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

Índice

1. Introducción

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

2. Glosario de Términos

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

3. Modelo de negocio

3.1. Proceso de venta

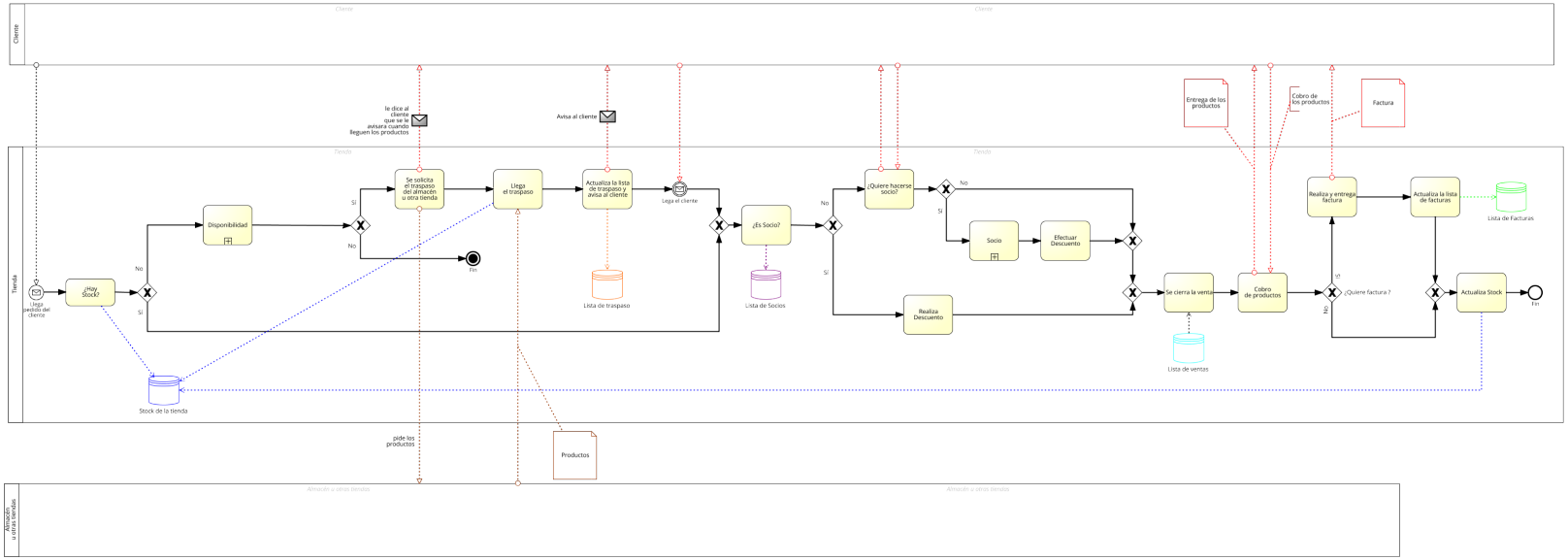
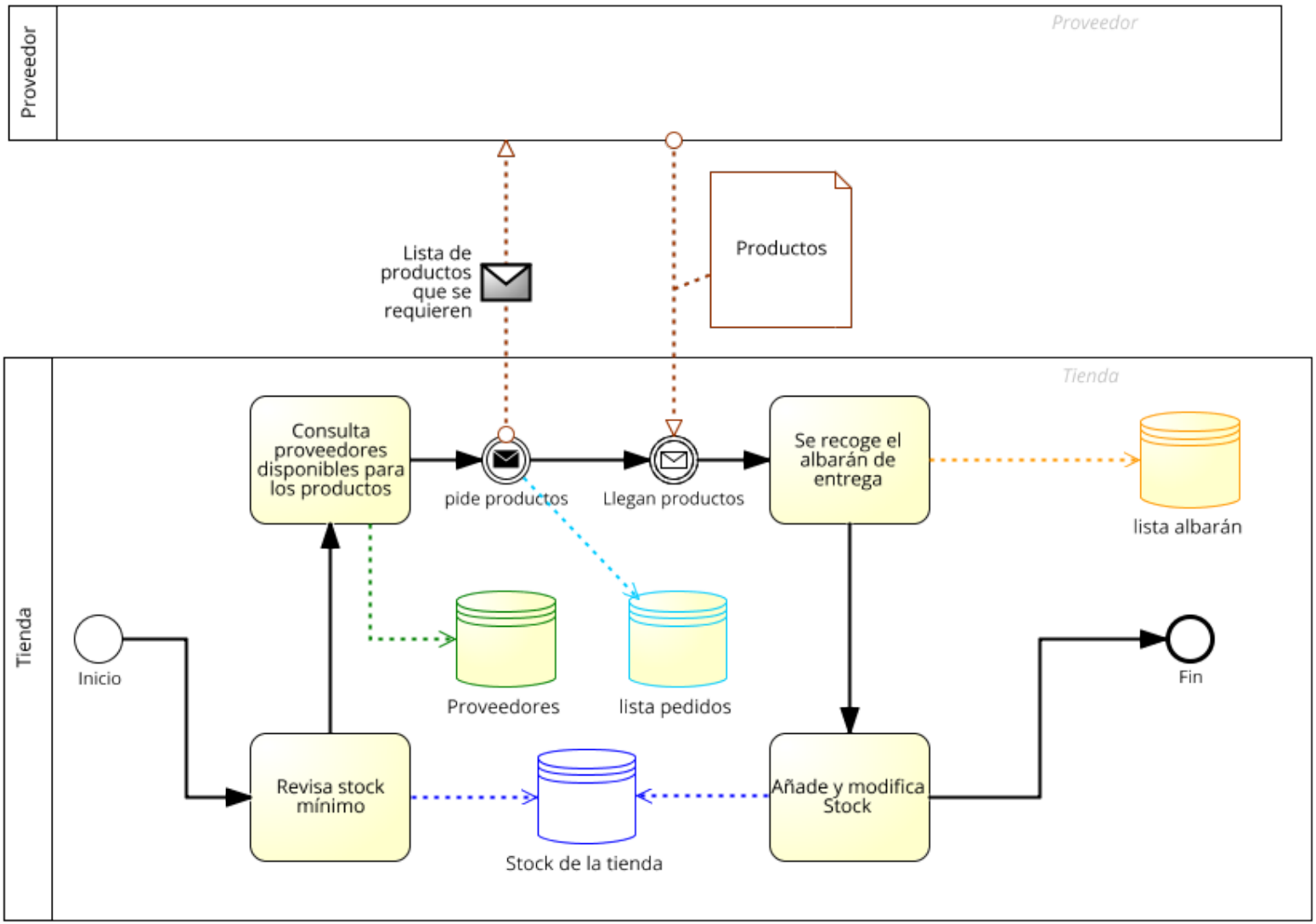
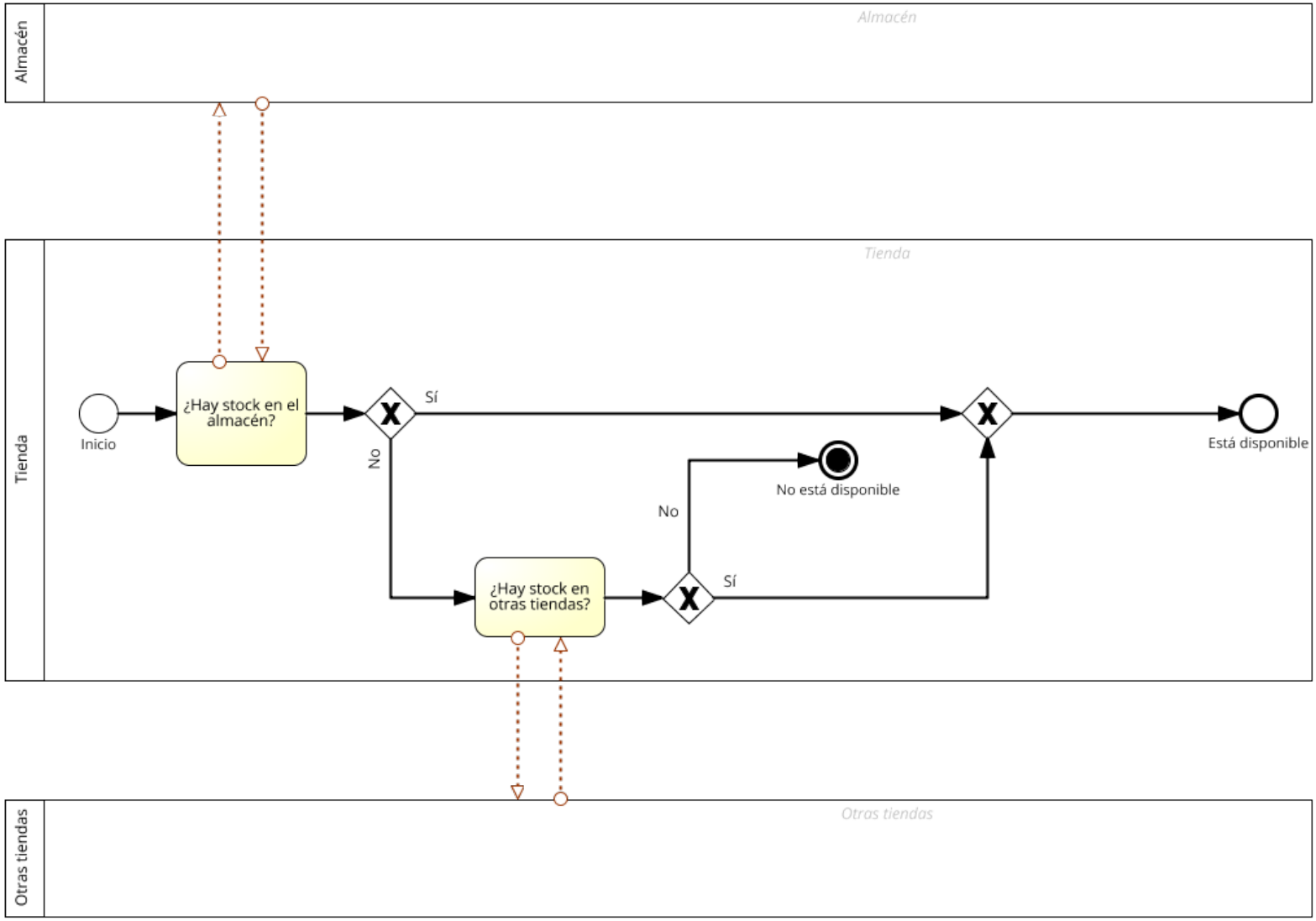


