

UNIVERSIDAD DE SEVILLA

IISSI

PROYECTO

---

# Modas Omaita

---

*Autores:*

Daniel González Corzo

Jesús Pineda Márquez

José Luis Mármol Romero

Roberto Hueso Gómez

21 de junio de 2017



Escuela Técnica Superior de  
Ingeniería Informática

# Índice

<b>1. Introducción</b>	<b>3</b>
<b>2. Glosario de Términos</b>	<b>3</b>
<b>3. Modelo de negocio</b>	
3.1. Proceso de venta . . . . .	
3.2. Proceso de compra al proveedor . . . . .	
3.3. Proceso de disponibilidad . . . . .	
3.4. Proceso de creación de socios . . . . .	
<b>4. Visión General del Sistema</b>	
4.1. Descripción . . . . .	
4.2. Historias de usuario . . . . .	
<b>5. Catálogo de requisitos</b>	
5.1. Requisitos de información . . . . .	
5.2. Requisitos funcionales . . . . .	
5.3. Requisitos no funcionales . . . . .	
5.4. Reglas de negocio . . . . .	
<b>6. Pruebas de aceptación</b>	
<b>7. Modelo Conceptual</b>	
7.1. Diagramas de clases UML . . . . .	
7.2. Escenarios de prueba . . . . .	
<b>8. Matrices de trazabilidad</b>	
8.1. Pruebas de aceptación frente a requisitos . . . . .	
8.2. Pruebas de aceptación frente a escenarios de prueba . . . . .	
8.3. Tipos de UML frente a Requisitos . . . . .	
<b>9. Modelos relacionales</b>	
9.1. 3FN Venta . . . . .	
9.2. 3FN Traspaso - Pedido . . . . .	

## **10.Código SQL de la base de datos**

- 10.1. Tablas . . . . .
- 10.2. Funciones y Procedures . . . . .
- 10.3. Triggers . . . . .
- 10.4. Pruebas . . . . .

## **11.Apéndice**

- 11.1. Actas de reunión . . . . .

## 1. Introducción

Omaita Modas es una tienda situada en la localidad de Alcalá de Guadaira y más exactamente en la calle Pepe Lucas 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

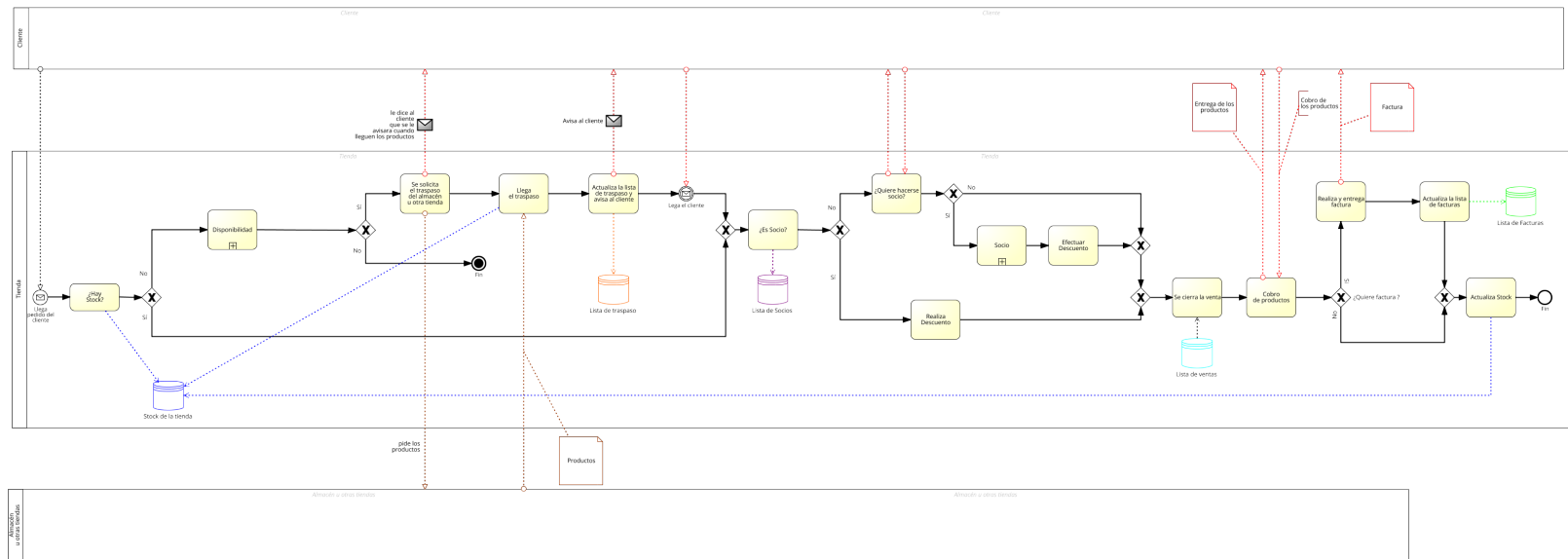
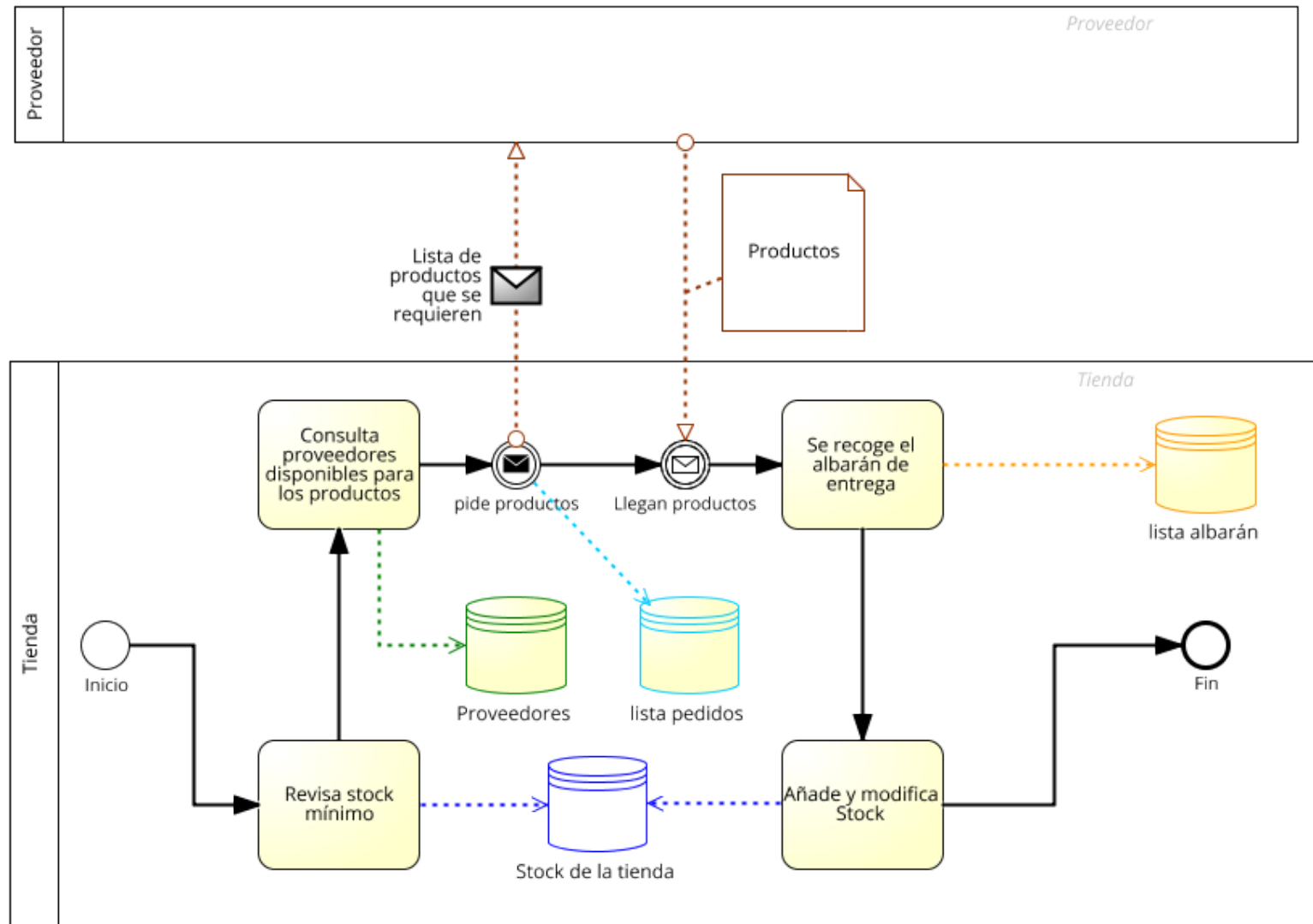


Figura 1: Proceso de venta



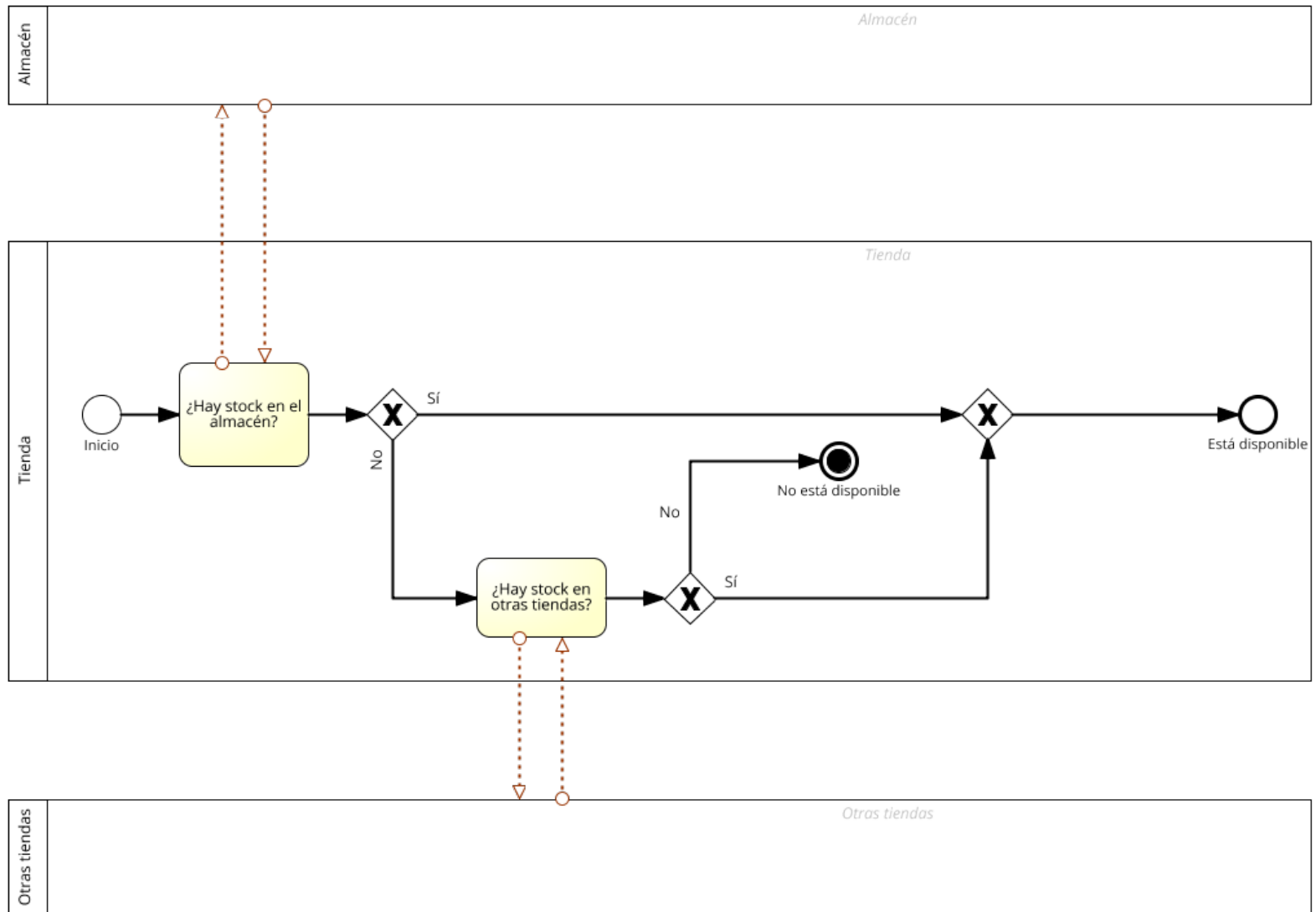
### 3.2. Proceso de compra al proveedor







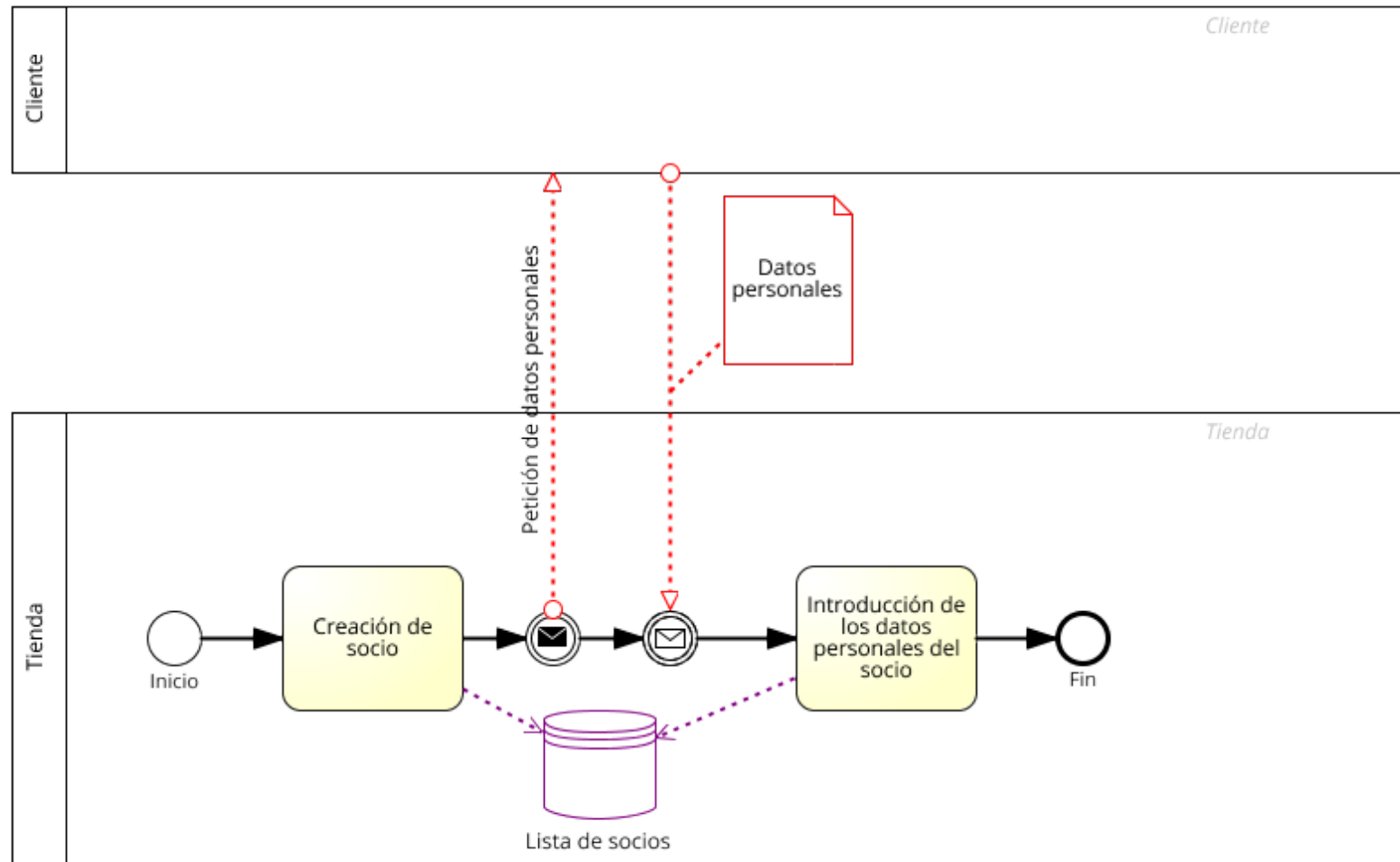
### 3.3. Proceso de disponibilidad



.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	<ul style="list-style-type: none"><li>▪ <b>Como</b> trabajador</li><li>▪ <b>Quiero</b> conocer las ventas diarias del negocio</li><li>▪ <b>Para</b> hacer ajustes a las ofertas del mismo</li></ul>
HU-2	<ul style="list-style-type: none"><li>▪ <b>Como</b> propietario de la cadena</li><li>▪ <b>Quiero</b> conocer el stock disponible en cada negocio</li><li>▪ <b>Para</b> organizar los pedidos a proveedores</li></ul>
HU-3	<ul style="list-style-type: none"><li>▪ <b>Como</b> propietario de la cadena</li><li>▪ <b>Quiero</b> conocer las ganancias mensuales</li><li>▪ <b>Para</b> tener un balance mensual</li></ul>
HU-4	<ul style="list-style-type: none"><li>▪ <b>Como</b> cliente</li><li>▪ <b>Quiero</b> ver una lista de los productos ofrecidos y disponibles</li><li>▪ <b>Para</b> realizar mis compras</li></ul>

