

UNIVERSIDAD DE SEVILLA

IISSI

PROYECTO

Modas Omaita

Autores:

Daniel González Corzo

Jesús Pineda Márquez

José Luis Mármol Romero

Roberto Hueso Gómez

21 de junio de 2017



Escuela Técnica Superior de
Ingeniería Informática

Índice

1. Introducción	3
2. Glosario de Términos	3
3. Modelo de negocio	
3.1. Proceso de venta	
3.2. Proceso de compra al proveedor	
3.3. Proceso de disponibilidad	
3.4. Proceso de creación de socios	
4. Visión General del Sistema	
4.1. Descripción	
5. Catálogo de requisitos	
5.1. Requisitos generales	
5.2. Requisitos de información	
5.3. Requisitos funcionales	
5.4. Requisitos no funcionales	
5.5. Reglas de negocio	
6. Modelo Conceptual	
6.1. Diagramas de clases UML	
6.2. Escenarios de prueba	
7. Matrices de trazabilidad	
7.1. Pruebas de aceptación frente a requisitos	
7.2. Pruebas de aceptación frente a escenarios de prueba	
7.3. Tipos de UML frente a Requisitos	
8. Modelos relacionales	
8.1. 3FN Venta	
8.2. 3FN Traspaso - Pedido	
9. Código SQL de la base de datos	
9.1. Tablas	
9.2. Funciones y Procedures	
9.3. Triggers	

9.4. Pruebas	
10. Apéndice	
10.1. Actas de reunión	

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

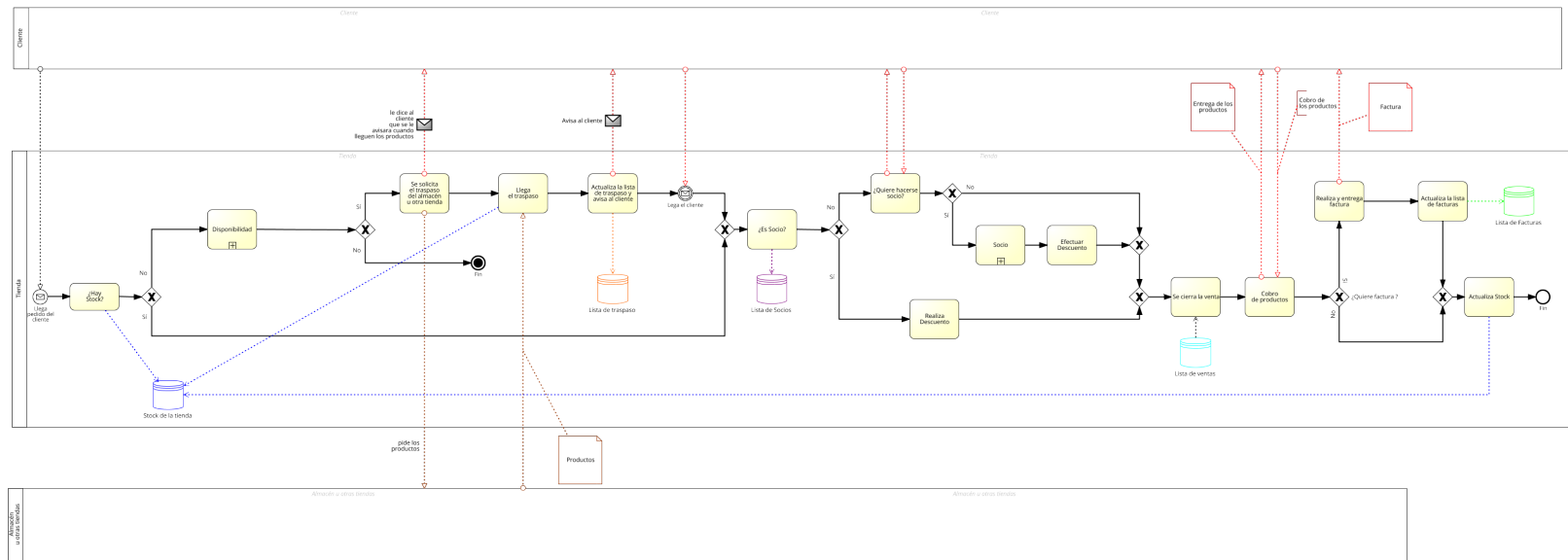
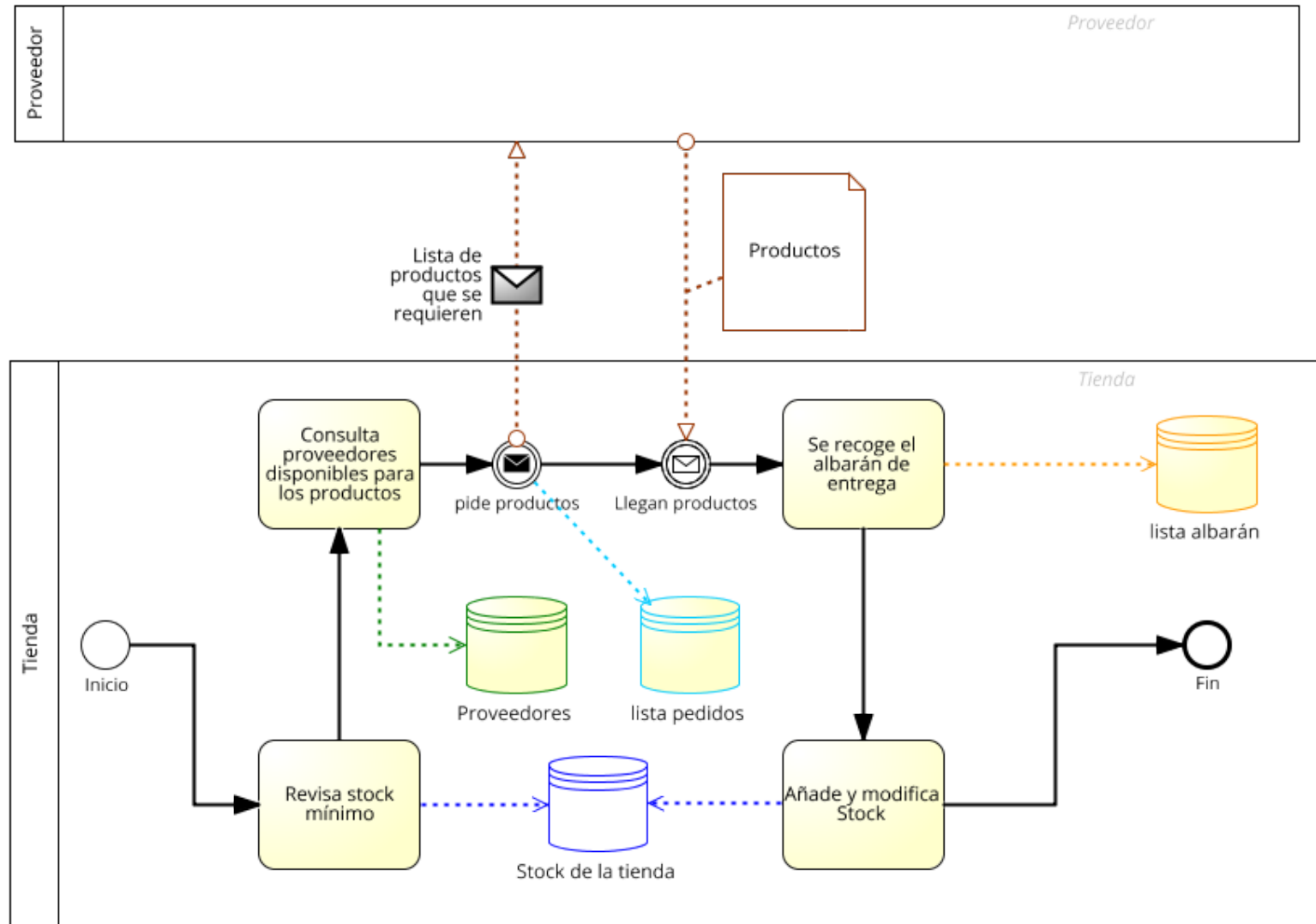
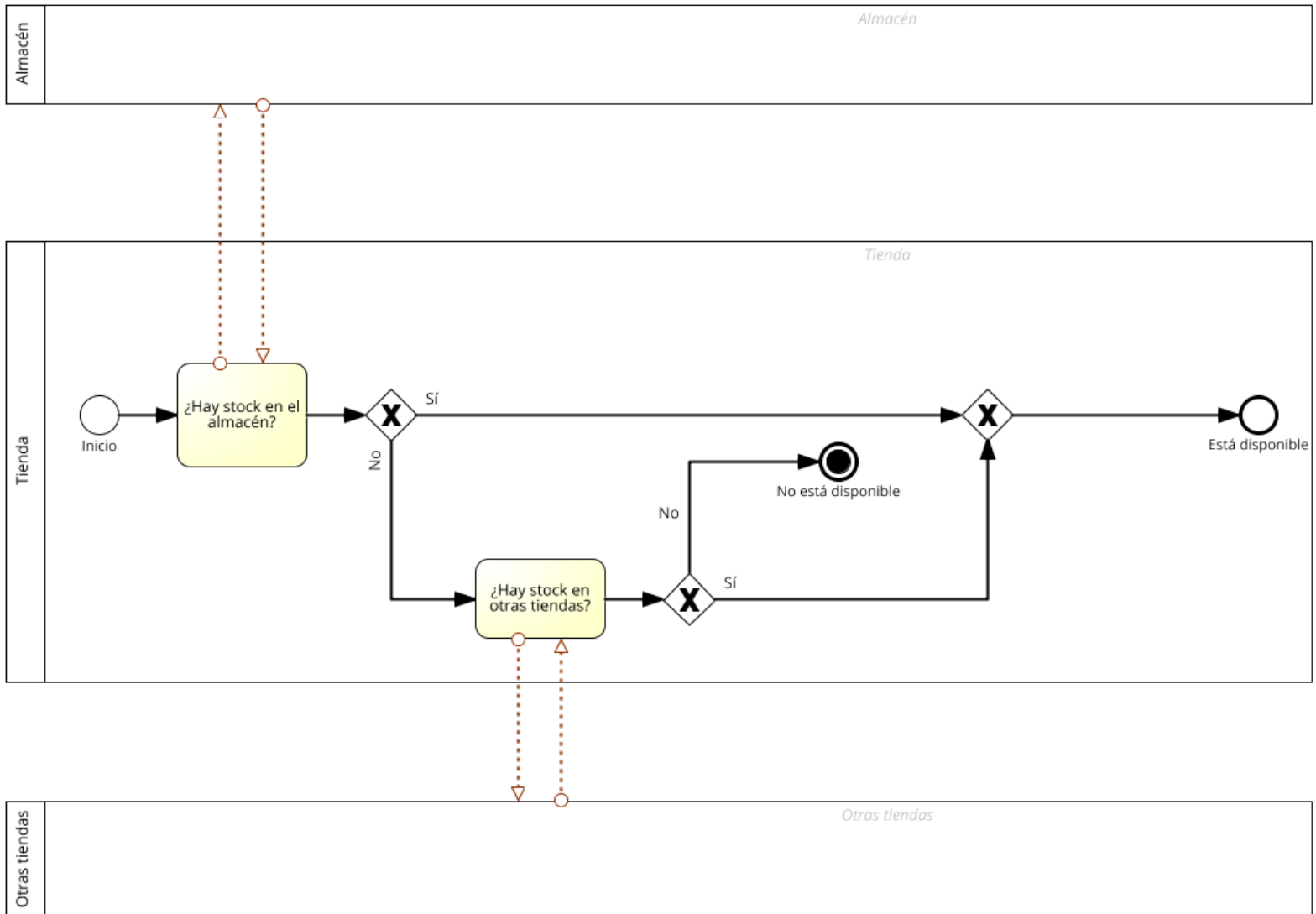


Figura 1: Proceso de venta

3.2. Proceso de compra al proveedor



3.3. Proceso de disponibilidad



3.4. Proceso de creación de socios

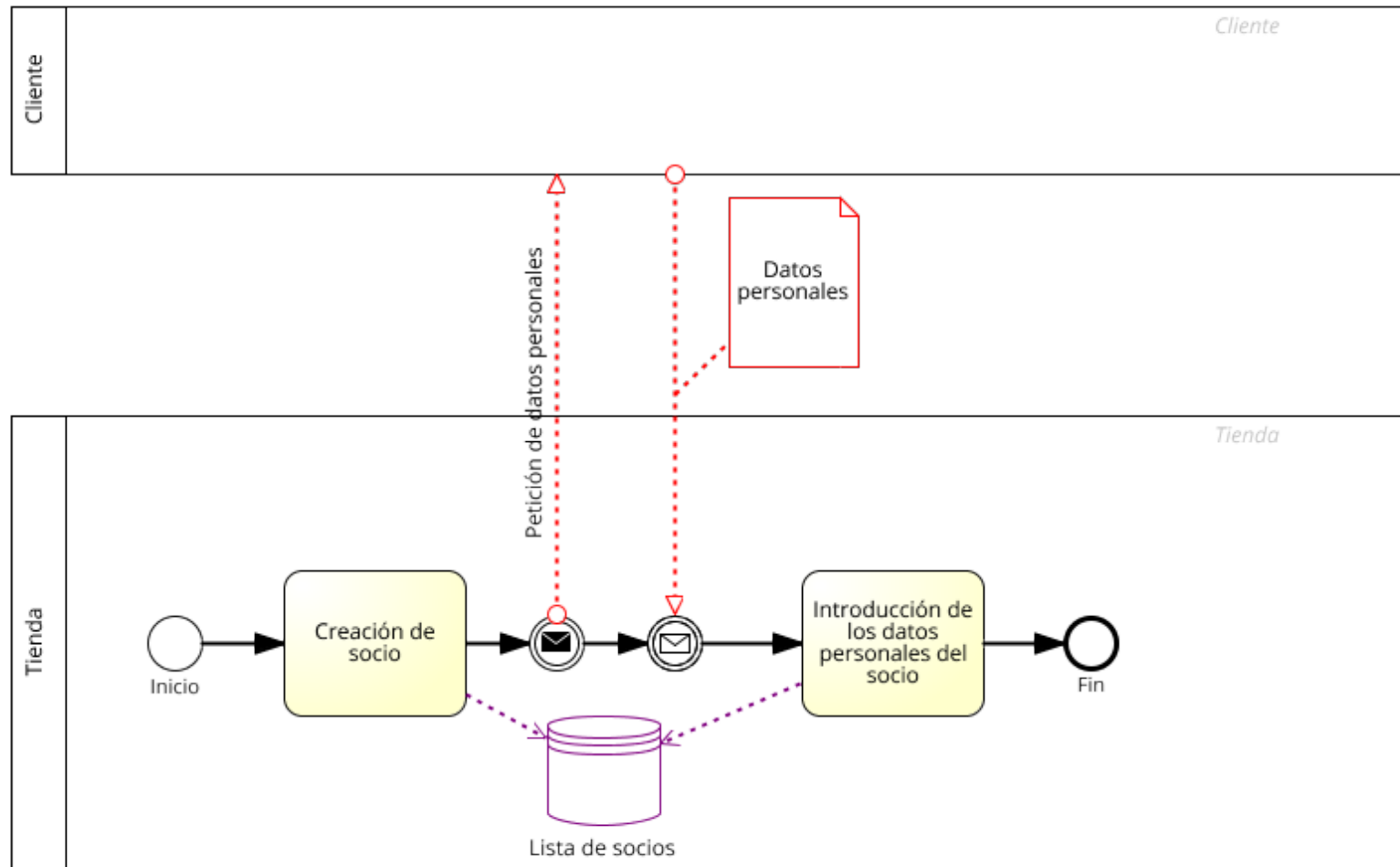


Figura 4: Proceso de creación de socios

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

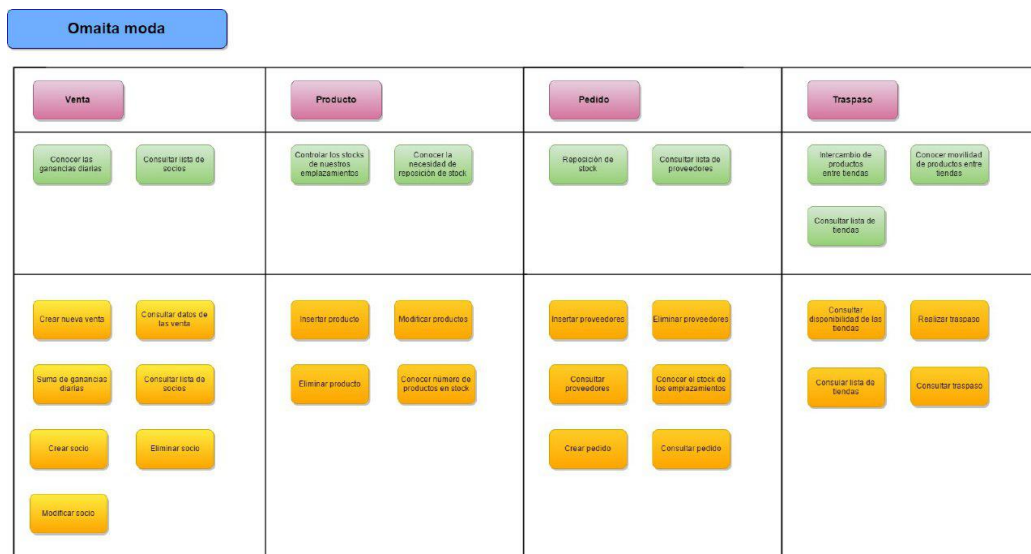


Figura 5: Mapa de requisitos

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 correcta 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 trasposos 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 trasposos 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

