Universidad de Sevilla

IISSI

Proyecto

Modas Omaita

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

14 de mayo 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

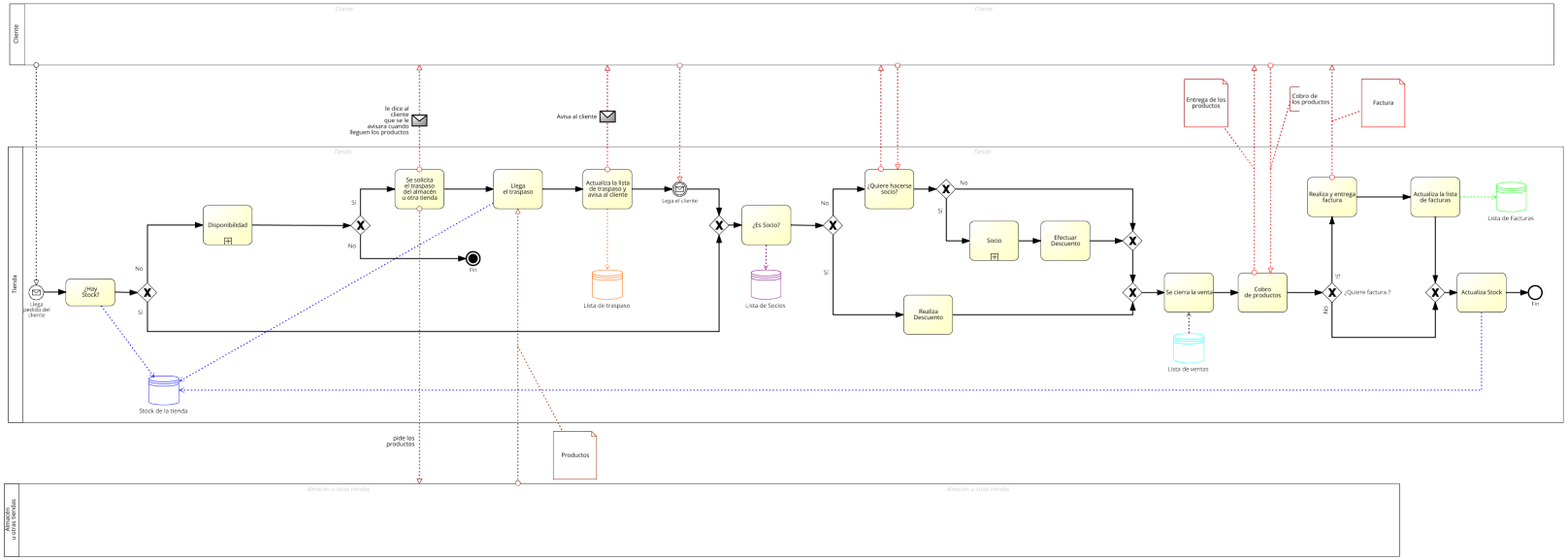
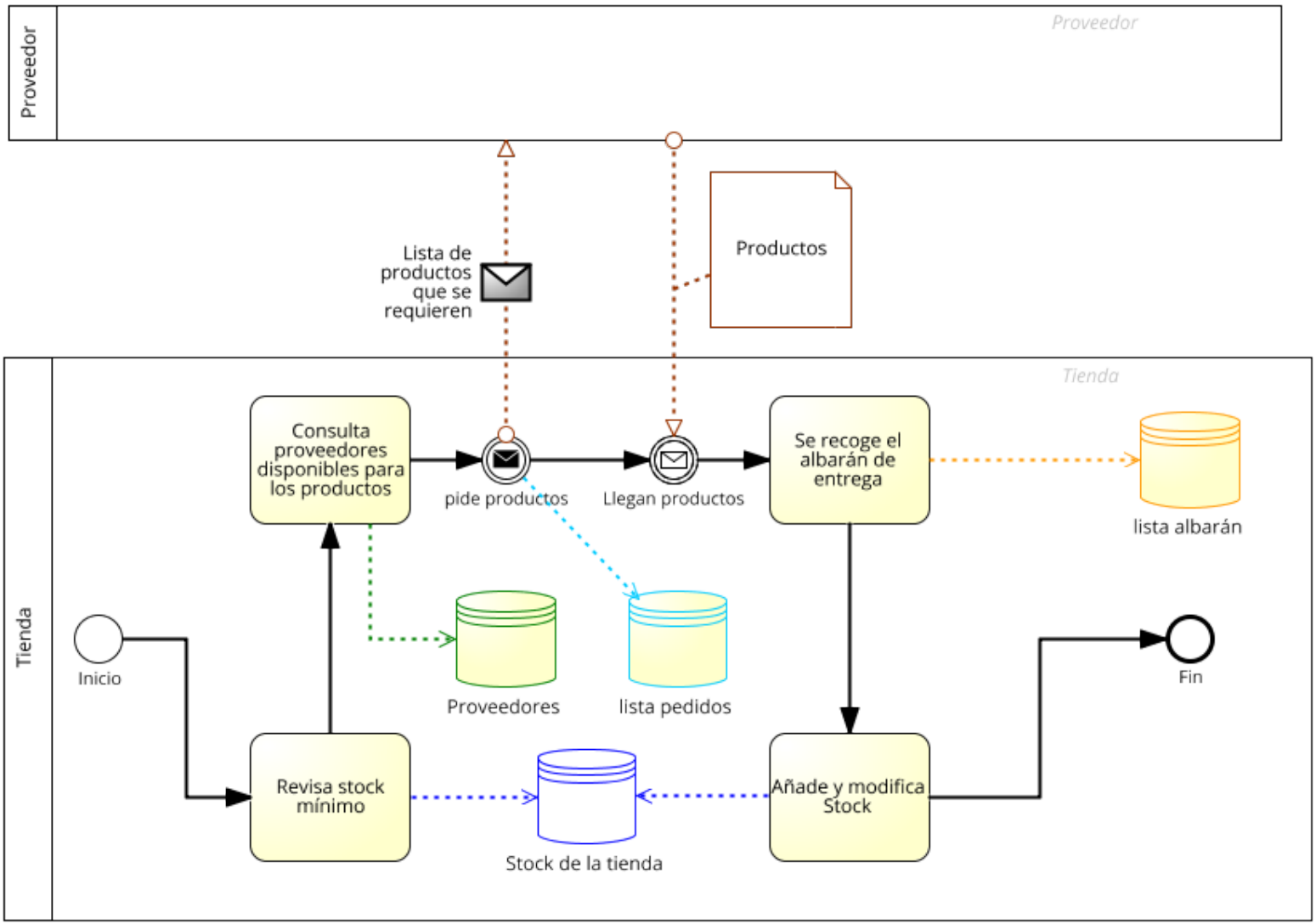
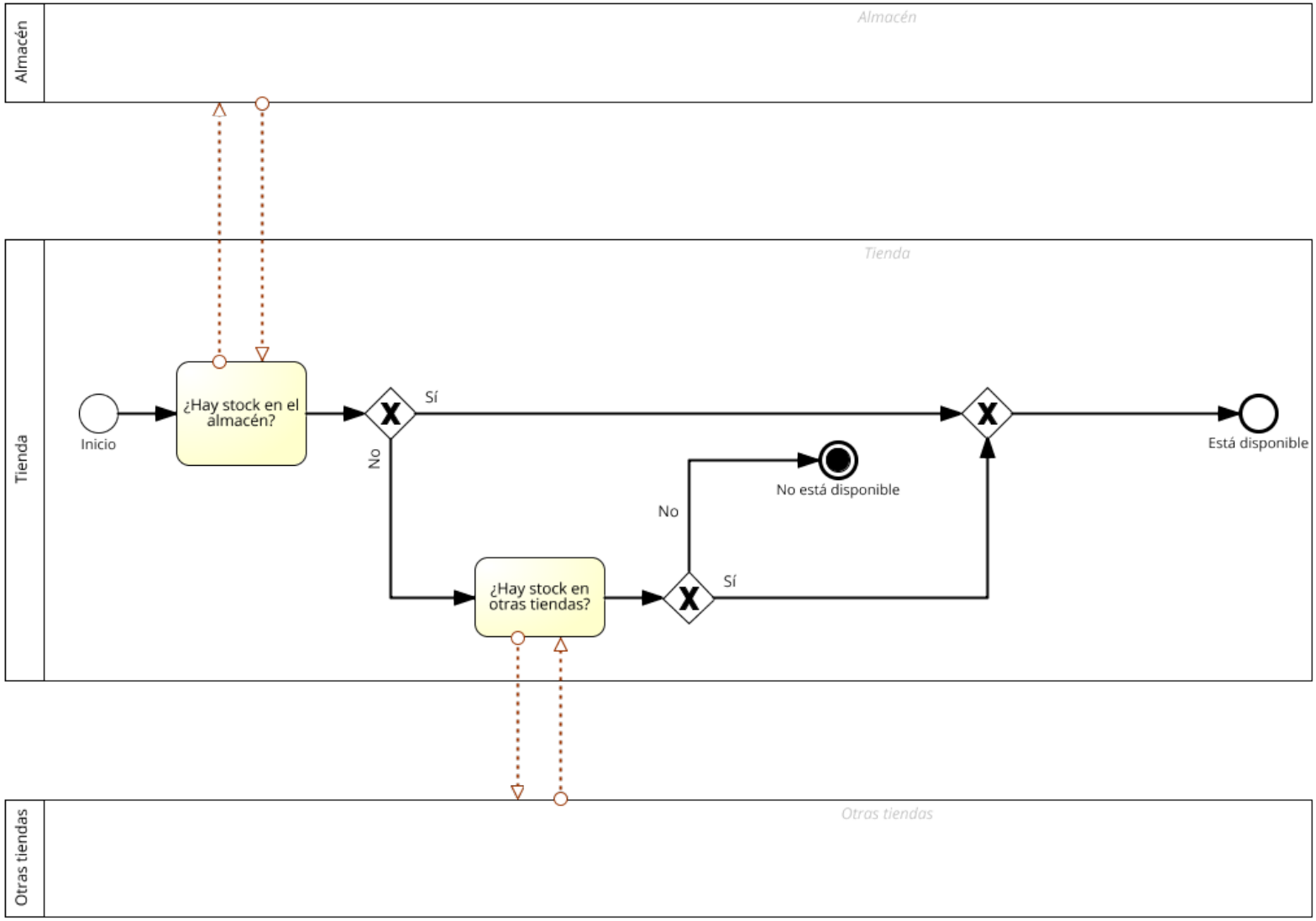