Figura 1: Proceso de venta

3.2. Proceso de compra al proveedor

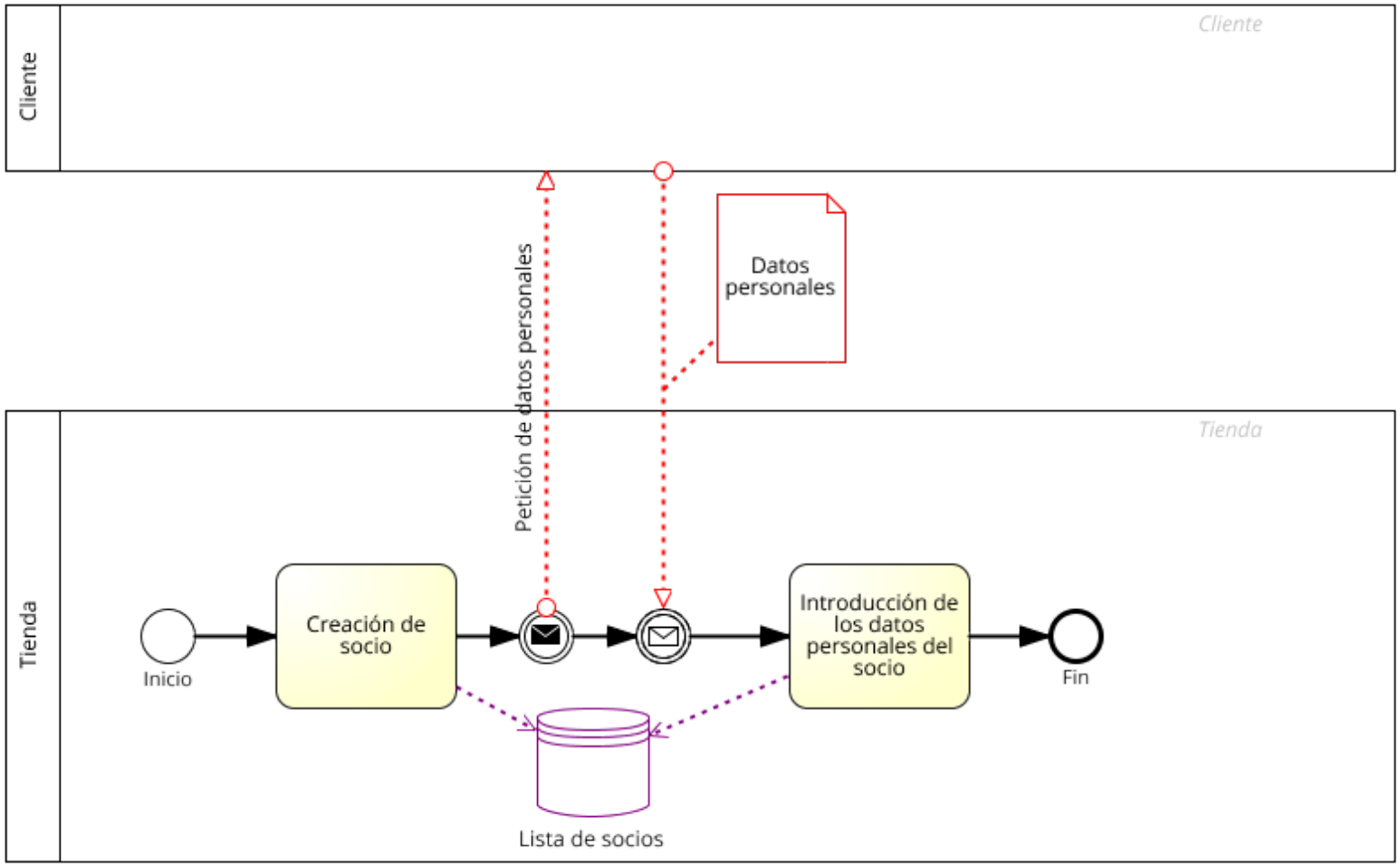


3.3. Proceso de disponibilidad



.png

3.4. Proceso de creación de socios



.png

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.

4.2. Historias de usuario

ID	Historia
HU-1	▪ **Como** trabajador ▪ **Quiero** conocer las ventas diarias del negocio ▪ **Para** hacer ajustes a las ofertas del mismo
HU-2	▪ **Como** propietario de la cadena ▪ **Quiero** conocer el stock disponible en cada negocio ▪ **Para** organizar los pedidos a proveedores
HU-3	▪ **Como** propietario de la cadena ▪ **Quiero** conocer las ganancias mensuales ▪ **Para** tener un balance mensual
HU-4	▪ **Como** cliente ▪ **Quiero** ver una lista de los productos ofrecidos y disponibles ▪ **Para** realizar mis compras

5. Catálogo de requisitos

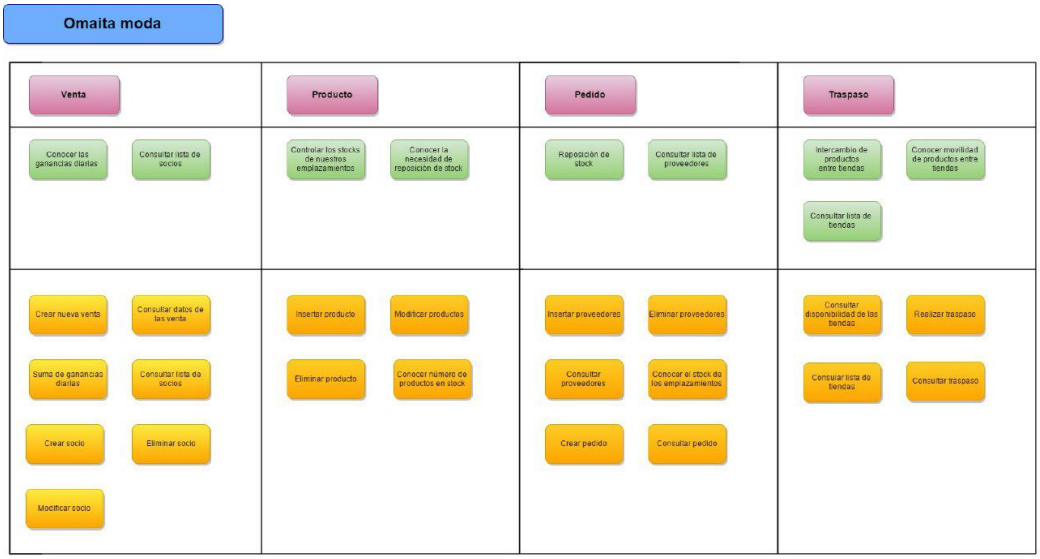


Figura 5: Mapa de requisitos

5.1. 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
  - Precio total de la venta.
  - 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.

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:
  - Precio final de la venta.
  - 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.

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.

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.

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

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

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

---

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.
    - Precio total del pedido.
    - Una asociación que controla la cantidad de cada producto del pedido, el precio y el IVA.
- **Para** conocer los datos de los pedidos realizados.

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.

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.
  - Precio total.
- **Para** tener un control de los pedidos recibidos los cuales son controlados por los albaranes.

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.

5.2. Requisitos funcionales

RF-01 - Crear ventas

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

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

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.

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.

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.

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.

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.

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.

RF-09 - Crear pedidos

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

---

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.

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.

RF-12 - Crear traspasos

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

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

---

RF-14 - Consultar traspasos

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

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.

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.

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.

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.

5.3. 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.

5.4. 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

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.

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

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

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

6. Pruebas de aceptación

PA-01 Ventas:

- Se realiza una venta y con una consulta vemos que la fecha y el precio total es el esperado.

PA-02 Facturas y Devoluciones:

- Se realiza una factura y con una consulta vemos que el precio final de la venta, la fecha, el número de la factura y el IVA es el esperado.
- Un cliente devuelve un producto dentro del plazo de 30 días y por tanto se devuelve el dinero al cliente, realizamos una consulta sobre la factura del producto devuelto y vemos la factura original con el campo devuelto como True.
- Tras la situación anterior el trabajador crea una nueva factura con todos los productos originales de la compra menos con el que se ha devuelto, entonces se realiza una consulta a la factura y vemos como se ha eliminado correctamente el producto devuelto y no se ha contabilizado en el precio de la factura. (Situación que solo se da si el producto devuelto pertenece a una factura de más productos)

- Un cliente desea devolver un producto tras 30 días de su compra, los artículos no pueden ser devueltos debido a que ha pasado más de 30 días desde su compra.

PA-03 Socios:

- Se añade un nuevo socio, se actualiza la lista de socios y este aparece con todos los siguientes datos: nombre, apellidos, DNI, dirección, fecha de nacimiento y email.
- Se elimina un socio, se actualiza la lista de socios y en ella no aparece el socio.
- Se modifica los datos de un socio, se actualiza la lista de socios y este aparece con sus datos modificados.
- Se intenta añadir un nuevo socio, pero le faltan datos, el sistema no permite añadirlo.
- Se intentar añadir un nuevo socio, pero ya está en la base de datos y el sistema no nos lo permite.

PA-04 Productos:

- Se añade un nuevo producto, se actualiza la lista de productos y este aparece con todos los siguientes datos: nombre, descripción, categoría, precio del producto e IVA.
- Se elimina un producto, se actualiza la lista de productos y en ella no aparece el producto eliminado.
- Se modifica los datos de un producto, se actualiza la lista de productos y este aparece con sus datos modificados.
- Se intenta añadir un producto nuevo, pero le faltan datos, el sistema no permite añadirlo.
- Se desea añadir un producto que ya está en la base de datos y el sistema no nos lo permite.

PA-05 Stock:

- Si el stock de un producto es de 10 y un cliente quiere comprar 11, el stock es superado y el sistema no permite esa compra.
- Si el stock de un producto es de 10 y un cliente quiere comprar 3, se le permite la venta y se modifica el stock en la base de datos.
- Se realiza un traspaso entre 2 emplazamientos, hacemos 2 consultas y comprobamos que el stock de cada emplazamiento es el correcto tras la transacción.
- El stock de un producto es de 6 y un cliente realiza una compra de ese producto, entonces se muestra un aviso de que el stock de ese producto ha alcanzado el stock mínimo.
- Consultamos el stock de la tienda y después recibimos el albarán del pedido que habíamos realizado, consultamos de nuevo el stock y vemos que se ha actualizado correctamente.

PA-06 Proveedores:

- Se registra un proveedor nuevo, se actualiza la lista de proveedores y aparece el proveedor con los siguientes datos: nombre, CIF, número de teléfono y email.
- Se modifican los datos de un proveedor, se actualiza la lista de proveedores y sale el proveedor con los datos modificados.
- Se elimina un proveedor, al actualizar la lista de los proveedores, el proveedor eliminado ya no aparece.
- Se intenta añadir un proveedor que ya está en la base de datos y el sistema no lo permite.

PA-07 Pedidos y albaranes:

- El almacén ha llegado a stock mínimo de algunos de sus productos y se realiza un pedido al proveedor o proveedores, tras recibir los productos del proveedor hacemos una consulta para ver el albarán está en orden y que los productos del pedido son los que se solicitaron.
- Se realiza una consulta sobre el pedido que se había realizado y vemos que se muestran los siguientes datos: fecha en la que se realizó, precio total y emplazamiento que realizó el pedido.