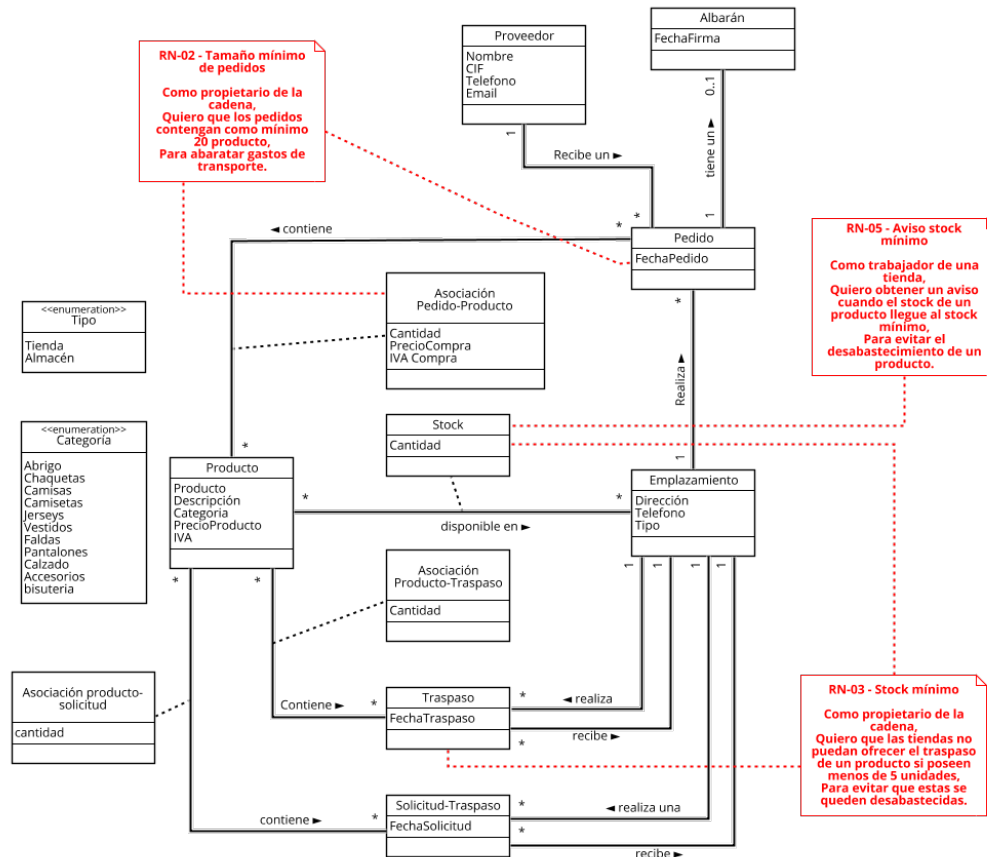


Figura 6: UML Traspaso / Pedido

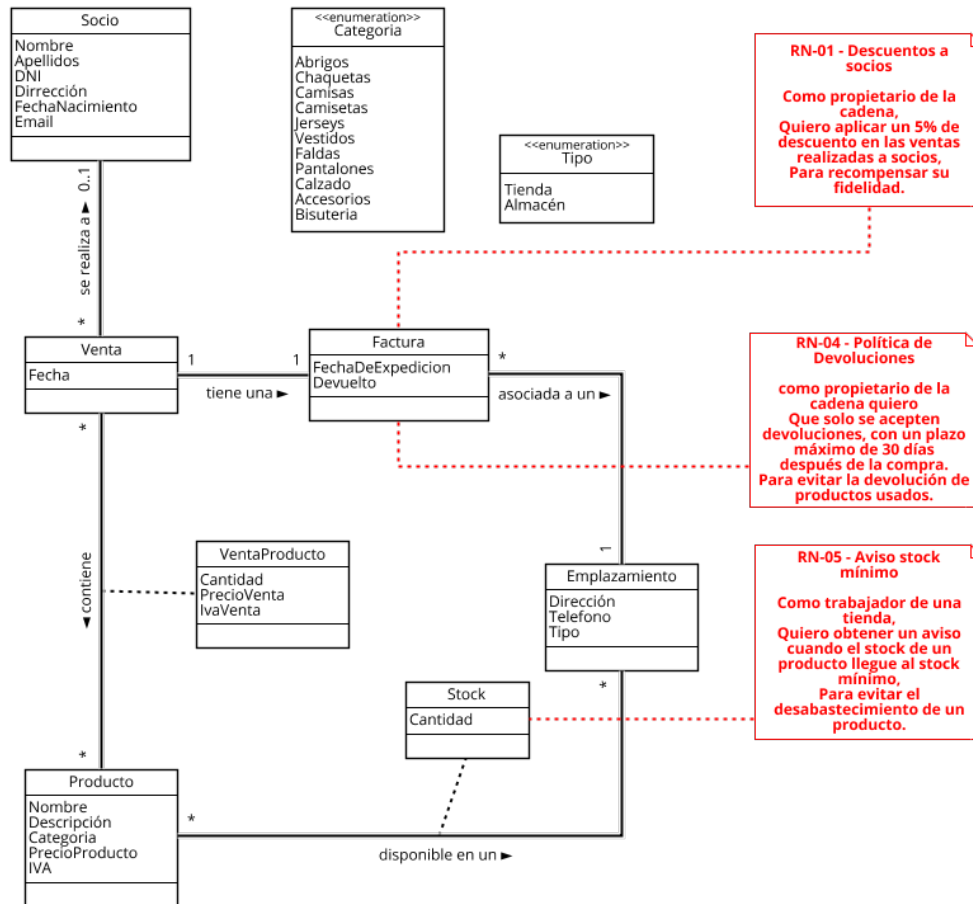


Figura 7: UML Venta

6.2. Escenarios de prueba

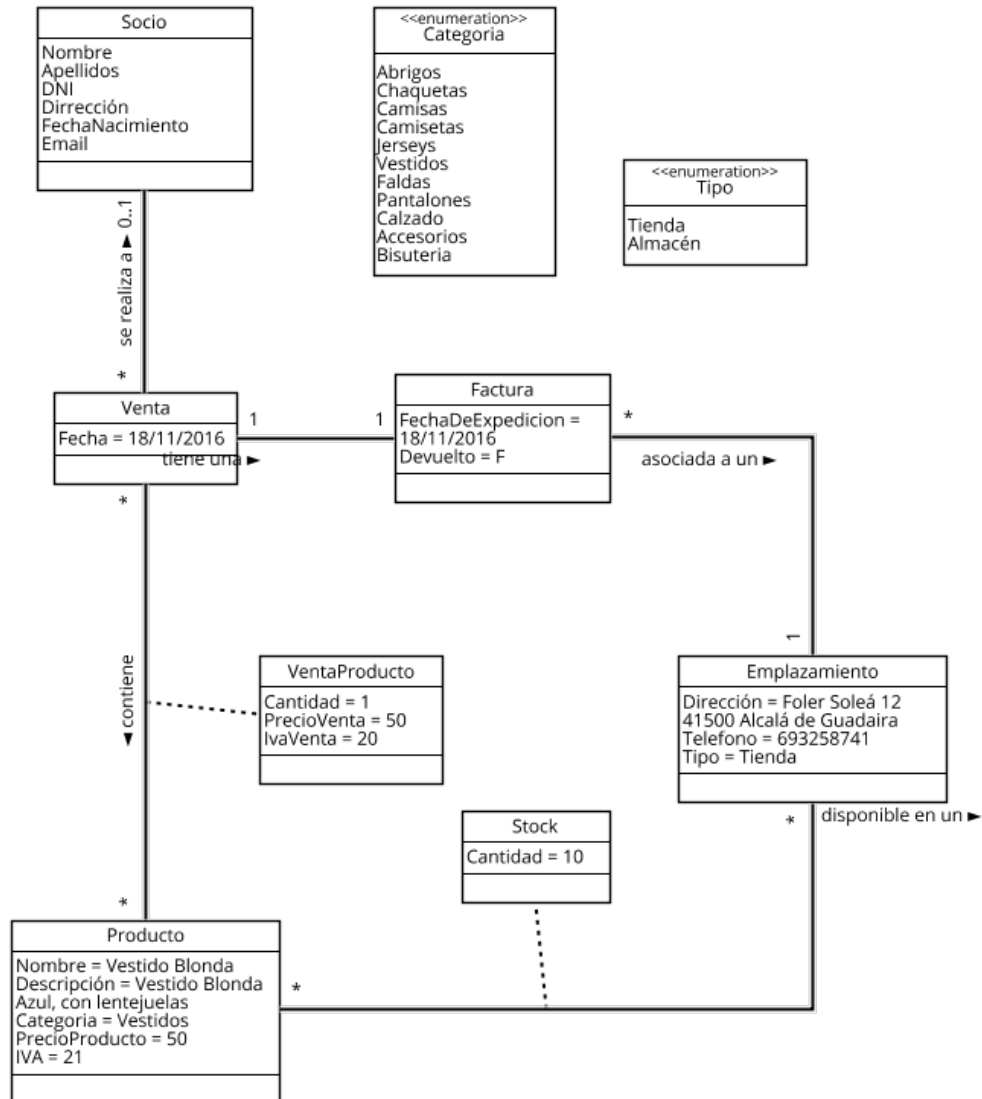


Figura 8: UML Venta NO Socio

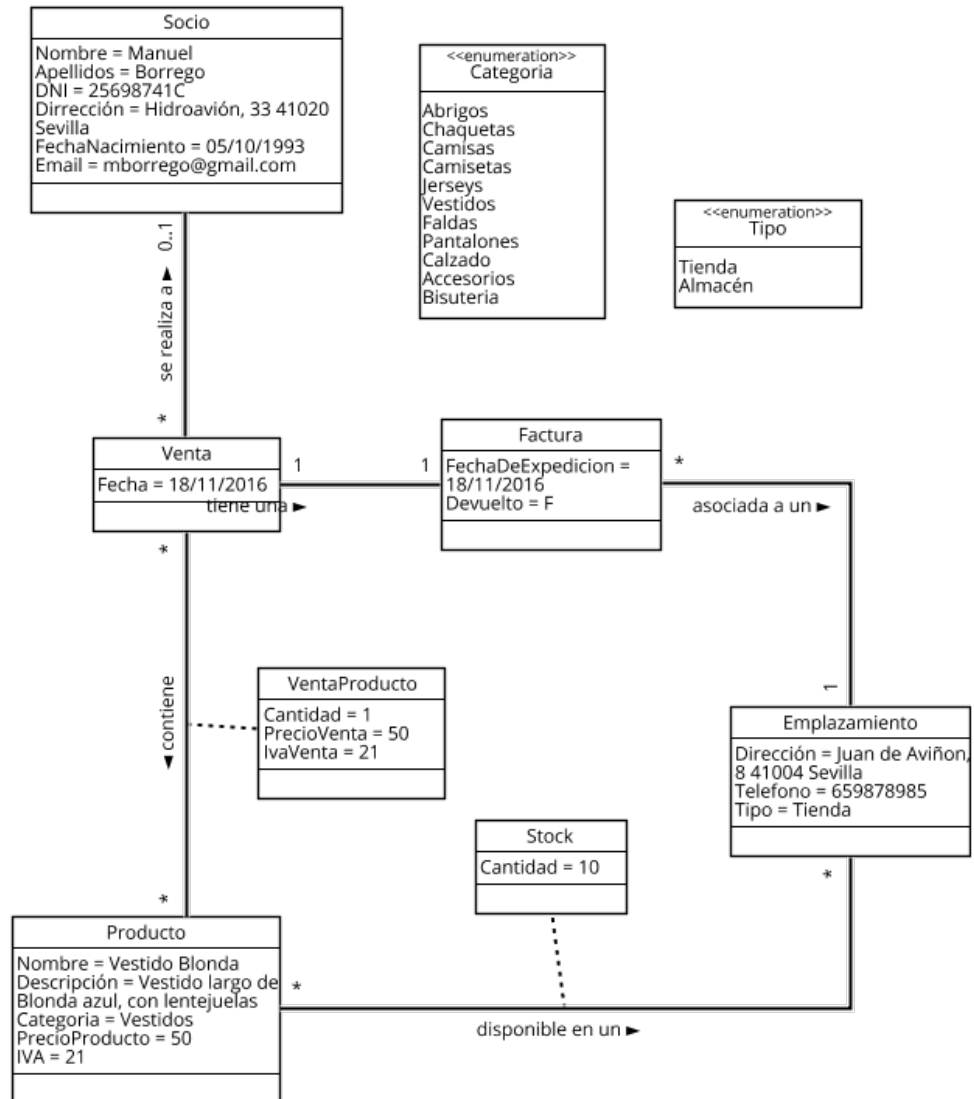


Figura 9: UML Venta Socio

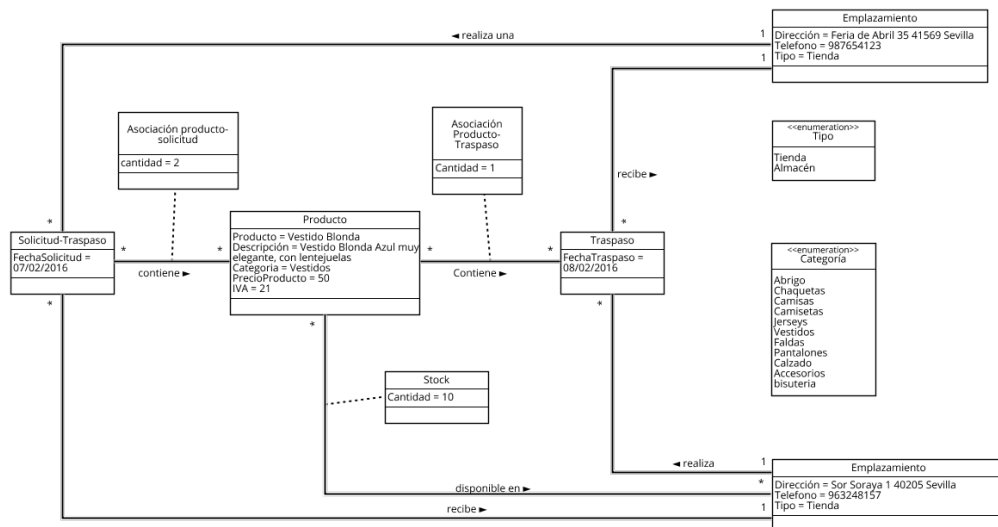


Figura 10: UML Traspaso Tienda - Tienda

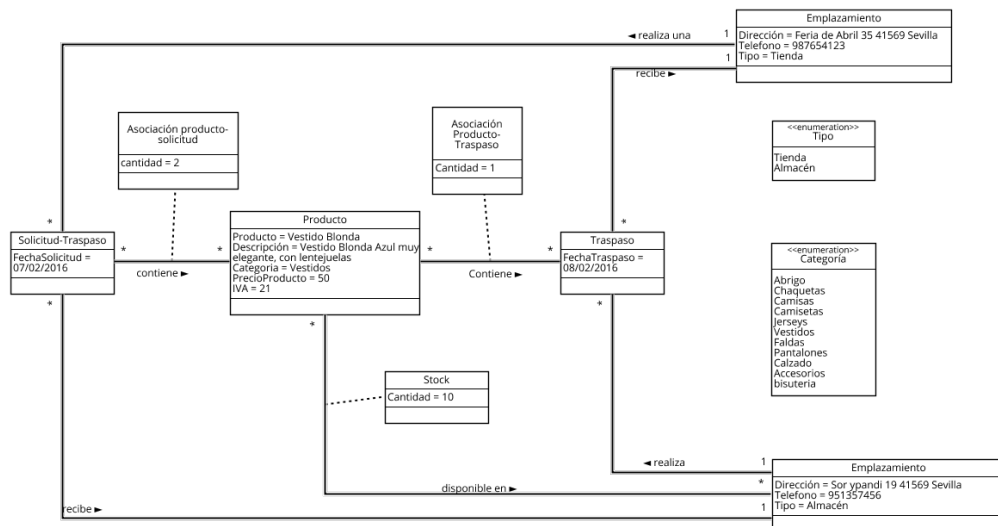


Figura 11: UML Traspaso Tienda - Almacén

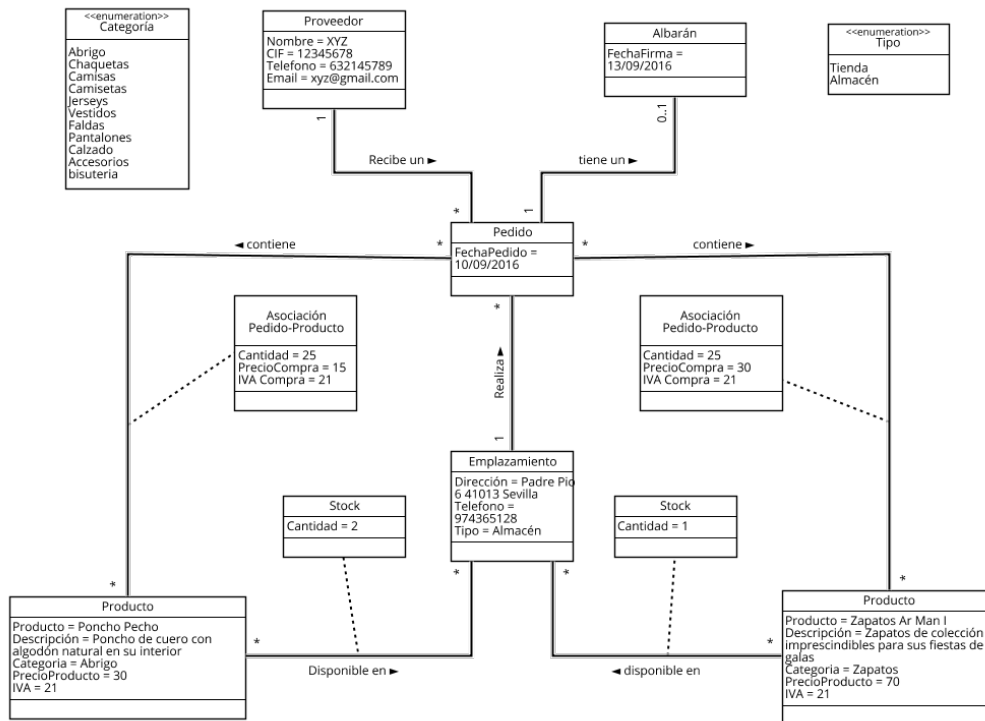


Figura 12: UML Pedido Tienda - Proveedor

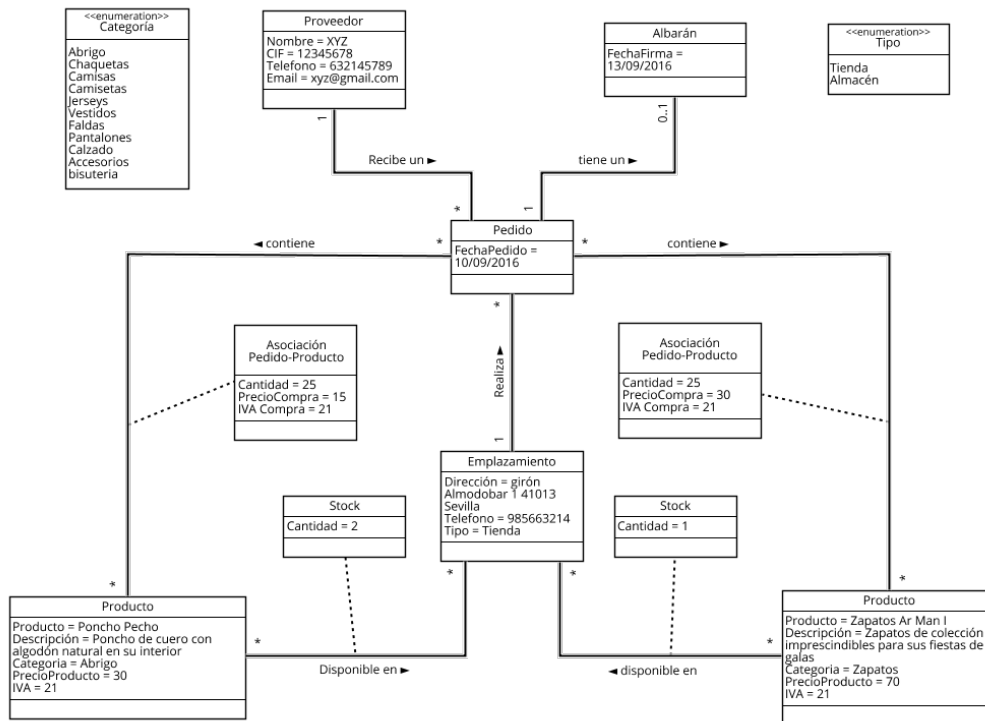


Figura 13: UML Pedido Almacén - Proveedor

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																		X
	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

7.3. Tipos de UML frente a Requisitos

Cuadro 5: 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 6: Reglas de negocio

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

Cuadro 7: Requisitos Funcionales

Venta	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 foreigners 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.