Figura 1: Proceso de venta

3.2. Proceso de compra al proveedor

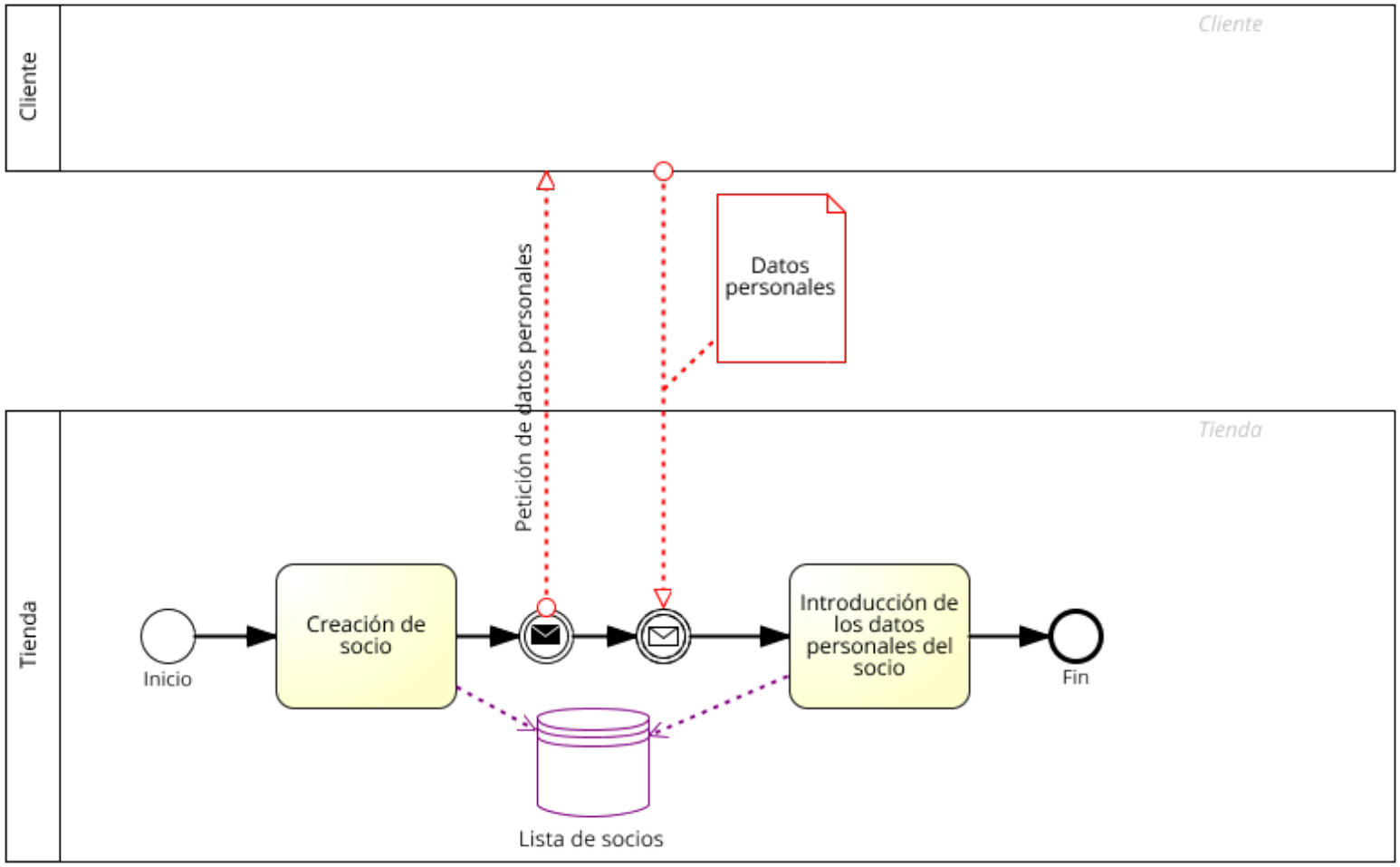


3.3. Proceso de disponibilidad



.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

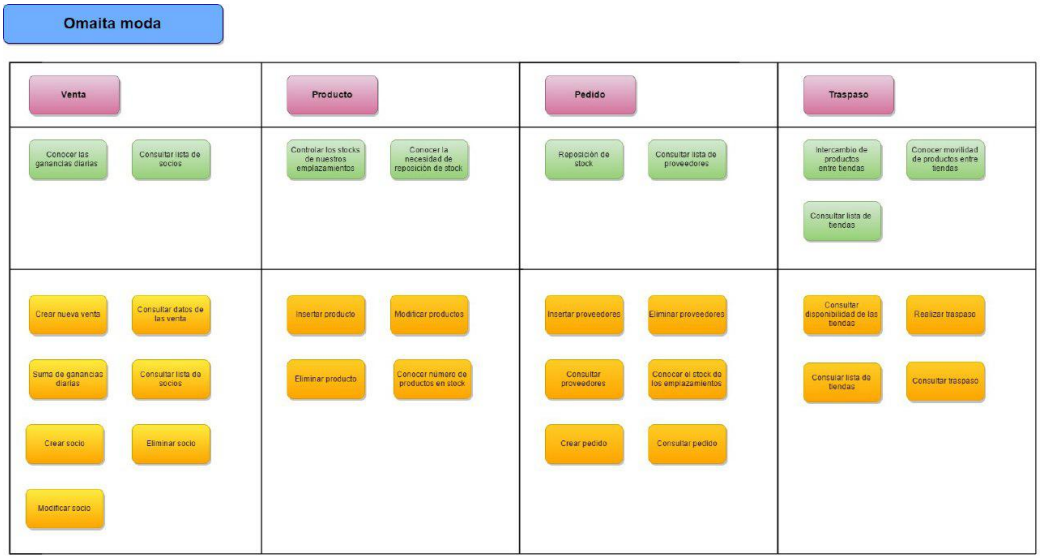


Figura 5: Mapa de requisitos

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

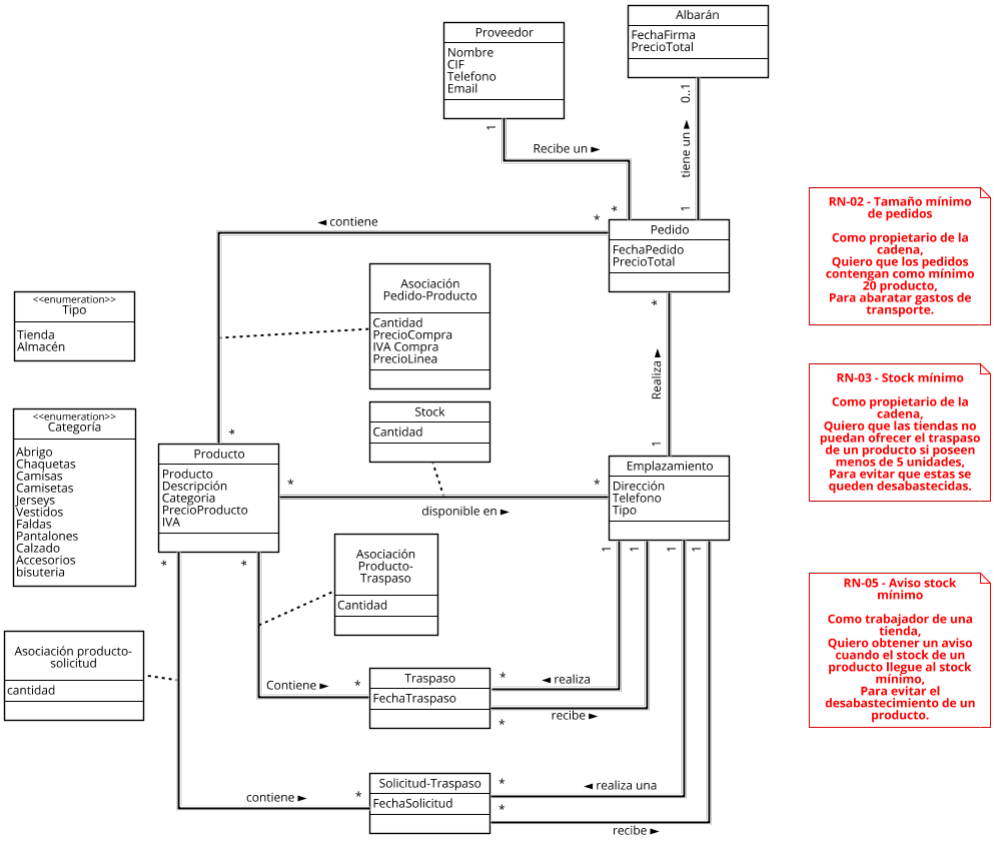


Figura 6: UML Traspaso / Pedido

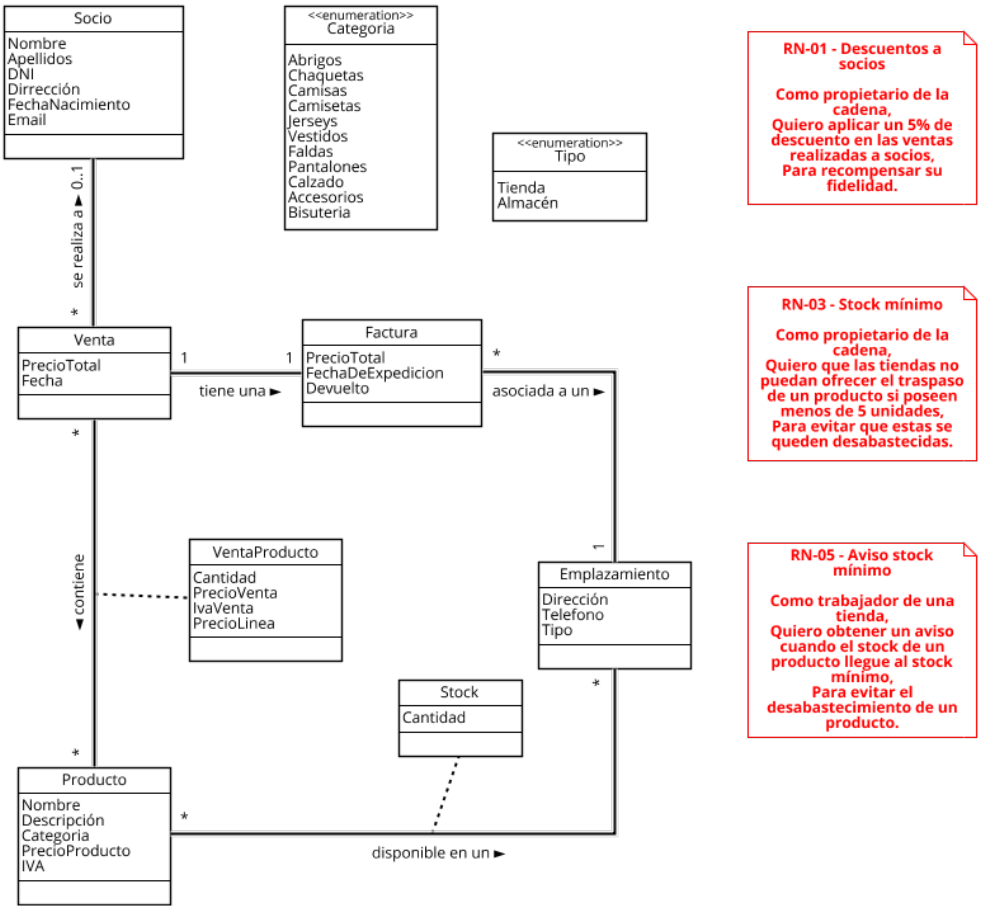


Figura 7: UML Venta

7.2. Escenarios de prueba

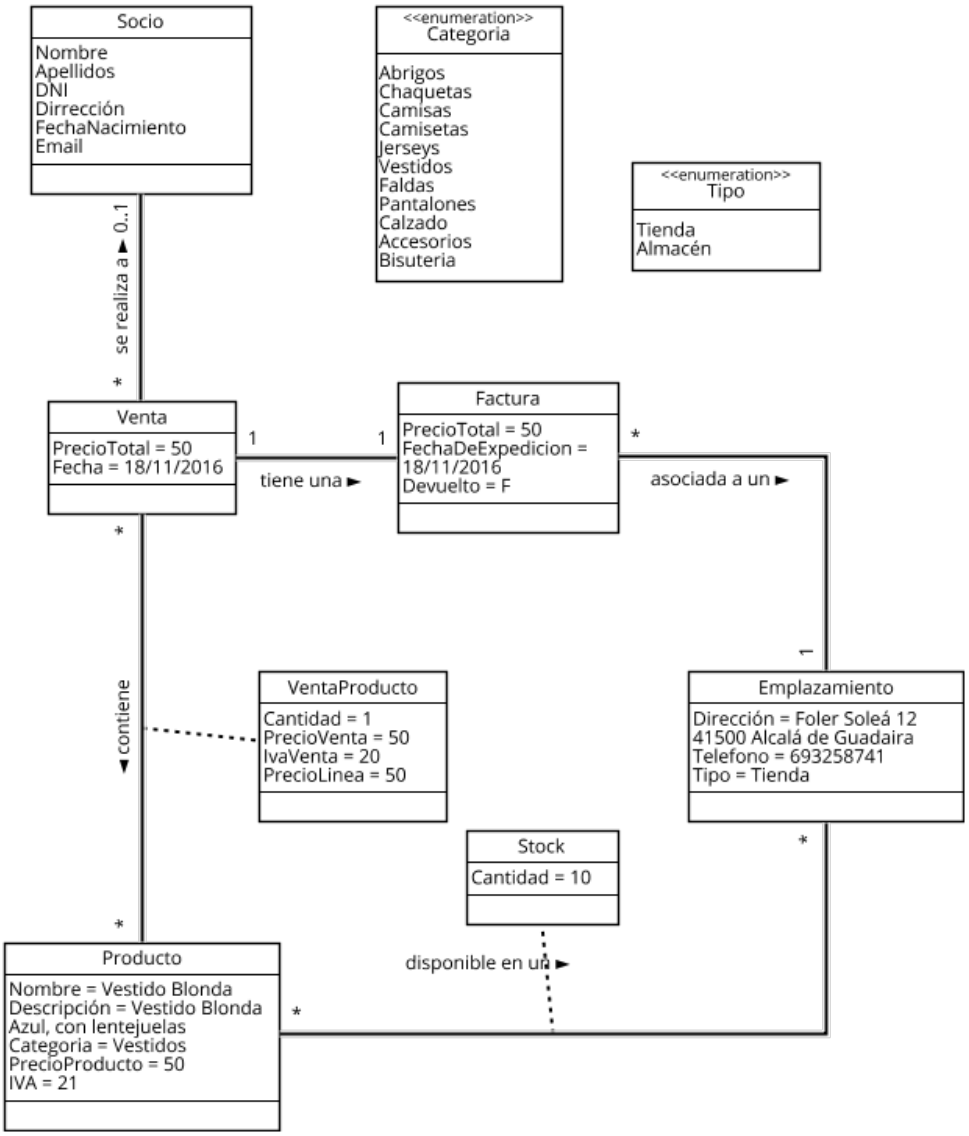


Figura 8: UML Venta NO Socio

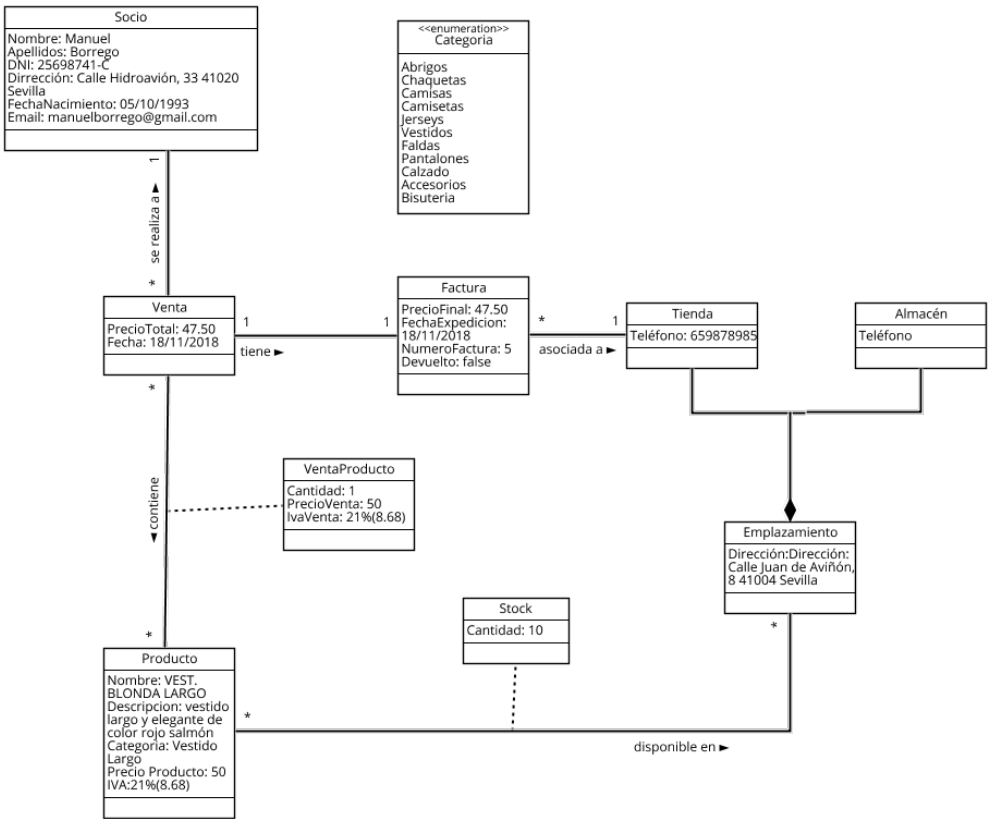


Figura 9: UML Venta Socio

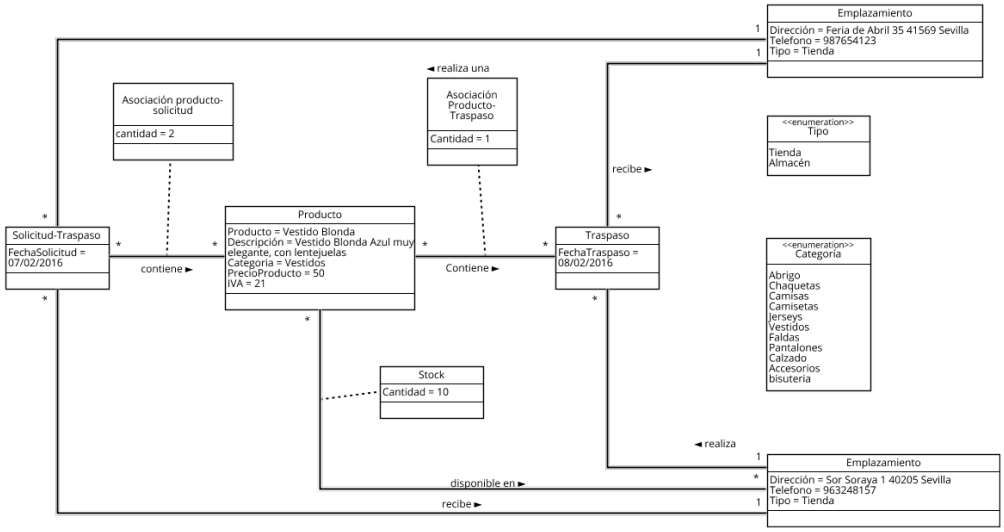


Figura 10: UML Traspaso Tienda - Tienda

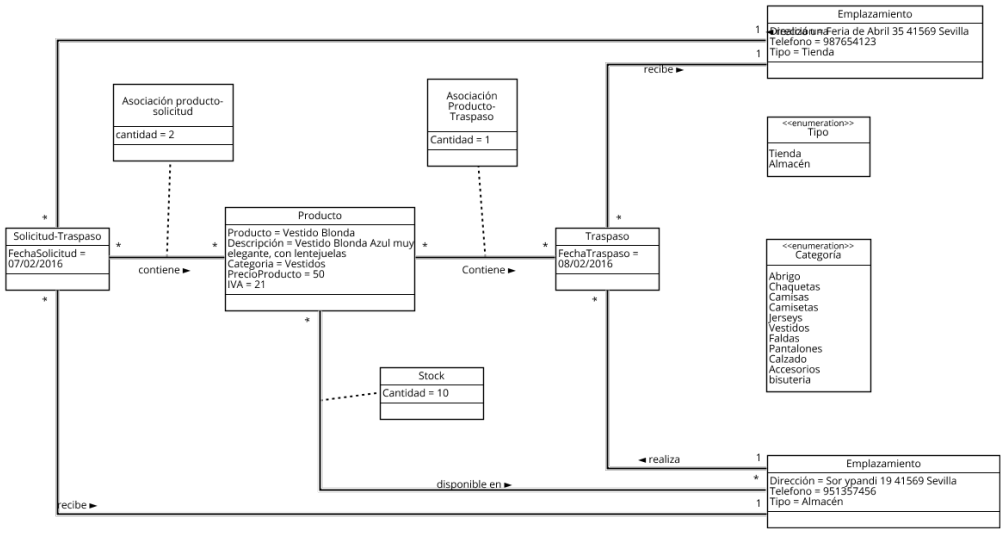


Figura 11: UML Traspaso Tienda - Almacén

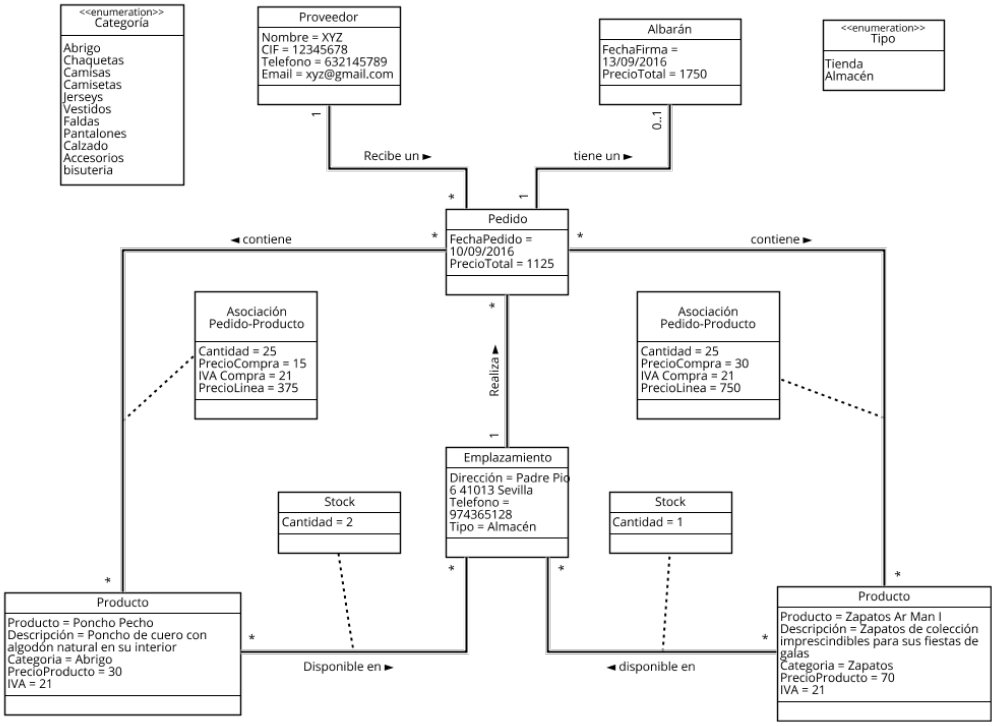


Figura 12: UML Pedido Tienda - Proveedor

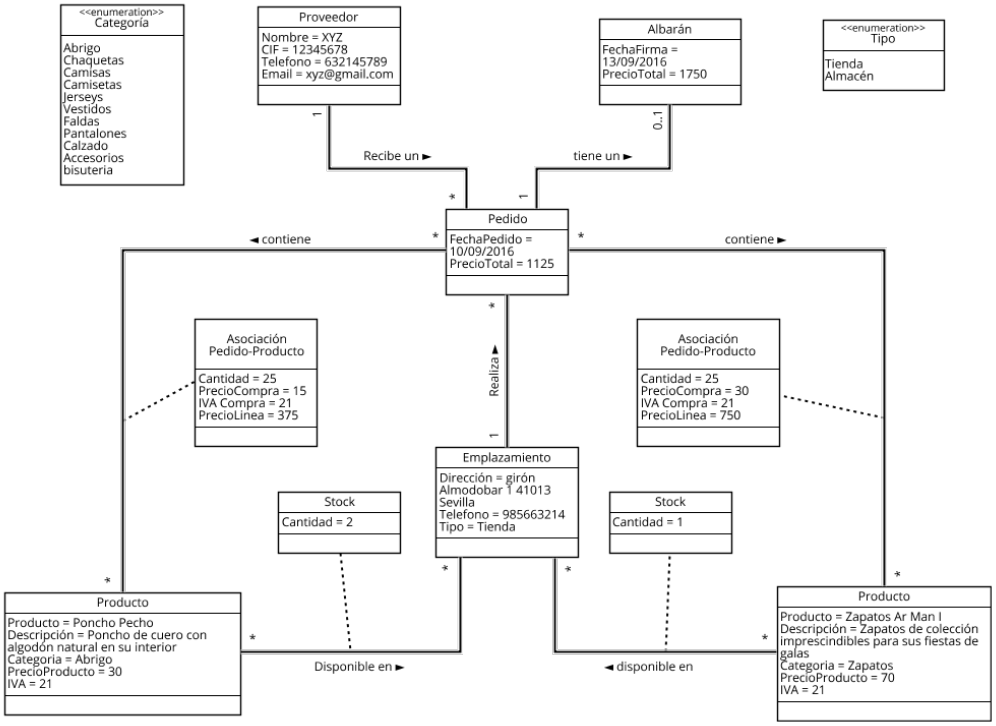


Figura 13: UML Pedido Almacén - Proveedor

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

9. Modelos relacionales

9.0.1. 3FN Venta

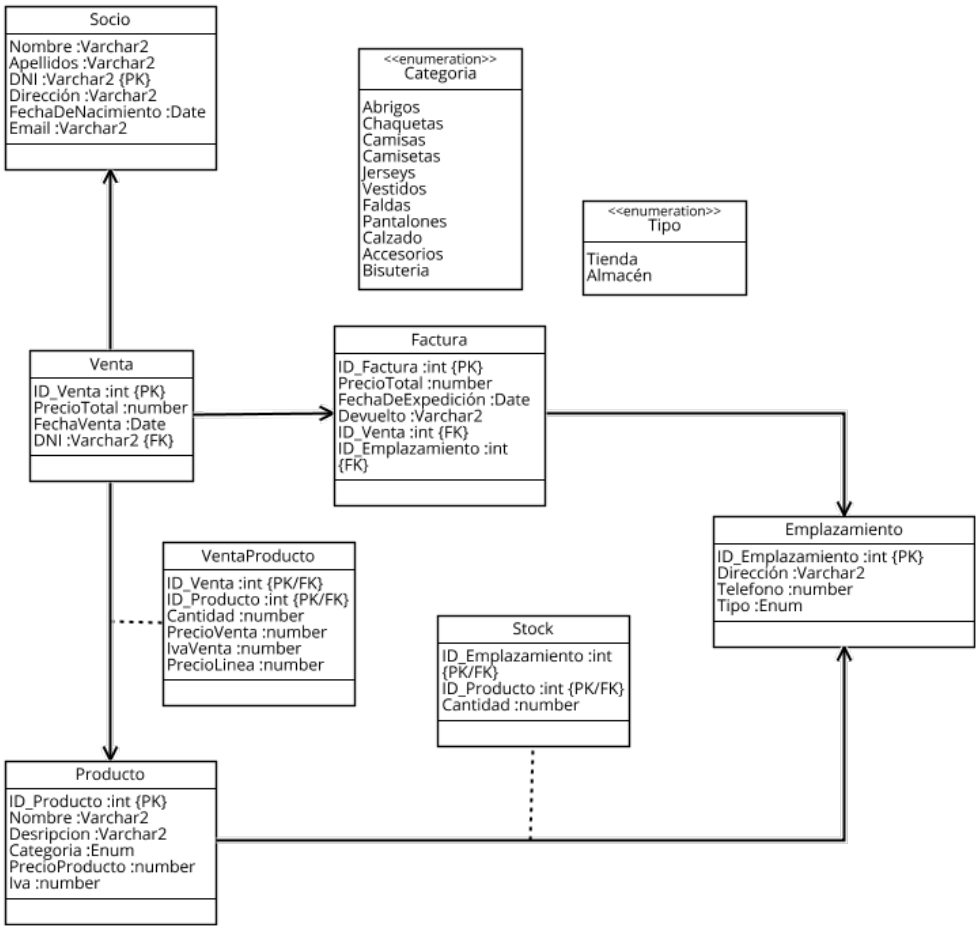


Figura 14: 3FN Venta

9.0.2. 3FN Traspaso - Pedido

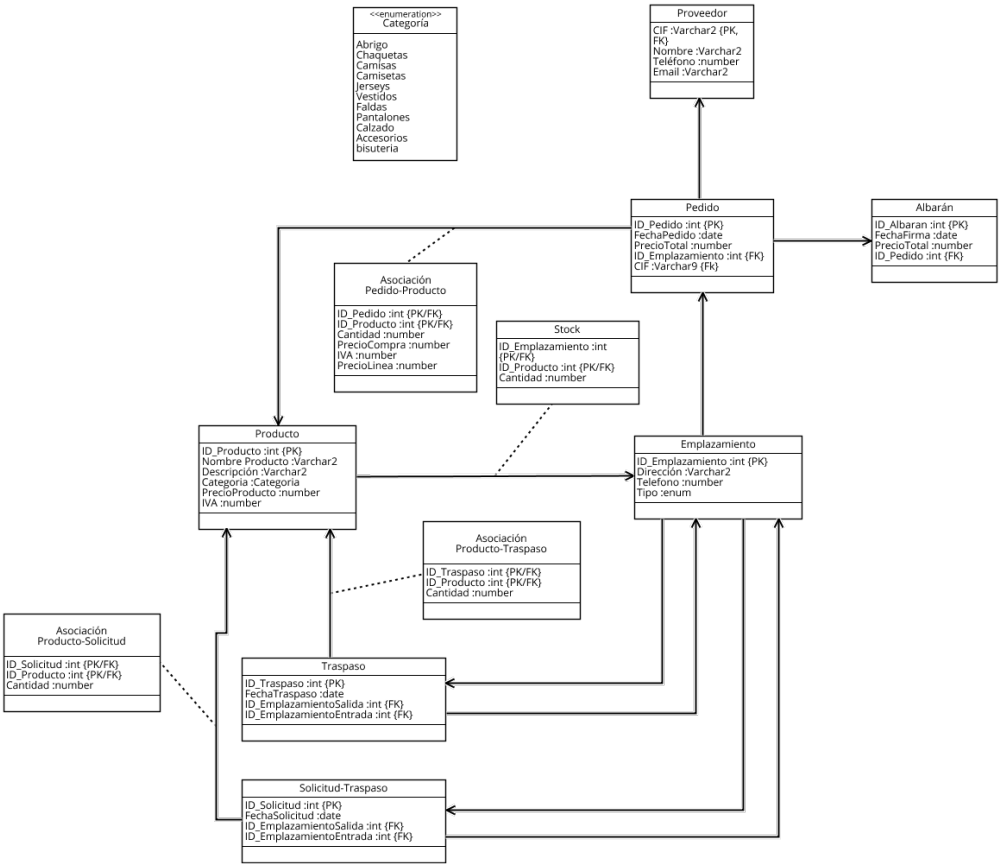


Figura 15: 3FN Traspaso - Pedido

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,
  PrecioTotal Number NOT NULL,
  FechaVenta DATE NOT NULL,
  DNI VARCHAR2(9),
  FOREIGN KEY(DNI) REFERENCES SOCIO
);

CREATE TABLE FACTURA (
  ID_FACTURA int PRIMARY KEY,
  PrecioTotal Number,
  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,
PrecioTotal NUMBER 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,
PrecioTotal NUMBER 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 ('Abrigo','Chaquetas','Camisas'