- Se intenta realizar un pedido de 15 unidades y salta un mensaje de error, ya que el pedido tiene que ser de 20 unidades mínimo.
- Se crea un albarán con los datos recibidos del albarán de entrega y hacemos una consulta para ver que se muestra correctamente.

PA-08 Traspasos y solicitudes:

- Falta stock de un producto y se solicita a otro emplazamiento, el otro emplazamiento realiza el traspaso y se reciben los productos. Hacemos una consulta para la solicitud y otra consulta para traspaso y vemos que los productos son correctos, además de que aparecen las respectivas fechas y los emplazamientos de la relación.
- Se intenta solicitar un traspaso de 3 unidades de un producto a una tienda, pero la tienda a la que se solicita el traspaso solo posee 6 unidades del producto por lo que salta un mensaje de que no se puede solicitar el traspaso a esta tienda.

PA-09 Emplazamientos:

- Se añade una nueva tienda, se realiza una consulta y vemos que se ha añadido correctamente.
- Se cambia la localización del almacén, se realiza una consulta y vemos que se ha cambiado correctamente.
- Se elimina una tienda, se realiza una consulta y vemos que se ha eliminado correctamente y ya no aparece en la lista de emplazamientos.
- Se realiza una consulta sobre el stock de una tienda y este se muestra correctamente.

PA-10 Descuentos:

- Un socio va a gastar un total de 100 euros en nuestros productos, al ser socio se le aplica un descuento del 5 %, por tanto, se le rebajan 5 euros en la factura.
- Un cliente va a gastar un total de 50 euros, al no ser socio no se le aplica el descuento y tiene que pagar los 50 euros completos.

PA-11 Tamaño mínimo pedidos:

- Se desea realizar un pedido de 10 productos y de cada producto se requieren 20 unidades, por tanto, se puede hacer el pedido.
- Se desea realizar el pedido de un producto con una cantidad equivalente a 15, este pedido no se podría llevar a cabo debido a que no llega a las 20 unidades necesarias.