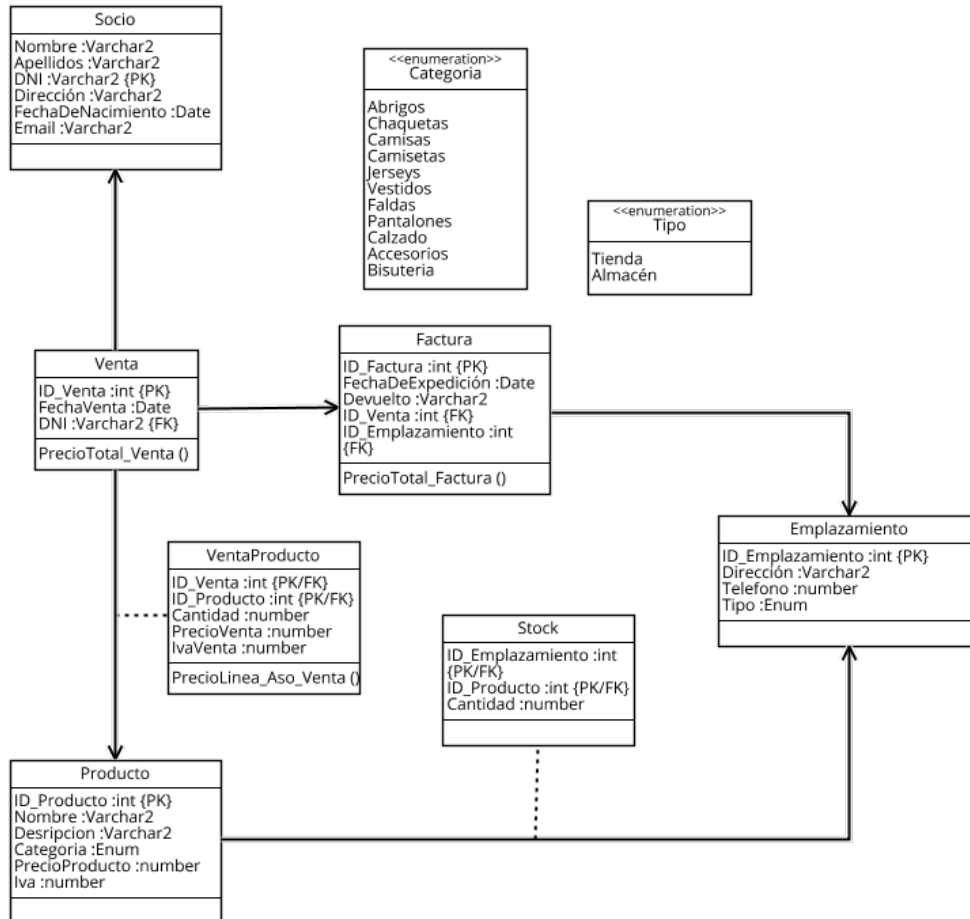


Figura 14: 3FN Venta

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 foreign 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-Traspaso, 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 tratar 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-Traspaso Tiene una relación n - m con Producto, 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_SOLICITUD(primary key y foreign key). Estas claves son provocadas por la relación n - m entre Solicitud-Traspaso y Producto.


```

9| DROP TABLE PROVEEDOR;
10| DROP TABLE STOCK;
11| DROP TABLE PRODUCTO;
12| DROP TABLE TRASPASO;
13| DROP TABLE SOLICITUD_TRASPASO;
14| DROP TABLE SOCIO;
15| DROP TABLE EMPLAZAMIENTO;

17|
18| CREATE TABLE EMPLAZAMIENTO(
19|     ID_Emplazamiento int PRIMARY KEY,
20|     Direccion VARCHAR2(100) NOT NULL,
21|     Telefono NUMBER(9),
22|     Tipo VARCHAR2(10) CHECK( Tipo IN('TIENDA','ALMACEN'))
23| );

25| CREATE TABLE SOCIO (
26|     Nombre VARCHAR2(25) NOT NULL,
27|     Apellidos VARCHAR2(50) NOT NULL,
28|     Direccion VARCHAR2(200) NOT NULL,
29|     FechaDeNacimiento DATE NOT NULL,
30|     Email VARCHAR2(50) NOT NULL,
31|     DNI VARCHAR2(9) PRIMARY KEY
32| );

33|
34| CREATE TABLE VENTA (
35|     ID_VENTA int PRIMARY KEY,
36|     FechaVenta DATE NOT NULL,
37|     DNI VARCHAR2(9),
38|     FOREIGN KEY(DNI) REFERENCES SOCIO
39| );

41| CREATE TABLE FACTURA (
42|     ID_FACTURA int PRIMARY KEY,
43|     FechaDeExpedicion DATE DEFAULT SYSDATE,
44|     Devuelto VARCHAR2(1) CHECK( Devuelto IN('F','T')),
45|     ID_Venta int,
46|     ID_Emplazamiento int,
47|     FOREIGN KEY (ID_Venta) REFERENCES VENTA,
48|     FOREIGN KEY (ID_Emplazamiento) REFERENCES Emplazamiento
49| );

51| CREATE TABLE PROVEEDOR (
52|     CIF VARCHAR2(9) PRIMARY KEY,
53|     Nombre VARCHAR2(75) NOT NULL,
54|     Telefono NUMBER(9) NOT NULL,
55|     Email VARCHAR2(50) NOT NULL
56| );

57| CREATE TABLE PEDIDO(
58|     ID_Pedido int PRIMARY KEY,
59|     FechaPedido DATE NOT NULL,
60|     ID_Emplazamiento int,
61|     CIF VARCHAR2(9),
62|     FOREIGN KEY (CIF) REFERENCES PROVEEDOR,
63|     FOREIGN KEY (ID_Emplazamiento) REFERENCES EMPLAZAMIENTO
64| );

65|
66| CREATE TABLE ALBARAN (
67|     ID_Albaran int NOT NULL,
68|     FechaFirma DATE NOT NULL,
69|     ID_Pedido INT PRIMARY KEY,
70|     FOREIGN KEY (ID_Pedido) REFERENCES PEDIDO);

73|
74| CREATE TABLE PRODUCTO(
75|     ID_Producto int PRIMARY KEY,
76|     Nombre VARCHAR2(50) NOT NULL,
77|     Descripcion VARCHAR2(300) NOT NULL,
78|     Categoria VARCHAR2(20) CHECK ( Categoria IN ('Abrigos','Chaquetas','Camisas',
79|         'Camisetas','Jerseys','Vestidos','Faldas',
80|         'Pantalones','Calzado','Accesorios','Bisuteria')),
81|     PrecioProducto NUMBER NOT NULL check(PrecioProducto>=0),
82|     IVA NUMBER NOT NULL check(IVA>=0 AND IVA<=1)
83| );

85| CREATE TABLE STOCK(

```

```

87     ID_Emplazamiento int,
88     ID_Producto int,
89     PRIMARY KEY (ID_Emplazamiento, ID_Producto),
90     Cantidad NUMBER(6) NOT NULL check(Cantidad>=0),
91     FOREIGN KEY (ID_Emplazamiento) REFERENCES EMPLAZAMIENTO,
92     FOREIGN KEY (ID_Producto) REFERENCES PRODUCTO
93 );
94
95 CREATE TABLE ASOCIACION_PEDIDO_PRODUCTO(
96     ID_PEDIDO int,
97     ID_PRODUCTO int,
98     PRIMARY KEY (ID_PEDIDO, ID_PRODUCTO),
99     Cantidad NUMBER(10) NOT NULL check(Cantidad>=20),
100    PrecioCompra NUMBER NOT NULL check(PrecioCompra>=0),
101    IVA NUMBER NOT NULL check(IVA>=0 AND IVA<=1),
102    FOREIGN KEY (ID_PEDIDO) REFERENCES PEDIDO,
103    FOREIGN KEY (ID_PRODUCTO) REFERENCES PRODUCTO
104 );
105
106 CREATE TABLE TRASPASO(
107     ID_Traspaso int PRIMARY KEY,
108     FechaTraspaso DATE NOT NULL,
109     ID_EmplazamientoSalida int,
110     ID_EmplazamientoEntrada int,
111     FOREIGN KEY (ID_EmplazamientoSalida) REFERENCES EMPLAZAMIENTO,
112     FOREIGN KEY (ID_EmplazamientoEntrada) REFERENCES EMPLAZAMIENTO
113 );
114
115 CREATE TABLE ASOCIACION_PRODUCTO_TRASPASO(
116     ID_Traspaso int,
117     ID_Producto int,
118     PRIMARY KEY (ID_Producto, ID_Traspaso),
119     Cantidad NUMBER(10) NOT NULL check(Cantidad>=0),
120     FOREIGN KEY (ID_Traspaso) REFERENCES TRASPASO,
121     FOREIGN KEY (ID_Producto) REFERENCES producto
122 );
123
124
125 CREATE TABLE SOLICITUD_TRASPASO(
126     ID_Solicitud int PRIMARY KEY,
127     FechaSolicitud DATE NOT NULL,
128     ID_EmplazamientoSalida int,
129     ID_EmplazamientoEntrada int,
130     FOREIGN KEY (ID_EmplazamientoSalida) REFERENCES EMPLAZAMIENTO,
131     FOREIGN KEY (ID_EmplazamientoEntrada) REFERENCES EMPLAZAMIENTO
132 );
133
134
135 CREATE TABLE ASOCIACION_PRODUCTO_SOLICITUD(
136     ID_Solicitud int,
137     ID_Producto int,
138     PRIMARY KEY (ID_Producto, ID_Solicitud),
139     Cantidad NUMBER(10) NOT NULL check(Cantidad>=0),
140     FOREIGN KEY (ID_Solicitud) REFERENCES SOLICITUD_TRASPASO,
141     FOREIGN KEY (ID_Producto) REFERENCES producto
142 );
143
144 CREATE TABLE ASOCIACION_VENTA_PRODUCTO(
145     ID_Venta int,
146     ID_Producto int,
147     PRIMARY KEY (ID_Venta, ID_Producto),
148     FOREIGN KEY (ID_Venta) REFERENCES VENTA,
149     FOREIGN KEY (ID_Producto) REFERENCES PRODUCTO,
150     Cantidad NUMBER(6) check(Cantidad>=0),
151     PrecioVenta NUMBER check(PrecioVenta>=0),
152     IvaVenta NUMBER check(IvaVenta>=0 AND IvaVenta<=1)
153 );
154
155 /* SECUENCIAS */
156
157 DROP SEQUENCE S_ID_Producto;
158 DROP SEQUENCE S_ID_Traspaso;
159 DROP SEQUENCE S_ID_Solicitud;
160 DROP SEQUENCE S_ID_Pedido;
161 DROP SEQUENCE S_ID_Albaran;
162 DROP SEQUENCE S_ID_Factura;
163 DROP SEQUENCE S_ID_Emplazamiento;

```

```

163 DROP SEQUENCE S_ID_Venta;
165
167
169 CREATE SEQUENCE S_ID_Emplazamiento START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
171 CREATE SEQUENCE S_ID_Venta START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
173 CREATE SEQUENCE S_ID_Pedido START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
175 CREATE SEQUENCE S_ID_Albaran START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
177 CREATE SEQUENCE S_ID_Producto START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
179 CREATE SEQUENCE S_ID-Traspaso START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
181 CREATE SEQUENCE S_ID_Solicitud START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
183 CREATE SEQUENCE S_ID_Factura START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
185
187 CREATE OR REPLACE TRIGGER crea_ID_Pedido
189 BEFORE INSERT ON Pedido
191 FOR EACH ROW
193 BEGIN
195 SELECT S_ID_Pedido.NEXTVAL INTO:NEW.ID_Pedido FROM DUAL;
197 END crea_ID_Pedido;
199
201 /
203
205 CREATE OR REPLACE TRIGGER crea_ID_Albaran
207 BEFORE INSERT ON ALBARAN
209 FOR EACH ROW
211 BEGIN
213 SELECT S_ID_Albaran.NEXTVAL INTO:NEW.ID_Albaran FROM DUAL;
215 END crea_ID_Albaran;
217
219 /
221
223 CREATE OR REPLACE TRIGGER Crea_ID_Producto
225 BEFORE INSERT ON PRODUCTO
227 FOR EACH ROW
229 BEGIN
231 SELECT S_ID_Producto.NEXTVAL INTO:NEW.ID_Producto FROM DUAL;
233 END Crea_Nuevo_Producto;
235
237 /
239
241 CREATE OR REPLACE TRIGGER Crea_ID-Traspaso
243 BEFORE INSERT ON TRASPASO
245 FOR EACH ROW
247 BEGIN
249 SELECT S_ID-Traspaso.NEXTVAL INTO:NEW.ID-Traspaso FROM DUAL;
251 END Crea_Nuevo-Traspaso;
253
255 /
257
259 CREATE OR REPLACE TRIGGER Crea_ID_Solicitud
261 BEFORE INSERT ON SOLICITUD_TRASPASO
263 FOR EACH ROW
265 BEGIN
267 SELECT S_ID_Solicitud.NEXTVAL INTO:NEW.ID_Solicitud FROM DUAL;
269 END Crea_Nuevo_Solicitud;
271
273 /
275
277 CREATE OR REPLACE TRIGGER crea_ID_Factura
279 BEFORE INSERT ON FACTURA
281 FOR EACH ROW
283 BEGIN
285 SELECT S_ID_Factura.NEXTVAL INTO:NEW.ID_Factura FROM DUAL;
287 END crea_ID_Factura;
289
291 /
293
295 CREATE OR REPLACE TRIGGER crea_ID_Emplazamiento
297 BEFORE INSERT ON EMPLAZAMIENTO
299 FOR EACH ROW
301 BEGIN
303 SELECT S_ID_Emplazamiento.NEXTVAL
305 INTO :NEW.ID_Emplazamiento FROM DUAL;
307 END crea_ID_Emplazamiento;
309
311 /

```

```

241 CREATE OR REPLACE TRIGGER crea_ID_Venta
    BEFORE INSERT ON Venta
    FOR EACH ROW
243 BEGIN
    SELECT S_ID_Venta.NEXTVAL
245 INTO :NEW.ID_Venta FROM DUAL;
END crea_ID_Venta;
247 /