                           ,'Camisetas','Jerseys','Vestidos','Faldas'
                           ,'Pantalones','Calzado','Accesorios','Bisuteria')),
```

```
    PrecioProducto NUMBER NOT NULL,
    IVA NUMBER NOT NULL
);

CREATE TABLE STOCK(
    ID_Emplazamiento int,
    ID_Producto int,
    PRIMARY KEY (ID_Emplazamiento, ID_Producto),
    Cantidad NUMBER(6) NOT NULL,
    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,
    PrecioCompra NUMBER NOT NULL,
    IVA NUMBER NOT NULL,
    PrecioLinea NUMBER,
    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,
    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,
    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),
    PrecioVenta NUMBER,
    IvaVenta NUMBER,
    PrecioLinea Number
);

/* 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_Nuevo_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_Nuevo_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_Nuevo_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
```

```
239   FOR EACH ROW
      BEGIN
241     SELECT S_ID_Emplazamiento.NEXTVAL
        INTO :NEW.ID_Emplazamiento FROM DUAL;
243 END crea_ID_Emplazamiento;
    /
245
    CREATE OR REPLACE TRIGGER crea_ID_Venta
247   BEFORE INSERT ON Venta
      FOR EACH ROW
249   BEGIN
        SELECT S_ID_Venta.NEXTVAL
251     INTO :NEW.ID_Venta FROM DUAL;
    END crea_ID_Venta;
253 /
```

sql/tablas.sql

10.2. Funciones y Procedures

```
1   CREATE OR REPLACE PROCEDURE Socio_Nuevo (
      p_Nombre IN SOCIO.Nombre%TYPE,
3     p_Apellidos IN SOCIO.Apellidos%TYPE,
      p_Direccion IN SOCIO.Direccion%TYPE,
5     p_FechaDeNacimiento IN SOCIO.FechaDeNacimiento%TYPE,
      p_Email IN SOCIO.Email%TYPE,
7     p_DNI IN SOCIO.DNI%TYPE)
     IS BEGIN