7. Modelo Conceptual

7.1. Diagramas de clases UML

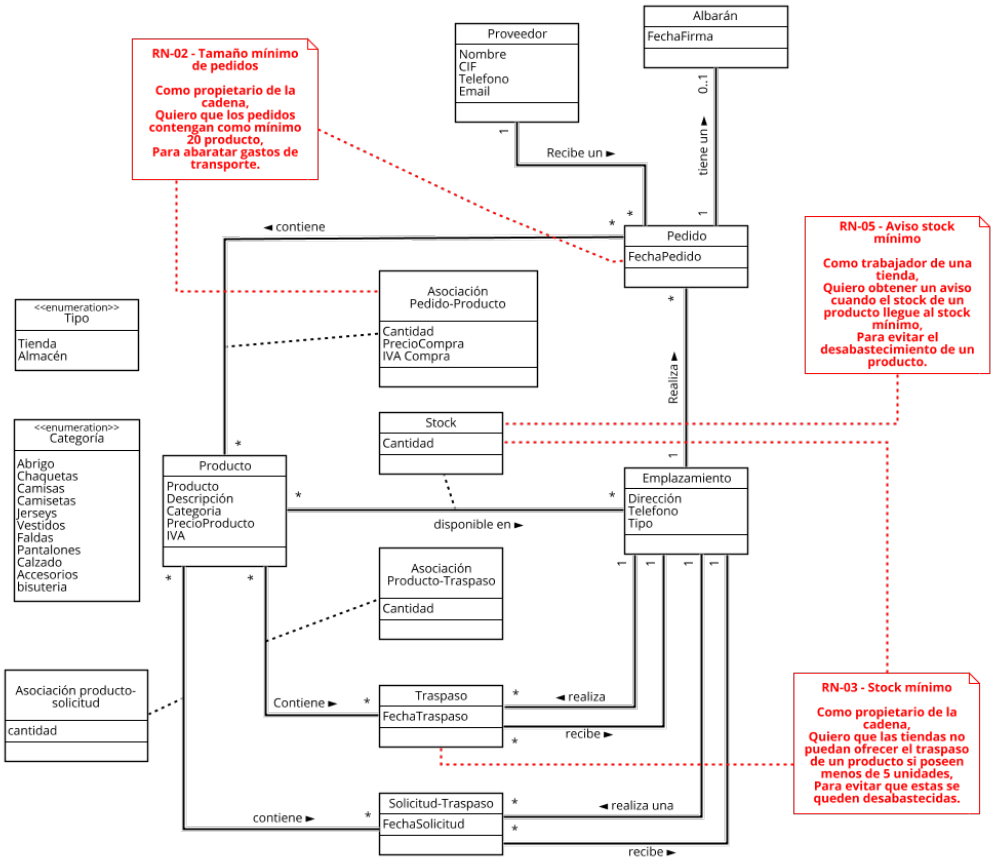


Figura 6: UML Traspaso / Pedido

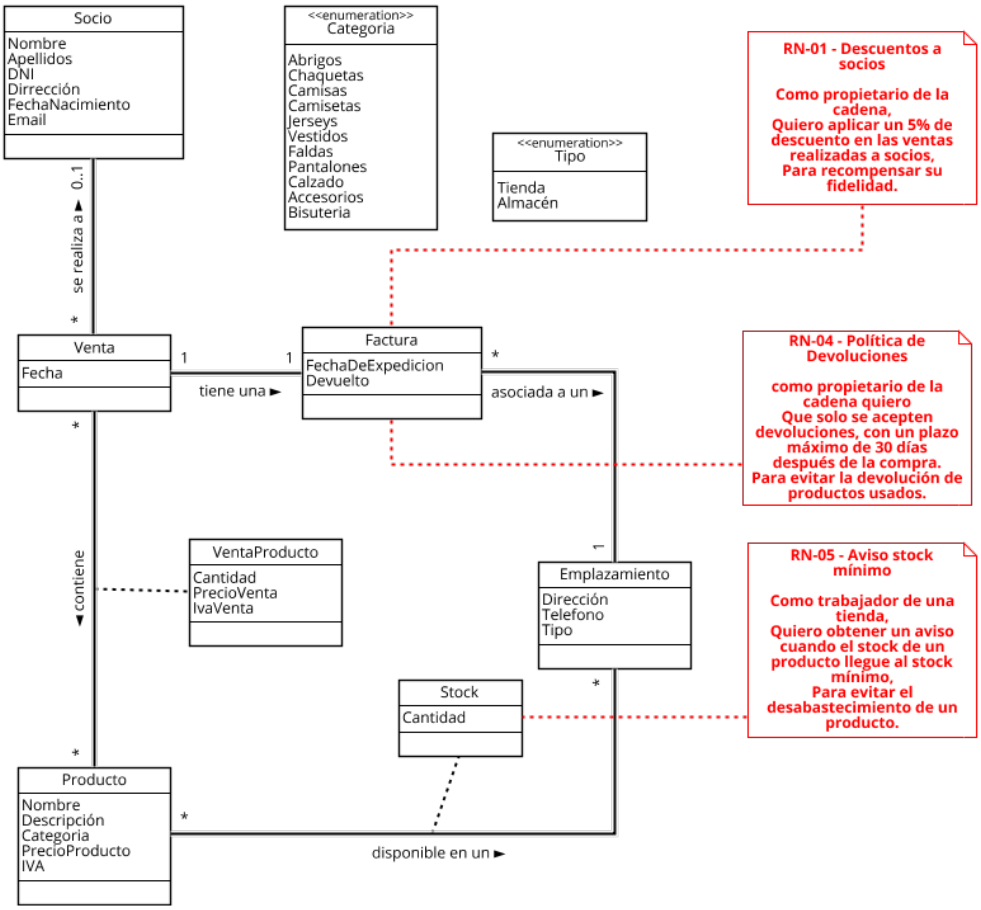


Figura 7: UML Venta

7.2. Escenarios de prueba

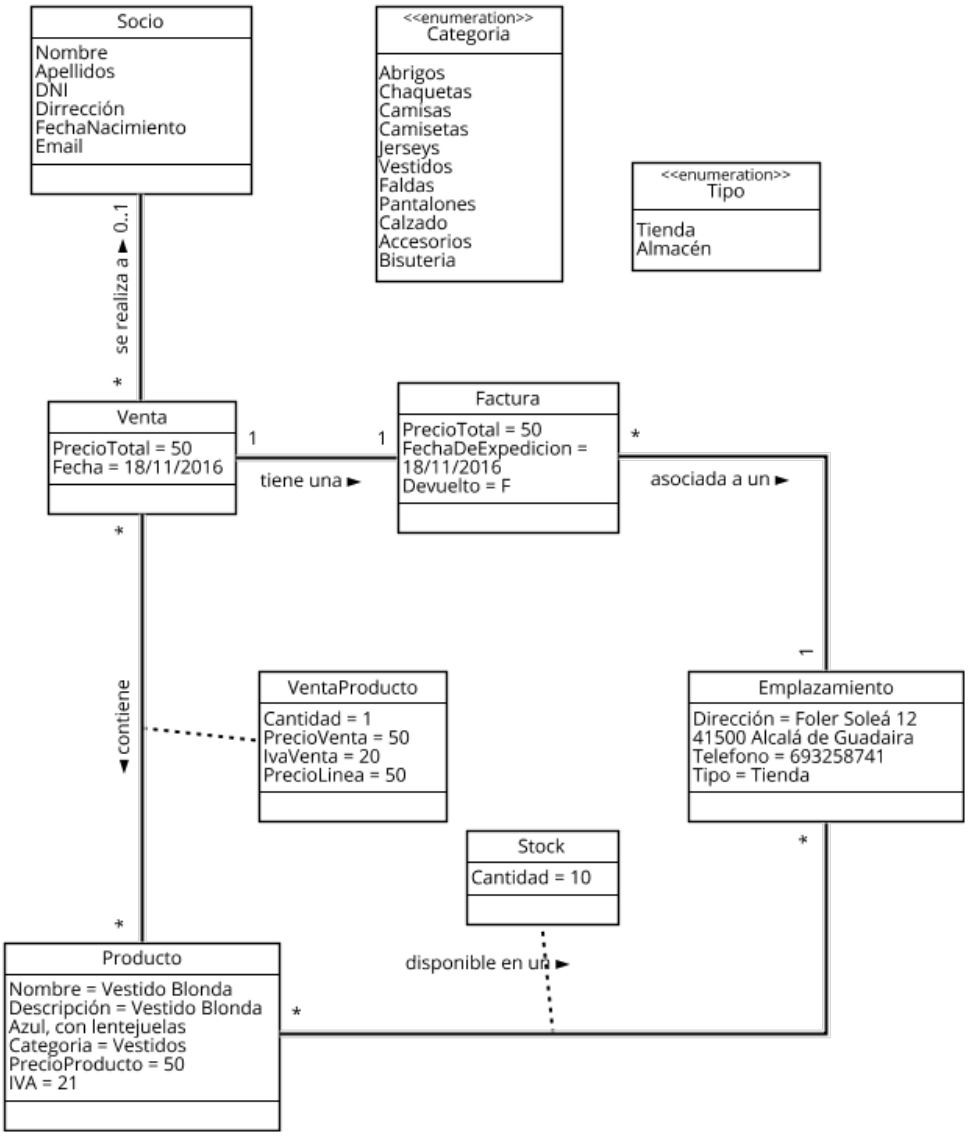


Figura 8: UML Venta NO Socio

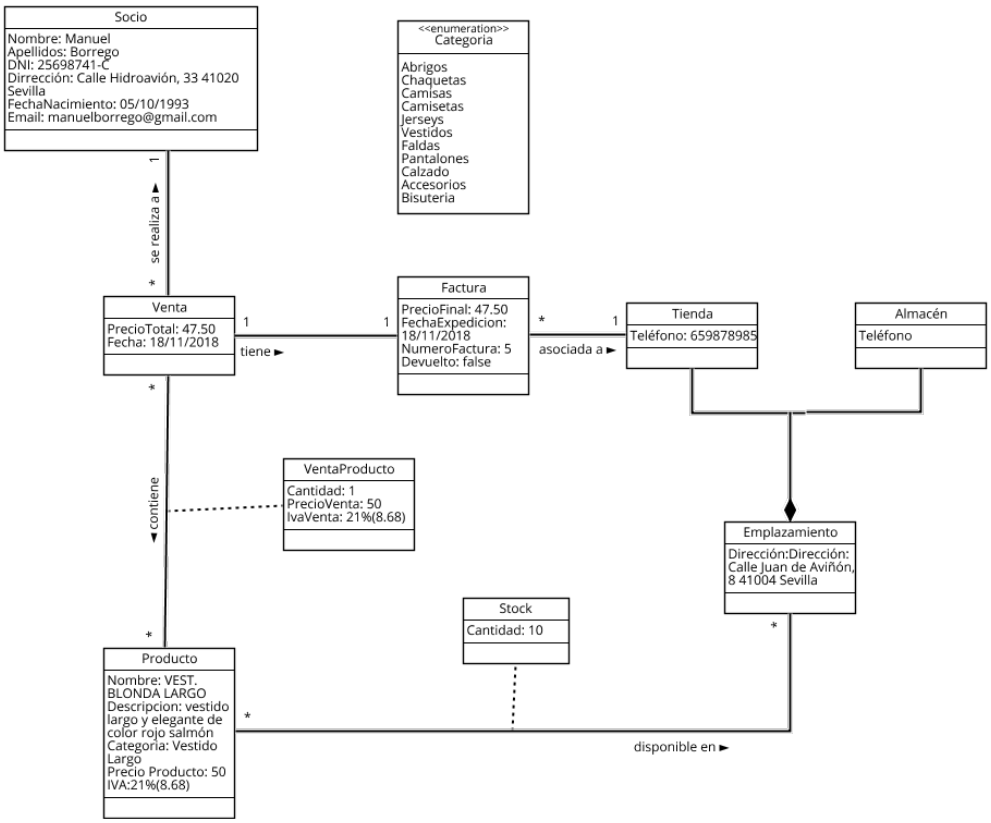


Figura 9: UML Venta Socio

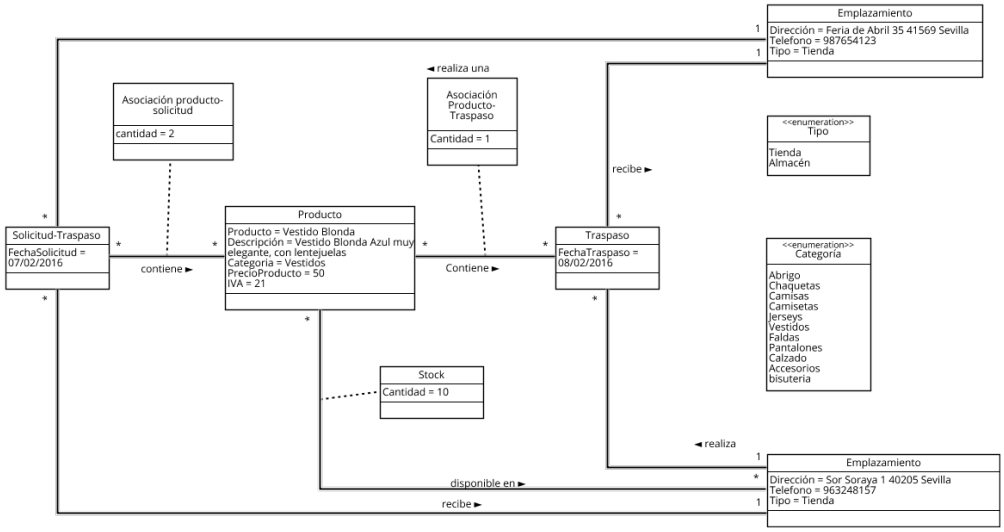


Figura 10: UML Traspaso Tienda - Tienda

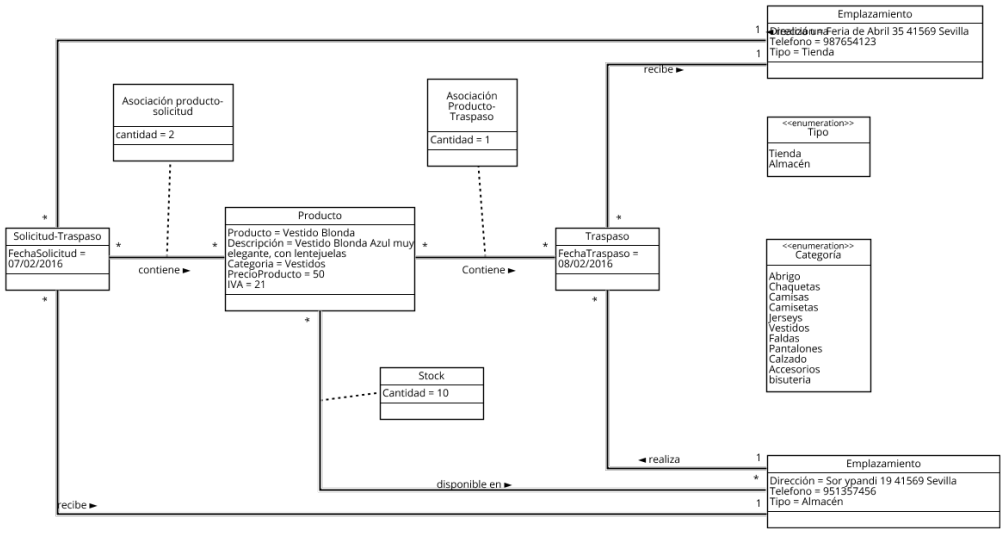


Figura 11: UML Traspaso Tienda - Almacén

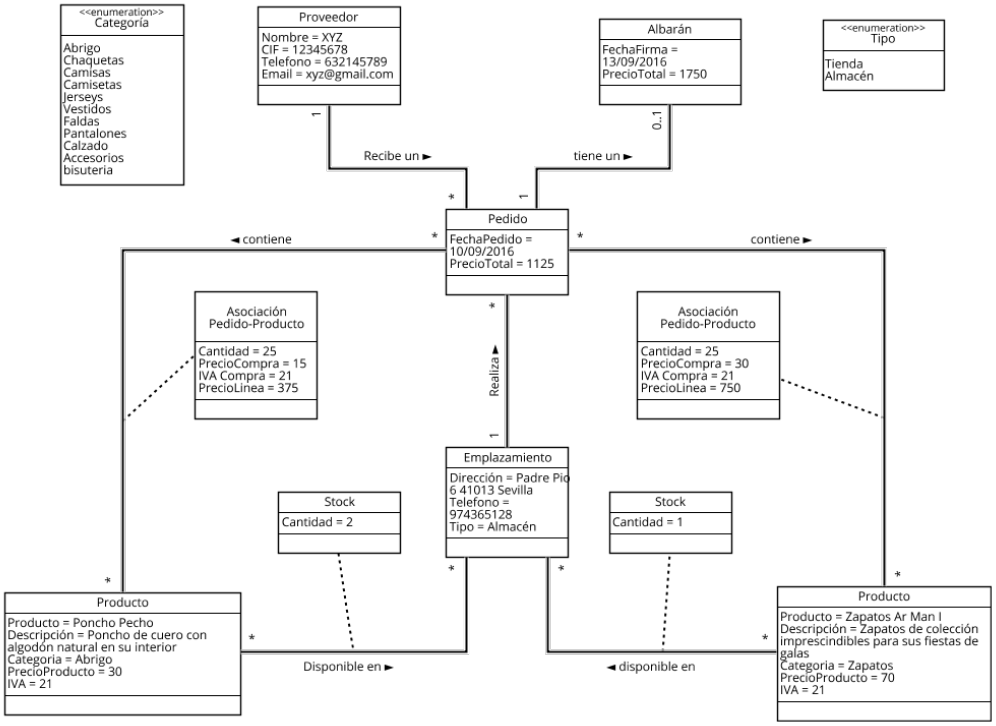


Figura 12: UML Pedido Tienda - Proveedor

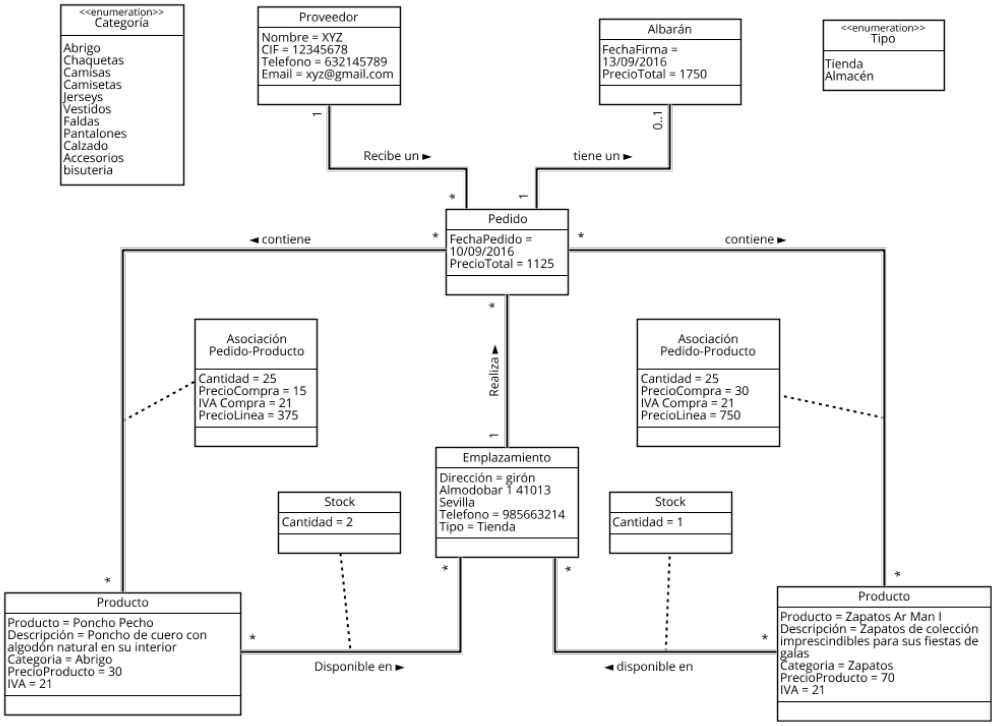


Figura 13: UML Pedido Almacén - Proveedor

8. Matrices de trazabilidad

8.1. Pruebas de aceptación frente a requisitos

Cuadro 1: **Requisitos funcionales**

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

Cuadro 2: **Requisitos de informacion**

PA-01	x										
PA-02		x									
PA-03			x								
PA-04				x							
PA-05					x						
PA-06							x				
PA-07								x		x	
PA-08									x		x
PA-09						x					
PA-10											
PA-11											
	01	**02**	**03**	**04**	**05**	**06**	**07**	**08**	**09**	**10**	**11**

Cuadro 3: **Reglas de negocio**

PA-01					x
PA-02	x			x	
PA-03	x				
PA-04		x		x	
PA-05			x		x
PA-06		x			
PA-07		x			x
PA-08			x		
PA-09			x	x	
PA-10	x				
PA-11		x			
	01	**02**	**03**	**04**	**05**

8.2. Pruebas de aceptación frente a escenarios de prueba

Cuadro 4: **Escenarios**

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

8.3. 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	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**

9. 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.