## 5. Catálogo de requisitos

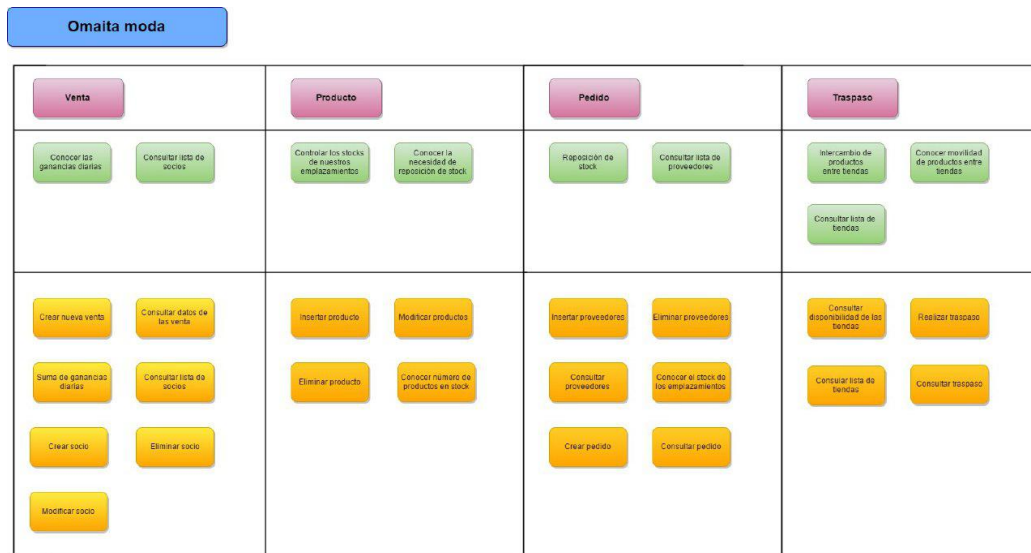


Figura 5: Mapa de requisitos

### 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 intenta 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

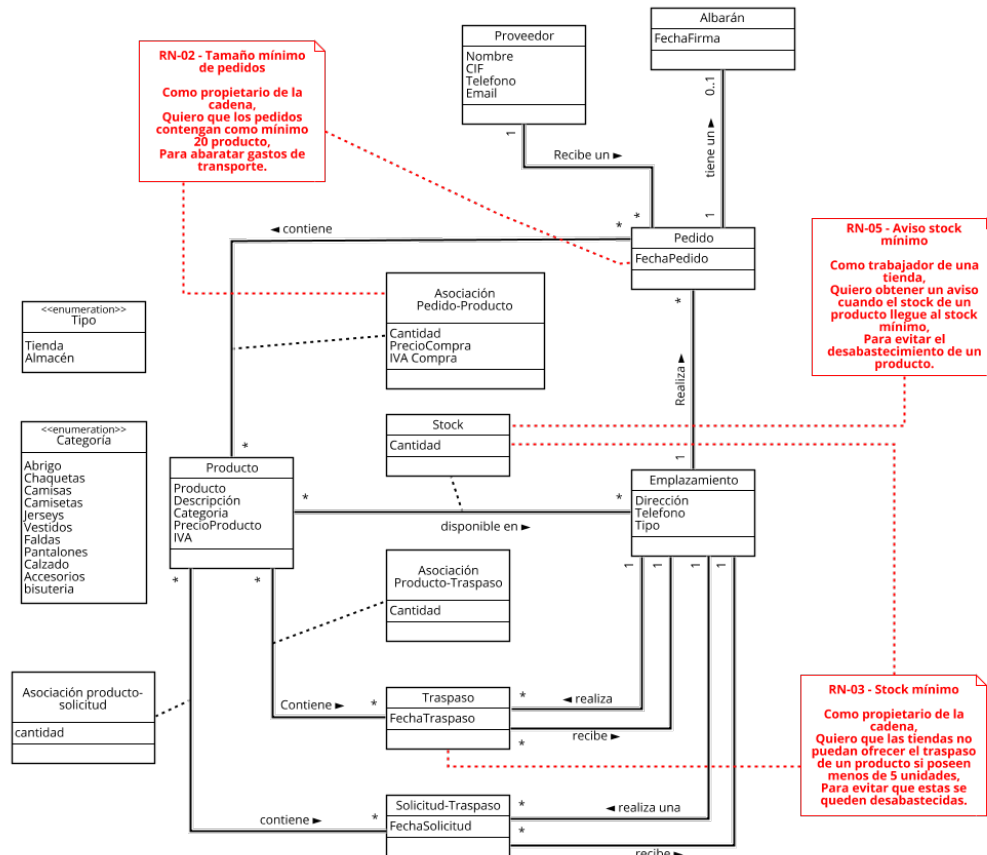


Figura 6: UML Traspaso / Pedido

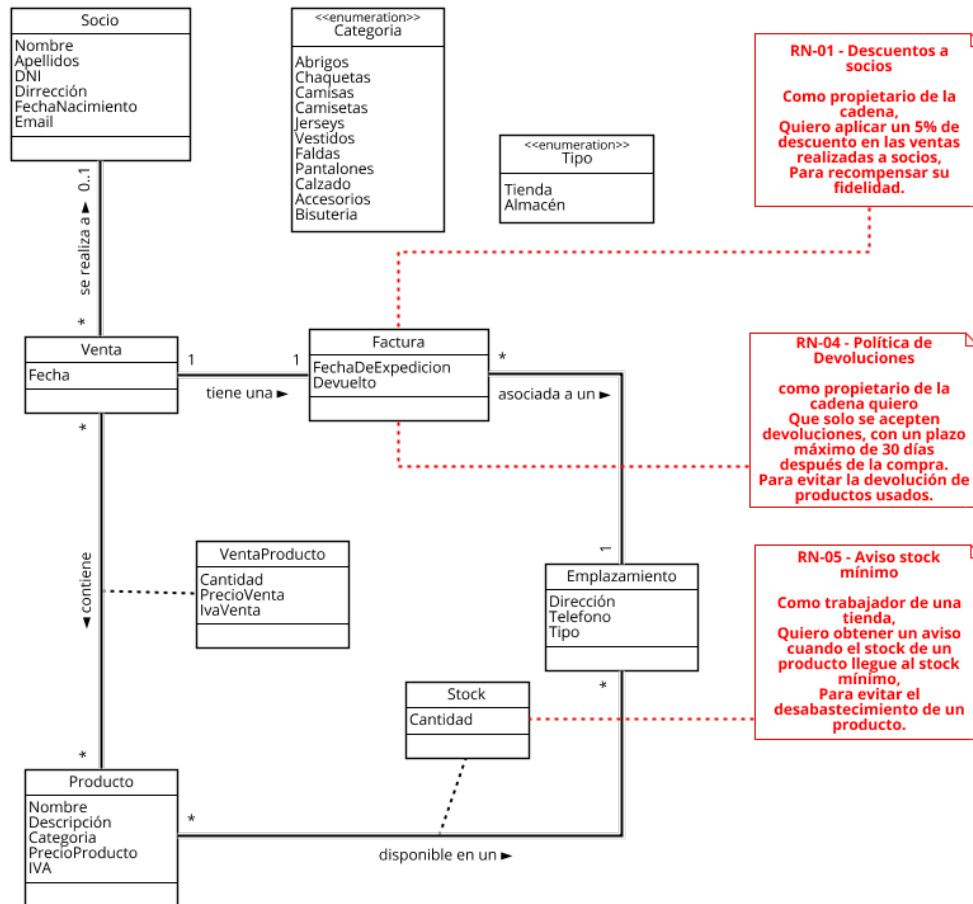


Figura 7: UML Venta

## 7.2. Escenarios de prueba

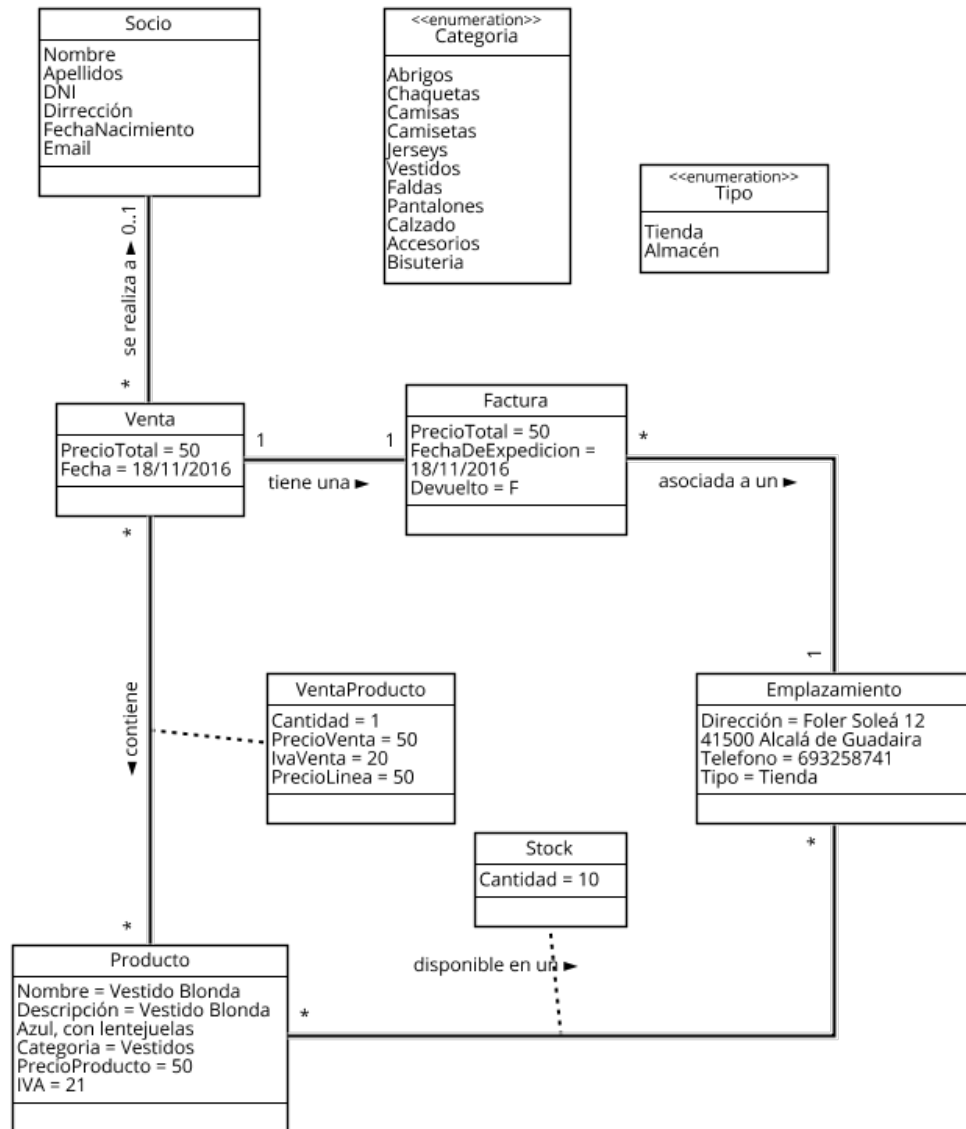


Figura 8: UML Venta NO Socio



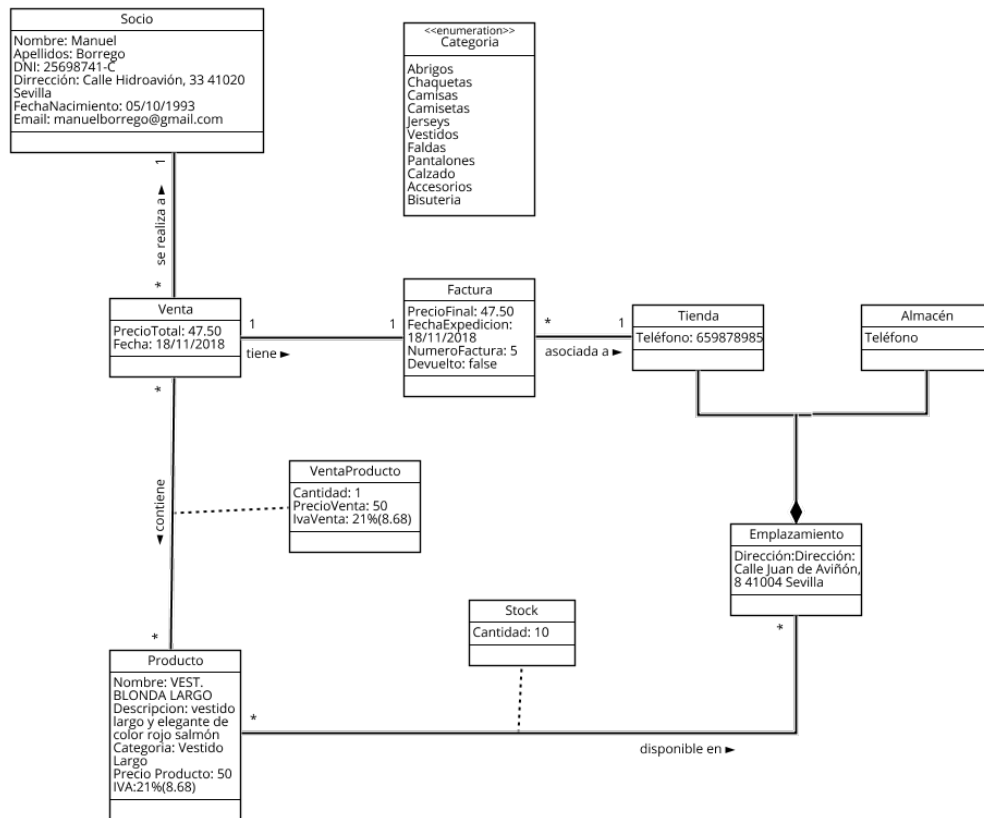


Figura 9: UML Venta Socio

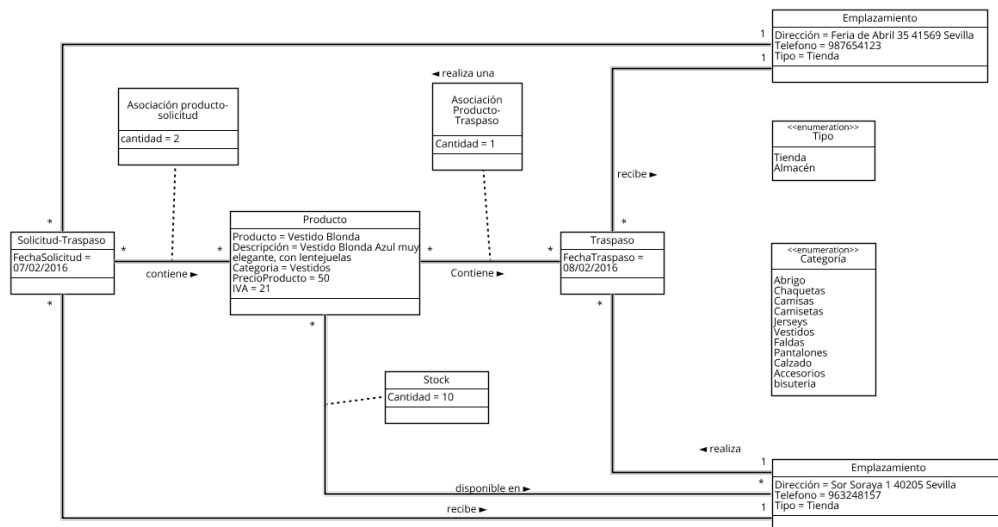


Figura 10: UML Traspaso Tienda - Tienda

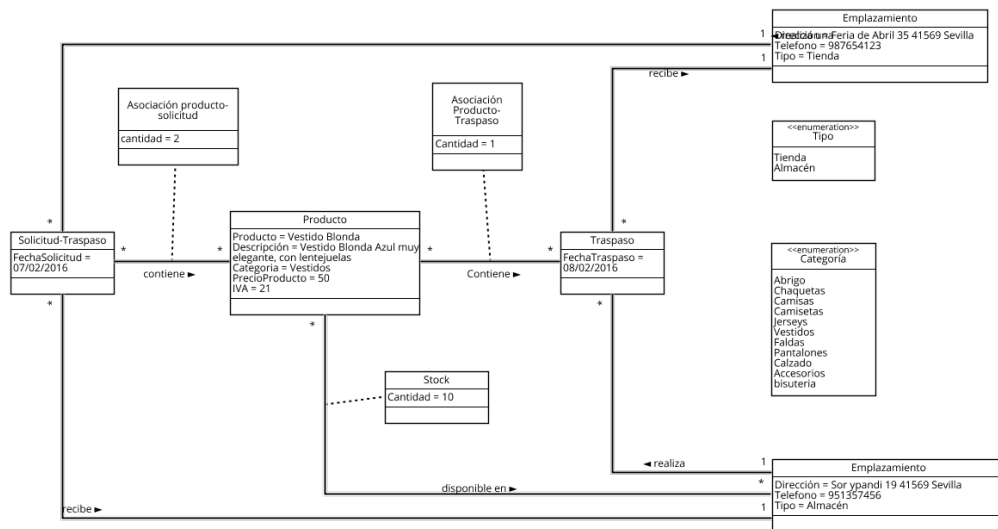


Figura 11: UML Traspaso Tienda - Almacén

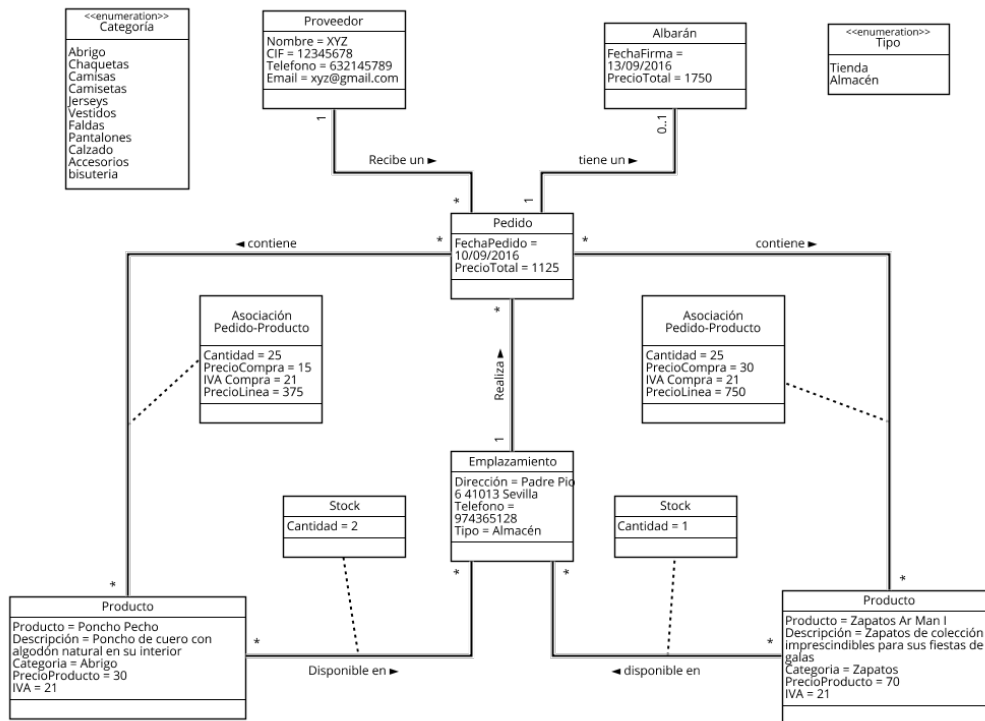


Figura 12: UML Pedido Tienda - Proveedor

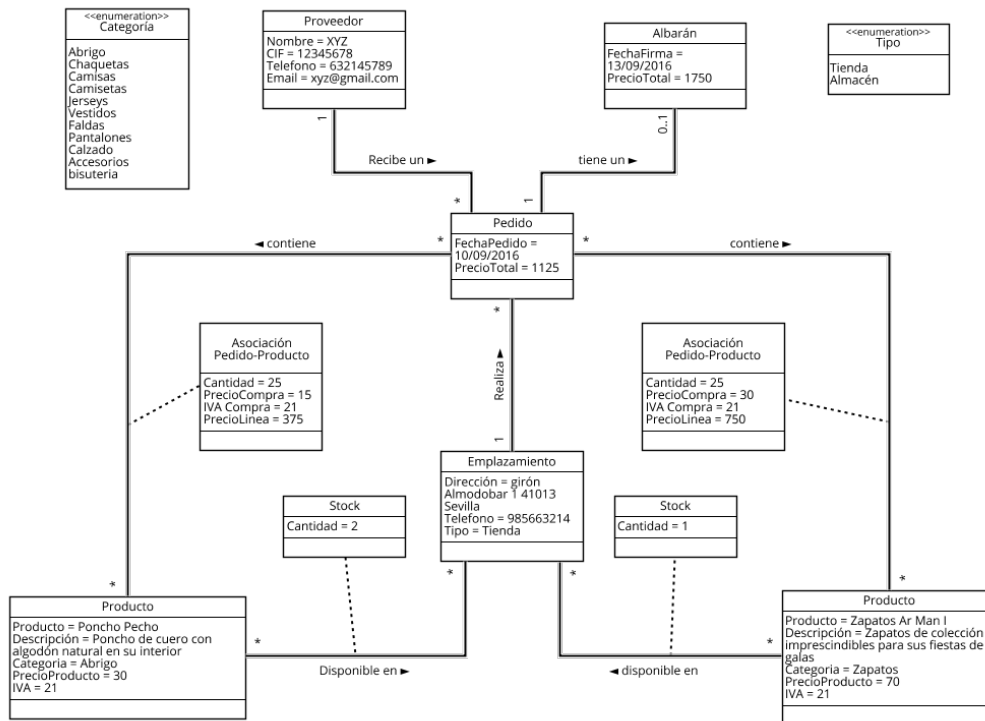


Figura 13: UML Pedido Almacén - Proveedor

## 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

<b>PA-01</b>					x
<b>PA-02</b>	x			x	
<b>PA-03</b>	x				
<b>PA-04</b>		x		x	
<b>PA-05</b>			x		x
<b>PA-06</b>		x			
<b>PA-07</b>		x			x
<b>PA-08</b>			x		
<b>PA-09</b>			x	x	
<b>PA-10</b>	x				
<b>PA-11</b>		x			
	<b>01</b>	<b>02</b>	<b>03</b>	<b>04</b>	<b>05</b>

## 8.2. Pruebas de aceptación frente a escenarios de prueba

Cuadro 4: Escenarios

<b>Escenario 01</b>				X		X	X		X		X
<b>Escenario 02</b>				X		X	X		X		X
<b>Escenario 03</b>				X	X			X	X		
<b>Escenario 04</b>				X	X			X	X		
<b>Escenario 05</b>	X	X		X	X						
<b>Escenario 06</b>	X	X	X	X	X					X	
	<b>01</b>	<b>02</b>	<b>03</b>	<b>04</b>	<b>05</b>	<b>06</b>	<b>07</b>	<b>08</b>	<b>09</b>	<b>10</b>	<b>11</b>

## 8.3. Tipos de UML frente a Requisitos

Cuadro 5: Requisitos de informacion

<b>Venta</b>	x	x	x	x	x	x					
<b>Traspaso - Pedido</b>				x	x	x	x	x	x	x	x
	<b>01</b>	<b>02</b>	<b>03</b>	<b>04</b>	<b>05</b>	<b>06</b>	<b>07</b>	<b>08</b>	<b>09</b>	<b>10</b>	<b>11</b>

**Cuadro 6: Reglas de negocio**

<b>Venta</b>	x			x	x
<b>Traspaso - Pedido</b>		x	x		x
	<b>01</b>	<b>02</b>	<b>03</b>	<b>04</b>	<b>05</b>

**Cuadro 7: Requisitos Funcionales**

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

## 9. Modelos relacionales

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

### 9.1. 3FN Venta

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

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

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

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

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

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

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



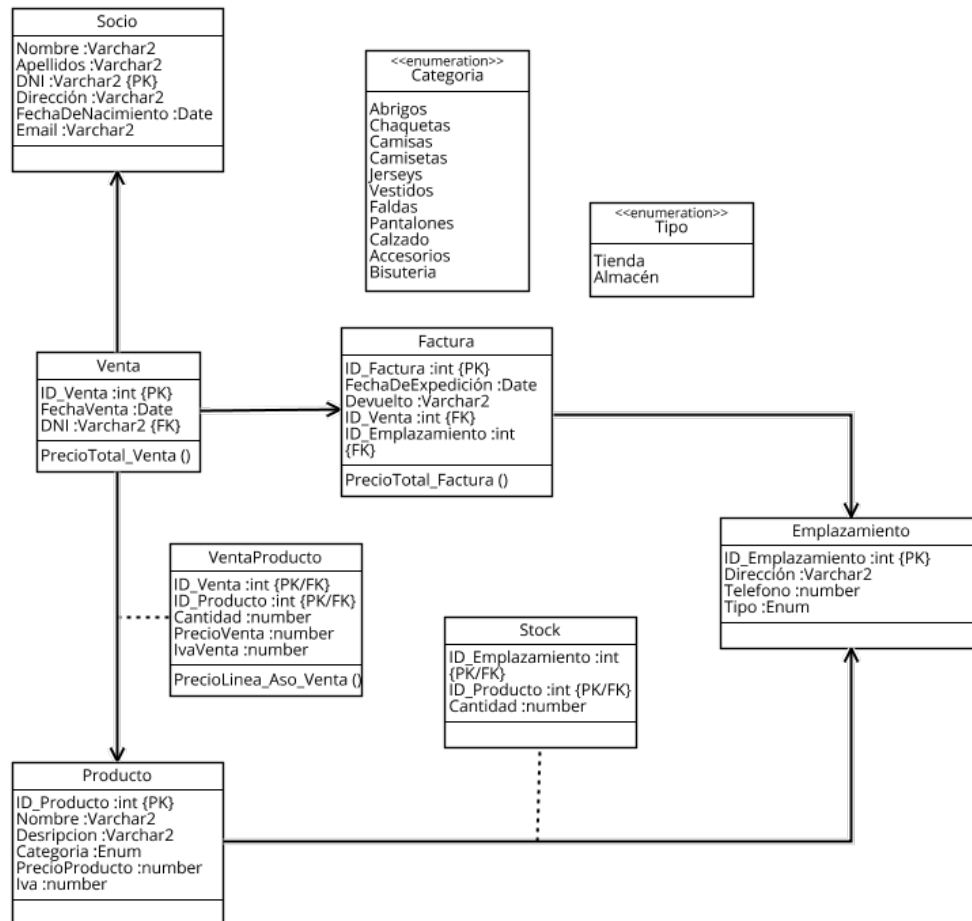


Figura 14: 3FN Venta

## 9.2. 3FN Traspaso - Pedido

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

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

key y un ID\_PROVEEDOR como foreign key debido a la relación con Proveedor.

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

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

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