9     INSERT INTO SOCIO
      VALUES (p_Nombre, p_Apellidos, p_Direccion, p_FechaDeNacimiento,
11            p_Email,p_DNI);
    END Socio_Nuevo;
13
    /
15
    CREATE OR REPLACE PROCEDURE Producto_Nuevo(
17    P_ID_Producto IN PRODUCTO.ID_Producto%TYPE,
      P_Nombre IN PRODUCTO.Nombre%TYPE,
19    P_Descripcion IN PRODUCTO.Descripcion%TYPE,
      P_Categoria IN PRODUCTO.Categoria%TYPE,
21    P_PrecioProducto IN PRODUCTO.PrecioProducto%TYPE,
      P_IVA IN PRODUCTO.IVA%TYPE)
23    IS BEGIN
      INSERT INTO PRODUCTO
25      VALUES(P_ID_Producto,P_Nombre,P_Descripcion,P_Categoria,
              P_PrecioProducto, P_IVA);
27  END Producto_Nuevo;

29  /

31  CREATE OR REPLACE PROCEDURE Stock_Nuevo (
        P_ID_Emplazamiento IN STOCK.ID_Emplazamiento%TYPE,
33      P_ID_Producto IN STOCK.ID_Producto%TYPE,
        P_Cantidad IN STOCK.Cantidad%TYPE)
35      IS BEGIN
        INSERT INTO STOCK
37      VALUES (P_ID_Emplazamiento, P_ID_Producto, P_Cantidad);
    END Stock_Nuevo;
39
    /
41
    CREATE OR REPLACE PROCEDURE PRODUCTO_TRASPASO_Nuevo (
43      p_ID_Traspaso IN ASOCIACION_PRODUCTO_TRASPASO.ID_Traspaso%TYPE,
        p_ID_Producto IN ASOCIACION_PRODUCTO_TRASPASO.ID_Producto%TYPE,
45      p_Cantidad IN ASOCIACION_PRODUCTO_TRASPASO.Cantidad%TYPE)
        IS BEGIN
47      INSERT INTO ASOCIACION_PRODUCTO_TRASPASO
        VALUES(p_ID_Traspaso,p_ID_Producto,p_Cantidad);
49  END PRODUCTO_TRASPASO_Nuevo;

51  /
```

```
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_PrecioTotal IN FACTURA.PrecioTotal%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_PrecioTotal, 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_PrecioTotal IN ALBARAN.PrecioTotal%TYPE,
  p_ID_Pedido IN ALBARAN.ID_PEDIDO%TYPE
  )IS BEGIN