9.1. 3FN Venta

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

Venta Tiene una relación n - m con Producto, se crea una tabla intermedia debido a esta relación que es la tabla VentaProducto, con los siguientes

atributos: cantidad, precioVenta e ivaVenta. Además tiene ID_VENTA y ID_PRODUCTO como primary keys y foreigns keys, estas claves son provocadas por la relación n - m entre Venta y Producto.

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

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

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

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

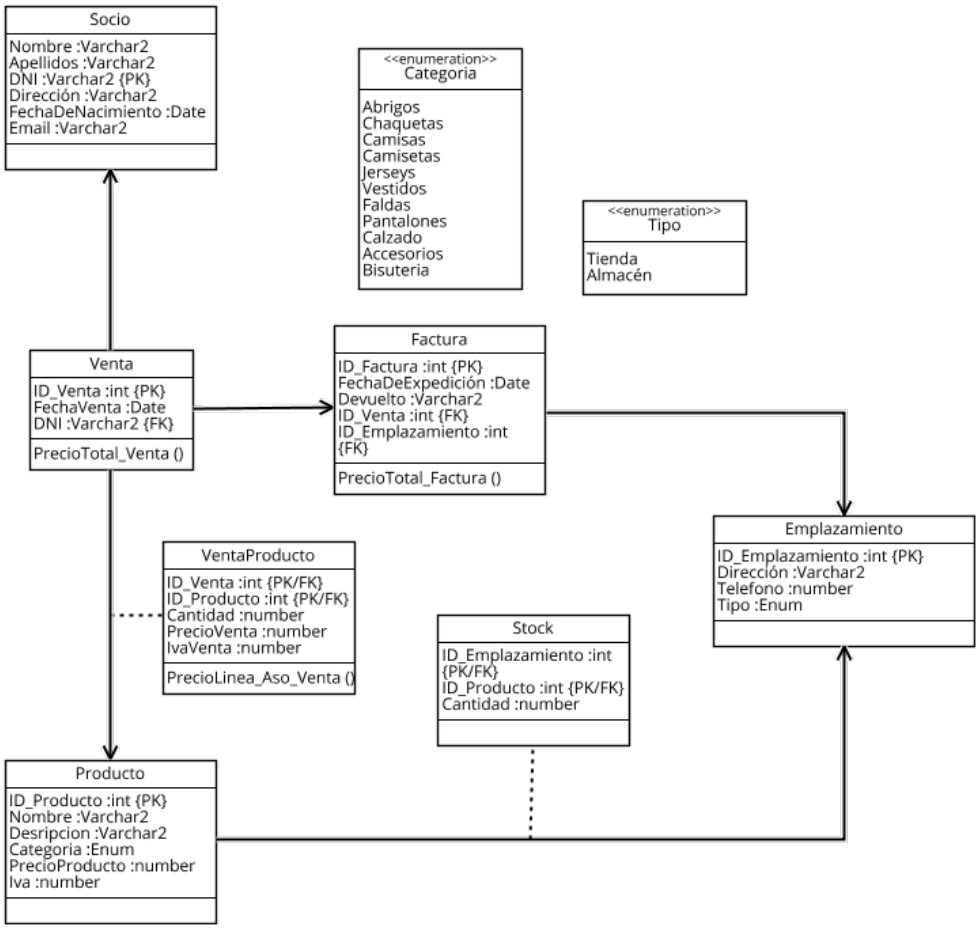


Figura 14: 3FN Venta

9.2. 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 primray keys y foreigns keys, estas claves son provocadas por la relación n - m entre Pedido y Producto.

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

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

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

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

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

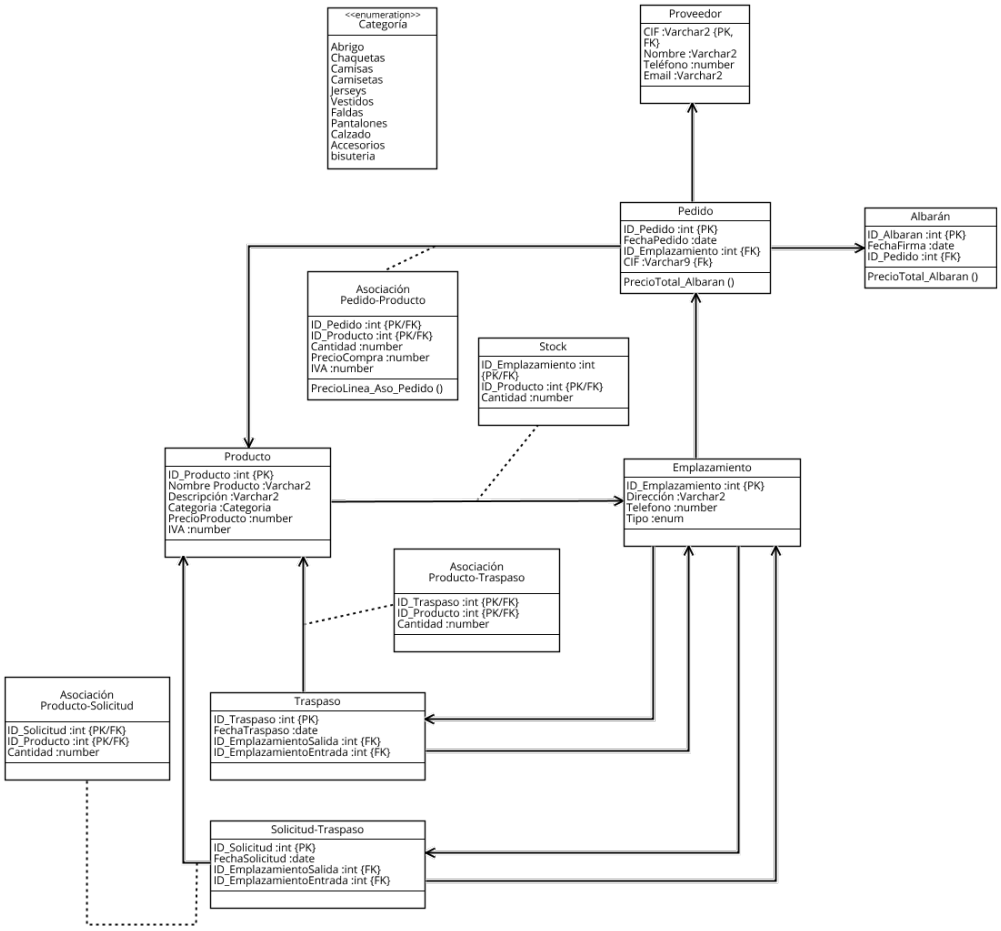


Figura 15: 3FN Traspaso - Pedido

10. Código SQL de la base de datos

10.1. Tablas

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

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


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

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

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

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

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

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

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


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

CREATE TABLE STOCK(
```

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

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

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

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



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

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

/* SECUENCIAS */

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

```
DROP SEQUENCE S_ID_Venta;



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

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

/

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

/

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

/

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

/

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

/

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

/

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

```

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

sql/tablas.sql

¡

10.2. Funciones y Procedures

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

/

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

/

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

/

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

/

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

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

/

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

/

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

/



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

/

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

/

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

/

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

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

/

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

/

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

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

/


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

/

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

/

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

/

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

```
END MODIFICA_PROVEEDOR_Telefono;

/

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

/

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

/

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

/

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

/

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

/

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

/

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

```
  COMMIT WORK;
END MODIFICA_SOCIO_EMAIL;

/

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

/

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

END MODIFICA_EMPLAZAMIENTO_TEL;

/

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

/

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

/

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

/

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

/

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

/

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

/


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

```
RETURN(f_PrecioLinea);
END precio_A_Venta_producto;

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

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

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

/
*/

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

/

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

/

```

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

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

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

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

/

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

/

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

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

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

```
509     f_precioTotal := (precio_AUX*m_cantidad) + f_precioTotal;
        END IF;
511   END LOOP;
      CLOSE all_prods;
513 RETURN(f_PrecioTotal);
    END precioTotal_Pedido;
515
    /
```

sql/funciones_y_procedures.sql

10.3. 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;
38  :NEW.FechaVenta := SYSDATE;
    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_preciotot_venta
108   BEFORE INSERT OR UPDATE ON ASOCIACION_VENTA_PRODUCTO
   for each row
110   begin
   UPDATE venta set preciotot = PRECIOTOTAL + (:New.cantidad * :New.precioVenta) where id_venta = :
      New.id_venta;
112 END inicializa_preciotot_venta;

114 /


116
CREATE OR REPLACE TRIGGER modifica_stock_pedido
118 BEFORE INSERT OR UPDATE ON ALBARAN
FOR EACH ROW
120 DECLARE
e_ID_Emplazamiento Emplazamiento.ID_Emplazamiento%TYPE;
122 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;
```

```
BEGIN

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

/

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

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

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

/


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

/

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

/

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