```

sql/tablas.sql

i

9.2. Funciones y Procedures

```

1 CREATE OR REPLACE PROCEDURE Socio_Nuevo (
    p_Nombre IN SOCIO.Nombre%TYPE,
3    p_Apellidos IN SOCIO.Apellidos%TYPE,
    p_Direccion IN SOCIO.Direccion%TYPE,
5    p_FechaDeNacimiento IN SOCIO.FechaDeNacimiento%TYPE,
    p_Email IN SOCIO.Email%TYPE,
7    p_DNI IN SOCIO.DNI%TYPE)
IS BEGIN
9    INSERT INTO SOCIO
    VALUES (p_Nombre, p_Apellidos, p_Direccion, p_FechaDeNacimiento,
11         p_Email,p_DNI);
END Socio_Nuevo;
13
15 /
16 CREATE OR REPLACE PROCEDURE Producto_Nuevo(
17     P_ID_Producto IN PRODUCTO.ID_Producto%TYPE,
    P_Nombre IN PRODUCTO.Nombre%TYPE,
19     P_Descripcion IN PRODUCTO.Descripcion%TYPE,
    P_Categoria IN PRODUCTO.Categoria%TYPE,
21     P_PrecioProducto IN PRODUCTO.PrecioProducto%TYPE,
    P_IVA IN PRODUCTO.IVA%TYPE)
23 IS BEGIN
    INSERT INTO PRODUCTO
25     VALUES(P_ID_Producto,P_Nombre,P_Descripcion,P_Categoria,
        P_PrecioProducto, P_IVA);
27 END Producto_Nuevo;
29 /
30
31 CREATE OR REPLACE PROCEDURE Stock_Nuevo (
    P_ID_Emplazamiento IN STOCK.ID_Emplazamiento%TYPE,
33     P_ID_Producto IN STOCK.ID_Producto%TYPE,
    P_Cantidad IN STOCK.Cantidad%TYPE)
35 IS BEGIN
    INSERT INTO STOCK
37     VALUES (P_ID_Emplazamiento, P_ID_Producto, P_Cantidad);
END Stock_Nuevo;
39
41 /
42
43 CREATE OR REPLACE PROCEDURE PRODUCTO_TRASPASO_Nuevo (
    p_ID_Traspaso IN ASOCIACION_PRODUCTO_TRASPASO.ID_Traspaso%TYPE,
45     p_ID_Producto IN ASOCIACION_PRODUCTO_TRASPASO.ID_Producto%TYPE,
    p_Cantidad IN ASOCIACION_PRODUCTO_TRASPASO.Cantidad%TYPE)
47 IS BEGIN
    INSERT INTO ASOCIACION_PRODUCTO_TRASPASO
49     VALUES(p_ID_Traspaso,p_ID_Producto,p_Cantidad);
END PRODUCTO_TRASPASO_Nuevo;
51 /
52
53 CREATE OR REPLACE PROCEDURE PRODUCTO_SOLICITUD_Nuevo (
    p_ID_Solicitud IN ASOCIACION_PRODUCTO_SOLICITUD.ID_Solicitud%TYPE,
55     p_ID_Producto IN ASOCIACION_PRODUCTO_SOLICITUD.ID_Producto%TYPE,
    p_Cantidad IN ASOCIACION_PRODUCTO_SOLICITUD.Cantidad%TYPE)

```



```

57      IS BEGIN
58          INSERT INTO ASOCIACION_PRODUCTO_SOLICITUD
59          VALUES (p_ID_Solicitud, p_ID_Producto, p_Cantidad);
60      END PRODUCTO_SOLICITUD_Nuevo;
61
62  /
63
64  CREATE OR REPLACE PROCEDURE TRASPASO_Nuevo (
65      p_ID_Traspaso IN TRASPASO.ID_Traspaso %TYPE,
66      p_FechaTraspaso IN TRASPASO.FechaTraspaso %TYPE,
67      p_ID_EmplazamientoSalida IN TRASPASO.ID_EmplazamientoSalida %TYPE,
68      p_ID_EmplazamientoEntrada IN TRASPASO.ID_EmplazamientoEntrada %TYPE)
69  IS BEGIN
70      INSERT INTO TRASPASO
71      VALUES (p_ID_Traspaso, p_FechaTraspaso, p_ID_EmplazamientoSalida,
72              p_ID_EmplazamientoEntrada);
73  END TRASPASO_Nuevo;
74
75  /
76
77  CREATE OR REPLACE PROCEDURE SOLICITUD_Nuevo (
78      p_ID_Solicitud IN SOLICITUD_TRASPASO.ID_Solicitud %TYPE,
79      p_FechaSolicitud IN SOLICITUD_TRASPASO.FechaSolicitud %TYPE,
80      p_ID_EmplazamientoSalida IN SOLICITUD_TRASPASO.ID_EmplazamientoSalida %TYPE,
81      p_ID_EmplazamientoEntrada IN SOLICITUD_TRASPASO.ID_EmplazamientoEntrada %TYPE)
82  IS BEGIN
83      INSERT INTO SOLICITUD_TRASPASO
84      VALUES (p_ID_Solicitud, p_FechaSolicitud, p_ID_EmplazamientoSalida,
85              p_ID_EmplazamientoEntrada);
86  END SOLICITUD_Nuevo;
87
88  /
89
90
91  CREATE OR REPLACE PROCEDURE Emplazamiento_Nuevo(
92      P_ID_Emplazamiento IN EMPLAZAMIENTO.ID_Emplazamiento %TYPE,
93      P_Direccion IN EMPLAZAMIENTO.Direccion %TYPE,
94      P_Telefono IN EMPLAZAMIENTO.Telefono %TYPE,
95      P_Tipo IN EMPLAZAMIENTO.Tipo %TYPE)
96  IS BEGIN
97      INSERT INTO EMPLAZAMIENTO
98      VALUES (P_ID_Emplazamiento, P_Direccion, P_Telefono, P_Tipo);
99  END Emplazamiento_Nuevo;
100
101  /
102
103  CREATE OR REPLACE PROCEDURE Factura_Nueva(
104      p_ID IN FACTURA.ID_FACTURA %TYPE,
105      p_FechaDeExpedicion IN FACTURA.FechaDeExpedicion %TYPE,
106      p_Devuelto IN FACTURA.Devuelto %TYPE,
107      p_ID_Venta IN FACTURA.ID_Venta %TYPE,
108      p_ID_Emplazamiento IN FACTURA.ID_Emplazamiento %TYPE
109  )
110  IS BEGIN
111      INSERT INTO FACTURA VALUES(
112          p_ID, p_FechaDeExpedicion, p_Devuelto, p_ID_Venta,
113          p_ID_Emplazamiento);
114  END Factura_Nueva;
115
116  /
117
118  CREATE OR REPLACE PROCEDURE PROVEEDOR_Nuevo (
119      p_CIF IN PROVEEDOR.CIF %TYPE,
120      p_Nombre IN PROVEEDOR.Nombre %TYPE,
121      p_Telefono IN PROVEEDOR.Telefono %TYPE,
122      p_Email IN PROVEEDOR.Email %TYPE
123  ) IS BEGIN
124      INSERT INTO PROVEEDOR
125      VALUES (p_CIF, p_Nombre, p_Telefono, p_Email);
126  END PROVEEDOR_Nuevo;
127
128  /
129
130  CREATE OR REPLACE PROCEDURE ALBARAN_Nuevo (
131      p_ID_Albaran IN ALBARAN.ID_Albaran %TYPE,
132      p_FechaFirma IN ALBARAN.FechaFirma %TYPE,

```

```

135     p_ID_Pedido IN ALBARAN.ID_PEDIDO %TYPE
136   ) IS BEGIN
137     INSERT INTO ALBARAN
138     VALUES ( p_ID_Albaran, p_FechaFirma, p_ID_Pedido);
139   END ALBARAN_Nuevo;
140
141 /
142
143 CREATE OR REPLACE PROCEDURE PEDIDO_Nuevo (
144   p_ID_Pedido IN PEDIDO.ID_Pedido %TYPE,
145   p_FechaPedido IN PEDIDO.FechaPedido %TYPE,
146   p_ID_Emplazamiento IN PEDIDO.ID_Emplazamiento %TYPE,
147   p_CIF IN PEDIDO.CIF %TYPE
148 ) IS BEGIN
149   INSERT INTO PEDIDO
150   VALUES ( p_ID_Pedido, p_FechaPedido,
151             p_ID_Emplazamiento, p_CIF);
152 END PEDIDO_Nuevo;
153
154 /
155
156 CREATE OR REPLACE PROCEDURE PEDIDO_PRODUCTO_Nuevo (
157   p_ID_Producto IN ASOCIACION_PEDIDO_PRODUCTO.ID_Producto %TYPE,
158   p_ID_Pedido IN ASOCIACION_PEDIDO_PRODUCTO.ID_Pedido %TYPE,
159   p_Cantidad IN ASOCIACION_PEDIDO_PRODUCTO.Cantidad %TYPE,
160   p_PrecioCompra IN ASOCIACION_PEDIDO_PRODUCTO.PrecioCompra %TYPE,
161   p_IVA IN ASOCIACION_PEDIDO_PRODUCTO.IVA %TYPE) IS
162 BEGIN
163   INSERT INTO ASOCIACION_PEDIDO_PRODUCTO
164   VALUES (p_ID_Producto, p_ID_Pedido, p_Cantidad, p_PrecioCompra,
165           p_IVA);
166 END PEDIDO_PRODUCTO_Nuevo;
167
168 /
169
170
171 CREATE OR REPLACE PROCEDURE Venta_Nueva(
172   P_ID_Venta IN VENTA.ID_Venta %TYPE,
173   P_FechaVenta IN VENTA.FechaVenta %TYPE,
174   P_DNI IN VENTA.DNI %TYPE)
175 IS BEGIN
176   INSERT INTO VENTA
177   VALUES(P_ID_Venta, P_FechaVenta, P_DNI);
178 END Venta_Nueva;
179
180 /
181
182
183 CREATE OR REPLACE PROCEDURE VENTA_PRODUCTO_Nueva(
184   P_ID_Venta IN ASOCIACION_VENTA_PRODUCTO.ID_Venta %TYPE,
185   P_ID_Producto IN ASOCIACION_VENTA_PRODUCTO.ID_Producto %TYPE,
186   P_Cantidad IN ASOCIACION_VENTA_PRODUCTO.Cantidad %TYPE,
187   P_PrecioVenta IN ASOCIACION_VENTA_PRODUCTO.PrecioVenta %TYPE,
188   P_IvaVenta IN ASOCIACION_VENTA_PRODUCTO.IvaVenta %TYPE
189 )
190 IS
191 BEGIN
192   INSERT INTO ASOCIACION_VENTA_PRODUCTO
193   VALUES(P_ID_Venta, P_ID_Producto, P_Cantidad, P_PrecioVenta, P_IvaVenta);
194 END VENTA_PRODUCTO_Nueva;
195
196 /
197
198
199 CREATE OR REPLACE PROCEDURE MODIFICA_PROVEEDOR_NOMBRE
200 (p_CIF IN PROVEEDOR.CIF %TYPE,
201  p_Nombre IN PROVEEDOR.Nombre %TYPE) IS
202 BEGIN
203   UPDATE PROVEEDOR SET Nombre = p_Nombre WHERE p_CIF = CIF;
204 END MODIFICA_PROVEEDOR_NOMBRE;
205
206 /
207
208
209 CREATE OR REPLACE PROCEDURE MODIFICA_PROVEEDOR_Telefono
210 (p_CIF IN PROVEEDOR.CIF %TYPE,
211  p_Telefono IN PROVEEDOR.Telefono %TYPE) IS
212 BEGIN
213   UPDATE PROVEEDOR SET Telefono = p_Telefono WHERE p_CIF = CIF;

```

```

211 END MODIFICA_PROVEEDOR_Telefono;

213 /

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

221 /

223 CREATE OR REPLACE PROCEDURE MODIFICA_PRODUCTO_DESCRIPCION (
224 p_ID_Producto IN PRODUCTO.ID_Producto %TYPE,
225 p_Nombre IN PRODUCTO.Nombre %TYPE,
226 p_Descripcion IN PRODUCTO.Descripcion %TYPE,
227 p_Categoria IN PRODUCTO.Categoria %TYPE,
228 p_PrecioProducto IN PRODUCTO.PrecioProducto %TYPE,
229 p_IVA IN PRODUCTO.IVA %TYPE)
230 IS BEGIN
231 UPDATE PRODUCTO SET Descripcion = p_Descripcion WHERE p_ID_Producto = ID_Producto;
232 END MODIFICA_PRODUCTO_DESCRIPCION;

233 /

235 /

237 CREATE OR REPLACE PROCEDURE MODIFICA_PRODUCTO_PRECIO (
238 p_ID_Producto IN PRODUCTO.ID_Producto %TYPE,
239 p_Nombre IN PRODUCTO.Nombre %TYPE,
240 p_Descripcion IN PRODUCTO.Descripcion %TYPE,
241 p_Categoria IN PRODUCTO.Categoria %TYPE,
242 p_PrecioProducto IN PRODUCTO.PrecioProducto %TYPE,
243 p_IVA IN PRODUCTO.IVA %TYPE)
244 IS BEGIN
245 UPDATE PRODUCTO SET PrecioProducto = p_PrecioProducto WHERE p_ID_Producto = ID_Producto;
246 END MODIFICA_PRODUCTO_PRECIO;

247 /

249 /

251 CREATE OR REPLACE PROCEDURE MODIFICA_PRODUCTO_IVA (
252 p_ID_Producto IN PRODUCTO.ID_Producto %TYPE,
253 p_Nombre IN PRODUCTO.Nombre %TYPE,
254 p_Descripcion IN PRODUCTO.Descripcion %TYPE,
255 p_Categoria IN PRODUCTO.Categoria %TYPE,
256 p_PrecioProducto IN PRODUCTO.PrecioProducto %TYPE,
257 p_IVA IN PRODUCTO.IVA %TYPE)
258 IS BEGIN
259 UPDATE PRODUCTO SET IVA = p_IVA
260 WHERE p_ID_Producto = ID_Producto;
261 END MODIFICA_PRODUCTO_IVA;

262 /

263 /

265 CREATE OR REPLACE PROCEDURE MODIFICA_STOCK_CANTIDAD
266 (p_ID_Emplazamiento IN STOCK.ID_Emplazamiento %TYPE,
267 p_ID_Producto IN STOCK.ID_Producto %TYPE,
268 p_Cantidad IN STOCK.Cantidad %TYPE)
269 IS BEGIN
270 UPDATE STOCK SET Cantidad = p_Cantidad
271 WHERE p_ID_Emplazamiento = ID_Emplazamiento AND p_ID_PRODUCTO = ID_PRODUCTO;
272 END MODIFICA_STOCK_CANTIDAD;

273 /

275 create or replace PROCEDURE MODIFICA_SOCIO_DIRECCION(
276 m_DNI IN SOCIO.DNI %TYPE,
277 m_DIRECCION IN SOCIO.DIRECCION %TYPE)
278 IS BEGIN
279 UPDATE SOCIO SET DIRECCION = m_DIRECCION where m_DNI = DNI;
280 COMMIT WORK;
281 END MODIFICA_SOCIO_DIRECCION;

282 /

283 /

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

```

```

289 COMMIT WORK;
END MODIFICA_SOCIO_EMAIL;

291 /

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

299 /

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

307 /

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

315 /

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

323 /

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

331 /

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

339 /

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

345 /

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

351 /

353 /*FUNCIONES*/
355 /* Esta abajo con menos parametros
357 CREATE OR REPLACE FUNCTION precio_A_Venta_producto
359     (f_ID_Venta IN ASOCIACION_VENTA_PRODUCTO.ID_Venta%TYPE,
361     f_Cantidad IN ASOCIACION_VENTA_PRODUCTO.Cantidad%TYPE,
363     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;

```

```

363 RETURN(f_PrecioLinea);
END precio_A_Venta_producto;

365 /
366 */
367 /*
368
369 CREATE OR REPLACE FUNCTION precio_Venta
(f_ID_Venta IN VENTA.ID_Venta%TYPE)
371 RETURN NUMBER is f_PrecioTotal VENTA.PRECIO_TOTAL%TYPE;
BEGIN
373 select SUM(precioLinea) into f_PrecioTotal from ASOCIACION_VENTA_PRODUCTO
where ID_Venta = f_ID_Venta;
375 RETURN(f_PrecioTotal);
END precio_Venta;

377 /
378 */
379 /*
/* Esta abajo con menos parametros
381 CREATE OR REPLACE FUNCTION precio_A_Pedido_producto
(f_ID_Pedido IN ASOCIACION_PEDIDO_PRODUCTO.ID_Pedido%TYPE,
383 f_Cantidad IN ASOCIACION_PEDIDO_PRODUCTO.Cantidad%TYPE,
f_PrecioCompra IN ASOCIACION_PEDIDO_PRODUCTO.PrecioCompra%TYPE)
385 RETURN NUMBER is f_PrecioLinea ASOCIACION_PEDIDO_PRODUCTO.PrecioLinea%TYPE;
BEGIN
387 f_PrecioLinea := f_PrecioCompra * f_Cantidad;
RETURN(f_PrecioLinea);
389 END precio_A_Pedido_producto;

391 /
392 */
393 /*
CREATE OR REPLACE FUNCTION precio_Pedido
(f_ID_Pedido IN PEDIDO.ID_Pedido%TYPE)
395 RETURN NUMBER is f_PrecioTotal PEDIDO.PRECIO_TOTAL%TYPE;
BEGIN
397 select SUM(PrecioLinea) into f_PrecioTotal from ASOCIACION_PEDIDO_PRODUCTO
399 where ID_Pedido = f_ID_Pedido;
RETURN(f_PrecioTotal);
401 END precio_Pedido;

403 /
404 */
405
CREATE OR REPLACE FUNCTION precioLinea_Aso_Pedido
407 (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;
409 f_Cantidad ASOCIACION_PEDIDO_PRODUCTO.CANTIDAD%TYPE;
f_PrecioCompra ASOCIACION_PEDIDO_PRODUCTO.PRECIOCOMPRA%TYPE;
411 BEGIN
select Cantidad into f_Cantidad from ASOCIACION_PEDIDO_PRODUCTO where ID_Pedido = f_ID_PEDIDO AND
ID_PRODUCTO = f_ID_PRODUCTO;
413 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;
415 RETURN(f_PrecioLinea);
END precioLinea_Aso_Pedido;

417 /
418
419 CREATE OR REPLACE FUNCTION precioLinea_Aso_Venta
421 (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;
423 f_Cantidad ASOCIACION_VENTA_PRODUCTO.CANTIDAD%TYPE;
f_PrecioVenta ASOCIACION_VENTA_PRODUCTO.PRECIOVENTA%TYPE;
425 BEGIN
select Cantidad into f_Cantidad from ASOCIACION_VENTA_PRODUCTO where ID_VENTA = f_ID_VENTA AND
ID_PRODUCTO = f_ID_PRODUCTO;
427 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;
429 RETURN(f_PrecioLinea);
END precioLinea_Aso_Venta;

431 /
432
433