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

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

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

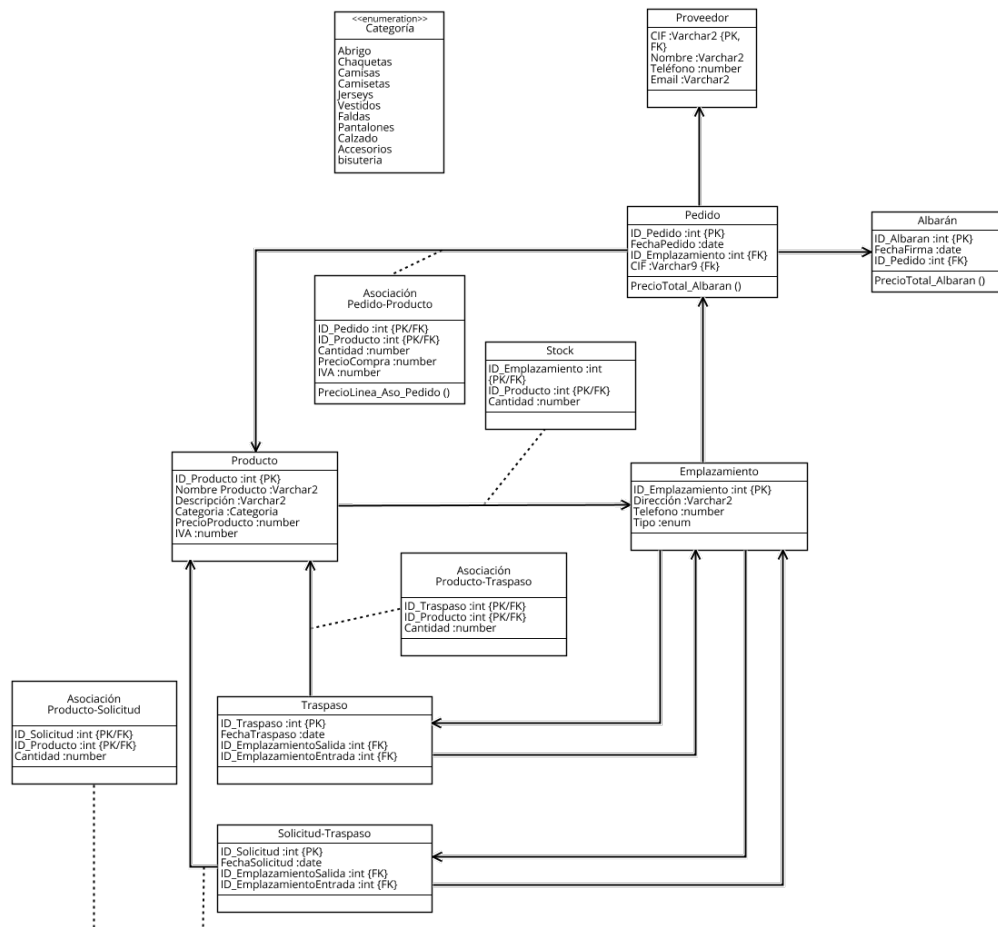


Figura 15: 3FN Traspaso - Pedido

## 10. Código SQL de la base de datos

### 10.1. Tablas

```

1 DROP TABLE ALBARAN;
2 DROP TABLE ASOCIACION_PRODUCTO_TRASPASO;
3 DROP TABLE ASOCIACION_PRODUCTO_SOLICITUD;
4 DROP TABLE ASOCIACION_PEDIDO_PRODUCTO;
5 DROP TABLE ASOCIACION_VENTA_PRODUCTO;
6 DROP TABLE FACTURA;
7 DROP TABLE VENTA;
8 DROP TABLE PEDIDO;
  
```

```

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

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

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

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

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

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

57|
58| CREATE TABLE PEDIDO(
59|     ID_Pedido int PRIMARY KEY,
60|     FechaPedido DATE NOT NULL,
61|     ID_Emplazamiento int,
62|     CIF VARCHAR2(9),
63|     FOREIGN KEY (CIF) REFERENCES PROVEEDOR,
64|     FOREIGN KEY (ID_Emplazamiento) REFERENCES EMPLAZAMIENTO
65| );

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

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

85| CREATE TABLE STOCK(

```

```

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

```

```

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

```

```

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

```

sql/tablas.sql

i

## 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 /
17 CREATE OR REPLACE PROCEDURE Producto_Nuevo(
    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 /
43 CREATE OR REPLACE PROCEDURE PRODUCTO_TRASPASO_Nuevo (
    p_ID_Traspaso IN ASOCIACION_PRODUCTO_TRASPASO.ID_Traspaso%TYPE,
    p_ID_Producto IN ASOCIACION_PRODUCTO_TRASPASO.ID_Producto%TYPE,
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 /
53 CREATE OR REPLACE PROCEDURE PRODUCTO_SOLICITUD_Nuevo (
    p_ID_Solicitud IN ASOCIACION_PRODUCTO_SOLICITUD.ID_Solicitud%TYPE,
55    p_ID_Producto IN ASOCIACION_PRODUCTO_SOLICITUD.ID_Producto%TYPE,
    p_Cantidad IN ASOCIACION_PRODUCTO_SOLICITUD.Cantidad%TYPE)

```

```

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

```



```

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

141 /

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

153 /

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

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

167 /

169 /

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

179 /

181 /

183 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,
185     P_Cantidad IN ASOCIACION_VENTA_PRODUCTO.Cantidad %TYPE,
    P_PrecioVenta IN ASOCIACION_VENTA_PRODUCTO.PrecioVenta %TYPE,
187     P_IvaVenta IN ASOCIACION_VENTA_PRODUCTO.IvaVenta %TYPE
    )
189 IS
    BEGIN
191     INSERT INTO ASOCIACION_VENTA_PRODUCTO
        VALUES(P_ID_Venta , P_ID_Producto ,P_Cantidad ,P_PrecioVenta ,P_IvaVenta);
193 END VENTA_PRODUCTO_Nueva;