```

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

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

/

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

/

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

/

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

/

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

```
/

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

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

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

   num_lines := 3;

   dbms_output.get_lines(lines, num_lines);

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

sql/triggers.sql

10.4. Pruebas

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

/

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

/

/* DEFINICION PRUEBAS */
CREATE OR REPLACE PACKAGE PRUEBAS_SOCIO AS
    PROCEDURE inicializar;
    PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_nombre VARCHAR2, i_apellidos VARCHAR2,
```

```
        i_DNI VARCHAR2, i_direccion VARCHAR2, i_nacimiento DATE, i_email VARCHAR2
        ,salidaEsperada BOOLEAN);
    PROCEDURE actualizar
        (nombre_prueba VARCHAR2, a_DNI VARCHAR2, a_direccion VARCHAR2,
        a_email VARCHAR2, salidaEsperada BOOLEAN);
    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_DNI VARCHAR2, salidaEsperada BOOLEAN);
END PRUEBAS_SOCIO;

/

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

/

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

/

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

/

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

/

CREATE OR REPLACE PACKAGE PRUEBAS_PRODUCTO AS
    PROCEDURE inicializar;
    PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_nombre VARCHAR2, i_descripcion VARCHAR2,
        i_categoria VARCHAR2, i_precioProducto INT, i_iva INT, salidaEsperada BOOLEAN);
    PROCEDURE actualizar
        (nombre_prueba VARCHAR2, a_id_producto INT , a_descripcion VARCHAR2,
        a_precioProducto INT, a_iva INT, salidaEsperada BOOLEAN);
    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_producto INT, salidaEsperada BOOLEAN);
END PRUEBAS_PRODUCTO;

/

CREATE OR REPLACE PACKAGE PRUEBAS_VENTA AS
    PROCEDURE inicializar;
    PROCEDURE insertar
```

```
        (nombre_prueba VARCHAR2, i_fecha DATE, i_dni VARCHAR2,
        salidaEsperada BOOLEAN);
    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_venta INT, salidaEsperada BOOLEAN);
    /*procedure descuento
        (nombre_prueba VARCHAR2, v_id_venta int,v_preciototal number,v_dni_socio varchar2,
     salidaEsperada BOOLEAN);*/
END PRUEBAS_VENTA;

/

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

/

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

/

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

/

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

/

CREATE OR REPLACE PACKAGE PRUEBA_STOCK AS
    PROCEDURE inicializar;
    PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_id_emplazamiento INT, i_id_producto INT,
        i_cantidad INT, salidaEsperada BOOLEAN);
    PROCEDURE actualizar
        (nombre_prueba VARCHAR2, a_id_emplazamiento INT, a_id_producto INT,
        a_cantidad INT, salidaEsperada BOOLEAN);
    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,
        salidaEsperada BOOLEAN);
    PROCEDURE stock_correcto
        (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,stock_previo int, cantidad
      INT, salidaEsperada BOOLEAN);
    PROCEDURE stock_correcto_p
        (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,stock_previo int, cantidad
      INT, salidaEsperada BOOLEAN);
    PROCEDURE stock_correcto_t
```

```
        (nombre_prueba VARCHAR2, e_id_emplazamiento_envia int, e_id_emplazamiento_recibe int,
     stock_previo_envia int, stock_previo_recibe int,
178         cantidad int, e_id_producto int, salidaEsperada BOOLEAN);
END PRUEBA_STOCK;
180
/
182
CREATE OR REPLACE PACKAGE PRUEBAS_A_PRODUCTO_SOLICITUD AS
184     PROCEDURE inicializar;
    PROCEDURE insertar
186         (nombre_prueba VARCHAR2,i_id_producto INT, i_id_Solicitud INT,
            i_cantidad NUMBER, salidaEsperada BOOLEAN);
188     PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_Solicitud INT, e_id_producto INT, salidaEsperada BOOLEAN);
190 END PRUEBAS_A_PRODUCTO_SOLICITUD;

192 /

194 CREATE OR REPLACE PACKAGE PRUEBAS_A_PRODUCTO_TRASPASO AS
    PROCEDURE inicializar;
196     PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_id_producto INT, i_id_Traspaso INT,
198         i_cantidad NUMBER, salidaEsperada BOOLEAN);
    PROCEDURE eliminar
200         (nombre_prueba VARCHAR2, e_id_Traspaso INT, e_id_producto INT, salidaEsperada BOOLEAN);
END PRUEBAS_A_PRODUCTO_TRASPASO;
202
/
204
CREATE OR REPLACE PACKAGE PRUEBAS_A_PEDIDO_PRODUCTO AS
206     PROCEDURE inicializar;
    /*PROCEDURE insertar
208         (nombre_prueba VARCHAR2, i_id_pedido INT, i_id_producto INT,
        i_cantidad NUMBER, i_precioCompra number, i_iva INT,
210         salidaEsperada BOOLEAN);*/
    PROCEDURE eliminar
212         (nombre_prueba VARCHAR2, e_id_pedido INT, e_id_producto INT,
        salidaEsperada BOOLEAN);
214 END PRUEBAS_A_PEDIDO_PRODUCTO;

216 /
/*
218 CREATE OR REPLACE PACKAGE PRUEBAS_FUNCIONES AS
    PROCEDURE ganancias
220         (nombre_prueba VARCHAR2, mes NUMBER, anyo NUMBER, esperado number, salidaEsperada BOOLEAN);
    PROCEDURE precioLinea_A_Pedido
222         (nombre_prueba VARCHAR2, ID_PRODUCTO NUMBER, ID_PEDIDO NUMBER, esperado number,
     salidaEsperada BOOLEAN);
    PROCEDURE precioLinea_A_Venta
224         (nombre_prueba VARCHAR2, ID_PRODUCTO NUMBER, ID_VENTA NUMBER, esperado number, salidaEsperada
      BOOLEAN);
226 END PRUEBAS_FUNCIONES;
/
228 */
/* CUERPOS DE PRUEBAS */

230 CREATE OR REPLACE PACKAGE BODY PRUEBAS_SOCIO AS

232     PROCEDURE inicializar AS
    BEGIN
234         DELETE FROM SOCIO;
    END inicializar;
236
    PROCEDURE insertar
238         (nombre_prueba VARCHAR2, i_nombre VARCHAR2, i_apellidos VARCHAR2,
        i_DNI VARCHAR2, i_direccion VARCHAR2, i_nacimiento DATE, i_email VARCHAR2
240         ,salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
242     actual socio%ROWTYPE;
    BEGIN
244         INSERT INTO socio VALUES (i_nombre, i_apellidos, i_direccion, i_nacimiento,
        i_email, i_DNI);
246
        SELECT * INTO actual FROM SOCIO WHERE DNI = i_DNI;
248
        IF (actual.nombre<>i_nombre) OR (actual.apellidos<>i_apellidos) OR (actual.DNI<>i_DNI) OR (
     actual.direccion<>i_direccion) OR (actual.fechadenacimiento<>i_nacimiento) OR (actual.email<>
```

```
     i_email) THEN
250             salida := false;
        END IF;
252
        PRINTR(nombre_prueba, salida, salidaEsperada);
254
        EXCEPTION
256     WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
258         ROLLBACK;
    END insertar;
260
    PROCEDURE actualizar
262     (nombre_prueba VARCHAR2, a_DNI VARCHAR2, a_direccion VARCHAR2,
        a_email VARCHAR2, salidaEsperada BOOLEAN) AS
264     salida BOOLEAN := true;
        actual socio%ROWTYPE;
266     BEGIN
            UPDATE SOCIO SET DIRECCION = a_direccion, EMAIL = a_email where DNI = a_DNI;
268
            SELECT * INTO actual FROM SOCIO WHERE DNI = a_DNI;
270
            IF (actual.direccion<>a_direccion) OR (actual.email<>a_email) THEN
272             salida := false;
            END IF;
274
            PRINTR(nombre_prueba, salida, salidaEsperada);
276
            EXCEPTION
278         WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
280             ROLLBACK;
        END actualizar;
282
    PROCEDURE eliminar
284     (nombre_prueba VARCHAR2, e_DNI VARCHAR2, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
286     cantidad INT;
        BEGIN
288         DELETE FROM SOCIO WHERE DNI = e_DNI;

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

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

296         PRINTR(nombre_prueba, salida, salidaEsperada);

298         EXCEPTION
            WHEN OTHERS THEN
300             PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
302 END eliminar;
END PRUEBAS_SOCIO;
304
/
306
CREATE OR REPLACE PACKAGE BODY PRUEBAS_PROVEEDOR AS
308
    PROCEDURE inicializar AS
310 BEGIN
        DELETE FROM PROVEEDOR;
312 END inicializar;

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

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

324     IF (actual.nombre<>i_nombre) OR (actual.telefono<>i_telefono) OR (actual.CIF<>i_CIF) OR (
     actual.email<>i_email) THEN
```

```
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

            SELECT * INTO actual FROM PROVEEDOR WHERE CIF = a_cif;

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

            PRINTR(nombre_prueba, salida, salidaEsperada);

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

    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_CIF VARCHAR2, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
        cantidad INT;
        BEGIN
            DELETE FROM PROVEEDOR WHERE CIF = e_CIF;

            SELECT COUNT(*) INTO cantidad FROM PROVEEDOR WHERE CIF = e_cif;

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

            PRINTR(nombre_prueba, salida, salidaEsperada);

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

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_EMPLAZAMIENTO AS

    PROCEDURE inicializar AS

    BEGIN
        DELETE FROM EMPLAZAMIENTO;
    END inicializar;

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

```

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

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

        PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

            SELECT * INTO actual FROM EMPLAZAMIENTO WHERE ID_EMPLAZAMIENTO = a_id_emplazamiento;

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

            PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

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

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

            PRINTR(nombre_prueba, salida, salidaEsperada);

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

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_PRODUCTO AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM PRODUCTO;
    END inicializar;

    PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_nombre VARCHAR2, i_descripcion VARCHAR2,
        i_categoria VARCHAR2, i_precioProducto INT, i_iva INT, salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
    actual producto%ROWTYPE;
```

```
    w_ID_Producto INT;
476 BEGIN
        INSERT INTO producto VALUES(null,i_nombre, i_descripcion, i_categoria, i_precioProducto,
     i_iva);
478
        w_ID_Producto := S_ID_Producto.currval;
480     SELECT * INTO actual FROM PRODUCTO WHERE ID_Producto = w_ID_Producto;

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

486     PRINTR(nombre_prueba, salida, salidaEsperada);

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

494 PROCEDURE actualizar
        (nombre_prueba VARCHAR2, a_id_producto INT , a_descripcion VARCHAR2,
496     a_precioProducto INT, a_iva INT, salidaEsperada BOOLEAN)AS
        salida BOOLEAN := true;
498     actual producto%ROWTYPE;
        BEGIN
500         UPDATE producto SET DESCRIPCION = a_descripcion, PRECIOPRODUCTO = a_precioProducto, IVA =
      a_iva
            where ID_Producto = a_id_producto;
502
            SELECT * INTO actual FROM PRODUCTO WHERE ID_Producto = a_id_producto;
504         IF (actual.descripcion<>a_descripcion) OR
                (actual.precioProducto<>a_precioProducto) OR
506             (actual.iva<>a_iva) THEN
                salida := false;
508         END IF;

