, i_iva INT, salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
    actual producto%ROWTYPE;
    w_ID_Producto INT;
    BEGIN
        INSERT INTO producto VALUES(null,i_nombre, i_descripcion, i_categoria, i_precioProducto,
     i_iva);

        w_ID_Producto := S_ID_Producto.currval;
        SELECT * INTO actual FROM PRODUCTO WHERE ID_Producto = w_ID_Producto;

        IF (actual.nombre<>i_nombre) OR (actual.descripcion<>i_descripcion) OR (actual.categoria<>
     i_categoria) OR (actual.precioProducto<>i_precioProducto) OR (actual.iva<>i_iva)THEN
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
    END insertar;

    PROCEDURE actualizar
        (nombre_prueba VARCHAR2, a_id_producto INT , a_descripcion VARCHAR2,
        a_precioProducto INT, a_iva INT, salidaEsperada BOOLEAN)AS
        salida BOOLEAN := true;
        actual producto%ROWTYPE;
        BEGIN
            UPDATE producto SET DESCRIPCION = a_descripcion, PRECIOPRODUCTO = a_precioProducto, IVA =
      a_iva
            where ID_Producto = a_id_producto;

            SELECT * INTO actual FROM PRODUCTO WHERE ID_Producto = a_id_producto;
            IF (actual.descripcion<>a_descripcion) OR
                (actual.precioProducto<>a_precioProducto) OR
                (actual.iva<>a_iva) THEN
                salida := false;
            END IF;

            PRINTR(nombre_prueba, salida, salidaEsperada);

            EXCEPTION
            WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
        END actualizar;

    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_producto INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
        cantidad INT;
        BEGIN
            DELETE FROM PRODUCTO  WHERE ID_Producto = e_id_producto;

            SELECT COUNT(*) INTO cantidad FROM PRODUCTO WHERE ID_Producto = e_id_producto;
            IF (cantidad<>0) THEN
                salida := false;
            END IF;
```

```
            PRINTR(nombre_prueba, salida, salidaEsperada);

            EXCEPTION
            WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
    END eliminar;
END PRUEBAS_PRODUCTO;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_FACTURA AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM FACTURA;
    END inicializar;

    PROCEDURE insertar
      (nombre_prueba VARCHAR2, i_fechaDeExpedicion DATE, i_devuelto VARCHAR2,
      i_id_venta INT, i_id_emplazamiento INT, salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
    actual factura%ROWTYPE;
    w_ID_factura INT;
    BEGIN
        INSERT INTO factura VALUES(null, i_fechaDeExpedicion, i_devuelto, i_id_venta,
     i_id_emplazamiento);

        w_ID_factura := S_ID_Factura.currval;
        SELECT * INTO actual FROM factura WHERE ID_factura = w_ID_factura;

        IF (actual.fechaDeExpedicion<>i_fechaDeExpedicion) OR (actual.devuelto<>i_devuelto) OR (
     actual.id_venta<>i_id_venta) OR (actual.id_emplazamiento<>i_id_emplazamiento)THEN
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
    END insertar;

    PROCEDURE actualizar
        (nombre_prueba VARCHAR2, a_id_factura INT,a_devuelto VARCHAR2,
        salidaEsperada BOOLEAN)AS
        salida BOOLEAN := true;
        actual factura%ROWTYPE;
        BEGIN
            UPDATE factura SET DEVUELTO = a_devuelto where ID_factura = a_id_factura;

            SELECT * INTO actual FROM factura WHERE ID_factura = a_id_factura;
            IF (actual.devuelto<>a_devuelto) THEN
                salida := false;
            END IF;

            PRINTR(nombre_prueba, salida, salidaEsperada);

            EXCEPTION
            WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
        END actualizar;

    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_factura INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
        cantidad INT;
        BEGIN
            DELETE FROM factura  WHERE ID_factura = e_id_factura;

            SELECT COUNT(*) INTO cantidad FROM factura WHERE ID_factura = e_id_factura;
            IF (cantidad<>0) THEN
                salida := false;
            END IF;
```

```
610
            PRINTR(nombre_prueba, salida, salidaEsperada);
612
            EXCEPTION
614         WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
616             ROLLBACK;
    END eliminar;
618 END PRUEBAS_factura;

620 /

622 CREATE OR REPLACE PACKAGE BODY PRUEBA_STOCK AS
    PROCEDURE inicializar AS
624 BEGIN
        DELETE FROM stock;
626 END inicializar;

628 PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_id_emplazamiento INT, i_id_producto INT,
630     i_cantidad INT, salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
632 actual stock%ROWTYPE;
    BEGIN
634     INSERT INTO stock VALUES(i_id_emplazamiento, i_id_producto, i_cantidad);

636     SELECT * INTO actual FROM stock WHERE ID_Emplazamiento = i_id_emplazamiento and ID_Producto =
      i_id_producto;

638     IF (actual.id_emplazamiento<>i_id_emplazamiento) OR (actual.id_producto<>i_id_producto) OR (
     actual.cantidad<>i_cantidad) THEN
            salida := false;
640     END IF;

642     PRINTR(nombre_prueba, salida, salidaEsperada);

644     EXCEPTION
        WHEN OTHERS THEN
646         PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
648 END insertar;

650 PROCEDURE actualizar
        (nombre_prueba VARCHAR2, a_id_emplazamiento INT, a_id_producto INT,
652     a_cantidad INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
654     actual stock%ROWTYPE;
        BEGIN
656         UPDATE stock SET CANTIDAD = a_cantidad  where  ID_Emplazamiento = a_id_emplazamiento and
     ID_Producto = a_id_producto;

658         SELECT * INTO actual FROM stock WHERE  ID_Emplazamiento = a_id_emplazamiento and
     ID_Producto = a_id_producto;
            IF (actual.cantidad<>a_cantidad) THEN
660             salida := false;
            END IF;
662
            PRINTR(nombre_prueba, salida, salidaEsperada);
664
            EXCEPTION
666         WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
668             ROLLBACK;
        END actualizar;
670
    PROCEDURE eliminar
672     (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,
        salidaEsperada BOOLEAN) AS
674     salida BOOLEAN := true;
        cantidad INT;
676     BEGIN
            DELETE FROM stock  WHERE ID_Emplazamiento = e_id_emplazamiento and ID_Producto =
     e_id_producto;
678
            SELECT COUNT(*) INTO cantidad FROM stock WHERE ID_Emplazamiento = e_id_emplazamiento and
     ID_Producto = e_id_producto;
680         IF (cantidad<>0) THEN
```