195 /

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

203 /

205 /

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

```

```

211 END MODIFICA_PROVEEDOR_Telefono;

213 /

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

221 /

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

233 /

235 /

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

247 /

249 /

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

262 /

263 /

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

273 /

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

282 /

283 /

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

```

```

289 COMMIT WORK;
END MODIFICA_SOCIO_EMAIL;

291 /

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

299 /

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

307 /

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

315 /

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

323 /

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

331 /

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

339 /

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

345 /

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

351 /

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

```

```

363 RETURN(f_PrecioLinea);
364 END precio_A_Venta_producto;
365
366 /
367 */
368 /*
369 CREATE OR REPLACE FUNCTION precio_Venta
370 (f_ID_Venta IN VENTA.ID_Venta%TYPE)
371 RETURN NUMBER is f_PrecioTotal VENTA.PRECIO_TOTAL%TYPE;
372 BEGIN
373 select SUM(precioLinea) into f_PrecioTotal from ASOCIACION_VENTA_PRODUCTO
374 where ID_Venta = f_ID_Venta;
375 RETURN(f_PrecioTotal);
376 END precio_Venta;
377
378 /
379 */
380 /* Esta abajo con menos para metros
381 CREATE OR REPLACE FUNCTION precio_A_Pedido_producto
382 (f_ID_Pedido IN ASOCIACION_PEDIDO_PRODUCTO.ID_Pedido%TYPE,
383 f_Cantidad IN ASOCIACION_PEDIDO_PRODUCTO.Cantidad%TYPE,
384 f_PrecioCompra IN ASOCIACION_PEDIDO_PRODUCTO.PrecioCompra%TYPE)
385 RETURN NUMBER is f_PrecioLinea ASOCIACION_PEDIDO_PRODUCTO.PrecioLinea%TYPE;
386 BEGIN
387 f_PrecioLinea := f_PrecioCompra * f_Cantidad;
388 RETURN(f_PrecioLinea);
389 END precio_A_Pedido_producto;
390
391 /
392 */
393 /*
394 CREATE OR REPLACE FUNCTION precio_Pedido
395 (f_ID_Pedido IN PEDIDO.ID_Pedido%TYPE)
396 RETURN NUMBER is f_PrecioTotal PEDIDO.PRECIO_TOTAL%TYPE;
397 BEGIN
398 select SUM(PrecioLinea) into f_PrecioTotal from ASOCIACION_PEDIDO_PRODUCTO
399 where ID_Pedido = f_ID_Pedido;
400 RETURN(f_PrecioTotal);
401 END precio_Pedido;
402
403 /
404 */
405
406 CREATE OR REPLACE FUNCTION precioLinea_Aso_Pedido
407 (f_ID_PRODUCTO IN ASOCIACION_PEDIDO_PRODUCTO.ID_PRODUCTO%TYPE, f_ID_PEDIDO IN
408 ASOCIACION_PEDIDO_PRODUCTO.ID_PEDIDO%TYPE)
409 RETURN NUMBER is f_PrecioLinea ASOCIACION_PEDIDO_PRODUCTO.PRECIOCOMPRA%TYPE;
410 f_Cantidad ASOCIACION_PEDIDO_PRODUCTO.CANTIDAD%TYPE;
411 f_PrecioCompra ASOCIACION_PEDIDO_PRODUCTO.PRECIOCOMPRA%TYPE;
412 BEGIN
413 select Cantidad into f_Cantidad from ASOCIACION_PEDIDO_PRODUCTO where ID_Pedido = f_ID_PEDIDO AND
414 ID_PRODUCTO = f_ID_PRODUCTO;
415 select PrecioCompra into f_PrecioCompra from ASOCIACION_PEDIDO_PRODUCTO where ID_Pedido = f_ID_PEDIDO
416 AND ID_PRODUCTO = f_ID_PRODUCTO;
417 f_PrecioLinea := f_Cantidad * f_PrecioCompra;
418 RETURN(f_PrecioLinea);
419 END precioLinea_Aso_Pedido;
420
421 /
422
423 CREATE OR REPLACE FUNCTION precioLinea_Aso_Venta
424 (f_ID_PRODUCTO IN ASOCIACION_VENTA_PRODUCTO.ID_PRODUCTO%TYPE, f_ID_VENTA IN ASOCIACION_VENTA_PRODUCTO.
425 ID_VENTA%TYPE)
426 RETURN NUMBER is f_PrecioLinea ASOCIACION_VENTA_PRODUCTO.PRECIOVENTA%TYPE;
427 f_Cantidad ASOCIACION_VENTA_PRODUCTO.CANTIDAD%TYPE;
428 f_PrecioVenta ASOCIACION_VENTA_PRODUCTO.PRECIOVENTA%TYPE;
429 BEGIN
430 select Cantidad into f_Cantidad from ASOCIACION_VENTA_PRODUCTO where ID_VENTA = f_ID_VENTA AND
431 ID_PRODUCTO = f_ID_PRODUCTO;
432 select PrecioVenta into f_PrecioVenta from ASOCIACION_VENTA_PRODUCTO where ID_VENTA = f_ID_VENTA AND
433 ID_PRODUCTO = f_ID_PRODUCTO;
434 f_PrecioLinea := f_Cantidad * f_PrecioVenta;
435 RETURN(f_PrecioLinea);
436 END precioLinea_Aso_Venta;
437
438 /
439

```

```

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

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

445 m_id_venta ASOCIACION_VENTA_PRODUCTO.id_venta %TYPE;
m_id_producto Producto.id_producto %type;
447 m_cantidad ASOCIACION_VENTA_PRODUCTO.cantidad %TYPE;
BEGIN
449 f_precioTotal := 0;
OPEN all_prods;
451 LOOP
Fetch all_prods INTO m_id_venta, m_id_producto, m_cantidad;
453 EXIT WHEN all_prods %NOTFOUND;
if m_id_venta = f_id_venta
455 THEN
select precioVenta into precio_AUX from asociacion_venta_producto where id_venta = m_id_venta and
id_producto = m_id_producto;
457 f_precioTotal := (precio_AUX * m_cantidad) + f_precioTotal;
459 END IF;
END LOOP;
461 CLOSE all_prods;
RETURN (f_PrecioTotal);
463 END precioTotal_Venta;

465 /

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

483 /

485 CREATE OR REPLACE FUNCTION precioTotal_PEDIDO
(f_ID_PEDIDO IN ASOCIACION_PEDIDO_PRODUCTO.ID_PEDIDO %TYPE)
487 RETURN NUMBER is f_precioTotal ASOCIACION_PEDIDO_PRODUCTO.PRECIOCOMPRA %TYPE;
precio_AUX ASOCIACION_PEDIDO_PRODUCTO.PRECIOCOMPRA %TYPE;
489 CURSOR all_prods
IS
491 SELECT id_pedido, id_producto, cantidad
FROM ASOCIACION_PEDIDO_PRODUCTO
493 ORDER BY ID_producto;

495 m_id_pedido ASOCIACION_PEDIDO_PRODUCTO.id_pedido %TYPE;
m_id_producto Producto.id_producto %type;
497 m_cantidad ASOCIACION_PEDIDO_PRODUCTO.cantidad %TYPE;
BEGIN
499 f_precioTotal := 0;
OPEN all_prods;
501 LOOP
Fetch all_prods INTO m_id_pedido, m_id_producto, m_cantidad;
503 EXIT WHEN all_prods %NOTFOUND;
if m_id_pedido = f_id_pedido
505 THEN
select precioCompra into precio_AUX from ASOCIACION_PEDIDO_PRODUCTO where id_pedido = m_id_pedido
and id_producto = m_id_producto;
507

```

```

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

```

sql/funciones\_y\_procedures.sql

## 10.3. Triggers

```

/* TRIGGERS */
2
3 CREATE OR REPLACE TRIGGER descuento_socio
4   BEFORE INSERT ON factura
5   FOR EACH ROW
6   declare v_preciototal VENTA.PRECIOTOTAL%TYPE;
7   v_DNI VENTA.dni%TYPE;
8 BEGIN
9   select preciototal into v_preciototal from venta where id_Venta = :NEW.id_venta;
10  select dni into v_dni from venta where id_venta = :NEW.id_Venta;
11  IF v_DNI IS NOT NULL then
12    UPDATE venta set preciototal = v_preciototal * 0.95 where id_Venta = :NEW.id_venta;
13    :New.preciototal := v_preciototal * 0.95;
14  else
15    update venta set preciototal = v_preciototal where id_venta = :NEW.id_venta;
16  END IF;
17 END;
18
19 /
20
21 /*
22 CREATE OR REPLACE TRIGGER stock_minimo
23   AFTER INSERT OR UPDATE ON stock
24   FOR EACH ROW
25 BEGIN
26   IF :NEW.cantidad < 0
27   THEN
28     raise_application_error(-20601, :NEW.cantidad || 'No se puede realizar esta operacion');
29   END IF;
30 END;
31
32 /
33 */
34 CREATE OR REPLACE TRIGGER Inicializa_nueva_Venta
35 BEFORE INSERT ON VENTA
36 FOR EACH ROW
37 BEGIN
38   :NEW.PrecioTotal := 0;
39   :NEW.FechaVenta := SYSDATE;
40 END Comprueba_Venta;
41
42 /
43
44 CREATE OR REPLACE TRIGGER Inicializa_nuevo_Pedido
45 BEFORE INSERT ON PEDIDO
46 FOR EACH ROW
47 BEGIN
48   :NEW.PrecioTotal := 0;
49   :NEW.FechaPedido := SYSDATE;
50 END Comprueba_Pedido;
51
52 /
53
54 CREATE OR REPLACE TRIGGER Inicializa_nueva_Factura
55 BEFORE INSERT ON Factura
56 FOR EACH ROW
57 BEGIN
58   :NEW.FechaDeExpedicion := SYSDATE;
59 END Inicializa_nueva_Factura;

```

```

60 /
62
64 CREATE OR REPLACE TRIGGER modifica_stock_venta
BEFORE INSERT ON FACTURA
FOR EACH ROW
66 DECLARE
e_ID_Emplazamiento Emplazamiento.ID_Emplazamiento %TYPE;
68 v_ID_Venta Venta.ID_VENTA %TYPE;

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

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

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

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

96 /
/*
98 CREATE OR REPLACE TRIGGER inicializa_preciolinea_alv
BEFORE INSERT OR UPDATE ON ASOCIACION_VENTA_PRODUCTO
100 FOR EACH ROW
BEGIN
102 :NEW.preciolinea := :New.cantidad * :New.precioVenta;
END inicializa_preciolinea_alv;
104

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

114 /

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

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

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

```

```

134 BEGIN
136 select ID_Emplazamiento into e_ID_Emplazamiento from pedido where ID_pedido = :NEW.ID_Pedido;
137 select ID_PEDIDO into p_id_pedido from pedido where ID_pedido = :NEW.ID_Pedido;
138 OPEN all_prods;
139 LOOP
140     Fetch all_prods INTO m_id_pedido,m_id_producto,m_cantidad;
141     EXIT WHEN all_prods%NOTFOUND;
142     if m_id_pedido = p_id_pedido
143     THEN
144         update stock set cantidad = cantidad+m_cantidad where id_emplazamiento = e_ID_Emplazamiento AND
145             id_producto =m_ID_Producto;
146     END IF;
147 END LOOP;
148 CLOSE all_prods;
149 END modifica_stock_pedido;
150 /
151
152 CREATE OR REPLACE TRIGGER modifica_stock_traspaso
153 BEFORE INSERT OR UPDATE ON ASOCIACION_PRODUCTO_TRASPASO
154 FOR EACH ROW
155 DECLARE
156 e_ID_Emplazamiento_salida Emplazamiento.ID_Emplazamiento%TYPE;
157 e_ID_Emplazamiento_entrada Emplazamiento.ID_Emplazamiento%TYPE;
158 t_ID_traspaso traspaso.ID_traspaso%TYPE;
159
160 BEGIN
161 select ID_Emplazamientosalida into e_ID_Emplazamiento_salida from traspaso where ID_traspaso = :NEW.
162     ID_traspaso;
163 select ID_Emplazamientoentrada into e_ID_Emplazamiento_entrada from traspaso where ID_traspaso = :NEW
164     .ID_traspaso;
165 select ID_traspaso into t_id_traspaso from traspaso where ID_traspaso = :NEW.ID_traspaso;
166 update stock set cantidad = (cantidad- :NEW.cantidad) where id_emplazamiento =
167     e_ID_Emplazamiento_salida AND id_producto = :NEW.id_producto;
168 update stock set cantidad = (cantidad+ :NEW.cantidad) where id_emplazamiento =
169     e_ID_Emplazamiento_entrada AND id_producto = :NEW.id_producto;
170
171 END modifica_stock_traspaso;
172 /
173
174 CREATE OR REPLACE TRIGGER Inicializa_Nuevo_Pedido
175 BEFORE INSERT ON Pedido
176 FOR EACH ROW
177 BEGIN
178 :NEW.PrecioTotal := 0;
179 :NEW.FechaPedido := SYSDATE;
180 END Inicializa_Nuevo_Pedido;
181 /
182
183 CREATE OR REPLACE TRIGGER Inicializa_Nueva_ST
184 BEFORE INSERT ON Solicitud_Traspaso
185 FOR EACH ROW
186 BEGIN
187 :NEW.FechaSolicitud := SYSDATE;
188 END Inicializa_Nueva_ST;
189 /
190
191 CREATE OR REPLACE TRIGGER Inicializa_Nuevo_Traspaso
192 BEFORE INSERT ON Traspaso
193 FOR EACH ROW
194 BEGIN
195 :NEW.FechaTraspaso := SYSDATE;
196 END Inicializa_Nuevo_Traspaso;
197 /
198
199 create or replace TRIGGER solicitud_stock_minimo
200 BEFORE INSERT ON asociacion_producto_solicitud
201 FOR EACH ROW
202 DECLARE
203 e_id_emplazamientoentrada SOLICITUD_TRASPASO.ID_EMPLAZAMIENTOENTRADA%TYPE;
204 e_cantidad Stock.cantidad%TYPE;

```



```

BEGIN
206  SELECT id_emplazamientoentrada into e_id_emplazamientoentrada from solicitud_traspaso where
      id_solicitud = :NEW.id_solicitud;
      select cantidad into e_cantidad from stock where id_emplazamiento = e_id_emplazamientoentrada and
      id_producto = :NEW.id_producto;
208  if (e_cantidad - :NEW.cantidad) <=5
      THEN raise_application_error(-20601, :NEW.cantidad || 'No se permite la solicitud, el otro
      emplazamiento alcanzara su stock minimo');
210  END IF;
END;
212
214  /
/*
216  create or replace TRIGGER inicializa_preciolinea_alp
      BEFORE INSERT OR UPDATE ON ASOCIACION_PEDIDO_PRODUCTO
      FOR EACH ROW
218  BEGIN
      :NEW.preciolinea := :New.cantidad * :New.preciocompra;
220  END inicializa_preciolinea_alp;
      /
222  */
224  create or replace TRIGGER inicializa_preciototal_albaran
      BEFORE INSERT OR UPDATE ON Albaran
      for each row
226  DECLARE
      p_preciototal pedido.preciototal %TYPE;
228  begin
      select preciototal into p_preciototal from pedido where id_pedido = :New.id_pedido;
230  :New.preciototal := p_preciototal;
      END inicializa_preciototal_albaran;
232
234  /
236  create or replace TRIGGER inicializa_preciototal_pedido
      BEFORE INSERT OR UPDATE ON ASOCIACION_pedido_producto
      for each row
238  begin
      UPDATE pedido set preciototal = PRECIOTOTAL + (:New.cantidad * :New.preciocompra) where id_pedido =
      :New.id_pedido;
240  END inicializa_preciototal_pedido;
242  /
244  create or replace trigger inicializa_precioventa_apv
      before insert on asociacion_venta_producto
246  for each row
      declare
248  pPrecio producto.precioproducto %TYPE;
      begin
250  select precioproducto into pPrecio from producto where id_producto = :New.id_producto;
      :NEW.precioventa := pPrecio;
252  END inicializa_precioventa_apv;
254  /
256  create or replace trigger inicializa_IVA_apv
      before insert on asociacion_venta_producto
258  for each row
      declare
260  p_IVA PRODUCTO.IVA %TYPE;
      begin
262  select Iva into p_IVA from producto where id_producto = :New.id_producto;
      :NEW.IVAVENTA := p_IVA;
264  END inicializa_IVA_apv;
266  /
268  create or replace trigger inicializa_IVA_APP
      before insert on asociacion_pedido_producto
270  for each row
      declare
272  p_IVA PRODUCTO.IVA %TYPE;
      begin
274  select IVA into p_IVA from producto where id_producto = :New.id_producto;
      :NEW.IVA := p_IVA;
276  END inicializa_IVA_APP;

```

```

278 /
280 create or replace trigger devolucion
before update on factura
282 for each row
begin
284   if(sysdate - :old.fechadeexpedicion)>30 then
       raise_application_error(-20601, :NEW.fechadeexpedicion || 'No se permite la devolucion, han pasado
       mas de 30 dias');
286   END IF;
END devolucion;
288 /
/*
290 create or replace trigger mensaje
before update of cantidad on stock
292 for each row
DECLARE
294   lines dbms_output.chararr;
   num_lines number;
296 BEGIN
298   if (:old.cantidad - :new.cantidad)<5 then
       -- enable the buffer with default size 20000
300   dbms_output.enable;
302   dbms_output.put_line('Hello Reader!');
   dbms_output.put_line('Hope you have enjoyed the tutorials!');
304   dbms_output.put_line('Have a great time exploring pl/sql!');
306   num_lines := 3;
308   dbms_output.get_lines(lines, num_lines);
310   FOR i IN 1..num_lines LOOP
       dbms_output.put_line(lines(i));
312   END LOOP;
   end IF;
314 END mensaje;
/
316 */

```

sql/triggers.sql

## 10.4. Pruebas

```

/* FUNCION AUXILIAR */
2 CREATE OR REPLACE FUNCTION EQUALS(salida BOOLEAN, salidaEsperada BOOLEAN)
RETURN VARCHAR2 AS
4 BEGIN
   IF (salida = salidaEsperada) THEN
6     RETURN 'EXITO';
   ELSE
8     RETURN 'FALLO';
   END IF;
10 END EQUALS;
12 /
14 CREATE OR REPLACE PROCEDURE PRINTR(nombre_prueba VARCHAR2, salida BOOLEAN,
   salidaEsperada BOOLEAN)
16 IS BEGIN
   DBMS_OUTPUT.put_line(nombre_prueba || ':' || EQUALS(salida, salidaEsperada));
18 END PRINTR;
20 /
22 /* DEFINICION PRUEBAS */
CREATE OR REPLACE PACKAGE PRUEBAS_SOCIO AS
24   PROCEDURE inicializar;
   PROCEDURE insertar
26     (nombre_prueba VARCHAR2, i_nombre VARCHAR2, i_apellidos VARCHAR2,

```