  INSERT INTO ALBARAN
  VALUES ( p_ID_Albaran, p_FechaFirma, p_PrecioTotal, 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_PrecioTotal IN PEDIDO.PrecioTotal%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_PrecioTotal ,
           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,
    p_PrecioLinea IN ASOCIACION_PEDIDO_PRODUCTO.preciolinea%TYPE)IS
    BEGIN

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

/


CREATE OR REPLACE PROCEDURE Venta_Nueva(
  P_ID_Venta IN VENTA.ID_Venta%TYPE,
  P_PrecioTotal IN VENTA.PrecioTotal%TYPE,
  P_FechaVenta IN VENTA.FechaVenta%TYPE,
  P_DNI IN VENTA.DNI%TYPE)
  IS BEGIN
  INSERT INTO VENTA
    VALUES(P_ID_Venta, P_PrecioTotal, 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,
    P_PrecioLinea IN ASOCIACION_VENTA_PRODUCTO.PrecioLinea%TYPE
    )
    IS
    BEGIN
    INSERT INTO ASOCIACION_VENTA_PRODUCTO
        VALUES(P_ID_Venta, P_ID_Producto,P_Cantidad,P_PrecioVenta,P_IvaVenta,P_PrecioLinea);
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;

/

/*FUNCIONES*/

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;

/

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 ganancias_mensuales(fechainicio IN factura.fechadeexpedicion%TYPE,fechafin
      IN factura.fechadeexpedicion%TYPE )
return number is f_preciototal factura.preciototal%TYPE;
v_preciototal factura.preciototal%TYPE;
p_preciototal albaran.preciototal%TYPE;
begin
select sum(preciototal) into v_preciototal from factura where fechadeexpedicion between fechainicio
     and fechafin;
select sum(preciototal) into p_preciototal from albaran where fechafirma between fechainicio and
     fechafin;
f_preciototal := v_preciototal - p_preciototal;
return (f_preciototal);
end ganancias_mensuales;