```
                salida := false;
682             END IF;

684             PRINTR(nombre_prueba, salida, salidaEsperada);

686             EXCEPTION
                WHEN OTHERS THEN
688                 PRINTR(nombre_prueba, false, salidaEsperada);
                    ROLLBACK;
690         END eliminar;

692     PROCEDURE stock_correcto
            (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,stock_previo int, cantidad
        INT, salidaEsperada BOOLEAN) AS
694         salida BOOLEAN := true;
            stock_nuevo int;
696         BEGIN
              select cantidad into stock_nuevo from stock where id_emplazamiento = e_id_emplazamiento and
        id_producto= e_id_producto;
698             if(stock_previo-cantidad)<> stock_nuevo then
                    salida := false;
700             END IF;

702             PRINTR(nombre_prueba, salida, salidaEsperada);

704             EXCEPTION
                WHEN OTHERS THEN
706                 PRINTR(nombre_prueba, false, salidaEsperada);
                    ROLLBACK;
708
            END stock_correcto;
710
        PROCEDURE stock_correcto_p
712         (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,stock_previo int, cantidad
        INT, salidaEsperada BOOLEAN) AS
            salida BOOLEAN := true;
714         stock_nuevo int;
            BEGIN
716           select cantidad into stock_nuevo from stock where id_emplazamiento = e_id_emplazamiento and
        id_producto= e_id_producto;
              if(stock_previo+cantidad)<> stock_nuevo then
718               salida := false;
              END IF;
720
              PRINTR(nombre_prueba, salida, salidaEsperada);
722
              EXCEPTION
724           WHEN OTHERS THEN
                  PRINTR(nombre_prueba, false, salidaEsperada);
726               ROLLBACK;
          end stock_correcto_p;
728
          PROCEDURE stock_correcto_t
730         (nombre_prueba VARCHAR2, e_id_emplazamiento_envia int, e_id_emplazamiento_recibe int,
        stock_previo_envia int, stock_previo_recibe int,
            cantidad int, e_id_producto int, salidaEsperada BOOLEAN) AS
732         salida BOOLEAN := true;
            stock_nuevo_envia int;
734         stock_nuevo_recibe int;
            BEGIN
736           select cantidad into stock_nuevo_envia from stock where id_emplazamiento =
         e_id_emplazamiento_envia and id_producto = e_id_producto;
              select cantidad into stock_nuevo_recibe from stock where id_emplazamiento =
         e_id_emplazamiento_recibe and id_producto = e_id_producto;
738           if(stock_previo_envia-cantidad)<>stock_nuevo_envia or (stock_previo_recibe+cantidad)<>
         stock_nuevo_recibe then
                  salida := false;
740           END IF;
              PRINTR(nombre_prueba, salida, salidaEsperada);
742
              EXCEPTION
744           WHEN OTHERS THEN
                  PRINTR(nombre_prueba, false, salidaEsperada);
746               ROLLBACK;
            end stock_correcto_t;
748
END PRUEBA_STOCK;
```

```
/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_VENTA AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM VENTA;
    END inicializar;

    PROCEDURE insertar
      (nombre_prueba VARCHAR2,i_fecha DATE, i_dni VARCHAR2,
        salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
    actual venta%ROWTYPE;
    w_ID_Venta INT;
    BEGIN
        INSERT INTO venta VALUES(null, i_fecha, i_dni);

        w_ID_Venta := S_ID_Venta.currval;
        SELECT * INTO actual FROM venta WHERE ID_Venta = w_ID_Venta;

        IF (actual.fechaVenta<>i_fecha) OR (actual.dni<>i_dni) THEN
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
    END insertar;

    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_venta INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
        cantidad INT;
        BEGIN
            DELETE FROM venta WHERE ID_Venta = e_id_venta;

            SELECT COUNT(*) INTO cantidad FROM venta WHERE ID_Venta = e_id_venta;
            IF (cantidad<>0) THEN
                salida := false;
            END IF;

            PRINTR(nombre_prueba, salida, salidaEsperada);

            EXCEPTION
            WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
    END eliminar;
    /*
    procedure descuento
        (nombre_prueba VARCHAR2, v_id_venta int,v_preciototal number,v_dni_socio varchar2,
     salidaEsperada BOOLEAN) as
        salida BOOLEAN := true;
        p_preciototal number;
        begin
          select preciototal into p_preciototal from venta where id_venta = v_id_venta;
          if (v_dni_socio<>null) then
          if(v_preciototal * 0.95 <> p_preciototal) then
            salida := false;
          END IF;
          END IF;

          printr(nombre_prueba, salida, salidaEsperada);
          EXCEPTION
            WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
    end descuento;*/
END PRUEBAS_VENTA;

/
```

```
826 CREATE OR REPLACE PACKAGE BODY PRUEBAS_PEDIDO AS

828     PROCEDURE inicializar AS
        BEGIN
830         DELETE FROM pedido;
        END inicializar;
832
        PROCEDURE insertar
834       (nombre_prueba VARCHAR2, i_fecha DATE,
          i_id_emplazamiento INT, i_cif VARCHAR2, salidaEsperada BOOLEAN) AS
836     salida BOOLEAN := true;
        actual pedido%ROWTYPE;
838     w_ID_pedido INT;
        BEGIN
840         INSERT INTO pedido VALUES(null, i_fecha, i_id_emplazamiento, i_cif);

842         w_ID_pedido := S_ID_Pedido.currval;
            SELECT * INTO actual FROM pedido WHERE ID_Pedido = w_ID_pedido;
844
            IF (actual.fechaPedido<>i_fecha) OR (actual.id_emplazamiento<>i_id_emplazamiento) OR (actual.
         cif<>i_cif) THEN
846             salida := false;
            END IF;
848
            PRINTR(nombre_prueba, salida, salidaEsperada);
850
            EXCEPTION
852         WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
854             ROLLBACK;
        END insertar;
856
        PROCEDURE eliminar
858         (nombre_prueba VARCHAR2, e_id_pedido INT, salidaEsperada BOOLEAN) AS
            salida BOOLEAN := true;
860         cantidad INT;
            BEGIN
862             DELETE FROM pedido  WHERE ID_Pedido = e_id_pedido;