```

28         i_DNI VARCHAR2, i_direccion VARCHAR2, i_nacimiento DATE, i_email VARCHAR2
        ,salidaEsperada BOOLEAN);
29     PROCEDURE actualizar
30     (nombre_prueba VARCHAR2, a_DNI VARCHAR2, a_direccion VARCHAR2,
        a_email VARCHAR2, salidaEsperada BOOLEAN);
31     PROCEDURE eliminar
32     (nombre_prueba VARCHAR2, e_DNI VARCHAR2, salidaEsperada BOOLEAN);
33 END PRUEBAS_SOCIO;
34
35 /
36
37 CREATE OR REPLACE PACKAGE PRUEBAS_PROVEEDOR AS
38     PROCEDURE inicializar;
39     PROCEDURE insertar
40     (nombre_prueba VARCHAR2, i_CIF VARCHAR2, i_nombre VARCHAR2,
41      i_telefono INT, i_email VARCHAR2, salidaEsperada BOOLEAN);
42     PROCEDURE actualizar
43     (nombre_prueba VARCHAR2, a_cif VARCHAR2, a_nombre VARCHAR2,
44      a_telefono INT, a_email VARCHAR2, salidaEsperada BOOLEAN);
45     PROCEDURE eliminar
46     (nombre_prueba VARCHAR2, e_CIF VARCHAR2, salidaEsperada BOOLEAN);
47 END PRUEBAS_PROVEEDOR;
48
49 /
50
51 CREATE OR REPLACE PACKAGE PRUEBAS_EMPLAZAMIENTO AS
52     PROCEDURE inicializar;
53     PROCEDURE insertar
54     (nombre_prueba VARCHAR2, i_direccion VARCHAR2, i_telefono INT,
55      i_tipo VARCHAR2, salidaEsperada BOOLEAN);
56     PROCEDURE actualizar
57     (nombre_prueba VARCHAR2, a_id_emplazamiento INT, a_direccion VARCHAR2,
58      a_telefono INT, salidaEsperada BOOLEAN);
59     PROCEDURE eliminar
60     (nombre_prueba VARCHAR2, e_id_emplazamiento INT, salidaEsperada BOOLEAN);
61 END PRUEBAS_EMPLAZAMIENTO;
62
63 /
64
65 CREATE OR REPLACE PACKAGE PRUEBAS_ALBARAN AS
66     PROCEDURE inicializar;
67     PROCEDURE insertar
68     (nombre_prueba VARCHAR2, i_fecha DATE, i_id_pedido INT, salidaEsperada BOOLEAN);
69     PROCEDURE eliminar
70     (nombre_prueba VARCHAR2, e_id_albaran INT, salidaEsperada BOOLEAN);
71 END PRUEBAS_ALBARAN;
72
73 /
74
75 CREATE OR REPLACE PACKAGE PRUEBAS_PEDIDO AS
76     PROCEDURE inicializar;
77     PROCEDURE insertar
78     (nombre_prueba VARCHAR2, i_fecha DATE, i_id_emplazamiento INT, i_cif VARCHAR2, salidaEsperada
        BOOLEAN);
79     PROCEDURE eliminar
80     (nombre_prueba VARCHAR2, e_id_pedido INT, salidaEsperada BOOLEAN);
81 END PRUEBAS_PEDIDO;
82
83 /
84
85 CREATE OR REPLACE PACKAGE PRUEBAS_PRODUCTO AS
86     PROCEDURE inicializar;
87     PROCEDURE insertar
88     (nombre_prueba VARCHAR2, i_nombre VARCHAR2, i_descripcion VARCHAR2,
89      i_categoria VARCHAR2, i_precioProducto INT, i_iva INT, salidaEsperada BOOLEAN);
90     PROCEDURE actualizar
91     (nombre_prueba VARCHAR2, a_id_producto INT, a_descripcion VARCHAR2,
92      a_precioProducto INT, a_iva INT, salidaEsperada BOOLEAN);
93     PROCEDURE eliminar
94     (nombre_prueba VARCHAR2, e_id_producto INT, salidaEsperada BOOLEAN);
95 END PRUEBAS_PRODUCTO;
96
97 /
98
99 CREATE OR REPLACE PACKAGE PRUEBAS_VENTA AS
100     PROCEDURE inicializar;
101     PROCEDURE insertar
102

```

```

104         (nombre_prueba VARCHAR2, i_fecha DATE, i_dni VARCHAR2,
            salidaEsperada BOOLEAN);
106     PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_venta INT, salidaEsperada BOOLEAN);
108     /*procedure descuento
        (nombre_prueba VARCHAR2, v_id_venta int,v_precioTotal number,v_dni_socio varchar2,
            salidaEsperada BOOLEAN);*/
110 END PRUEBAS_VENTA;
112 /
114 CREATE OR REPLACE PACKAGE PRUEBAS_SOLICITUD_TRASPASO AS
116     PROCEDURE inicializar;
118     PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_fechaSolicitud DATE, i_id_emplazamientoSalida INT,
            i_id_emplazamientoEntrada INT, salidaEsperada BOOLEAN);
120     PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_solicitudTraspaso INT, salidaEsperada BOOLEAN);
122 END PRUEBAS_SOLICITUD_TRASPASO;
124 /
126 CREATE OR REPLACE PACKAGE PRUEBAS_FACTURA AS
128     PROCEDURE inicializar;
130     PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_fechaDeExpedicion DATE, i_devuelto VARCHAR2,
            i_id_venta INT, i_id_emplazamiento INT, salidaEsperada BOOLEAN);
132     PROCEDURE actualizar
        (nombre_prueba VARCHAR2, a_id_factura INT,a_devuelto VARCHAR2,
            salidaEsperada BOOLEAN);
134     PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_factura INT, salidaEsperada BOOLEAN);
136 END PRUEBAS_FACTURA;
138 /
140 CREATE OR REPLACE PACKAGE PRUEBAS_TRASPASO AS
142     PROCEDURE inicializar;
144     PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_fechaTraspaso DATE, i_id_emplazamientoSalida INT,
            i_id_emplazamientoEntrada INT, salidaEsperada BOOLEAN);
146     PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_Traspaso INT, salidaEsperada BOOLEAN);
148 END PRUEBAS_TRASPASO;
150 /
152 CREATE OR REPLACE PACKAGE PRUEBAS_A_VENTA_PRODUCTO AS
154     PROCEDURE inicializar;
156     PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_id_venta INT, i_id_producto INT,
            i_cantidad NUMBER, i_precioVenta NUMBER, i_ivaVenta NUMBER,
            salidaEsperada BOOLEAN);
158     PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_venta INT, e_id_producto INT, salidaEsperada BOOLEAN);
160 END PRUEBAS_A_VENTA_PRODUCTO;
162 /
164 CREATE OR REPLACE PACKAGE PRUEBAS_STOCK AS
166     PROCEDURE inicializar;
168     PROCEDURE insertar
        (nombre_prueba VARCHAR2, i_id_emplazamiento INT, i_id_producto INT,
            i_cantidad INT, salidaEsperada BOOLEAN);
170     PROCEDURE actualizar
        (nombre_prueba VARCHAR2, a_id_emplazamiento INT, a_id_producto INT,
            a_cantidad INT, salidaEsperada BOOLEAN);
172     PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,
            salidaEsperada BOOLEAN);
174     PROCEDURE stock_correcto
        (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,stock_previo int, cantidad
            INT, salidaEsperada BOOLEAN);
176     PROCEDURE stock_correcto_p
        (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,stock_previo int, cantidad
            INT, salidaEsperada BOOLEAN);
178     PROCEDURE stock_correcto_t

```

```

178         (nombre_prueba VARCHAR2, e_id_emplazamiento_envia int, e_id_emplazamiento_recibe int,
stock_previo_envia int, stock_previo_recibe int,
179         cantidad int, e_id_producto int, salidaEsperada BOOLEAN);
180     END PRUEBA_STOCK;
181 /
182
183     CREATE OR REPLACE PACKAGE PRUEBAS_A_PRODUCTO_SOLICITUD AS
184     PROCEDURE inicializar;
185     PROCEDURE insertar
186         (nombre_prueba VARCHAR2, i_id_producto INT, i_id_Solicitud INT,
187          i_cantidad NUMBER, salidaEsperada BOOLEAN);
188     PROCEDURE eliminar
189         (nombre_prueba VARCHAR2, e_id_Solicitud INT, e_id_producto INT, salidaEsperada BOOLEAN);
190     END PRUEBAS_A_PRODUCTO_SOLICITUD;
191 /
192
193     CREATE OR REPLACE PACKAGE PRUEBAS_A_PRODUCTO_TRASPASO AS
194     PROCEDURE inicializar;
195     PROCEDURE insertar
196         (nombre_prueba VARCHAR2, i_id_producto INT, i_id_Traspaso INT,
197          i_cantidad NUMBER, salidaEsperada BOOLEAN);
198     PROCEDURE eliminar
199         (nombre_prueba VARCHAR2, e_id_Traspaso INT, e_id_producto INT, salidaEsperada BOOLEAN);
200     END PRUEBAS_A_PRODUCTO_TRASPASO;
201 /
202
203     CREATE OR REPLACE PACKAGE PRUEBAS_A_PEDIDO_PRODUCTO AS
204     PROCEDURE inicializar;
205     /*PROCEDURE insertar
206         (nombre_prueba VARCHAR2, i_id_pedido INT, i_id_producto INT,
207          i_cantidad NUMBER, i_precioCompra number, i_iva INT,
208          salidaEsperada BOOLEAN);*/
209     PROCEDURE eliminar
210         (nombre_prueba VARCHAR2, e_id_pedido INT, e_id_producto INT,
211          salidaEsperada BOOLEAN);
212     END PRUEBAS_A_PEDIDO_PRODUCTO;
213 /
214
215     /*
216     CREATE OR REPLACE PACKAGE PRUEBAS_FUNCIONES AS
217     PROCEDURE ganancias
218         (nombre_prueba VARCHAR2, mes NUMBER, anyo NUMBER, esperado number, salidaEsperada BOOLEAN);
219     PROCEDURE precioLinea_A_Pedido
220         (nombre_prueba VARCHAR2, ID_PRODUCTO NUMBER, ID_PEDIDO NUMBER, esperado number,
221          salidaEsperada BOOLEAN);
222     PROCEDURE precioLinea_A_Venta
223         (nombre_prueba VARCHAR2, ID_PRODUCTO NUMBER, ID_VENTA NUMBER, esperado number, salidaEsperada
224          BOOLEAN);
225     END PRUEBAS_FUNCIONES;
226     */
227 /
228     /* CUERPOS DE PRUEBAS */
229
230     CREATE OR REPLACE PACKAGE BODY PRUEBAS_SOCIO AS
231     PROCEDURE inicializar AS
232     BEGIN
233         DELETE FROM SOCIO;
234     END inicializar;
235
236     PROCEDURE insertar
237     (nombre_prueba VARCHAR2, i_nombre VARCHAR2, i_apellidos VARCHAR2,
238      i_DNI VARCHAR2, i_direccion VARCHAR2, i_nacimiento DATE, i_email VARCHAR2
239      ,salidaEsperada BOOLEAN) AS
240     salida BOOLEAN := true;
241     actual socio%ROWTYPE;
242     BEGIN
243         INSERT INTO socio VALUES (i_nombre, i_apellidos, i_direccion, i_nacimiento,
244          i_email, i_DNI);
245
246         SELECT * INTO actual FROM SOCIO WHERE DNI = i_DNI;
247
248         IF (actual.nombre<>i_nombre) OR (actual.apellidos<>i_apellidos) OR (actual.DNI<>i_DNI) OR (
249          actual.direccion<>i_direccion) OR (actual.fechadenacimiento<>i_nacimiento) OR (actual.email<>

```

```

250         i_email) THEN
251             salida := false;
252         END IF;
253
254         PRINTR(nombre_prueba, salida, salidaEsperada);
255
256     EXCEPTION
257     WHEN OTHERS THEN
258         PRINTR(nombre_prueba, false, salidaEsperada);
259         ROLLBACK;
260 END insertar;
261
262 PROCEDURE actualizar
263     (nombre_prueba VARCHAR2, a_DNI VARCHAR2, a_direccion VARCHAR2,
264     a_email VARCHAR2, salidaEsperada BOOLEAN) AS
265     salida BOOLEAN := true;
266     actual socio%ROWTYPE;
267 BEGIN
268     UPDATE SOCIO SET DIRECCION = a_direccion, EMAIL = a_email where DNI = a_DNI;
269
270     SELECT * INTO actual FROM SOCIO WHERE DNI = a_DNI;
271
272     IF (actual.direccion<>a_direccion) OR (actual.email<>a_email) THEN
273         salida := false;
274     END IF;
275
276     PRINTR(nombre_prueba, salida, salidaEsperada);
277
278     EXCEPTION
279     WHEN OTHERS THEN
280         PRINTR(nombre_prueba, false, salidaEsperada);
281         ROLLBACK;
282 END actualizar;
283
284 PROCEDURE eliminar
285     (nombre_prueba VARCHAR2, e_DNI VARCHAR2, salidaEsperada BOOLEAN) AS
286     salida BOOLEAN := true;
287     cantidad INT;
288 BEGIN
289     DELETE FROM SOCIO WHERE DNI = e_DNI;
290
291     SELECT COUNT(*) INTO cantidad FROM SOCIO WHERE DNI = e_DNI;
292
293     IF (cantidad<>0) THEN
294         salida := false;
295     END IF;
296
297     PRINTR(nombre_prueba, salida, salidaEsperada);
298
299     EXCEPTION
300     WHEN OTHERS THEN
301         PRINTR(nombre_prueba, false, salidaEsperada);
302         ROLLBACK;
303 END eliminar;
304 END PRUEBAS_SOCIO;
305
306 /
307
308 CREATE OR REPLACE PACKAGE BODY PRUEBAS_PROVEEDOR AS
309
310     PROCEDURE inicializar AS
311     BEGIN
312         DELETE FROM PROVEEDOR;
313     END inicializar;
314
315     PROCEDURE insertar
316     (nombre_prueba VARCHAR2, i_CIF VARCHAR2, i_nombre VARCHAR2,
317     i_telefono INT, i_email VARCHAR2, salidaEsperada BOOLEAN) AS
318     salida BOOLEAN := true;
319     actual proveedor%ROWTYPE;
320 BEGIN
321     INSERT INTO PROVEEDOR VALUES (i_CIF, i_nombre, i_telefono, i_email);
322
323     SELECT * INTO actual FROM PROVEEDOR WHERE CIF = i_CIF;
324
325     IF (actual.nombre<>i_nombre) OR (actual.telefono<>i_telefono) OR (actual.CIF<>i_CIF) OR (
326     actual.email<>i_email) THEN

```

```

326         salida := false;
327     END IF;
328
329     PRINTR(nombre_prueba, salida, salidaEsperada);
330
331     EXCEPTION
332     WHEN OTHERS THEN
333         PRINTR(nombre_prueba, false, salidaEsperada);
334         ROLLBACK;
335 END insertar;
336
337 PROCEDURE actualizar
338 (nombre_prueba VARCHAR2, a_cif VARCHAR2, a_nombre VARCHAR2,
339 a_telefono INT, a_email VARCHAR2, salidaEsperada BOOLEAN) AS
340 salida BOOLEAN := true;
341 actual proveedor%ROWTYPE;
342 BEGIN
343     UPDATE PROVEEDOR SET NOMBRE = a_nombre, TELEFONO = a_telefono, EMAIL = a_email where CIF
344 = a_cif;
345
346     SELECT * INTO actual FROM PROVEEDOR WHERE CIF = a_cif;
347
348     IF (actual.nombre<>a_nombre) OR
349        (actual.telefono<>a_telefono) OR
350        (actual.email<>a_email) THEN
351         salida := false;
352     END IF;
353
354     PRINTR(nombre_prueba, salida, salidaEsperada);
355
356     EXCEPTION
357     WHEN OTHERS THEN
358         PRINTR(nombre_prueba, false, salidaEsperada);
359         ROLLBACK;
360 END actualizar;
361
362 PROCEDURE eliminar
363 (nombre_prueba VARCHAR2, e_CIF VARCHAR2, salidaEsperada BOOLEAN) AS
364 salida BOOLEAN := true;
365 cantidad INT;
366 BEGIN
367     DELETE FROM PROVEEDOR WHERE CIF = e_CIF;
368
369     SELECT COUNT(*) INTO cantidad FROM PROVEEDOR WHERE CIF = e_cif;
370
371     IF (cantidad<>0) THEN
372         salida := false;
373     END IF;
374
375     PRINTR(nombre_prueba, salida, salidaEsperada);
376
377     EXCEPTION
378     WHEN OTHERS THEN
379         PRINTR(nombre_prueba, false, salidaEsperada);
380         ROLLBACK;
381 END eliminar;
382 END PRUEBAS_PROVEEDOR;
383 /
384
385 CREATE OR REPLACE PACKAGE BODY PRUEBAS_EMPLAZAMIENTO AS
386
387     PROCEDURE inicializar AS
388
389     BEGIN
390         DELETE FROM EMPLAZAMIENTO;
391     END inicializar;
392
393     PROCEDURE insertar
394 (nombre_prueba VARCHAR2, i_direccion VARCHAR2, i_telefono INT,
395 i_tipo VARCHAR2, salidaEsperada BOOLEAN) AS
396 salida BOOLEAN := true;
397 actual emplazamiento%ROWTYPE;
398 w_ID_Emplazamiento INT;
399 BEGIN
400     INSERT INTO emplazamiento VALUES(null,i_direccion, i_telefono, i_tipo);

```