```

```

435 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;
437 precio_AUX ASOCIACION_VENTA_PRODUCTO.PRECIOVENTA %TYPE;

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

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

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

465 /

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

483 /

485 CREATE OR REPLACE FUNCTION precioTotal_Albaran
(f_ID_PEDIDO IN ASOCIACION_PEDIDO_PRODUCTO.ID_PEDIDO %TYPE)
487 RETURN NUMBER is f_precioTotal number(8,2);
precio_AUX ASOCIACION_PEDIDO_PRODUCTO.PRECIOCOMPRA %TYPE;
489 BEGIN
select precioTotal_Venta(f_ID_VENTA) into precio_AUX from dual;
491 f_precioTotal := precio_AUX;
RETURN(f_PrecioTotal);
493 END precioTotal_Factura;

495 /

497 CREATE OR REPLACE FUNCTION precioTotal_PEDIDO
(f_ID_PEDIDO IN ASOCIACION_PEDIDO_PRODUCTO.ID_PEDIDO %TYPE)
499 RETURN NUMBER is f_precioTotal ASOCIACION_PEDIDO_PRODUCTO.PRECIOCOMPRA %TYPE;
precio_AUX ASOCIACION_PEDIDO_PRODUCTO.PRECIOCOMPRA %TYPE;
501 CURSOR all_prods
IS
503 SELECT id_pedido,id_producto,cantidad
FROM ASOCIACION_PEDIDO_PRODUCTO
505 ORDER BY ID_producto;

507 m_id_pedido ASOCIACION_PEDIDO_PRODUCTO.id_pedido %TYPE;
509 m_id_producto Producto.id_producto %type;

```

```

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

521         f_precioTotal := (precio_AUX*m_cantidad) + f_precioTotal;
        END IF;
523     END LOOP;
    CLOSE all_prods;
525 RETURN(f_PrecioTotal);
    END precioTotal_Pedido;
527
/

```

sql/funciones_y_procedures.sql

9.3. Triggers

```

/* TRIGGERS */
2
CREATE OR REPLACE TRIGGER descuento_socio
4 BEFORE INSERT ON factura
  FOR EACH ROW
6 declare v_preciototal VENTA.PRECIOTOTAL%TYPE;
  v_DNI VENTA.dni%TYPE;
8 BEGIN
    select preciototal into v_preciototal from venta where id_venta = :NEW.id_venta;
10 select dni into v_dni from venta where id_venta = :NEW.id_venta;
    IF v_DNI IS NOT NULL then
12 UPDATE venta set preciototal = v_preciototal * 0.95 where id_venta = :NEW.id_venta;
    :New.preciototal := v_preciototal * 0.95;
14 else
    update venta set preciototal = v_preciototal where id_venta = :NEW.id_venta;
16 END IF;
END;
18
/
/*
20 CREATE OR REPLACE TRIGGER stock_minimo
22 AFTER INSERT OR UPDATE ON stock
  FOR EACH ROW
24 BEGIN
    IF :NEW.cantidad <0
26 THEN
        raise_application_error(-20601, :NEW.cantidad || 'No se puede realizar esta operacion');
28 END IF;
END;
30
/
32 */
CREATE OR REPLACE TRIGGER Inicializa_nueva_Venta
34 BEFORE INSERT ON VENTA
  FOR EACH ROW
36 BEGIN
    :NEW.PrecioTotal := 0;
    :NEW.FechaVenta := SYSDATE;
38 END Comprueba_Venta;
40
/
42
CREATE OR REPLACE TRIGGER Inicializa_nuevo_Pedido
44 BEFORE INSERT ON PEDIDO
  FOR EACH ROW
46 BEGIN

```

```

:NEW.PrecioTotal := 0;
48 :NEW.FechaPedido := SYSDATE;
END Comprueba_Pedido;
50
/
52
CREATE OR REPLACE TRIGGER Inicializa_nueva_Factura
54 BEFORE INSERT ON Factura
FOR EACH ROW
56 BEGIN
:NEW.FechaDeExpedicion := SYSDATE;
58 END Inicializa_nueva_Factura;
60
/
62
CREATE OR REPLACE TRIGGER modifica_stock_venta
64 BEFORE INSERT ON FACTURA
FOR EACH ROW
66 DECLARE
e_ID_Emplazamiento Emplazamiento.ID_Emplazamiento %TYPE;
68 v_ID_Venta Venta.ID_VENTA %TYPE;
70
CURSOR all_prods
IS
72 SELECT id_venta, id_producto, cantidad
FROM ASOCIACION_VENTA_PRODUCTO
74 ORDER BY ID_producto;
76
m_id_venta ASOCIACION_VENTA_PRODUCTO.id_venta %TYPE;
m_id_producto Producto.id_producto %type;
78 m_cantidad ASOCIACION_VENTA_PRODUCTO.cantidad %TYPE;
80
BEGIN
select ID_Emplazamiento into e_ID_Emplazamiento from Emplazamiento where ID_Emplazamiento = :NEW.
ID_Emplazamiento;
82 select ID_Venta into v_ID_Venta from Venta where ID_Venta = :New.ID_Venta;
84
OPEN all_prods;
LOOP
86 Fetch all_prods INTO m_id_venta, m_id_producto, m_cantidad;
EXIT WHEN all_prods %NOTFOUND;
88 if m_id_venta = v_id_Venta
THEN
90 update stock set cantidad = cantidad - m_cantidad where id_emplazamiento = e_ID_Emplazamiento AND
id_producto = m_ID_Producto;
END IF;
92 END LOOP;
CLOSE all_prods;
94 END modifica_stock_venta;
96
/*
98 CREATE OR REPLACE TRIGGER inicializa_preciolinea_alv
BEFORE INSERT OR UPDATE ON ASOCIACION_VENTA_PRODUCTO
100 FOR EACH ROW
BEGIN
102 :NEW.preciolinea := :New.cantidad * :New.precioVenta;
END inicializa_preciolinea_alv;
104
/
106
*/
CREATE OR REPLACE TRIGGER inicializa_preciototal_venta
108 BEFORE INSERT OR UPDATE ON ASOCIACION_VENTA_PRODUCTO
for each row
begin
110 UPDATE venta set preciototal = PRECIOTOTAL + (:New.cantidad * :New.precioVenta) where id_venta = :
New.id_venta;
112 END inicializa_preciototal_venta;
114
/
116
CREATE OR REPLACE TRIGGER modifica_stock_pedido
118 BEFORE INSERT OR UPDATE ON ALBARAN
FOR EACH ROW
120 DECLARE

```



```

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

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

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

134 BEGIN

136 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;
138 OPEN all_prods;
    LOOP
140     Fetch all_prods INTO m_id_pedido, m_id_producto, m_cantidad;
        EXIT WHEN all_prods %NOTFOUND;
142     if m_id_pedido = p_id_pedido
        THEN
144         update stock set cantidad = cantidad+m_cantidad where id_emplazamiento = e_ID_Emplazamiento AND
            id_producto = m_ID_Producto;
        END IF;
146     END LOOP;
        CLOSE all_prods;
148 END modifica_stock_pedido;

150 /

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

160 BEGIN
    select ID_Emplazamientosalida into e_ID_Emplazamiento_salida from traspaso where ID_traspaso = :NEW.
        ID_traspaso;
162 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;
164 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;
166 END modifica_stock_traspaso;

168 /

170 CREATE OR REPLACE TRIGGER Inicializa_Nuevo_Pedido
    BEFORE INSERT ON Pedido
    FOR EACH ROW
172 BEGIN
    :NEW.PrecioTotal := 0;
174 :NEW.FechaPedido := SYSDATE;
    END Inicializa_Nuevo_Pedido;
176 /

178 /

180 CREATE OR REPLACE TRIGGER Inicializa_Nueva_ST
    BEFORE INSERT ON Solicitud_Traspaso
182 FOR EACH ROW
    BEGIN
184 :NEW.FechaSolicitud := SYSDATE;
    END Inicializa_Nueva_ST;
186 /

188 /

190 CREATE OR REPLACE TRIGGER Inicializa_Nuevo_Traspaso
    BEFORE INSERT ON Traspaso
    FOR EACH ROW
192 BEGIN

```

```

:NEW.FechaTraspaso := SYSDATE;
194 END Inicializa_Nuevo_Traspaso;

196 /

198 create or replace TRIGGER solicitud_stock_minimo
BEFORE INSERT ON asociacion_producto_solicitud
200 FOR EACH ROW
DECLARE
202 e_id_emplazamientoentrada SOLICITUD_TRASPASO.ID_EMPLAZAMIENTOENTRADA %TYPE;
e_cantidad Stock.cantidad %TYPE;
204
BEGIN
206 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;
208 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');
210 END IF;
END;
212
/
214 /*
create or replace TRIGGER inicializa_preciolinea_alp
216 BEFORE INSERT OR UPDATE ON ASOCIACION_PEDIDO_PRODUCTO
FOR EACH ROW
218 BEGIN
:NEW.preciolinea := :New.cantidad * :New.preciocompra;
220 END inicializa_preciolinea_alp;
/
222 */
create or replace TRIGGER inicializa_preciototal_albaran
224 BEFORE INSERT OR UPDATE ON Albaran
for each row
226 DECLARE
p_preciototal pedido.preciototal %TYPE;
228 begin
select preciototal into p_preciototal from pedido where id_pedido = :New.id_pedido;
230 :New.preciototal := p_preciototal;
END inicializa_preciototal_albaran;
232
/
234
create or replace TRIGGER inicializa_preciototal_pedido
236 BEFORE INSERT OR UPDATE ON ASOCIACION_pedido_producto
for each row
238 begin
UPDATE pedido set preciototal = PRECIOTOTAL + (:New.cantidad * :New.preciocompra) where id_pedido =
:New.id_pedido;
240 END inicializa_preciototal_pedido;
242
/
244
create or replace trigger inicializa_precioventa_apv
before insert on asociacion_venta_producto
246 for each row
declare
248 pPrecio producto.precioproducto %TYPE;
begin
select precioproducto into p_precio from producto where id_producto = :New.id_producto;
:NEW.precioventa := p_precio;
252 END inicializa_precioventa_apv;
254
/
256
create or replace trigger inicializa_IVA_apv
before insert on asociacion_venta_producto
258 for each row
declare
260 p_IVA PRODUCTO.IVA %TYPE;
begin
262 select Iva into p_IVA from producto where id_producto = :New.id_producto;
:NEW.IVAVENTA := p_IVA;
264 END inicializa_IVA_apv;

```

```

266 /
268 create or replace trigger inicializa_IVA_APP
    before insert on asociacion_pedido_producto
270 for each row
    declare
272 p_IVA PRODUCTO.IVA%TYPE;
    begin
274 select IVA into p_IVA from producto where id_producto = :New.id_producto;
        :NEW.IVA := p_IVA;
276 END inicializa_IVA_APP;

278 /

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

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

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

306 num_lines := 3;

308 dbms_output.get_lines(lines, num_lines);

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

```

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

36 /

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

50 /

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

64 /

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

74 /

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

84 /

86 CREATE OR REPLACE PACKAGE PRUEBAS_PRODUCTO AS
87     PROCEDURE inicializar;
88     PROCEDURE insertar
89         (nombre_prueba VARCHAR2, i_nombre VARCHAR2, i_descripcion VARCHAR2,
90          i_categoria VARCHAR2, i_precioProducto INT, i_iva INT, salidaEsperada BOOLEAN);

```

```

92     PROCEDURE actualizar
        (nombre_prueba VARCHAR2, a_id_producto INT, a_descripcion VARCHAR2,
          a_precioProducto INT, a_iva INT, salidaEsperada BOOLEAN);
94     PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_producto INT, salidaEsperada BOOLEAN);
96 END PRUEBAS_PRODUCTO;

98 /

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

110 /

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

122 /

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

134 /

136 /

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

146 /

148 /

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

158 /

160 /

162 CREATE OR REPLACE PACKAGE PRUEBAS_STOCK AS
    PROCEDURE inicializar;
    PROCEDURE insertar
164         (nombre_prueba VARCHAR2, i_id_emplazamiento INT, i_id_producto INT,
          i_cantidad INT, salidaEsperada BOOLEAN);
166     PROCEDURE actualizar

```

```

168         (nombre_prueba VARCHAR2, a_id_emplazamiento INT, a_id_producto INT,
169          a_cantidad INT, salidaEsperada BOOLEAN);
170     PROCEDURE eliminar
171         (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,
172          salidaEsperada BOOLEAN);
173     PROCEDURE stock_correcto
174         (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT, stock_previo int, cantidad
175          INT, salidaEsperada BOOLEAN);
176     PROCEDURE stock_correcto_p
177         (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT, stock_previo int, cantidad
178          INT, salidaEsperada BOOLEAN);
179     PROCEDURE stock_correcto_t
180         (nombre_prueba VARCHAR2, e_id_emplazamiento envia int, e_id_emplazamiento_recibe int,
181          stock_previo envia int, stock_previo_recibe int,
182          cantidad int, e_id_producto int, salidaEsperada BOOLEAN);
183     END PRUEBA_STOCK;
184 /
185
186 CREATE OR REPLACE PACKAGE PRUEBAS_A_PRODUCTO_SOLICITUD AS
187     PROCEDURE inicializar;
188     PROCEDURE insertar
189         (nombre_prueba VARCHAR2, i_id_producto INT, i_id_Solicitud INT,
190          i_cantidad NUMBER, salidaEsperada BOOLEAN);
191     PROCEDURE eliminar
192         (nombre_prueba VARCHAR2, e_id_Solicitud INT, e_id_producto INT, salidaEsperada BOOLEAN);
193     END PRUEBAS_A_PRODUCTO_SOLICITUD;
194 /
195
196 CREATE OR REPLACE PACKAGE PRUEBAS_A_PRODUCTO_TRASPASO AS
197     PROCEDURE inicializar;
198     PROCEDURE insertar
199         (nombre_prueba VARCHAR2, i_id_producto INT, i_id_Traspaso INT,
200          i_cantidad NUMBER, salidaEsperada BOOLEAN);
201     PROCEDURE eliminar
202         (nombre_prueba VARCHAR2, e_id_Traspaso INT, e_id_producto INT, salidaEsperada BOOLEAN);
203     END PRUEBAS_A_PRODUCTO_TRASPASO;
204 /
205
206 CREATE OR REPLACE PACKAGE PRUEBAS_A_PEDIDO_PRODUCTO AS
207     PROCEDURE inicializar;
208     PROCEDURE insertar
209         (nombre_prueba VARCHAR2, i_id_pedido INT, i_id_producto INT,
210          i_cantidad NUMBER, i_precioCompra number, i_iva INT,
211          salidaEsperada BOOLEAN);
212     PROCEDURE eliminar
213         (nombre_prueba VARCHAR2, e_id_pedido INT, e_id_producto INT,
214          salidaEsperada BOOLEAN);
215     END PRUEBAS_A_PEDIDO_PRODUCTO;
216 /
217
218 CREATE OR REPLACE PACKAGE PRUEBAS_FUNCIONES AS
219     PROCEDURE precioLinea_A_Pedido
220         (nombre_prueba VARCHAR2, ID_PRODUCTO NUMBER, ID_PEDIDO NUMBER, esperado number,
221          salidaEsperada BOOLEAN);
222     PROCEDURE precioLinea_A_Venta
223         (nombre_prueba VARCHAR2, ID_PRODUCTO NUMBER, ID_VENTA NUMBER, esperado number, salidaEsperada
224          BOOLEAN);
225     PROCEDURE precio_Venta
226         (nombre_prueba VARCHAR2, ID_Venta NUMBER, esperado number, salidaEsperada BOOLEAN);
227     PROCEDURE precio_Factura
228         (nombre_prueba VARCHAR2, ID_Venta NUMBER, esperado number, salidaEsperada BOOLEAN);
229     PROCEDURE precio_Pedido
230         (nombre_prueba VARCHAR2, ID_PEDIDO NUMBER, esperado number, salidaEsperada BOOLEAN);
231     PROCEDURE precio_Albaran
232         (nombre_prueba VARCHAR2, ID_PEDIDO NUMBER, esperado number, salidaEsperada BOOLEAN);
233     END PRUEBAS_FUNCIONES;
234 /
235
236 /* CUERPOS DE PRUEBAS */
237
238 CREATE OR REPLACE PACKAGE BODY PRUEBAS_SOCIO AS
239     PROCEDURE inicializar AS

```

```

240 BEGIN
241     DELETE FROM SOCIO;
242 END inicializar;

243
244 PROCEDURE insertar
245     (nombre_prueba VARCHAR2, i_nombre VARCHAR2, i_apellidos VARCHAR2,
246      i_DNI VARCHAR2, i_direccion VARCHAR2, i_nacimiento DATE, i_email VARCHAR2
247      ,salidaEsperada BOOLEAN) AS
248     salida BOOLEAN := true;
249     actual socio %ROWTYPE;
250 BEGIN
251     INSERT INTO socio VALUES (i_nombre, i_apellidos, i_direccion, i_nacimiento,
252                                i_email, i_DNI);
253
254     SELECT * INTO actual FROM SOCIO WHERE DNI = i_DNI;
255
256     IF (actual.nombre<>i_nombre) OR (actual.apellidos<>i_apellidos) OR (actual.DNI<>i_DNI) OR (
257         actual.direccion<>i_direccion) OR (actual.fechadenacimiento<>i_nacimiento) OR (actual.email<>
258         i_email) THEN
259         salida := false;
260     END IF;
261
262     PRINTR(nombre_prueba, salida, salidaEsperada);
263
264     EXCEPTION
265     WHEN OTHERS THEN
266         PRINTR(nombre_prueba, false, salidaEsperada);
267         ROLLBACK;
268 END insertar;