/
```

sql/funciones_y_procedures.sql

10.3. Triggers

```
/* TRIGGERS */

CREATE OR REPLACE TRIGGER pedido_minimo
  BEFORE INSERT or UPDATE ON ASOCIACION_PEDIDO_PRODUCTO
  FOR EACH ROW
BEGIN
  IF :NEW.cantidad < 20
  THEN raise_application_error(-20600, :NEW.cantidad || 'No se pueden pedir menos de 20 unidades de
      un producto');
  END IF;
```

```
END;

/

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

/

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

/

CREATE OR REPLACE TRIGGER Inicializa_nueva_Venta
BEFORE INSERT ON VENTA
FOR EACH ROW
BEGIN
:NEW.PrecioTotal := 0;
:NEW.FechaVenta := SYSDATE;
END Comprueba_Venta;

/

CREATE OR REPLACE TRIGGER Inicializa_nuevo_Pedido
BEFORE INSERT ON PEDIDO
FOR EACH ROW
BEGIN
:NEW.PrecioTotal := 0;
:NEW.FechaPedido := SYSDATE;
END Comprueba_Pedido;

/

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


/

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

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

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

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

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

/

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

/

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

/


CREATE OR REPLACE TRIGGER modifica_stock_pedido
BEFORE INSERT OR UPDATE ON ALBARAN
FOR EACH ROW
DECLARE
e_ID_Emplazamiento Emplazamiento.ID_Emplazamiento%TYPE;
p_ID_PEDIDO PEDIDO.ID_PEDIDO%TYPE;

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

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

BEGIN

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

```
/

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

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

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

/
```

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_precio NUMBER, 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_precio NUMBER,
        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_precioVenta Number, 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_precioTotal INT, 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,
            i_precioLinea NUMBER, salidaEsperada BOOLEAN);
    PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_venta INT, e_id_producto INT, salidaEsperada BOOLEAN);
END PRUEBAS_A_VENTA_PRODUCTO;
```

```
/

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

/

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

/

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

/

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

/

/* CUERPOS DE PRUEBAS */

CREATE OR REPLACE PACKAGE BODY PRUEBAS_SOCIO AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM SOCIO;
    END 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) AS
    salida BOOLEAN := true;
```

```
234     actual socio%ROWTYPE;
        BEGIN
236         INSERT INTO socio VALUES (i_nombre, i_apellidos, i_direccion, i_nacimiento,
            i_email, i_DNI);
238
            SELECT * INTO actual FROM SOCIO WHERE DNI = i_DNI;
240
            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
242             salida := false;
            END IF;
244
            PRINTR(nombre_prueba, salida, salidaEsperada);
246
            EXCEPTION
248         WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
250             ROLLBACK;
        END insertar;
252
        PROCEDURE actualizar
254         (nombre_prueba VARCHAR2, a_DNI VARCHAR2, a_direccion VARCHAR2,
            a_email VARCHAR2, salidaEsperada BOOLEAN) AS
256         salida BOOLEAN := true;
            actual socio%ROWTYPE;
258         BEGIN
                UPDATE SOCIO SET DIRECCION = a_direccion, EMAIL = a_email where DNI = a_DNI;
260
                SELECT * INTO actual FROM SOCIO WHERE DNI = a_DNI;
262
                IF (actual.direccion<>a_direccion) OR (actual.email<>a_email) THEN
264                 salida := false;
                END IF;
266
                PRINTR(nombre_prueba, salida, salidaEsperada);
268
                EXCEPTION
270             WHEN OTHERS THEN
                    PRINTR(nombre_prueba, false, salidaEsperada);
272                 ROLLBACK;
            END actualizar;
274
        PROCEDURE eliminar
276         (nombre_prueba VARCHAR2, e_DNI VARCHAR2, salidaEsperada BOOLEAN) AS
            salida BOOLEAN := true;
278         cantidad INT;
            BEGIN
280             DELETE FROM SOCIO WHERE DNI = e_DNI;
282             SELECT COUNT(*) INTO cantidad FROM SOCIO WHERE DNI = e_DNI;
284             IF (cantidad<>0) THEN
                    salida := false;
286             END IF;
288             PRINTR(nombre_prueba, salida, salidaEsperada);
290             EXCEPTION
                WHEN OTHERS THEN
292                 PRINTR(nombre_prueba, false, salidaEsperada);
                    ROLLBACK;
294     END eliminar;
    END PRUEBAS_SOCIO;
296
    /
298
    CREATE OR REPLACE PACKAGE BODY PRUEBAS_PROVEEDOR AS
300
        PROCEDURE inicializar AS
302     BEGIN
            DELETE FROM PROVEEDOR;
304     END inicializar;
306     PROCEDURE insertar
          (nombre_prueba VARCHAR2, i_CIF VARCHAR2, i_nombre VARCHAR2,
308       i_telefono INT, i_email VARCHAR2, salidaEsperada BOOLEAN) AS
```

```
    salida BOOLEAN := true;
    actual proveedor%ROWTYPE;
    BEGIN
        INSERT INTO PROVEEDOR VALUES (i_CIF, i_nombre, i_telefono, i_email);

        SELECT * INTO actual FROM PROVEEDOR WHERE CIF = i_CIF;

        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
```

```
458     BEGIN
            DELETE FROM PRODUCTO;
460     END inicializar;

462     PROCEDURE insertar
            (nombre_prueba VARCHAR2, i_nombre VARCHAR2, i_descripcion VARCHAR2,
464         i_categoria VARCHAR2, i_precioProducto INT, i_iva INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
466     actual producto%ROWTYPE;
        w_ID_Producto INT;
468     BEGIN
            INSERT INTO producto VALUES(null,i_nombre, i_descripcion, i_categoria, i_precioProducto,
         i_iva);
470
            w_ID_Producto := S_ID_Producto.currval;
472         SELECT * INTO actual FROM PRODUCTO WHERE ID_Producto = w_ID_Producto;

474         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;
476         END IF;

478         PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

502             PRINTR(nombre_prueba, salida, salidaEsperada);

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

510     PROCEDURE eliminar
            (nombre_prueba VARCHAR2, e_id_producto INT, salidaEsperada BOOLEAN) AS
512         salida BOOLEAN := true;
            cantidad INT;
514         BEGIN
                DELETE FROM PRODUCTO  WHERE ID_Producto = e_id_producto;
516
                SELECT COUNT(*) INTO cantidad FROM PRODUCTO WHERE ID_Producto = e_id_producto;
518             IF (cantidad<>0) THEN
                    salida := false;
520             END IF;

522             PRINTR(nombre_prueba, salida, salidaEsperada);

524             EXCEPTION
                WHEN OTHERS THEN
526                 PRINTR(nombre_prueba, false, salidaEsperada);
                    ROLLBACK;
528     END eliminar;
    END PRUEBAS_PRODUCTO;
530
    /
```

```
532
CREATE OR REPLACE PACKAGE BODY PRUEBAS_FACTURA AS
534
    PROCEDURE inicializar AS
536     BEGIN
            DELETE FROM FACTURA;
538     END inicializar;

540     PROCEDURE insertar
          (nombre_prueba VARCHAR2, i_precioTotal INT, i_fechaDeExpedicion DATE, i_devuelto VARCHAR2,
542       i_id_venta INT, i_id_emplazamiento INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
544     actual factura%ROWTYPE;
        w_ID_factura INT;
546     BEGIN
            INSERT INTO factura VALUES(null,i_precioTotal, i_fechaDeExpedicion, i_devuelto, i_id_venta,
         i_id_emplazamiento);
548
            w_ID_factura := S_ID_Factura.currval;
550         SELECT * INTO actual FROM factura WHERE ID_factura = w_ID_factura;

552         IF (actual.precioTotal<>i_precioTotal) OR (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;
554         END IF;

556         PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

572             SELECT * INTO actual FROM factura WHERE ID_factura = a_id_factura;
                IF (actual.devuelto<>a_devuelto) THEN
574                 salida := false;
                END IF;
576
                PRINTR(nombre_prueba, salida, salidaEsperada);
578
                EXCEPTION
580             WHEN OTHERS THEN
                    PRINTR(nombre_prueba, false, salidaEsperada);
582                 ROLLBACK;
            END actualizar;
584
    PROCEDURE eliminar
586         (nombre_prueba VARCHAR2, e_id_factura INT, salidaEsperada BOOLEAN) AS
            salida BOOLEAN := true;
588         cantidad INT;
            BEGIN
590             DELETE FROM factura  WHERE ID_factura = e_id_factura;

592             SELECT COUNT(*) INTO cantidad FROM factura WHERE ID_factura = e_id_factura;
                IF (cantidad<>0) THEN
594                 salida := false;
                END IF;
596
                PRINTR(nombre_prueba, salida, salidaEsperada);
598
                EXCEPTION
600             WHEN OTHERS THEN
                    PRINTR(nombre_prueba, false, salidaEsperada);
602                 ROLLBACK;
    END eliminar;
604 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,
        i_cantidad INT, salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
    actual stock%ROWTYPE;
    BEGIN
        INSERT INTO stock VALUES(i_id_emplazamiento, i_id_producto, i_cantidad);

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

        IF (actual.id_emplazamiento<>i_id_emplazamiento) OR (actual.id_producto<>i_id_producto) 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 actualizar
        (nombre_prueba VARCHAR2, a_id_emplazamiento INT, a_id_producto INT,
        a_cantidad INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
        actual stock%ROWTYPE;
        BEGIN
            UPDATE stock SET CANTIDAD = a_cantidad  where  ID_Emplazamiento = a_id_emplazamiento and
     ID_Producto = a_id_producto;

            SELECT * INTO actual FROM stock WHERE  ID_Emplazamiento = a_id_emplazamiento and
     ID_Producto = a_id_producto;
            IF (actual.cantidad<>a_cantidad) 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, e_id_producto INT,
        salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
        cantidad INT;
        BEGIN
            DELETE FROM stock  WHERE ID_Emplazamiento = e_id_emplazamiento and ID_Producto =
     e_id_producto;