864             SELECT COUNT(*) INTO cantidad FROM pedido WHERE ID_Pedido = e_id_pedido;
                IF (cantidad<>0) THEN
866                 salida := false;
                END IF;
868
                PRINTR(nombre_prueba, salida, salidaEsperada);
870
                EXCEPTION
872             WHEN OTHERS THEN
                    PRINTR(nombre_prueba, false, salidaEsperada);
874                 ROLLBACK;
        END eliminar;
876 END PRUEBAS_PEDIDO;

878 /

880 CREATE OR REPLACE PACKAGE BODY PRUEBAS_TRASPASO AS

882     PROCEDURE inicializar AS
        BEGIN
884         DELETE FROM traspaso;
        END inicializar;
886
        PROCEDURE insertar
888        (nombre_prueba VARCHAR2, i_fechaTraspaso DATE, i_id_emplazamientoSalida INT,
           i_id_emplazamientoEntrada INT, salidaEsperada BOOLEAN) AS
890     salida BOOLEAN := true;
        actual traspaso%ROWTYPE;
892     w_ID_traspaso INT;
        BEGIN
894         INSERT INTO traspaso VALUES(null, i_fechatraspaso, i_id_emplazamientoSalida,
         i_id_emplazamientoEntrada);

896         w_ID_traspaso := S_ID_traspaso.currval;
            SELECT * INTO actual FROM traspaso WHERE ID_traspaso = w_ID_traspaso;
898
            IF (actual.fechaTraspaso<>i_fechaTraspaso) OR (actual.id_emplazamientoSalida<>
         i_id_emplazamientoSalida) OR (actual.id_emplazamientoEntrada<>i_id_emplazamientoEntrada) THEN
```

```
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
    END insertar;

    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_Traspaso INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
        cantidad INT;
        BEGIN
            DELETE FROM traspaso  WHERE ID_traspaso = e_id_traspaso;

            SELECT COUNT(*) INTO cantidad FROM traspaso WHERE ID_traspaso = e_id_traspaso;
            IF (cantidad<>0) THEN
                salida := false;
            END IF;

            PRINTR(nombre_prueba, salida, salidaEsperada);

            EXCEPTION
            WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
    END eliminar;
END PRUEBAS_TRASPASO;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_Solicitud_Traspaso AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM Solicitud_Traspaso;
    END inicializar;

    PROCEDURE insertar
       (nombre_prueba VARCHAR2, i_fechaSolicitud DATE, i_id_emplazamientoSalida INT,
        i_id_emplazamientoEntrada INT, salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
    actual Solicitud_Traspaso%ROWTYPE;
    w_ID_Solicitud INT;
    BEGIN
        INSERT INTO Solicitud_Traspaso VALUES(null, i_fechaSolicitud, i_id_emplazamientoSalida,
     i_id_emplazamientoEntrada);

        w_ID_Solicitud := S_ID_Solicitud.currval;
        SELECT * INTO actual FROM Solicitud_Traspaso WHERE ID_Solicitud = w_ID_Solicitud;

        IF (actual.fechaSolicitud<>i_fechaSolicitud) OR (actual.id_emplazamientoSalida<>
     i_id_emplazamientoSalida) OR (actual.id_emplazamientoEntrada<>i_id_emplazamientoEntrada) THEN
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
    END insertar;

    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_solicitudTraspaso INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
        cantidad INT;
        BEGIN
            DELETE FROM Solicitud_Traspaso  WHERE ID_Solicitud = e_id_solicitudTraspaso;

            SELECT COUNT(*) INTO cantidad FROM Solicitud_Traspaso WHERE ID_Solicitud =
     e_id_solicitudTraspaso;
            IF (cantidad<>0) THEN
```

```
974                 salida := false;
            END IF;
976
            PRINTR(nombre_prueba, salida, salidaEsperada);
978
            EXCEPTION
980         WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
982             ROLLBACK;
    END eliminar;
984 END PRUEBAS_Solicitud_Traspaso;

986 /

988 CREATE OR REPLACE PACKAGE BODY PRUEBAS_Albaran AS

990     PROCEDURE inicializar AS
        BEGIN
992         DELETE FROM Albaran;
    END inicializar;
994
    PROCEDURE insertar
996        (nombre_prueba VARCHAR2, i_fecha DATE,
          i_id_pedido INT, salidaEsperada BOOLEAN) AS
998     salida BOOLEAN := true;
    actual Albaran%ROWTYPE;
1000    w_ID_Albaran INT;
    BEGIN
1002        INSERT INTO Albaran VALUES(null, i_fecha, i_id_pedido);

1004        w_ID_Albaran := S_ID_Albaran.currval;
            SELECT * INTO actual FROM Albaran WHERE ID_Albaran = w_ID_Albaran;
1006
            IF (actual.fechaFirma<>i_fecha) OR (actual.id_pedido<>i_id_pedido) THEN
1008            salida := false;
            END IF;
1010
            PRINTR(nombre_prueba, salida, salidaEsperada);
1012
            EXCEPTION
1014        WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
1016            ROLLBACK;
    END insertar;
1018
    PROCEDURE eliminar
1020        (nombre_prueba VARCHAR2, e_id_albaran INT, salidaEsperada BOOLEAN) AS
            salida BOOLEAN := true;
1022        cantidad INT;
            BEGIN
1024            DELETE FROM Albaran  WHERE ID_Albaran = e_id_albaran;

1026            SELECT COUNT(*) INTO cantidad FROM Albaran WHERE ID_Albaran = e_id_albaran;
                IF (cantidad<>0) THEN
1028                salida := false;
                END IF;
1030
                PRINTR(nombre_prueba, salida, salidaEsperada);
1032
                EXCEPTION
1034            WHEN OTHERS THEN
                    PRINTR(nombre_prueba, false, salidaEsperada);
1036                ROLLBACK;
    END eliminar;
1038 END PRUEBAS_Albaran;

1040 /

1042 CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_PRODUCTO_TRASPASO AS

1044    PROCEDURE inicializar AS
        BEGIN
1046        DELETE FROM asociacion_Producto_traspaso;
    END inicializar;
1048
    PROCEDURE insertar
1050        (nombre_prueba VARCHAR2, i_id_producto INT, i_id_Traspaso INT,
```