269
270 PROCEDURE actualizar
271     (nombre_prueba VARCHAR2, a_DNI VARCHAR2, a_direccion VARCHAR2,
272      a_email VARCHAR2, salidaEsperada BOOLEAN) AS
273     salida BOOLEAN := true;
274     actual socio %ROWTYPE;
275 BEGIN
276     UPDATE SOCIO SET DIRECCION = a_direccion, EMAIL = a_email where DNI = a_DNI;
277
278     SELECT * INTO actual FROM SOCIO WHERE DNI = a_DNI;
279
280     IF (actual.direccion<>a_direccion) OR (actual.email<>a_email) THEN
281         salida := false;
282     END IF;
283
284     PRINTR(nombre_prueba, salida, salidaEsperada);
285
286     EXCEPTION
287     WHEN OTHERS THEN
288         PRINTR(nombre_prueba, false, salidaEsperada);
289         ROLLBACK;
290 END actualizar;

291
292 PROCEDURE eliminar
293     (nombre_prueba VARCHAR2, e_DNI VARCHAR2, salidaEsperada BOOLEAN) AS
294     salida BOOLEAN := true;
295     cantidad INT;
296 BEGIN
297     DELETE FROM SOCIO WHERE DNI = e_DNI;
298
299     SELECT COUNT(*) INTO cantidad FROM SOCIO WHERE DNI = e_DNI;
300
301     IF (cantidad<>0) THEN
302         salida := false;
303     END IF;
304
305     PRINTR(nombre_prueba, salida, salidaEsperada);
306
307     EXCEPTION
308     WHEN OTHERS THEN
309         PRINTR(nombre_prueba, false, salidaEsperada);
310         ROLLBACK;
311 END eliminar;
312
313 END PRUEBAS_SOCIO;
314
315 /
316
317 CREATE OR REPLACE PACKAGE BODY PRUEBAS_PROVEEDOR AS

```

```

314 PROCEDURE inicializar AS
316 BEGIN
318     DELETE FROM PROVEEDOR;
318 END inicializar;

320 PROCEDURE insertar
321     (nombre_prueba VARCHAR2, i_CIF VARCHAR2, i_nombre VARCHAR2,
322      i_telefono INT, i_email VARCHAR2, salidaEsperada BOOLEAN) AS
323     salida BOOLEAN := true;
324     actual proveedor %ROWTYPE;
325 BEGIN
326     INSERT INTO PROVEEDOR VALUES (i_CIF, i_nombre, i_telefono, i_email);
327
328     SELECT * INTO actual FROM PROVEEDOR WHERE CIF = i_CIF;
329
330     IF (actual.nombre <> i_nombre) OR (actual.telefono <> i_telefono) OR (actual.CIF <> i_CIF) OR (
331         actual.email <> i_email) THEN
332         salida := false;
333     END IF;
334
335     PRINTR(nombre_prueba, salida, salidaEsperada);
336
337     EXCEPTION
338     WHEN OTHERS THEN
339         PRINTR(nombre_prueba, false, salidaEsperada);
340         ROLLBACK;
341 END insertar;

342 PROCEDURE actualizar
343     (nombre_prueba VARCHAR2, a_cif VARCHAR2, a_nombre VARCHAR2,
344      a_telefono INT, a_email VARCHAR2, salidaEsperada BOOLEAN) AS
345     salida BOOLEAN := true;
346     actual proveedor %ROWTYPE;
347 BEGIN
348     UPDATE PROVEEDOR SET NOMBRE = a_nombre, TELEFONO = a_telefono, EMAIL = a_email where CIF
349     = a_cif;
350
351     SELECT * INTO actual FROM PROVEEDOR WHERE CIF = a_cif;
352
353     IF (actual.nombre <> a_nombre) OR
354         (actual.telefono <> a_telefono) OR
355         (actual.email <> a_email) THEN
356         salida := false;
357     END IF;
358
359     PRINTR(nombre_prueba, salida, salidaEsperada);
360
361     EXCEPTION
362     WHEN OTHERS THEN
363         PRINTR(nombre_prueba, false, salidaEsperada);
364         ROLLBACK;
365 END actualizar;

366 PROCEDURE eliminar
367     (nombre_prueba VARCHAR2, e_CIF VARCHAR2, salidaEsperada BOOLEAN) AS
368     salida BOOLEAN := true;
369     cantidad INT;
370 BEGIN
371     DELETE FROM PROVEEDOR WHERE CIF = e_CIF;
372
373     SELECT COUNT(*) INTO cantidad FROM PROVEEDOR WHERE CIF = e_cif;
374
375     IF (cantidad <> 0) THEN
376         salida := false;
377     END IF;
378
379     PRINTR(nombre_prueba, salida, salidaEsperada);
380
381     EXCEPTION
382     WHEN OTHERS THEN
383         PRINTR(nombre_prueba, false, salidaEsperada);
384         ROLLBACK;
385 END eliminar;
386 END PRUEBAS_PROVEEDOR;
387 /

```



```

390 CREATE OR REPLACE PACKAGE BODY PRUEBAS_EMPLAZAMIENTO AS
392     PROCEDURE inicializar AS
394     BEGIN
395         DELETE FROM EMPLAZAMIENTO;
396     END inicializar;
398     PROCEDURE insertar
399     (nombre_prueba VARCHAR2, i_direccion VARCHAR2, i_telefono INT,
400      i_tipo VARCHAR2, salidaEsperada BOOLEAN) AS
401     salida BOOLEAN := true;
402     actual emplazamiento%ROWTYPE;
403     w_ID_Emplazamiento INT;
404     BEGIN
405         INSERT INTO emplazamiento VALUES(null,i_direccion, i_telefono, i_tipo);
406
407         w_ID_Emplazamiento := S_ID_Emplazamiento.currval;
408         SELECT * INTO actual FROM EMPLAZAMIENTO WHERE ID_EMPLAZAMIENTO = w_ID_Emplazamiento;
409
410         IF (actual.direccion<>i_direccion) OR (actual.telefono<>i_telefono) OR (actual.tipo<>i_tipo)
411         THEN
412             salida := false;
413         END IF;
414
415         PRINTR(nombre_prueba, salida, salidaEsperada);
416
417         EXCEPTION
418         WHEN OTHERS THEN
419             PRINTR(nombre_prueba, false, salidaEsperada);
420             ROLLBACK;
421     END insertar;
422
423     PROCEDURE actualizar
424     (nombre_prueba VARCHAR2, a_id_emplazamiento INT, a_direccion VARCHAR2,
425      a_telefono INT, salidaEsperada BOOLEAN)AS
426     salida BOOLEAN := true;
427     actual emplazamiento%ROWTYPE;
428     BEGIN
429         UPDATE emplazamiento SET DIRECCION = a_direccion, TELEFONO = a_telefono where
430         ID_EMPLAZAMIENTO = a_id_emplazamiento;
431
432         SELECT * INTO actual FROM EMPLAZAMIENTO WHERE ID_EMPLAZAMIENTO = a_id_emplazamiento;
433
434         IF (actual.direccion<>a_direccion) OR
435            (actual.telefono<>a_telefono) THEN
436             salida := false;
437         END IF;
438
439         PRINTR(nombre_prueba, salida, salidaEsperada);
440
441         EXCEPTION
442         WHEN OTHERS THEN
443             PRINTR(nombre_prueba, false, salidaEsperada);
444             ROLLBACK;
445     END actualizar;
446
447     PROCEDURE eliminar
448     (nombre_prueba VARCHAR2, e_id_emplazamiento INT, salidaEsperada BOOLEAN) AS
449     salida BOOLEAN := true;
450     cantidad INT;
451     BEGIN
452         DELETE FROM EMPLAZAMIENTO WHERE ID_EMPLAZAMIENTO = e_id_emplazamiento;
453
454         SELECT COUNT(*) INTO cantidad FROM EMPLAZAMIENTO WHERE ID_EMPLAZAMIENTO =
455         e_id_emplazamiento;
456
457         IF (cantidad<>0) THEN
458             salida := false;
459         END IF;
460
461         PRINTR(nombre_prueba, salida, salidaEsperada);
462
463         EXCEPTION
464         WHEN OTHERS THEN
465             PRINTR(nombre_prueba, false, salidaEsperada);

```

```

464         ROLLBACK;
465     END eliminar;
466 END PRUEBAS_EMPLAZAMIENTO;
467
468 /
469 CREATE OR REPLACE PACKAGE BODY PRUEBAS_PRODUCTO AS
470
471     PROCEDURE inicializar AS
472     BEGIN
473         DELETE FROM PRODUCTO;
474     END inicializar;
475
476     PROCEDURE insertar
477     (nombre_prueba VARCHAR2, i_nombre VARCHAR2, i_descripcion VARCHAR2,
478      i_categoria VARCHAR2, i_precioProducto INT, i_iva INT, salidaEsperada BOOLEAN) AS
479     salida BOOLEAN := true;
480     actual_producto %ROWTYPE;
481     w_ID_Producto INT;
482     BEGIN
483         INSERT INTO producto VALUES (null, i_nombre, i_descripcion, i_categoria, i_precioProducto,
484                                       i_iva);
485
486         w_ID_Producto := S_ID_Producto.currval;
487         SELECT * INTO actual FROM PRODUCTO WHERE ID_Producto = w_ID_Producto;
488
489         IF (actual.nombre <> i_nombre) OR (actual.descripcion <> i_descripcion) OR (actual.categoria <>
490            i_categoria) OR (actual.precioProducto <> i_precioProducto) OR (actual.iva <> i_iva) THEN
491             salida := false;
492         END IF;
493
494         PRINTR(nombre_prueba, salida, salidaEsperada);
495
496     EXCEPTION
497     WHEN OTHERS THEN
498         PRINTR(nombre_prueba, false, salidaEsperada);
499         ROLLBACK;
500     END insertar;
501
502     PROCEDURE actualizar
503     (nombre_prueba VARCHAR2, a_id_producto INT, a_descripcion VARCHAR2,
504      a_precioProducto INT, a_iva INT, salidaEsperada BOOLEAN) AS
505     salida BOOLEAN := true;
506     actual_producto %ROWTYPE;
507     BEGIN
508         UPDATE producto SET DESCRIPCION = a_descripcion, PRECIOPRODUCTO = a_precioProducto, IVA =
509            a_iva
510            where ID_Producto = a_id_producto;
511
512         SELECT * INTO actual FROM PRODUCTO WHERE ID_Producto = a_id_producto;
513         IF (actual.descripcion <> a_descripcion) OR
514            (actual.precioProducto <> a_precioProducto) OR
515            (actual.iva <> a_iva) THEN
516             salida := false;
517         END IF;
518
519         PRINTR(nombre_prueba, salida, salidaEsperada);
520
521     EXCEPTION
522     WHEN OTHERS THEN
523         PRINTR(nombre_prueba, false, salidaEsperada);
524         ROLLBACK;
525     END actualizar;
526
527     PROCEDURE eliminar
528     (nombre_prueba VARCHAR2, e_id_producto INT, salidaEsperada BOOLEAN) AS
529     salida BOOLEAN := true;
530     cantidad INT;
531     BEGIN
532         DELETE FROM PRODUCTO WHERE ID_Producto = e_id_producto;
533
534         SELECT COUNT(*) INTO cantidad FROM PRODUCTO WHERE ID_Producto = e_id_producto;
535         IF (cantidad <> 0) THEN
536             salida := false;
537         END IF;
538
539         PRINTR(nombre_prueba, salida, salidaEsperada);

```

```

538         EXCEPTION
539         WHEN OTHERS THEN
540             PRINTR(nombre_prueba, false, salidaEsperada);
541             ROLLBACK;
542     END eliminar;
543 END PRUEBAS_PRODUCTO;
544
545 /
546
547 CREATE OR REPLACE PACKAGE BODY PRUEBAS_FACTURA AS
548
549     PROCEDURE inicializar AS
550     BEGIN
551         DELETE FROM FACTURA;
552     END inicializar;
553
554     PROCEDURE insertar
555     (nombre_prueba VARCHAR2, i_fechaDeExpedicion DATE, i_devuelto VARCHAR2,
556      i_id_venta INT, i_id_emplazamiento INT, salidaEsperada BOOLEAN) AS
557     salida BOOLEAN := true;
558     actual factura%ROWTYPE;
559     w_ID_factura INT;
560     BEGIN
561         INSERT INTO factura VALUES(null, i_fechaDeExpedicion, i_devuelto, i_id_venta,
562                                     i_id_emplazamiento);
563
564         w_ID_factura := S_ID_Factura.currval;
565         SELECT * INTO actual FROM factura WHERE ID_factura = w_ID_factura;
566
567         IF (actual.fechaDeExpedicion<>i_fechaDeExpedicion) OR (actual.devuelto<>i_devuelto) OR (
568             actual.id_venta<>i_id_venta) OR (actual.id_emplazamiento<>i_id_emplazamiento)THEN
569             salida := false;
570         END IF;
571
572         PRINTR(nombre_prueba, salida, salidaEsperada);
573
574     EXCEPTION
575     WHEN OTHERS THEN
576         PRINTR(nombre_prueba, false, salidaEsperada);
577         ROLLBACK;
578     END insertar;
579
580     PROCEDURE actualizar
581     (nombre_prueba VARCHAR2, a_id_factura INT, a_devuelto VARCHAR2,
582      salidaEsperada BOOLEAN)AS
583     salida BOOLEAN := true;
584     actual factura%ROWTYPE;
585     BEGIN
586         UPDATE factura SET DEVUELTO = a_devuelto where ID_factura = a_id_factura;
587
588         SELECT * INTO actual FROM factura WHERE ID_factura = a_id_factura;
589         IF (actual.devuelto<>a_devuelto) THEN
590             salida := false;
591         END IF;
592
593         PRINTR(nombre_prueba, salida, salidaEsperada);
594
595     EXCEPTION
596     WHEN OTHERS THEN
597         PRINTR(nombre_prueba, false, salidaEsperada);
598         ROLLBACK;
599     END actualizar;
600
601     PROCEDURE eliminar
602     (nombre_prueba VARCHAR2, e_id_factura INT, salidaEsperada BOOLEAN) AS
603     salida BOOLEAN := true;
604     cantidad INT;
605     BEGIN
606         DELETE FROM factura WHERE ID_factura = e_id_factura;
607
608         SELECT COUNT(*) INTO cantidad FROM factura WHERE ID_factura = e_id_factura;
609         IF (cantidad<>0) THEN
610             salida := false;
611         END IF;
612
613         PRINTR(nombre_prueba, salida, salidaEsperada);

```

```

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

620  /

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

628      PROCEDURE insertar
629          (nombre_prueba VARCHAR2, i_id_emplazamiento INT, i_id_producto INT,
630           i_cantidad INT, salidaEsperada BOOLEAN) AS
631          salida BOOLEAN := true;
632          actual stock%ROWTYPE;
633      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 =
637              i_id_producto;

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

642          PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

656          SELECT * INTO actual FROM stock WHERE ID_Emplazamiento = a_id_emplazamiento and
657              ID_Producto = a_id_producto;
658          IF (actual.cantidad<>a_cantidad) THEN
659              salida := false;
660          END IF;

661          PRINTR(nombre_prueba, salida, salidaEsperada);

662      EXCEPTION
663      WHEN OTHERS THEN
664          PRINTR(nombre_prueba, false, salidaEsperada);
665          ROLLBACK;
666      END actualizar;

667      PROCEDURE eliminar
668          (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,
669           salidaEsperada BOOLEAN) AS
670          salida BOOLEAN := true;
671          cantidad INT;
672      BEGIN
673          DELETE FROM stock WHERE ID_Emplazamiento = e_id_emplazamiento and ID_Producto =
674              e_id_producto;

675          SELECT COUNT(*) INTO cantidad FROM stock WHERE ID_Emplazamiento = e_id_emplazamiento and
676              ID_Producto = e_id_producto;
677          IF (cantidad<>0) THEN
678              salida := false;
679          END IF;

```

```

684         PRINTR(nombre_prueba, salida, salidaEsperada);
686     EXCEPTION
688     WHEN OTHERS THEN
689         PRINTR(nombre_prueba, false, salidaEsperada);
690         ROLLBACK;
691     END eliminar;
692
693     PROCEDURE stock_correcto
694     (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT, stock_previo int, cantidad
695     INT, salidaEsperada BOOLEAN) AS
696     salida BOOLEAN := true;
697     stock_nuevo int;
698     BEGIN
699         select cantidad into stock_nuevo from stock where id_emplazamiento = e_id_emplazamiento and
700         id_producto= e_id_producto;
701         if(stock_previo-cantidad)<> stock_nuevo then
702             salida := false;
703         END IF;
704
705         PRINTR(nombre_prueba, salida, salidaEsperada);
706
707     EXCEPTION
708     WHEN OTHERS THEN
709         PRINTR(nombre_prueba, false, salidaEsperada);
710         ROLLBACK;
711
712     END stock_correcto;
713
714     PROCEDURE stock_correcto_p
715     (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT, stock_previo int, cantidad
716     INT, salidaEsperada BOOLEAN) AS
717     salida BOOLEAN := true;
718     stock_nuevo int;
719     BEGIN
720         select cantidad into stock_nuevo from stock where id_emplazamiento = e_id_emplazamiento and
721         id_producto= e_id_producto;
722         if(stock_previo+cantidad)<> stock_nuevo then
723             salida := false;
724         END IF;
725
726         PRINTR(nombre_prueba, salida, salidaEsperada);
727
728     EXCEPTION
729     WHEN OTHERS THEN
730         PRINTR(nombre_prueba, false, salidaEsperada);
731         ROLLBACK;
732
733     end stock_correcto_p;
734
735     PROCEDURE stock_correcto_t
736     (nombre_prueba VARCHAR2, e_id_emplazamiento_envia int, e_id_emplazamiento_recibe int,
737     stock_previo_envia int, stock_previo_recibe int,
738     cantidad int, e_id_producto int, salidaEsperada BOOLEAN) AS
739     salida BOOLEAN := true;
740     stock_nuevo_envia int;
741     stock_nuevo_recibe int;
742     BEGIN
743         select cantidad into stock_nuevo_envia from stock where id_emplazamiento =
744         e_id_emplazamiento_envia and id_producto = e_id_producto;
745         select cantidad into stock_nuevo_recibe from stock where id_emplazamiento =
746         e_id_emplazamiento_recibe and id_producto = e_id_producto;
747         if(stock_previo_envia-cantidad)<>stock_nuevo_envia or (stock_previo_recibe+cantidad)<>
748         stock_nuevo_recibe then
749             salida := false;
750         END IF;
751         PRINTR(nombre_prueba, salida, salidaEsperada);
752
753     EXCEPTION
754     WHEN OTHERS THEN
755         PRINTR(nombre_prueba, false, salidaEsperada);
756         ROLLBACK;
757
758     end stock_correcto_t;
759
760 END PRUEBA_STOCK;
761
762 /

```