510         PRINTR(nombre_prueba, salida, salidaEsperada);

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

518 PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_producto INT, salidaEsperada BOOLEAN) AS
520     salida BOOLEAN := true;
        cantidad INT;
522     BEGIN
            DELETE FROM PRODUCTO  WHERE ID_Producto = e_id_producto;
524
            SELECT COUNT(*) INTO cantidad FROM PRODUCTO WHERE ID_Producto = e_id_producto;
526         IF (cantidad<>0) THEN
                salida := false;
528         END IF;

530         PRINTR(nombre_prueba, salida, salidaEsperada);

532         EXCEPTION
            WHEN OTHERS THEN
534             PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
536 END eliminar;
END PRUEBAS_PRODUCTO;
538
/
540
CREATE OR REPLACE PACKAGE BODY PRUEBAS_FACTURA AS
542
    PROCEDURE inicializar AS
544 BEGIN
        DELETE FROM FACTURA;
546 END inicializar;

548 PROCEDURE insertar
```

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

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

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

        PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

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

            PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

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

            PRINTR(nombre_prueba, salida, salidaEsperada);

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

/

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

    PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_id_emplazamiento INT, i_id_producto INT,
```

```
624         i_cantidad INT, salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
626     actual stock%ROWTYPE;
    BEGIN
628         INSERT INTO stock VALUES(i_id_emplazamiento, i_id_producto, i_cantidad);

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

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

636         PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

652             SELECT * INTO actual FROM stock WHERE  ID_Emplazamiento = a_id_emplazamiento and
     ID_Producto = a_id_producto;
                IF (actual.cantidad<>a_cantidad) THEN
654                 salida := false;
                END IF;
656
                PRINTR(nombre_prueba, salida, salidaEsperada);
658
                EXCEPTION
660             WHEN OTHERS THEN
                    PRINTR(nombre_prueba, false, salidaEsperada);
662                 ROLLBACK;
            END actualizar;
664
    PROCEDURE eliminar
666         (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,
            salidaEsperada BOOLEAN) AS
668         salida BOOLEAN := true;
            cantidad INT;
670         BEGIN
                DELETE FROM stock  WHERE ID_Emplazamiento = e_id_emplazamiento and ID_Producto =
     e_id_producto;
672
                SELECT COUNT(*) INTO cantidad FROM stock WHERE ID_Emplazamiento = e_id_emplazamiento and
     ID_Producto = e_id_producto;
674             IF (cantidad<>0) THEN
                    salida := false;
676             END IF;

678             PRINTR(nombre_prueba, salida, salidaEsperada);

680             EXCEPTION
                WHEN OTHERS THEN
682                 PRINTR(nombre_prueba, false, salidaEsperada);
                    ROLLBACK;
684         END eliminar;

686     PROCEDURE stock_correcto
            (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,stock_previo int, cantidad
      INT, salidaEsperada BOOLEAN) AS
688         salida BOOLEAN := true;
            stock_nuevo int;
690         BEGIN
              select cantidad into stock_nuevo from stock where id_emplazamiento = e_id_emplazamiento and
      id_producto= e_id_producto;
692             if(stock_previo-cantidad)<> stock_nuevo then
```

```
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

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

    END stock_correcto;

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

        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
   end stock_correcto_p;

   PROCEDURE stock_correcto_t
    (nombre_prueba VARCHAR2, e_id_emplazamiento_envia int, e_id_emplazamiento_recibe int,
  stock_previo_envia int, stock_previo_recibe int,
    cantidad int, e_id_producto int, salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
    stock_nuevo_envia int;
    stock_nuevo_recibe int;
    BEGIN
      select cantidad into stock_nuevo_envia from stock where id_emplazamiento =
  e_id_emplazamiento_envia and id_producto = e_id_producto;
      select cantidad into stock_nuevo_recibe from stock where id_emplazamiento =
  e_id_emplazamiento_recibe and id_producto = e_id_producto;
        if(stock_previo_envia-cantidad)<>stock_nuevo_envia or (stock_previo_recibe+cantidad)<>
  stock_nuevo_recibe then
            salida := false;
        END IF;
        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
    end stock_correcto_t;

END PRUEBA_STOCK;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_VENTA AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM VENTA;
    END inicializar;

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

        w_ID_Venta := S_ID_Venta.currval;
```

```
764         SELECT * INTO actual FROM venta WHERE ID_Venta = w_ID_Venta;

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

770         PRINTR(nombre_prueba, salida, salidaEsperada);

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

778     PROCEDURE eliminar
            (nombre_prueba VARCHAR2, e_id_venta INT, salidaEsperada BOOLEAN) AS
780         salida BOOLEAN := true;
            cantidad INT;
782         BEGIN
                DELETE FROM venta WHERE ID_Venta = e_id_venta;
784
                SELECT COUNT(*) INTO cantidad FROM venta WHERE ID_Venta = e_id_venta;
786             IF (cantidad<>0) THEN
                    salida := false;
788             END IF;

790             PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

818 /

820 CREATE OR REPLACE PACKAGE BODY PRUEBAS_PEDIDO AS

822     PROCEDURE inicializar AS
        BEGIN
824         DELETE FROM pedido;
        END inicializar;
826
        PROCEDURE insertar
828       (nombre_prueba VARCHAR2, i_fecha DATE,
          i_id_emplazamiento INT, i_cif VARCHAR2, salidaEsperada BOOLEAN) AS
830     salida BOOLEAN := true;
        actual pedido%ROWTYPE;
832     w_ID_pedido INT;
        BEGIN
834         INSERT INTO pedido VALUES(null, i_fecha, i_id_emplazamiento, i_cif);

836         w_ID_pedido := S_ID_Pedido.currval;
            SELECT * INTO actual FROM pedido WHERE ID_Pedido = w_ID_pedido;
838
```

```
        IF (actual.fechaPedido<>i_fecha) OR (actual.id_emplazamiento<>i_id_emplazamiento) OR (actual.
    cif<>i_cif) THEN
            salida := false;
        END IF;

        PRINTR(nombre_prueba, salida, salidaEsperada);

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

    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_pedido INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
        cantidad INT;
        BEGIN
            DELETE FROM pedido  WHERE ID_Pedido = e_id_pedido;

            SELECT COUNT(*) INTO cantidad FROM pedido WHERE ID_Pedido = e_id_pedido;
            IF (cantidad<>0) THEN
                salida := false;
            END IF;

            PRINTR(nombre_prueba, salida, salidaEsperada);

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

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_TRASPASO AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM traspaso;
    END inicializar;

    PROCEDURE insertar
       (nombre_prueba VARCHAR2, i_fechaTraspaso DATE, i_id_emplazamientoSalida INT,
        i_id_emplazamientoEntrada INT, salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
    actual traspaso%ROWTYPE;
    w_ID_traspaso INT;
    BEGIN
        INSERT INTO traspaso VALUES(null, i_fechatraspaso, i_id_emplazamientoSalida,
     i_id_emplazamientoEntrada);

        w_ID_traspaso := S_ID_traspaso.currval;
        SELECT * INTO actual FROM traspaso WHERE ID_traspaso = w_ID_traspaso;

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

        PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

            SELECT COUNT(*) INTO cantidad FROM traspaso WHERE ID_traspaso = e_id_traspaso;
```

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

            PRINTR(nombre_prueba, salida, salidaEsperada);

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

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_Solicitud_Traspaso AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM Solicitud_Traspaso;
    END inicializar;

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

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

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

        PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

            SELECT COUNT(*) INTO cantidad FROM Solicitud_Traspaso WHERE ID_Solicitud =
     e_id_solicitudTraspaso;
            IF (cantidad<>0) THEN
                salida := false;
            END IF;

            PRINTR(nombre_prueba, salida, salidaEsperada);

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

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_Albaran AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM Albaran;
```

```
    END inicializar;
988
    PROCEDURE insertar
990    (nombre_prueba VARCHAR2, i_fecha DATE,
        i_id_pedido INT, salidaEsperada BOOLEAN) AS
992 salida BOOLEAN := true;
    actual Albaran%ROWTYPE;
994 w_ID_Albaran INT;
    BEGIN
996     INSERT INTO Albaran VALUES(null, i_fecha, i_id_pedido);

998     w_ID_Albaran := S_ID_Albaran.currval;
        SELECT * INTO actual FROM Albaran WHERE ID_Albaran = w_ID_Albaran;
1000
        IF (actual.fechaFirma<>i_fecha) OR (actual.id_pedido<>i_id_pedido) THEN
1002        salida := false;
        END IF;
1004
        PRINTR(nombre_prueba, salida, salidaEsperada);
1006
        EXCEPTION
1008    WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
1010        ROLLBACK;
    END insertar;
1012
    PROCEDURE eliminar
1014    (nombre_prueba VARCHAR2, e_id_albaran INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
1016    cantidad INT;
        BEGIN
1018        DELETE FROM Albaran  WHERE ID_Albaran = e_id_albaran;

1020        SELECT COUNT(*) INTO cantidad FROM Albaran WHERE ID_Albaran = e_id_albaran;
            IF (cantidad<>0) THEN
1022            salida := false;
            END IF;
1024
            PRINTR(nombre_prueba, salida, salidaEsperada);
1026
            EXCEPTION
1028        WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
1030            ROLLBACK;
    END eliminar;
1032 END PRUEBAS_Albaran;

1034 /

1036 CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_PRODUCTO_TRASPASO AS

1038    PROCEDURE inicializar AS
        BEGIN
1040        DELETE FROM asociacion_Producto_traspaso;
    END inicializar;
1042
    PROCEDURE insertar
1044        (nombre_prueba VARCHAR2, i_id_producto INT, i_id_Traspaso INT,
                i_cantidad NUMBER, salidaEsperada BOOLEAN) AS
1046    salida BOOLEAN := true;
    actual asociacion_Producto_traspaso%ROWTYPE;
1048    BEGIN
        INSERT INTO asociacion_Producto_traspaso VALUES(i_id_Traspaso, i_id_producto, i_cantidad);
1050
        SELECT * INTO actual FROM asociacion_Producto_traspaso WHERE ID_Producto = i_id_producto and
     ID_Traspaso = i_id_Traspaso;
1052
        IF (actual.id_producto<>i_id_producto) OR (actual.id_Traspaso<>i_id_Traspaso) OR (actual.
     cantidad<>i_cantidad) THEN
1054        salida := false;
        END IF;
1056
        PRINTR(nombre_prueba, salida, salidaEsperada);
1058
        EXCEPTION
1060    WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
```

```
            ROLLBACK;
    END insertar;

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

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

            PRINTR(nombre_prueba, salida, salidaEsperada);

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

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_PRODUCTO_SOLICITUD AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM asociacion_Producto_Solicitud;
    END inicializar;

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

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

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

        PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

            SELECT COUNT(*) INTO cantidad FROM asociacion_Producto_Solicitud WHERE ID_Producto =
     e_id_producto and ID_Solicitud = e_id_Solicitud;
            IF (cantidad<>0) THEN
                salida := false;
            END IF;

            PRINTR(nombre_prueba, salida, salidaEsperada);