            SELECT COUNT(*) INTO cantidad FROM stock WHERE ID_Emplazamiento = e_id_emplazamiento 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;
```

```
678     PROCEDURE stock_correcto
            (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,stock_previo int, cantidad
        INT, salidaEsperada BOOLEAN) AS
680       salida BOOLEAN := true;
          stock_nuevo int;
682       BEGIN
            select cantidad into stock_nuevo from stock where id_emplazamiento = e_id_emplazamiento and
        id_producto= e_id_producto;
684           if(stock_previo-cantidad)<> stock_nuevo then
                salida := false;
686           END IF;

688           PRINTR(nombre_prueba, salida, salidaEsperada);

690           EXCEPTION
              WHEN OTHERS THEN
692               PRINTR(nombre_prueba, false, salidaEsperada);
                  ROLLBACK;
694
          END stock_correcto;
696
      PROCEDURE stock_correcto_p
698       (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,stock_previo int, cantidad
        INT, salidaEsperada BOOLEAN) AS
          salida BOOLEAN := true;
700       stock_nuevo int;
          BEGIN
702         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
704             salida := false;
            END IF;
706
            PRINTR(nombre_prueba, salida, salidaEsperada);
708
            EXCEPTION
710         WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
712             ROLLBACK;
       end stock_correcto_p;
714
       PROCEDURE stock_correcto_t
716      (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
718      salida BOOLEAN := true;
         stock_nuevo_envia int;
720      stock_nuevo_recibe int;
         BEGIN
722        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;
724        if(stock_previo_envia-cantidad)<>stock_nuevo_envia or (stock_previo_recibe+cantidad)<>
       stock_nuevo_recibe then
               salida := false;
726          END IF;
             PRINTR(nombre_prueba, salida, salidaEsperada);
728
             EXCEPTION
730          WHEN OTHERS THEN
                 PRINTR(nombre_prueba, false, salidaEsperada);
732              ROLLBACK;
         end stock_correcto_t;
734
END PRUEBA_STOCK;
736
/
738
CREATE OR REPLACE PACKAGE BODY PRUEBAS_VENTA AS
740
    PROCEDURE inicializar AS
742 BEGIN
        DELETE FROM VENTA;
744 END inicializar;
```

```
    PROCEDURE insertar
      (nombre_prueba VARCHAR2, i_precioventa number,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_precioventa, i_fecha, i_dni);

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

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

        PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

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

            PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_PEDIDO AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM pedido;
    END inicializar;

    PROCEDURE insertar
      (nombre_prueba VARCHAR2, i_fecha DATE, i_precio NUMBER,
      i_id_emplazamiento INT, i_cif VARCHAR2, salidaEsperada BOOLEAN) AS
```

```
    salida BOOLEAN := true;
    actual pedido%ROWTYPE;
    w_ID_pedido INT;
    BEGIN
        INSERT INTO pedido VALUES(null, i_fecha, i_precio, i_id_emplazamiento, i_cif);

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

        IF (actual.fechaPedido<>i_fecha) OR (actual.precioTotal<>i_precio) 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;
```

```
896     PROCEDURE eliminar
            (nombre_prueba VARCHAR2, e_id_Traspaso INT, salidaEsperada BOOLEAN) AS
898         salida BOOLEAN := true;
            cantidad INT;
900         BEGIN
                DELETE FROM traspaso  WHERE ID_traspaso = e_id_traspaso;
902
                SELECT COUNT(*) INTO cantidad FROM traspaso WHERE ID_traspaso = e_id_traspaso;
904             IF (cantidad<>0) THEN
                    salida := false;
906             END IF;

908             PRINTR(nombre_prueba, salida, salidaEsperada);

910             EXCEPTION
                WHEN OTHERS THEN
912                 PRINTR(nombre_prueba, false, salidaEsperada);
                    ROLLBACK;
914     END eliminar;
    END PRUEBAS_TRASPASO;
916
    /
918
    CREATE OR REPLACE PACKAGE BODY PRUEBAS_Solicitud_Traspaso AS
920
        PROCEDURE inicializar AS
922     BEGIN
            DELETE FROM Solicitud_Traspaso;
924     END inicializar;

926     PROCEDURE insertar
           (nombre_prueba VARCHAR2, i_fechaSolicitud DATE, i_id_emplazamientoSalida INT,
928         i_id_emplazamientoEntrada INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
930     actual Solicitud_Traspaso%ROWTYPE;
        w_ID_Solicitud INT;
932     BEGIN
            INSERT INTO Solicitud_Traspaso VALUES(null, i_fechaSolicitud, i_id_emplazamientoSalida,
         i_id_emplazamientoEntrada);
934
            w_ID_Solicitud := S_ID_Solicitud.currval;
936         SELECT * INTO actual FROM Solicitud_Traspaso WHERE ID_Solicitud = w_ID_Solicitud;

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

942         PRINTR(nombre_prueba, salida, salidaEsperada);

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

950     PROCEDURE eliminar
            (nombre_prueba VARCHAR2, e_id_solicitudTraspaso INT, salidaEsperada BOOLEAN) AS
952         salida BOOLEAN := true;
            cantidad INT;
954         BEGIN
                DELETE FROM Solicitud_Traspaso  WHERE ID_Solicitud = e_id_solicitudTraspaso;
956
                SELECT COUNT(*) INTO cantidad FROM Solicitud_Traspaso WHERE ID_Solicitud =
         e_id_solicitudTraspaso;
958             IF (cantidad<>0) THEN
                    salida := false;
960             END IF;

962             PRINTR(nombre_prueba, salida, salidaEsperada);

964             EXCEPTION
                WHEN OTHERS THEN
966                 PRINTR(nombre_prueba, false, salidaEsperada);
                    ROLLBACK;
968     END eliminar;
```

```
END PRUEBAS_Solicitud_Traspaso;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_Albaran AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM Albaran;
    END inicializar;

    PROCEDURE insertar
       (nombre_prueba VARCHAR2, i_fecha DATE,
        i_precio NUMBER, i_id_pedido INT, salidaEsperada BOOLEAN) AS
    salida BOOLEAN := true;
    actual Albaran%ROWTYPE;
    w_ID_Albaran INT;
    BEGIN
        INSERT INTO Albaran VALUES(null, i_fecha, i_precio, i_id_pedido);

        w_ID_Albaran := S_ID_Albaran.currval;
        SELECT * INTO actual FROM Albaran WHERE ID_Albaran = w_ID_Albaran;

        IF (actual.fechaFirma<>i_fecha) OR (actual.id_pedido<>i_id_pedido) 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_albaran INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
        cantidad INT;
        BEGIN
            DELETE FROM Albaran  WHERE ID_Albaran = e_id_albaran;

            SELECT COUNT(*) INTO cantidad FROM Albaran WHERE ID_Albaran = e_id_albaran;
            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_Albaran;

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_PRODUCTO_TRASPASO AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM asociacion_Producto_traspaso;
    END inicializar;

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

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

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

        PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

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

            PRINTR(nombre_prueba, salida, salidaEsperada);

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

/

CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_PRODUCTO_SOLICITUD AS

    PROCEDURE inicializar AS
    BEGIN
        DELETE FROM asociacion_Producto_Solicitud;
    END inicializar;

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

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

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

        PRINTR(nombre_prueba, salida, salidaEsperada);

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

    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 INT, i_iva INT, i_precioLinea 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, i_precioLinea);

        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
```

```
1186
    PROCEDURE inicializar AS
1188    BEGIN
            DELETE FROM asociacion_Venta_Producto;
1190    END inicializar;

1192    PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_id_venta INT, i_id_producto INT,
1194        i_cantidad NUMBER, i_precioVenta NUMBER, i_ivaVenta NUMBER,
            i_precioLinea NUMBER, salidaEsperada BOOLEAN) AS
1196    salida BOOLEAN := true;
        actual asociacion_Venta_Producto%ROWTYPE;
1198    BEGIN
            INSERT INTO asociacion_Venta_Producto VALUES(i_id_venta, i_id_producto, i_cantidad,
         i_precioVenta, i_ivaVenta, i_precioLinea);
1200
            SELECT * INTO actual FROM asociacion_Venta_Producto WHERE ID_Venta = i_id_venta and
         ID_producto = i_id_producto;
1202
            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) OR (actual.
         precioLinea<>i_precioLinea) THEN
1204            salida := false;
            END IF;
1206
            PRINTR(nombre_prueba, salida, salidaEsperada);
1208
            EXCEPTION
1210        WHEN OTHERS THEN
                PRINTR(nombre_prueba, false, salidaEsperada);
1212            ROLLBACK;
        END insertar;
1214
        PROCEDURE eliminar
1216        (nombre_prueba VARCHAR2, e_id_venta INT, e_id_producto INT, salidaEsperada BOOLEAN) AS
            salida BOOLEAN := true;
1218        cantidad INT;
            BEGIN
1220            DELETE FROM asociacion_Venta_Producto  WHERE ID_Venta = e_id_venta and ID_producto =
         e_id_producto;

1222            SELECT COUNT(*) INTO cantidad FROM asociacion_Venta_Producto WHERE ID_Venta = e_id_venta
         and ID_producto = e_id_producto;
                IF (cantidad<>0) THEN
1224                salida := false;
                END IF;
1226
                PRINTR(nombre_prueba, salida, salidaEsperada);
1228
                EXCEPTION
1230            WHEN OTHERS THEN
                    PRINTR(nombre_prueba, false, salidaEsperada);
1232                ROLLBACK;
        END eliminar;
1234 END PRUEBAS_A_VENTA_PRODUCTO;