```

752 CREATE OR REPLACE PACKAGE BODY PRUEBAS_VENTA AS
754
756     PROCEDURE inicializar AS
758     BEGIN
759         DELETE FROM VENTA;
760     END inicializar;
761
762     PROCEDURE insertar
763     (nombre_prueba VARCHAR2, i_fecha DATE, i_dni VARCHAR2,
764     salidaEsperada BOOLEAN) AS
765     salida BOOLEAN := true;
766     actual_venta%ROWTYPE;
767     w_ID_Venta INT;
768     BEGIN
769         INSERT INTO venta VALUES(null, i_fecha, i_dni);
770
771         w_ID_Venta := S_ID_Venta.currval;
772         SELECT * INTO actual FROM venta WHERE ID_Venta = w_ID_Venta;
773
774         IF (actual.fechaVenta <> i_fecha) OR (actual.dni <> i_dni) THEN
775             salida := false;
776         END IF;
777
778         PRINTR(nombre_prueba, salida, salidaEsperada);
779
780         EXCEPTION
781         WHEN OTHERS THEN
782             PRINTR(nombre_prueba, false, salidaEsperada);
783             ROLLBACK;
784     END insertar;
785
786     PROCEDURE eliminar
787     (nombre_prueba VARCHAR2, e_id_venta INT, salidaEsperada BOOLEAN) AS
788     salida BOOLEAN := true;
789     cantidad INT;
790     BEGIN
791         DELETE FROM venta WHERE ID_Venta = e_id_venta;
792
793         SELECT COUNT(*) INTO cantidad FROM venta WHERE ID_Venta = e_id_venta;
794         IF (cantidad <> 0) THEN
795             salida := false;
796         END IF;
797
798         PRINTR(nombre_prueba, salida, salidaEsperada);
799
800         EXCEPTION
801         WHEN OTHERS THEN
802             PRINTR(nombre_prueba, false, salidaEsperada);
803             ROLLBACK;
804     END eliminar;
805
806     /*
807     procedure descuento
808     (nombre_prueba VARCHAR2, v_id_venta int, v_preciototal number, v_dni_socio varchar2,
809     salidaEsperada BOOLEAN) as
810     salida BOOLEAN := true;
811     p_preciototal number;
812     begin
813         select preciototal into p_preciototal from venta where id_venta = v_id_venta;
814         if (v_dni_socio <> null) then
815             if (v_preciototal * 0.95 <> p_preciototal) then
816                 salida := false;
817             END IF;
818         END IF;
819
820         printr(nombre_prueba, salida, salidaEsperada);
821         EXCEPTION
822         WHEN OTHERS THEN
823             PRINTR(nombre_prueba, false, salidaEsperada);
824             ROLLBACK;
825     end descuento;*/
826 END PRUEBAS_VENTA;
827
828 /
829
830 CREATE OR REPLACE PACKAGE BODY PRUEBAS_PEDIDO AS

```

```

828 PROCEDURE inicializar AS
829 BEGIN
830     DELETE FROM pedido;
831 END inicializar;
832
833 PROCEDURE insertar
834     (nombre_prueba VARCHAR2, i_fecha DATE,
835      i_id_emplazamiento INT, i_cif VARCHAR2, salidaEsperada BOOLEAN) AS
836 salida BOOLEAN := true;
837 actual_pedido %ROWTYPE;
838 w_ID_pedido INT;
839 BEGIN
840     INSERT INTO pedido VALUES(null, i_fecha, i_id_emplazamiento, i_cif);
841
842     w_ID_pedido := S_ID_Pedido.currval;
843     SELECT * INTO actual FROM pedido WHERE ID_Pedido = w_ID_pedido;
844
845     IF (actual.fechaPedido<>i_fecha) OR (actual.id_emplazamiento<>i_id_emplazamiento) OR (actual.
846 cif<>i_cif) THEN
847         salida := false;
848     END IF;
849
850     PRINTR(nombre_prueba, salida, salidaEsperada);
851
852     EXCEPTION
853     WHEN OTHERS THEN
854         PRINTR(nombre_prueba, false, salidaEsperada);
855         ROLLBACK;
856 END insertar;
857
858 PROCEDURE eliminar
859     (nombre_prueba VARCHAR2, e_id_pedido INT, salidaEsperada BOOLEAN) AS
860 salida BOOLEAN := true;
861 cantidad INT;
862 BEGIN
863     DELETE FROM pedido WHERE ID_Pedido = e_id_pedido;
864
865     SELECT COUNT(*) INTO cantidad FROM pedido WHERE ID_Pedido = e_id_pedido;
866     IF (cantidad<>0) THEN
867         salida := false;
868     END IF;
869
870     PRINTR(nombre_prueba, salida, salidaEsperada);
871
872     EXCEPTION
873     WHEN OTHERS THEN
874         PRINTR(nombre_prueba, false, salidaEsperada);
875         ROLLBACK;
876 END eliminar;
877 END PRUEBAS_PEDIDO;
878 /
879
880 CREATE OR REPLACE PACKAGE BODY PRUEBAS_TRASPASO AS
881
882 PROCEDURE inicializar AS
883 BEGIN
884     DELETE FROM traspaso;
885 END inicializar;
886
887 PROCEDURE insertar
888     (nombre_prueba VARCHAR2, i_fechaTraspaso DATE, i_id_emplazamientoSalida INT,
889      i_id_emplazamientoEntrada INT, salidaEsperada BOOLEAN) AS
890 salida BOOLEAN := true;
891 actual_traspaso %ROWTYPE;
892 w_ID_traspaso INT;
893 BEGIN
894     INSERT INTO traspaso VALUES(null, i_fechaTraspaso, i_id_emplazamientoSalida,
895 i_id_emplazamientoEntrada);
896
897     w_ID_traspaso := S_ID_traspaso.currval;
898     SELECT * INTO actual FROM traspaso WHERE ID_traspaso = w_ID_traspaso;
899
900     IF (actual.fechaTraspaso<>i_fechaTraspaso) OR (actual.id_emplazamientoSalida<>
901 i_id_emplazamientoSalida) OR (actual.id_emplazamientoEntrada<>i_id_emplazamientoEntrada) THEN
902         salida := false;
903     END IF;

```

```

902      PRINTR(nombre_prueba, salida, salidaEsperada);
904
905      EXCEPTION
906      WHEN OTHERS THEN
907          PRINTR(nombre_prueba, false, salidaEsperada);
908          ROLLBACK;
909  END insertar;
910
911  PROCEDURE eliminar
912      (nombre_prueba VARCHAR2, e_id_Traspaso INT, salidaEsperada BOOLEAN) AS
913      salida BOOLEAN := true;
914      cantidad INT;
915  BEGIN
916      DELETE FROM traspaso WHERE ID_traspaso = e_id_traspaso;
917
918      SELECT COUNT(*) INTO cantidad FROM traspaso WHERE ID_traspaso = e_id_traspaso;
919      IF (cantidad <> 0) THEN
920          salida := false;
921      END IF;
922
923      PRINTR(nombre_prueba, salida, salidaEsperada);
924
925      EXCEPTION
926      WHEN OTHERS THEN
927          PRINTR(nombre_prueba, false, salidaEsperada);
928          ROLLBACK;
929  END eliminar;
930  END PRUEBAS_TRASPASO;
931
932  /
933
934  CREATE OR REPLACE PACKAGE BODY PRUEBAS_Solicitud_Traspaso AS
935
936      PROCEDURE inicializar AS
937      BEGIN
938          DELETE FROM Solicitud_Traspaso;
939      END inicializar;
940
941      PROCEDURE insertar
942      (nombre_prueba VARCHAR2, i_fechaSolicitud DATE, i_id_emplazamientoSalida INT,
943       i_id_emplazamientoEntrada INT, salidaEsperada BOOLEAN) AS
944      salida BOOLEAN := true;
945      actual Solicitud_Traspaso%ROWTYPE;
946      w_ID_Solicitud INT;
947  BEGIN
948      INSERT INTO Solicitud_Traspaso VALUES(null, i_fechaSolicitud, i_id_emplazamientoSalida,
949       i_id_emplazamientoEntrada);
950
951      w_ID_Solicitud := S_ID_Solicitud.currval;
952      SELECT * INTO actual FROM Solicitud_Traspaso WHERE ID_Solicitud = w_ID_Solicitud;
953
954      IF (actual.fechaSolicitud <> i_fechaSolicitud) OR (actual.id_emplazamientoSalida <>
955       i_id_emplazamientoSalida) OR (actual.id_emplazamientoEntrada <> i_id_emplazamientoEntrada) THEN
956          salida := false;
957      END IF;
958
959      PRINTR(nombre_prueba, salida, salidaEsperada);
960
961      EXCEPTION
962      WHEN OTHERS THEN
963          PRINTR(nombre_prueba, false, salidaEsperada);
964          ROLLBACK;
965  END insertar;
966
967      PROCEDURE eliminar
968      (nombre_prueba VARCHAR2, e_id_solicitudTraspaso INT, salidaEsperada BOOLEAN) AS
969      salida BOOLEAN := true;
970      cantidad INT;
971  BEGIN
972      DELETE FROM Solicitud_Traspaso WHERE ID_Solicitud = e_id_solicitudTraspaso;
973
974      SELECT COUNT(*) INTO cantidad FROM Solicitud_Traspaso WHERE ID_Solicitud =
975       e_id_solicitudTraspaso;
976      IF (cantidad <> 0) THEN
977          salida := false;
978      END IF;

```



```

976         PRINTR(nombre_prueba, salida, salidaEsperada);
978     EXCEPTION
980     WHEN OTHERS THEN
981         PRINTR(nombre_prueba, false, salidaEsperada);
982         ROLLBACK;
983     END eliminar;
984 END PRUEBAS_Solicitud_Traspaso;
985
986 /
987
988 CREATE OR REPLACE PACKAGE BODY PRUEBAS_Albaran AS
989
990     PROCEDURE inicializar AS
991     BEGIN
992         DELETE FROM Albaran;
993     END inicializar;
994
995     PROCEDURE insertar
996     (nombre_prueba VARCHAR2, i_fecha DATE,
997      i_id_pedido INT, salidaEsperada BOOLEAN) AS
998     salida BOOLEAN := true;
999     actual Albaran%ROWTYPE;
1000     w_ID_Albaran INT;
1001     BEGIN
1002         INSERT INTO Albaran VALUES(null, i_fecha, i_id_pedido);
1003
1004         w_ID_Albaran := S_ID_Albaran.currval;
1005         SELECT * INTO actual FROM Albaran WHERE ID_Albaran = w_ID_Albaran;
1006
1007         IF (actual.fechaFirma<>i_fecha) OR (actual.id_pedido<>i_id_pedido) THEN
1008             salida := false;
1009         END IF;
1010
1011         PRINTR(nombre_prueba, salida, salidaEsperada);
1012
1013     EXCEPTION
1014     WHEN OTHERS THEN
1015         PRINTR(nombre_prueba, false, salidaEsperada);
1016         ROLLBACK;
1017     END insertar;
1018
1019     PROCEDURE eliminar
1020     (nombre_prueba VARCHAR2, e_id_albaran INT, salidaEsperada BOOLEAN) AS
1021     salida BOOLEAN := true;
1022     cantidad INT;
1023     BEGIN
1024         DELETE FROM Albaran WHERE ID_Albaran = e_id_albaran;
1025
1026         SELECT COUNT(*) INTO cantidad FROM Albaran WHERE ID_Albaran = e_id_albaran;
1027         IF (cantidad<>0) THEN
1028             salida := false;
1029         END IF;
1030
1031         PRINTR(nombre_prueba, salida, salidaEsperada);
1032
1033     EXCEPTION
1034     WHEN OTHERS THEN
1035         PRINTR(nombre_prueba, false, salidaEsperada);
1036         ROLLBACK;
1037     END eliminar;
1038 END PRUEBAS_Albaran;
1039
1040 /
1041
1042 CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_PRODUCTO_TRASPASO AS
1043
1044     PROCEDURE inicializar AS
1045     BEGIN
1046         DELETE FROM asociacion_Producto_traspaso;
1047     END inicializar;
1048
1049     PROCEDURE insertar
1050     (nombre_prueba VARCHAR2, i_id_producto INT, i_id_Traspaso INT,
1051      i_cantidad NUMBER, salidaEsperada BOOLEAN) AS
1052     salida BOOLEAN := true;

```

```

1054 actual asociacion_Producto_traspaso %ROWTYPE;
1055 BEGIN
1056     INSERT INTO asociacion_Producto_traspaso VALUES(i_id_Traspaso, i_id_producto, i_cantidad);
1057
1058     SELECT * INTO actual FROM asociacion_Producto_traspaso WHERE ID_Producto = i_id_producto and
1059     ID_Traspaso = i_id_Traspaso;
1060
1061     IF (actual.id_producto<>i_id_producto) OR (actual.id_Traspaso<>i_id_Traspaso) OR (actual.
1062     cantidad<>i_cantidad) THEN
1063         salida := false;
1064     END IF;
1065
1066     PRINTR(nombre_prueba, salida, salidaEsperada);
1067
1068     EXCEPTION
1069     WHEN OTHERS THEN
1070         PRINTR(nombre_prueba, false, salidaEsperada);
1071         ROLLBACK;
1072 END insertar;
1073
1074 PROCEDURE eliminar
1075 (nombre_prueba VARCHAR2, e_id_Traspaso INT, e_id_producto INT, salidaEsperada BOOLEAN) AS
1076 salida BOOLEAN := true;
1077 cantidad INT;
1078 BEGIN
1079     DELETE FROM asociacion_Producto_traspaso WHERE ID_Producto = e_id_producto and
1080     ID_Traspaso = e_id_Traspaso;
1081
1082     SELECT COUNT(*) INTO cantidad FROM asociacion_Producto_traspaso WHERE ID_Producto =
1083     e_id_producto and ID_Traspaso = e_id_Traspaso;
1084     IF (cantidad<>0) THEN
1085         salida := false;
1086     END IF;
1087
1088     PRINTR(nombre_prueba, salida, salidaEsperada);
1089
1090     EXCEPTION
1091     WHEN OTHERS THEN
1092         PRINTR(nombre_prueba, false, salidaEsperada);
1093         ROLLBACK;
1094 END eliminar;
1095
1096 END PRUEBAS_A_PRODUCTO_TRASPASO;
1097
1098 /
1099
1100 CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_PRODUCTO_SOLICITUD AS
1101
1102     PROCEDURE inicializar AS
1103     BEGIN
1104         DELETE FROM asociacion_Producto_Solicitud;
1105     END inicializar;
1106
1107     PROCEDURE insertar
1108     (nombre_prueba VARCHAR2, i_id_producto INT, i_id_Solicitud INT,
1109     i_cantidad NUMBER, salidaEsperada BOOLEAN) AS
1110     salida BOOLEAN := true;
1111     actual asociacion_Producto_Solicitud %ROWTYPE;
1112     BEGIN
1113         INSERT INTO asociacion_Producto_Solicitud VALUES(i_id_Solicitud, i_id_producto, i_cantidad);
1114
1115         SELECT * INTO actual FROM asociacion_Producto_Solicitud WHERE ID_Producto = i_id_producto and
1116         ID_Solicitud = i_id_Solicitud;
1117
1118         IF (actual.id_producto<>i_id_producto) OR (actual.id_Solicitud<>i_id_Solicitud) OR (actual.
1119         cantidad<>i_cantidad) THEN
1120             salida := false;
1121         END IF;
1122
1123         PRINTR(nombre_prueba, salida, salidaEsperada);
1124
1125         EXCEPTION
1126         WHEN OTHERS THEN
1127             PRINTR(nombre_prueba, false, salidaEsperada);
1128             ROLLBACK;
1129     END insertar;
1130
1131     PROCEDURE eliminar

```

```

1124      (nombre_prueba VARCHAR2, e_id_Solicitud INT, e_id_producto INT, salidaEsperada BOOLEAN) AS
1125      salida BOOLEAN := true;
1126      cantidad INT;
1127      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;
1131          IF (cantidad <> 0) THEN
1132              salida := false;
1133          END IF;

1134          PRINTR(nombre_prueba, salida, salidaEsperada);

1135          EXCEPTION
1136          WHEN OTHERS THEN
1137              PRINTR(nombre_prueba, false, salidaEsperada);
1138              ROLLBACK;
1139      END eliminar;
1140  END PRUEBAS_A_PRODUCTO_SOLICITUD;
1141  /
1142
1143  CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_PEDIDO_PRODUCTO AS
1144
1145      PROCEDURE inicializar AS
1146      BEGIN
1147          DELETE FROM asociacion_Pedido_Producto;
1148      END inicializar;
1149
1150      PROCEDURE insertar
1151      (nombre_prueba VARCHAR2, i_id_pedido INT, i_id_producto INT,
1152       i_cantidad NUMBER, i_precioCompra number, i_iva INT,
1153       salidaEsperada BOOLEAN) AS
1154      salida BOOLEAN := true;
1155      actual asociacion_Pedido_Producto%ROWTYPE;
1156      BEGIN
1157          INSERT INTO asociacion_Pedido_Producto VALUES(i_id_pedido, i_id_producto, i_cantidad,
i_precioCompra, i_iva);

1158          SELECT * INTO actual FROM asociacion_Pedido_Producto WHERE ID_pedido = i_id_pedido and
ID_producto = i_id_producto;

1159          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
1160              salida := false;
1161          END IF;