```
            i_cantidad NUMBER, salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
    actual asociacion_Producto_traspaso%ROWTYPE;
    BEGIN
        INSERT INTO asociacion_Producto_traspaso VALUES(i_id_Traspaso, i_id_producto, i_cantidad);

        SELECT * INTO actual FROM asociacion_Producto_traspaso WHERE ID_Producto = i_id_producto and
     ID_Traspaso = i_id_Traspaso;

        IF (actual.id_producto<>i_id_producto) OR (actual.id_Traspaso<>i_id_Traspaso) OR (actual.
     cantidad<>i_cantidad) THEN
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
    END insertar;

    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_Traspaso INT, e_id_producto INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
        cantidad INT;
        BEGIN
            DELETE FROM asociacion_Producto_traspaso  WHERE ID_Producto = e_id_producto and
     ID_Traspaso = e_id_Traspaso;

            SELECT COUNT(*) INTO cantidad FROM asociacion_Producto_traspaso WHERE ID_Producto =
     e_id_producto and ID_Traspaso = e_id_Traspaso;
            IF (cantidad<>0) THEN
                salida := false;
            END IF;

            PRINTR(nombre_prueba, salida, salidaEsperada);

            EXCEPTION
            WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
    END eliminar;
END PRUEBAS_A_PRODUCTO_TRASPASO;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_PRODUCTO_SOLICITUD AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM asociacion_Producto_Solicitud;
    END inicializar;

    PROCEDURE insertar
        (nombre_prueba VARCHAR2,i_id_producto INT, i_id_Solicitud INT,
            i_cantidad NUMBER, salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
    actual asociacion_Producto_Solicitud%ROWTYPE;
    BEGIN
        INSERT INTO asociacion_Producto_Solicitud VALUES(i_id_Solicitud, i_id_producto, i_cantidad);

        SELECT * INTO actual FROM asociacion_Producto_Solicitud WHERE ID_Producto = i_id_producto and
      ID_Solicitud = i_id_Solicitud;

        IF (actual.id_producto<>i_id_producto) OR (actual.id_Solicitud<>i_id_Solicitud) OR (actual.
     cantidad<>i_cantidad) THEN
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
    END insertar;
```

```
1122
    PROCEDURE eliminar
1124        (nombre_prueba VARCHAR2, e_id_Solicitud INT, e_id_producto INT, salidaEsperada BOOLEAN) AS
            salida BOOLEAN := true;
1126        cantidad INT;
            BEGIN
1128            DELETE FROM asociacion_Producto_Solicitud  WHERE ID_Producto = e_id_producto and
     ID_Solicitud = e_id_Solicitud;

1130            SELECT COUNT(*) INTO cantidad FROM asociacion_Producto_Solicitud WHERE ID_Producto =
     e_id_producto and ID_Solicitud = e_id_Solicitud;
                IF (cantidad<>0) THEN
1132                salida := false;
                END IF;
1134
                PRINTR(nombre_prueba, salida, salidaEsperada);
1136
                EXCEPTION
1138            WHEN OTHERS THEN
                    PRINTR(nombre_prueba, false, salidaEsperada);
1140                ROLLBACK;
    END eliminar;
1142 END PRUEBAS_A_PRODUCTO_SOLICITUD;

1144 /

1146 CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_PEDIDO_PRODUCTO AS

1148    PROCEDURE inicializar AS
        BEGIN
1150        DELETE FROM asociacion_Pedido_Producto;
    END inicializar;
1152
    PROCEDURE insertar
1154      (nombre_prueba VARCHAR2, i_id_pedido INT, i_id_producto INT,
          i_cantidad NUMBER, i_precioCompra number, i_iva INT,
1156      salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
1158 actual asociacion_Pedido_Producto%ROWTYPE;
    BEGIN
1160        INSERT INTO asociacion_Pedido_Producto VALUES(i_id_pedido, i_id_producto, i_cantidad,
     i_precioCompra, i_iva);

1162        SELECT * INTO actual FROM asociacion_Pedido_Producto WHERE ID_pedido = i_id_pedido and
     ID_producto = i_id_producto;

1164        IF (actual.id_pedido<>i_id_pedido) OR (actual.id_producto<>i_id_producto) OR (actual.cantidad
     <>i_cantidad) OR (actual.precioCompra<>i_precioCompra) OR (actual.iva<>i_iva) THEN
                salida := false;
1166        END IF;

1168        PRINTR(nombre_prueba, salida, salidaEsperada);

1170        EXCEPTION
            WHEN OTHERS THEN
1172            PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
1174    END insertar;

1176    PROCEDURE eliminar
            (nombre_prueba VARCHAR2, e_id_pedido INT, e_id_producto INT,
1178        salidaEsperada BOOLEAN) AS
            salida BOOLEAN := true;
1180        cantidad INT;
            BEGIN
1182            DELETE FROM asociacion_Pedido_Producto  WHERE ID_pedido = e_id_pedido and ID_producto =
     e_id_producto;

1184            SELECT COUNT(*) INTO cantidad FROM asociacion_Pedido_Producto WHERE ID_pedido =
     e_id_pedido and ID_producto = e_id_producto;
                IF (cantidad<>0) THEN
1186                salida := false;
                END IF;
1188
                PRINTR(nombre_prueba, salida, salidaEsperada);
1190
                EXCEPTION
```

```
            WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
    END eliminar;

END PRUEBAS_A_PEDIDO_PRODUCTO;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_VENTA_PRODUCTO AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM asociacion_Venta_Producto;
    END inicializar;

    PROCEDURE insertar
    (nombre_prueba VARCHAR2, i_id_venta INT, i_id_producto INT,
        i_cantidad NUMBER, i_precioVenta NUMBER, i_ivaVenta NUMBER,
        salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
    actual asociacion_Venta_Producto%ROWTYPE;
    BEGIN
        INSERT INTO asociacion_Venta_Producto VALUES(i_id_venta, i_id_producto, i_cantidad,
     i_precioVenta, i_ivaVenta);

        SELECT * INTO actual FROM asociacion_Venta_Producto WHERE ID_Venta = i_id_venta and
     ID_producto = i_id_producto;