1236 /

1238 DROP SEQUENCE S_ID_Producto;
     DROP SEQUENCE S_ID_Traspaso;
1240 DROP SEQUENCE S_ID_Solicitud;
     DROP SEQUENCE S_ID_Pedido;
1242 DROP SEQUENCE S_ID_Albaran;
     DROP SEQUENCE S_ID_Factura;
1244 DROP SEQUENCE S_ID_Emplazamiento;
     DROP SEQUENCE S_ID_Venta;
1246
     CREATE SEQUENCE S_ID_Emplazamiento START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1248 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;
1250 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;
1252 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;
1254 CREATE SEQUENCE S_ID_Factura START WITH 1 INCREMENT BY 1 MAXVALUE 9999;

1256 /* EXECUTE DE PRUEBAS */
```

```
-- Inicializacion
EXECUTE pruebas_albaran.inicializar();
EXECUTE PRUEBAS_A_PRODUCTO_TRASPASO.inicializar();
EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.inicializar();
EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.inicializar();
EXECUTE PRUEBAS_A_VENTA_PRODUCTO.inicializar();
EXECUTE PRUEBAS_FACTURA.inicializar();
EXECUTE pruebas_venta.inicializar();
EXECUTE pruebas_pedido.inicializar();
EXECUTE pruebas_proveedor.inicializar();
EXECUTE PRUEBA_STOCK.inicializar();
EXECUTE pruebas_producto.inicializar();
EXECUTE PRUEBAS_TRASPASO.inicializar();
EXECUTE PRUEBAS_SOLICITUD_TRASPASO.inicializar();
EXECUTE PRUEBAS_SOCIO.inicializar();
EXECUTE pruebas_emplazamiento.inicializar();
--SOCIO
EXECUTE PRUEBAS_SOCIO.inicializar();
EXECUTE PRUEBAS_SOCIO.insertar('Socio Insertar', 'Daniel', 'Gonzalez', '54148787G',  'Avenida del
     pepinillo', TO_DATE('10102015','DDMMYYYY'), 'dani@us.es', true);
EXECUTE PRUEBAS_SOCIO.insertar('Socio Insertar', 'Jose Luis', 'Marmol Romero', '29613400A',  'Calle
     Virgen', TO_DATE('10101996','DDMMYYYY'), 'jlmr@us.es', true);
EXECUTE PRUEBAS_SOCIO.actualizar('Socio Actualizar', '54148787G', 'Avenida del pepinillo234', '
     dani@us.es', true);
EXECUTE PRUEBAS_SOCIO.eliminar('Socio Eliminar', '54148787G', true);

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

--PRODUCTO
EXECUTE PRUEBAS_PRODUCTO.inicializar();
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion','Zapatos cutres','unos zaparos cutres pero
     calentitos','Calzado',12.5,21,true);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion','Cool shoes','Zapatos de cuero italiano','
     Calzado',5.95,21,true);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion','Sport Shoes','Zapatos de plastico','Calzado'
     ,6.95,21,true);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion','Brown Sandals','Sandalias de playa marrones',
     'Calzado',7.95,21,true);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion','Chupa de cuero','gran imitacion de la chupa
     de Han Solo','Calzado',500,21,true);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion','Chaleco de lana','por una gran disenadora','
     Jerseys',46.95,21,true);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion','Sombrero de paja','el gran sombrero de
     Mugiwara','Accesorios',20,21,true);
EXECUTE PRUEBAS_PRODUCTO.actualizar('Producto actualizacion',S_ID_Producto.currval-2,'Unos zapatos
     cutres pero calentitos, estan rotos',10.5,21,true);
EXECUTE PRUEBAS_PRODUCTO.eliminar('Producto elminar',S_ID_Producto.currval,true);

--EMPLAZAMIENTO
EXECUTE PRUEBAS_EMPLAZAMIENTO.inicializar();
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);
```

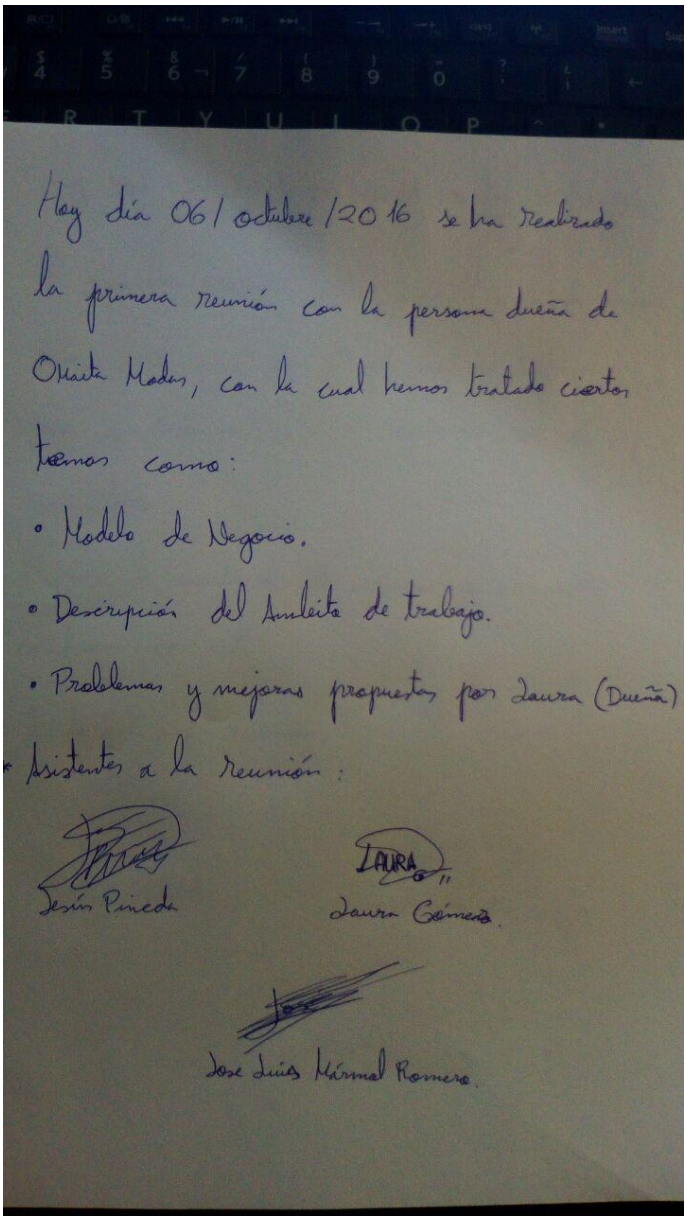
```
EXECUTE PRUEBA_STOCK.insertar('Stock insercion',S_ID_Emplazamiento.currval-2,S_ID_Producto.currval
    -3,30,true);
EXECUTE PRUEBA_STOCK.insertar('Stock insercion',S_ID_Emplazamiento.currval-1,S_ID_Producto.currval
    -1,11,true);
EXECUTE PRUEBA_STOCK.insertar('Stock insercion',S_ID_Emplazamiento.currval-1,S_ID_Producto.currval
    -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',0,sysdate,'29613400A',true);
EXECUTE pruebas_venta.insertar('Venta insercion',0,sysdate,null,true);
EXECUTE pruebas_venta.insertar('Venta insercion',0,sysdate,null,true);
EXECUTE pruebas_venta.eliminar('Venta eliminar',S_ID_venta.currval,true);

--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-2,3,10.5,21,31.5,true);
EXECUTE Pruebas_A_venta_producto.insertar('Venta-Producto insercion',s_id_venta.currval-1,
    s_id_producto.currval-1,2,46.95,21,93.9,true);
EXECUTE PRUEBAS_A_VENTA_PRODUCTO.insertar('Venta-Producto insercion',s_id_venta.currval-2,
    s_id_producto.currval-2,3,10.5,21,31.5,true);
EXECUTE Pruebas_A_venta_producto.insertar('Venta-Producto insercion',s_id_venta.currval-2,
    s_id_producto.currval-1,2,46.95,21,93.9,true);

--FACTURA
EXECUTE PRUEBAS_factura.inicializar();
EXECUTE Pruebas_factura.insertar('Factura insercion',119.13,sysdate,'F',s_id_venta.currval-2,
    S_ID_Emplazamiento.currval-1,true);
EXECUTE Pruebas_factura.insertar('Factura insercion',125.4,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);
EXECUTE Prueba_stock.stock_correcto('Actualizacion de stock tras venta',s_id_emplazamiento.currval-1,
    s_id_producto.currval-1,11,2,true);

--PEDIDO
EXECUTE PRUEBAS_PEDIDO.inicializar();
EXECUTE PRUEBAS_PEDIDO.insertar('Pedido insercion',sysdate,0,S_ID_Emplazamiento.currval-1,'B54129999'
    ,true);
EXECUTE PRUEBAS_PEDIDO.insertar('Pedido insercion',sysdate,0,S_ID_Emplazamiento.currval-2,'B54129999'
    ,true);
EXECUTE PRUEBAS_PEDIDO.insertar('Pedido insercion',sysdate,0,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-1,21,10.5,21,220.5,true);
EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.insertar('PEDIDO_PRODUCTO insercion',S_ID_Pedido.currval-2,
    s_id_producto.currval-1,22,46.95,21,0,true);
EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.insertar('PEDIDO_PRODUCTO insercion',S_ID_Pedido.currval-1,
    s_id_producto.currval-2,22,46.95,21,0,true);

--ALBARAN

EXECUTE PRUEBAS_ALBARAN.inicializar();
EXECUTE PRUEBAS_ALBARAN.insertar('Albaran insercion',sysdate,0,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,0,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
     -1, S_ID_Solicitud.currval, 3, true);
EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.eliminar('A-Producto-Solicitud Eliminar', S_ID_Solicitud.currval
     , S_ID_Producto.currval-1, true);

--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-2,31,29,4,s_ID_producto.currval-1,true);
EXECUTE PRUEBAS_A_PRODUCTO_TRASPASO.eliminar('A-Producto-Traspaso Actualizar', S_ID_traspaso.currval
     -1, S_ID_Producto.currval-1, true);
```

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