1162          PRINTR(nombre_prueba, salida, salidaEsperada);

1163          EXCEPTION
1164          WHEN OTHERS THEN
1165              PRINTR(nombre_prueba, false, salidaEsperada);
1166              ROLLBACK;
1167      END insertar;
1168
1169      PROCEDURE eliminar
1170      (nombre_prueba VARCHAR2, e_id_pedido INT, e_id_producto INT,
1171       salidaEsperada BOOLEAN) AS
1172      salida BOOLEAN := true;
1173      cantidad INT;
1174      BEGIN
1175          DELETE FROM asociacion_Pedido_Producto WHERE ID_pedido = e_id_pedido and ID_producto =
e_id_producto;

1176          SELECT COUNT(*) INTO cantidad FROM asociacion_Pedido_Producto WHERE ID_pedido =
e_id_pedido and ID_producto = e_id_producto;
1177          IF (cantidad <> 0) THEN
1178              salida := false;
1179          END IF;

1180          PRINTR(nombre_prueba, salida, salidaEsperada);

1181          EXCEPTION
1182          WHEN OTHERS THEN
1183              PRINTR(nombre_prueba, false, salidaEsperada);

```

```

1194         ROLLBACK;
1195     END eliminar;
1196
1197 END PRUEBAS_A_PEDIDO_PRODUCTO;
1198
1199 /
1200
1201 CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_VENTA_PRODUCTO AS
1202
1203     PROCEDURE inicializar AS
1204     BEGIN
1205         DELETE FROM asociacion_Venta_Producto;
1206     END inicializar;
1207
1208     PROCEDURE insertar
1209     (nombre_prueba VARCHAR2, i_id_venta INT, i_id_producto INT,
1210      i_cantidad NUMBER, i_precioVenta NUMBER, i_ivaVenta NUMBER,
1211      salidaEsperada BOOLEAN) AS
1212     salida BOOLEAN := true;
1213     actual asociacion_Venta_Producto %ROWTYPE;
1214     BEGIN
1215         INSERT INTO asociacion_Venta_Producto VALUES(i_id_venta, i_id_producto, i_cantidad,
1216             i_precioVenta, i_ivaVenta);
1217
1218         SELECT * INTO actual FROM asociacion_Venta_Producto WHERE ID_Venta = i_id_venta and
1219             ID_producto = i_id_producto;
1220
1221         IF (actual.id_venta <> i_id_venta) OR (actual.id_producto <> i_id_producto) OR (actual.cantidad <>
1222             i_cantidad) OR (actual.precioVenta <> i_precioVenta) OR (actual.ivaVenta <> i_ivaVenta) THEN
1223             salida := false;
1224         END IF;
1225
1226         PRINTR(nombre_prueba, salida, salidaEsperada);
1227
1228     EXCEPTION
1229     WHEN OTHERS THEN
1230         PRINTR(nombre_prueba, false, salidaEsperada);
1231         ROLLBACK;
1232     END insertar;
1233
1234     PROCEDURE eliminar
1235     (nombre_prueba VARCHAR2, e_id_venta INT, e_id_producto INT, salidaEsperada BOOLEAN) AS
1236     salida BOOLEAN := true;
1237     cantidad INT;
1238     BEGIN
1239         DELETE FROM asociacion_Venta_Producto WHERE ID_Venta = e_id_venta and ID_producto =
1240             e_id_producto;
1241
1242         SELECT COUNT(*) INTO cantidad FROM asociacion_Venta_Producto WHERE ID_Venta = e_id_venta
1243             and ID_producto = e_id_producto;
1244         IF (cantidad <> 0) THEN
1245             salida := false;
1246         END IF;
1247
1248         PRINTR(nombre_prueba, salida, salidaEsperada);
1249
1250     EXCEPTION
1251     WHEN OTHERS THEN
1252         PRINTR(nombre_prueba, false, salidaEsperada);
1253         ROLLBACK;
1254     END eliminar;
1255 END PRUEBAS_A_VENTA_PRODUCTO;
1256
1257 /
1258
1259 CREATE OR REPLACE PACKAGE BODY PRUEBAS_FUNCIONES AS
1260
1261     PROCEDURE precioLinea_A_Pedido
1262     (nombre_prueba VARCHAR2, ID_PRODUCTO NUMBER, ID_PEDIDO NUMBER, esperado number, salidaEsperada
1263      BOOLEAN) AS
1264     salida BOOLEAN := true;
1265     precio number;
1266     BEGIN
1267         SELECT precioLinea_Aso_Pedido(ID_PRODUCTO, ID_PEDIDO) INTO precio FROM dual;
1268

```

```

1266         IF (precio<>esperado) THEN
            salida := false;
1268         END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);
1270
        EXCEPTION
1272         WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
1274         ROLLBACK;
        END PRECIOLINEA_A_PEDIDO;
1276
        PROCEDURE precioLinea_A_Venta
1278         (nombre_prueba VARCHAR2, ID_PRODUCTO NUMBER, ID_VENTA NUMBER, esperado number, salidaEsperada
            BOOLEAN) AS
            salida BOOLEAN := true;
1280         precio number;
        BEGIN
1282             SELECT precioLinea_Aso_VENTA(ID_PRODUCTO, ID_VENTA) INTO precio FROM dual;

            IF (precio<>esperado) THEN
1284                 salida := false;
1286             END IF;

            PRINTR(nombre_prueba, salida, salidaEsperada);
1288
            EXCEPTION
1290            WHEN OTHERS THEN
1292                PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
1294            END PRECIOLINEA_A_VENTA;

1296        PROCEDURE PRECIO_VENTA
            (nombre_prueba VARCHAR2, ID_Venta NUMBER, esperado number, salidaEsperada BOOLEAN) AS
1298            salida BOOLEAN := true;
            precio number;
1300        BEGIN
            SELECT precioTotal_Venta(ID_VENTA) INTO precio FROM dual;
1302
            IF (precio<>esperado) THEN
1304                 salida := false;
            END IF;
1306
            PRINTR(nombre_prueba, salida, salidaEsperada);
1308
            EXCEPTION
1310            WHEN OTHERS THEN
1312                PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
            END PRECIO_VENTA;
1314
1316        PROCEDURE precio_Factura
            (nombre_prueba VARCHAR2, ID_Venta NUMBER, esperado number, salidaEsperada BOOLEAN) AS
            salida BOOLEAN := true;
1318        precio number;
        BEGIN
1320            SELECT precioTotal_Factura(ID_VENTA) INTO precio FROM dual;

            IF (precio<>esperado) THEN
1322                 salida := false;
1324             END IF;

            PRINTR(nombre_prueba, salida, salidaEsperada);
1326
            EXCEPTION
1328            WHEN OTHERS THEN
1330                PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
1332        END precio_Factura;

1334        PROCEDURE precio_Pedido
            (nombre_prueba VARCHAR2, ID_PEDIDO NUMBER, esperado number, salidaEsperada BOOLEAN) AS
1336            salida BOOLEAN := true;
            precio number;
1338        BEGIN
            SELECT precioTotal_Pedido(ID_PEDIDO) INTO precio FROM dual;
1340

```

```

1342         IF (precio<>esperado) THEN
1343             salida := false;
1344         END IF;
1345
1346         PRINTR(nombre_prueba, salida, salidaEsperada);
1347
1348         EXCEPTION
1349         WHEN OTHERS THEN
1350             PRINTR(nombre_prueba, false, salidaEsperada);
1351             ROLLBACK;
1352     END precio_Pedido;
1353
1354     PROCEDURE precio_Albaran
1355     (nombre_prueba VARCHAR2, ID_PEDIDO NUMBER, esperado number, salidaEsperada BOOLEAN) AS
1356     salida BOOLEAN := true;
1357     precio number;
1358     BEGIN
1359         SELECT precioTotal_Albaran(ID_PEDIDO) INTO precio FROM dual;
1360
1361         IF (precio<>esperado) THEN
1362             salida := false;
1363         END IF;
1364
1365         PRINTR(nombre_prueba, salida, salidaEsperada);
1366
1367         EXCEPTION
1368         WHEN OTHERS THEN
1369             PRINTR(nombre_prueba, false, salidaEsperada);
1370             ROLLBACK;
1371     END precio_Albaran;
1372
1373 END PRUEBAS_FUNCIONES;
1374 /
1375 DROP SEQUENCE S_ID_Producto;
1376 DROP SEQUENCE S_ID-Traspaso;
1377 DROP SEQUENCE S_ID_Solicitud;
1378 DROP SEQUENCE S_ID_Pedido;
1379 DROP SEQUENCE S_ID_Albaran;
1380 DROP SEQUENCE S_ID_Factura;
1381 DROP SEQUENCE S_ID_Emplazamiento;
1382 DROP SEQUENCE S_ID_Venta;
1383
1384 CREATE SEQUENCE S_ID_Emplazamiento START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1385 CREATE SEQUENCE S_ID_Venta START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1386 CREATE SEQUENCE S_ID_Pedido START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1387 CREATE SEQUENCE S_ID_Albaran START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1388 CREATE SEQUENCE S_ID_Producto START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1389 CREATE SEQUENCE S_ID-Traspaso START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1390 CREATE SEQUENCE S_ID_Solicitud START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1391 CREATE SEQUENCE S_ID_Factura START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1392
1393 /* EXECUTE DE PRUEBAS */
1394 -- Inicializacion
1395 EXECUTE pruebas_albaran.inicializar();
1396 EXECUTE PRUEBAS_A_PRODUCTO_TRASPASO.inicializar();
1397 EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.inicializar();
1398 EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.inicializar();
1399 EXECUTE PRUEBAS_A_VENTA_PRODUCTO.inicializar();
1400 EXECUTE PRUEBAS_FACTURA.inicializar();
1401 EXECUTE pruebas_venta.inicializar();
1402 EXECUTE pruebas_pedido.inicializar();
1403 EXECUTE pruebas_proveedor.inicializar();
1404 EXECUTE PRUEBAS_STOCK.inicializar();
1405 EXECUTE pruebas_producto.inicializar();
1406 EXECUTE PRUEBAS_TRASPASO.inicializar();
1407 EXECUTE PRUEBAS_SOLICITUD_TRASPASO.inicializar();
1408 EXECUTE PRUEBAS_SOCIO.inicializar();
1409 EXECUTE pruebas_emplazamiento.inicializar();
1410 --SOCIO
1411 EXECUTE PRUEBAS_SOCIO.inicializar();
1412 EXECUTE PRUEBAS_SOCIO.insertar('Socio Insertar', 'Daniel', 'Gonzalez', '54148787G', 'Avenida del
1413 pepinillo', TO_DATE('10102015','DDMMYYYY'), 'dani@us.es', true);
1414 EXECUTE PRUEBAS_SOCIO.insertar('Socio Insertar', 'Jose Luis', 'Marmol Romero', '29613400A', 'Calle
1415 Virgen', TO_DATE('10101996','DDMMYYYY'), 'jlmr@us.es', true);
1416 EXECUTE PRUEBAS_SOCIO.actualizar('Socio Actualizar', '54148787G', 'Avenida del pepinillo234', '
1417 dani@us.es', true);
1418 EXECUTE PRUEBAS_SOCIO.eliminar('Socio Eliminar', '54148787G', true);

```

```

1416 --PROVEEDOR
EXECUTE PRUEBAS_PROVEEDOR.inicializar();
1418 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);
1420 EXECUTE PRUEBAS_PROVEEDOR.actualizar('Proveedor Actualizar', 'B54120214', 'Piexs', 111111111, 'a@pi.
es', true);
EXECUTE PRUEBAS_PROVEEDOR.eliminar('Proveedor Eliminar', 'B54120214', true);

1422 --PRODUCTO
1424 EXECUTE PRUEBAS_PRODUCTO.inicializar();
COMMIT WORK;
1426 EXECUTE PRUEBAS_PRODUCTO.insertar('Producto con IVA invalido','Chaleco de lana','por una gran
diseñadora','Jerseys',46.95,2,false);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto con categoria invalida','Chaleco de lana','por una gran
diseñadora','Ropa',46.95,0.21,false);
1428 EXECUTE PRUEBAS_PRODUCTO.insertar('Producto con precio invalido','Chaleco de lana','por una gran
diseñadora','Jerseys',-2,0.21,false);

1430 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);
1432 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);
1434 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 diseñadora','
Jerseys',46.95,0.21,true);
1436 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);
1438 EXECUTE PRUEBAS_PRODUCTO.eliminar('Producto eliminar',S_ID_Producto.currval,true);
--EMPLAZAMIENTO
1440 EXECUTE PRUEBAS_EMPLAZAMIENTO.inicializar();
COMMIT WORK;
1442 EXECUTE PRUEBAS_EMPLAZAMIENTO.insertar('Emplazamiento con tipo invalido', 'Calle de la piruleta 25',
958632666,'Emplazamiento', false);

1444 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);
1446 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);
1448 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
1450 EXECUTE PRUEBA_STOCK.inicializar();
1452 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);
1454 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);
1456 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);
1458 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);

1460 --VENTA
1462 EXECUTE pruebas_venta.inicializar();
EXECUTE pruebas_venta.insertar('Venta insercion',sysdate,'29613400A',true);

```

```

1464 EXECUTE pruebas_venta.insertar('Venta insercion',sysdate,null,true);
EXECUTE pruebas_venta.insertar('Venta insercion',sysdate,null,true);
1466 EXECUTE pruebas_venta.eliminar('Venta eliminar',S_ID_venta.currval,true);
/*
1468 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);
1470 INSERT INTO FACTURA VALUES(0,0,sysdate,'F',4,3);
*/
1472 --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;
1474 --ASOCIACION_VENTA_PRODUCTO
EXECUTE PRUEBAS_A_VENTA_PRODUCTO.inicializar();
1476 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);
1478 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);
1480 EXECUTE PRUEBAS_A_VENTA_PRODUCTO.eliminar('Venta-Producto eliminar',s_id_venta.currval-2,
    s_id_producto.currval-1,true);

1482 --FACTURA
EXECUTE PRUEBAS_factura.inicializar();
1484 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);
1486 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);
1488 --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);
1490 EXECUTE Prueba_stock.stock_correcto('Actualizacion de stock tras venta',s_id_emplazamiento.currval-1,
    s_id_producto.currval-2,21,3,false);

1492 --PEDIDO
EXECUTE PRUEBAS_PEDIDO.inicializar();
1494 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);
1496 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);
1498
1500 --PEDIDO_PRODUCTO
EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.inicializar();
1502 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);
1504 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);
1506 --ALBARAN
EXECUTE PRUEBAS_ALBARAN.inicializar();
1508 EXECUTE PRUEBAS_ALBARAN.insertar('Albaran insercion',sysdate,S_ID_Pedido.currval-1,true);
EXECUTE PRUEBAS_STOCK.stock_correcto_p('Actualizacion de stock tras comprar',s_id_emplazamiento.
    currval-2,s_id_producto.currval-1,8,21,true);
EXECUTE PRUEBAS_STOCK.stock_correcto_p('Actualizacion de stock tras comprar',s_id_emplazamiento.
    currval-2,s_id_producto.currval-1,8,15,false);
1512 EXECUTE PRUEBAS_ALBARAN.insertar('Albaran insercion',sysdate,S_ID_Pedido.currval-2,true);
EXECUTE PRUEBAS_ALBARAN.eliminar('Albaran eliminacion',S_ID_Albaran.currval,true);
1514
1516 --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);

```



```

1518 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);
1520 --ASOC_PRODUCTO_SOLICITUD
EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.inicializar();
1522 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);
1524 EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.insertar('A-Producto-Solicitud Insertar', S_ID_Producto.currval
      -2, S_ID_Solicitud.currval, 3, true);
COMMIT WORK;
1526 EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.insertar('Prueba de Trigger solicitud_stock_minimo',
      S_ID_Producto.currval-1, S_ID_Solicitud.currval, 28, false);

1528 --TRASPASO
EXECUTE PRUEBAS_TRASPASO.inicializar();
1530 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);
1532 EXECUTE PRUEBAS_TRASPASO.eliminar('Traspaso eliminar',S_ID_traspaso.currval,true);

1534 --ASOC_PRODUCTO_TRASPASO
EXECUTE PRUEBAS_A_PRODUCTO_TRASPASO.inicializar();
1536 EXECUTE PRUEBAS_A_PRODUCTO_TRASPASO.insertar('A-Producto-Traspaso Insertar', S_ID_Producto.currval-1,
      S_ID_traspaso.currval-1, 4, true);
EXECUTE PRUEBAS_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);
1538 EXECUTE PRUEBAS_A_PRODUCTO_TRASPASO.eliminar('A-Producto-Traspaso Eliminar', S_ID_traspaso.currval-1,
      S_ID_Producto.currval-1, true);
--Pruebas de funciones

1540

1542 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);
1544 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);
1546 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);
1548 EXECUTE PRUEBAS_FUNCIONES.precio_Albaran('Funcion calculo precio total albaran',2,220.5,true);
-- Tiene que dar 220.5
1550 --SELECT precioLinea_Aso_Pedido(9,2) FROM DUAL;

1552 -- Tiene que dar 220.5
--select preciototal_pedido(2) from dual;
1554
-- Tiene que dar 220.5
--select preciototal_albaran(2) from dual;
1556

-- Tiene que dar 31.5
--select preciototal_venta(2) from dual;
1558

-- Tiene que dar 31.5*0.95 = 29.93
--select preciototal_factura(1) from dual;
1560

--Tienda que dar 93.9
--SELECT precioLinea_Aso_Venta(9,2) FROM DUAL;
1562

-- 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

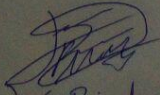
10. Apéndice


10.1. Actas de reunión


Hay día 06/ octubre /2016 se ha realizado la primera reunión con la persona dueña de Oñite Mader, con la cual hemos tratado ciertos temas como:

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

Asistentes a la reunión:


Jesús Pineda


Laura Gómez.


José Luis Kármel Romero.

Hoy Día 20/ Octubre/ 2016 se ha realizado la segunda reunión con Laura (Dona de Ornato Kados) para enseñarle el proceso del trabajo para que todo estuviere de acuerdo con sus expectativas.

Después de mostrarle el avance se encuentra conforme con este.

* Asistentes a la reunión:

Jesús Pineda

Laura Gómez