        IF (actual.id_Venta<>i_id_Venta) OR (actual.id_producto<>i_id_producto) OR (actual.cantidad<>
     i_cantidad) OR (actual.precioVenta<>i_precioVenta) OR (actual.ivaVenta<>i_ivaVenta) THEN
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
    END insertar;

    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_venta INT, e_id_producto INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
        cantidad INT;
        BEGIN
            DELETE FROM asociacion_Venta_Producto  WHERE ID_Venta = e_id_venta and ID_producto =
     e_id_producto;

            SELECT COUNT(*) INTO cantidad FROM asociacion_Venta_Producto WHERE ID_Venta = e_id_venta
     and ID_producto = e_id_producto;
            IF (cantidad<>0) THEN
                salida := false;
            END IF;

            PRINTR(nombre_prueba, salida, salidaEsperada);

            EXCEPTION
            WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
    END eliminar;
END PRUEBAS_A_VENTA_PRODUCTO;

/



CREATE OR REPLACE PACKAGE BODY PRUEBAS_FUNCIONES AS

    PROCEDURE precioLinea_A_Pedido
    (nombre_prueba VARCHAR2, ID_PRODUCTO NUMBER, ID_PEDIDO NUMBER, esperado number, salidaEsperada
     BOOLEAN) AS
    salida BOOLEAN := true;
    precio number;
    BEGIN
```

```
        SELECT precioLinea_Aso_Pedido(ID_PRODUCTO,ID_PEDIDO) INTO precio FROM dual;

        IF (precio<>esperado) THEN
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
    END PRECIOLINEA_A_PEDIDO;

    PROCEDURE precioLinea_A_Venta
        (nombre_prueba VARCHAR2, ID_PRODUCTO NUMBER, ID_VENTA NUMBER, esperado number, salidaEsperada
      BOOLEAN) AS
        salida BOOLEAN := true;
    precio number;
    BEGIN
        SELECT precioLinea_Aso_VENTA(ID_PRODUCTO,ID_VENTA) INTO precio FROM dual;

        IF (precio<>esperado) THEN
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
    END PRECIOLINEA_A_VENTA;

    PROCEDURE PRECIO_VENTA
        (nombre_prueba VARCHAR2, ID_Venta NUMBER, esperado number, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
    precio number;
    BEGIN
        SELECT precioTotal_Venta(ID_VENTA) INTO precio FROM dual;

        IF (precio<>esperado) THEN
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
    END PRECIO_VENTA;

    PROCEDURE precio_Factura
        (nombre_prueba VARCHAR2, ID_Venta NUMBER, esperado number, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
    precio number;
    BEGIN
        SELECT precioTotal_Factura(ID_VENTA) INTO precio FROM dual;

        IF (precio<>esperado) THEN
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
    END precio_Factura;

    PROCEDURE precio_Pedido
        (nombre_prueba VARCHAR2, ID_PEDIDO NUMBER, esperado number, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
    precio number;
    BEGIN
```

```
        SELECT precioTotal_Pedido(ID_PEDIDO) INTO precio FROM dual;

        IF (precio<>esperado) THEN
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
    END precio_Pedido;

    PROCEDURE precio_Albaran
        (nombre_prueba VARCHAR2, ID_PEDIDO NUMBER, esperado number, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
    precio number;
    BEGIN
        SELECT precioTotal_Albaran(ID_PEDIDO) INTO precio FROM dual;

        IF (precio<>esperado) THEN
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
    END precio_Albaran;

END PRUEBAS_FUNCIONES;
/
DROP SEQUENCE S_ID_Producto;
DROP SEQUENCE S_ID_Traspaso;
DROP SEQUENCE S_ID_Solicitud;
DROP SEQUENCE S_ID_Pedido;
DROP SEQUENCE S_ID_Albaran;
DROP SEQUENCE S_ID_Factura;
DROP SEQUENCE S_ID_Emplazamiento;
DROP SEQUENCE S_ID_Venta;

CREATE SEQUENCE S_ID_Emplazamiento START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Venta START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Pedido START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Albaran START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Producto START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Traspaso START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Solicitud START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Factura START WITH 1 INCREMENT BY 1 MAXVALUE 9999;

/* EXECUTE DE PRUEBAS */
-- Inicializacion
EXECUTE pruebas_albaran.inicializar();
EXECUTE PRUEBAS_A_PRODUCTO_TRASPASO.inicializar();
EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.inicializar();
EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.inicializar();
EXECUTE PRUEBAS_A_VENTA_PRODUCTO.inicializar();
EXECUTE PRUEBAS_FACTURA.inicializar();
EXECUTE pruebas_venta.inicializar();
EXECUTE pruebas_pedido.inicializar();
EXECUTE pruebas_proveedor.inicializar();
EXECUTE PRUEBA_STOCK.inicializar();
EXECUTE pruebas_producto.inicializar();
EXECUTE PRUEBAS_TRASPASO.inicializar();
EXECUTE PRUEBAS_SOLICITUD_TRASPASO.inicializar();
EXECUTE PRUEBAS_SOCIO.inicializar();
EXECUTE pruebas_emplazamiento.inicializar();
--SOCIO
EXECUTE PRUEBAS_SOCIO.inicializar();
EXECUTE PRUEBAS_SOCIO.insertar('Socio Insertar', 'Daniel', 'Gonzalez', '54148787G',  'Avenida del
     pepinillo', TO_DATE('10102015','DDMMYYYY'), 'dani@us.es', true);
EXECUTE PRUEBAS_SOCIO.insertar('Socio Insertar', 'Jose Luis', 'Marmol Romero', '29613400A',  'Calle
     Virgen', TO_DATE('10101996','DDMMYYYY'), 'jlmr@us.es', true);
```

```
EXECUTE PRUEBAS_SOCIO.actualizar('Socio Actualizar', '54148787G', 'Avenida del pepinillo234', '
    dani@us.es', true);
EXECUTE PRUEBAS_SOCIO.eliminar('Socio Eliminar', '54148787G', true);