            EXCEPTION
            WHEN OTHERS THEN
```

```
                PRINTR(nombre_prueba, false, salidaEsperada);
                ROLLBACK;
    END eliminar;
END PRUEBAS_A_PRODUCTO_SOLICITUD;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_PEDIDO_PRODUCTO AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM asociacion_Pedido_Producto;
    END inicializar;
  /*
    PROCEDURE insertar
      (nombre_prueba VARCHAR2, i_id_pedido INT, i_id_producto INT,
      i_cantidad NUMBER, i_precioCompra number, i_iva INT,
      salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
    actual asociacion_Pedido_Producto%ROWTYPE;
    BEGIN
        INSERT INTO asociacion_Pedido_Producto VALUES(i_id_pedido, i_id_producto, i_cantidad,
     i_precioCompra, i_iva,);

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

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

        PRINTR(nombre_prueba, salida, salidaEsperada);

        EXCEPTION
        WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
    END insertar;
    */
    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_pedido INT, e_id_producto INT,
        salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
        cantidad INT;
        BEGIN
            DELETE FROM asociacion_Pedido_Producto  WHERE ID_pedido = e_id_pedido and ID_producto =
     e_id_producto;

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

            PRINTR(nombre_prueba, salida, salidaEsperada);

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

END PRUEBAS_A_PEDIDO_PRODUCTO;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_VENTA_PRODUCTO AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM asociacion_Venta_Producto;
    END inicializar;

    PROCEDURE insertar
    (nombre_prueba VARCHAR2, i_id_venta INT, i_id_producto INT,
        i_cantidad NUMBER, i_precioVenta NUMBER, i_ivaVenta NUMBER,
```

```
        salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
    actual asociacion_Venta_Producto%ROWTYPE;
    BEGIN
        INSERT INTO asociacion_Venta_Producto VALUES(i_id_venta, i_id_producto, i_cantidad,
     i_precioVenta, i_ivaVenta);

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

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

        PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

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

            PRINTR(nombre_prueba, salida, salidaEsperada);

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

/


/*
CREATE OR REPLACE PACKAGE BODY PRUEBAS_FUNCIONES AS
  PROCEDURE ganancias
    (nombre_prueba VARCHAR2, mes NUMBER, anyo NUMBER, esperado number, salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
    ganancia number;
    BEGIN
        SELECT ganancias_mensuales(mes,anyo) INTO ganancia FROM dual;

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

        PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

```
1276
        IF (precio<>esperado) THEN
1278        salida := false;
        END IF;
1280
        PRINTR(nombre_prueba, salida, salidaEsperada);
1282
        EXCEPTION
1284    WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
1286        ROLLBACK;
    END PRECIOLINEA_A_PEDIDO;
1288
    PROCEDURE precioLinea_A_Venta
1290    (nombre_prueba VARCHAR2, ID_PRODUCTO NUMBER, ID_VENTA NUMBER, esperado number, salidaEsperada
      BOOLEAN) AS
        salida BOOLEAN := true;
1292 precio number;
    BEGIN
1294    SELECT precioLinea_Aso_VENTA(ID_PRODUCTO,ID_VENTA) INTO precio FROM dual;

1296    IF (precio<>esperado) THEN
            salida := false;
1298    END IF;

1300    PRINTR(nombre_prueba, salida, salidaEsperada);

1302    EXCEPTION
        WHEN OTHERS THEN
1304        PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
1306 END PRECIOLINEA_A_VENTA;

1308 END PRUEBAS_FUNCIONES;
/
1310 */
DROP SEQUENCE S_ID_Producto;
1312 DROP SEQUENCE S_ID_Traspaso;
DROP SEQUENCE S_ID_Solicitud;
1314 DROP SEQUENCE S_ID_Pedido;
DROP SEQUENCE S_ID_Albaran;
1316 DROP SEQUENCE S_ID_Factura;
DROP SEQUENCE S_ID_Emplazamiento;
1318 DROP SEQUENCE S_ID_Venta;

1320 CREATE SEQUENCE S_ID_Emplazamiento START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Venta START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1322 CREATE SEQUENCE S_ID_Pedido START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Albaran START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1324 CREATE SEQUENCE S_ID_Producto START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Traspaso START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1326 CREATE SEQUENCE S_ID_Solicitud START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
CREATE SEQUENCE S_ID_Factura START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1328
/* EXECUTE DE PRUEBAS */
1330 -- Inicializacion
EXECUTE pruebas_albaran.inicializar();
1332 EXECUTE PRUEBAS_A_PRODUCTO_TRASPASO.inicializar();
EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.inicializar();
1334 EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.inicializar();
EXECUTE PRUEBAS_A_VENTA_PRODUCTO.inicializar();
1336 EXECUTE PRUEBAS_FACTURA.inicializar();
EXECUTE pruebas_venta.inicializar();
1338 EXECUTE pruebas_pedido.inicializar();
EXECUTE pruebas_proveedor.inicializar();
1340 EXECUTE PRUEBA_STOCK.inicializar();
EXECUTE pruebas_producto.inicializar();
1342 EXECUTE PRUEBAS_TRASPASO.inicializar();
EXECUTE PRUEBAS_SOLICITUD_TRASPASO.inicializar();
1344 EXECUTE PRUEBAS_SOCIO.inicializar();
EXECUTE pruebas_emplazamiento.inicializar();
1346 --SOCIO
EXECUTE PRUEBAS_SOCIO.inicializar();
1348 EXECUTE PRUEBAS_SOCIO.insertar('Socio Insertar', 'Daniel', 'Gonzalez', '54148787G',  'Avenida del
     pepinillo', TO_DATE('10102015','DDMMYYYY'), 'dani@us.es', true);
EXECUTE PRUEBAS_SOCIO.insertar('Socio Insertar', 'Jose Luis', 'Marmol Romero', '29613400A',  'Calle
     Virgen', TO_DATE('10101996','DDMMYYYY'), 'jlmr@us.es', true);
```

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

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

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

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

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

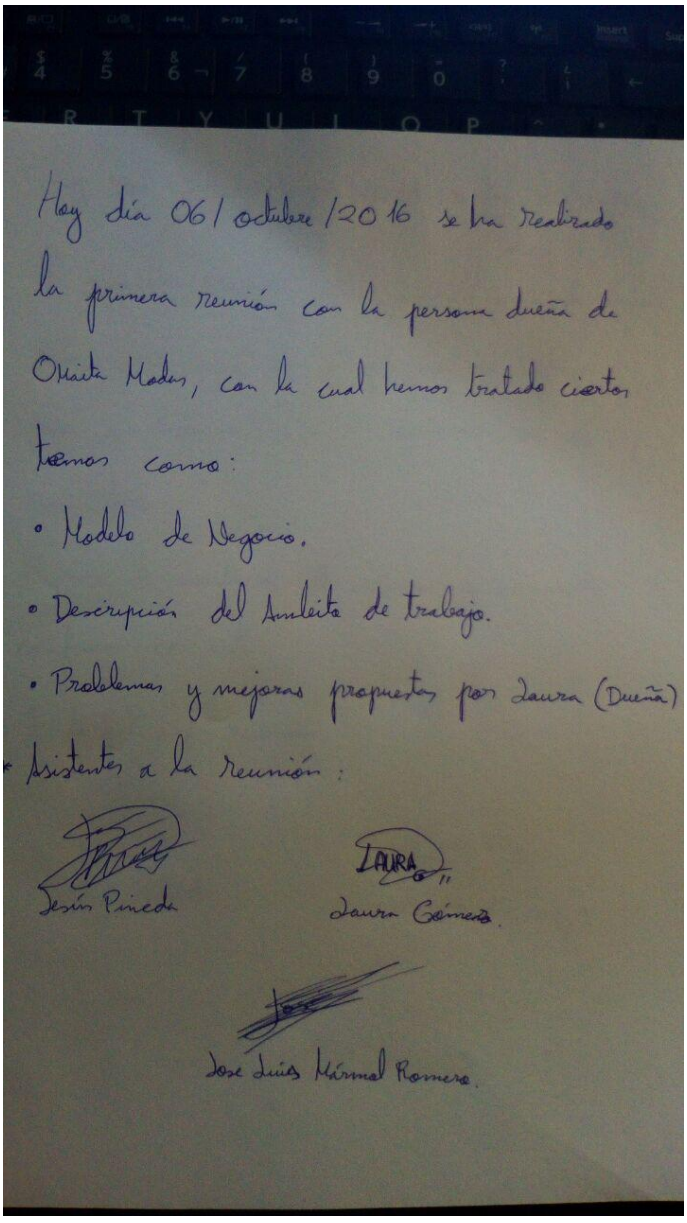
```
--VENTA
EXECUTE pruebas_venta.inicializar();
EXECUTE pruebas_venta.insertar('Venta insercion',sysdate,'29613400A',true);
EXECUTE pruebas_venta.insertar('Venta insercion',sysdate,null,true);
EXECUTE pruebas_venta.insertar('Venta insercion',sysdate,null,true);
EXECUTE pruebas_venta.eliminar('Venta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

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

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

--PEDIDO_PRODUCTO

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

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

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

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

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

--ASOC_PRODUCTO_TRASPASO
EXECUTE PRUEBAS_A_PRODUCTO_TRASPASO.inicializar();
EXECUTE PRUEBAS_A_PRODUCTO_TRASPASO.insertar('A-Producto-Traspaso Insertar', S_ID_Producto.currval-1,
      S_ID_traspaso.currval-1, 4, true);
EXECUTE PRUEBA_STOCK.STOCK_CORRECTO_T('Actualizacion de stock tras traspaso',s_ID_emplazamiento.
     currval-1, s_ID_emplazamiento.currval-2,33,29,4,s_ID_producto.currval-1,true);
EXECUTE PRUEBAS_A_PRODUCTO_TRASPASO.eliminar('A-Producto-Traspaso Eliminar', S_ID_traspaso.currval-1,
      S_ID_Producto.currval-1, true);
--Pruebas de funciones
/*
EXECUTE PRUEBAS_FUNCIONES.ganancias('Funcion Ganancias mensuales',6,2017,-1008.87,true);
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);
*/
--Tiene que dar -1008.87
--select ganancias_mensuales(6,2017) from dual;
-- Tiene que dar 220.5
--SELECT precioLinea_Aso_Pedido(9,2) FROM DUAL;
--Tienda que dar 93.9
--SELECT precioLinea_Aso_Venta(9,2) FROM DUAL;
-- Select que muestra los productos ofrecidos y disponibles
--SELECT nombre,id_emplazamiento from producto inner join stock on stock.ID_PRODUCTO = producto.
     id_producto;
```

sql/pruebas.sql

11. Apéndice

11.1. Actas de reunión



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

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

+ Asistentes a la reunión:

Jesús Pineda

Laura Gómez.

José Luis Mármol Romero.

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

Después de mostrarle el avance se encuentra conforme con este.

* Asistentes a la reunión:

Jesús Pineda

LAURA G.

Laura Gómez