```

402      w_ID_Emplazamiento := S_ID_Emplazamiento.currval;
403      SELECT * INTO actual FROM EMPLAZAMIENTO WHERE ID_EMPLAZAMIENTO = w_ID_Emplazamiento;
404
405      IF (actual.direccion<>i_direccion) OR (actual.telefono<>i_telefono) OR (actual.tipo<>i_tipo)
406      THEN
407          salida := false;
408      END IF;
409
410      PRINTR(nombre_prueba, salida, salidaEsperada);
411
412      EXCEPTION
413      WHEN OTHERS THEN
414          PRINTR(nombre_prueba, false, salidaEsperada);
415          ROLLBACK;
416      END insertar;
417
418      PROCEDURE actualizar
419      (nombre_prueba VARCHAR2, a_id_emplazamiento INT, a_direccion VARCHAR2,
420      a_telefono INT, salidaEsperada BOOLEAN) AS
421      salida BOOLEAN := true;
422      actual emplazamiento %ROWTYPE;
423      BEGIN
424          UPDATE emplazamiento SET DIRECCION = a_direccion, TELEFONO = a_telefono where
425          ID_EMPLAZAMIENTO = a_id_emplazamiento;
426
427          SELECT * INTO actual FROM EMPLAZAMIENTO WHERE ID_EMPLAZAMIENTO = a_id_emplazamiento;
428
429          IF (actual.direccion<>a_direccion) OR
430          (actual.telefono<>a_telefono) THEN
431              salida := false;
432          END IF;
433
434          PRINTR(nombre_prueba, salida, salidaEsperada);
435
436          EXCEPTION
437          WHEN OTHERS THEN
438              PRINTR(nombre_prueba, false, salidaEsperada);
439              ROLLBACK;
440      END actualizar;
441
442      PROCEDURE eliminar
443      (nombre_prueba VARCHAR2, e_id_emplazamiento INT, salidaEsperada BOOLEAN) AS
444      salida BOOLEAN := true;
445      cantidad INT;
446      BEGIN
447          DELETE FROM EMPLAZAMIENTO WHERE ID_EMPLAZAMIENTO = e_id_emplazamiento;
448
449          SELECT COUNT(*) INTO cantidad FROM EMPLAZAMIENTO WHERE ID_EMPLAZAMIENTO =
450          e_id_emplazamiento;
451
452          IF (cantidad<>0) THEN
453              salida := false;
454          END IF;
455
456          PRINTR(nombre_prueba, salida, salidaEsperada);
457
458          EXCEPTION
459          WHEN OTHERS THEN
460              PRINTR(nombre_prueba, false, salidaEsperada);
461              ROLLBACK;
462      END eliminar;
463      END PRUEBAS_EMPLAZAMIENTO;
464
465      /
466
467      CREATE OR REPLACE PACKAGE BODY PRUEBAS_PRODUCTO AS
468
469      PROCEDURE inicializar AS
470      BEGIN
471          DELETE FROM PRODUCTO;
472      END inicializar;
473
474      PROCEDURE insertar
475      (nombre_prueba VARCHAR2, i_nombre VARCHAR2, i_descripcion VARCHAR2,
476      i_categoria VARCHAR2, i_precioProducto INT, i_iva INT, salidaEsperada BOOLEAN) AS
477      salida BOOLEAN := true;
478      actual producto %ROWTYPE;

```



```

476 w_ID_Producto INT;
477 BEGIN
478     INSERT INTO producto VALUES(null,i_nombre, i_descripcion, i_categoria, i_precioProducto,
479     i_iva);
480
481     w_ID_Producto := S_ID_Producto.currval;
482     SELECT * INTO actual FROM PRODUCTO WHERE ID_Producto = w_ID_Producto;
483
484     IF (actual.nombre<>i_nombre) OR (actual.descripcion<>i_descripcion) OR (actual.categoria<>
485     i_categoria) OR (actual.precioProducto<>i_precioProducto) OR (actual.iva<>i_iva) THEN
486         salida := false;
487     END IF;
488
489     PRINTR(nombre_prueba, salida, salidaEsperada);
490
491     EXCEPTION
492     WHEN OTHERS THEN
493         PRINTR(nombre_prueba, false, salidaEsperada);
494         ROLLBACK;
495 END insertar;
496
497 PROCEDURE actualizar
498 (nombre_prueba VARCHAR2, a_id_producto INT, a_descripcion VARCHAR2,
499 a_precioProducto INT, a_iva INT, salidaEsperada BOOLEAN) AS
500 salida BOOLEAN := true;
501 actual producto %ROWTYPE;
502 BEGIN
503     UPDATE producto SET DESCRIPCION = a_descripcion, PRECIOPRODUCTO = a_precioProducto, IVA =
504     a_iva
505     where ID_Producto = a_id_producto;
506
507     SELECT * INTO actual FROM PRODUCTO WHERE ID_Producto = a_id_producto;
508     IF (actual.descripcion<>a_descripcion) OR
509     (actual.precioProducto<>a_precioProducto) OR
510     (actual.iva<>a_iva) THEN
511         salida := false;
512     END IF;
513
514     PRINTR(nombre_prueba, salida, salidaEsperada);
515
516     EXCEPTION
517     WHEN OTHERS THEN
518         PRINTR(nombre_prueba, false, salidaEsperada);
519         ROLLBACK;
520 END actualizar;
521
522 PROCEDURE eliminar
523 (nombre_prueba VARCHAR2, e_id_producto INT, salidaEsperada BOOLEAN) AS
524 salida BOOLEAN := true;
525 cantidad INT;
526 BEGIN
527     DELETE FROM PRODUCTO WHERE ID_Producto = e_id_producto;
528
529     SELECT COUNT(*) INTO cantidad FROM PRODUCTO WHERE ID_Producto = e_id_producto;
530     IF (cantidad<>0) THEN
531         salida := false;
532     END IF;
533
534     PRINTR(nombre_prueba, salida, salidaEsperada);
535
536     EXCEPTION
537     WHEN OTHERS THEN
538         PRINTR(nombre_prueba, false, salidaEsperada);
539         ROLLBACK;
540 END eliminar;
541 END PRUEBAS_PRODUCTO;
542
543 /
544
545 CREATE OR REPLACE PACKAGE BODY PRUEBAS_FACTURA AS
546
547     PROCEDURE inicializar AS
548     BEGIN
549         DELETE FROM FACTURA;
550     END inicializar;
551
552     PROCEDURE insertar

```

```

550      (nombre_prueba VARCHAR2, i_fechaDeExpedicion DATE, i_devuelto VARCHAR2,
551       i_id_venta INT, i_id_emplazamiento INT, salidaEsperada BOOLEAN) AS
552      salida BOOLEAN := true;
553      actual factura%ROWTYPE;
554      w_ID_factura INT;
555      BEGIN
556          INSERT INTO factura VALUES(null, i_fechaDeExpedicion, i_devuelto, i_id_venta,
557          i_id_emplazamiento);
558          w_ID_factura := S_ID_Factura.currval;
559          SELECT * INTO actual FROM factura WHERE ID_factura = w_ID_factura;
560          IF (actual.fechaDeExpedicion<>i_fechaDeExpedicion) OR (actual.devuelto<>i_devuelto) OR (
561          actual.id_venta<>i_id_venta) OR (actual.id_emplazamiento<>i_id_emplazamiento) THEN
562              salida := false;
563          END IF;
564          PRINTR(nombre_prueba, salida, salidaEsperada);
565
566          EXCEPTION
567          WHEN OTHERS THEN
568              PRINTR(nombre_prueba, false, salidaEsperada);
569              ROLLBACK;
570      END insertar;
571
572      PROCEDURE actualizar
573      (nombre_prueba VARCHAR2, a_id_factura INT, a_devuelto VARCHAR2,
574       salidaEsperada BOOLEAN) AS
575      salida BOOLEAN := true;
576      actual factura%ROWTYPE;
577      BEGIN
578          UPDATE factura SET DEVUELTO = a_devuelto where ID_factura = a_id_factura;
579
580          SELECT * INTO actual FROM factura WHERE ID_factura = a_id_factura;
581          IF (actual.devuelto<>a_devuelto) THEN
582              salida := false;
583          END IF;
584          PRINTR(nombre_prueba, salida, salidaEsperada);
585
586          EXCEPTION
587          WHEN OTHERS THEN
588              PRINTR(nombre_prueba, false, salidaEsperada);
589              ROLLBACK;
590      END actualizar;
591
592      PROCEDURE eliminar
593      (nombre_prueba VARCHAR2, e_id_factura INT, salidaEsperada BOOLEAN) AS
594      salida BOOLEAN := true;
595      cantidad INT;
596      BEGIN
597          DELETE FROM factura WHERE ID_factura = e_id_factura;
598
599          SELECT COUNT(*) INTO cantidad FROM factura WHERE ID_factura = e_id_factura;
600          IF (cantidad<>0) THEN
601              salida := false;
602          END IF;
603          PRINTR(nombre_prueba, salida, salidaEsperada);
604
605          EXCEPTION
606          WHEN OTHERS THEN
607              PRINTR(nombre_prueba, false, salidaEsperada);
608              ROLLBACK;
609      END eliminar;
610  END PRUEBAS_factura;
611
612 /
613
614 CREATE OR REPLACE PACKAGE BODY PRUEBA_STOCK AS
615     PROCEDURE inicializar AS
616     BEGIN
617         DELETE FROM stock;
618     END inicializar;
619
620     PROCEDURE insertar
621     (nombre_prueba VARCHAR2, i_id_emplazamiento INT, i_id_producto INT,

```

```

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

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

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

636             PRINTR(nombre_prueba, salida, salidaEsperada);

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

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

652             SELECT * INTO actual FROM stock WHERE ID_Emplazamiento = a_id_emplazamiento and
                ID_Producto = a_id_producto;
                IF (actual.cantidad<>a_cantidad) THEN
654                 salida := false;
                END IF;

656             PRINTR(nombre_prueba, salida, salidaEsperada);

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

664     PROCEDURE eliminar
        (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT,
666         salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
668         cantidad INT;
        BEGIN
670             DELETE FROM stock WHERE ID_Emplazamiento = e_id_emplazamiento and ID_Producto =
                e_id_producto;

672             SELECT COUNT(*) INTO cantidad FROM stock WHERE ID_Emplazamiento = e_id_emplazamiento and
                ID_Producto = e_id_producto;
                IF (cantidad<>0) THEN
674                 salida := false;
                END IF;

676             PRINTR(nombre_prueba, salida, salidaEsperada);

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

684     PROCEDURE stock_correcto
        (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT, stock_previo int, cantidad
686         INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
688         stock_nuevo int;
        BEGIN
690             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
692

```

```

694         salida := false;
        END IF;

696         PRINTR(nombre_prueba, salida, salidaEsperada);

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

702     END stock_correcto;

704 PROCEDURE stock_correcto_p
706     (nombre_prueba VARCHAR2, e_id_emplazamiento INT, e_id_producto INT, stock_previo int, cantidad
    INT, salidaEsperada BOOLEAN) AS
708     salida BOOLEAN := true;
    stock_nuevo int;
    BEGIN
710         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
712             salida := false;
            END IF;

714         PRINTR(nombre_prueba, salida, salidaEsperada);

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

722 PROCEDURE stock_correcto_t
724     (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
726     salida BOOLEAN := true;
    stock_nuevo_envia int;
    stock_nuevo_recibe int;
    BEGIN
730         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;
732         if(stock_previo_envia-cantidad)<>stock_nuevo_envia or (stock_previo_recibe+cantidad)<>
    stock_nuevo_recibe then
            salida := false;
            END IF;
            PRINTR(nombre_prueba, salida, salidaEsperada);

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

742 END PRUEBA_STOCK;

744 /

746 CREATE OR REPLACE PACKAGE BODY PRUEBAS_VENTA AS

748     PROCEDURE inicializar AS
    BEGIN
750         DELETE FROM VENTA;
    END inicializar;

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

762         w_ID_Venta := S_ID_Venta.currval;

```

```

764      SELECT * INTO actual FROM venta WHERE ID_Venta = w_ID_Venta;
766      IF (actual.fechaVenta<>i_fecha) OR (actual.dni<>i_dni) THEN
768          salida := false;
768      END IF;
770      PRINTR(nombre_prueba, salida, salidaEsperada);
772      EXCEPTION
772      WHEN OTHERS THEN
774          PRINTR(nombre_prueba, false, salidaEsperada);
774          ROLLBACK;
776      END insertar;
778      PROCEDURE eliminar
778      (nombre_prueba VARCHAR2, e_id_venta INT, salidaEsperada BOOLEAN) AS
780      salida BOOLEAN := true;
780      cantidad INT;
782      BEGIN
782          DELETE FROM venta WHERE ID_Venta = e_id_venta;
784          SELECT COUNT(*) INTO cantidad FROM venta WHERE ID_Venta = e_id_venta;
786          IF (cantidad<>0) THEN
786              salida := false;
788          END IF;
790          PRINTR(nombre_prueba, salida, salidaEsperada);
792          EXCEPTION
792          WHEN OTHERS THEN
794              PRINTR(nombre_prueba, false, salidaEsperada);
794              ROLLBACK;
796      END eliminar;
798      /*
798      procedure descuento
798      (nombre_prueba VARCHAR2, v_id_venta int,v_preciototal number,v_dni_socio varchar2,
798      salidaEsperada BOOLEAN) as
800      salida BOOLEAN := true;
800      p_preciototal number;
802      begin
802          select preciototal into p_preciototal from venta where id_venta = v_id_venta;
804          if (v_dni_socio<>null) then
804              if(v_preciototal * 0.95 <> p_preciototal) then
806                  salida := false;
806              END IF;
808          END IF;
810          printr(nombre_prueba, salida, salidaEsperada);
812          EXCEPTION
812          WHEN OTHERS THEN
814              PRINTR(nombre_prueba, false, salidaEsperada);
814              ROLLBACK;
816      end descuento;*/
816      END PRUEBAS_VENTA;
818      /
820      CREATE OR REPLACE PACKAGE BODY PRUEBAS_PEDIDO AS
822      PROCEDURE inicializar AS
822      BEGIN
824          DELETE FROM pedido;
824      END inicializar;
826      PROCEDURE insertar
826      (nombre_prueba VARCHAR2, i_fecha DATE,
826      i_id_emplazamiento INT, i_cif VARCHAR2, salidaEsperada BOOLEAN) AS
830      salida BOOLEAN := true;
830      actual_pedido %ROWTYPE;
832      w_ID_pedido INT;
832      BEGIN
834          INSERT INTO pedido VALUES(null, i_fecha, i_id_emplazamiento, i_cif);
836          w_ID_pedido := S_ID_Pedido.currval;
838          SELECT * INTO actual FROM pedido WHERE ID_Pedido = w_ID_pedido;

```

```

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

844     PRINTR(nombre_prueba, salida, salidaEsperada);

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

850
PROCEDURE eliminar
852     (nombre_prueba VARCHAR2, e_id_pedido INT, salidaEsperada BOOLEAN) AS
854     salida BOOLEAN := true;
856     cantidad INT;
BEGIN
858         DELETE FROM pedido WHERE ID_Pedido = e_id_pedido;

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

864     PRINTR(nombre_prueba, salida, salidaEsperada);

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

872 /

874 CREATE OR REPLACE PACKAGE BODY PRUEBAS_TRASPASO AS

876     PROCEDURE inicializar AS
878     BEGIN
880         DELETE FROM traspaso;
END inicializar;

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

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

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

896     PRINTR(nombre_prueba, salida, salidaEsperada);

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

904
PROCEDURE eliminar
906     (nombre_prueba VARCHAR2, e_id_Traspaso INT, salidaEsperada BOOLEAN) AS
908     salida BOOLEAN := true;
910     cantidad INT;
BEGIN
912         DELETE FROM traspaso WHERE ID_traspaso = e_id_traspaso;