--PROVEEDOR
EXECUTE PRUEBAS_PROVEEDOR.inicializar();
EXECUTE PRUEBAS_PROVEEDOR.insertar('Proveedor Insertar', 'B54120214', 'Pimex', 954852320, 'pimex@pi.
    es', true);
EXECUTE PRUEBAS_PROVEEDOR.insertar('Proveedor Insertar', 'B54129999', 'Pimex34', 954896666, '
    pimex34@pi.es', true);
EXECUTE PRUEBAS_PROVEEDOR.actualizar('Proveedor Actualizar', 'B54120214', 'Piexs', 111111111, 'a@pi.
    es', true);
EXECUTE PRUEBAS_PROVEEDOR.eliminar('Proveedor Eliminar', 'B54120214', true);

--PRODUCTO
EXECUTE PRUEBAS_PRODUCTO.inicializar();
COMMIT WORK;
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto con IVA invalido','Chaleco de lana','por una gran
    disenadora','Jerseys',46.95,2,false);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto con categoria invalida','Chaleco de lana','por una gran
    disenadora','Ropa',46.95,0.21,false);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto con precio invalido','Chaleco de lana','por una gran
    disenadora','Jerseys',-2,0.21,false);

EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion','Zapatos cutres','unos zaparos cutres pero
    calentitos','Calzado',12.5,0.21,true);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion','Cool shoes','Zapatos de cuero italiano','
    Calzado',5.95,0.21,true);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion','Sport Shoes','Zapatos de plastico','Calzado'
    ,6.95,0.21,true);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion','Brown Sandals','Sandalias de playa marrones',
    'Calzado',7.95,0.21,true);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion','Chupa de cuero','gran imitacion de la chupa
    de Han Solo','Calzado',500,0.21,true);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion','Chaleco de lana','por una gran disenadora','
    Jerseys',46.95,0.21,true);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion','Sombrero de paja','el gran sombrero de
    Mugiwara','Accesorios',20,0.21,true);
EXECUTE PRUEBAS_PRODUCTO.actualizar('Producto actualizacion',S_ID_Producto.currval-2,'Unos zapatos
    cutres pero calentitos, estan rotos',10.5,0.21,true);
EXECUTE PRUEBAS_PRODUCTO.eliminar('Producto eliminar',S_ID_Producto.currval,true);
--EMPLAZAMIENTO
EXECUTE PRUEBAS_EMPLAZAMIENTO.inicializar();
COMMIT WORK;
EXECUTE PRUEBAS_EMPLAZAMIENTO.insertar('Emplazamiento con tipo invalido', 'Calle de la piruleta 25',
    958632666,'Emplazamiento', false);

EXECUTE PRUEBAS_EMPLAZAMIENTO.insertar('Emplazamiento insercion', 'Calle de la piruleta', 954147741,'
    TIENDA', true);
EXECUTE PRUEBAS_EMPLAZAMIENTO.insertar('Emplazamiento insercion', 'Avenida Reina Mercedes',
    958632147,'TIENDA', true);
EXECUTE PRUEBAS_EMPLAZAMIENTO.insertar('Emplazamiento insercion', 'Calle de la piruleta 24',
    958632147,'TIENDA', true);
EXECUTE PRUEBAS_EMPLAZAMIENTO.insertar('Emplazamiento insercion', 'Calle de la piruleta 25',
    958632666,'ALMACEN', true);
EXECUTE PRUEBAS_EMPLAZAMIENTO.actualizar('Emplazamiento actualizacion',S_ID_Emplazamiento.currval-2,'
    Avenida del pepinillo2', 954147871, true);
EXECUTE PRUEBAS_EMPLAZAMIENTO.eliminar('Emplazamiento eliminar', S_ID_Emplazamiento.currval, true);
--STOCK
EXECUTE PRUEBA_STOCK.inicializar();
EXECUTE PRUEBA_STOCK.insertar('Stock insercion',S_ID_Emplazamiento.currval-2,S_ID_Producto.currval
    -1,10,true);
EXECUTE PRUEBA_STOCK.insertar('Stock insercion',S_ID_Emplazamiento.currval-2,S_ID_Producto.currval
    -2,20,true);
EXECUTE PRUEBA_STOCK.insertar('Stock insercion',S_ID_Emplazamiento.currval-2,S_ID_Producto.currval
    -3,30,true);
EXECUTE PRUEBA_STOCK.insertar('Stock insercion',S_ID_Emplazamiento.currval-1,S_ID_Producto.currval
    -1,11,true);
EXECUTE PRUEBA_STOCK.insertar('Stock insercion',S_ID_Emplazamiento.currval-1,S_ID_Producto.currval
    -2,22,true);
EXECUTE PRUEBA_STOCK.insertar('Stock insercion',S_ID_Emplazamiento.currval-1,S_ID_Producto.currval
    -3,33,true);
EXECUTE PRUEBA_STOCK.actualizar('Stock actualizacion',S_ID_Emplazamiento.currval-2,S_ID_Producto.
    currval -2,30,true);
EXECUTE PRUEBA_STOCK.eliminar('Stock eliminar',S_ID_Emplazamiento.currval-2,S_ID_Producto.currval-3,
    true);
```

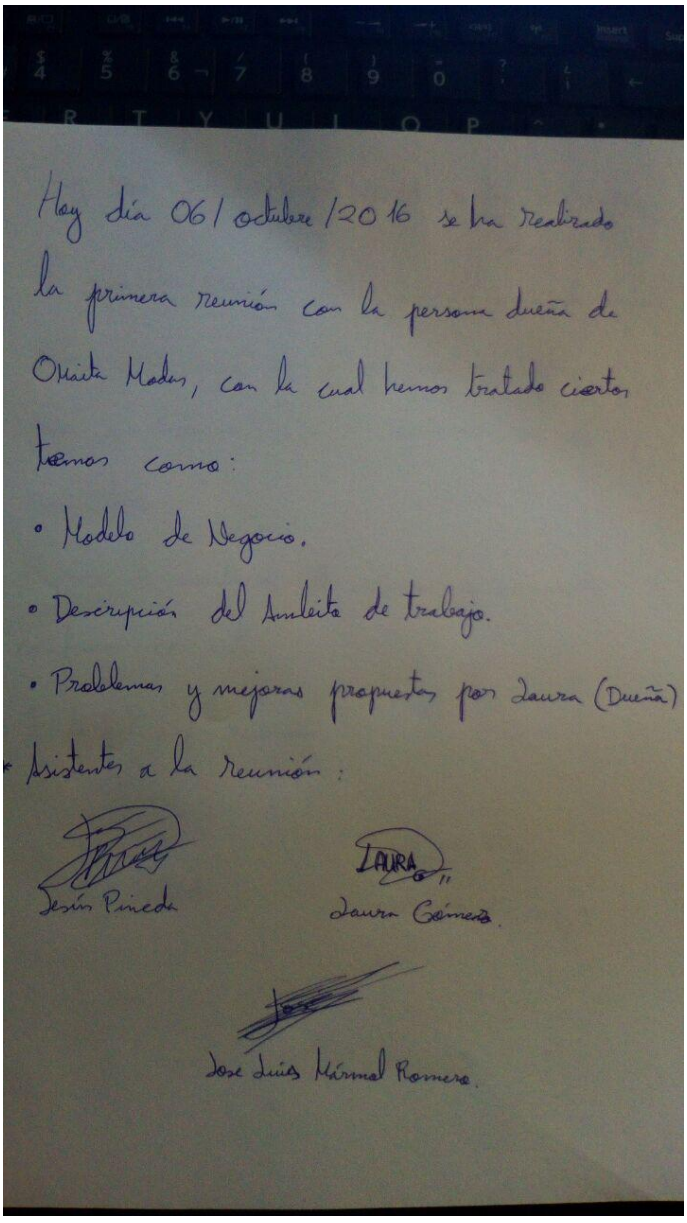
```
--VENTA
EXECUTE pruebas_venta.inicializar();
EXECUTE pruebas_venta.insertar('Venta insercion',sysdate,'29613400A',true);
EXECUTE pruebas_venta.insertar('Venta insercion',sysdate,null,true);
EXECUTE pruebas_venta.insertar('Venta insercion',sysdate,null,true);
EXECUTE pruebas_venta.eliminar('Venta eliminar',S_ID_venta.currval,true);
/*
EXECUTE pruebas_venta.insertar('Venta insercion',0,sysdate,null,true);
EXECUTE PRUEBAS_A_VENTA_PRODUCTO.insertar('Venta-Producto insercion',s_id_venta.currval,s_id_producto
    .currval-1,28,10.5,0.21,0,true);
INSERT INTO FACTURA VALUES(0,0,sysdate,'F',4,3);
*/
--EXECUTE Pruebas_factura.insertar('Factura insercion',0,sysdate,'F',s_id_venta.currval,
    S_ID_Emplazamiento.currval-1,true);
--select S_ID_Emplazamiento.currval-1 from dual;
--ASOCIACION_VENTA_PRODUCTO
EXECUTE PRUEBAS_A_VENTA_PRODUCTO.inicializar();
EXECUTE PRUEBAS_A_VENTA_PRODUCTO.insertar('Venta-Producto insercion',s_id_venta.currval-1,
    s_id_producto.currval-2,3,10.5,0.21,true);
EXECUTE Pruebas_A_venta_producto.insertar('Venta-Producto insercion',s_id_venta.currval-1,
    s_id_producto.currval-1,2,46.95,0.21,true);
EXECUTE PRUEBAS_A_VENTA_PRODUCTO.insertar('Venta-Producto insercion',s_id_venta.currval-2,
    s_id_producto.currval-2,3,10.5,0.21,true);
EXECUTE Pruebas_A_venta_producto.insertar('Venta-Producto insercion',s_id_venta.currval-2,
    s_id_producto.currval-1,2,46.95,0.21,true);
EXECUTE PRUEBAS_A_VENTA_PRODUCTO.eliminar('Venta-Producto eliminar',s_id_venta.currval-2,
    s_id_producto.currval-1,true);

--FACTURA
EXECUTE PRUEBAS_factura.inicializar();
EXECUTE Pruebas_factura.insertar('Factura insercion',sysdate,'F',s_id_venta.currval-2,
    S_ID_Emplazamiento.currval-1,true);
EXECUTE Pruebas_factura.insertar('Factura insercion',sysdate,'F',s_id_venta.currval-1,
    S_ID_Emplazamiento.currval-2,true);
EXECUTE Pruebas_factura.actualizar('Factura actualizar',s_id_factura.currval,'T',true);
--EXECUTE Pruebas_venta.descuento('Descuento factura-venta',s_id_venta.currval-2,125.4,'29613400A',
    true);
--EXECUTE Pruebas_venta.descuento('Descuento factura-venta',s_id_venta.currval-1,125.4,null,true);
EXECUTE Prueba_stock.stock_correcto('Actualizacion de stock tras venta',s_id_emplazamiento.currval-1,
    s_id_producto.currval-2,22,3,true);
EXECUTE Prueba_stock.stock_correcto('Actualizacion de stock tras venta',s_id_emplazamiento.currval-1,
    s_id_producto.currval-2,21,3,false);

--PEDIDO
EXECUTE PRUEBAS_PEDIDO.inicializar();
EXECUTE PRUEBAS_PEDIDO.insertar('Pedido insercion',sysdate,S_ID_Emplazamiento.currval-1,'B54129999',
    true);
EXECUTE PRUEBAS_PEDIDO.insertar('Pedido insercion',sysdate,S_ID_Emplazamiento.currval-2,'B54129999',
    true);
EXECUTE PRUEBAS_PEDIDO.insertar('Pedido insercion',sysdate,S_ID_Emplazamiento.currval-2,'B54129999',
    true);
EXECUTE PRUEBAS_PEDIDO.eliminar('Pedido eliminar',S_ID_Pedido.currval,true);

--PEDIDO_PRODUCTO

EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.inicializar();
EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.insertar('PEDIDO_PRODUCTO insercion',S_ID_Pedido.currval-1,
    s_id_producto.currval-1,21,10.5,0.21,true);
EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.insertar('PEDIDO_PRODUCTO insercion',S_ID_Pedido.currval-2,
    s_id_producto.currval-1,22,46.95,0.21,true);
EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.insertar('PEDIDO_PRODUCTO insercion',S_ID_Pedido.currval-1,
    s_id_producto.currval-2,22,46.95,0.21,true);
EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.eliminar('PEDIDO_PRODUCTO eliminar',S_ID_Pedido.currval-1,
    s_id_producto.currval-2,true);
--ALBARAN