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

```

```

914         IF (cantidad<>0) THEN
            salida := false;
        END IF;
916
        PRINTR(nombre_prueba, salida, salidaEsperada);
918
        EXCEPTION
920         WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
922     END eliminar;
924 END PRUEBAS_TRASPASO;
926 /
928 CREATE OR REPLACE PACKAGE BODY PRUEBAS_Solicitud_Traspaso AS
930     PROCEDURE inicializar AS
        BEGIN
932         DELETE FROM Solicitud_Traspaso;
        END inicializar;
934
        PROCEDURE insertar
936         (nombre_prueba VARCHAR2, i_fechaSolicitud DATE, i_id_emplazamientoSalida INT,
            i_id_emplazamientoEntrada INT, salidaEsperada BOOLEAN) AS
938         salida BOOLEAN := true;
        actual Solicitud_Traspaso%ROWTYPE;
        w_ID_Solicitud INT;
        BEGIN
942         INSERT INTO Solicitud_Traspaso VALUES(null, i_fechaSolicitud, i_id_emplazamientoSalida,
            i_id_emplazamientoEntrada);
944
            w_ID_Solicitud := S_ID_Solicitud.currval;
            SELECT * INTO actual FROM Solicitud_Traspaso WHERE ID_Solicitud = w_ID_Solicitud;
946
            IF (actual.fechaSolicitud<>i_fechaSolicitud) OR (actual.id_emplazamientoSalida<>
            i_id_emplazamientoSalida) OR (actual.id_emplazamientoEntrada<>i_id_emplazamientoEntrada) THEN
948                 salida := false;
            END IF;
950
            PRINTR(nombre_prueba, salida, salidaEsperada);
952
            EXCEPTION
954         WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
956     END insertar;
958
        PROCEDURE eliminar
960         (nombre_prueba VARCHAR2, e_id_solicitudTraspaso INT, salidaEsperada BOOLEAN) AS
        salida BOOLEAN := true;
        cantidad INT;
        BEGIN
964         DELETE FROM Solicitud_Traspaso WHERE ID_Solicitud = e_id_solicitudTraspaso;
966
            SELECT COUNT(*) INTO cantidad FROM Solicitud_Traspaso WHERE ID_Solicitud =
            e_id_solicitudTraspaso;
            IF (cantidad<>0) THEN
968                 salida := false;
            END IF;
970
            PRINTR(nombre_prueba, salida, salidaEsperada);
972
            EXCEPTION
974         WHEN OTHERS THEN
            PRINTR(nombre_prueba, false, salidaEsperada);
            ROLLBACK;
976     END eliminar;
978 END PRUEBAS_Solicitud_Traspaso;
980 /
982 CREATE OR REPLACE PACKAGE BODY PRUEBAS_Albaran AS
984     PROCEDURE inicializar AS
        BEGIN
986         DELETE FROM Albaran;

```

```

988      END inicializar;

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

998          w_ID_Albaran := S_ID_Albaran.currval;
999          SELECT * INTO actual FROM Albaran WHERE ID_Albaran = w_ID_Albaran;
1000
1001          IF (actual.fechaFirma<>i_fecha) OR (actual.id_pedido<>i_id_pedido) THEN
1002              salida := false;
1003          END IF;

1004          PRINTR(nombre_prueba, salida, salidaEsperada);

1006          EXCEPTION
1007          WHEN OTHERS THEN
1008              PRINTR(nombre_prueba, false, salidaEsperada);
1009              ROLLBACK;
1010      END insertar;

1012      PROCEDURE eliminar
1013          (nombre_prueba VARCHAR2, e_id_albaran INT, salidaEsperada BOOLEAN) AS
1014          salida BOOLEAN := true;
1015          cantidad INT;
1016      BEGIN
1017          DELETE FROM Albaran WHERE ID_Albaran = e_id_albaran;

1018          SELECT COUNT(*) INTO cantidad FROM Albaran WHERE ID_Albaran = e_id_albaran;
1019          IF (cantidad<>0) THEN
1020              salida := false;
1021          END IF;

1022          PRINTR(nombre_prueba, salida, salidaEsperada);

1024          EXCEPTION
1025          WHEN OTHERS THEN
1026              PRINTR(nombre_prueba, false, salidaEsperada);
1027              ROLLBACK;
1028      END eliminar;
1029  END PRUEBAS_Albaran;

1030  /

1031  CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_PRODUCTO_TRASPASO AS

1032      PROCEDURE inicializar AS
1033      BEGIN
1034          DELETE FROM asociacion_Producto_traspaso;
1035      END inicializar;

1036      PROCEDURE insertar
1037          (nombre_prueba VARCHAR2, i_id_producto INT, i_id_Traspaso INT,
1038           i_cantidad NUMBER, salidaEsperada BOOLEAN) AS
1039          salida BOOLEAN := true;
1040          actual asociacion_Producto_traspaso%ROWTYPE;
1041      BEGIN
1042          INSERT INTO asociacion_Producto_traspaso VALUES(i_id_Traspaso, i_id_producto, i_cantidad);

1043          SELECT * INTO actual FROM asociacion_Producto_traspaso WHERE ID_Producto = i_id_producto and
1044          ID_Traspaso = i_id_Traspaso;

1045          IF (actual.id_producto<>i_id_producto) OR (actual.id_Traspaso<>i_id_Traspaso) OR (actual.
1046          cantidad<>i_cantidad) THEN
1047              salida := false;
1048          END IF;

1049          PRINTR(nombre_prueba, salida, salidaEsperada);

1050          EXCEPTION
1051          WHEN OTHERS THEN
1052              PRINTR(nombre_prueba, false, salidaEsperada);

```



```

1062         ROLLBACK;
1063     END insertar;
1064
1065     PROCEDURE eliminar
1066         (nombre_prueba VARCHAR2, e_id_Traspaso INT, e_id_producto INT, salidaEsperada BOOLEAN) AS
1067         salida BOOLEAN := true;
1068         cantidad INT;
1069     BEGIN
1070         DELETE FROM asociacion_Producto_traspaso WHERE ID_Producto = e_id_producto and
1071             ID_Traspaso = e_id_Traspaso;
1072
1073         SELECT COUNT(*) INTO cantidad FROM asociacion_Producto_traspaso WHERE ID_Producto =
1074             e_id_producto and ID_Traspaso = e_id_Traspaso;
1075         IF (cantidad <> 0) THEN
1076             salida := false;
1077         END IF;
1078
1079         PRINTR(nombre_prueba, salida, salidaEsperada);
1080
1081         EXCEPTION
1082         WHEN OTHERS THEN
1083             PRINTR(nombre_prueba, false, salidaEsperada);
1084             ROLLBACK;
1085     END eliminar;
1086 END PRUEBAS_A_PRODUCTO_TRASPASO;
1087
1088 /
1089
1090 CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_PRODUCTO_SOLICITUD AS
1091
1092     PROCEDURE inicializar AS
1093     BEGIN
1094         DELETE FROM asociacion_Producto_Solicitud;
1095     END inicializar;
1096
1097     PROCEDURE insertar
1098         (nombre_prueba VARCHAR2, i_id_producto INT, i_id_Solicitud INT,
1099          i_cantidad NUMBER, salidaEsperada BOOLEAN) AS
1100     salida BOOLEAN := true;
1101     actual asociacion_Producto_Solicitud%ROWTYPE;
1102     BEGIN
1103         INSERT INTO asociacion_Producto_Solicitud VALUES(i_id_Solicitud, i_id_producto, i_cantidad);
1104
1105         SELECT * INTO actual FROM asociacion_Producto_Solicitud WHERE ID_Producto = i_id_producto and
1106             ID_Solicitud = i_id_Solicitud;
1107
1108         IF (actual.id_producto <> i_id_producto) OR (actual.id_Solicitud <> i_id_Solicitud) OR (actual.
1109             cantidad <> i_cantidad) THEN
1110             salida := false;
1111         END IF;
1112
1113         PRINTR(nombre_prueba, salida, salidaEsperada);
1114
1115         EXCEPTION
1116         WHEN OTHERS THEN
1117             PRINTR(nombre_prueba, false, salidaEsperada);
1118             ROLLBACK;
1119     END insertar;
1120
1121     PROCEDURE eliminar
1122         (nombre_prueba VARCHAR2, e_id_Solicitud INT, e_id_producto INT, salidaEsperada BOOLEAN) AS
1123         salida BOOLEAN := true;
1124         cantidad INT;
1125     BEGIN
1126         DELETE FROM asociacion_Producto_Solicitud WHERE ID_Producto = e_id_producto and
1127             ID_Solicitud = e_id_Solicitud;
1128
1129         SELECT COUNT(*) INTO cantidad FROM asociacion_Producto_Solicitud WHERE ID_Producto =
1130             e_id_producto and ID_Solicitud = e_id_Solicitud;
1131         IF (cantidad <> 0) THEN
1132             salida := false;
1133         END IF;
1134
1135         PRINTR(nombre_prueba, salida, salidaEsperada);
1136
1137         EXCEPTION
1138         WHEN OTHERS THEN

```

```

1134         PRINTR(nombre_prueba, false, salidaEsperada);
1135         ROLLBACK;
1136     END eliminar;
1137 END PRUEBAS_A_PRODUCTO_SOLICITUD;
1138 /
1139
1140 CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_PEDIDO_PRODUCTO AS
1141
1142     PROCEDURE inicializar AS
1143     BEGIN
1144         DELETE FROM asociacion_Pedido_Producto;
1145     END inicializar;
1146
1147     /*
1148     PROCEDURE insertar
1149     (nombre_prueba VARCHAR2, i_id_pedido INT, i_id_producto INT,
1150      i_cantidad NUMBER, i_precioCompra number, i_iva INT,
1151      salidaEsperada BOOLEAN) AS
1152     salida BOOLEAN := true;
1153     actual asociacion_Pedido_Producto%ROWTYPE;
1154     BEGIN
1155         INSERT INTO asociacion_Pedido_Producto VALUES(i_id_pedido, i_id_producto, i_cantidad,
1156         i_precioCompra, i_iva,);
1157
1158         SELECT * INTO actual FROM asociacion_Pedido_Producto WHERE ID_pedido = i_id_pedido and
1159         ID_producto = i_id_producto;
1160
1161         IF (actual.id_pedido<>i_id_pedido) OR (actual.id_producto<>i_id_producto) OR (actual.cantidad
1162         <>i_cantidad) OR (actual.precioCompra<>i_precioCompra) OR (actual.iva<>i_iva) THEN
1163             salida := false;
1164         END IF;
1165
1166         PRINTR(nombre_prueba, salida, salidaEsperada);
1167
1168         EXCEPTION
1169         WHEN OTHERS THEN
1170             PRINTR(nombre_prueba, false, salidaEsperada);
1171             ROLLBACK;
1172     END insertar;
1173     */
1174
1175     PROCEDURE eliminar
1176     (nombre_prueba VARCHAR2, e_id_pedido INT, e_id_producto INT,
1177      salidaEsperada BOOLEAN) AS
1178     salida BOOLEAN := true;
1179     cantidad INT;
1180     BEGIN
1181         DELETE FROM asociacion_Pedido_Producto WHERE ID_pedido = e_id_pedido and ID_producto =
1182         e_id_producto;
1183
1184         SELECT COUNT(*) INTO cantidad FROM asociacion_Pedido_Producto WHERE ID_pedido =
1185         e_id_pedido and ID_producto = e_id_producto;
1186         IF (cantidad<>0) THEN
1187             salida := false;
1188         END IF;
1189
1190         PRINTR(nombre_prueba, salida, salidaEsperada);
1191
1192         EXCEPTION
1193         WHEN OTHERS THEN
1194             PRINTR(nombre_prueba, false, salidaEsperada);
1195             ROLLBACK;
1196     END eliminar;
1197 END PRUEBAS_A_PEDIDO_PRODUCTO;
1198 /
1199
1200 CREATE OR REPLACE PACKAGE BODY PRUEBAS_A_VENTA_PRODUCTO AS
1201
1202     PROCEDURE inicializar AS
1203     BEGIN
1204         DELETE FROM asociacion_Venta_Producto;
1205     END inicializar;
1206
1207     PROCEDURE insertar
1208     (nombre_prueba VARCHAR2, i_id_venta INT, i_id_producto INT,
1209      i_cantidad NUMBER, i_precioVenta NUMBER, i_ivaVenta NUMBER,

```

```

1206         salidaEsperada BOOLEAN) AS
1207     salida BOOLEAN := true;
1208     actual asociacion_Venta_Producto %ROWTYPE;
1209     BEGIN
1210         INSERT INTO asociacion_Venta_Producto VALUES(i_id_venta, i_id_producto, i_cantidad,
1211             i_precioVenta, i_ivaVenta);
1212
1213         SELECT * INTO actual FROM asociacion_Venta_Producto WHERE ID_Venta = i_id_venta and
1214             ID_producto = i_id_producto;
1215
1216         IF (actual.id_venta<>i_id_venta) OR (actual.id_producto<>i_id_producto) OR (actual.cantidad<>
1217             i_cantidad) OR (actual.precioVenta<>i_precioVenta) OR (actual.ivaVenta<>i_ivaVenta) THEN
1218             salida := false;
1219         END IF;
1220
1221         PRINTR(nombre_prueba, salida, salidaEsperada);
1222
1223         EXCEPTION
1224         WHEN OTHERS THEN
1225             PRINTR(nombre_prueba, false, salidaEsperada);
1226             ROLLBACK;
1227     END insertar;
1228
1229     PROCEDURE eliminar
1230     (nombre_prueba VARCHAR2, e_id_venta INT, e_id_producto INT, salidaEsperada BOOLEAN) AS
1231     salida BOOLEAN := true;
1232     cantidad INT;
1233     BEGIN
1234         DELETE FROM asociacion_Venta_Producto WHERE ID_Venta = e_id_venta and ID_producto =
1235             e_id_producto;
1236
1237         SELECT COUNT(*) INTO cantidad FROM asociacion_Venta_Producto WHERE ID_Venta = e_id_venta
1238             and ID_producto = e_id_producto;
1239         IF (cantidad<>0) THEN
1240             salida := false;
1241         END IF;
1242
1243         PRINTR(nombre_prueba, salida, salidaEsperada);
1244
1245         EXCEPTION
1246         WHEN OTHERS THEN
1247             PRINTR(nombre_prueba, false, salidaEsperada);
1248             ROLLBACK;
1249     END eliminar;
1250 END PRUEBAS_A_VENTA_PRODUCTO;
1251
1252 /
1253
1254 /*
1255 CREATE OR REPLACE PACKAGE BODY PRUEBAS_FUNCIONES AS
1256     PROCEDURE ganancias
1257     (nombre_prueba VARCHAR2, mes NUMBER, anyo NUMBER, esperado number, salidaEsperada BOOLEAN) AS
1258     salida BOOLEAN := true;
1259     ganancia number;
1260     BEGIN
1261         SELECT ganancias_mensuales(mes,anyo) INTO ganancia FROM dual;
1262
1263         IF (ganancia<>esperado) THEN
1264             salida := false;
1265         END IF;
1266
1267         PRINTR(nombre_prueba, salida, salidaEsperada);
1268
1269         EXCEPTION
1270         WHEN OTHERS THEN
1271             PRINTR(nombre_prueba, false, salidaEsperada);
1272             ROLLBACK;
1273     END ganancias;
1274
1275     PROCEDURE precioLinea_A_Pedido
1276     (nombre_prueba VARCHAR2, ID_PRODUCTO NUMBER, ID_PEDIDO NUMBER, esperado number, salidaEsperada
1277         BOOLEAN) AS
1278     salida BOOLEAN := true;
1279     precio number;
1280     BEGIN
1281         SELECT precioLinea_Aso_Pedido(ID_PRODUCTO,ID_PEDIDO) INTO precio FROM dual;

```

```

1276         IF (precio<>esperado) THEN
1278             salida := false;
1280         END IF;

1282         PRINTR(nombre_prueba, salida, salidaEsperada);

1284         EXCEPTION
1286         WHEN OTHERS THEN
1288             PRINTR(nombre_prueba, false, salidaEsperada);
1290             ROLLBACK;
1292         END PRECIOLINEA_A_PEDIDO;

1294     PROCEDURE precioLinea_A_Venta
1296     (nombre_prueba VARCHAR2, ID_PRODUCTO NUMBER, ID_VENTA NUMBER, esperado number, salidaEsperada
1298     BOOLEAN) AS
1300     salida BOOLEAN := true;
1302     precio number;
1304     BEGIN
1306         SELECT precioLinea_Aso_VENTA(ID_PRODUCTO, ID_VENTA) INTO precio FROM dual;

1308         IF (precio<>esperado) THEN
1310             salida := false;
1312         END IF;

1314         PRINTR(nombre_prueba, salida, salidaEsperada);

1316         EXCEPTION
1318         WHEN OTHERS THEN
1320             PRINTR(nombre_prueba, false, salidaEsperada);
1322             ROLLBACK;
1324         END PRECIOLINEA_A_VENTA;

1326     END PRUEBAS_FUNCIONES;
1328 /
1330 */
1332 DROP SEQUENCE S_ID_Producto;
1334 DROP SEQUENCE S_ID-Traspaso;
1336 DROP SEQUENCE S_ID_Solicitud;
1338 DROP SEQUENCE S_ID_Pedido;
1340 DROP SEQUENCE S_ID_Albaran;
1342 DROP SEQUENCE S_ID_Factura;
1344 DROP SEQUENCE S_ID_Emplazamiento;
1346 DROP SEQUENCE S_ID_Venta;

1348 CREATE SEQUENCE S_ID_Emplazamiento START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1350 CREATE SEQUENCE S_ID_Venta START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1352 CREATE SEQUENCE S_ID_Pedido START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1354 CREATE SEQUENCE S_ID_Albaran START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1356 CREATE SEQUENCE S_ID_Producto START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1358 CREATE SEQUENCE S_ID-Traspaso START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1360 CREATE SEQUENCE S_ID_Solicitud START WITH 1 INCREMENT BY 1 MAXVALUE 9999;
1362 CREATE SEQUENCE S_ID_Factura START WITH 1 INCREMENT BY 1 MAXVALUE 9999;

1364 /* EXECUTE DE PRUEBAS */
1366 -- Inicializacion
1368 EXECUTE pruebas_albaran.inicializar();
1370 EXECUTE PRUEBAS_A_PRODUCTO_TRASPASO.inicializar();
1372 EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.inicializar();
1374 EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.inicializar();
1376 EXECUTE PRUEBAS_A_VENTA_PRODUCTO.inicializar();
1378 EXECUTE PRUEBAS_FACTURA.inicializar();
1380 EXECUTE pruebas_venta.inicializar();
1382 EXECUTE pruebas_pedido.inicializar();
1384 EXECUTE pruebas_proveedor.inicializar();
1386 EXECUTE PRUEBAS_STOCK.inicializar();
1388 EXECUTE pruebas_producto.inicializar();
1390 EXECUTE PRUEBAS_TRASPASO.inicializar();
1392 EXECUTE PRUEBAS_SOLICITUD_TRASPASO.inicializar();
1394 EXECUTE PRUEBAS_SOCIO.inicializar();
1396 EXECUTE pruebas_emplazamiento.inicializar();
1398 --SOCIO
1400 EXECUTE PRUEBAS_SOCIO.inicializar();
1402 EXECUTE PRUEBAS_SOCIO.insertar('Socio Insertar', 'Daniel', 'Gonzalez', '54148787G', 'Avenida del
1404 pepinillo', TO_DATE('10102015', 'DDMMYYYY'), 'dani@us.es', true);
1406 EXECUTE PRUEBAS_SOCIO.insertar('Socio Insertar', 'Jose Luis', 'Marmol Romero', '29613400A', 'Calle
1408 Virgen', TO_DATE('10101996', 'DDMMYYYY'), 'jlmr@us.es', true);

```

```

1350 EXECUTE PRUEBAS_SOCIO.actualizar('Socio Actualizar', '54148787G', 'Avenida del pepinillo234', '
      dani@us.es', true);
EXECUTE PRUEBAS_SOCIO.eliminar('Socio Eliminar', '54148787G', true);
1352
--PROVEEDOR
1354 EXECUTE PRUEBAS_PROVEEDOR.inicializar();
EXECUTE PRUEBAS_PROVEEDOR.insertar('Proveedor Insertar', 'B54120214', 'Pimex', 954852320, 'pimex@pi.
      es', true);
1356 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);
1358 EXECUTE PRUEBAS_PROVEEDOR.eliminar('Proveedor Eliminar', 'B54120214', true);

--PRODUCTO
1360 EXECUTE PRUEBAS_PRODUCTO.inicializar();
1362 COMMIT WORK;
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto con IVA invalido', 'Chaleco de lana', 'por una gran
      disenadora', 'Jerseys', 46.95, 2, false);
1364 EXECUTE PRUEBAS_PRODUCTO.insertar('Producto con categoria invalida', 'Chaleco de lana', 'por una gran
      disenadora', 'Ropa', 46.95, 0.21, false);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto con precio invalido', 'Chaleco de lana', 'por una gran
      disenadora', 'Jerseys', -2, 0.21, false);
1366
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion', 'Zapatos cutres', 'unos zaparos cutres pero
      calentitos', 'Calzado', 12.5, 0.21, true);
1368 EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion', 'Cool shoes', 'Zapatos de cuero italiano', '
      Calzado', 5.95, 0.21, true);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion', 'Sport Shoes', 'Zapatos de plastico', 'Calzado'
      , 6.95, 0.21, true);
1370 EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion', 'Brown Sandals', 'Sandalias de playa marrones',
      'Calzado', 7.95, 0.21, true);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion', 'Chupa de cuero', 'gran imitacion de la chupa
      de Han Solo', 'Calzado', 500, 0.21, true);
1372 EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion', 'Chaleco de lana', 'por una gran disenadora', '
      Jerseys', 46.95, 0.21, true);
EXECUTE PRUEBAS_PRODUCTO.insertar('Producto insercion', 'Sombrero de paja', 'el gran sombrero de
      Mugiwara', 'Accesorios', 20, 0.21, true);
1374 EXECUTE PRUEBAS_PRODUCTO.actualizar('Producto actualizacion', S_ID_Producto.currval-2, 'Unos zapatos
      cutres pero calentitos, estan rotos', 10.5, 0.21, true);
EXECUTE PRUEBAS_PRODUCTO.eliminar('Producto eliminar', S_ID_Producto.currval, true);
1376 --EMPLAZAMIENTO
EXECUTE PRUEBAS_EMPLAZAMIENTO.inicializar();
1378 COMMIT WORK;
EXECUTE PRUEBAS_EMPLAZAMIENTO.insertar('Emplazamiento con tipo invalido', 'Calle de la piruleta 25',
      958632666, 'Emplazamiento', false);
1380
EXECUTE PRUEBAS_EMPLAZAMIENTO.insertar('Emplazamiento insercion', 'Calle de la piruleta', 954147741, '
      TIENDA', true);
1382 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);
1384 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);
1386 EXECUTE PRUEBAS_EMPLAZAMIENTO.eliminar('Emplazamiento eliminar', S_ID_Emplazamiento.currval, true);
--STOCK
1388 EXECUTE PRUEBA_STOCK.inicializar();
EXECUTE PRUEBA_STOCK.insertar('Stock insercion', S_ID_Emplazamiento.currval-2, S_ID_Producto.currval
      -1, 10, true);
1390 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);
1392 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);
1394 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);
1396 EXECUTE PRUEBA_STOCK.eliminar('Stock eliminar', S_ID_Emplazamiento.currval-2, S_ID_Producto.currval-3,
      true);

```

```

1398 --VENTA
EXECUTE pruebas_venta.inicializar();
1400 EXECUTE pruebas_venta.insertar('Venta insercion',sysdate,'29613400A',true);
EXECUTE pruebas_venta.insertar('Venta insercion',sysdate,null,true);
1402 EXECUTE pruebas_venta.insertar('Venta insercion',sysdate,null,true);
EXECUTE pruebas_venta.eliminar('Venta eliminar',S_ID_venta.currval,true);
1404 /*
EXECUTE pruebas_venta.insertar('Venta insercion',0,sysdate,null,true);
1406 EXECUTE PRUEBAS_A_VENTA_PRODUCTO.insertar('Venta-Producto insercion',s_id_venta.currval,s_id_producto
    .currval-1,28,10.5,0.21,0,true);
INSERT INTO FACTURA VALUES(0,0,sysdate,'F',4,3);
1408 */
--EXECUTE Pruebas_factura.insertar('Factura insercion',0,sysdate,'F',s_id_venta.currval,
    S_ID_Emplazamiento.currval-1,true);
1410 --select S_ID_Emplazamiento.currval-1 from dual;
--ASOCIACION_VENTA_PRODUCTO
1412 EXECUTE PRUEBAS_A_VENTA_PRODUCTO.inicializar();
EXECUTE PRUEBAS_A_VENTA_PRODUCTO.insertar('Venta-Producto insercion',s_id_venta.currval-1,
    s_id_producto.currval-2,3,10.5,0.21,true);
1414 EXECUTE Pruebas_A_venta_producto.insertar('Venta-Producto insercion',s_id_venta.currval-1,
    s_id_producto.currval-1,2,46.95,0.21,true);
EXECUTE PRUEBAS_A_VENTA_PRODUCTO.insertar('Venta-Producto insercion',s_id_venta.currval-2,
    s_id_producto.currval-2,3,10.5,0.21,true);
1416 EXECUTE Pruebas_A_venta_producto.insertar('Venta-Producto insercion',s_id_venta.currval-2,
    s_id_producto.currval-1,2,46.95,0.21,true);
EXECUTE PRUEBAS_A_VENTA_PRODUCTO.eliminar('Venta-Producto eliminar',s_id_venta.currval-2,
    s_id_producto.currval-1,true);
1418 --FACTURA
EXECUTE PRUEBAS_factura.inicializar();
EXECUTE Pruebas_factura.insertar('Factura insercion',sysdate,'F',s_id_venta.currval-2,
    S_ID_Emplazamiento.currval-1,true);
1422 EXECUTE Pruebas_factura.insertar('Factura insercion',sysdate,'F',s_id_venta.currval-1,
    S_ID_Emplazamiento.currval-2,true);
EXECUTE Pruebas_factura.actualizar('Factura actualizar',s_id_factura.currval,'T',true);
1424 --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);
1426 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);
1428 --PEDIDO
EXECUTE PRUEBAS_PEDIDO.inicializar();
EXECUTE PRUEBAS_PEDIDO.insertar('Pedido insercion',sysdate,S_ID_Emplazamiento.currval-1,'B54129999',
    true);
1432 EXECUTE PRUEBAS_PEDIDO.insertar('Pedido insercion',sysdate,S_ID_Emplazamiento.currval-2,'B54129999',
    true);
EXECUTE PRUEBAS_PEDIDO.insertar('Pedido insercion',sysdate,S_ID_Emplazamiento.currval-2,'B54129999',
    true);
1434 EXECUTE PRUEBAS_PEDIDO.eliminar('Pedido eliminar',S_ID_Pedido.currval,true);
1436 --PEDIDO_PRODUCTO
EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.inicializar();
EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.insertar('PEDIDO_PRODUCTO insercion',S_ID_Pedido.currval-1,
    s_id_producto.currval-1,21,10.5,0.21,true);
1440 EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.insertar('PEDIDO_PRODUCTO insercion',S_ID_Pedido.currval-2,
    s_id_producto.currval-1,22,46.95,0.21,true);
EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.insertar('PEDIDO_PRODUCTO insercion',S_ID_Pedido.currval-1,
    s_id_producto.currval-2,22,46.95,0.21,true);
1442 EXECUTE PRUEBAS_A_PEDIDO_PRODUCTO.eliminar('PEDIDO_PRODUCTO eliminar',S_ID_Pedido.currval-1,
    s_id_producto.currval-2,true);
--ALBARAN
1444 EXECUTE PRUEBAS_ALBARAN.inicializar();
EXECUTE PRUEBAS_ALBARAN.insertar('Albaran insercion',sysdate,S_ID_Pedido.currval-1,true);
EXECUTE PRUEBAS_STOCK.stock_correcto_p('Actualizacion de stock tras comprar',s_id_emplazamiento.
    currval-2,s_id_producto.currval-1,8,21,true);
1448 EXECUTE PRUEBAS_STOCK.stock_correcto_p('Actualizacion de stock tras comprar',s_id_emplazamiento.
    currval-2,s_id_producto.currval-1,8,15,false);
EXECUTE PRUEBAS_ALBARAN.insertar('Albaran insercion',sysdate,S_ID_Pedido.currval-2,true);
1450 EXECUTE PRUEBAS_ALBARAN.eliminar('Albaran eliminacion',S_ID_Albaran.currval,true);
1452 --SOLICITUD_TRASPASO
EXECUTE PRUEBAS_SOLICITUD_TRASPASO.inicializar();

```

```

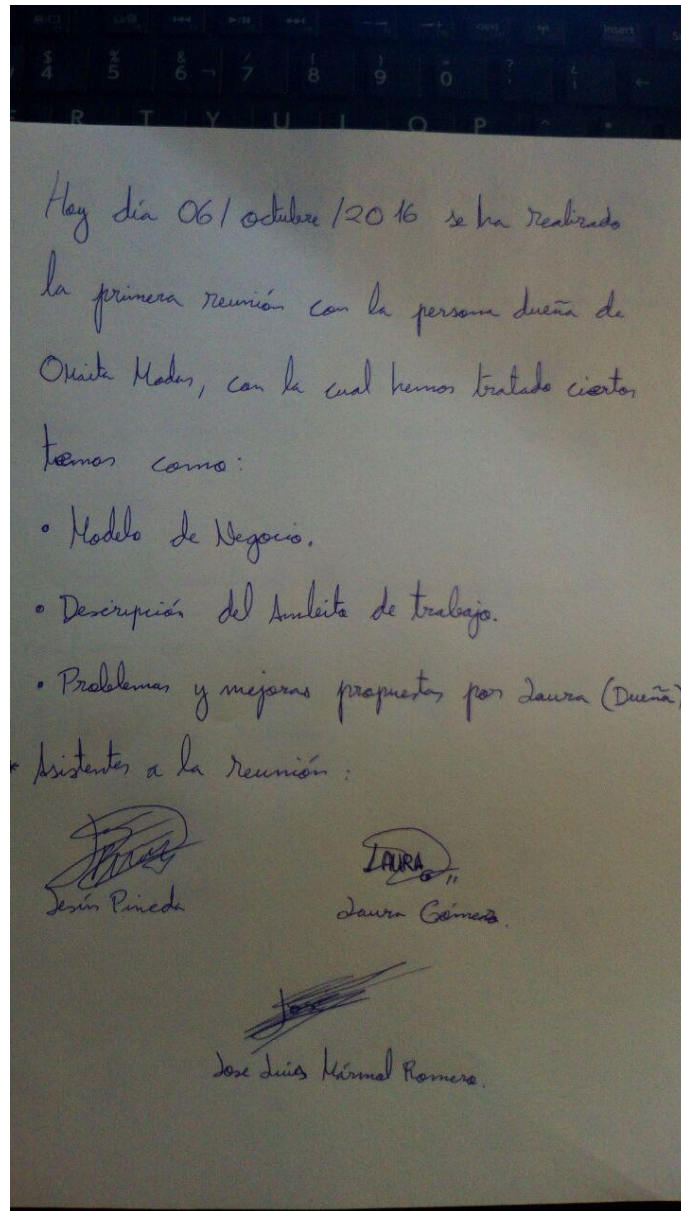
1454 EXECUTE PRUEBAS_SOLICITUD_TRASPASO.insertar('Solicitud traspaso Insertar', sysdate,
      S_ID_Emplazamiento.currval-1, S_ID_Emplazamiento.currval-2, true);
EXECUTE PRUEBAS_SOLICITUD_TRASPASO.eliminar('Solicitud traspaso Eliminar', S_ID_Solicitud.currval,
      true);
1456 EXECUTE PRUEBAS_SOLICITUD_TRASPASO.insertar('Solicitud traspaso Insertar', sysdate,
      S_ID_Emplazamiento.currval-1, S_ID_Emplazamiento.currval-2, true);
--ASOC_PRODUCTO_SOLICITUD
1458 EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.inicializar();
EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.insertar('A-Producto-Solicitud Insertar', S_ID_Producto.currval
      -2, S_ID_Solicitud.currval, 3, true);
1460 EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.eliminar('A-Producto-Solicitud Eliminar', S_ID_Solicitud.currval
      , S_ID_Producto.currval-2, true);
EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.insertar('A-Producto-Solicitud Insertar', S_ID_Producto.currval
      -2, S_ID_Solicitud.currval, 3, true);
1462 COMMIT WORK;
EXECUTE PRUEBAS_A_PRODUCTO_SOLICITUD.insertar('Prueba de Trigger solicitud_stock_minimo',
      S_ID_Producto.currval-1, S_ID_Solicitud.currval, 28, false);
1464
--TRASPASO
1466 EXECUTE PRUEBAS_TRASPASO.inicializar();
EXECUTE PRUEBAS_TRASPASO.insertar('Traspaso insercion',sysdate,S_ID_Emplazamiento.currval-1,
      S_ID_Emplazamiento.currval-2,true);
1468 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);
1470
--ASOC_PRODUCTO_TRASPASO
1472 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);
1474 EXECUTE PRUEBAS_STOCK.STOCK_CORRECTO_T('Actualizacion de stock tras traspaso',s_ID_emplazamiento.
      currval-1, s_ID_emplazamiento.currval-2,33,29,4,s_ID_producto.currval-1,true);
EXECUTE PRUEBAS_A_PRODUCTO_TRASPASO.eliminar('A-Producto-Traspaso Eliminar', S_ID_traspaso.currval-1,
      S_ID_Producto.currval-1, true);
1476 --Pruebas de funciones
/*
1478 EXECUTE PRUEBAS_FUNCIONES.ganancias('Funcion Ganancias mensuales',6,2017,-1008.87,true);
EXECUTE PRUEBAS_FUNCIONES.precioLinea_A_Pedido('Funcion precio linea aso-pedido',9,2,220.5,true);
1480 EXECUTE PRUEBAS_FUNCIONES.precioLinea_A_Venta('Funcion precio linea aso-venta',9,2,93.9,true);
*/
1482 --Tiene que dar -1008.87
--select ganancias_mensuales(6,2017) from dual;
1484 -- Tiene que dar 220.5
--SELECT precioLinea_Aso_Pedido(9,2) FROM DUAL;
1486 --Tienda que dar 93.9
--SELECT precioLinea_Aso_Venta(9,2) FROM DUAL;
1488 -- Select que muestra los productos ofrecidos y disponibles
--SELECT nombre,id_emplazamiento from producto inner join stock on stock.ID_PRODUCTO = producto.
      id_producto;

```

sql/pruebas.sql

## 11. Apéndice

### 11.1. Actas de reunión





Hoy Día 20/ Octubre /2016 se ha realizado la segunda reunión con Laura (Dona de Ornato Rodas) 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 Gómez