EXECUTE PRUEBAS_ALBARAN.inicializar();
EXECUTE PRUEBAS_ALBARAN.insertar('Albaran insercion',sysdate,S_ID_Pedido.currval-1,true);
EXECUTE PRUEBA_STOCK.stock_correcto_p('Actualizacion de stock tras comprar',s_ID_emplazamiento.
    currval-2,s_ID_producto.currval-1,8,21,true);
EXECUTE PRUEBA_STOCK.stock_correcto_p('Actualizacion de stock tras comprar',s_ID_emplazamiento.
    currval-2,s_ID_producto.currval-1,8,15,false);
EXECUTE PRUEBAS_ALBARAN.insertar('Albaran insercion',sysdate,S_ID_Pedido.currval-2,true);
EXECUTE PRUEBAS_ALBARAN.eliminar('Albaran eliminaxion',S_ID_Albaran.currval,true);

--SOLICITUD_TRASPASO
EXECUTE PRUEBAS_SOLICITUD_TRASPASO.inicializar();
```

```
EXECUTE PRUEBAS_SOLICITUD_TRASPASO.insertar('Solicitud traspaso Insertar', sysdate,
     S_ID_Emplazamiento.currval-1, S_ID_Emplazamiento.currval-2, true);
EXECUTE PRUEBAS_SOLICITUD_TRASPASO.eliminar('Solicitud traspaso Eliminar', S_ID_Solicitud.currval,
     true);
EXECUTE PRUEBAS_SOLICITUD_TRASPASO.insertar('Solicitud traspaso Insertar', sysdate,
     S_ID_Emplazamiento.currval-1, S_ID_Emplazamiento.currval-2, true);
--ASOC_PRODUCTO_SOLICITUD
EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.inicializar();
EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.insertar('A-Producto-Solicitud Insertar', S_ID_Producto.currval
     -2, S_ID_Solicitud.currval, 3, true);
EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.eliminar('A-Producto-Solicitud Eliminar', S_ID_Solicitud.currval
     , S_ID_Producto.currval-2, true);
EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.insertar('A-Producto-Solicitud Insertar', S_ID_Producto.currval
     -2, S_ID_Solicitud.currval, 3, true);
COMMIT WORK;
EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.insertar('Prueba de Trigger solicitud_stock_minimo',
     S_ID_Producto.currval-1, S_ID_Solicitud.currval, 28, false);

--TRASPASO
EXECUTE PRUEBAS_TRASPASO.inicializar();
EXECUTE PRUEBAS_TRASPASO.insertar('Traspaso insercion',sysdate,S_ID_Emplazamiento.currval-1,
     S_ID_Emplazamiento.currval-2,true);
EXECUTE PRUEBAS_TRASPASO.insertar('Traspaso insercion',sysdate,S_ID_Emplazamiento.currval-1,
     S_ID_Emplazamiento.currval-3,true);
EXECUTE PRUEBAS_TRASPASO.eliminar('Traspaso eliminar',S_ID_traspaso.currval,true);

--ASOC_PRODUCTO_TRASPASO
EXECUTE PRUEBAS_A_PRODUCTO_TRASPASO.inicializar();
EXECUTE PRUEBAS_A_PRODUCTO_TRASPASO.insertar('A-Producto-Traspaso Insertar', S_ID_Producto.currval-1,
      S_ID_traspaso.currval-1, 4, true);
EXECUTE PRUEBA_STOCK.STOCK_CORRECTO_T('Actualizacion de stock tras traspaso',s_ID_emplazamiento.
     currval-1, s_ID_emplazamiento.currval-2,33,29,4,s_ID_producto.currval-1,true);
EXECUTE PRUEBAS_A_PRODUCTO_TRASPASO.eliminar('A-Producto-Traspaso Eliminar', S_ID_traspaso.currval-1,
      S_ID_Producto.currval-1, true);
--Pruebas de funciones


EXECUTE PRUEBAS_FUNCIONES.precioLinea_A_Pedido('Funcion precio linea aso-pedido',9,2,220.5,true);
EXECUTE PRUEBAS_FUNCIONES.precioLinea_A_Venta('Funcion precio linea aso-venta',9,2,93.9,true);
EXECUTE PRUEBAS_FUNCIONES.precio_Venta('Funcion calculo precio total venta',1,31.5,true);
EXECUTE PRUEBAS_FUNCIONES.precio_Factura('Funcion calculo precio total factura con socio',1,29.93,
     true);
EXECUTE PRUEBAS_FUNCIONES.precio_Factura('Funcion calculo precio total factura sin socio',2,125.4,
     true);
EXECUTE PRUEBAS_FUNCIONES.precio_Pedido('Funcion calculo precio total pedido',2,220.5,true);
EXECUTE PRUEBAS_FUNCIONES.precio_Albaran('Funcion calculo precio total albaran',2,220.5,true);
-- Tiene que dar 220.5
--SELECT precioLinea_Aso_Pedido(9,2) FROM DUAL;

-- Tiene que dar 220.5
--select preciototal_pedido(2) from dual;

-- Tiene que dar 220.5
--select preciototal_albaran(2) from dual;

-- Tiene que dar 31.5
--select preciototal_venta(2) from dual;

-- Tiene que dar 31.5*0.95 = 29.93
--select preciototal_factura(1) from dual;

--Tienda que dar 93.9
--SELECT precioLinea_Aso_Venta(9,2) FROM DUAL;

-- Select que muestra los productos ofrecidos y disponibles
--SELECT nombre,id_emplazamiento from producto inner join stock on stock.ID_PRODUCTO = producto.
     id_producto;
```

sql/pruebas.sql

10. Apéndice

10.1. Actas de reunión



Hoy día 06/octubre/2016 se ha realizado la primera reunión con la persona dueña de Oficina Modas, con la cual hemos tratado ciertos temas como:

- Modelo de Negocio.
- Descripción del Análisis de trabajo.
- Problemas y mejoras propuestas por Laura (Dueña)

+ Asistentes a la reunión:

Jesús Pineda

Laura Gómez.

José Luis Mármol Romero.

Hoy Día 20/octubre/2016 se ha realizado la segunda reunión con Laura (Dueña de Omaita Kodas) para enseñarle el proceso del trabajo para que todo estuviese de acuerdo con sus expectativas.

Después de mostrarle el avance se encuentra conforme con este.

* Asistentes a la reunión:

Jesús Pineda

LAURA G.

Laura Gómez